Laser-Scan Ltd.

DTMCREATE

Reference Manual

Volume I

Issue 1.8 - 12-October-1992

Science Park, Milton Road, Cambridge, England CB4 4FY tel: (01223) 420414

Document "DTMCREATE REFERENCE", Category "REFERENCE"
Document Issue 1.8 Steve Townrow (modified 12-Oct-1992)

Laser-Scan Ltd.

DTMCREATE

Reference Manual

Volume II

Issue 1.8 - 12-October-1992

Science Park, Milton Road, Cambridge, England CB4 4FY tel: (01223) 420414

Document "DTMCREATE REFERENCE", Category "REFERENCE"
Document Issue 1.8 Steve Townrow (modified 12-Oct-1992)

CONTENTS

DTMCREATE reference documentation change record

Version 1.0 Tim Hartnall 23-March-1988

First customer issue of DTMCREATE Reference Manual

Version 1.1 Tim Hartnall 15-April-1988

TRIGRID chapter added, various minor tidies.

Version 1.2 Tim Hartnall 03-May-1988

TRIANG and TRIGRID ASSIGN and DEASSIGN commands replace some SELECT and DESELECT commands. SELECT ALL command added. SELECT and DESELECT command functionality now the same as LITES2 commands of same name. Various minor tidies.

Version 1.3 Tim Hartnall 14-Jul-1988

TRIANG now differentiates between open and closed cliffs automatically. ASSIGN and DEASSIGN CLOSED_CLIFF_FC, OPEN_CLIFF_FC, CLOSED_CLIFF_LAYER and OPEN_CLIFF_LAYER replaced by ASSIGN and DEASSIGN CLIFF_FC and CLIFF_LAYER. TRIANG general Description section has been enlarged to include more detailed guidance for digitising clifflines.

Version 1.4 Tim Hartnall 05-Aug-1988

To reflect the new ability of TRIANG to handle cliff features which have changing Z-values along their length, the TRIANG cliff description section has been enlarged. TRIEDIT IFF output descriptions have also been revised.

Version 1.5 Dave Catlow 23-Aug-1988

TRIGRID. Option to create a MIKE or ALVY removed, and option of creating a LSLA type header added. DTED type header changed to TED4 type header. See HEADER_TYPE command.

APPENDIX A Now enlarged and provides new, annotated, examples of DTMCREATE lookup files.

Version 1.6 Tim Hartnall 23-Aug-1988

All modules have revised SHOW commands.

TRIANG, TRIEDIT. New DATUM and ENABLE/DISABLE INVERSE commands.

All modules now attempt to translate logical name LSL$DTMCREATE_WORK for location of .DTA, .NOD and .DER files.

TRIANG. Better description for MAXPOINTS command.

TRIANG and TRIEDIT. Errors in ENABLE/DISABLE TOFEET and ENABLE/DISABLE TOMETRES command descriptions corrected.

TRIANG, TRIGRID and TRIEDIT. New UNITS command added.

Version 1.7 Steve Townrow 30-Jan-1992

All modules have revised message sections to ensure each message capable of being produced by the modules is documented.

Version 1.8 Steve Townrow 12-Oct-1992

All modules have a description of the new logical, LSL$DTMCREATE_RESOLUTION introduced to control the scaling resolution of input coordinates.

PREFACE

Intended audience

This manual is intended for all users of the Laser-Scan DTMCREATE package running under the VAX/VMS operating system.

Structure of this document

This document is composed of 2 major sections.

The first three chapters provide an overview of DTMCREATE and are intended as a quick reference guide to the salient features of the DTMCREATE package.

There then follow the User Reference Guides for the individual modules which comprise DTMCREATE. Each individual module contains the same basic categories of information. These are:

MODULE	- the name of the DTMCREATE module.
REPLACES	- which older Laser-Scan programs it replaces.
FUNCTION	- a synopsis of what the modules does
FORMAT	- a summary of the module command format and command qualifiers. Default qualifier settings are indicated.
PROMPT	- how it prompts the user.
PARAMETERS	- description of expected command parameters.
COMMAND QUALIFIERS	- description of all command qualifiers. Qualifiers are ordered alphabetically and default argument values are indicated.
DESCRIPTION	- the definitive description of the module action.
EXAMPLES	- annotated examples of module useage.
MESSAGES	- all classes of message are listed and described and suggested user action given. The messages are divided into sections according to message severity within which the messages are ordered alphabetically by message mnemonic.

Where applicable, additional categories are available for some modules. Some modules, for example, have a "RESTRICTIONS" category.

Associated documents

For summary information about DTI (Digital Terrain Image) files see the MATRIX Reference Manual

---

Conventions used in this document

Convention	Meaning
<CR>	The user should press the carriage return key on the terminal
<CTRL/x>	The phrase <CTRL/x> indicates that the user must press the key labelled CTRL while simultaneously pressing another key, for example, <CTRL/Z>.
$ **GO<CR>**	Command examples show all user entered commands in **bold** type.
25 columns complete . . .	Vertical series of periods, or ellipsis, mean either that not all the data that DTMCREATE displays in response to the particular command is shown or that not all the data that the user would enter is shown.
file-spec...	Horizontal elipsis indicate that additional parameters, values or information can be entered.
[logical-name]	Square brackets indicate that the enclosed item is optional. (Square brackets are not, however, optional in the syntax of a directory name in a file-specification, or in the syntax of a substring specification in a VMS assignment statement).
'integer'	An integer number is expected in the specified input or output field.
'real'	A real number is expected in the specified input or output field.

Convention	Meaning
FSN 'integer' ('integer')	FSN followed by two integer arguments indicates an IFF feature serial number. The integer number enclosed in round brackets is the feature internal sequence number.
00003DE7	A hexadecimal address of a location within an IFF file. DTMCREATE modules express all IFF addresses using hexadecimal radix. The address is always padded with leading zeros to a standard field width of 8 characters.

Command line interpretation

DTMCREATE utilities use the LSLLIB Command Interpreter (CMDLIB) to get and parse the program command lines.

DTMCREATE command line decoding operates in decimal radix.

CHAPTER 1

INTRODUCTION

INTRODUCTION

The Laser-Scan DTM software packages provide tools for the production, validation, manipulation and viewing of matrix data, particularly of elevation (height) data. The DTMCREATE package consists of four interlocking software modules for the generation of DTM data from Laser-Scan IFF (Internal Feature Format) vector data.

In most cases the output matrix, or DTM, will be derived from contour, 3D river, 3D ridgeline and spot height information. Other forms of input data can be handled. For instance, 2D seismic line information, random borehole or oil well data. Laser-Scan DTMCREATE software represents an attempt to provide a single mapping system that copes well with all these types.

It is however optimised and extensively used for grid terrain model generation.

Incoming string data usually consists of digitised x,y data locations, each with an associated z or height value. An alternative for isarithmic data are strings of x,y locations with a single implied surface value. Single point (spot height) data can also be used.

Data entering the system may contain information about surface characteristics other then height, as well as defining the location of particular values. The usual case is either implied or specified detail concerning the local slope at the data point. This could be of various types:

1. smooth continuous variation in slope around a point,

2. slope discontinuities at the point (ridge lines, river courses, sea lake or embankment edges, non-vertical geological faults), or,

3. discontinuities in the thematic variable (cliffs, vertical faults, boundaries).

All three types are still represented by location and value, but in addition every string must also possess a type identifier corresponding to the classes above.

In addition to data type, individual data points may also be flagged to represent the following geomorphological attributes:

- Point is part of a river
- Point is part of a ridgeline
- Point has no particular geomorphological significance.

The Laser-Scan DTM creation software uses such data coded with height, type and geomorphological flag attributes to produce a TIN (Triangular Irregular Network). An interactive graphical editor is provided to enable the user to alter and refine the triangulation.

The triangulation is then used to generate a rectangular grid spaced at some geographical interval over the area concerned. It is not possible in such a grid - unless it is very fine - to ensure that the data values themselves are represented at the grid nodes, but the general convenience of using the very simple structure of direct row storage makes grid manipulation and automated comparison of one grid with another quite straightforward.

The blend of triangulation and grid based methods provided by the Laser-Scan DTM software offers economy of execution speed and convenience of use. Once calculated and stored, further editing and revision of a triangular digital terrain model (DTM) can occur at any time. It is a simple process to output a grid DTM based on the triangular structure at the resolution required by the user. The same triangular DTM can then be used, without further calculation, to generate as many rectangular grids as required from all or part of the data area.

This manual attempts to explain the philosophy of the DTMCREATE triangulation approach to DTM production, together with a module by module reference guide to the package.

CHAPTER 2

MODULE PACKAGE_OVERVIEW

Laser-Scan DTM software - an overview

General

The Laser-Scan DTM software packages consist of discrete software modules which may be used to form a DTM production flowline. Salient modules are:

TRIANG - data structure generation

Contour strings are assumed to have no connection with each other. If they do, and the spatial relationships between them and any other points are known, some other data structure creation module would be more sensibly employed. TRIANG determines the Thiessen (Brassel, 1979) neighbours for each point in the complete data set. From the neighbours surrounding the point a system of triangles can be built (McCullagh, 1980) which uniquely segment the map area - the Delaunay (1934) triangulation (dual of the Thiessen polygon - also known as Dirichlet tesselation, Vorodnoi network, etc.)

The area of coverage for data structuring is specified by the user. As the triangulation is unique it is not necessary to map the whole area at once, but it is perfectly possible to join areas seamlessly by allowing a minimum of one triangle of overlap. This is a very powerful feature of the system.

All data are converted to integer on input to enhance execution speed. The basic system operates at a resolution of 150000 units in the longest map axis direction; equivalent to a positional accuracy of 0.015 mm for a map 1 metre long. Choice of coverage is therefore not crucial, but if super high resolution is required the map can be divided into sections and each section structured separately, gridded using TRIGRID and then assembled using DTITILE.

TRIANG supports the following data types:

1. breaklines - (X, Y,constant Z) and (X, Y,continuous Z)

2. clifflines - (X, Y,constant Z, constant Z) only

3. normal strings, eg contours - (X, Y, constant Z)

4. spot heights - (X, Y, Z)

In addition TRIANG supports the following node flags:

1. ordinary string/data node

2. river string/data node

3. ridge-line string/data node

The neighbour search procedure does not rely on contour integrity - gaps are allowed, as are spot heights and discontinuities of slope. Thus one can use partially digitised maps where the digitising has attempted to maintain a given density of information rather than represent every contour. The effect of variable contour interval is shown in Figure 2.1. The information content is much more evenly spaced in the bottom section of the Figure than in the top, and would result in a better (or at least more even) surface representation. This is an acceptable form of data input, and can speed contour acquisition using manual methods in areas of varying relief. Contour strings can be supplemented by data strings for cliffs, ridge, river and valley lines to provide slope discontinuities, always bearing in mind that all locations will have to be tagged with height values.

TRIANG offers a triangulation constraint option which forces the Delaunay triangulation to honour the connectivity of the original data strings, thus preventing triangles from "leaking" through contours and other strings, see Figure 2.2.

If edge matching to adjacent DTMs is required, TRIANG provides the option to include relevant post values from the adjacent DTI format DTM as edge control to the current triangulation.

Output from TRIANG is in the form of binary node and Thiessen neighbour files. These are used directly by TRIEDIT (editor) and TRIGRID (grid creation).

TRIEDIT - graphical structure display and editor

It is possible that when a data structure has been created it contains defects caused by insufficient or incorrect data in certain areas of the map leading to the incorrect definition of particular characteristics of the shape. Examples of this type of flaw could be:

1. inaccurate location of nodes,

2. inaccurate node flagging,

3. inaccurate node height values.

TRIEDIT may be used to insert extra nodes or strings of nodes into the existing network in such a manner that links along the strings will remain connected in the network irrespective of string point spacing. This is particularly important for highway engineering examples or other boundary limit situations where the spacing between the points may well lead to severe "breaking" of discontinuity lines if included in the original data structuring.

The process of data insertion in TRIEDIT assumes permanent connections between points in an input string. The resulting triangulation is a distortion of the Delaunay criterion, and hence any retriangulation of the data set including these unbreakable strings could lead to problems. The triangular structure modified in this manner can therefore only be used in TRIGRID and later phases.

TRIDER - slope derivative estimation

TRIDER takes the binary triangulation node and neighbour files created by TRIANG (or edited output from TRIEDIT) and produces an output file containing slope derivatives at each data point in the triangulation. These data may then be used by TRIGRID in conjunction with the node and data files as the basis for DTM grid estimation. Once TRIDER has been used to generate a slope derivative file many subsequent runs of TRIGRID may be made to produce DTM grids at differing resolutions.

TRIDER is also used to provide Z values for the imaginary points generated around the edge of the triangulation by TRIANG. Four interpolation options are available for imaginary point estimation, the default is based on a trend surface fit through data points for which Z values are known. TRIDER must be re-run to produce a new derivative file every time the triangulation node and data files are modified.

TRIGRID - grid generation

Grids (DTMs) of virtually unlimited size can be created from the spatial structure created by TRIANG (or TRIEDIT) and TRIDER using TRIGRID. The program automatically chops the DTM into sections that hold as many rows of the DTM as possible for the amount of memory available. An example of the grid generation process is shown in Figure 2.3. In this example, three passes through the neighbour network are needed to complete the grid. The program will often contain the entire DTM at one pass. For larger DTMs the program will automatically make the appropriate number of passes through the data to ensure that the grid is complete.

The Thiessen node-neighbour file created by TRIANG supplies the structure from which the triangles are generated. These triangles are then mapped into the DTM (Figure 2.4) so that all grid nodes that fall within triangles are assigned values based on the values, and possibly the estimated derivatives, at the vertices of the triangles. Triangles can be dumped quickly into the DTM without need for point in polygon checks.

TRIGRID uses two methods of interpolating the height value for a grid node position within a given triangle area:

1. Interpolation based on a linear triangle facet. Although the interior triangle surface will be smooth, none of the derivatives along the triangle edges or at the vertices will be continuous.

2. Interpolation using a smooth surface patch fitted to the triangle vertices and estimated derivatives in a manner to ensure both a continuous surface and at least continuous first order derivatives.

The linear facet approach results in a perfectly acceptable surface where the data density is high in relation to the grid being generated, or the surface gradient is low. In such cases the flat facet of any triangle will probably supply only one or two nodes of the final DTM, and will have a slope in close agreement with the actual surface.

Smooth surface interpolation requires considerable computation owing to the irregular triangle shapes but will be worthwhile where the surface is considered "smooth" - for instance where the DTM is intended to represent not terrain but some thematic variable such as gravity data isarithms. The requirements for many DTMs are, however, rather different. In the case of terrain the existence of varying morphology within the DTM area - plains, rolling hills, mountains, cliffs, lakes and sea shores - may make the use of a smooth curved patch function essential. The patch must be continuous but not necessarily all directions. For instance where a ridge or river line indicates a discontinuity that discontinuity must be reflected in the patch covering the triangle area. Other limitations may also apply. For instance where contour data has been used, and in all possible contours of a fixed height interval are present, the interpolation procedure must not give a value that is above the next contour height, or below the last one.

When using smooth patch interpolation an option is provided to apply automatic interpolation limits which take into account triangle facet slope and the feature codes of the triangle vertex points.

TRIGRID offers an option to trace along the original data strings upon which the triangulation is based. Up-hill/down-hill side of line information is collected for each string. This may be used to supplement or override the slope derivatives estimated for triangles which have all three vertices at the same height on the same string, eg a contour.

Estimation of the derivatives at each data point is an essential precursor to any smooth patch calculation. This process employs a separate pass through the Thiessen node-neighbour file to enable local distance weighted estimates of the partial derivatives to be calculated.

Figure 2.5 shows both the standard situation with continuity guaranteed across the triangle and between triangles, and also the desired situation where a river is present along one side of the triangle. The same situation can occur on one, two or all three sides of the triangle in cases such as lakes and coastal margins. For most geologic mapping from randomly located data the situation on the left is the preferred result. For topographic or geologically faulted surfaces however, the example on the right may be the required surface form. This causes problems!

Once the grid section (or complete grid) has been calculated the rows covered are written out to file.

TRIGRID outputs the Laser-Scan internal DTI (Digital Terrain Image) format DTM files which are the common file medium for the TVES (Terrain Visualisation and Exploitation Software) package.

Format converters are available to produce alternative DTM formats. Examples of such format conversions are UHL1 or TED4 type Defense Mapping Agency file format, or simple ASCII text listings.

CHAPTER 3

MODULE TRIANGULATION

Laser-Scan DTM software - description of triangulation

Why use triangles?

The objections to grid based methods are:

1. considerable computer time to interpolate a large regular grid to represent relatively few data points,

2. lack of flexibility in responding to variable data densities in different parts of a map,

3. non-honouring of data points caused by insufficiently fine a grid in order to keep computer time down to reasonable levels, and

4. difficulty in representing fault information adequately on a continuous surface.

These have led to the intensive development in recent years of alternative methods of generating contour maps.

The two most widely known are methods based on triangulation of the data set, and the process of contour following without either grid or triangulation as a guide.

The human cartographer, given a scattered data set to contour, will visualize a set of triangles in the area in which he is working that helps him locate the contour he is tracing. These triangles have no existence but provide a structure for him to use to estimate position of the contour line relative to other data points. It would seem likely that an automated approach that did the same would have considerable benefits, particularly as it would always honour all data points - as the data points would form the vertices of the triangulation. This approach is summarised in Figure 3.1 where the data points connected by a triangulation can then be contoured using linear approximations across the planar facets of the triangles that have been created. There used to be little interest in the automatic triangulation of seismic and other data sets because:

1. it appeared impossible to generate the same triangulation, and hence the same map, from the same set of data, independent of the starting point of the triangulation process, and

FIGURE 3.1

2. the time taken for automatic triangulation was exorbitant. Although the time taken for gridding is related to the square of the number of data points, some of the methods of automatic triangulation were at best related to the cube!

Improvements in the last ten years have now produced reliable triangulation procedures that produce the 'most equilateral' (and therefore unique) set of triangles possible - the Delaunay triangulation - in a time linearly related to the number of data points, and without the need for large computer memory requirements. Some of the multitude of alternative names for the same procedure (or its dual) found in the literature are Thiessen, Vorodonoi, Dirichlet, and Deltri. For any given data set it is now much faster to generate unsmoothed contours from an automatic triangulation procedure than from use of a grid interpolation approach. At the same time all data points are honoured, the resolution of the map varies with the data density, and maps can be joined together without error at the margins.

The problem with the triangulation technique is presently that of smooth contouring. A number of 'patch' functions exist as for grid contouring, but their calculation is more involved because the network of triangles is irregular and no calculation short cuts can be wrung from nice orthogonal axes. Smooth contouring from triangles presently takes two to five times longer than smooth contouring of a regular rectangular grid.

A major reason for the upsurge of interest in triangulation techniques has been that they are ideally suited to fault insertion. If the fault location is entered as a set of data points, the triangulation process will include them and will automatically relate them to the rest of the data set. Then, the triangulation can be 'unzipped' so that there is no direct connection in the data structure between the two halves of the fault. Contouring can then take place and the result will be a perfect edge to the fault depending only on the input resolution of the fault line.

Any triangulation that is to be used for the basis of isarithm map production must have the properties of stability, equilateralness, and non-intersection. It is desirable that the triangulation resulting from any data distribution should be independent of the starting point of the triangulation process inside the distribution. This is particularly important in cases where ambiguity of triangulation might be expected to occur, for instance where four near equidistant data points could be divided into two triangles in either of two ways. From the triangulation view point there would be little difference, but in terms of the contoured surface there could be dramatic changes related not to real variation but purely to the imposed triangulation. Any triangular approach usually attempts to achieve a set of triangles that are as equilateral as possible with minimum line strength. In the past this has often meant iterative processes such as those in SCA (1975) and GTN (1977) that attempted refinement of an initial triangulation. This did not produce a unique solution, and was expensive in computer time.

FIGURE 3.2

FIGURE 3.3

Brassel and Reif (1979) published a paper concerning the sub-division of a two-dimensional area into Thiessen polygons based on the location of a set of random data points. Their method is related to work by Rhynsburger (1973), Shamos and Bentley (1978), Green and Sibson (1978), Gold (1977) and Elfick (1979). The Thiessen polygon is an important concept in geographical thought as it can be used to define the region of influence of any point in a real context. It is however only one form of display of the final solution in that instead of showing the regions surrounding points it is also possible to connect the neighbouring points, or Thiessen neighbours, to produce the dual of Thiessen polygons, the Delauney triangulation. The same calculation procedure that calculates one can be use to generate the other. Figure 3.2 shows the relation between the Delauney triangle connecting a point with its neighbours and the Theissen polygon surrounding a points. In Brassel and Reif's case the polygon network was the most important. In the present instance the triangulation is wanted.

The Delaunay triangulation (Delaunay, 1934) has all the desired properties for use as a base for automatic contouring. The problem of calculating the triangulation is closely akin to that of Thiessen polygon generation but certain modifications can be made which increase the speed of computation, help the algorithm reach linearity, and allow certain calculations to be omitted. The only major problem is that considered by Yoeli (1977) related to the representation of known topographic structure where 'break-lines' may have to be included to maintain ridges and valleys. More recently there has been a great increase of interest in triangulation approaches to mapping, and a number of algorithms and reviews have appeared (Sabin 1980, Peucker 1980, Sibson 1981, Watson 1982).

Algorithmic organisation

The Brassel and Reif algorithm approaches the problem of Thiessen polygon formation by choosing an arbitrary starting point and, as necessity demands, creating a set of imaginary guaranteed neighbours outside the data area to be polygonised. Once a known neighbour has been determined by this arbitrary starting method each neighbour of a given point can be found by rotation about that point in a clockwise direction, at the same time building up an index list of other neighbourhood relationships for use later on.

The algorithm works on a one-dimensionally sorted data list and has to check a considerable number of points that may be the correct next neighbour out of the neighbours surrounding the data points. Figure 3.3 illustrates the principle. If point 0 is the point whose neighbours are to be discovered and 1 is a known neighbour, the line 0-1 is a base from which the next rightmost neighbour can be determined. A new rightmost neighbour has been found when a circumscribing circle passing through the base line 0-1 and the new point, 2, contains within it no other data point. The line 0-2 then becomes the new base-line and point 3 can be found as the next neighbour, and so on around the rotation point, 0, until point 1 is reached again. At this time all the neighbours of point 0 have been defined and a Thiessen polygon for point 0 could be calculated. In

the Brassel and Reif algorithm the formation of the Theissen polygon coincides with the discovery of the neighbours. While the neighbours for the rotation point are being discovered it is possible to update indices of the neighbours for all connected points, greatly reducing calculation effort at later stages. In a random data set about two-thirds of all calculations will have already been made by the time any given data point is investigated. It should be noted that if Delaunay triangulation is required no polygon calculation is necessary as it is simply the triangle set indicated by ABCD etc in Figure 3.3.

Problems

Much of the Brassel and Reif algorithm is related to searching for the next neighbour for a given base line. In Figure 3.3 their method is to calculate the centre and radius of the circumscribing circle for some suitable starting point using points from the sorted list to define the area of search. If a point is found inside the circle a new centre and radius are calculated for the new circumscribed circle and the process iterates until all possible points in the sorted list have been checked. This process has two disadvantages. First, in a direction perpendicular to the sort there is no segregation, so by using a singly sorted list the band of points to be considered within the radius of the present circumscribing circle can be very numerous in a large data set. It would appear therefore that a two-dimensional sort structure should be used to minimise searching time for any given point. Associated with the single direction sort is the fact that all points in the sorted list within the radius limit bounds have to be checked. There is no way of checking the points which are nearer the base line in preference to those which are less likely to be neighbours because they lie further away.

The second problem relates to the amount of calculation involved in determining the new circumscribing circle centre and radius. Although some method must be used to determine point position in relation to a possible circumscribing circle the centre calculation should be avoided for Delaunay triangulation as it necessarily involves considerable floating point calculation and should be used only when a Thiessen vertex needs to be calculated and all neighbours are known.

A major strength of their approach, however, is that it enables them to proceed in a logical spatial manner through the data set, never covering the same ground again. Once neighbours have been determined no more points will be found in that region. This makes for considerable economy in storage.

Delaunay triangulation for mapping

Figure 3.4 shows a map of the location of 50 data points that are to be triangulated and the resulting Delaunay triangulation of those data points. It can be seen by inspection that the points, A, B, and C form a set of Thiessen neighbours and a Delaunay triangle as the circumscribing circle through those points includes no other points in the data distribution. This is also the case for all other triangles shown in the network on the right of Figure 3.4.

In all triangulation systems there is a boundary problem that must be solved in some manner. The points lying within the data window (left box, Figure 3.4) may well not be isolated but only part of a larger data set. If this is the case any arbitrary triangulation around the outside of the present data area must be incorrect. As it is impossible to know what lies outside the data window some boundary condition must be set up to act as a frame and to provide a set of boundary triangles so that isarithms can be extrapolated outside the present apparent data area. Many possibilities exist for this but one of the most efficient is to place a set of imaginary points around the outside of the area, just outside the data window. The position of these imaginary points is shown on the right of Figure 3.4, as is the Delaunay triangular relationship with the real data points. Once these imaginary points have been added to the original data set the whole area can be triangulated starting with any pair of imaginary points as initial known neighbours. The question of how many imaginary points should be used and their distribution is considered later.

FIGURE 3.4

Data structure

A suitable data structure that enables fast access to points lying in the immediate proximity of others according to Knuth (1973) is a two-dimensional sort that can be likened to a box structure. Figure 3.5 shows a simplified data set containing some 16 points. A single sort in the X direction results in the X order shown at the bottom of the diagram and similarly the result of a Y order is given to the right of the diagram. Only the four points labelled 1, 4, 8, and 12 have been entered in the X and Y order. If first a Y order is performed, and then multiple X orders, a box structure can be achieved. The Y order has first been split into a series of Y sections.

This operation of putting data points in boxes is very fast and linear as no sorting is necessary. A division of the X and Y coordinates by the box side length provides an instant reference to the box that contains the point. In Figure 3.5 boxes A and B form the first Y section and C and D the second section. Each Y section is then sorted in terms of X and the resulting X order can be divided into sections in its turn thus segregating A and B into separate boxes and later segregating C and D when the second Y section has been put in X order. The result is to create a succession of points in an ordered listing such that each row of boxes is in Y order, but the data inside each box is in X order. An index to the first point in each box can then be created as in the right of Figure 3.5. Given the coordinate position of any point in the area it is then very fast to find the boxes that will contain its probable neighbours. In the simple 2 x 2 box structure of Figure 3.5 there is only a small advantage to be gained. The larger the data set, however, the greater the saving will be providing the box structure becomes similarly more detailed.

An immediate question is the number of points which should be contained on average within each box in order to maximise the likelihood of finding the desired Theissen neighbour as quickly as possible. As the search procedure is local both the rotation point and the known neighbour of Figure 3.3 will often exist in the same box. As the rightmost neighbour is wanted in the clockwise algorithm of Figure 3.3, the data distribution is assumed random, and the two base line points are probably in the box. An average of four points seems reasonable. Very few redundant points will be searched while looking for the Thiessen neighbour for any given base line.

Of course base line length will be variable depending on data distribution. In those cases where the base line is very long, the number of boxes containing on average four points that will have to be opened and searched will be quite large. They will not, however, contain as many points as if a uni-dimensional sort was employed.

Figure 3.6 shows the application of the structure to a more complex data set. Here the total data set includes about 80 points which leads to a rectangular 20 box structure with 5 boxes along the X axis and 4 boxes down the Y axis. A single sort algorithm would search on the basis of an outwards-inwards approach in that the points most likely to be found first and considered as neighbours are those likely to lie furthest away from the base line. On average many incorrect

neighbours will be tested before the right point is found.

Neighbour search algorithm

A better approach would be to attempt to find the most probable neighbours first, thus reducing calculation but giving a complicated program structure. In Figure 3.5 the base line A-B has the rotation point B and known neighbour A. As the algorithm is always looking for the most clockwise neighbour only points above the diagonal line need be considered. All boxes below the diagonal line can be ignored. It is clear from the diagram that the neighbour that will form the Delaunay triangle is included in box 14 and happens to lie within the circle of which line A-B is a diameter. An efficient starting point for a search would therefore be one that considered in the initial cases all boxes covered by the circle with A-B as diameter - boxes 13 and 14. Any points below the line A-B could be discarded immediately leaving four possible neighbours. These could then be checked individually using either the circumscribed circle centre calculation approach of Brassel and Reif or alternatively using a faster method proposed later in this Chapter. It is possible that the point, while lying within a box that is to be searched, may not lie within the circle shown in Figure 3.6. If no other points exist in boxes 13 and 14 other than points A and B the preliminary search will be unsuccessful. It will, however, have taken very little time as search time is only related to points that are present, and opening a box to check its contents is very fast. If boxes 13 and 14 had been empty the search would then have enlarged over the surrounding area of boxes 15, 18, 19 and 20 according to the inwards-outwards rule. It is worth comparing the average number of points likely to be searched using the outwards-inwards approach and the inwards-outwards approach. In the former the equivalent of approximately 5 boxes would be searched containing on average 4 points, giving a total of 20 points. Many of these would not involve a complete check as they are on the wrong side of the line or obviously too far away, but would be perfunctorily checked. Using the alternative approach only on average 8 points would be checked, with only partial checking of many of the points. In a large data set, the box structure still on average holding 4 points, the necessary search would still require 8 points. In the single sort method the number considered would have risen dramatically.

An inwards-outwards search involves an iterative expansion of the area being searched if no suitable points are found. If there are large empty areas in the data set there will be a large number of empty boxes. As these can be searched very quickly speed is not significantly degraded in the empty areas. There is, however, some decrease in speed in the populated areas as, although the average contents per box would have been four, the actual contents in the populated area will be proportionately higher to compensate for the empty areas. The process of iterative expansion is shown in Figure 3.7 indicating the method of determining the Delaunay triangle and of resolving problem cases. The algorithms always start from a rotation point and a known neighbour which form a base line for further computation. The area of interest for the first search is defined as the series of boxes covered by the circle having as its diameter the

base line. A first approximation is that the boxes to be searched are those covered by the square that encloses the circle. This ensures fast determination of the desired area but can lead to difficulties at the corners of the square but outside the search circle. In Figure 3.7 point 3 can be seen to lie within circle A and would definitely be the Thiessen neighbour required to form the Delaunay triangle with the base line. As it is quite possible that more than one point will lie within the circle it is important to be able to distinguish immediately which point inside the search circle will be the Thiessen neighbour.

Point choice

Consider a point lying on circle A on the clockwise side of the base line. Wherever that point may fall the angle subtends from the base line to that point will always be the same. In the special case when the base line is the diameter of the circle that angle will be a right angle. Any point lying outside circle A will have an angle more acute and any point inside subtended an angle more obtuse than the circle perimeter angle. Simple geometry shows that the desired neighbour is the point lying inside the circle which has the largest angle subtended from the base line. Point 3 will have an angle rather greater than 90 . If there were another point inside circle A as well as point 3 and its angle were larger than that subtended by point 3 then it would be the Thiessen neighbour and would form the Delaunay triangle.

Imagine that circle A has no point inside it. The search circle area must then be increased by some factor to the size indicated by circle B. All boxes covered by the square enclosing circle B are then opened and the points inside inspected. Points 1, 2, 4 and 10 would be discovered. The only point subtending an angle larger than the perimeter angle of circle B would be point 4 and hence it would be discovered as a neighbour. The inefficiency of using the enclosing square rather than the circle B as the search area is that point 10 would be investigated as it lies within the square but outside the circle but would be discarded as its subtending angle is less than the circle B perimeter angle. This inefficiency can be quite useful in that if no points exist inside the search circle the new maximum search circle that need be employed is that which would include point 10 on its perimeter rather than the much larger circle C in Figure 3.7. Thus, although point 10 would be ignored in the presence of point 1, 2 and 4, it would serve as a limitation for further search if those interior points were not present.

It is possible that in some cases, particularly with regularly distributed data, more than one point may lie exactly on the circle perimeter whereas no points lie inside the circle. It is necessary to employ a decision rule to decide which (in Figure 3.7) of the points 1 and 2 should be chosen as the Thiessen neighbour of the rotation point and the known neighbour. The angles subtended at points 1 and 2 are the same, but in terms of a clockwise rotation around the rotation point it is clear that point 1 should be chosen in preference to point 2. The correct neighbour can be determined once the point closest to the known neighbour and the point closest to the rotation point have

been determined. The chosen point is the one that is closest to the known neighbour providing it is not also closest to the rotation point.

If no points are found in either circle A or B the search circle would be enlarged again to size C. This process could continue indefinitely at the edge of the data point distribution as there would be no further points to discover and no limit to which the circle size could be increased that would discover any further points. This is one reason why the imaginary points that lie outside the data window in Figure 3.4 are essential. Any expanding circle search inside the data area will be halted at some stage by contact with an imaginary point that will always be able to form a Delaunay triangle and act as a Theissen neighbour. When the rotation point is itself an imaginary point as is the known neighbour, further effort is abandoned as the two points are already known to be neighbours. It is therefore essential to include the imaginary points in the basic box structure of the data. Every perimeter box contains one imaginary data point. This ensures that a limit is set on all searches no matter which boxes are opened during a search for a neighbour.

The increase in circle size at each phase of a possible search must ensure that the absolute minimum of redundant searching is performed and that the circle area does not grow so rapidly that a vast number of new points are found. A reasonable area increase lies between two and three times for each circle expansion. If the base line is very small this is usually indicative of an area in the window where data is densely packed. Although absolute circle size growth will be slow, it is likely a neighbour will be found very quickly. If the base line is large there is always the possibility that an over large number of points will be searched, but as the algorithm starts with the base line as diameter, as in Figure 3.7, even then the number of points being searched is unlikely to be too excessive.

Triangulation progression

The triangulations resulting from many triangular mapping program systems have been critically affected by the choice of the starting point for the triangulation process. It is essential that the triangulation should be stable within the data area irrespective of starting position. This is particularly necessary to ensure that maps of consecutive areas can be overlapped successfully without join marks so that updating and modification of maps can be performed without complete redrawing of an entire area. Figure 3.8 shows the triangulated data set of Figure 3.4 in two orientations. In A the orientation is the same as in Figure 3.4 and in b the data set has been rotated through 180 degrees. As the Delaunay network is unique, the triangulation is the same in both cases. The process of triangulation is in both cases that of an expanding wave.

As an arbitrary decision the bottom left hand corner is chosen as the starting point. The neighbours for the bottom left hand corner point are those in contact with shell 1 in Figure 3.8. Thus after the points on shell 1 are found we have complete rotational information for the lower left corner point and partial information for the points

lying on shell 1. The information available for these shell points concerns all their relationships to the previous shell. At the start the only information available is related to their neighbours on the first shell and to the original starting point, which is on shell 0. Each point on shell 1 is then considered in turn as a rotation point and a new set of neighbours built up on shell 2. By the time shell 2 is complete all neighbours for points on shell 1 have been found and some of the neighbours for shell 2 including all neighbours on previous shells. For both Figure 3.8A and 3.8B the process continues shell after shell until by shell 6 all neighbours for all points have been found. Every point is only visited as a rotation point once unless its neighbours have already been discovered from previous rotation points in which case it is complete. An updating process enables track to be kept of all the relationships. Investigation of the length of each shell indicates that in large data sheets only a small proportion of the points exist on any given shell. As these points are the only active ones in terms of finding new neighbours and also for the relationships between points, they are the only ones that need to be kept in memory. Relationships that have to be maintained for instant reference are kept quite small. The algorithm can therefore be run equally efficiently on small or large computers. In all, the number of active relationship lists that need be kept will be on average approximately 1.5 times the square root of the number of data points for a randomly distributed data set, with six items per list.

Although the same starting location was used for both A and B in Figure 3.8, the shell formation is different as it adjusts to the different data distribution encountered. The final pattern of the triangulation, however, is the same.

In Laser-Scan DTM software a rather different order of calculation is used. The first shell includes all the imaginary points. Thus the shells are circular. This uses rather more memory as the live shell is approximately 4 times the square root of the number of points (at maximum).

CHAPTER 4

MODULE TRIANG

MODULE TRIANG

REPLACES PANACEA module PANIC

FUNCTION

TRIANG is the main data structuring program in the DTMCREATE package. Its purpose is to extract heighted data from IFF (Internal Feature Format) files and existing DTMs held in DTI (Digital Terrain Image) files to produce a triangular data structure.

FORMAT

$ TRIANG

COMMAND QUALIFIERS

None, TRIANG is command driven.

DESCRIPTION

General

TRIANG can handle input from IFF (Internal Feature Format) files consisting of strings of contour, seismic, or other types of data, marked as either continuous or discontinuous data points or cliffs. In addition data can be input from existing DTI (Digital Terrain Image) format DTM files as an aid to intra-DTM edge matching.

The input data are used to form a Delaunay triangulation. This provides a structure that relates every point with its Thiessen neighbours. The triangulation is optimum in the sense that it has the most equilateral set of triangles possible for the data set under consideration. This does not necessarily imply that locally the triangles will be very equilateral.

As an alternative to production of an optimal Delaunay triangulation a CONSTRAINT option is offered. This option forces the triangulation to honour the connectivity of contour, coastline and river strings etc. such that no string is cut through by a triangle. This approach is more suitable for the definition of complex geomorphological surfaces than the Delaunay ideal and is strongly recommended for geomorphological surface modelling.

TRIANG enables the user to apply feature flagging to the incoming data points to identify each point as either a river point, a ridgeline point or an unflagged point. This deature flagging is used within TRIGRID to control the limits applied to smooth surface interpolation.

Coordinate system

DTMCREATE assumes absolute coordinates, (i.e. all coordinates in input files are offset by the file (X,Y) origin offset) in the specification of windows and positions.

The UNITS command enables the user to specify in what units of measurement he wishes to define the triangulation window using the WINDOW command, or in what units of measurement details from the header of the DTI file are displayed.

By default metre units are assumed.

The UNITS command should be given before specifying the triangulation window the user wishes to specify the window in units other than metres.

Currently the projection information for the first input file read with a FILEIN command is copied to the .NOD and .DTA output files for subsequent use in setting up the header of the LSLA type DTI file created in TRIGRID. In a future release of DTMCREATE, coordinate consistency checks between input files will be implemented.

Output data files

Output from module TRIANG is in the form of two binary files, not directly readable by the user. For details of default file specifications used for the binary output files see the FILEIN command section below.

The binary output files are used to convey the triangulation data structure between the various modules of the DTMCREATE package. The first file (filename.DTA) contains the data in scaled integer form, together with various indices and tables concerning the distribution of the data over the triangulation area. This data set is augmented by a set of imaginary points acting as a frame around the edge of the map area. The second file (filename.NOD) contains the data structure itself with a list of the neighbours of each data point in the set.

These triangulation files must always be handled as matched pairs, as the data contained in one is the key to the inter-node structures described in the other. If one file is to be deleted then BOTH files must be deleted. DTMCREATE modules TRIEDIT, TRIDER and TRIGRID which require input from .NOD and .DTA files check that the two files share a common version number and complain if they do not!

Never use the DCL SET FILE, RENAME or COPY commands to alter the version numbers of mismatched .NOD and .DTA files to make them into a matched pair.

Input data files

Input to module TRIANG consists of contour and breakline strings and spot heights in IFF files, or DTI format DTM files. To swap input between these different file types the FORMAT command should be used.

TRIANG and input from IFF files

TRIANG is designed to be compatible with the "new" type IFF files introduced in conjunction with the IMP (IFF Map Processing) package. The origin offset entry in a type 2 MD (Map Descriptor) is used to offset coordinate values within an input IFF file. Although downwards compatible with "old" type IFF files a warning message is issued if an IFF file is found not to contain a set type 2 MD (Map Descriptor) entry.

Data within IFF files may have string type and feature type attributes assigned within TRIANG via the IFF feature code and layer values. Commands such as ASSIGN BREAKLINE_FC, ASSIGN RIDGE_LAYER etc are provided to make the feature type/code assignments to enable TRIANG to realize that a string is to be stored as a breakline or a ridgeline etc.

Internal scaling of input coordinates

TRIANG scales all the input coordinates of the IFF file into integer form in order to speed up calculations. The range of integers into which the coordinates will be scaled will lie between 0 and 300000 by default.

However, the user may define a logical, LSL$DTMCREATE_RESOLUTION, to be a value between 300000 and 10000000 to increase this resolution and will default to 300000 if it is not given.

The purpose of this is because very dense datasets could have points so close together that TRIANG would become 'stuck' or produce the STOPNOD error message as it was unable to distinguish between them.

IMPORTANT

> It is essential that the logical value LSL$DTMCREATE_RESOLUTION remains the same when going from TRIANG all the way through to TRIGRID on a particular dataset. If the resolution is altered between any of the stages, unpredictable results will occur and programs may fail.

IFF Heights

IFF Heights - storage in contours and spot heights

In IFF files height values are transmitted either via AC (Ancillary Code) entry values associated with 2D ST (STring) entries, or, CB (Coordinate Block) entries which provide per point height tagging. By default contour and spot height Z values are read from type 3 ACs as floating point values. By use of the ENABLE INTEGER_HEIGHT command, integer heights may be read from type 2 ACs.

IFF Heights - storage in cliff-lines

Cliff lines heights may be transmitted in one of two ways:

1. The IFF feature contains two constant floating point height values; one for the terrain to the left of the cliffline (relative to the direction of digitising) and one for the height of the terrain to the right. The left hand height is held in a type 80 AC and the right hand height in a type 81 AC.

2. The cliffline coordinates have per-point attributes for the height on the left and right of the cliffline (relative to the direction of digitising). This enables clifflines to have heights which vary along their top and bottom. The per-point attributes are held in CB (Coordinate Block) entries. CB entries replace the use of ST (STring) and ZS (3D string) entries.

   IFF features which represent clifflines will have the X,Y and two attribute fields set for each coordinate in their CBs. The attributes are 80 ("cliff left height") and 81 ("cliff right height"), both of which contain a real (floating point) height value.

   To generate CB entries containing (X,Y,Cliff-left,Cliff-right) coordinates, the user will have to digitise the clifflines using LITES2 with the IFF output revision level set to 1. For details of IFF input and output revision levels and the use of LITES2 for the generation of CBs see the MAPPING Reference Manual and the LITES2 Reference Manual.

   LITES2 macros can be set up to facilitate digitising cliffline features which are to receive varying Z values along their left and right sides. The following unsophisticated examples are provided for guidance:

```
%DECLARE REAL LEFT
%DECLARE REAL RIGHT
%ENABLE SUBSTITUTION

%MACRO CLIFF_START
%INQUIRE LEFT "Cliff left height  "
%INQUIRE RIGHT "Cliff right height  "
%MESSAGE "  "
%SET ATTRIBUTE 80 'LEFT'
%SET ATTRIBUTE 81 'RIGHT'
%START
%ENDMACRO

%MACRO CLIFF_STOP
%INQUIRE LEFT "Cliff left height (final point)  "
%INQUIRE RIGHT "Cliff right height (final point)  "
%MESSAGE "  "
%SET ATTRIBUTE 80 'LEFT'
%SET ATTRIBUTE 81 'RIGHT'
%END
%ENDMACRO
```

This command sequence could be typed into a LITES2 command file and the command file read at the start of the LITES2 session by use of the LITES2 @'file-spec' command.

If possible a puck or function button should be defined to issue each of the two macros. A single press of the button defined for the CLIFF_START macro will then cause the user to be prompted for the left and right height and the point will then be inserted with the appropriate attribute values. Subsequent points should be inserted using the same button. The button defined for the CLIFF_STOP macro should be pressed to insert the last point in the cliff string.

It is essential that the user sets the IFF output revision level to 1, before starting the LITES2 session. Failure to do this will result in all the per-point attributes being lost when the user exits the LITES2 session.

IFF Heights - changing height datum and IFF height allocations

IFF files are read into TRIANG using the FILEIN command. More then one input file may be specified using a new FILEIN command for each file. Defaults may be changed between reading files. This means that IFF files with different allocations of IFF entries for height information or with heights relative to different data may be combined.

Height information within a single IFF file may be stored relative to different height data providing that the features pertaining to each height datum can be distinguished by feature code or layer. The DATUM command can be used to set the height datum for a subsequent FILEIN command. If required, a single IFF file may be read in many times relative to different height data, feature selection being achieved by SELECT FC and SELECT LAYER commands.

Imperial heights may be converted to metric on input with the ENABLE TOMETRES command. The reverse is possible with the ENABLE TOFEET command. Incoming heights may be multiplied or divided by a user specified constant with the ENABLE MULTIPLYBY and ENABLE DIVIDEBY commands respectively.

Incoming heights may be inverted if the ENABLE INVERSE command is specified prior to reading an IFF file. This enables modelling of hydrographic data where sea depths are often stored as positive heights, drying zone heights as negative and land heights as positive!

BREAKLINES

In order that all surface discontinuities are honoured in the final DTM it may be necessary to designate some IFF strings as "breaklines", which will ensure a change of slope character at that line. A breakline in the IFF file may be identified by its feature code or position in a separate layer. The allocation of IFF feature codes and layers for breaklines must be input to TRIANG using the ASSIGN BREAKLINE_FC and ASSIGN BREAKLINE_LAYER commands respectively. As with the IFF height storage mechanisms different allocations of layers and feature codes for breaklines may be made between successive FILEIN commands.

As breaklines define important slope discontinuities in the landscape, it is important that their points are always retained. TRIANG takes great care to ensure that breaklines are never violated during the (optional) triangulation constraint phase, and that all river, coast, and isoline strings are cut where crossed by a breakline. It is therefore important that care is exercised in the registration of breaklines relative to other source data during data preparation.

It is recommended that breaklines, clifflines and spot-heights are read in before the bulk of contour data. Spot heights often contribute much to the form of hill tops and valley floors. All other data may then follow, in an order reflecting the importance attached to the landscape elements which they describe.

Once all the source data is read into TRIANG, triangulation may be started. Unless the CONSTRAINT option has been enabled, (see below), at the end of the triangulation phase the binary files containing the triangulation will be output.

TRIANGULATION CONSTRAINT

If CONSTRAINT is in force the following procedure is adopted. TRIANG's internal workspace is reorganised and if necessary a random access disk file is set up to enable rapid selection of node neighbours. Breaklines are identified and their course traced through the data structure. If the neighbour list of the nth node does not contain a reference to the (nth + 1) node along a string then a logical break has occurred. This break is repaired by the insertion of an additional point at the intersection of the line between the nth and (nth + 1) node in the string and every triangle link which cuts that line. These additional 'patch' nodes are stored in a separate area of workspace for later use. The node neighbour lists are updated and the next node in the breakline string is considered. The process is repeated until all breakline strings have any gaps filled with additional points.

All the 'patches' of additional points are then inserted into the breakline gaps and the strings are reordered so that the patch points appear in the correct positions within the strings. The whole process is then repeated for non-breakline strings. However, any breakline connections encountered while tracking along a non-breakline string will be considered inviolable and the non-breakline string will be broken at the intersection point. After all string patching is complete, the (probably somewhat enlarged) intermediate files will be output to disk for use by TRIDER or TRIEDIT.

The number of data points generated to patch strings will depend on the original data density and distribution of the data points. A river containing points at 10cm intervals which lies between contours having points at 1mm intervals will probably be cut by triangle connections many times. Many points will be generated in the river string patches to restore the continuity of the string. If the relative point density was reversed between the two features, it is most unlikely that the river would be cut by triangle links at all.

CLIFFS

Cliffs are a comparatively complex feature to digitise as two heights must be associated with each (X,Y) coordinate pair, one height for the right hand side of the cliffline and one for the left hand side. The sides of the cliff line are defined relative to the direction of digitising. (See **Heights** above).

A cliff feature should be given a type 80 AC (Ancillary Code) to carry the height on the left side of the cliff (relative to the direction of digitising) and a type 81 AC (Ancillary Code) to carry the height on the right side of the cliff.

A cliffline must have at least 4 points.

Internally DTMCREATE supports 2 cliff data types:

1. open cliffs, which may rise out of and then sink back into a continuous surface. When digitising a line which is to be interpreted by TRIANG as an open cliff, care should be taken to observe the following:

    - The locations of the first and last points in the digitised string should not lie in such a position that each will become a neighbour of the other within the triangulation. If they do fall within each others neighbour lists, the line will be treated as a closed cliff, with a continuous top and bottom loop, (like an island in the sea).

    - TRIANG will treat the first and last points in the digitised line as the positions from which the cliff starts to rise from the surface and finally sink back into the surface. The distance between the first to the second point and the distance between the penultimate point and the last point controls how rapidly the cliff rises from and falls back into the surface. If it is desired that the cliff should rise and fall abruptly at its ends, these inter-point distances should be kept as small as possible without digitising the points so close together that TRIANG discards one of each on initial read-in.

    - The link between the first point and the second, and the penultimate with the last is not flagged as a line of derivative discontinuity.

    - TRIANG will automatically assign to the first and last points in the string the minimum of the two cliff heights (left and right)

2. Closed cliffs, which in plan form a continuous closed loop, (for example, an island in the sea or a lake).

    - The locations of the first and last points in the digitised string should lie in such a position that each will become a neighbour of the other within the triangulation. If they do not fall within each others neighbour lists the line will be treated as an open cliff, which rises from and then sinks back into the surrounding surface.

The relationship between cliffs and coastlines

It is recommended that all coastlines are defined as cliff lines having as their seaward height 0.0 and their landward height 1.0 (or more). This will ensure a clean "step" between land and sea. Water bodies in estuarine areas may be similarly defined. Because clifflines are treated as lines of surface discontinuity, their use at

the edge of water bodies like the sea, ensures that no rippling of the flat water surface occurs. Such surface rippling is a result of the use of a smooth patch function as employed during DTM grid estimation within TRIGRID.

In sea areas, if no other data lies between a coastal cliff and the edge of the triangulation area, a breakline of height 0.0 should be used along the edge of the triangulation area. This will ensure that heights estimated for imaginary points along the edge of the sea area do no cause slopes in the sea!

TRIANG and input from DTI files

TRIANG can accept for input MIKE, UHL1, TED4 and LSLA type DTI (Digital Terrain Image) files. The elevation data within these files may be held as word (default), real*4 and longword values. Byte and bit valued DTI files are not supported by DTMCREATE.

TRIANG is designed to enable the user to edge match with DTI files which contain null DTM post values (i.e. post values which have been assigned a height value of -32767). This is achieved by ignoring all posts which have the null height value.

TRIANG reads and interprets the header information from the input DTI file. DTI files of type UHL1, TED4 and LSLA with projection record, contain an (X,Y) origin offset value for the SW corner (row 1, column 1) of the file in the file header information. As TRIANG is particularly concerned with the registration of the DTI file data with data from IFF input sources, unless the origin offset information is available from the file header, the user is prompted for the position of the SW corner of the DTI file.

The MIKE type of DTI header is now obsolete. DTMCREATE continues to support input of data from this file type but it is recommended that the user converts any existing MIKE type DTI files to the new LSLA type with projection record by using the MATRIX utilities DTICONVERT and DTITRANS. This gives the advantage that all the file locational information is stored in the header of an LSLA type DTI file and consequently the user is not prompted for manual specificatyion of origin offset coordinates at run time.

It is important that the DTI file (X,Y) grid intervals and the value supplied for the position of the SW corner of the DTI file are expressed in the same units of measurement as used in the IFF input files. It is possible to change the header values in a DTI file by use of the MATRIX package DTIPATCH utility although care should be exercised to ensure that changes are made that are compatible with other stages of the production flowline.

It is essential to ensure that all IFF data is transformed into the coordinate system of the output DTMs prior to using DTMCREATE. Failure to do this will mean that the completed DTM will have to be transformed using the MATRIX utility DTITRANS.

Data extracted from DTI format DTM files are always read in as normal 3D strings, the discrete post values of a row or column of the DTI file being logically strung together for ease of data handling.

EDGE MATCHING

DTMCREATE maintains a concept of "imaginary points" which lie around the edge of a triangulation. These are used to stop the DTM surface from "falling off the edge of the world" during surface fitting. When created by TRIANG these imaginary points have location but no Z value. Later, when estimating slope derivatives TRIDER offers the user several interpolation options to estimate suitable Z values for the imaginary points at the edge of the triangulation. Of course the resulting heights are only estimates and it would be better to have actual data points available at the triangle vertices near the edge of the triangulation.

DTMCREATE offers two approaches to edge matching between DTMs using real data points:

1. Overlap of source data used for triangulation. This can be achieved by triangulating data covering a bigger area than that of the final DTM. The overlap area that this approach requires ensures that real world, rather than imaginary control is available for the edge of the DTM area. When the adjacent DTM area is triangulated, a corresponding overlap area is taken into the previously triangulated area. The result is that the same control points are available for DTM post estimation within the edge zone of both DTMs. The same post values should then result, plus or minus arithmetic rounding errors.

2. TRIANG allows the user to force the edges of adjacent DTMs to have common post heights by supplying as data to TRIANG the edge posts of completed DTMs. A maximum of 2 rows or columns should be used and are selected out of the DTI file using the TRIANG WINDOW command in reference to the origin and extents of the DTI file. Because the coincident posts of adjacent DTMs are treated as nodes in the triangulation, DTMCREATE preserves their location and height throughout the modelling process. The end result is that exactly the same post values will appear in both models. For perfect edge matching this is the recommended procedure.

The DTMCREATE system assumes default settings for all options and parameters unless otherwise specified. The status for all options may be examined at any time using the SHOW ENABLE command.

Obligatory command sequence

Although TRIANG offers the user a bewildering number of possible commands, only 6 are obligatory for a successful TRIANG run. TRIANG contains interlocks to ensure that processing steps cannot be entered without having first defined the necessary defaults.

The obligatory TRIANG command sequence is:

```
WINDOW 'real' 'real' 'real' 'real' ! define area of triangulation
ZLIMITS 'real' 'real'              ! specifies the Z-range of the data
MAXPOINTS 'integer'                ! specifies approx. number of points
FILEOUT 'file-spec'                ! create and open output files
FILEIN  'file-spec'                ! input data (default IFF)
GO                                 ! triangulate it
```

This set of commands is sufficient to produce a triangulation containing no breaklines or geomorphological feature tagging. Input is to be all the data from a single IFF file that falls within the specified window.

The exact sequence given above is not rigidly enforced. The only hard and fast rules are that WINDOW, MAXPOINTS, ZLIMITS, FILEIN and FILEOUT must come before GO and that the other commands come, in any order, before FILEIN.

TRIANG commands

@

Take command input from the specified file.

FORMAT: @file-spec<CR>

Command parameters:

file-spec

The file to be opened and used for command input.

Any parts of the file-spec not supplied for the @ command will be taken from the default specification 'SYS$DISK:[].COM;0'.

DESCRIPTION:

TRIANG offers the facility of command input from an indirect command file. The '@' character preceding a file-spec will cause TRIANG to open and read commands from the specified file until:

1. a RETURN command is detected and command input is returned to SYS$COMMAND.

2. a GO command is detected - after completion of the triangulation TRIANG exits.

3. end-of-file is detected. This provokes an error message and command input is returned to SYS$COMMAND.

Nested command files are not supported (i.e. a command file containing an '@' command), although sequential '@' commands are supported when read from SYS$COMMAND.

As an aid to batch log interpretation TRIANG will echo all commands read from an indirect command file.

Messages:

The following messages are specific to the @ command:

*** WARNING *** "@" must precede a file-spec

*** WARNING *** Indirect file error - returning to terminal input

*** ERROR *** Can't open indirect command file 'file-spec'

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:30:39 20-AUG-87
TRIANG> @FLOW2<CR>
TRIANG> ENABLE DIAGNOSTICS
TRIANG> FRT FLOW2
FRT file LSL$FRT:FLOW2.FRT;8 opened for read
TRIANG> ASSIGN CLIFF_FC OUTCROPS,7,COAST
TRIANG> RETURN
TRIANG>
```

!

Treat all text to the right of the '!' as a comment.

FORMAT: ! [comment text]

Command parameters:

comment text

text that is to be treated as a comment and which will be excluded from command interpretation.

DESCRIPTION:

An exclamation mark is the standard DTMCREATE package comment delimiter. All text (and numbers) which lie to the right of a '!' character are excluded from command interpretation. Comments are useful for annotating command procedures used in batch processing etc.

Messages: None.

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:53:27 3-FEB-89
TRIANG> ! a comment for the sake of it<CR>
TRIANG> WINDOW 0.0 0.0 900.0 900.0<CR>
TRIANG> MAXPOINTS 23000<CR>
TRIANG> ZLIMITS 0.0 1290.0 !  limits are in metres<CR>
TRIANG> FILEOUT TEST3<CR>
.DTA file DUA3:[DEMONSTRATION]TEST3.DTA;6 opened for write
.NOD file DUA3:[DEMONSTRATION]TEST3.NOD;6 opened for write
TRIANG> FILEIN JOE.IFF!  get the first IFF file<CR>
IFF file LSL$IF:JOE.IFF;0 opened for read
TRIANG> !  do the triangulation<CR>
TRIANG> GO<CR>
 ELAPSED:  00:05:25.84 CPU:  0:00:05.71 BUFIO:  281 DIRIO:  46 FAULTS:  263
```

ASSIGN BREAKLINE_FC

Specifies the feature codes of IFF features which are to be treated as breaklines when included in the triangulation.

FORMAT: ASSIGN BREAKLINE_FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. ASSIGN BREAKLINE_FC 2:6 will result in the assignment of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIANG any valid feature code group names may be used as arguments to the ASSIGN BREAKLINE_FC command.

DESCRIPTION:

The ASSIGN BREAKLINE_FC command complements the DEASSIGN BREAKLINE_FC command. ASSIGN BREAKLINE_FC enables the user to specify the feature codes of IFF features which are to be treated as breaklines when included in the triangulation.

By default no feature codes are assigned for breaklines, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation. Explicit IFF layer or feature codes may be removed from a assignment list by use of the appropriate DEASSIGN command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the ASSIGN BREAKLINE_FC command:

*** WARNING *** You must have read an FRT file to be able to use group names
*** WARNING *** No groups have been defined in the FRT
*** WARNING *** Illegal feature code 'integer'
*** WARNING *** Bad group name 'group-name'

Examples:

TRIANG> **ASSIGN BREAKLINE_FC 6:9,WATER,126<CR>**
TRIANG>

ASSIGN BREAKLINE_LAYER

Specifies the IFF layer numbers containing features which are to be treated as breaklines when included in the triangulation.

FORMAT: ASSIGN BREAKLINE_LAYER layer[,...]

Command parameters:

layer

An IFF layer number which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. ASSIGN BREAKLINE_LAYER 2:6 will result in the assignment of layers 2,3,4,5 and 6.

DESCRIPTION:

The ASSIGN BREAKLINE_LAYER command complements the DEASSIGN BREAKLINE_LAYER command. The ASSIGN BREAKLINE_LAYER command enables the user to specify the IFF layer numbers containing features which are to be treated as breaklines when included in the triangulation.

By default no layers are assigned for breaklines, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation. Explicit IFF layer or feature codes may be removed from a assignment list by use of the appropriate DEASSIGN command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the ASSIGN BREAKLINE_LAYER command:

*** WARNING *** Too many layer arguments in one command
*** WARNING *** Illegal layer number 'integer'

Examples:

TRIANG> **ASSIGN BREAKLINE_LAYER 21:29,126<CR>**
TRIANG>

ASSIGN CLIFF_FC

Specifies the feature codes of IFF features which are to be treated as cliffs when included in the triangulation.

FORMAT: ASSIGN CLIFF_FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. ASSIGN CLIFF_FC 2:6 will result in the assignment of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIANG any valid feature code group names may be used as arguments to the ASSIGN CLIFF_FC command.

DESCRIPTION:

The ASSIGN CLIFF_FC command complements the DEASSIGN CLIFF_FC command. The ASSIGN CLIFF_FC command enables the user to specify the feature codes of IFF features which are to be treated as cliffs when included in the triangulation.

For information about digitising and the internal treatment of clifflines, see the main Description section above.

By default no feature codes are assigned for breaklines, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation. Explicit IFF layer or feature codes may be removed from a assignment list by use of the appropriate DEASSIGN command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the ASSIGN CLIFF_FC command:

*** WARNING *** You must have read an FRT file to be able to use group names
*** WARNING *** No groups have been defined in the FRT
*** WARNING *** Illegal feature code 'integer'
*** WARNING *** Bad group name 'group-name'

Examples:

```
TRIANG> ASSIGN CLIFF_FC 6:9,WATER,126<CR>
TRIANG>
```

ASSIGN CLIFF_LAYER

Specifies the IFF layer numbers containing features which are to be treated as cliffs when included in the triangulation.

FORMAT: ASSIGN CLIFF_LAYER layer[,...]

Command parameters:

layer

An IFF layer number which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. ASSIGN CLIFF_LAYER 2:6 will result in the assignment of layers 2,3,4,5 and 6.

DESCRIPTION:

The ASSIGN CLIFF_LAYER command complements the DEASSIGN CLIFF_LAYER command. The ASSIGN CLIFF_LAYER command enables the user to specify the IFF layer numbers containing features which are to be treated as breaklines when included in the triangulation.

For information about digitising and the internal treatment of clifflines, see the main Description section above.

By default no layers are assigned for cliffs, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation. Explicit IFF layer or feature codes may be removed from a assignment list by use of the appropriate DEASSIGN command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the ASSIGN CLIFF_LAYER command:

*** WARNING *** Too many layer arguments in one command
*** WARNING *** Illegal layer number 'integer'

Examples:

```
TRIANG> ASSIGN CLIFF_LAYER 21:29,126<CR>
TRIANG>
```

ASSIGN RIDGE_FC

Specifies the feature codes of IFF features which are to be treated as ridgelines when included in the triangulation.

FORMAT: ASSIGN RIDGE_FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. ASSIGN RIDGE_FC 2:6 will result in the assignment of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIANG any valid feature code group names may be used as arguments to the ASSIGN RIDGE_FC command.

DESCRIPTION:

The ASSIGN RIDGE_FC command complements the DEASSIGN RIDGE_FC command. ASSIGN RIDGE_FC enables the user to specify the feature codes of IFF features which are to be treated as breaklines when included in the triangulation.

By default no feature codes are assigned for ridgelines, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation. Explicit IFF layer or feature codes may be removed from a assignment list by use of the appropriate DEASSIGN command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the ASSIGN RIDGE_FC command:

*** WARNING *** You must have read an FRT file to be able to use group names
*** WARNING *** No groups have been defined in the FRT
*** WARNING *** Illegal feature code 'integer'
*** WARNING *** Bad group name 'group-name'

Examples:

```
TRIANG> ASSIGN RIDGE_FC 6:9,WATER,126<CR>
TRIANG>
```

ASSIGN RIDGE_LAYER

Specifies the IFF layer numbers containing features which are to be treated as ridgelines when included in the triangulation.

FORMAT: ASSIGN RIDGE_LAYER layer[,...]

Command parameters:

layer

An IFF layer which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. ASSIGN RIDGE_LAYER 2:6 will result in the assignment of layers 2,3,4,5 and 6 to identify ridgelines.

DESCRIPTION:

The ASSIGN RIDGE_LAYER command complements the DEASSIGN RIDGE_LAYER command. The ASSIGN RIDGE_LAYER command enables the user to specify the IFF layer numbers containing features which are to be treated as ridgelines when included in the triangulation.

By default no layers are assigned for breaklines, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation. Explicit IFF layer or feature codes may be removed from a assignment list by use of the appropriate DEASSIGN command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the ASSIGN RIDGE_LAYER command:

*** WARNING *** Too many layer arguments in one command
*** WARNING *** Illegal layer number 'integer'

Examples:

TRIANG> **ASSIGN RIDGE_LAYER 21:29,126<CR>**
TRIANG>

ASSIGN RIVER_FC

Specifies the feature codes of IFF features which are to be treated as breaklines when included in the triangulation.

FORMAT: ASSIGN RIVER_FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. ASSIGN RIVER_FC 2:6 will result in the assignment of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIANG any valid feature code group names may be used as arguments to the ASSIGN RIVER_FC command.

DESCRIPTION:

The ASSIGN RIVER_FC command complements the DEASSIGN RIVER_FC command. ASSIGN RIVER_FC enables the user to specify the feature codes of IFF features which are to be treated as breaklines when included in the triangulation.

By default no feature codes are assigned for cliffs, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation. Explicit IFF layer or feature codes may be removed from a assignment list by use of the appropriate DEASSIGN command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the ASSIGN FC command:

*** WARNING *** You must have read an FRT file to be able to use group names
*** WARNING *** No groups have been defined in the FRT
*** WARNING *** Illegal feature code 'integer'
*** WARNING *** Bad group name 'group-name'

Examples:

```
TRIANG> ASSIGN RIVER_FC 6:9,WATER,126<CR>
TRIANG>
```

ASSIGN RIVER_LAYER

Specifies the IFF layer numbers containing features which are to be treated as rivers when included in the triangulation.

FORMAT: ASSIGN RIVER_LAYER layer[,...]

Command parameters:

layer

An IFF layer which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. ASSIGN RIVER_LAYER 2:6 will result in the assignment of layers 2,3,4,5 and 6 to identify rivers.

DESCRIPTION:

The ASSIGN RIVER_LAYER command complements the DEASSIGN RIVER_LAYER command. The ASSIGN RIVER_LAYER command enables the user to specify the IFF layer numbers containing features which are to be treated as breaklines when included in the triangulation.

By default no layers are assigned for breaklines, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation. Explicit IFF layer or feature codes may be removed from a assignment list by use of the appropriate DEASSIGN command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the ASSIGN RIVER_LAYER command:

*** WARNING *** Too many layer arguments in one command
*** WARNING *** Illegal layer number 'integer'

Examples:

TRIANG> **ASSIGN RIVER_LAYER 21:29,126<CR>**
TRIANG>

DATUM

Enables specification of a height datum to be added to IFF and DTI heights read using subsequent FILEIN commands.

FORMAT: DATUM value

Command parameters:

value

A floating-point height value which is to be added to IFF and DTI heights read using subsequent FILEIN commands. DATUM values are expressed relative to the contents of the DTI or IFF file to be read in.

DESCRIPTION:

The DATUM command enables specification of a height datum to be added to IFF and DTI heights read using subsequent FILEIN commands.

On program statup a DATUM default value of 0.0 is assumed.

Height information within a single IFF file may be stored relative to different height data providing that the features pertaining to each height datum can be distinguished by feature code or layer. The DATUM command can be used to set the height datum for a subsequent FILEIN command. If required a single IFF file may be read in many times relative to different height data, feature selection being achieved by SELECT FC and SELECT LAYER commands.

If the DATUM command is specified prior to reading a DTI file (using the FORMAT DTI and FILEIN commands), the datum change is applied to all posts within the DTI file.

If the INVERSE command has been used to specify height inversion the heights are inverted before the datum value is added.

Messages:

The following messages are specific to the DATUM command:

*** ERROR *** You must specify a floating point argument to the DATUM command
*** WARNING *** You must specify a floating point argument to the DATUM command

---

Examples:

TRIANG> **DATUM 8.2<CR>**
TRIANG> **FILEIN HUNSTANTON_BATHEMETRY<CR>**
IFF file LSL$IF:HUNSTANTON_BATHEMETRY.IFF;0 opened for read

```
+-------------------------------------------------------------+
|                                                             |
|               Starting pass through IFF file                |
|                                                             |
+-------------------------------------------------------------+


There are now 671 points in the DTM area
TRIANG> DATUM 0.0<CR>
TRIANG> FILEIN ATILLA<CR>
IFF file LSL$IF:ATTILA.IFF;0 opened for read


+-------------------------------------------------------------+
|                                                             |
|               Starting pass through IFF file                |
|                                                             |
+-------------------------------------------------------------+


There are now 55527 points in the DTM area
TRIANG>
```

DEASSIGN BREAKLINE_FC

Deassigns specified IFF feature codes from the list of feature codes identifying breaklines to be included in the triangulation.

FORMAT: DEASSIGN BREAKLINE_FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. DEASSIGN BREAKLINE_FC 2:6 will result in the deassignment of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIANG any valid feature code group names may be used as arguments to the DEASSIGN BREAKLINE_FC command.

DESCRIPTION:

The DEASSIGN BREAKLINE_FC command complements the ASSIGN BREAKLINE_FC command. DEASSIGN BREAKLINE_FC enables the user to remove specified feature codes from the list of feature codes that have been assigned for interpretation as breaklines.

By default no feature codes are assigned for breaklines, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the DEASSIGN BREAKLINE_FC command:

*** WARNING *** You must have read an FRT file to be able to use group names
*** WARNING *** No groups have been defined in the FRT
*** WARNING *** Illegal feature code 'integer'
*** WARNING *** Bad group name 'group-name'

Examples:

```
TRIANG> DEASSIGN BREAKLINE_FC 6:9,CLIFFS,126<CR>
TRIANG>
```

DEASSIGN BREAKLINE_LAYER

Deassigns specified IFF layers from the list of those assigned for use as breaklines within the triangulation.

FORMAT: DEASSIGN BREAKLINE_LAYER layer[,...]

Command parameters:

layer

An IFF layer number which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. DEASSIGN BREAKLINE_LAYER 2:6 will result in the deassignment of layers 2,3,4,5 and 6.

DESCRIPTION:

The DEASSIGN BREAKLINE_LAYER command complements the ASSIGN BREAKLINE_LAYER command. DEASSIGN BREAKLINE_LAYER enables the user to remove specified layers from the list of layers that have been assigned for interpretation as breaklines.

By default no layers are assigned for breaklines, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the DEASSIGN BREAKLINE_LAYER command:

*** WARNING *** Too many layer arguments in one command
*** WARNING *** Illegal layer number 'integer'

Examples:

TRIANG> **DEASSIGN BREAKLINE_LAYER 21:29,126<CR>**
TRIANG>

DEASSIGN CLIFF_FC

Deassigns specified IFF feature codes from the list of feature codes identifying cliffs to be included in the triangulation.

FORMAT: DEASSIGN CLIFF_FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. DEASSIGN CLIFF_FC 2:6 will result in the deassignment of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIANG any valid feature code group names may be used as arguments to the DEASSIGN CLIFF_FC command.

DESCRIPTION:

The DEASSIGN CLIFF_FC command complements the ASSIGN CLIFF_FC command. DEASSIGN CLIFF_FC enables the user to remove specified feature codes from the list of feature codes that have been assigned for interpretation as cliffs.

For information about digitising and the internal treatment of clifflines, see the main Description section above.

By default no feature codes are assigned for cliffs, breaklines, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the DEASSIGN CLIFF_FC command:

*** WARNING *** You must have read an FRT file to be able to use group names
*** WARNING *** No groups have been defined in the FRT
*** WARNING *** Illegal feature code 'integer'
*** WARNING *** Bad group name 'group-name'

Examples:

```
TRIANG> DEASSIGN CLIFF_FC 6:9,WATER,126<CR>
TRIANG>
```

DEASSIGN CLIFF_LAYER

Deassigns specified IFF layers from the list of those assigned for use as cliffs within the triangulation.

FORMAT: DEASSIGN CLIFF_LAYER layer[,...]

Command parameters:

layer

An IFF layer number which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. DEASSIGN CLIFF_LAYER 2:6 will result in the deassignment of layers 2,3,4,5 and 6.

DESCRIPTION:

The DEASSIGN CLIFF_LAYER command complements the ASSIGN CLIFF_LAYER command. DEASSIGN CLIFF_LAYER enables the user to remove specified layers from the list of layers that have been assigned for interpretation as cliffs during triangulation formation.

For information about digitising and the internal treatment of clifflines, see the main Description section above.

By default no layers are assigned for cliffs, breaklines, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the DEASSIGN CLIFF_LAYER command:

*** WARNING *** Too many layer arguments in one command
*** WARNING *** Illegal layer number 'integer'

Examples:

TRIANG> **DEASSIGN CLIFF_LAYER 21:29,126<CR>**
TRIANG>

DEASSIGN RIDGE_FC

Deassigns specified IFF feature codes from the list of feature codes identifying ridgelines to be included in the triangulation.

FORMAT: DEASSIGN RIDGE_FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. DEASSIGN RIDGE_FC 2:6 will result in the deassignment of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIANG any valid feature code group names may be used as arguments to the DEASSIGN RIDGE_FC command.

DESCRIPTION:

The DEASSIGN RIDGE_FC command complements the ASSIGN RIDGE_FC command. DEASSIGN RIDGE_FC enables the user to remove specified IFF feature codes from the list of those assigned for use as ridgelines within the triangulation.

By default no feature codes are assigned for breaklines, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the DEASSIGN FC command:

*** WARNING *** You must have read an FRT file to be able to use group names
*** WARNING *** No groups have been defined in the FRT
*** WARNING *** Illegal feature code 'integer'
*** WARNING *** Bad group name 'group-name'

Examples:

TRIANG> **DEASSIGN RIDGE_FC 6:9,WATER,126<CR>**
TRIANG>

DEASSIGN RIDGE_LAYER

Deassigns specified IFF layers from the list of layers identifying ridgelines to be included in the triangulation.

FORMAT: DEASSIGN RIDGE_LAYER layer[,...]

Command parameters:

layer

An IFF layer which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. DEASSIGN RIDGE_LAYER 2:6 will result in the deassignment of layers 2,3,4,5 and 6.

DESCRIPTION:

The DEASSIGN RIDGE_LAYER command complements the ASSIGN RIDGE_LAYER command. DEASSIGN RIDGE_LAYER enables the user to remove specified layers from the list of layers that have been assigned for interpretation as ridgelines during triangulation formation.

By default no layers are selected for breaklines, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the DEASSIGN RIDGE_LAYER command:

*** WARNING *** Too many layer arguments in one command
*** WARNING *** Illegal layer number 'integer'

Examples:

TRIANG> **DEASSIGN RIDGE_LAYER 21:29,126<CR>**
TRIANG>

DEASSIGN RIVER_FC

Deassigns specified IFF feature codes from the list of feature codes identifying rivers to be included in the triangulation.

FORMAT: DEASSIGN RIVER_FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. DEASSIGN RIVER_FC 2:6 will result in the deassignment of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIANG any valid feature code group names may be used as arguments to the DEASSIGN RIVER_FC command.

DESCRIPTION:

The DEASSIGN RIVER_FC command complements the ASSIGN RIVER_FC command. DEASSIGN RIVER_FC enables the user to remove specified IFF feature codes from the list of those assigned for use as rivers within the triangulation.

By default no feature codes are assigned for breaklines, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the DEASSIGN FC command:

*** WARNING *** You must have read an FRT file to be able to use group names
*** WARNING *** No groups have been defined in the FRT
*** WARNING *** Illegal feature code 'integer'
*** WARNING *** Bad group name 'group-name'

Examples:

TRIANG> **DEASSIGN RIVER_FC 6:9,WATER,126<CR>**
TRIANG>

DEASSIGN RIVER_LAYER

Deassigns specified IFF layers from the list of layers containing rivers to be included in the triangulation.

FORMAT: DEASSIGN RIVER_LAYER layer[,...]

Command parameters:

layer

An IFF layer which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. DEASSIGN RIVER_LAYER 2:6 will result in the deassignemnt of layers 2,3,4,5 and 6.

DESCRIPTION:

The DEASSIGN RIVER_LAYER command complements the ASSIGN RIVER_LAYER command. DEASSIGN RIVER_LAYER enables the user to remove specified layers from the list of layers that have been selected for interpretation as rivers during triangulation formation.

By default no layers are assigned for breaklines, rivers etc. and all IFF data that lie within the WINDOW bounds are included in the triangulation.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the DEASSIGN RIVER_LAYER command:

*** WARNING *** Too many layer arguments in one command
*** WARNING *** Illegal layer number 'integer'

Examples:

TRIANG> **DEASSIGN RIVER_LAYER 21:29,126<CR>**
TRIANG>

DESELECT FC

Deselects specified IFF feature codes. Any IFF features having the specified feature codes will be excluded from the triangulation.

FORMAT: DESELECT FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. DESELECT FC 2:6 will result in the deselection of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIANG any valid feature code group names may be used as arguments to the DESELECT FC command.

DESCRIPTION:

The DESELECT FC command complements the SELECT FC command. DESELECT FC enables the user to prevent TRIANG from reading in any IFF features which have the specified feature codes.

On program startup all FCs are selected for input.

Specific FC selections may then be made with the SELECT FC command. Only the specified FCs will be used for input. All FCs not explicitly specified in a SELECT FC command will be then excluded from input.

ALL FC and layer selections are cancelled by the SELECT ALL command; i.e. all layers and FCs are reselected for input.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers.

Messages:

The following warning messages are specific to the DESELECT FC command:

*** WARNING *** You must have read an FRT file to be able to use group names
*** WARNING *** No groups have been defined in the FRT
*** WARNING *** Illegal feature code 'integer'
*** WARNING *** Bad group name 'group-name'

Examples:

```
TRIANG> DESELECT FC 6:9,WATER,126<CR>
TRIANG>
```

DESELECT LAYER

Deselects specified IFF layers. Any IFF features lying within the specified layers will be excluded from the triangulation.

FORMAT: DESELECT LAYER layer[,...]

Command parameters:

layer

An IFF layer number must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. DESELECT LAYER 2:6 will result in the deselection of layers 2,3,4,5 and 6.

DESCRIPTION:

The DESELECT LAYER command complements the SELECT LAYER command. DESELECT LAYER enables the user to prevent TRIANG from reading in any IFF features which lie within the specified layers.

The DESELECT LAYER command complements the SELECT LAYER command. DESELECT LAYER enables the user to prevent TRIANG from reading in any IFF features which lie within the specified layers.

On program startup all layers are selected for input.

Specific layer selections may then be made with the SELECT FC command. Only the specified layers will be used for input. All layers not explicitly specified in SELECT LAYER commands will be then excluded from input.

ALL FC and layer selections are cancelled by the SELECT ALL command; i.e. all layers and FCs are reselected for input.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. The use can use the SHOW SELECTIONS command to examine current input selections and SHOW ASSIGNMENTS to examine current assignments.

Messages:

The following warning messages are specific to the DESELECT LAYER command:

*** WARNING *** Too many layer arguments in one command
*** WARNING *** Illegal layer number 'integer'

Examples:

TRIANG> **DESELECT LAYER 21:29,126<CR>**
TRIANG>

DISABLE CONSTRAINT

Disables the constraint of a Delaunay triangulation to input string paths.

FORMAT: DISABLE CONSTRAINT

Command parameters: None.

DESCRIPTION:

TRIANG always seeks to produce a Delaunay triangulation. Unfortunately a triangulation which fits the Delaunay ideal may not necessarily honour the paths of the string input from IFF file. The constraint option ensures that triangulation never cuts through through any string features.

The CONSTRAINT option is applied by default and must be explicitly disabled with the DISABLE CONSTRAINT command if an unmodified Delaunay triangulation is required.

The current status of the CONSTRAINT option can be examined using the SHOW ENABLE command.

Messages: None.

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:30:39 20-AUG-87
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... Off
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG> DISABLE CONSTRAINT<CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... Off
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... OFF
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
```

```
INVERSE ..................................................... Off
MULTIPLY .................................................... Off
PME ......................................................... Off
SQUARE ...................................................... On
TOFEET ...................................................... Off
TOMETRES .................................................... Off
TRIANG>
```

DISABLE DIAGNOSTICS

Disables a previous ENABLE DIAGNOSTICS command.

FORMAT: DISABLE DIAGNOSTICS

Command parameters: None.

DESCRIPTION:

DISABLE DIAGNOSTICS allows the user to disable a previous ENABLE DIAGNOSTICS command.

The current status of the DIAGNOSTICS option can be examined using the SHOW ENABLE command.

Messages: None.

Examples:

```
TRIANG> ENABLE DIAGNOSTICS<CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... On
DIVIDEBY ....................................................... Off
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG> DISABLE DIAGNOSTICS<CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... Off
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG>
```

DISABLE DIVIDEBY

Disables a previous ENABLE DIVIDEBY command.

FORMAT: DISABLE DIVIDEBY

Command parameters: None.

DESCRIPTION:

DISABLE DIVIDEBY allows the user to disable a previous ENABLE DIVIDEBY command.

The current status of the DIVIDEBY option can be examined using the SHOW ENABLE or SHOW HEIGHTS commands.

Messages: None.

Examples:

```
TRIANG> SHOW DIVIDEBY<CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... Off
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG> ENABLE DIVIDEBY 39.2<CR>
TRIANG> SHOW DIVIDEBY<CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... On
Incoming heights to be divided by ..... 39.2
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG>
```

DISABLE GRAPHICS

Disables any previous ENABLE GRAPHICS command.

FORMAT: DISABLE GRAPHICS

Command parameter: None.

DESCRIPTION:

The DISABLE GRAPHICS command cancels the effect of a previous ENABLE GRAPHICS command.

The current status of the GRAPHICS option can be examined using the SHOW ENABLE command.

Messages: None.

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:30:39 20-AUG-87
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... Off
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG> ENABLE GRAPHICS<CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... Off
GRAPHICS ....................................................... On
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG>
```

DISABLE INTEGER_HEIGHT

DISABLE INTEGER_HEIGHT disables the effect of a previous ENABLE INTEGER_HEIGHT command.

FORMAT: DISABLE INTEGER_HEIGHT

Command parameters: None.

DESCRIPTION:

In IFF files height values are transmitted via AC (Ancillary Code) or ZS (3D string) entries. By default contour and spot height Z values are read from type 3 ACs as floating point values. By use of the ENABLE INTEGER_HEIGHT command, integer heights may be read from type 2 ACs.

The DISABLE INTEGER_HEIGHT command disables the effect of a previous ENABLE INTEGER_HEIGHT command and heights will be read from type 3 (floating point) ACs.

The current status of the INTEGER_HEIGHT option can be examined using the SHOW ENABLE or SHOW HEIGHTS commands.

Messages: None.

Examples:

```
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... On
Incoming heights to be divided by ..............................  4.900
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG> ENABLE INTEGER_HEIGHT<CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... On
Incoming heights to be divided by ..............................  4.900
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. On
(Incoming IFF heights expected in type 2 AC entries)
```

```
INVERSE ..................................................... Off
MULTIPLY ..................................................  Off
PME ......................................................... Off
SQUARE ...................................................... On
TOFEET ...................................................... Off
TOMETRES .................................................... Off
TRIANG>
```

DISABLE INVERSE

Disables a previous ENABLE INVERSE command.

FORMAT: DISABLE INVERSE

Command parameters: None.

DESCRIPTION:

DISABLE INVERSE allows the user to disable a previous ENABLE INVERSE command. By default heights are not inverted.

The current status of the INVERSE option can be examined using the SHOW ENABLE or SHOW HEIGHTS commands.

Messages: None.

Examples:

```
TRIANG> ENABLE INVERSE <CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... Off
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ On
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG>
```

DISABLE MULTIPLYBY

Disables a previous ENABLE MULTIPLYBY command.

FORMAT: DISABLE MULTIPLYBY

Command parameters: None.

DESCRIPTION:

DISABLE MULTIPLYBY allows the user to disable a previous ENABLE MULTIPLYBY command.

The current status of the MULTIPLYBY option can be examined using the SHOW ENABLE or SHOW HEIGHTS commands.

Messages: None.

Examples:

TRIANG> **ENABLE MULTIPLYBY 39.2<CR>**
TRIANG>

DISABLE PME

DISABLE PME disables the effect of a previous ENABLE PME command.

FORMAT: DISABLE PME

Command parameters: None.

DESCRIPTION:

The ENABLE PME and DISABLE PME commands are reserved for Laser-Scan use. PME is a code optimisation tool and should be invoked by LSL software personnel only.

DISABLE PME disables the effect of a previous ENABLE PME command and causes the PME_EXIT routine to be invoked.

The current status of the PME option can be examined using the SHOW ENABLE command.

Message:

The following warning message is specific to the DISABLE PME command:

*** WARNING *** You were not using PME anyway!

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:30:39 20-AUG-87
TRIANG> ENABLE PME<CR>
TRIANG> WINDOW 0.  0.  800.  760.<CR>
TRIANG> ZLIMITS 0 765<CR>
TRIANG> MAXPOINTS 78000<CR>
TRIANG> FILEOUT CHARITY<CR>
.DTA file DUA3:[DEMONSTRATION]CHARITY.DTA;6 opened for write
.NOD file DUA3:[DEMONSTRATION]CHARITY.NOD;6 opened for write
TRIANG> FRT CHASTITY<CR>
FRT file LSL$FRT:CHASTITY.FRT;8 opened for read
TRIANG> FILEIN FAITH<CR>
IFF file LSL$IF:FAITH.IFF;0 opened for read
TRIANG> ASSIGN BREAKLINE_FC 1:7,WATER,80,102<CR>
TRIANG> FILEIN ATILLA<CR>
TRIANG> DISABLE PME<CR>
TRIANG> GO<CR>
 ELAPSED:  00:05:25.84 CPU:  0:00:05.71 BUFIO:  281 DIRIO:  46 FAULTS:  263
$
```

DISABLE SQUARE

The DISABLE SQUARE command disables the default TRIANG squaring test for IFF control points.

FORMAT: DISABLE SQUARE

Command parameters: None.

DESCRIPTION:

By default TRIANG checks that the control points in an IFF input file are square to the map coordinate system. If the map held in the IFF file is not square to the coordinate system axes there is a danger that some rows or columns of the DTM will be missed out or be incomplete as DTMCREATE uses only axis minima and maxima to define the model area. If the IFF control point values are not square and the SQUARE option is active then an error message is issued and the run aborted.

The DISABLE SQUARE command disables the squaring test for IFF control points.

The current status of the SQUARE option can be examined using the SHOW ENABLE command.

Messages: None.

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:30:39 20-AUG-87
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... Off
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG> DISABLE SQUARE<CR>
TRIANG>
```

DISABLE TOFEET

Cancels the effect of a previous ENABLE TOFEET command.

FORMAT: DISABLE TOFEET

Command parameters: None.

DESCRIPTION:

It is possible that different input files may have heights recorded in different measurement systems. The model must be relative to one system only. Two preset height conversion options are available: ENABLE TOMETRES and ENABLE TOFEET.

The ENABLE TOFEET command enables the conversion of heights held in the IFF file in metres to feet. It has the same effect as an explicit ENABLE MULTIPLYBY 3.2808455 command. It is possible to read in one file with heights in feet with one FILEIN command and then use the ENABLE TOFEET command and read in another file with heights in metres converting to feet during read-in. The DTM will then be produced in feet.

If one of the height modification options is enabled using ENABLE MULTIPLYBY, ENABLE TOFEET etc., you must give the ZLIMITS in the target measurement system or height range (ie feet for the ENABLE TOFEET command). Failure to do this may result in flattening of all model relief!.

N.B. DISABLE TOFEET will not cancel an explicit ENABLE MULTIPLYBY 3.2808455 command.

The current status of the TOFEET option can be examined using the SHOW ENABLE or SHOW HEIGHTS commands.

Messages: None.

Examples:

TRIANG> **ENABLE TOFEET<CR>**
TRIANG>

DISABLE TOMETRES

Cancels the effect of a previous ENABLE TOMETRES command.

FORMAT: DISABLE TOMETRES

Command parameters: None.

DESCRIPTION:

It is possible that different input files may have heights recorded in different measurement systems. The model must be relative to one system only. Two height conversion options are available: ENABLE TOFEET and ENABLE TOMETRES.

The ENABLE TOMETRES command results in the conversion of heights held in the IFF file in feet to metres. It has the same effect as an explicit ENABLE DIVIDEBY 3.2808455 command. It is possible to read in one file with heights in feet with one FILEIN command and then use the ENABLE TOMETRES command and read in another file with heights in feet converting to metres during read-in. The DTM will then be produced in metres.

If one of the height modification options is enabled using ENABLE MULTIPLYBY, ENABLE TOMETRES etc., you must give the ZLIMITS in the target measurement system or height range (ie metres for the ENABLE TOMETRES command). Failure to do this may result in flattening of all model relief.

N.B. DISABLE TOMETRES will not cancel an explicit ENABLE DIVIDEBY 3.2808455 command.

The current status of the TOMETRES option can be examined using the SHOW ENABLE or SHOW HEIGHTS commands.

Messages: None.

Examples:

TRIANG> **ENABLE TOMETRES<CR>**
TRIANG>

ENABLE CONSTRAINT

Enables the constraint of a Delaunay triangulation to input string paths.

FORMAT: ENABLE CONSTRAINT

Command parameters: None.

DESCRIPTION:

TRIANG always seeks to produce a Delaunay triangulation. Unfortunately a triangulation which fits the Delaunay ideal may not necessarily honour the paths of the string input from IFF file. The constraint option ensures that triangulation never cuts through through any string features.

The CONSTRAINT option is applied by default and must be explicitly disabled with the DISABLE CONSTRAINT command if an unmodified Delaunay triangulation is required.

The current status of the CONSTRAINT option can be examined using the SHOW ENABLE command.

Messages: None.

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:30:39 20-AUG-87
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... Off
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG> DISABLE CONSTRAINT<CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... Off
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... Off
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
```

```
INVERSE ..................................................... Off
MULTIPLY .................................................... Off
PME ......................................................... Off
SQUARE ...................................................... On
TOFEET ...................................................... Off
TOMETRES .................................................... Off
TRIANG>
```

ENABLE DIAGNOSTICS

ENABLE DIAGNOSTICS allows the user to enable diagnostic printout.

FORMAT: ENABLE DIAGNOSTICS

Command parameters: None.

DESCRIPTION:

ENABLE DIAGNOSTICS allows the user to enable diagnostic printout.

Because it is usually used in a batch processing environment, by default TRIANG produces minimal diagnostic printout. If however, the user wishes to receive indications of processing progress and of the effect of input data assignments and selections on data input, diagnostic printout may be selected with the ENABLE DIAGNOSTICS command.

It should be noted that if DIAGNOSTICS are enabled, TRIANG can produce voluminous printout, particularly if used during the input phase in conjunction with selections from IFF files.

On a heavily loaded computer it may be reassuring to ENABLE DIAGNOSTICS for the triangulation stage of TRIANG processing to indicate progress through the data set. If SYS$OUTPUT is directed to a video screen terminal, messages indicating percentage progress are issued.

The current status of the DIAGNOSTICS option can be examined using the SHOW ENABLE command.

Messages: None.

Examples:

```
TRIANG> ENABLE DIAGNOSTICS<CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... On
DIVIDEBY ....................................................... OFF
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG>
```

ENABLE DIVIDEBY

ENABLE DIVIDEBY allows the user to enable division of input file heights by a specified floating point constant.

FORMAT: ENABLE DIVIDEBY denominator

Command parameters:

denominator

The value by which all input file heights are to be divided.

DESCRIPTION:

The ENABLE DIVIDEBY enables the user to divide all incoming heights by a specified (floating point) constant. For example, the command ENABLE DIVIDEBY 2.0 will cause all incoming heights to be divided by 2.0. An ENABLE DIVIDEBY 3.2808455 command has the same effect as an ENABLE TOMETRES command.

The current status of the DIVIDEBY option can be examined using the SHOW ENABLE or SHOW HEIGHTS commands.

Messages:

The following messages are specific to the ENABLE DIVIDEBY command:

*** WARNING *** You are already planning to multiply by 'constant'
ENABLE DIVIDEBY command now overrides ENABLE MULTIPLYBY command
*** WARNING *** You must specify a value for DIVIDEBY

Examples:

TRIANG> **ENABLE DIVIDEBY 39.2<CR>**
TRIANG>

ENABLE GRAPHICS

Enable TRIANG graphics output.

FORMAT: ENABLE GRAPHICS

Command parameters: None.

DESCRIPTION:

TRIANG offers the option to generate graphic output to indicate processing progress. By default graphic output is disabled. To prevent a user selecting graphics when it is inappropriate to the current terminal, TRIANG uses a lookup table of terminal characteristics associated with all available terminal lines (see Appendix 1). An invalid graphics selection will result in a warning message and the default NO GRAPHICS option being selected.

Graphics selection may be cancelled with the DISABLE GRAPHICS command.

The current status of the GRAPHICS option can be examined using the SHOW ENABLE command.

Messages:

The following messages are specific to the ENABLE GRAPHICS command:

*** ERROR *** reading lookup file at line 'integer'
*** WARNING *** Unable to open "LSL$LOOKUP:TERMTYPE.DAT"
Sorry 'name' terminal 'terminal-ident' isn't in the lookup table
Sorry 'name' terminal 'terminal-ident' can't support graphics
*** ERROR *** translating logical name LSL$DTMCREATETERMINAL

Examples:

$ **TRIANG<CR>**
DTMCREATE module TRIANG of 13:30:39 20-AUG-87
TRIANG> **SHOW ENABLE<CR>**
CONSTRAINT ..... On
DIAGNOSTICS ..... Off
DIVIDEBY ..... OFF
GRAPHICS ..... Off
INTEGER_HEIGHT ..... Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ..... Off
MULTIPLY ..... Off
PME ..... Off
SQUARE ..... On
TOFEET ..... Off
TOMETRES ..... Off
TRIANG> **ENABLE GRAPHICS<CR>**

```
TRIANG>
```

ENABLE INTEGER_HEIGHT

Take feature heights from type 2 (integer) AC (Ancillary Code) entries in an IFF input file.

FORMAT: ENABLE INTEGER_HEIGHT

Command parameters: None.

DESCRIPTION:

In IFF files, height values are transmitted via AC (Ancillary Code) or ZS (3D string) entries. By default contour and spot height Z values are read from type 3 ACs as floating point values. By use of the ENABLE INTEGER_HEIGHT command, integer heights may be read from type 2 ACs.

ENABLE INTEGER_HEIGHT causes TRIANG to take heights from type 2 (integer) AC (Ancillary Code) entries in an IFF input file.

The current status of the INTEGER_HEIGHT option can be examined using the SHOW ENABLE or SHOW HEIGHTS commands.

Messages: None.

Examples:

```
TRIANG> ENABLE INTEGER_HEIGHT<CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... Off
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... OFF
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. On
(Incoming IFF heights expected in type 2 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG>
```

ENABLE INVERSE

Enable height inversion.

FORMAT: ENABLE INVERSE

Command parameters: None.

DESCRIPTION:

Incoming IFF heights may be inverted if the ENABLE INVERSE command is specified prior to reading an IFF file with a FILEIN command. This enables modelling of hydrographic data where sea depths are often stored as positive heights, drying zone heights as negative and land heights as positive!

By default heights are not inverted.

If the DATUM command has been used to specify a change of height datum the heights are inverted before the datum value is added.

The current status of the INVERSE option can be examined using the SHOW ENABLE or SHOW HEIGHTS commands.

Messages: None.

Examples:

```
TRIANG> DATUM 8.2<CR>
TRIANG> ENABLE INVERSE <CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... Off
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... Off
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. On
(Incoming IFF heights expected in type 2 AC entries)
INVERSE ........................................................ On
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG> FILEIN HUNSTANTON_BATHEMETRY<CR>
IFF file LSL$IF:HUNSTANTON_BATHEMETRY.IFF;0 opened for read


+-------------------------------------------------------------+
|                                                             |
|               Starting pass through IFF file                |
|                                                             |
```

```
+-------------------------------------------------------------+


There are now 671 points in the DTM area
TRIANG> DATUM 0.0<CR>
TRIANG> DISABLE INVERSE <CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... Off
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... OFF
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. On
(Incoming IFF heights expected in type 2 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG> FILEIN ATILLA<CR>
IFF file LSL$IF:ATILLA.IFF;0 opened for read


+-------------------------------------------------------------+
|                                                             |
|               Starting pass through IFF file                |
|                                                             |
+-------------------------------------------------------------+


There are now 55527 points in the DTM area
TRIANG>
```

ENABLE MULTIPLYBY

ENABLE MULTIPLYBY allows the user to enable multiplication of input file heights by a specified floating point constant.

FORMAT: ENABLE MULTIPLYBY multiplicand

Command parameters:

multiplicand

The value by which all input file heights are to be multiplied.

DESCRIPTION:

The ENABLE MULTIPLYBY enables the user to multiply all incoming heights by a specified (floating point) constant. For example, the command ENABLE MULTIPLYBY 2.0 will cause all incoming heights to be multiplied by 2.0. An ENABLE MULTIPLYBY 3.2808455 command has the same effect as an ENABLE TOFEET command.

The current status of the MULTIPLYBY option can be examined using the SHOW ENABLE or SHOW HEIGHTS commands.

Messages:

The following messages are specific to the ENABLE MULTIPLYBY command:

```
*** WARNING *** You are already planning to divide by 'constant'
                ENABLE MULTIPLY command now overrides ENABLE DIVIDEBY command
*** WARNING *** You must specify a value for MULTIPLYBY
```

Examples:

```
TRIANG> ENABLE MULTIPLYBY 39.2<CR>
TRIANG>
```

ENABLE PME

ENABLE PME enables the PME performance monitor.

FORMAT: ENABLE PME

Command parameters: None.

DESCRIPTION:

The ENABLE PME and DISABLE PME commands are reserved for Laser-Scan use. PME is a code optimisation tool and should be invoked by LSL software personnel only.

ENABLE PME causes the PME_INIT routine to be invoked.

The current status of the PME option can be examined using the SHOW PME command.

Message:

The following warning message is specific to the ENABLE PME command:

*** WARNING *** You are already using PME!

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:30:39 20-AUG-87
TRIANG> ENABLE PME<CR>
TRIANG> ENABLE DIAGNOSTICS<CR>
TRIANG> WINDOW 0.  0.  800.  760.<CR>
TRIANG> ZLIMITS 0 765<CR>
TRIANG> MAXPOINTS 78000<CR>
TRIANG> FILEOUT CHARITY<CR>
.DTA file DUA3:[DEMONSTRATION]CHARITY.DTA;6 opened for write
.NOD file DUA3:[DEMONSTRATION]CHARITY.NOD;6 opened for write
TRIANG> FRT CHASTITY<CR>
FRT file LSL$FRT:CHASTITY.FRT;8 opened for read
TRIANG> FILEIN FAITH<CR>
TRIANG> ASSIGN BREAKLINE_FC 1:7,WATER,80,102<CR>
TRIANG> FILEIN ATILLA<CR>
IFF file LSL$IF:ATILLA.IFF;0 opened for read
TRIANG> DISABLE PME<CR>
TRIANG> GO<CR>
 ELAPSED:  00:05:25.84 CPU:  0:00:05.71 BUFIO:  281 DIRIO:  46 FAULTS:  263
$
```

ENABLE SQUARE

The ENABLE SQUARE command enables a squaring test for IFF control points.

FORMAT: ENABLE SQUARE

Command parameters: None.

DESCRIPTION:

By default TRIANG checks that the control points in an IFF input file are square to the map coordinate system. If the map held in the IFF file is not square to the coordinate system axes there is a danger that some rows or columns of the DTM will be missed out or be incomplete as DTMCREATE uses only axis minima and maxima to define the model area. If the IFF control point values are not square and the SQUARE option is active then an error message is issued and the run aborted.

The ENABLE SQUARE command enables the squaring test for IFF control points.

The current status of the SQUARE option can be examined using the SHOW ENABLE command.

Messages: None.

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:30:39 20-AUG-87
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... OFF
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG> DISABLE SQUARE<CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... Off
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... OFF
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
```

```
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ..................................................... Off
MULTIPLY .................................................... Off
PME ......................................................... Off
SQUARE ...................................................... Off
TOFEET ...................................................... Off
TOMETRES .................................................... Off
TRIANG>
```

ENABLE TOFEET

The ENABLE TOFEET command enables the conversion of input file heights from metres to feet.

FORMAT: ENABLE TOFEET

Command parameters: None.

DESCRIPTION:

It is possible that different input files may have heights recorded in different measurement systems. The model must be relative to one system only. Two height conversion options are available: ENABLE TOMETRES and ENABLE TOFEET.

The ENABLE TOFEET command enables the conversion of input file heights from metres to feet. It has the same effect as an explicit ENABLE MULTIPLYBY 3.2808455 command. It is possible to read in one file with heights in feet with one FILEIN command and then use the ENABLE TOFEET command and read in another file with heights in metres converting to feet during read-in. The DTM will then be produced in feet.

If one of the height modification options is enabled using ENABLE MULTIPLYBY, ENABLE TOFEET etc., you must give the ZLIMITS in the target measurement system or height range (ie feet for the ENABLE TOFEET command). Failure to do this may result in flattening of all model relief!.

The current status of the TOFEET option can be examined using the SHOW ENABLE or SHOW HEIGHTS commands.

Messages: None.

Examples:

TRIANG> **ENABLE TOFEET<CR>**
TRIANG>

ENABLE TOMETRES

The ENABLE TOMETRES command enables the conversion of input file heights from feet to metres.

FORMAT: ENABLE TOMETRES

Command parameters: None.

DESCRIPTION:

It is possible that different input files may have heights recorded in different measurement systems. The model must be relative to one system only. Two height conversion options are available: ENABLE TOFEET and ENABLE TOMETRES.

The ENABLE TOMETRES command results in the conversion of heights held in the IFF file in feet to metres. It has the same effect as an explicit ENABLE DIVIDEBY 3.2808455 command. It is possible to read in one file with heights in metres with one FILEIN command and then use the ENABLE TOMETRES command and read in another file with heights in feet converting to metres during read-in. The DTM will then be produced in metres.

If one of the height modification options is enabled using ENABLE MULTIPLYBY, ENABLE TOMETRES etc., you must give the ZLIMITS in the target measurement system or height range (ie metres for the ENABLE TOMETRES command). Failure to do this may result in flattening of all model relief.

The current status of the TOMETRES option can be examined using the SHOW ENABLE or SHOW HEIGHTS commands.

Messages: None.

Examples:

TRIANG> **ENABLE TOMETRES<CR>**
TRIANG>

FILEIN

Specifies a file that is to be opened and used for data input.

FORMAT: FILEIN file-spec

COMMAND PARAMETERS:

file-spec

The specification of the file to be opened for data input.

Any parts of the file-spec not supplied for the FILEIN command will be taken from a default specification dependent on the current format specified with the FORMAT command.

If ENABLE FORMAT IFF (default) is specified then the default file specification 'LSL$IF:IFF.IFF;0' is used. If ENABLE FORMAT DTI is specified then the default file specification 'LSL$DTI:DTI.DTI;0' is used.

DESCRIPTION:

The FILEIN command causes the specified file-spec to be opened and used as an input file to TRIANG. A FILEIN command cannot be issued until FILEOUT, ZLIMITS, MAXPOINTS and WINDOW commands have been given.

Messages:

The following messages are specific to the FILEIN command:

*** WARNING *** You must specify a file-spec argument to the FILEIN command
*** ERROR *** Unable to interpret input file-spec

Examples:

```
TRIANG> FILEIN DUA3:[DEMONSTRATION]IDAHO<CR>
IFF file DUA3:[DEMONSTRATION]IDAHO.IFF;1 opened for read
TRIANG> !  Now using the FORMAT command to change the format
TRIANG> !  of the input file from IFF (default) to DTI
TRIANG> FORMAT DTI
TRIANG> FILEIN AREA5
DTI file LSL$DTI:AREA5.DTI;0 opened for read
TRIANG>
```

FILEOUT

Specifies the file-spec to be used as the generic specification for the two TRIANG output file-specifications, one of which will be given the extension .NOD and the other .DTA.

FORMAT: FILEOUT file-spec

COMMAND PARAMETERS:

file-spec

The file-spec from which the generic name for the two TRIANG output files is to be taken.

All components of the supplied file-spec are used to form the output file specifications but with the substitution of the extensions .NOD and .DTA and version number ';0', i.e. latest version (shared by both files).

The default file-spec used to make up missing parts of the FILEIN file-spec parameter is dependent on the status of logical name LSL$DTMCREATE_WORK.

If logical name LSL$DTMCREATE_WORK is defined, DTMCREATE utilities translate the logical name to get the default file-spec for input and output of triangulation files. The logical name should be defined to provide a device and directory name only. The DTMCREATE programs themselves provide the default filename and extension fields of the specification. For example, a valid definition of logical name LSL$DTMCREATE_WORK is:

```
$ DEFINE LSL$DTMCREATE_WORK LSL$DATA_ROOT:[LSL.DTMCREATE]
```

This mechanism allows all DTMCREATE triangulation files to be stored in a central directory, rather than scattered in many different user directories. It thus mimics the use of logical names LSL$IF for IFF files and LSL$DTI for DTI files.

If the logical name is not defined, any parts of the file-spec not supplied for the FILEOUT command will be taken from the defaults 'SYS$DISK:[].NOD;0' and 'SYS$DISK:[].DTA;0'. These defaults result in the output files being created in your current default directory, set using the VMS SET DEFAULT or Laser-Scan SD commands.

DESCRIPTION:

The FILEOUT command causes the specified file-spec to be used as the generic name for the two TRIANG output files used to transfer the triangulation data structure between the DTMCREATE modules.

The FILEOUT command must be issued before the first FILEIN command.

Messages:

The following messages are specific to the FILEOUT command:

*** WARNING *** You must specify a file-spec argument to the FILEOUT command
*** ERROR *** reading output file-spec

Examples:

TRIANG> **FILEOUT DUA3:[DEMONSTRATION]IDAHO<CR>**
.DTA file DUA3:[DEMONSTRATION]IDAHO.DTA;6 opened for write
.NOD file DUA3:[DEMONSTRATION]IDAHO.NOD;6 opened for write
TRIANG>

FORMAT

Specifies the file format that is to be read into TRIANG using a FILEIN command.

FORMAT: FORMAT format

Command parameters:

format

This is a keyword chosen from 'DTI' or 'IFF' which signifies the type of file to be read.

DESCRIPTION:

The FORMAT command enables the user to specify the file format that is to be read into TRIANG during the data initialisation phase. The FORMAT command may be used swap between input file formats as often as required. The default file format is 'IFF'.

It is important to remember to issue the appropriate FORMAT command before the FILEIN command if the file format is different to that read in using a previous FILEIN command.

Messages:

The following warning messages are specific to the FORMAT command:

*** WARNING *** Unknown format - must be either DTI or IFF
*** WARNING *** You must specify an argument to the FORMAT command

---

Examples:

$ **TRIANG<CR>**
DTMCREATE module TRIANG of 13:30:39 20-AUG-87
TRIANG> **WINDOW 0. 0. 800. 760.<CR>**
TRIANG> **ZLIMITS 0 765<CR>**
TRIANG> **MAXPOINTS 78000<CR>**
TRIANG> **FILEOUT CHARITY<CR>**
.DTA file DUA3:[DEMONSTRATION]CHARITY.DTA;6 opened for write
.NOD file DUA3:[DEMONSTRATION]CHARITY.NOD;6 opened for write
TRIANG> **FORMAT DTI<CR>**
TRIANG> **FILEIN FAITH.DTI<CR>**

```
Origin?  1000.0 2000.0
TRIANG> FRT CHASTITY<CR>
FRT file LSL$FRT:CHASTITY.FRT;8 opened for read
TRIANG> FORMAT IFF<CR>
TRIANG> ASSIGN BREAKLINE_FC 1:7,WATER,80,102<CR>
TRIANG> FILEIN ATILLA<CR>
IFF file LSL$IF:ATILLA.IFF;0 opened for read
TRIANG> GO<CR>
 ELAPSED:  00:05:25.84 CPU:  0:00:05.71 BUFIO:  281 DIRIO:  46 FAULTS:  263
$
```

FRT

Specifies an FRT file which contains feature code group definitions.

FORMAT: FRT file-spec

Command parameters:

file-spec

The specification of an FRT (Feature Representation Table) file required to define feature code groups.

Missing parts from the FRT file-spec argument are taken from the default specification LSL$FRT:FRT.FRT;0.

DESCRIPTION:

The FRT command allows the user to specify an FRT file which contains feature code group definitions. The availability of feature code groups simplifies the specification of complex feature code assignments for breaklines etc.

Messages:

The following messages are specific to the FRT command:

*** ERROR *** reading FRT file-spec
*** ERROR *** unable to open specified FRT

Examples:

TRIANG> **ASSIGN CLIFF_FC OUTCROPS,7,COAST<CR>**
*** WARNING *** You must have read an FRT file to be able to use group names
TRIANG> **FRT HOVER<CR>**
FRT file LSL$FRT:HOVER.FRT;8 opened for read
TRIANG> **ASSIGN CLIFF_FC OUTCROPS,7,COAST<CR>**
TRIANG>

GO

Initiates the triangulation of the data read in using FILEIN commands.

FORMAT: GO

Command parameters: None.

DESCRIPTION:

When all necessary files have been read in the GO command will commence the triangulation process. Unless relatively small data-sets are being handled (say less then 50,000 data points) it is strongly recommended that TRIANG is run in batch mode at an off-peak time.

The GO command will cause TRIANG to first produce an ideal Delaunay triangulation. If the CONSTRAINT option is enabled TRIANG will then constrain the idealised triangulation to the paths of input data strings. When all triangulation creation and modification is complete TRIANG writes out the data structure to binary disk files and then exits.

Messages:

Once the GO command has been issued no more conversational messages of the *** WARNING *** format will be issued. Any messages will relate to serious processing problems and will normally result in abnormal TRIANG termination. The messages relating to non-interactive processing problems are presented at the end of this document.

Examples:

TRIANG> **GO<CR>**

ELAPSED: 0 00:05:21.82 CPU: 0:00:01.40 BUFIO: 51 DIRIO: 15 FAULTS: 170
$

HELP

Give help on a subject

FORMAT: HELP subject

Command parameters:

subject

The subject on which help is required

Description:

The HELP command looks the rest of the line up in the DTMCREATE HELP library. This library contains a brief summary of the operation of each command.

The information is looked up in the TRIANG section of the DTMCREATE help library, LSL$HELP:DTMCREATE.HLB.

Messages:

Where required, warning messages are output via the VMS LBR$OUTPUT_HELP utility.

Examples:

TRIANG> **HELP MAXPOINTS<CR>**

MAXPOINTS

The value supplied for the MAXPOINTS command determines the dimensioning of TRIANG internal workspace and the number of imaginary points placed around the edge of the triangulation. It is very likely that the user will have only a very rough idea of the amount of data expected within the window defined using the WINDOW command. This does not matter as only a rough estimate is needed. If ludicrously few points are estimated, say 10% of the actual data-set, then program running times will be significantly increased because the box data structure will not accurately reflect changes in data density. A rough figure, say 80% accurate, is better than nothing.

The number of points specified using the MAXPOINTS command must reflect the total from **ALL** the files input using FILEIN commands which lie within the triangulation window specified using the WINDOW command.

To calculate the number of points extracted from a DTI file determine how much of the DTI files lies within the TRIANG window, divide this by the DTI file x or y grid step as appropriate then multiply the resulting number of row and columns together. The DTI file size, coverage and x and y grid steps and Z-range may be simply determined by reading the file into MATRIX utility DTIEDIT.

The IMP utility IINFO can be used to determine how many points there are within an IFF file, in total, by feature code and by layer. It cannot, however, reveal the number of points that lie within a subwindow of the whole IFF file. The Z range of the IFF file may be determined using the same run of IINFO if the /HEIGHT qualifier is specified.

If there are more than the permitted number of points within the window then a smaller WINDOW should be specified and the data triangulated in two or more separate triangulations. Allow a small amount of overlap between adjacent windows, the sub-DTMs created by TRIGRID from these separate triangulations will then perfectly edge match when joined together using MATRIX utility DTITILE.

TRIANG>

MAXPOINTS

Specifies estimated number of input data points for workspace allocation.

FORMAT: MAXPOINTS estimate

Command parameters:

estimate

An estimate of the number of input points.

Description:

The value supplied for the MAXPOINTS command determines the dimensioning of TRIANG internal workspace and the number of imaginary points placed around the edge of the triangulation. It is very likely that the user will have only a very rough idea of the amount of data expected within the window defined using the WINDOW command. This does not matter as only a rough estimate is needed. If ludicrously few points are estimated, say 10% of the actual data-set, then program running times will be significantly increased because the box data structure will not accurately reflect changes in data density. A rough figure, say 80% accurate, is better than nothing.

The number of points specified using the MAXPOINTS command must reflect the total from **ALL** the files input using FILEIN commands which lie within the triangulation window specified using the WINDOW command.

To calculate the number of points extracted from a DTI file determine how much of the DTI files lies within the TRIANG window, divide this by the DTI file x or y grid step as appropriate then multiply the resulting number of row and columns together. The DTI file size, coverage and x and y grid steps and Z-range may be simply determined by reading the file into MATRIX utility DTIEDIT.

The IMP utility IINFO can be used to determine how many points there are within an IFF file, in total, by feature code and by layer. It cannot, however, reveal the number of points that lie within a subwindow of the whole IFF file. The Z range of the IFF file may be determined using the same run of IINFO if the /HEIGHT qualifier is specified.

If there are more than the permitted number of points within the window then a smaller WINDOW should be specified and the data triangulated in two or more separate triangulations. Allow a small amount of overlap between adjacent windows, the sub-DTMs created by TRIGRID from these separate triangulations will then perfectly edge match when joined together using MATRIX utility DTITILE.

Messages:

The following warning messages are specific to the MAXPOINTS command:

*** WARNING *** You can't change MAXPOINTS now as workspace allocation has begun
*** WARNING *** You must specify an integer argument to the MAXPOINTS command
*** WARNING *** Too many points. (Max. possible number without redimensioning = 'integer')
*** WARNING *** MAXPOINTS has not been set
*** WARNING *** Overrun of neighbour stack space may occur

Examples:

TRIANG> **MAXPOINTS 75000<CR>**
TRIANG> **SHOW MAXPOINTS<CR>**
Estimated number of incoming points = 75000
(Max possible number without redimensioning = 99730)
TRIANG>

PAUSE

Pauses TRIANG execution.

FORMAT: PAUSE

Command parameters: None.

DESCRIPTION:

Pauses TRIANG execution and issues a prompt for a carriage return to continue execution. This command is designed for use in software demonstration situations.

Messages: None.

Examples:

TRIANG> **PAUSE<CR>**

Press <RETURN> to continue**<CR>**
TRIANG>

QUIT

Quit from TRIANG.

FORMAT: QUIT

Command parameters: None.

Description:

The QUIT command causes TRIANG to exit immediately, closing all input files and closing and deleting all output files.

<CTRL/Z> (pressing the Ctrl and Z keys together) may also be used to quit from the program.

Messages: None.

Examples:

```
TRIANG> QUIT<CR>
 ELAPSED:  00:05:25.84 CPU:  0:00:05.71 BUFIO:  281 DIRIO:  46 FAULTS:  263
$
```

RETURN

Restores command input from an indirect file to SYS$COMMAND.

FORMAT: RETURN

Command parameters: None.

DESCRIPTION:

Restores command input from an indirect file to SYS$COMMAND.

A typical application is to allow the user to use an indirect command file to set up those run time defaults which are constant within a flowline and then return to input from the terminal (or batch stream) for the run specific commands. To do this RETURN must be the last command in the indirect command file.

Messages:

The following messages are specific to the RETURN command:

RETURN command detected - returning to terminal input
RETURN command ignored - command input is already from terminal

---

Examples:

```
TRIANG> @FLOW2<CR>
TRIANG> ENABLE DIAGNOSTICS
TRIANG> FRT FLOW2
FRT file LSL$FRT:FLOW2.FRT;8 opened for read
TRIANG> ASSIGN CLIFF_FC OUTCROPS,7,COAST
TRIANG> RETURN
TRIANG>
```

SELECT ALL

Resets all feature selections made with the SELECT and DESELECT commands.

FORMAT: SELECT ALL

Command parameters: None

DESCRIPTION:

This command resets all feature input selections. If features are subsequently selected using the other SELECT commands then all features are first implicitly deselected.

Messages:

The following message is specific to the SELECT command.

*** ERROR *** Specifying command SELECT
Command qualifiers are ALL, FC or LAYER

Examples:

TRIANG>**SELECT ALL <CR>**

At this point all features are selected.

TRIANG>**SELECT FC 7:10,56:78**

Here only features with the specified feature codes are selected

TRIANG>**SELECT FC 11:20**

At this point the specified feature codes are added to the currently selected features code, i.e. features with FC 7-20 and 56-78 are now selected.

TRIANG>

SELECT FC

Selects specified IFF feature codes for inclusion in the triangulation.

FORMAT: SELECT FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. SELECT FC 2:6 will result in the selection of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIANG any valid feature code group names may be used as arguments to the SELECT FC command.

DESCRIPTION:

The SELECT FC command complements the DESELECT FC command. SELECT FC enables the user to select any IFF features which have the specified feature codes.

On program startup all FCs are selected for input. IFF features which are not required for input must be specifically excluded using the appropriate DESELECT FC and DESELECT LAYER commands. For example, the command DESELECT FC 0:7 11:300 302:32767 will leave only features with FCs 8,9,10, and 301 selected for input.

The first SELECT FC command has the effect of deselecting all FCs from input except those explicitly specified as the arguments to the SELECT FC command. Subsequent SELECT FC commands have the effect of adding the specified FCs to the list of FCs selected for input.

All layers and FCs may be reselected for input by specifying the SELECT ALL command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the SELECT FC command:

*** WARNING *** You must have read an FRT file to be able to use group names
*** WARNING *** No groups have been defined in the FRT

*** WARNING *** Illegal feature code 'integer'
*** WARNING *** Bad group name 'group-name'

Examples:

TRIANG> **SELECT FC 6:9,WATER,126<CR>**
TRIANG>

SELECT LAYER

Selects specified IFF layers for inclusion in the triangulation.

FORMAT: SELECT LAYER layer[,...]

Command parameters:

layer

An IFF layer number must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. SELECT LAYER 2:6 will result in the selection of layers 2,3,4,5 and 6.

DESCRIPTION:

The SELECT LAYER command complements the DESELECT LAYER command. SELECT LAYER enables the user to select any IFF features which are contained within the specified layers.

By default TRIANG will input all features within an IFF file, regardless of the layer in which they lie.

On program startup all layers are selected for input. IFF features which are not required for input must be specifically excluded using the appropriate DESELECT FC and DESELECT LAYER commands. For example, the command DESELECT LAYER 0:7 11:300 302:32767 will leave only features which lie within layers 8,9,10, and 301 selected for input.

The first SELECT LAYER command has the effect of deselecting all layers from input except those explicitly specified as the arguments to the SELECT LAYER command. Subsequent SELECT LAYER commands have the effect of adding the specified layers to the list of FCs selected for input.

All layers and FCs may be reselected for input by specifying the SELECT ALL command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the SELECT LAYER command:

*** WARNING *** Too many layer arguments in one command

*** WARNING *** Illegal layer number 'integer'

Examples:

TRIANG> **SELECT LAYER 21:29,126<CR>**
TRIANG>

SHOW

Shows current status of TRIANG option and parameter settings.

FORMAT: SHOW subject

Command parameters:

subject

The subject that is to be displayed, chosen from:

ASSIGNMENTS	BREAKLINES	CLIFFLINES	DATUM	ENABLE
FC	FILES	FORMAT	FRT	HEIGHTS
LAYER	MAXPOINTS	RIDGELINES	RIVERS	SELECTIONS
UNITS	WINDOW	ZLIMITS		

DESCRIPTION:

SHOW enables the user to examine the current status of TRIANG options and parameter settings.

Messages:

TRIANG issues the following message if the SHOW command is specified without an argument:

*** ERROR *** Specifying command SHOW

Available SHOW command qualifiers are:

ASSIGNMENTS	BREAKLINES	CLIFFLINES	DATUM	ENABLE	FC
FILES	FORMAT	FRT	HEIGHTS	LAYER	
MAXPOINTS	RIDGELINES	RIVERS	SELECTIONS		
UNITS	WINDOW	ZLIMITS			

This feature can be used to advantage if the user wishes to quickly determine for which items the SHOW facility is available.

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:53:27  3-FEB-89
TRIANG> SHOW<CR>
*** ERROR *** Specifying command SHOW

Available SHOW command qualifiers are:

ASSIGNMENTS      BREAKLINES      CLIFFLINES     DATUM      ENABLE    FC

FILES            FORMAT          FRT            HEIGHTS    LAYER

MAXPOINTS        RIDGELINES      RIVERS         SELECTIONS

UNITS            WINDOW          ZLIMITS

TRIANG> SHOW BREAKLINES<CR>

BREAKLINES:
No layers assigned for breaklines
No feature codes assigned for breaklines

TRIANG>
```

SHOW ASSIGNMENTS

Shows current status of TRIANG feature code and layer assignments made using ASSIGN and DEASSIGN commands.

FORMAT: SHOW ASSIGNMENTS

Command parameters: None.

DESCRIPTION:

Shows current status of TRIANG feature code and layer assignments made using ASSIGN and DEASSIGN commands. The SHOW ASSIGNMENTS command has the effect of issuing SHOW BREAKLINES, SHOW CLIFFLINES, SHOW RIDGELINES and SHOW RIVERS commands sequentially.

Messages: None.

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:53:27  3-FEB-89
TRIANG> SHOW ASSIGNMENTS<CR>

BREAKLINES:
No layers assigned for breaklines
No feature codes assigned for breaklines

CLIFFLINES:
No layers assigned for cliffs
No feature codes assigned for cliffs

RIDGELINES:
No layers assigned for ridgelines
No feature codes assigned for ridgelines

RIVERS:
No layers assigned for rivers
No feature codes assigned for rivers

TRIANG>
```

SHOW BREAKLINES

Shows current status of TRIANG feature code and layer breakline assignments made using ASSIGN and DEASSIGN commands.

FORMAT: SHOW BREAKLINES

Command parameters: None.

DESCRIPTION:

Shows current status of TRIANG feature code and layer breakline assignments made using ASSIGN and DEASSIGN commands.

Messages: None.

Examples:

$ **TRIANG<CR>**
DTMCREATE module TRIANG of 13:53:27 3-FEB-89
TRIANG> **SHOW BREAKLINES<CR>**
No layers assigned for breaklines
No feature codes assigned for breaklines
TRIANG>

SHOW CLIFFLINES

Shows current status of TRIANG feature code and layer cliffline assignments made using ASSIGN and DEASSIGN commands.

FORMAT: SHOW CLIFFLINES

Command parameters: None.

DESCRIPTION:

Shows current status of TRIANG feature code and layer cliffline assignments made using ASSIGN and DEASSIGN commands.

Messages: None.

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:53:27  3-FEB-89
TRIANG> SHOW CLIFFLINES<CR>
No layers assigned for clifflines
No feature codes assigned for clifflines
TRIANG>
```

SHOW DATUM

Shows current status of of the height datum which is to be added to all incoming IFF and DTI heights.

FORMAT: SHOW DATUM

Command parameters: None.

DESCRIPTION:

Shows current status of of the height datum which is to be added to all incoming IFF and DTI heights.

Messages: None.

Examples:

$ **TRIANG<CR>**
DTMCREATE module TRIANG of 13:53:27 3-FEB-89
TRIANG> **SHOW DATUM<CR>**
Height datum 0.0 to be added to all incoming heights
TRIANG>

SHOW ENABLE

Shows current status of all TRIANG processing options.

FORMAT: SHOW ENABLE

Command parameters: None.

DESCRIPTION:

Shows current status of all TRIANG processing options.

Messages: None.

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:53:27  3-FEB-89
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... Off
DIVIDEBY ....................................................... OFF
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG>
```

SHOW FC

Shows current status of TRIANG feature code input selections.

FORMAT: **SHOW FC**

Command parameters: None.

DESCRIPTION:

Shows current status of TRIANG feature code input selections.

Messages: None.

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:53:27  3-FEB-89
TRIANG> SHOW FC<CR>
Feature codes selected for input:
0-32767
TRIANG>
```

SHOW FILES

Shows current status of TRIANG input file and output file useage.

FILES: SHOW FILES

Command parameters: None.

DESCRIPTION:

Shows current status of TRIANG input file FILES and output file useage.

SHOW FILES enables the user to determine whether he has opened the output files (by using the FILEOUT command) and what names have been used. It also enables the user to keep track of which IFF and DTI files have been successfully read into TRIANG using FILEIN commands. Up to 20 input files can be displayed using the SHOW FILES command.

Messages: None.

Examples:

$ **TRIANG<CR>**
DTMCREATE module TRIANG of 13:53:27 3-FEB-89
TRIANG> **SHOW FILES<CR>**
INPUT:
No input files sucessfully read yet.

OUTPUT:
Output filename not yet specified.

TRIANG>

SHOW FORMAT

Shows current status of TRIANG input file format selection.

FORMAT: SHOW FORMAT

Command parameters: None.

DESCRIPTION:

Shows current status of TRIANG input file format selection. TRIANG can read either IFF or DTI files, the file type is set using the FORMAT command.

Messages: None.

Examples:

$ **TRIANG<CR>**
DTMCREATE module TRIANG of 13:53:27 3-FEB-89
TRIANG> **SHOW FORMAT<CR>**
Input expected from IFF file
TRIANG>

SHOW FRT

Shows current status of TRIANG FRT selection.

FRT: SHOW FRT

Command parameters: None.

DESCRIPTION:

Shows current status of TRIANG FRT selection.

Messages: None.

Examples:

$ **TRIANG<CR>**
DTMCREATE module TRIANG of 13:53:27 3-FEB-89
TRIANG> **SHOW FRT<CR>**
No FRT file selected
TRIANG>

SHOW HEIGHTS

Shows current status of TRIANG height modification and datum options.

HEIGHTS: SHOW HEIGHTS

Command parameters: None.

DESCRIPTION:

Shows current status of TRIANG height modification and datum options.

Messages: None.

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:53:27  3-FEB-89
TRIANG> SHOW HEIGHTS<CR>
Incoming heights expected in IFF type 3 AC entries
MULTIPLYBY and DIVIDEBY are disabled.
No imperial or metric conversion to be applied to incoming heights
Height datum     0.000 to be added to all incoming heights
No inversion to be applied to incoming heights
TRIANG>
```

SHOW LAYER

Shows current status of TRIANG layer input selections.

FORMAT: SHOW LAYER

Command parameters: None.

DESCRIPTION:

Shows current status of TRIANG layer input selections.

Messages: None.

Examples:

$ **TRIANG<CR>**
DTMCREATE module TRIANG of 13:53:27 3-FEB-89
TRIANG> **SHOW LAYER<CR>**
Layers selected for input:
0-32767
TRIANG>

SHOW MAXPOINTS

Shows current status of the MAXPOINTS parameter.

FORMAT: SHOW MAXPOINTS

Command parameters: None.

DESCRIPTION:

Shows current status of the MAXPOINTS parameter.

Messages: None.

Examples:

$ **TRIANG<CR>**
DTMCREATE module TRIANG of 13:53:27 3-FEB-89
TRIANG> **SHOW MAXPOINTS<CR>**
Estimated number of incoming points = 0
(Max possible number without redimensioning = 79380)
TRIANG>

SHOW RIDGELINES

Shows current status of TRIANG feature code and layer ridgeline assignments made using ASSIGN and DEASSIGN commands.

FORMAT: SHOW RIDGELINES

Command parameters: None.

DESCRIPTION:

Shows current status of TRIANG feature code and layer ridgeline assignments made using ASSIGN and DEASSIGN commands.

Messages: None.

Examples:

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 13:53:27  3-FEB-89
TRIANG> SHOW RIDGELINES<CR>
No layers assigned for ridgelines
No feature codes assigned for ridgelines
TRIANG>
```

SHOW RIVERS

Shows current status of TRIANG feature code and layer river assignments made using ASSIGN and DEASSIGN commands.

FORMAT: SHOW RIVERS

Command parameters: None.

DESCRIPTION:

Shows current status of TRIANG feature code and layer river assignments made using ASSIGN and DEASSIGN commands.

Messages: None.

Examples:

$ **TRIANG<CR>**
DTMCREATE module TRIANG of 13:53:27 3-FEB-89
TRIANG> **SHOW RIVERS<CR>**
No layers assigned for rivers
No feature codes assigned for rivers
TRIANG>

SHOW SELECTIONS

Shows current status of TRIANG layer and feature code input selections.

FORMAT: SHOW SELECTIONS

Command parameters: None.

DESCRIPTION:

Shows current status of TRIANG layer and feature code input selections.

SHOW selections has the same effect as issuing SHOW FC and SHOW LAYER commands sequentially.

Messages: None.

Examples:

$ **TRIANG<CR>**
DTMCREATE module TRIANG of 13:53:27 3-FEB-89
TRIANG> **SHOW SELECTIONS<CR>**
Feature codes selected for input:
0-32767
Layers selected for input:
0-32767
TRIANG>

SHOW UNITS

Shows current status of TRIANG window units as set using the UNITS command.

FORMAT: SHOW UNITS

Command parameters: None.

DESCRIPTION:

Shows current status of TRIANG window units as set using the UNITS command.

Messages: None.

Examples:

$ **TRIANG<CR>**
DTMCREATE module TRIANG of 13:53:27 3-FEB-89
TRIANG> **SHOW UNITS<CR>**
Window to be specified in metres
TRIANG>

SHOW WINDOW

Shows current triangulation window values.

FORMAT: SHOW WINDOW

Command parameters: None.

DESCRIPTION:

Shows current triangulation window values.

The units of the WINDOW command parameters are set using the UNITS command. By default metre units are assumed. If it is more convenient to specify the window in latitude and longitude the sexagesimal lat long values may be supplied after specifying a UNITS LATLONG command. This assumes that the data read in from the DTI and IFF files are in units of tenths second of arc. A similar assumption is made for UNITS SECONDS.

The SHOW WINDOW command displays the window values in the units currently selected by the UNITS command. No projection transformation is performed, so the unwise user could easily specify that the window be shown as latitude longitude despite the fact that his data are in metres!

Messages: None.

Examples:

$ **TRIANG<CR>**
DTMCREATE module TRIANG of 13:53:27 3-FEB-89
TRIANG> **SHOW WINDOW<CR>**
Window to be specified in metres
Triangulation window currently unset
TRIANG>

SHOW ZLIMITS

Shows current status of the ZLIMITS parameters.

FORMAT: SHOW ZLIMITS

Command parameters: None.

DESCRIPTION:

Shows current status of the ZLIMITS parameters.

Messages: None.

Examples:

$ **TRIANG<CR>**
DTMCREATE module TRIANG of 13:53:27 3-FEB-89
TRIANG> **SHOW ZLIMITS<CR>**
Z-limits currently undefined
TRIANG>

SPAWN

The SPAWN command enables you to create a subprocess while within TRIANG.

FORMAT: SPAWN command-line

Command parameters:

command-line

Specifies a DCL command string to be executed as if typed in response to a '$' prompt. When the command completes, the subprocess terminates and control is returned to TRIANG. The command string cannot exceed 80 characters.

DESCRIPTION:

The SPAWN command enables you to create a subprocess while within TRIANG. When the subprocess terminates control is returned to TRIANG.

Messages:

The following warning messages are specific to the SPAWN command:

*** WARNING *** SPAWN requires a valid DCL command line
*** ERROR *** Unable to spawn command, returning to TRIANG

Examples:

TRIANG> **SPAWN DIRECTORY *.DTA;*<CR>**

Directory DUA3:[DTMCREATE.ACCEPTANCE_TESTS]

TEST1.DTA;1 8/8 18-AUG-1987 07:56 [LSL,TIM]
TEST2.DTA;2 7/8 18-AUG-1987 17:17 [LSL,TIM]
TEST2.DTA;1 7/8 18-AUG-1987 17:07 [LSL,TIM]

Total of 3 files, 22/24 blocks.

TRIANG>

UNITS

Specifies the units of measurement that will be used when defining the triangulation window using the WINDOW command.

The command also controls the units of measurement which will be used when displaying DTI file header details.

FORMAT: UNITS units

Command parameters:

units

A keyword defining the measurement units, chosen from:

METRES	Metres on the ground
LATLONG	Latitude and Longitude (in degrees, minutes and seconds)
SECONDS	Seconds of arc
PROJECTION	Projection units

DESCRIPTION:

The UNITS command enables the user to specify in what units of measurement he wishes to define the triangulation window using the WINDOW command, or in what units of measurement details from the header of the DTI file are displayed.

By default metre units are assumed.
The UNITS command should be given before specifying the triangulation window if the user wishes to specify the window in non-metre units.

If UNITS SECONDS or UNITS LATLONG are used it is assumed that the data read in using FILEIN commands are in units of tenth seconds of arc.

UNITS METRES (the default) or UNITS PROJECTION assume that the the data read in using FILEIN commands are in metres or projection units. In a future release of DTMCREATE coordinate consistency checks between input files will be implemented.

Currently the projection information for the first input file read with a FILEIN command is copied to the .NOD and .DTA output files for subsequent use in setting up the header of the LSLA type DTI file created in TRIGRID.

Messages:

The following error messages are specific to the UNITS command:

*** ERROR *** Specifying command UNITS
Command qualifiers are METRES,PROJECTION,SECONDS or LATLONG

Examples:

```
TRIANG> UNITS LATLONG<CR>
TRIANG> WINDOW 52 00 00N 08 30 00 E 52 30 00 N 09 00 00 E<CR>
TRIANG>
```

WAIT

Suspend processing for the specified number of seconds.

FORMAT: WAIT seconds

Command parameters:

seconds

The number (floating point) of seconds for which TRIANG processing is to be suspended.

DESCRIPTION:

The WAIT command causes processing to be suspended for a specified number of seconds. It is designed for use in software demonstration situations and is of no value in a production flowline.

Messages:

The following warning message is specific to the WAIT command:

*** WARNING *** You must specify the number of seconds to wait

---

Examples:

TRIANG> **WAIT 4.0<CR>**
TRIANG>

WINDOW

Specifies the limits of the data area to be triangulated.

FORMAT: WINDOW xmin ymin xmax ymax

Command parameters:

xmin ymin

The coordinates of the bottom left hand corner of the defining rectangle, (including any IFF origin offset values).

xmax ymax

The coordinates of the top right hand corner of the defining rectangle, (including any IFF origin offset values).

The units of the WINDOW command parameters are set using the UNITS command. By default metre units are assumed. If it is more convenient to specify the window in latitude and longitude the sexagesimal lat long values may be supplied after specifying a UNITS LATLONG command. This assumes that the data read in from the DTI and IFF files are in units of tenths second of arc. A similar assumption is made for UNITS SECONDS.

Remember that the WINDOW parameters must be specified in the order SW latitude longitude, NE latitude longitude if using latlong units, (i.e. x and y specification is reversed).

DESCRIPTION:

The command is used to define rectangular limits to the area of data to be included within the triangulation. The limits must be specified in the order bottom left hand (or south west) corner then top right hand (or north east) corner.

WINDOW command parameters must be expressed in absolute coordinate space (i.e. including the origin offset), not local map coordinate space (which excludes the origin offset). Only by doing this will the correct data clipping limits be applied to IFF and DTI data coming from different source files.

The WINDOW command can be used to clip data from input IFF or DTI files.

The WINDOW command is obligatory.

The rectangular limits set by the WINDOW command cannot be reset once a FILEIN command has been issued as the WINDOW command arguments are used to determine scaling between DTMCREATE internal workspace coordinates and map units.

Messages:

The following warning messages are specific to the WINDOW command:

```
*** WARNING **** Unable to read WINDOW arguments
*** WINDOW **** is still unset, please respecify the WINDOW command
*** ERROR **** window values define a zero width window
*** ERROR **** window values define a zero height window
```

Examples:

```
TRIANG> WINDOW 0.0 0.0 120.0 120.0<CR>
TRIANG>
```

ZLIMITS

Specifies an estimate of the minimum and maximum Z values in the input data files

---

FORMAT: ZLIMITS real1 real2

COMMAND PARAMETERS:

real1

The minimum Z value in the input data

real2

The maximum Z value in the input data

Description:

All data is scaled by TRIANG to an internal common form. Until all the required IFF and DTI files have been read in the full range of the Z-variable is unknown. The ZLIMITS command allows the user to specify the Z-variable range before any IFF files are read. The values that the user supplies are used to set up the Z-scaling. The ZLIMITS command requires only a rough indication of the expected minimum and maximum Z values. It is important to remember though, that the Z range specified should represent not only that of the first data set but also of any subsequent IFF files. The Z range should also reflect any later grid that may be interpolated allowing for any expected highs or lows outside the possible data range.

IMPORTANT If one of the height modification options is enabled using ENABLE MULTIPLYBY, ENABLE TOFEET etc., you must give the ZLIMITS in the target measurement system or height range (ie feet for the TOFEET command). Failure to do this may result in flattening of all model relief!

Messages:

The following warning messages are specific to the ZLIMITS command:

*** WARNING *** You must specify minimum and maximum Z value arguments
For example ZLIMITS 80.0 3000.0

*** WARNING *** You cannot reset the ZLIMITS once a file has been read in

---

Examples:

TRIANG> **SHOW ZLIMITS<CR>**
Z-limits currently undefined
TRIANG> **ZLIMITS 0 850<CR>**

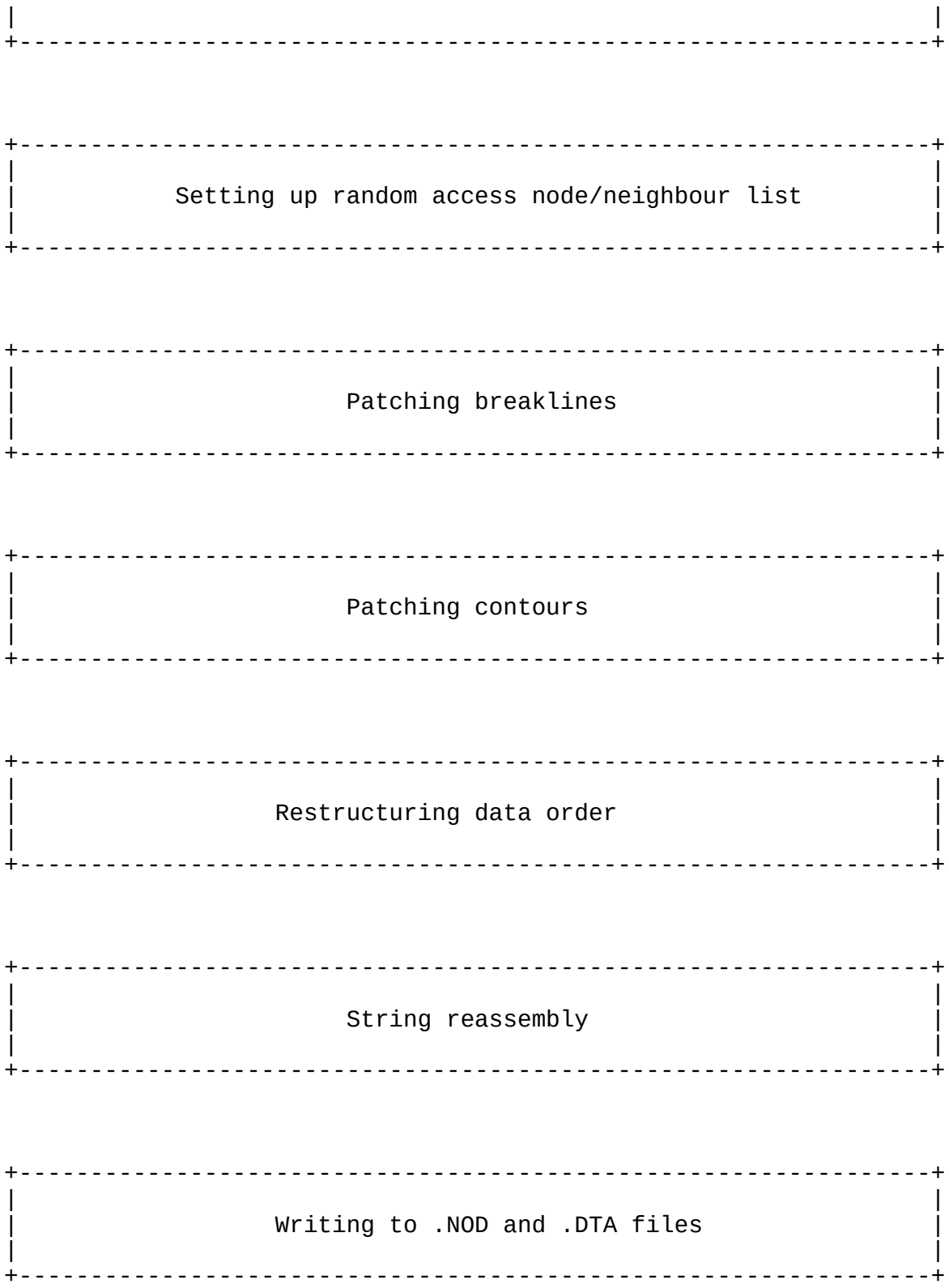
```
TRIANG> SHOW ZLIMITS<CR>
Zmin= 0.000 zmax= 850.000
TRIANG>
```

EXAMPLES

```
$ TRIANG<CR>
DTMCREATE module TRIANG of 14:17:49 24-AUG-87
TRIANG> ! set expected number of data points for internal scaling<CR>
TRIANG> ! purposes<CR>
TRIANG> MAXPOINTS 4000<CR>
TRIANG> ! set Z-limits for internal scaling purposes<CR>
TRIANG> ZLIMITS 90.0 560.0<CR>
TRIANG> ! define area that is to be triangulated<CR>
TRIANG> WINDOW 0.0 0.0 100.0 100.0
TRIANG> ! give the name to be used for the output .NOD and .DTA files<CR>
TRIANG> FILEOUT TESTPIECE<CR>
.DTA file SYS$DISK:[]TESTPIECE.DTA;0 opened for write
.NOD file SYS$DISK:[]TESTPIECE.NOD;0 opened for write
TRIANG> ENABLE DIAGNOSTICS      ! to show what is going on<CR>
TRIANG> SHOW ENABLE<CR>
CONSTRAINT ..................................................... On
DIAGNOSTICS .................................................... On
DIVIDEBY ....................................................... OFF
GRAPHICS ....................................................... Off
INTEGER_HEIGHT ................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ........................................................ Off
MULTIPLY ....................................................... Off
PME ............................................................ Off
SQUARE ......................................................... On
TOFEET ......................................................... Off
TOMETRES ....................................................... Off
TRIANG> SHOW SELECTIONS<CR>
Feature codes selected for input:
0-32767
Layers selected for input:
0-32767
TRIANG> ! specify the feature codes of features that are to be read in as<CR>
TRIANG> ! 1) breaklines<CR>
TRIANG> ASSIGN BREAKLINE_FC 77:92<CR>
TRIANG> ! 2) ridgelines<CR>
TRIANG> ASSIGN RIDGE_FC 12<CR>
TRIANG> ! 3) rivers<CR>
TRIANG> ASSIGN RIVER_FC 18:21,35<CR>
TRIANG> ! we have some unwanted road data in layer 16 which we will exclude<CR>
TRIANG> ! from the triangulation:<CR>
TRIANG> DESELECT LAYER 16<CR>
TRIANG> SHOW ASSIGNMENTS<CR>

BREAKLINES:
No layers assigned for breaklines
Feature codes assigned for breaklines:
77-92

CLIFFLINES:
No layers assigned for cliffs
```

```
No feature codes assigned for cliffs

RIDGELINES:
No layers assigned for ridgelines
Feature codes assigned for ridgelines:
12

RIVERS:
No layers assigned for rivers
Feature codes assigned for rivers:
18-21,35

TRIANG> SHOW SELECTIONS<CR>

LAYERS:
Layers assigned for input:
16
FCs:
All feature codes selected for input
TRIANG> ! get the data from the IFF file<CR>
TRIANG> FILEIN TESTDATA5<CR>
IFF file LSL$IF:TESTDATA5.IFF;0 opened for read


+-----------------------------------------------------------------+
|                                                                 |
|               Starting pass through IFF file                    |
|                                                                 |
+-----------------------------------------------------------------+


52 points retained out of 240 in feature with FSN 1 (1)
52 points retained out of 198 in feature with FSN 2 (2)
50 points retained out of 188 in feature with FSN 3 (3)
41 points retained out of 178 in feature with FSN 4 (4)
0 points retained out of 167 in feature with FSN 5 (5)
0 points retained out of 17 in feature with FSN 22 (22)
0 points retained out of 19 in feature with FSN 25 (25)
0 points retained out of 42 in feature with FSN 26 (26)
9 points retained out of 11 in feature with FSN 27 (27)
16 points retained out of 19 in feature with FSN 28 (28)
22 points retained out of 205 in feature with FSN 29 (29)
53 points retained out of 238 in feature with FSN 30 (30)
35 points retained out of 170 in feature with FSN 31 (31)
28 points retained out of 173 in feature with FSN 32 (32)
0 points retained out of 168 in feature with FSN 33 (33)
0 points retained out of 26 in feature with FSN 34 (34)
0 points retained out of 51 in feature with FSN 47 (47)
0 points retained out of 23 in feature with FSN 48 (48)
0 points retained out of 70 in feature with FSN 49 (49)
0 points retained out of 57 in feature with FSN 50 (50)
51 points retained out of 193 in feature with FSN 66 (66)
32 points retained out of 172 in feature with FSN 67 (67)
44 points retained out of 183 in feature with FSN 68 (68)
45 points retained out of 179 in feature with FSN 69 (69)
0 points retained out of 8 in feature with FSN 70 (70)
```

```
0 points retained out of 2 in feature with FSN 71 (71)
0 points retained out of 96 in feature with FSN 72 (72)
0 points retained out of 4 in feature with FSN 73 (73)
50 points retained out of 214 in feature with FSN 74 (74)
0 points retained out of 307 in feature with FSN 75 (75)
54 points retained out of 203 in feature with FSN 76 (76)
0 points retained out of 65 in feature with FSN 77 (77)
0 points retained out of 39 in feature with FSN 78 (78)
0 points retained out of 61 in feature with FSN 79 (79)
0 points retained out of 52 in feature with FSN 80 (80)
0 points retained out of 10 in feature with FSN 81 (81)
0 points retained out of 152 in feature with FSN 82 (82)
0 points retained out of 11 in feature with FSN 83 (83)
0 points retained out of 13 in feature with FSN 84 (84)
0 points retained out of 230 in feature with FSN 85 (85)
0 points retained out of 107 in feature with FSN 86 (86)
0 points retained out of 11 in feature with FSN 87 (87)
0 points retained out of 48 in feature with FSN 88 (88)
0 points retained out of 153 in feature with FSN 89 (89)
0 points retained out of 103 in feature with FSN 103 (103)
44 points retained out of 390 in feature with FSN 104 (104)
50 points retained out of 269 in feature with FSN 105 (105)
43 points retained out of 219 in feature with FSN 106 (106)
0 points retained out of 104 in feature with FSN 107 (107)
310 points retained out of 337 in feature with FSN 130 (130)
0 points retained out of 124 in feature with FSN 131 (131)
0 points retained out of 37 in feature with FSN 136 (136)
0 points retained out of 60 in feature with FSN 137 (137)
0 points retained out of 1 in feature with FSN 161 (161)
0 points retained out of 1 in feature with FSN 162 (162)
0 points retained out of 1 in feature with FSN 163 (163)
0 points retained out of 1 in feature with FSN 164 (164)
*** WARNING *** Feature with FSN 9998 (165) has no type 3 AC - feature ignored
*** WARNING *** Feature with FSN 9997 (166) has no type 3 AC - feature ignored
*** WARNING *** Feature with FSN 9999 (167) has no type 3 AC - feature ignored
*** WARNING *** Feature with FSN 9996 (168) has no type 3 AC - feature ignored
There are now 771 points in the DTM area
TRIANG> ! It is known that the four features which did not have type 3 ACs<CR>
TRIANG> ! were in fact some supplementary registration marks which should<CR>
TRIANG> ! have been deselected from input. As it is, their lack of height<CR>
TRIANG> ! tagging has resulted in TRIANG omitting them anyway.<CR>
TRIANG> ! As we have no further files to read in we can triangulate the data<CR>
TRIANG> GO<CR>

+-------------------------------------------------------------+
|                                                             |
|                Generating Delaunay triangulation            |
|                                                             |
+-------------------------------------------------------------+


Triangulation complete for 1205 points, of which 434 are imaginary

+-------------------------------------------------------------+
|                                                             |
|   Starting triangulation constraint - initialising structure   |
```



There are 1211 points in the constrained triangulation
ELAPSED: 0 00:14:22.40 CPU: 0:00:40.47 BUFIO: 468 DIRIO: 317 FAULTS: 8815

In this example TRIANG has been used to create a constrained Delaunay triangulation of a subset of data windowed out of a single IFF input file. For the purposes of explanation the diagnostics have been enabled, resulting in voluminous printout. The user has used copious comments in the command lines to tell us what he was doing. Such comments are extremely useful when writing command files to drive TRIANG in a batch processing environment, as the comments make it much easier to work out what is going on when processing problems arise or the command file has to be modified months after its creation.

Examination of the example printout shows that the user has set the MAXPOINTS argument to 4000 points. This value has been derived by using the IMP (IFF Map Processing) package module IINFO and as we are taking a windowed subset of the data, an educated guess. The MAXPOINTS value has only to be around 80% accurate for perfectly adequate allocation of TRIANG internal workspace.

The user has then used the ZLIMITS command to set the Z limits for the incoming data. Until all the required IFF data have been read the full range of the Z-variable is unknown. The values that the user supplies are used to set up the TRIANG internal Z-scaling. Like the MAXPOINTS command, the ZLIMITS command requires only a rough indication of the expected minimum and maximum values. It is important to remember, though, that the Z range specified should represent not only that of the first data set but also of any subsequent IFF files. The Z range should also reflect any later grid that may be interpolated allowing for any expected highs or lows outside the possible data range. In this example the user has used the IMP IINFO utility with the /HEIGHT_RANGE qualifier set to determine the Z minima and maxima in the file. To allow for any interpolation undershoot or overshoot, it is good practice to then extend the Z limits by one contour interval top and bottom.

The user has next specified the rectangular WINDOW which defines that portion of the data that is to be extracted from the IFF file and used for the triangulation. From the discussion of the diagnostic printout below, it is evident that the user has used the WINDOW command to clip out a comparatively small part of the whole data set.

In this simple example, input has been taken from only one IFF file. In more complex processing cases, where input is taken from several IFF and DTI files, it is important to remember that the WINDOW command arguments must reflect the data range for the desired triangulation, not just the coverage of the input file that happens to be read first.

Presumably for reassurance, the user has given the SHOW WINDOW command immediately after the WINDOW command. SHOW WINDOW caused TRIANG to print out the newly defined window bounds.

Satisfied with the window, the user has proceeded to specify the generic file specification to be used for the binary output files. All components of the supplied file-spec are used to form the output file specifications but with the substitution of the extensions .NOD and .DTA. Any parts of the file-spec not supplied for the FILEOUT command are taken from the defaults 'SYS$DISK:[].NOD;0' and 'SYS$DISK:[].DTA;0'. This indicates that logical name

LSL$DTMCREATE_WORK is not currently defined, the missing parts of the file file-spec being taken from the user's current default device and directory. (For details of the use of this logical name see the FILEOUT command above). These defaults result in the output files being created in your current default directory, set using the VMS SET DEFAULT or Laser-Scan SD commands. For reassurance TRIANG issues a message as the output files are successfully created and opened.

The next command issued is the SHOW ENABLE command. The printout produced by this command will tell the user all the current option settings. The SHOW SELECTIONS command is used to give feature code and layer selection defaults. It is evident that at the start of a TRIANG run there are no feature code or layer selections set. The SHOW ENABLE command can be issued at any time to a TRIANG> prompt to display the current command settings.

Because it is usually used in a batch processing environment, by default TRIANG produces minimal diagnostic printout. In this example the user wishes to receive indications of processing progress and of the effect of selections on data input and so has enabled diagnostic printout using the ENABLE DIAGNOSTICS command.

From this example it is obvious that if DIAGNOSTICS are enabled, TRIANG can produce voluminous printout, particularly if used during the input phase associated with FILEIN commands. It is recommended that in a batch processing production environment, diagnostic printout is suppressed.

The user next supplied feature code assignments for breaklines, ridgelines and rivers by specifying feature code arguments to ASSIGN sub-commands. Notice the use of comments and also the use of feature code ranges, which save a lot of typing!

The user has supplied a comment telling us that are some road data present in layer 16 of the IFF file which are not wanted in the triangulation. It is quite common for data unsuitable for use as DTM control to be mixed up with the required heighted information. Here the user knows that all the road data are isolated in layer 16 and has issued the DESELECT LAYER 16 command to prevent TRIANG from reading in any data from that layer. It would similarly have been possible to have used the DESELECT FC command with appropriate feature code arguments to perform the same task if the road data were more simply differentiated by feature code.

It is important to remember that the DESELECT LAYER and DESELECT FC commands are very powerful and will override any assignments made on the same layers and feature codes. This will cause features with those assignments to be excluded from the triangulation, even if, for example, the user has already assigned them to breaklines. It is therefore of the utmost importance to check feature code and layer selections and deselections most carefully to ensure that such assignment conflicts do not arise.

Very sensibly the user has next issued a SHOW SELECTIONS and a SHOW ASSIGNMENTS command so that a check for selection clashes can be made.

Satisfied with the selections made, the user has issued a FILEIN command to open and read data from the specified input file. As no FORMAT DTI command has been issued, TRIANG assumes that input is from an IFF file. Those parts of the file-spec not supplied for the FILEIN command are taken from the default file specification 'LSL$IF:IFF.IFF;0'. The file is successfully opened and a reassuring message output.

Because diagnostic printout has been enabled, TRIANG announces that it is starting a pass through the IFF file and proceeds to issue a diagnostic message for every IFF feature read from the file. The message "'integer' points retained out of 'integer' in feature with FSN 'integer' ('integer')" tells the user how much data is being rejected due to:

- points lying too close together,
- points within the string lying outside of the WINDOW limits.

It is evident that a large number of features lie entirely outside the WINDOW limits as all their points are rejected.

Near the end of input from the IFF file TRIANG has issued four warning messages. The features lack a type 3 (real height) AC (Ancillary Code) and are ignored - i.e. omitted from the triangulation. The user appears to be happy with this situation judging from the comment supplied.

What is more disturbing is that the user has specified feature codes for breaklines, ridgelines and rivers, but TRIANG has issued no diagnostics to indicate that such features have been found. If diagnostic printout is enabled, TRIANG will always issue a message each time a cliff, breakline, ridge or river feature is read.

The user seems to be content with this situation, however, and has proceeded to issue the GO command which starts the Delaunay triangulation process.

A message is output to indicate the progress of the triangulation process at 5% intervals. As only 1081 points lie within the triangulation WINDOW the triangulation takes little time. When the triangulation is complete a message is output to indicate that there are now 1205 points in the triangulation area as 124 imaginary points have been generated around the edge of the triangulation.

As no DISABLE CONSTRAINT command has been issued TRIANG proceeds to constrain the Delaunay triangulation to the paths of the input IFF strings. To do this a random access node/neighbour list has to be set up in memory, and if the data set is very large, a random access scratch file is set up on disk. If a big dataset is being constrained this setup may take some time and so messages are output at 5% intervals to indicate progress.

Triangulation constraint is conducted first on breaklines, which must always override other string features in the data structure for they define lines of slope discontinuity. Next contours, ridge and river strings are constrained. Finally the nodes are reorganised in workspace to reflect the new string patterns, and then any string patches are inserted into strings broken by the original Delaunay triangulation. The progress of each of these stages is indicated at 5% intervals.

Finally the reorganised, constrained triangulation is written to the two binary output files opened using the FILEOUT command.

The run has completed successfully and DCL symbol $STATUS is set to SS$_NORMAL, normal successful completion.

The two binary output files are ready for use by TRIDER, the slope derivative estimator, or TRIEDIT, the triangulation editor (which could be used to insert the missing river, ridgeline and breakline strings).

MESSAGES (INFORMATIONAL)

These messages give information only, and require no immediate action by the user. They are used to provide information on the current state of the program, or to supply explanatory information in support of a warning or error message.

COORDS, problem occurred at coordinates: %F0.3 %F0.3

Explanation: An error has occurred at the absolute coordinates given. This messages will be accompanied by the actual error that occurred. It is good practice to always check the input IFF file(s) using DTMPREPARE utility ITCHECK before running TRIANG.

User action: Investigate the IFF file at the coordinates given and try to determine the nature of the problem. This may be crossing breaklines or clifflines etc and correcting these may fix the problem.

MESSAGES (WARNING)

These messages are output when an error has occurred that can be corrected immediately by the user or that the program will attempt to overcome.

NOV2MD, IFF map descriptor in %S is not version 2

> **Explanation:** TRIANG expects input files to have type 2 map descriptors as it offers offset merging functionality based on the contents of the map descriptor. TRIANG is downwards compatible with old pattern IFF files which have type 1 map descriptors, but no origin offset facility is supported for the earlier pattern files.
>
> **User action:** If origin offsetting is required use ITRANS/DESCRIPTOR to create a copy of the IFF file having a type 2 map descriptor.

UNSETMD, map descriptor in %S is unset

> **Explanation:** The Map Descriptor in the specified IFF input file is unset.
>
> **User action:** If origin offsetting is required or other input files have characteristics which require the map descriptor to be set, use ITRANS/DESCRIPTOR to set up the map descriptor. Re-run TRIANG.

MESSAGES (ERROR)

These messages indicate an error in processing which will cause the program to terminate. The most likely causes are a corrupt or otherwise invalid input file, or an error related to command line processing and file manipulation.

CLODTI, error closing DTI file

> **Explanation:** An error occurred as the DTI file was being closed. The accompanying messages will help diagnose the reason.
>
> **User action:** User action will depend on the information in the accompanying DTILIB, LSLLIB or RMS messages.

OPNSCR, error opening scratch file SYS$DISK:[]TRIANG.TMP;

> **Explanation:** The data set read in has exceeded the current TRIANG memory quota and a random access disk file was going to be opened to contain the surplus. However, TRIANG has failed to open this random access file.
>
> **User action:** The supplementary message given after this error should help you to decide what has gone wrong (e.g. disk full, file protection error, etc.). Correct this problem and then re-run TRIANG.

SQRCP, file control points are not square

> **Explanation:** The IFF file which you are reading in has control points which are not square to their coordinate system axes. This may lead to problems of edge matching between DTMs at the DTITILE stage and may lead to lost data on input to TRIANG due to edge clipping.
>
> **User action:** Check that the IFF control points are meant to be non-square. If they are and the dangers of modelling with data in relative to non-square corner points are realised and accepted the the squaring test may be disables with the DISABLE SQUARE command.

STACKOVR, stack overflow - triangulation will be destroyed

> **Explanation:** When in constraint mode TRIANG has only a finite amount of space available to store all the new points that have been inserted into the data structure. You have now used up all this space and TRIANG will have to give up, losing the triangulation in the process. It is very unlikely that you will get this message as a minimum of 33% of workspace is reserved for storage of points generated during constraint. The data supplied to TRIANG must have been very poorly distributed to have caused this failure.
>
> **User action:** Use LITES2 to check the distribution of data points, or run TRIANG in graphics mode and note the distribution of triangles immediately prior to failure. If the problem persists please submit an SPR to Laser-Scan.

TOOMNYCLF, too many clifflines - maximum allowed is %N

Explanation: TRIANG has internal storage available for the specified number of clifflines and this number has been exceeded.

User action: Use the WINDOW command to divide up the input data. Generate a DTM for each resulting sub-area. Use DTITILE to reassemble the completed sub-DTMs as a final stage.

TOOMNYPTS, too many points - maximum allowed is %N

Explanation: TRIANG does not currently have sufficient workspace to handle more than the specified number of points.

User action: Use the WINDOW command to divide up the input data. Generate a DTM for each resulting sub-area. Use DTITILE to reassemble the completed sub-DTMs as a final stage.

TOOMNYSTR, too many strings - maximum allowed is %N

Explanation: TRIANG has a limit on the number of input strings that it can handle. The input data contain more than this number of strings. NB. a spot height is treated as a single point "string" inside TRIANG.

User action: Divide the input data up before submitting it to TRIANG to reduce the number of strings input at. Use DTITILE to assemble a DTM of the whole area from the sub-DTMs as a final stage.

UNEXPEOF, unexpected end of IFF file

Explanation: This message indicates there is something seriously wrong with the IFF file which has caused immediate termination of the program. TRIANG has detected the end of the IFF file, but has not detected an IFF 'EJ' entry.

User action: Use IMEND on the file, which will correctly position the EOF marker and insert an EJ entry at the end of the file. Re-run TRIANG on the corrected file.

WRTDTA, error writing to .DTA file

Explanation: An error has occurred while writing to the .DTA file on disk. This error message will be accompanied by a supplementary RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient disk space) before attempting to re-run TRIANG.

WRTNOD, error writing to .NOD file

Explanation: An error has occurred while writing to the .NOD file on disk. This error message will be accompanied by a supplementary RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient disk space) before attempting to re-run TRIANG.

MESSAGES (FATAL)

These messages indicate a severe error in processing, or some form of system failure, which has caused the program to terminate.

BADCOUNT, bad data point count in routine REASSM

Explanation: This should never happen! When inserting additional nodes in the optional CONSTRAINT phase TRIANG has to reconnect the neighbours lists of all the old nodes neighbours to the new one. However TRIANG can't find the old node in the node neighbour list.

User action: Please preserve all the input data files and then submit an SPR to Laser-Scan.

CLIFFINS, cliff insertion unexpectedly stopped

Explanation: TRIANG has unexpectedly failed to insert a cliffline into the Delaunay triangulation.

User action: Unless another problem has occurred during this TRIANG run this error should not occur. Please save all the data used for input and then submit an SPR to Laser-Scan.

INSUFFCYC, insufficient cyclic space

Explanation: While creating the Thiessen neighbours for the current node TRIANG has filled all available extension space provided for nodes which have excessive numbers of neighbours (currently greater than 10) As a result there is no space available to continue cycling through further possible neighbours.

User action: If a suitable graphics device is available (see SPS for details) re-run TRIANG on this data set with the graphics option selected. You will then be able to see where the source data is at fault. Double digitising and loops in the source digitising may produce this effect. Once detected, use LITES2 to correct the data error and then re-run TRIANG.

LIST, NODB not in list for NODA

Explanation: This error should never occur! If it does it will be associated with a CONSTRAINT command. As new points are added to the data set new inter-node relationships are formed in memory which will be added to the node/neighbour file. A new node has been formed within an existing triangle but now that the inter-node links are being formed, TRIANG is unable to locate one of the original triangle vertices (node B) in the neighbour list of node A.

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

LOST, TRIANG is lost - 4th point believed outside the imaginary point frame!

Explanation: This error should never occur! If it does it will be associated with a CONSTRAINT command. As new points are added to the data set new inter-node relationships are formed in memory which will be added to the node/neighbour file. However, the search for the neighbours for one of the new points has failed and TRIANG has begun looking outside of the imaginary point frame!

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

LOST4TH, GTFRTH could not find a 4th point

Explanation: This error should never occur! If it does it will be associated with a CONSTRAINT command. As new points are added to the data set new inter-node relationships are formed in memory which will be added to the node/neighbour file. A new node has been formed within an existing triangle but now that the inter-node links are being formed, TRIANG is unable to locate one of the original triangle vertices.

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

NODEOF, unexpected end of node file

Explanation: TRIANG has read off the end of the node file.

User action: Unless another problem has occurred during this TRIANG run this error should not occur. Please save all the data used for input and then submit an SPR to Laser-Scan.

NODPAIR, NODA/NODB pair not found in routine INJOIN

Explanation: Another error that should never occur! If it does it will be associated with a CONSTRAINT command. As new points are added to the data set new inter-node relationships are formed in memory which will be added to the node/neighbour file. TRIANG is unable to locate a link between two triangle vertices.

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

NOEXT, run out of extension space

Explanation: There are no more rows of memory extension space left to hold the apparently excessive number of neighbours associated with some of the nodes in your data set.

User action: If a suitable graphics device is available (see SPS for details) re-run TRIANG on this data set with the graphics option selected. You will then be able to see where the source data is at fault. Look for isolated points which will be connected to enormous numbers of other points within the triangulation. Once detected, use LITES2 to correct the data error by adding formlines and then re-run TRIANG. If the problem persists please save all the input files and the submit an SPR to Laser-Scan.

NOLD, old node not found for termination

Explanation: This error should never occur! If it does it will be associated with a CONSTRAINT command. During the insertion of a string patch of new points TRIANG was unable to find the original node which should lie at the far end of the insertion.

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

RANDRD, error during random read

Explanation: An error has occurred reading from a random access disk file used when there are too many nodes to hold in memory. This error message will be accompanied by a supplementary FORTRAN or RMS message which will indicate the cause of the error.

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

RANDWRT, error during random write in routine %S

Explanation: An error has occurred writing to a random access disk file used when there are too many nodes to hold in memory. This error message will be accompanied by a supplementary FORTRAN or RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient disk space) before attempting to re-run TRIANG

RDNOD, error reading .NOD file

Explanation: TRIANG has suffered a read error when reading back from the .NOD file.

User action: Unless another problem has occurred during this TRIANG run this error should not occur. Please save all the data used for input and then submit an SPR to Laser-Scan.

STARNOD, Unable to find start node in routine COLLEC

Explanation: While collecting the neighbours of the current node the start node could not be found again in the node list.

User action: If a suitable graphics device is available (see SPS for details) re-run TRIANG on this data set with the graphics option selected. You will then be able to see where the source data is at fault. If after attempting to use LITES2 to correct the source IFF data (or if input is from DTI file only) please preserve all the input data and then submit an SPR to Laser-Scan.

STOPNOD, Stop node in triangle - but not found

Explanation: During triangle CONSTRAINT all string patches must terminate on an existing node. The current patch stop node lies inside a triangle - not at a triangle vertex!

User action: Unless another problem has occurred during this TRIANG run this error should not occur. Please save all the data used for input and then submit an SPR to Laser-Scan.

TOMNYNEIB, too many neighbours

Explanation: This is a problem which only arises when the source data is very unevenly distributed. The addition of some additional formlines to the source data-set will always solve this problem.

User action: Use LITES2 to add formlines in the IFF input file in the area of uneven data distribution and the re-run TRIANG.

TOOMNYPAT, too many string patches

Explanation: When the optional CONSTRAINT mode is invoked TRIANG generates short sections of linework or "patches" containing new nodes. This are used to fill in the gaps in logical strings caused by triangle leakage through the strings. There is only a finite amount of memory allocated for storage of these patches. This has now been exceeded. The triangulation has been lost. The source data points supplied to TRIANG must have been very unevenly distributed for this problem to arise.

User action: Checkplot the source data and examine the data distribution before reporting this problem to Laser-Scan. If there is nothing that can be done with the source data, then the data should be divided into two or more parts before the re-running TRIANG.

UNRECREC, unrecognised record number

Explanation: Each node entry in workspace is identified by a positive record number, if there are more records than can be held in memory then the remainder are written to a random access disk file. Somehow routine GETNAY has found a record with an identification that is either less than 1 or greater than the current maximum recorded number of records.

User action: Unless another problem has occurred during this TRIANG run this error should not occur. Please save all the data used for input and then submit an SPR to Laser-Scan.

VERSUBS, error in vertex substitution

Explanation: This error should never occur! If it does it will be associated with a CONSTRAINT command. An unspecified error has occurred during vertex substitution resulting from the insertion of new data points.

User action: Please preserve all the input data and then submit an SPR to Laser-Scan.

MESSAGES (OTHER)

In addition to the above messages which are generated by the program itself, other messages may be produced by the command line interpreter (CLI) and by Laser-Scan libraries. In particular, messages may be generated by the DTILIB library and by the Laser-Scan I/O library, LSLLIB. DTILIB library messages are introduced by '%DTILIB' and are documented in the MATRIX package reference manual. In most cases DTI errors will be due to a corrupt input file, and this should be the first area of investigation. If the cause of the error cannot be traced by the user, and Laser-Scan are consulted, then the output file should be preserved to facilitate diagnosis. LSLLIB messages are introduced by '%LSLLIB' and are generally self-explanatory. They are used to explain the details of program generated errors.

CHAPTER 5

MODULE TRIDER

MODULE TRIDER

REPLACES PANACEA module PANDER

FUNCTION

TRIDER takes the triangulation node and data files created by TRIANG (or edited output from TRIEDIT) and produces an output file containing slope derivatives at each data point in the triangulation.

FORMAT

$ TRIDER

COMMAND QUALIFIERS

None, TRIDER is command driven.

DESCRIPTION

General

TRIDER takes the triangulation node and data files created by TRIANG (or edited output from TRIEDIT) and produces an output file containing slope derivatives at each data point in the triangulation. These data may then be used by TRIGRID in conjunction with the node and data files as the basis for DTM grid estimation. Once TRIDER has been used to generate a slope derivative file many subsequent runs of TRIGRID may be made to produce DTM grids at differing resolutions.

TRIDER always works on the whole area defined within the triangulation files. It is not possible to specify a subset of the data for slope estimation. This is to allow continuity of slope estimation in edge areas which may otherwise be impaired if the data area is segmented.

TRIDER input and output files

TRIDER expects as input the binary structured data files (the matched pair of .NOD and .DTA files) produced by TRIANG or TRIEDIT. It produces an output file containing the slope derivatives estimated for each data point contained in the triangulation.

The TRIDER input file specification is used as a generic file-spec for both input (.NOD and .DTA) files and as the source for the TRIDER output file specification.

Any parts missing from the generic file-specification are taken from the defaults SYS$DISK:[].DTA;0 and SYS$DISK:[].NOD;0.

The specification of the output file has the same device, directory filename and version number as the input files but is given the extension .DER.

Since it is essential that the file version numbers of the .NOD, .DTA and .DER file always match, TRIGRID performs checks on file version numbers. If mismatches are found, TRIDER complains and aborts execution.

Never use the VAX/VMS RENAME or COPY commands to alter the version numbers of .NOD, .DTA files or .DER files to make them into a matched set. TRIDER will have to be rerun every time that the .NOD and .DTA files are modified using TRIEDIT.

The output files from TRIANG will be scaled to lie between 0 and the value defined by logical, LSL$DTMCREATE_RESOLUTION, or 300000 if the logical is not defined. The valid range for this resolution is between 300000 and 10000000.

TRIDER uses this logical value to determine the internal resolution of the .NOD and .DTA files.

IMPORTANT

It is therefore essential that the logical value LSL$DTMCREATE_RESOLUTION remains the same when going from TRIANG all the way through to TRIGRID on a particular dataset. If the resolution is altered between any of the stages, unpredictable results will occur and programs may fail.

TRIDER and imaginary point estimation

Imaginary points - a review

Beyond the edge of the triangulation there lies the abyss where there are no heights to control the form of the surface. Obviously if the user wishes to make full use of the triangulated area by gridding up to the very edges, some sensible estimate must be made of the surface beyond the actual data limits. The imaginary point frame created by TRIANG helps to remove uncertainty by providing a cosmetic solution. The imaginary points have location but no Z value.

Introduction

One of the functions of TRIDER is to provide Z values for the imaginary points generated around the edge of the triangulation by TRIANG. Four interpolation options are available for imaginary point estimation:

- o FIXED - force all imaginary points to take the same fixed height value
- o TREND - (default) - estimate imaginary point heights from a trend surface

  Interpolation options:

- o BOX - interpolate height from nodes obtained using expanding box search
- o SHELL/NEIGHBOUR - interpolate height from nodes in expanding shells of neighbours

Limits may be set to the interpolation process used to estimate heights for the imaginary points which allow upper and lower Z value range clipping. This facility may be used, for example, in coastal areas to ensure that no imaginary point Z value is allowed to go below sea level, ie 0.0.

Imaginary point heights as fixed values

The fixed option assumes that the user wishes the surface to decline to some fixed value all the way around the edge of the triangulated area at some distance (yet to be determined) from the main grid coverage. By using the FIXED option the user can specify the value to be used for the imaginary point Z values. TRIDER will then automatically insert this value at all imaginary point locations and will also insert zero partial derivatives at each of the imaginary points. This will have the effect of making the surface flatten to a plane by the time that the imaginary points are reached. This plane will be horizontal because the derivatives have been set to zero.

Imaginary point height estimation using the trend surface option (default)

Quite often the user will not have any idea what individual values to give the imaginary points but will know that he does not want a fixed horizontal plane at the edge of his data. The second (startup default) solution therefore is to assign the values of the imaginary points to be those calculated from a linear trend surface fitted through all the data points in the interpolation area. The derivatives assigned to the imaginary points will then be those partial derivatives estimated for the plane itself. Thus if the data

set has a definite trend (for instance increasing in height to the south west) this option would produce an equivalent trend in the heights of the imaginary points.

Imaginary point height estimation using interpolation options

There are two alternative options for the estimation of imaginary point values based on "reasonably" close data points.

- o BOX
- o SHELL/NEIGHBOUR

Four interpolation techniques are offered for the BOX and SHELL/NEIGHBOUR interpolation options:

1. Unweighted,
2. linear,
3. quadratic, or,
4. quartic.

- o IMAGINARY BOX - This option performs a distance weighted interpolation based on points discovered by means of an expanding hollow box search. The box search is in turn based around the box structure generated by TRIANG. The interpolation is formed by standard methods. The box size expands automatically until sufficient points have been discovered. This method is probably most applicable to genuinely variable data of the seismic or non-contour type. If spot heights form a large proportion of the data set or any other irregularly distributed data is used, the box option should be chosen as it will provide a very smooth edge around the data area.

- o IMAGINARY SHELL/NEIGHBOUR - The method of collection of data points employed is that of successively finding the neighbours of the point in question by running through the triangular node/neighbour data structure created by program TRIANG and stored in a file. The process can be thought of as that of picking up points in a series of layers. Thus the first layer consists of the neighbours of the point itself. The second layer contains the neighbours of the neighbours that have been picked up but not including any previously found points. This process could obviously continue until all data points in the data set have been found!

  A height for every imaginary point will then be estimated using a specified weighting function, using the neighbours that have been found. If an insufficient number of real neighbours has been found then the program will automatically search deeper into the

data structure.

Imaginary point relocation

All imaginary points are capable of relocation by a factor from 0 to 1 (the maximum grid dimension). This can be very useful if one wishes to produce a grid of a larger area than the triangulation covers. By positioning the imaginary points out as far as possible (ie by choosing 1) the area of coverage of the triangulation is effectively extended by a factor of nearly 9. Of course this does nothing for the quality of the map as such but does ensure that some sort of "reasonable" surface value will be put in these regions that are very distant from real data. If the extension is not sufficiently far then default values will be inserted into the grid which can later be altered by the user to the default of his choice.

When the FIXED or TREND options have been chosen it is sensible to locate the imaginary points reasonably far away from the edge of the grid. This is necessary to ensure that the surface has a fairly flexible approach to reaching the imaginary point values. In areas where the surface is not close to the trend or fixed value it needs room to manoeuvre itself to these values! If an interpolation option has been chosen for the imaginary points then in most cases it will be sensible, if not vital, to keep the imaginary points close to the grid itself. This ensures that the imaginary points are most closely controlled by those points falling closest to them.

TRIDER graphics output option

TRIDER offers the user graphics output of the type offered by TRIANG and TRIGRID. The progress of imaginary and fixed point derivative estimation may then be watched. The convoluted process of imaginary point derivative and height estimation using the shell/neighbour approach may be observed! The graphics option is a useful aid for instruction or for the analysis of troublesome data sets.

TRIDER commands

@

Take command input from the specified file.

FORMAT: @file-spec<CR>

Command parameters:

file-spec

The file to be opened and used for command input.

Any parts of the file-spec not supplied for the @ command will be taken from the default specification 'SYS$DISK:[].COM;0'.

DESCRIPTION:

TRIDER offers the facility of command input from an indirect command file. The '@' character preceding a file-spec will cause TRIDER to open and read commands from the specified file until:

1. a RETURN command is detected and command input is returned to SYS$COMMAND.

2. a GO command is detected - after completion of derivative estimation TRIDER exits.

3. end-of-file is detected. This provokes an error message and command input is returned to SYS$COMMAND.

Nested command files are not supported (i.e. a command file containing an '@' command), although sequential '@' commands are supported when read from SYS$COMMAND.

As an aid to batch log interpretation TRIDER will echo all commands read from an indirect command file.

Messages:

The following messages are specific to the @ command:

*** WARNING *** "@" must precede a file-spec

*** WARNING *** Indirect file error - returning to terminal input

*** ERROR *** Can't open indirect command file 'file-spec'

Examples:

```
$ TRIDER<CR>
DTMCREATE module TRIDER of 16:30:12 20-NOV-87
TRIDER> @FLOW2<CR>
TRIDER> ENABLE DIAGNOSTICS
TRIDER> ENABLE GRAPHICS
TRIDER> IMAGINARY FIXED 0.0 1.0
TRIDER> RETURN
TRIDER>
```

!

Treat all text to the right of the '!' as a comment.

FORMAT: ! [comment text]

Command parameters:

comment text

text that is to be treated as a comment and which will be excluded from command interpretation.

DESCRIPTION:

An exclamation mark is the standard DTM package comment delimiter. All text (and numbers) which lie to the right of a '!' character are excluded from command interpretation. Comments are useful for annotating command procedures used in batch processing etc.

Messages: None.

Examples:

```
TRIDER> ! a comment for the sake of it<CR>
TRIDER> ENABLE GRAPHICS !  turn graphics on<CR>
TRIDER> !  get files and estimate the derivatives<CR>
TRIDER> FILEIN DUA0:[TESTDATA]IDAHO<CR>
.DTA file DUA3:[DEMONSTRATION]IDAHO.DTA;15 opened for read
.NOD file DUA3:[DEMONSTRATION]IDAHO.NOD;15 opened for read
.DER file DUA3:[DEMONSTRATION]IDAHO.DER;15 opened for write
TRIDER> GO<CR>
 ELAPSED:  00:05:25.84 CPU:  0:00:05.71 BUFIO:  281 DIRIO:  46 FAULTS:  263
$
```

DISABLE DIAGNOSTICS

Disables a previous ENABLE DIAGNOSTICS command.

FORMAT: DISABLE DIAGNOSTICS

Command parameters: None.

DESCRIPTION:

DISABLE DIAGNOSTICS allows the user to cancel the effect of a previous ENABLE DIAGNOSTICS command.

Messages: None.

Examples:

```
TRIDER> ENABLE DIAGNOSTICS<CR>
TRIDER> SHOW ENABLE<CR>
DIAGNOSTICS ..................................................  On
GRAPHICS .....................................................  Off
PME ..........................................................  Off
TRIDER> DISABLE DIAGNOSTICS<CR>
TRIDER> SHOW ENABLE<CR>
DIAGNOSTICS ..................................................  Off
GRAPHICS .....................................................  Off
PME ..........................................................  Off
TRIDER>
```

DISABLE GRAPHICS

Disables any previous ENABLE GRAPHICS command.

FORMAT: DISABLE GRAPHICS

Command parameter: None.

DESCRIPTION:

The DISABLE GRAPHICS command cancels the effect of a previous ENABLE GRAPHICS command.

Messages: None.

Examples:

```
$ TRIDER<CR>
DTMCREATE module TRIDER of 13:30:39 20-AUG-87
TRIDER> SHOW ENABLE<CR>
DIAGNOSTICS ..................................................  Off
GRAPHICS .....................................................  Off
PME ..........................................................  Off
TRIDER> ENABLE GRAPHICS<CR>
TRIDER> SHOW ENABLE<CR>
DIAGNOSTICS ..................................................  Off
GRAPHICS .....................................................  On
PME ..........................................................  Off
TRIDER>
```

DISABLE PME

DISABLE PME disables the effect of a previous ENABLE PME command.

FORMAT: DISABLE PME

Command parameters: None.

DESCRIPTION:

The ENABLE PME and DISABLE PME commands are reserved for Laser-Scan use. PME is a code optimisation tool and should be invoked by LSL software personnel only.

DISABLE PME disables the effect of a previous ENABLE PME command and causes the PME_EXIT routine to be invoked.

Message:

The following warning message is specific to the DISABLE PME command:

*** WARNING *** You were not using PME anyway!

Examples:

```
$ TRIDER<CR>
DTMCREATE module TRIDER of 13:30:39 20-AUG-87
TRIDER> ENABLE PME !  turn PME on<CR>
TRIDER> DISABLE PME !  turn it off again!!<CR>
TRIDER> FILEIN FAITH<CR>
.DTA file DUA3:[DEMONSTRATION]FAITH.DTA;15 opened for read
.NOD file DUA3:[DEMONSTRATION]FAITH.NOD;15 opened for read
.DER file DUA3:[DEMONSTRATION]FAITH.DER;15 opened for write
TRIDER> GO<CR>
 ELAPSED:  00:05:25.84 CPU:  0:00:05.71 BUFIO:  281 DIRIO:  46 FAULTS:  263
$
```

ENABLE DIAGNOSTICS

ENABLE DIAGNOSTICS allows the user to enable diagnostic printout.

FORMAT: ENABLE DIAGNOSTICS

Command parameters: None.

DESCRIPTION:

ENABLE DIAGNOSTICS allows the user to enable diagnostic printout.

Because it is usually used in a batch processing environment, by default TRIDER produces minimal diagnostic printout. If however, the user wishes to receive indications of processing progress and of the effect of selections on data input, diagnostic printout may be selected with the ENABLE DIAGNOSTICS command.

It should be noted that if DIAGNOSTICS are enabled, TRIDER can produce voluminous printout.

If SYS$OUTPUT is directed to a video screen terminal, messages indicating percentage progress are issued.

Messages: None.

Examples:

```
TRIDER> ENABLE DIAGNOSTICS<CR>
TRIDER> SHOW ENABLE<CR>
DIAGNOSTICS ..................................................  On
GRAPHICS .....................................................  Off
PME ..........................................................  Off
TRIDER> DISABLE DIAGNOSTICS<CR>
TRIDER> SHOW ENABLE<CR>
DIAGNOSTICS ..................................................  Off
GRAPHICS .....................................................  Off
PME ..........................................................  Off
TRIDER>
```

ENABLE GRAPHICS

Enable TRIDER graphics output.

FORMAT: ENABLE GRAPHICS

Command parameters: None.

DESCRIPTION:

TRIDER offers the option to generate graphic output to indicate processing progress. By default graphic output is disabled. To prevent a user selecting graphics when it is inappropriate to the current terminal, TRIDER uses a lookup table of terminal characteristics associated with all available terminal lines (see Appendix 1). An invalid graphics selection will result in a warning message and the default NO GRAPHICS option being selected.

Graphics selection may be cancelled with the DISABLE GRAPHICS command.

Messages:

The following messages are specific to the ENABLE GRAPHICS command:

*** ERROR *** reading lookup file at line 'integer'
*** WARNING *** Unable to open "LSL$LOOKUP:TERMTYPE.DAT"
Sorry 'name' terminal 'terminal-ident' isn't in the lookup table
Sorry 'name' terminal 'terminal-ident' can't support graphics
*** ERROR *** translating logical name LSL$DTMCREATETERMINAL

Examples:

$ **TRIDER<CR>**
DTMCREATE module TRIDER of 13:30:39 20-AUG-87
TRIDER> **SHOW ENABLE<CR>**
DIAGNOSTICS ..... Off
GRAPHICS ..... Off
PME ..... Off
TRIDER> **ENABLE GRAPHICS <CR>**
TRIDER> **SHOW ENABLE<CR>**
DIAGNOSTICS ..... Off
GRAPHICS ..... On
PME ..... Off
TRIDER> **DISABLE GRAPHICS<CR>**
TRIDER> **SHOW ENABLE<CR>**
DIAGNOSTICS ..... Off
GRAPHICS ..... Off
PME ..... Off
TRIDER>

ENABLE PME

ENABLE PME enables the PME performance monitor.

FORMAT: ENABLE PME

Command parameters: None.

DESCRIPTION:

The ENABLE PME and DISABLE PME commands are reserved for Laser-Scan use. PME is a code optimisation tool and should be invoked by LSL software personnel only.

ENABLE PME causes the PME_INIT routine to be invoked.

Message:

The following warning message is specific to the ENABLE PME command:

*** WARNING *** You are already using PME!

Examples:

```
$ TRIDER<CR>
DTMCREATE module TRIDER of 13:30:39 20-AUG-87
TRIDER> ENABLE PME<CR>
TRIDER> FILEIN FAITH<CR>
.DTA file DUA3:[DEMONSTRATION]FAITH.DTA;15 opened for read
.NOD file DUA3:[DEMONSTRATION]FAITH.NOD;15 opened for read
.DER file DUA3:[DEMONSTRATION]FAITH.DER;15 opened for write
TRIDER> GO<CR>
 ELAPSED:  00:05:25.84 CPU:  0:00:05.71 BUFIO:  281 DIRIO:  46 FAULTS:  263
$
```

FILEIN

Specifies the generic (.DTA and .NOD) file-spec that is to be used for input.

FORMAT: FILEIN file-spec

COMMAND PARAMETERS:

file-spec

The generic specification of the .NOD and .DTA files to be opened for data input.

All components of the supplied file-spec are used to form the input file specifications but with the substitution of the extensions .NOD and .DTA and version number ';0', i.e. latest version (shared by both files).

The default file-spec used to make up missing parts of the FILEIN file-spec parameter is dependent on the status of logical name LSL$DTMCREATE_WORK.

If logical name LSL$DTMCREATE_WORK is defined, DTMCREATE utilities translate the logical name to get the default file-spec for input and output of triangulation files. The logical name should be defined to provide a device and directory name only. The DTMCREATE programs themselves provide the default filename and extension fields of the specification. For example, a valid definition of logical name LSL$DTMCREATE_WORK is:

$ DEFINE LSL$DTMCREATE_WORK LSL$DATA_ROOT:[LSL.DTMCREATE]

This mechanism allows all DTMCREATE triangulation files to be stored in a central directory, rather than scattered in many different user directories. It thus mimics the use of logical names LSL$IF for IFF files and LSL$DTI for DTI files.

If the logical name is not defined, any parts of the file-spec not supplied for the FILEIN command will be taken from the defaults 'SYS$DISK:[].NOD;0' and 'SYS$DISK:[].DTA;0'. These defaults result in the files being searched for in your current default directory, set using the VMS SET DEFAULT or Laser-Scan SD commands.

DESCRIPTION:

TRIDER expects as input the 2 binary structured data files (the matched pair of .NOD and .DTA files) produced by TRIANG or TRIEDIT. It produces an output file containing the slope derivatives estimated for each data point contained in the triangulation.

The specification of the output file has the same device, directory filename and version number as the input files but is given the extension .DER.

Since it is essential that the file version numbers of the .NOD, .DTA and .DER file always match, TRIGRID performs checks on file version numbers. If mismatches are found, TRIDER complains and aborts execution. Similarly, an error occurs if TRIDER attempts to create a new .DER file which has a version number which already exists.

Messages:

The following messages are specific to the FILEIN command:

*** WARNING *** You must specify a file-spec argument to the FILEIN command
For example FILEIN SWAREA
*** ERROR *** Unable to interpret input file-spec

*** ERROR *** opening input file

.DTA file 'file-spec' opened for read
.NOD file 'file-spec' opened for read
.DER file 'file-spec' opened for write

Examples:

TRIDER> **FILEIN DUA3:[DEMONSTRATION]IDAHO<CR>**
.DTA file DUA3:[DEMONSTRATION]IDAHO.DTA;15 opened for read
.NOD file DUA3:[DEMONSTRATION]IDAHO.NOD;15 opened for read
.DER file DUA3:[DEMONSTRATION]IDAHO.DER;15 opened for write
TRIDER>

GO

Initiates the processing of the data read in using FILEIN commands.

FORMAT: GO

Command parameters: None.

DESCRIPTION:

When all necessary files have been read in the GO command will commence the gridding process. Unless relatively small data-sets are being handled (say less then 50,000 data points) it is strongly recommended that TRIDER is run in batch mode at an off-peak time.

When grid creation is complete, TRIDER closes the output DTI file and then exits.

Messages:

Once the GO command has been issued no more conversational messages of the *** WARNING *** format will be issued. Any messages will relate to serious processing problems and will normally result in abnormal TRIDER termination. The messages relating to non-interactive processing problems are presented at the end of this document.

Examples:

TRIDER> **GO<CR>**

ELAPSED: 0 00:05:21.82 CPU: 0:00:01.40 BUFIO: 51 DIRIO: 15 FAULTS: 170
$

HELP

Give help on a subject

FORMAT: HELP subject

Command parameters:

subject

The subject on which help is required

Description:

The HELP command looks the rest of the line up in the DTMCREATE HELP library. This library contains a brief summary of the operation of each command.

The information is looked up in the TRIDER section of the DTMCREATE help library, LSL$HELP:DTMCREATE.HLB.

Messages:

Where required, warning messages are output via the VMS LBR$OUTPUT_HELP utility.

Examples:

TRIDER> **HELP RETURN<CR>**

RETURN

Restores command input from an indirect file to SYS$COMMAND.

A typical application is to allow the user to use an indirect command file to set up those run time defaults which are constant within a flowline and then return to input from the terminal (or batch stream) for the run specific commands. To do this RETURN must be the last command in the indirect command file.

TRIDER>

IMAGINARY BOX

The IMAGINARY BOX command specifies interpolation of imaginary point heights on the basis of known heights found using an expanding hollow box search.

FORMAT: IMAGINARY BOX interpolation reloc

Command parameters:

interpolation

A keyword argument specifying the type of interpolation to be used, chosen from:

- LINEAR
- UNWEIGHTED
- QUADRATIC
- QUARTIC

reloc

The imaginary point relocation proportion (in the range 0.0 to 1.0)

DESCRIPTION:

This imaginary point height estimation option performs a distance weighted interpolation based on points discovered by means of an expanding hollow box search. The box search is in turn based around the box structure generated by TRIANG. The interpolation is formed by standard methods. The box size expands automatically until sufficient points have been discovered. This method is probably most applicable to genuinely variable data of the seismic or non contour type. If spot heights form a large proportion of the data set or any other irregularly distributed data is used, the box option should be chosen as it will provide a very smooth edge around the data area.

IMAGINARY BOX and imaginary point relocation

As an interpolation option has been chosen for the imaginary points then in most cases it will be sensible, if not vital, to keep the imaginary points close to the grid itself. This ensures that the imaginary points are most closely controlled by those points falling closest to them. Thus set the imaginary point relocation proportion to 0.0 or close to 0.0.

Messages:

```
*** WARNING *** Please use the IMAGINARY command again and choose from FIXED
                TREND, BOX or SHELLNEIGHBOUR as the first command argument.
                The default method of imaginary point estimation is
                'method', which remains unchanged

*** ERROR *** Missing IMAGINARY command arguments

*** ERROR *** Missing weighting function name

*** WARNING *** "LINEAR" interpolation selected by default

*** ERROR *** Missing relocation proportion

*** ERROR *** Reading relocation proportion
```

Examples:

TRIDER> **IMAGINARY BOX QUARTIC 0.0<CR>**
TRIDER>

IMAGINARY FIXED

The IMAGINARY FIXED command specifies that imaginary point heights should be fixed at a specified constant value.

FORMAT: IMAGINARY FIXED fixval reloc

Command parameters:

fixval

A constant height value to be used for all imaginary points.

reloc

The imaginary point relocation proportion (in the range 0.0 to 1.0)

DESCRIPTION:

The fixed option assumes that the user wishes the surface to decline to some fixed value all the way around the edge of the triangulated area at some distance (yet to be determined) from the main grid coverage. By using the FIXED option the user can specify the value to be used for the imaginary point Z values. TRIDER will then automatically insert this value at all imaginary point locations and will also insert zero partial derivatives at each of the imaginary points. This will have the effect of making the surface flatten to a plane by the time that the imaginary points are reached. This plane will be horizontal because the derivatives have been set to zero.

IMAGINARY FIXED and imaginary point relocation

When the IMAGINARY FIXED option is chosen, it is sensible to locate the imaginary points reasonably far away from the edge of the grid. This is necessary to ensure that the surface has a fairly flexible approach to reaching the imaginary point values. In areas where the surface is not close to the fixed value it needs room to manoeuvre itself to these values!

Messages:

*** WARNING *** Please use the IMAGINARY command again and choose from FIXED TREND, BOX or SHELLNEIGHBOUR as the first command argument. The default method of imaginary point estimation is 'method', which remains unchanged

*** ERROR *** Missing IMAGINARY command arguments

*** ERROR *** Missing relocation proportion

*** ERROR *** Reading relocation proportion

Examples:

TRIDER> **IMAGINARY FIXED 0.0 1.0<CR>**
TRIDER>

IMAGINARY SHELLNEIGHBOUR

The IMAGINARY SHELLNEIGHBOUR command specifies interpolation of imaginary point heights on the basis of known heights of nodes found in expanding shells of neighbours

---

FORMAT: IMAGINARY SHELLNEIGHBOUR interpolation reloc

Command parameters:

interpolation

A keyword argument specifying the type of interpolation to be used, chosen from:

- LINEAR
- UNWEIGHTED
- QUADRATIC
- QUARTIC

reloc

The imaginary point relocation proportion (in the range 0.0 to 1.0)

DESCRIPTION:

IMAGINARY SHELL/NEIGHBOUR - The method of collection of data points employed is that of successively finding the neighbours of the point in question by running through the triangular node/neighbour data structure created by program TRIANG and stored in a file. The process can be thought of as that of picking up points in a series of layers. Thus the first layer consists of the neighbours of the point itself. The second layer contains the neighbours of the neighbours that have been picked up but not including any previously found points. This process could obviously continue until all data points in the data set have been found!

A height for every imaginary point will then be estimated using a specified weighting function, using the neighbours that have been found. If an insufficient number of of real neighbours has been found then the program will automatically search deeper into the data structure.

If any of the first layer neighbours contain points marked as discontinuous during the initial data formation stage at the beginning of TRIANG then a further search is made. This checks whether any layer 1 adjacent neighbours are:

- o discontinuous, and,
- o have the same z value.

If so then the interpolation point is assigned the same value as these. The reason for this is that it is quite possible that there will be flat plateau-like areas on a contour map which should remain flat at the edges of a map. Usually, and hopefully, there will be control points in these flat areas at the edges to help the automatic calculation process along. But sometimes this can be difficult and hence this check for discontinuous points is an attempt to ensure locally that the imaginary points are biased towards representing locally flat discontinuous structures.

IMAGINARY SHELLNEIGHBOUR and imaginary point relocation

As an interpolation option has been chosen for the imaginary points then in most cases it will be sensible, if not vital, to keep the imaginary points close to the grid itself. This ensures that the imaginary points are most closely controlled by those points falling closest to them. Thus set the imaginary point relocation proportion to 0.0 or close to 0.0.

Messages:

```
*** WARNING *** Please use the IMAGINARY command again and choose from FIXED
                TREND, BOX or SHELLNEIGHBOUR as the first command argument.
                The default method of imaginary point estimation is
                'method', which remains unchanged '

*** ERROR *** Missing IMAGINARY command arguments

*** ERROR *** Missing weighting function name

*** WARNING *** "LINEAR" interpolation selected by default

*** ERROR *** Missing relocation proportion

*** ERROR *** Reading relocation proportion
```

Examples:

```
TRIDER> IMAGINARY SHELLNEIGHBOUR QUADRATIC 0.0<CR>
TRIDER>
```

IMAGINARY TREND

The IMAGINARY TREND command specifies estimation of imaginary point height values from a trend surface fitted through the known node heights.

FORMAT: IMAGINARY TREND reloc

Command parameters:

reloc
The imaginary point relocation proportion (in the range 0.0 to 1.0)

DESCRIPTION:

Quite often the user will not have any idea what individual values to give the imaginary points but will know that he does not want a fixed horizontal plane at the edge of his data. The default IMAGINARY option therefore is to assign the values of the imaginary points to be those calculated from a linear trend surface fitted through all the data points in the interpolation area. The derivatives assigned to the imaginary points will then be those partial derivatives estimated for the plane itself. Thus if the data set has a definite trend (for instance increasing in height to the south west) this option would produce an equivalent trend in the heights of the imaginary points.

IMAGINARY TREND and imaginary point relocation

When the IMAGINARY TREND option is chosen, it is sensible to locate the imaginary points reasonably far away from the edge of the grid. This is necessary to ensure that the surface has a fairly flexible approach to reaching the imaginary point values. In areas where the surface is not close to the trend value it needs room to manoeuvre itself to these values!

Messages:

```
*** WARNING *** Please use the IMAGINARY command again and choose from FIXED
                TREND, BOX or SHELLNEIGHBOUR as the first command argument.
                The default method of imaginary point estimation is
                'method', which remains unchanged '

*** ERROR *** Missing IMAGINARY command arguments

*** ERROR *** Missing relocation proportion

*** ERROR *** Reading relocation proportion
```

Examples:

```
TRIDER> IMAGINARY TREND 1.0<CR>
TRIDER>
```

PAUSE

Pauses TRIDER execution.

FORMAT: PAUSE

Command parameters: None.

DESCRIPTION:

Pauses TRIDER execution and issues a prompt for a carriage return to continue execution. This command is designed for use in software demonstration situations.

Messages: None.

Examples:

TRIDER> **PAUSE<CR>**

Press <RETURN> to continue**<CR>**
TRIDER>

QUIT

Quit from TRIDER.

FORMAT: QUIT

Command parameters: None.

Description:

The QUIT command causes TRIDER to exit immediately, closing all input files and closing and deleting all output files.

<CTRL/Z> (pressing the Ctrl and Z keys together) may also be used to quit from the program.

Messages: None.

Examples:

```
TRIDER> QUIT<CR>
 ELAPSED:  00:05:25.84 CPU:  0:00:05.71 BUFIO:  281 DIRIO:  46 FAULTS:  263
$
```

RETURN

Restores command input from an indirect file to SYS$COMMAND.

FORMAT: RETURN

Command parameters: None.

DESCRIPTION:

Restores command input from an indirect file to SYS$COMMAND.

A typical application is to allow the user to use an indirect command file to set up those run time defaults which are constant within a flowline and then return to input from the terminal (or batch stream) for the run specific commands. To do this RETURN must be the last command in the indirect command file.

Messages:

The following messages are specific to the RETURN command:

RETURN command detected - returning to terminal input
RETURN command ignored - command input is already from terminal

---

Examples:

```
TRIDER> @FLOW2<CR>
TRIDER> ENABLE DIAGNOSTICS
TRIDER> FRT FLOW2
FRT file LSL$FRT:FLOW2.FRT;8 opened for read
TRIDER> SELECT OPEN_CLIFF_FC OUTCROPS,7,COAST
TRIDER> RETURN
TRIDER>
```

SHOW

Shows current status of TRIDER option and parameter settings.

FORMAT: SHOW subject

Command parameters:

subject

The subject that is to be displayed, chosen from:

ENABLE FILES IMAGINARY ZLIMITS

DESCRIPTION:

SHOW enables the user to examine the current status of TRIDER options and parameter settings.

Messages:

TRIDER issues the following message if the SHOW command is specified without an argument:

*** ERROR *** Specifying command SHOW

Available SHOW command qualifiers are:

ENABLE FILES IMAGINARY ZLIMITS

This feature can be used to advantage if the user wishes to quickly determine for which items the SHOW facility is available.

Examples:

$ **TRIDER<CR>**
DTMCREATE module TRIDER of 17:02:12 23-JAN-89
TRIDER> **SHOW<CR>**
*** ERROR *** Specifying command SHOW

Available SHOW command qualifiers are:

ENABLE FILES IMAGINARY ZLIMITS

TRIDER> **SHOW ENABLE ! examine current option settings<CR>**
DIAGNOSTICS ..... Off
GRAPHICS ..... Off
PME ..... Off
TRIDER>

SHOW ENABLE

Shows current status of TRIDER option settings.

FORMAT: SHOW ENABLE

Command parameters: None.

DESCRIPTION:

SHOW ENABLE enables the user to examine the current status of TRIDER processing options that are set or unset using the ENABLE and DISABLE commands.

Messages: None.

Examples:

```
$ TRIDER<CR>
DTMCREATE module TRIDER of 17:02:12 23-JAN-89
TRIDER> SHOW ENABLE ! examine current option settings<CR>
DIAGNOSTICS ..................................................... Off
GRAPHICS ........................................................ Off
PME ............................................................. Off
TRIDER>
```

SHOW FILES

Shows current TRIDER input files.

FORMAT: SHOW FILES

Command parameters: None.

DESCRIPTION:

SHOW FILES enables the user to examine the current status of TRIDER input files.

Messages: None.

Examples:

```
$ TRIDER<CR>
DTMCREATE module TRIDER of 17:02:12 23-JAN-89
TRIDER> SHOW FILES ! examine input file-specs<CR>
Input filename not yet specified.
TRIDER> FILEIN SW100230 ! get input files<CR>
.DTA file DUA3:[DEMONSTRATION]SW100230.DTA;6 opened for read
.NOD file DUA3:[DEMONSTRATION]SW100230.NOD;6 opened for read
.DER file DUA3:[DEMONSTRATION]SW100230.DER;6 opened for write
TRIDER> SHOW FILES ! examine input file-specs<CR>
.DTA file:   DUA3:[DEMONSTRATION]SW100230.DTA;6
.NOD file:   DUA3:[DEMONSTRATION]SW100230.NOD;6
TRIDER>
```

SHOW IMAGINARY

Shows current status of TRIDER imaginary point option settings.

FORMAT: SHOW IMAGINARY

Command parameters: None.

DESCRIPTION:

SHOW IMAGINARY enables the user to examine the current status of TRIDER imaginary point processing options that are set using the IMAGINARY commands.

Messages: None.

Examples:

```
$ TRIDER<CR>
DTMCREATE module TRIDER of 17:02:12 23-JAN-89
TRIDER> SHOW IMAGINARY ! examine imaginary points settings<CR>
Imaginary point estimation option ..... TREND
Relocation proportion for imaginary points ..... 1.00
TRIDER>
```

SHOW ZLIMITS

Shows current status of TRIDER Z-limits.

FORMAT: SHOW ZLIMITS

Command parameters: None.

DESCRIPTION:

SHOW ZLIMITS enables the user to examine the current status of TRIDER Z-limits parameters set using the ZLIMITS commands.

Messages: None.

Examples:

```
$ TRIDER<CR>
DTMCREATE module TRIDER of 17:02:12 23-JAN-89
TRIDER> SHOW ZLIMITS ! examine ZLIMITS points settings<CR>
No Z-limits will be applied to imaginary point height estimation
TRIDER> ZLIMITS 0.0 450.0<CR>
TRIDER> SHOW ZLIMITS ! examine ZLIMITS points settings<CR>
Imaginary point height estimation lower z-value limit = 0.0
Imaginary point height estimation upper z-value limit = 450.0
TRIDER>
```

SPAWN

The SPAWN command enables you to create a subprocess while within TRIDER.

FORMAT: SPAWN command-line

Command parameters:

command-line

Specifies a DCL command string to be executed as if typed in response to a '$' prompt. When the command completes, the subprocess terminates and control is returned to TRIDER. The command string cannot exceed 80 characters.

DESCRIPTION:

The SPAWN command enables you to create a subprocess while within TRIDER. When the subprocess terminates control is returned to TRIDER.

Messages:

The following warning messages are specific to the SPAWN command:

*** WARNING *** SPAWN requires a valid DCL command line
*** ERROR *** Unable to spawn command, returning to TRIDER

Examples:

TRIDER> **SPAWN DIRECTORY *.DTA;*<CR>**

Directory DUA3:[DTMCREATE.ACCEPTANCE_TESTS]

TEST1.DTA;1 8/8 18-AUG-1987 07:56 [LSL,TIM]
TEST2.DTA;2 7/8 18-AUG-1987 17:17 [LSL,TIM]
TEST2.DTA;1 7/8 18-AUG-1987 17:07 [LSL,TIM]

Total of 3 files, 22/24 blocks.

TRIDER>

WAIT

Suspend processing for the specified number of seconds.

FORMAT: WAIT seconds

Command parameters:

seconds

The number (floating point) of seconds for which TRIDER processing is to be suspended.

DESCRIPTION:

The WAIT command causes processing to be suspended for a specified number of seconds. It is designed for use in software demonstration situations and is of no value in a production flowline.

Messages:

The following warning message is specific to the WAIT command:

*** WARNING *** You must specify the number of seconds to wait

---

Examples:

TRIDER> **WAIT 4.0<CR>**
TRIDER>

ZLIMITS

Specifies minimum and maximum Z-limits for imaginary point height estimation.

FORMAT: ZLIMITS real1 real2

COMMAND PARAMETERS:

real1

The minimum Z value which an imaginary point can have.

real2

The maximum Z value which an imaginary point can have.

Description:

The ZLIMITS command enables the user to specify lower and upper ZLIMITS to imaginary point height estimation. If an imaginary point height is calculated which falls outside of the specified limits the height will be truncated to the relevant limit.

By default TRIDER applies no limits to imaginary point height estimation.

It is not necessary to specify the ZLIMITS command if the IMAGINARY FIXED imaginary point height option is used.

IMPORTANT If one of the height modification options was selected in TRIANG using ENABLE MULTIPLYBY, ENABLE TOFEET etc., you must give the TRIDER ZLIMITS argument values in the target measurement system or height range (ie feet if the TRIANG TOFEET command was specified). Failure to do this may result in flattening of all model relief!

Messages:

The following messages are specific to the ZLIMITS command:

*** ERROR *** You must specify minimum and maximum ZLIMITS arguments
For example ZLIMITS 80.0 3000.0

Examples:

TRIDER> **SHOW ZLIMITS<CR>**
No ZLIMITS will be applied to imaginary point height estimation
TRIDER> ZLIMITS 0.0 800.0<CR>
TRIDER> **SHOW ZLIMITS<CR>**

```
Imaginary point height estimation lower z-value limit = 0.0
Imaginary point height estimation upper z-value limit = 800.0
TRIDER>
```

EXAMPLES

```
$ TRIDER<CR>
DTMCREATE module TRIDER of 15:53:45 26-NOV-87
TRIDER> ENABLE DIAGNOSTICS<CR>
TRIDER> FILEIN TST<CR>
.DTA file DATA$DISK:[DTMCREATE.TRIANG]TST.DTA;1 opened for read
.NOD file DATA$DISK:[DTMCREATE.TRIANG]TST.NOD;1 opened for read
.DER file DATA$DISK:[DTMCREATE.TRIANG]TST.DER;1 opened for write
TRIDER> GO<CR>

+-------------------------------------------------------------+
|                                                             |
|          Setting up memory/random file for neighbours       |
|                                                             |
+-------------------------------------------------------------+



+-------------------------------------------------------------+
|                                                             |
|                    Calculating slope derivatives            |
|                                                             |
+-------------------------------------------------------------+


Partial derivatives estimated for 774 points
 ELAPSED:    0 00:02:05.51  CPU: 0:00:25.60  BUFIO: 13  DIRIO: 25  FAULTS: 332
$
```

This example shows TRIDER in use with the default TREND imaginary point height estimation option selected. TRIDER first reads in all nodes and neighbour relationships from the matched pair of binary files:

DATA$DISK:[DTMCREATE.TRIANG]TST.DTA;1

DATA$DISK:[DTMCREATE.TRIANG]TST.NOD;1

The user has merely specified FILEIN TST, the rest of the input file specification components are taken from the default SYS$DISK:[].DTA and SYS$DISK:[].NOD, i.e. the user's current default device and directory. The file version number is taken from the actual files.

A binary output file DATA$DISK:[DTMCREATE.TRIANG]TST.NOD;1 is created and opened to receive the slope derivatives.

After the GO command TRIDER starts by estimating heights for the imaginary points around the edge of the data area. By default the imaginary points are relocated by the maximum possible amount away from the existing edge of the data area. This is to provide TRIDER with maximum leeway when fitting the quintic surface from which the

slope derivatives are taken. As ZLIMITS were not specified, the heights estimated for the imaginary points are not constrained in any way.

TRIDER then proceeds to fit quintic surface patches across the data area wherever possible. Where the nodes are too poorly distributed or simply too few for a high order surface patch to be fitted, TRIDER will lower the order of surface to be fitted and try again. As the slope derivatives for each node are estimated, TRIDER writes them to the derivative file DATA$DISK:[DTMCREATE.TRIANG]TST.DER;1.

The run has completed successfully and DCL symbol $STATUS is set to SS$_NORMAL, i.e. normal successful completion.

```
$ TRIDER<CR>
DTMCREATE module TRIDER of 15:53:45 26-NOV-87
TRIDER> ENABLE DIAGNOSTICS<CR>
TRIDER> ZLIMITS 80 500<CR>
TRIDER> IMAGINARY SHELLNEIGHBOUR QUADRATIC 0.0<CR>
TRIDER> FILEIN QUARRY5<CR>
.DTA file DATA$DISK:[DTMCREATE.TRIANG]QUARRY5.DTA;1 opened for read
.NOD file DATA$DISK:[DTMCREATE.TRIANG]QUARRY5.NOD;1 opened for read
.DER file DATA$DISK:[DTMCREATE.TRIANG]QUARRY5.DER;1 opened for write
TRIDER> GO<CR>

+-----------------------------------------------------------------+
|                                                                 |
|          Setting up memory/random file for neighbours           |
|                                                                 |
+-----------------------------------------------------------------+



+-----------------------------------------------------------------+
|                                                                 |
|     Imaginary point interpolation - collect star neighbours     |
|                                                                 |
+-----------------------------------------------------------------+


446 imaginary point neighbourhoods determined

+-----------------------------------------------------------------+
|                                                                 |
|               Starting derivative calculation phase             |
|                                                                 |
+-----------------------------------------------------------------+


%TRIDER-W-DERFAIL, derivative estimation failure at X = 2000.02 Y = 70.315
Partial derivatives estimated for 22402 points

+-----------------------------------------------------------------+
|                                                                 |
|       Imaginary point interpolation - interpolation phase       |
```

```
|                                                                  |
+------------------------------------------------------------------+


446 imaginary points interpolated

+------------------------------------------------------------------+
|                                                                  |
|                    Calculating slope derivatives                 |
|                                                                  |
+------------------------------------------------------------------+


Partial derivatives estimated for 68820 points
 ELAPSED:    0 00:05:58.79  CPU: 0:04:56.19  BUFIO: 317  DIRIO: 236 FAULTS: 1368
$
```

This example shows the SHELLNEIGHBOUR imaginary point height estimation option in use.

After opening the matched pair of binary input files and creating the binary slope derivative output file, TRIDER estimates heights for the imaginary points around the edge of the data area. In contrast to the previous example, which used the default TREND option for imaginary point height estimation, the imaginary points are left on the edge of the existing data area (i.e. the imaginary point relocation proportion is zero). Shells of neighbours to each imaginary point are collected as the basis for the interpolation used to estimate the imaginary point height. The user has specified that a quadratic (IMAGINARY SHELLNEIGHBOUR **QUADRATIC**) surface be fitted through the imaginary point's neighbours.

The ZLIMITS arguments specify minimum and maximum constraints for the heights interpolated for imaginary points.

TRIDER then proceeds to fit quintic surface patches across the data area wherever possible. Where the nodes are too poorly distributed or simply too few for a high order surface patch to be fitted, TRIDER will lower the order of surface to be fitted and try again. As the slope derivatives for each node are estimated, TRIDER writes them to the derivative file DATA$DISK:[DTMCREATE.TRIANG]QUARRY5.DER;1.

Attention is drawn to the message warning that derivative estimation has failed for the node at the specified location. Zero slope derivatives are assumed for this node and TRIDER processing continues. Such messages should occur comparatively rarely and reflect instances where the data are too poorly distributed (in x,y and z) to allow a surface patch to be fitted.

The run has completed successfully and DCL symbol $STATUS is set to SS$_NORMAL, i.e. normal successful completion.

MESSAGES (WARNING)

These messages are output when an error has occurred that can be corrected immediately by the user or that the program will attempt to overcome.

DERFAIL, derivative estimation failure at X ='real' Y = 'real'

Explanation: Derivative estimation has failed for the node at the specified location. Zero slope derivatives will be assumed for this node and TRIDER processing will continue.

User action: None, although if many such errors occur, check the distribution and nature of the input data.

INSPTLAY, insufficient points found in layers - abandoning option

Explanation: When the shell/neighbour approach to imaginary point estimation is selected, TRIDER goes through the neighbour list, keeping and storing in an array as much of it as possible. This is unlikely to be all of it because of lack of space in the array, but must be at least one shell's worth to contain all the imaginary points, or an error is detected. Then for every imaginary point in turn TRIDER picks up the real neighbour numbers, and assembles them without duplicates in the array. These points are tagged for derivative estimation, and written out to file for use by the interpolation algorithm. If any imaginary point has no neighbours (within the storage constraints of the array) then shell/neighbour estimation is abandoned and the box interpolation option is selected.

User action: None.

INSUFFPT, insufficient points for imaginary point interpolation

Explanation: During imaginary point estimation TRIDER cycles through the imaginary points, finding the extended neighbours. It then picks up points until sufficient points have been found and are sufficiently well distributed to perform the interpolation. Unfortunately for the present point there are no suitable neighbours in the search box. NB: the use of an expanding hollow square ensures that the points nearest to the interpolation point are included at the head of the list. This means that if space runs out then the most important ones have been collected already.

User action: None. If the message occurs very frequently, examine the distribution of the data points in the triangulation. This should give a clue as to what is wrong, e.g. extremely irregularly distributed data.

SEAFILE, Unable to find 'file-spec' - try again

Explanation: Before trying to open an input .NOD or .DTA file TRIDER searches the disk for the file specification supplied. If TRIDER finds it then all is well, the file-spec is parsed and the version number extracted to make up the name of the .DER file. Unfortunately you have specified a non-existent file-spec. TRIDER will allow you to try again.

User action: Respecify the input file-spec.

MESSAGES (ERROR)

These messages indicate an error in processing which will cause the program to terminate. The most likely causes are a corrupt or otherwise invalid input file, or an error related to command line processing and file manipulation.

NOPTSIM, no points for imaginary point interpolation

> **Explanation:** During imaginary point estimation TRIDER cycles through the imaginary points, finding the extended neighbours. It then picks up points in an expanding hollow square until sufficient points have been found and are sufficiently well distributed to perform the interpolation. Unfortunately for the present point there are no suitable neighbours in the search box. NB: the use of an expanding hollow square ensures that the points nearest to the interpolation point are included at the head of the list. This means that if space runs out then the most important ones have been collected already.
>
> **User action:** Examine the distribution of the data points in the triangulation. This should give a clue as to what is wrong, e.g. no data or extremely irregularly distributed data. Even so this should not normally happen. Please make a copy of all relevant data and the .NOD and .DTA files and submit an SPR to Laser-Scan.

OPNDER, error opening 'file-spec' for write

> **Explanation:** The system error return supplied with the message will help you to decide what to do.
>
> **User action:** Check:
>
> 1. That the disk is not full,
> 2. that you have the privilege to write to the current directory,
> 3. that the file is not already open or is locked.
> 4. that the file already exists. You will have to delete the existing copy before you re-run TRIDER.

OPNDTA, error opening 'file-spec' for read

> **Explanation:** The system error return supplied with the message will help you to decide what to do.
>
> **User action:** Check:
>
> 1. That the file exists,
> 2. that you are in the correct directory,
> 3. that you have the privilege to read the file.

OPNNOD, error opening 'file-spec' for read

Explanation: The system error return supplied with the message will help you to decide what to do.

User action: Check:

1. That the file exists,

2. that you are in the correct directory,

3. that you have the privilege to read the file.

OPNSCR, error opening scratch file SYS$DISK:[]TRIDER.TMP;

Explanation: TRIDER needs to open a scratch file to store extended neighbours of imaginary points. However, TRIDER has failed to open this disk file.

User action: The supplementary message given after this error should help you to decide what has gone wrong (e.g. disk full, file protection error, etc.). Correct this problem and then re-run TRIDER.

RDDTA, error reading from .DTA file

Explanation: An error has occurred while reading the .DTA file on disk. This error message will be accompanied by a supplementary RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient privilege or file protection violation) before attempting to re-run TRIDER.

RDNOD, error reading .NOD file

Explanation: TRIDER has suffered a read error when reading the .NOD file. This error message will be accompanied by a supplementary RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient privilege or file protection violation) before attempting to re-run TRIDER. If the supplementary RMS error message indicates that the end of file was unexpectedly found check that the TRIANG or TRIEDIT run which created the .NOD file terminated successfully. If it didn't then re-run TRIANG or TRIEDIT and then try TRIDER agaion on the corrected file. If all appears to be correct and the problem persists, please report the problem to Laser-Scan.

RDSCR, error reading from scratch file

Explanation: An error has occurred reading from a temporary scratch file used when there are too many nodes to hold in memory. This error message will be accompanied by a supplementary FORTRAN or RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient privilege or file protection violation) before attempting to re-run TRIDER.

WRTDER, error writing to .DER file

Explanation: An error has occurred while writing to the derivative .DER file on disk. This error message will be accompanied by a supplementary RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient disk space) before attempting to re-run TRIDER.

WRTSCR, error writing to scratch file

Explanation: An error has occurred while writing to the a temporary scratch file on disk. This error message will be accompanied by a supplementary RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient disk space) before attempting to re-run TRIDER.

MESSAGES (FATAL)

These messages indicate a severe error in processing, or some form of system failure, which has caused the program to terminate.

BOXOVR, too little space for boxes

Explanation: TRIDER uses the box structure created by TRIANG. Your current version of TRIDER is insufficiently dimensioned to cope with the number of boxes just read in.

User action: This should never happen as the DTMCREATE modules are dimensioned to be mutually compatible! Please report this error to Laser-Scan. Until TRIDER can be redimensioned divide up your original IFF file and re-run TRIANG on the resulting sub-areas. TRIDER will then probably be able to cope with the reduced data set size. The resulting sub-DTMs can be joined to form the whole DTM area using DTITILE.

DERIVOVR, Derivative storage overflow

Explanation: This should never happen.

User action: Please report this error to Laser-Scan. Until TRIDER can be redimensioned try dividing up your original IFF file and re-run TRIANG on the resulting sub-areas. TRIDER will then probably be able to cope with the reduced data set size. The resulting sub-DTMs can be joined to form the whole DTM area using DTITILE.

NODOVR, node has more than 150 neighbours

Explanation: TRIDER can currently can only handle nodes with less than 150 neighbours. The data is almost certainly very corrupt if this message appears, possibly as a result of over enthusiastic data insertion in TRIEDIT. Normally a node will only have up to about 9 neighbours.

User action: The triangulation is irrevocably damaged. Re-run TRIANG and try running TRIDER again. If the problem persists please contact Laser-Scan.

NONEIGHB, no neighbours found in TRACKN

Explanation: This should never happen!

User action: Please make a copy of all relevant data and the .NOD and .DTA files and submit an SPR to Laser-Scan.

RANDRD, error during random read

Explanation: An error has occurred reading from a random access disk file used when there are too many nodes to hold in memory. This error message will be accompanied by a supplementary FORTRAN or RMS message which will indicate the cause of the error.

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

RANDWRT, error during random write in routine 'file-spec'

Explanation: An error has occurred writing to a random access disk file used when there are too many nodes to hold in memory. This error message will be accompanied by a supplementary FORTRAN or RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient disk space) before attempting to re-run TRIDER.

STACKOVR, stack overflow

Explanation: TRIDER has only a finite amount of space available to store all the nodes and thir neighbour relationships. It is very unlikely that you will get this message as all modules in DTMCREATE share the same dimensioning parameters and it should be impossible for TRIANG or TRIEDIT to output .DTA and .NOD files which require such a large stack.

User action: Please submit an SPR to Laser-Scan.

TOMNYNEIB, too many neighbours

Explanation: TRIDER has encountered a node with more neighbours than can be safely stored. This should never happen as all the modules of the DTMCREATE package share common workspace dimensioning parameters and this node should have been rejected by TRIANG.

User action: Please report this problem to Laser-Scan.

UNRECREC, unrecognised record number

Explanation: Each node entry in workspace is identified by a positive record number, if there are more records than can be held in memory then the remainder are written to a random access disk file. Somehow TRIDER has found a record with an identification that is either less than 1 or greater than the current maximum recorded number of records.

User action: Unless another problem has occurred during this TRIDER run this error should not occur. Please save all the data used for input and then submit an SPR to Laser-Scan.

MESSAGES (OTHER)

In addition to the above messages which are generated by the program itself, other messages may be produced by the command line interpreter (CLI) and by Laser-Scan libraries. In particular, messages may be generated by the IFF library and by the Laser-Scan I/O library, LSLLIB. IFF library messages are introduced by '%IFF' and are documented in the IFF library users' guide. In most cases IFF errors will be due to a corrupt input file, and this should be the first area of investigation. If the cause of the error cannot be traced by the user, and Laser-Scan are consulted, then the output file should be preserved to facilitate diagnosis. LSLLIB messages are introduced by '%LSLLIB' and are generally self-explanatory. They are used to explain the details of program generated errors.

CHAPTER 6

MODULE TRIEDIT

MODULE TRIEDIT

REPLACES PANACEA module PANDEMON

FUNCTION

TRIEDIT is the DTMCREATE interactive graphic editor. It takes triangulation files produced by TRIANG and enables additional information to be added, and existing relationships and attributes to be modified.

FORMAT

$ TRIEDIT [file-spec] [[hardware-option] ...]

To provide maximum flexibility of graphics device selection, TRIEDIT uses lookup files to determine what type of graphics hardware is available on the current terminal line. The default hardware selection may be overridden by the user on the initial TRIEDIT command line, or at a later stage while the program is running by means of ENABLE command arguments. A detailed description of the hardware lookup files and selection mechanism is given in the DESCRIPTION section below.

TRIEDIT may be activated in 3 ways:

1. $ **TRIEDIT<CR>**

   when prompted for a file-spec supply the file-spec only. This will give you the default hardware configuration for the current terminal line.

2. $ **TRIEDIT file-spec<CR>**

   This will have the same effect as 1), but will avoid the prompt stage.

3. $ **TRIEDIT file-spec hardware-option [hardware-option]<CR>**

   Where 'hardware-option' is a graphics hardware option (see PARAMETERS below) This will result in the default hardware configuration for the current terminal line being overridden with your explicit selection.

   You must supply the file-spec argument on the command line if the default graphics options are to be overriden from the command line.

The TRIEDIT command line decoder is designed to give the user maximum flexibility with regard to the degree of parameter abbreviation and the ordering of the parameters. The only restrictions are:

- one of the selected device keywords must be a graphics device keyword
- the file-spec must be the first parameter on the command line

---

PARAMETERS

file-spec

Specifies the generic (.DTA and .NOD) file-spec that is to be used for input. If no file-spec argument is supplied on the command line, TRIEDIT issues a prompt for the file-spec once the hardware device has been initialised:

.DTA file>

TRIEDIT expects as input the 2 binary structured data files (the matched pair of .NOD and .DTA files) produced by TRIANG or previous TRIEDIT sessions.

The TRIEDIT input file specification is used as a generic file-spec for both input (.NOD and .DTA) files and as the default TRIEDIT output file specifications.

All components of the supplied file-spec are used to form the input file specifications but with the substitution of the extensions .NOD and .DTA and version number ';0', i.e. latest version (shared by both files).

The default file-spec used to make up missing parts of the TRIEDIT file-spec parameter is dependent on the status of logical name LSL$DTMCREATE_WORK.

If logical name LSL$DTMCREATE_WORK is defined, DTMCREATE utilities translate the logical name to get the default file-spec for input and output of triangulation files. The logical name should be defined to provide a device and directory name only. The DTMCREATE programs themselves provide the default filename and extension fields of the specification. For example, a valid definition of logical name LSL$DTMCREATE_WORK is:

```
$ DEFINE LSL$DTMCREATE_WORK LSL$DATA_ROOT:[LSL.DTMCREATE]
```

This mechanism allows all DTMCREATE triangulation files to be stored in a central directory, rather than scattered in many different user directories. It thus mimics the use of logical names LSL$IF for IFF files and LSL$DTI for DTI files.

If the logical name is not defined, any parts of the file-spec not supplied for the FILEIN command will be taken from the defaults 'SYS$DISK:[].NOD;0' and 'SYS$DISK:[].DTA;0'. These defaults result in the files being searched for in your current default directory, set using the VMS SET DEFAULT or Laser-Scan SD commands.

Since it is essential that the file version numbers of the .NOD and .DTA files match, TRIEDIT performs checks on file version numbers. If the two files don't have the same version number, TRIEDIT complains and aborts execution.

The output files from TRIANG will be scaled to lie between 0 and the value defined by logical, LSL$DTMCREATE_RESOLUTION, or 300000 if the logical is not defined. The valid range for this resolution is between 300000 and 10000000.

TRIEDIT uses this logical value to determine the internal resolution of the .NOD and .DTA files.

IMPORTANT

It is therefore essential that the logical value LSL$DTMCREATE_RESOLUTION remains the same when going from TRIANG all the way through to TRIGRID on a particular dataset. If the resolution is altered between any of the stages, unpredictable results will occur and programs may fail.

hardware-option

A hardware option keyword which will be used to override the default hardware configuration for the current terminal line.

When such hardware option keywords are specified, TRIEDIT checks in another site dependent lookup file that the desired options are available (and are in a valid combination) and then initialises the devices accordingly.

If only one hardware option keyword is supplied it must define a graphics terminal type. Additional keywords may define GIN (Graphics INput) options, e.g. joysticks, bitpads, etc.

Available hardware option keywords are:

KEYWORD	DEVICE
T4014	Standalone Tektronix 4014
MUART_T4014	Laser-Scan WOSP driven T4014
S7000	SIGMA ARGS 7000 colour display
S6100	SIGMEX 6100 colour display
T4010	Standalone Tektronix 4010
T4105	Standalone Tektronix 4105
T4106	Standalone Tektronix 4106
T4107	Standalone Tektronix 4107
T4109	Standalone Tektronix 4109
T4115	Standalone Tektronix 4115
VT100	Use VT100 for status area and interaction
GPX	VAXstation II, 2000 or 3000 series workstation
BITPAD	SIGMEX 6100 series bitpad
MUART_TABLE	Altek Datatab digitising table controlled via WOSP
TABLE	Altek Datatab digitising table controlled via Laser-Scan table monitor program
TRACKERBALL	Sigmex ARGS 7000 trackerball option
JOYSTICK	Tektronix 4100 series terminal joystick/joypad
MOUSE	VAXstation II, 2000 or 3000 mouse
THUMBWHEELS	Tektronix 4000 series thumbwheels
NOGRAPHICS	As it says, but will need a VT100 for interaction

Some example command lines using these keywords are as follows:

$ **TRIEDIT filename.DTA WITH A TABLE A S7000 AND A VT100<CR>**

This will result in a SIGMA 7000 screen being used for graphics, a VT100 for the command text, and interaction will be via the digitising table.

The same result will be achieved (rather curtly!) by:

$ **TRIEDIT filename.DTA TA S7 VT<CR>**

$ **TRIEDIT 'file-spec' WITH T4014<CR>**

This will result in both the graphics and the command text being sent to a Tektronix 4014. Unfortunately you have forgotten to specify a device for interaction with the graphics screen, which for the Tektronix would have to be thumbwheels or a digitising table. However, once the editing session is in progress it is possible to choose new GIN options with the ENABLE command (see below).

COMMAND QUALIFIERS

None, TRIEDIT is command driven.

DESCRIPTION

General

TRIEDIT can handle supplementary input from IFF (Internal Feature Format) files consisting of strings of contour, seismic, or other types of data, marked as either continuous or discontinuous nodes.

The input data are used to modify a (possibly constrained) Delaunay triangulation produced by TRIANG or a previous TRIEDIT editing session.

TRIEDIT and TRIANG share common commands for the definition of IFF file input options, such as IFF layer and feature code selection options, etc.

Like TRIANG, TRIEDIT enables the user to apply feature flags to the incoming nodes to identify each node as either a river node, a ridgeline node or an unflagged node. These flags are used within TRIGRID to control the limits applied to smooth surface interpolation.

Input and output data files

Triangulation input and output from module TRIEDIT is in the form of two binary triangulation files, not directly readable by the user. These files are used to convey the triangulation data structure between the various modules of the DTMCREATE package. The first file (input_filename.DTA) contains the data in scaled integer form, together with various indices and tables concerning the distribution of the data over the triangulation area. This data set is augmented by a set of imaginary nodes acting as a frame around the edge of the map area. The second file (input_filename.NOD) contains the data structure itself with a list of the neighbours of each data node in the set.

These triangulation files must always be handled as matched pairs, as the data contained in one is the key to the inter-node structures described in the other. If one file is to be deleted then BOTH files must be deleted. DTMCREATE modules TRIEDIT, TRIDER and TRIGRID which require input from .NOD and .DTA files check that the two files share a common version number and complain if they do not!

Never use the DCL SET FILE, RENAME or COPY commands to alter the version numbers of mismatched .NOD and .DTA files to make them into a matched pair.

TRIEDIT and input from IFF files

As an alternative to using the INSERT command to add new data interactively, TRIEDIT accepts input of contour and breakline strings and spot heights from IFF files.

TRIEDIT is designed to be compatible with the "new" type IFF files introduced in conjunction with the IMP (IFF Map Processing) package. The origin offset entry in a type 2 MD (Map Descriptor) entry is used to offset coordinate values within an input IFF file. Although downwards compatible with "old" type IFF files a warning message is issued if an IFF file is found not to contain a set type 2 MD (Map Descriptor) entry.

Data within IFF files may have string type and feature type attributes assigned within TRIEDIT via the IFF feature code and layer values. Commands such as ASSIGN BREAKLINE_FC, ASSIGN RIDGE_LAYER, etc. are provided to make the feature type/code assignments to enable TRIEDIT to realize that a string is to be stored as a breakline or a ridgeline, etc.

Heights

In IFF files height values are transmitted via AC (Ancillary Code) or ZS (3D string) entries. By default contour and spot height Z values are read from type 3 ACs as floating point values. By use of the ENABLE INTEGER_HEIGHT command, integer heights may be read from type 2 ACs.

IFF files are read into TRIEDIT using the FILEIN command. More then one input file may be specified using a new FILEIN command for each file. Defaults may be changed between reading files. This means that IFF files with different allocations of IFF entries for height information or with heights relative to different data may be combined.

Height information within a single IFF file may be stored relative to different height data providing that the features pertaining to each height datum can be distinguished by feature code or layer. The DATUM command can be used to set the height datum for a subsequent FILEIN command. If required a single IFF file may be read in many times relative to different height data, feature selection being achieved by SELECT FC and SELECT LAYER commands.

Imperial heights may be converted to metric on input with the ENABLE TOMETRES command. The reverse is possible with the ENABLE TOFEET command. Incoming heights may be multiplied or divided by a user specified constant with the ENABLE MULTIPLYBY and ENABLE DIVIDEBY commands respectively.

BREAKLINES

In order that all surface discontinuities are honoured in the final DTM it may be necessary to designate some IFF strings as "breaklines", which will ensure a change of slope character at that line. A breakline in the IFF file may be identified by its feature code or position in a separate layer. The allocation of IFF feature codes and layers for breaklines must be input to TRIEDIT using the ASSIGN BREAKLINE_FC and ASSIGN BREAKLINE_LAYER command respectively. As with the IFF height storage mechanisms different allocations of layers and feature codes for breaklines may be made between successive FILEIN commands.

TRIEDIT and output to IFF files

TRIEDIT offers three output options which result in the generation of IFF files:

- The IFF command causes output of the triangulated data to an IFF file as 2D and 3D IFF strings. Wherever possible the IFF strings will represent those input for triangulation by TRIANG or TRIEDIT earlier in the flowline. However, the original strings may be broken in many places due to the data thinning and string constraint rules applied in TRIANG and as a result of supplementary data insertion within TRIEDIT itself.

  The layer and feature code attributes of the output IFF file may be set by the user using the SET command during the TRIEDIT session prior to the issue of the IFF command. These attributes are often complex and it may be simpler to have an appropriate series of SET commands in a TRIEDIT command file which can be executed within TRIEDIT using the @file-spec facility (see commands section).

  For further details of the characteristics of IFF files generated by the IFF command and the TRIEDIT startup feature code defaults see the commands section.

- The DRAW CONTOURS command generates an IFF file containing contours derived from the currently windowed triangles.

  Two different feature codes are used to differentiate the contour types.

  For further details see the DRAW CONTOURS command.

- The DRAW TRIANGLES command generates an IFF file containing triangles derived from the currently display window.

  Three different feature codes are used to differentiate triangle data types.

For further details see the DRAW TRIANGLES command.

TRIEDIT and <Ctrl/C> handling

TRIEDIT contains a <Ctrl/C> handler. This means that certain modes of TRIEDIT processing may be terminated by holding down the Control key and typing C. This is useful if, for example, the user has specified labelling (DRAW LABELS) of a large triangulation and then realises how long this is going to take! By using <Ctrl/C> the label drawing is interrupted and TRIEDIT returns to the TRIEDIT> prompt. TRIEDIT is ready to accept a new command.

No harm will ensue from using <Ctrl/C> at any time during a TRIEDIT session. Receipt of a <Ctrl/C> at an inappropriate time will be ignored.

<Ctrl/C> will currently terminate execution of the following commands:

- DRAW LABELS
- DRAW NODES
- DRAW STRINGS
- DRAW TRIANGLES

If output is being directed to an IFF file, the file will be correctly terminated and closed, but of course no guarantee can be given with regard to validity or completeness of the file contents.

TRIEDIT and graphics input from digitising table

TRIEDIT supports two graphic input options from digitising table:

- Laser-Scan MUART controlled digitising table
- Laser-Scan TABLE MONITOR controlled digitising table

The digitising table option selection may either be made during program initialisation (either explicitly on the command line or via the default options lookup file), or via an ENABLE TABLE or ENABLE MUART_TABLE commmand during an edit session.

If a digitising table is chosen as an option the user will be asked to set up the table-to-map-space transformation. As DTMCREATE has no concept of fiducial location, only of extent, it is necessary to supply the coordinates of the fiducial marks to be used.

To provide fiducial values, when prompted the user can either type the X Y map coordinates for each fiducial in turn and then digitise that location, or supply a command file which contains the pre-typed fiducial coordinates in the order NW, SW, SE and NE, one X,Y pair per line of command file. The command file is specified by using the @file-spec mechanism when prompted for the NW fiducial position. Missing parts of the file specification will be taken from the default SYS$DISK:[].DAT.

If a command file is used TRIEDIT will then prompt the user to digitise each corner in turn. Clearly it is advantageous to use a corner point command file if the map area is likely to need several edit sessions, or if the X Y coordinates are in ground metres which tend to be prone to typing errors!

An indirect corner point command file must obey the following rules

1. The corner point coordinates must be in the order NW SW SE NW

2. There must be only one (complete) x,y coordinate per record

3. Comments must be delimited by a "!". Any characters to the right of a "!" will be ignored

An example indirect corner point file is shown below:

```
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
!TRIEDIT digitising table corner point file
! NW
58293.03 13645.37
! SW
58288.35 4331.02
! SE
67729.95 4348.8
! NE
67702.02 13656.67
```

ERROR MESSAGES RELATED TO DIGITISING TABLE INITIALISATION

*** WARNING *** Error assigning LSL$TK

While attempting to initialise the digitising table TRIEDIT was unable to assign the device assigned to the logical name LSL$TK.

Check that that no-one else already has LSL$TK allocated.

*** WARNING *** Can't open indirect command file

The indirect corner point command file that you have specified with the @'file-spec' command cannot be opened.

Ensure that the file exists or that you have privilege to read it.

*** WARNING *** Setup errors unacceptable. Please try again.

After calculating the four point transform for the digitising table setup TRIEDIT checks that the residual errors are acceptable (>0.5% of maximum sidelength).

Digitise more accurately next time or check the corner point values supplied to the program.

*** WARNING *** Error reading CP file. Returning to main loop.

*** WARNING *** Unexpected end of CP file. Returning to main loop

The indirect command file containing the corner point values for the current map has something wrong with it, e.g. an alphabetic character in the middle of a coordinate, only one number in a record, etc.

*** WARNING *** Error reading from table. Returning to main loop.

TRIEDIT is unable to decipher the signals (if any) from the table.

TRIEDIT commands

@

Take command input from the specified file.

FORMAT: @file-spec

Command parameters:

file-spec

The file to be opened and used for command input.

Any parts of the file-spec not supplied for the @ command will be taken from the default specification 'SYS$DISK:[].COM;0'.

DESCRIPTION:

TRIEDIT offers the facility of command input from an indirect command file. The '@' character preceding a file-spec will cause TRIEDIT to open and read commands from the specified file until:

1. a RETURN command is detected and command input is returned to SYS$COMMAND.

2. end-of-file is detected. This provokes an error message and command input is returned to SYS$COMMAND.

Nested command files are not supported (i.e. a command file containing an '@' command), although sequential '@' commands are supported when read from SYS$COMMAND.

As an aid to batch log interpretation, TRIEDIT will echo all commands read from an indirect command file.

Messages:

The following messages are specific to the @ command:

*** WARNING *** "@" must precede a file-spec

*** WARNING *** Indirect file error - returning to terminal input

*** ERROR *** Can't open indirect command file 'file-spec'

Examples:

```
TRIEDIT> @FLOW2<CR>
TRIEDIT> ENABLE DIAGNOSTICS
TRIEDIT> FRT FLOW2
FRT file LSL$FRT:FLOW2.FRT;8 opened for read
TRIEDIT> DESELECT FC OUTCROPS,7,COAST
TRIEDIT> RETURN
TRIEDIT>
```

!

Treat all text to the right of the '!' as a comment.

FORMAT: ! [comment text]

Command parameters:

comment text

text that is to be treated as a comment and which will be excluded from command interpretation.

DESCRIPTION:

An exclamation mark is the standard DTM package comment delimiter. All text (and numbers) which lie to the right of a '!' character are excluded from command interpretation. Comments are useful for annotating command procedures used in batch processing, etc.

Messages: None.

Examples:

TRIEDIT> **! a comment for the sake of it<CR>**
TRIEDIT>

ASSIGN BREAKLINE_FC

Specifies the feature codes of IFF features which are to be treated as breaklines when included in the triangulation.

FORMAT: ASSIGN BREAKLINE_FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. ASSIGN BREAKLINE_FC 2:6 will result in the assignment of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIEDIT any valid feature code group names may be used as arguments to the ASSIGN BREAKLINE_FC command.

DESCRIPTION:

The ASSIGN BREAKLINE_FC command complements the DEASSIGN BREAKLINE_FC command. It enables the user to specify the feature codes of IFF features which are to be treated as breaklines when included in the triangulation.

By default no feature codes are assigned for breaklines, rivers etc. and all IFF data that lie within the triangulation bounds are included in the triangulation. Explicit IFF layer or feature codes may be removed from an assignment list by use of the appropriate DEASSIGN command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the ASSIGN BREAKLINE_FC command:

*** WARNING *** You must have read an FRT file to be able to use group names

*** WARNING *** No groups have been defined in the FRT

*** WARNING *** Illegal feature code 'integer'

*** WARNING *** Bad group name 'group-name'

Examples:

TRIEDIT> **ASSIGN BREAKLINE_FC 6:9,WATER,126<CR>**
TRIEDIT>

ASSIGN BREAKLINE_LAYER

Specifies the IFF layer numbers containing features which are to be treated as breaklines when included in the triangulation.

FORMAT: ASSIGN BREAKLINE_LAYER layer[,...]

Command parameters:

layer

An IFF layer number which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. ASSIGN BREAKLINE_LAYER 2:6 will result in the assignment of layers 2,3,4,5 and 6.

DESCRIPTION:

The ASSIGN BREAKLINE_LAYER command complements the DEASSIGN BREAKLINE_LAYER command. It enables the user to specify the numbers of IFF layers containing features which are to be treated as breaklines when included in the triangulation.

By default no layers are assigned for breaklines, rivers etc. and all IFF data that lie within the triangulation bounds are included in the triangulation. Explicit IFF layer or feature codes may be removed from a assignment list by use of the appropriate DEASSIGN command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the ASSIGN BREAKLINE_LAYER command:

*** WARNING *** Too many layer arguments in one command

*** WARNING *** Illegal layer number 'integer'

Examples:

TRIEDIT> **ASSIGN BREAKLINE_LAYER 21:29,126<CR>**

```
TRIEDIT>
```

ASSIGN RIDGE_FC

Specifies the IFF feature codes of features which are to be flagged as ridgelines when included in the triangulation.

FORMAT: ASSIGN RIDGE_FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. ASSIGN RIDGE_FC 2:6 will result in the assignment of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIEDIT any valid feature code group names may be used as arguments to the ASSIGN RIDGE_FC command.

DESCRIPTION:

The ASSIGN RIDGE_FC command complements the DEASSIGN RIDGE_FC command. ASSIGN RIDGE_FC enables the user to add specified IFF feature codes to the list of those assigned for use as ridgelines within the triangulation.

By default no feature codes are assigned for ridgelines, rivers etc. and all IFF data that lie within the triangulation bounds are included in the triangulation. Explicit IFF layer or feature codes may be removed from a assignment list by use of the appropriate DEASSIGN command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the ASSIGN RIDGE_FC command:

*** WARNING *** You must have read an FRT file to be able to use group names

*** WARNING *** No groups have been defined in the FRT

*** WARNING *** Illegal feature code 'integer'

*** WARNING *** Bad group name 'group-name'

Examples:

TRIEDIT> **ASSIGN RIDGE_FC 6:9,WATER,126<CR>**
TRIEDIT>

ASSIGN RIDGE_LAYER

Specifies the IFF layer numbers containing features which are to be treated as ridgelines when included in the triangulation.

FORMAT: ASSIGN RIDGE_LAYER layer[,...]

Command parameters:

layer

An IFF layer number which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. ASSIGN RIDGE_LAYER 2:6 will result in the assignment of layers 2,3,4,5 and 6 to identify ridgelines.

DESCRIPTION:

The ASSIGN RIDGE_LAYER command complements the DEASSIGN RIDGE_LAYER command. ASSIGN RIDGE_LAYER enables the user to add specified layers to the list of layers that have been assigned for interpretation as ridgelines.

By default no layers are assigned for breaklines, rivers etc. and all IFF data that lie within the triangulation bounds are included in the triangulation. Explicit IFF layer or feature codes may be removed from a assignment list by use of the appropriate DEASSIGN command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the ASSIGN RIDGE_LAYER command:

*** WARNING *** Too many layer arguments in one command

*** WARNING *** Illegal layer number 'integer'

Examples:

TRIEDIT> **ASSIGN RIDGE_LAYER 21:29,126<CR>**
TRIEDIT>

ASSIGN RIVER_FC

Specifies the IFF feature codes of features which are to be flagged as rivers when included in the triangulation.

FORMAT: ASSIGN RIVER_FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. ASSIGN RIVER_FC 2:6 will result in the assignment of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIEDIT any valid feature code group names may be used as arguments to the ASSIGN RIVER_FC command.

DESCRIPTION:

The ASSIGN RIVER_FC command complements the DEASSIGN RIVER_FC command. ASSIGN RIVER_FC enables the user to add specified IFF feature codes to the list of those assigned for use as rivers within the triangulation.

By default no feature codes are assigned for rivers etc. and all IFF data that lie within the triangulation bounds are included in the triangulation. Explicit IFF layer or feature codes may be removed from a assignment list by use of the appropriate DEASSIGN command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the ASSIGN FC command:

*** WARNING *** You must have read an FRT file to be able to use group names
*** WARNING *** No groups have been defined in the FRT

*** WARNING *** Illegal feature code 'integer'

*** WARNING *** Bad group name 'group-name'

Examples:

TRIEDIT> **ASSIGN RIVER_FC 6:9,WATER,126<CR>**
TRIEDIT>

ASSIGN RIVER_LAYER

Specifies the IFF layer numbers containing features which are to be flagged as rivers when included in the triangulation.

FORMAT: ASSIGN RIVER_LAYER layer[,...]

Command parameters:

layer

An IFF layer number which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. ASSIGN RIVER_LAYER 2:6 will result in the assignment of layers 2,3,4,5 and 6 to identify rivers.

DESCRIPTION:

The ASSIGN RIVER_LAYER command complements the DEASSIGN RIVER_LAYER command. ASSIGN RIVER_LAYER enables the user to remove specified layers from the list of layers that have been assigned for interpretation as rivers during triangulation formation.

By default no layers are assigned for breaklines, rivers etc. and all IFF data that lie within the triangulation bounds are included in the triangulation. Explicit IFF layer or feature codes may be removed from a assignment list by use of the appropriate DEASSIGN command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the ASSIGN RIVER_LAYER command:

*** WARNING *** Too many layer arguments in one command

*** WARNING *** Illegal layer number 'integer'

Examples:

TRIEDIT> **ASSIGN RIVER_LAYER 21:29,126<CR>**
TRIEDIT>

CHANGE NODE FEATURE_FLAG

Enables the feature flag of a selected node to be changed to the specified flag.

FORMAT: **CHANGE NODE FEATURE_FLAG 'flag'**

Command parameters:

flag

The new feature flag for the node, chosen from:

1. UNFLAGGED
2. RIDGELINE
3. RIVER

DESCRIPTION:

The CHANGE NODE FEATURE_FLAG command enables the feature flag of a selected node to be changed to the specified flag. After issuing the CHANGE NODE FEATURE_FLAG command the user must position the cursor over a node and press the rightmost function button. TRIEDIT will draw a symbol over the changed node.

Messages:

The following warning messages are specific to the CHANGE NODE FEATURE_FLAG command:

*** WARNING *** Missing CHANGE NODE FEATURE_FLAG argument

*** ERROR *** reading CHANGE NODE FEATURE_FLAG argument

Examples:

TRIEDIT> **CHANGE NODE FEATURE_FLAG RIVER<CR>**
TRIEDIT>

CHANGE NODE HEIGHT

Enables the height of a selected node to be changed to the specified value.

FORMAT: **CHANGE NODE HEIGHT 'height'**

Command parameters:

height

The new height for the node (floating point).

DESCRIPTION:

The CHANGE NODE HEIGHT command enables the height of a selected node to be changed to the specified value. After issuing the CHANGE NODE HEIGHT command the user must position the cursor over a node and press the rightmost function button. TRIEDIT will draw a symbol over the changed node.

Messages:

The following warning messages are specific to the CHANGE NODE HEIGHT command:

*** WARNING *** Missing CHANGE NODE HEIGHT argument

*** ERROR *** reading CHANGE NODE HEIGHT argument

Examples:

TRIEDIT> **CHANGE NODE HEIGHT 458.9<CR>**
TRIEDIT>

CHANGE NODE TYPE

Enables the data type of a selected node to be changed to the specified type.

FORMAT: CHANGE NODE TYPE 'type'

Command parameters:

type

The new data type for the node, chosen from:

1. NORMAL

2. BREAKLINE

DESCRIPTION:

The CHANGE NODE TYPE command enables the data type of a selected node to be changed to the specified type. After issuing the CHANGE NODE TYPE command the user must position the cursor over a node and press the rightmost function button. TRIEDIT will draw a symbol over the changed node.

Nodes can have one of three data types within DTMCREATE programs:

1. NORMAL - slope derivatives are continuous at the node

2. BREAKLINE - slope derivatives are discontinuous at the node

3. CLIFF - the node is part of a cliffline. The top and bottom of the cliff are treated as perfectly superimposed breaklines.

TRIEDIT currently supports the insertion (either interactive or from IFF file), deletion and change of normal nodes and breakline nodes. Cliffline nodes cannot be manipulated in any way at present.

Messages:

The following warning messages are specific to the CHANGE NODE TYPE command:

*** WARNING *** Unexpected end of CHANGE NODE TYPE command
Specify either NORMAL or BREAKLINE

Examples:

TRIEDIT> **CHANGE NODE TYPE BREAKLINE<CR>**
TRIEDIT>

CHANGE STRING FEATURE_FLAG

Enables the feature flag of a whole selected string of nodes to be changed to the specified flag.

FORMAT: **CHANGE STRING FEATURE_FLAG 'flag'**

Command parameters:

flag

The new feature flag for all the nodes in the string, chosen from:

1. UNFLAGGED
2. RIDGELINE
3. RIVER

DESCRIPTION:

The CHANGE STRING FEATURE_FLAG command enables the feature flag of all the nodes in a selected string to be changed to the specified flag. After issuing the CHANGE STRING FEATURE_FLAG command the user must position the cursor over a node in the string and press the rightmost function button. TRIEDIT will draw a symbol over all the changed nodes in the string.

Messages:

The following warning messages are specific to the CHANGE STRING FEATURE_FLAG command:

*** WARNING *** Missing CHANGE STRING FEATURE_FLAG argument

*** ERROR *** reading CHANGE STRING FEATURE_FLAG argument

Examples:

TRIEDIT> **CHANGE STRING FEATURE_FLAG RIVER<CR>**
TRIEDIT>

CHANGE STRING HEIGHT

Enables the height of all nodes in a selected string to be changed to the specified value.

FORMAT: **CHANGE STRING HEIGHT 'height'**

Command parameters:

height

The new height for all the nodes in the string (floating point).

DESCRIPTION:

The CHANGE STRING HEIGHT command enables the height of all the nodes in a selected string to be changed to the specified value. After issuing the CHANGE STRING HEIGHT command the user must position the cursor over a node in the string and press the rightmost function button. TRIEDIT will draw a symbol over the changed nodes in the string.

Messages:

The following warning messages are specific to the CHANGE STRING HEIGHT command:

*** WARNING *** Missing CHANGE STRING HEIGHT argument

*** ERROR *** reading CHANGE STRING HEIGHT argument

Examples:

TRIEDIT> **CHANGE STRING HEIGHT 458.9<CR>**
TRIEDIT>

CHANGE STRING TYPE

Enables the data type of all the nodes in a selected STRING to be changed to the specified type.

FORMAT: **CHANGE STRING TYPE 'type'**

Command parameters:

type

The new data type for all the nodes in the string, chosen from:

1. NORMAL

2. BREAKLINE

DESCRIPTION:

The CHANGE STRING TYPE command enables the data type of all the nodes in a selected string to be changed to the specified type. After issuing the CHANGE STRING TYPE command, the user must position the cursor over a node in the string and press the rightmost function button. TRIEDIT will draw a symbol over the changed nodes in the string.

Nodes can have one of three data types within DTMCREATE programs:

1. NORMAL - slope derivatives are continuous at the node

2. BREAKLINE - slope derivatives are discontinuous at the node

3. CLIFF - the node is part of a cliffline. The top and the bottom of the cliff are treated as perfectly superimposed breaklines.

TRIEDIT currently supports the insertion (either interactive or from IFF file), deletion and change of normal nodes and breakline nodes. Cliffline nodes cannot be manipulated in any way at present.

Messages:

The following warning messages are specific to the CHANGE STRING TYPE command:

*** WARNING *** Unexpected end of CHANGE STRING TYPE command

Specify either NORMAL or BREAKLINE

Examples:

```
TRIEDIT> CHANGE STRING TYPE BREAKLINE<CR>
TRIEDIT>
```

CLEAR

Clear the graphics display and update the status area.

FORMAT: CLEAR

Command parameters: None.

DESCRIPTION:

The CLEAR command causes the graphics display to be cleared and the status area to be updated.

The user may optionally select automatic redrawing of various information on the graphics display immediately after a CLEAR command. The automatic redrawing options are:

- ENABLE DCUPDATE - redraw the calculated contours after a CLEAR command
- ENABLE DLUPDATE - redraw the selected labelling after a CLEAR command
- ENABLE DNUPDATE - redraw the nodes after a CLEAR command
- ENABLE DTUPDATE - redraw the triangulation after a CLEAR command

Once enabled these options remain active until explicitly disabled with the appropriate DISABLE command.

It is common for the user to wish to see the triangulation after each CLEAR command. The ENABLE DTUPDATE command will save having to issue an explicit DRAW TRIANGLES command after each CLEAR command.

Messages: None.

Examples:

TRIEDIT> **ENABLE DTUPDATE<CR>**
TRIEDIT> **CLEAR<CR>**
TRIEDIT> **INTERVAL 10<CR>**
TRIEDIT> **DRAW CONTOURS<CR>**
TRIEDIT> **CLEAR<CR>**
TRIEDIT>

DATUM

Enables specification of a height datum to be added to IFF heights read using subsequent FILEIN commands.

FORMAT: DATUM value

Command parameters:

value

A floating-point height value which is to be added to IFF heights read using subsequent FILEIN commands. DATUM values are expressed relative to the contents of the IFF file to be read in.

DESCRIPTION:

The DATUM command enables specification of a height datum to be added to IFF heights read using subsequent FILEIN commands.

On program statup a DATUM default value of 0.0 is assumed.

Height information within a single IFF file may be stored relative to different height data providing that the features pertaining to each height datum can be distinguished by feature code or layer. The DATUM command can be used to set the height datum for a subsequent FILEIN command. If required a single IFF file may be read in many times relative to different height data, feature selection being achieved by SELECT FC and SELECT LAYER commands.

If the INVERSE command has been used to specify height inversion the heights are inverted before the datum value is added.

Messages:

The following messages are specific to the DATUM command:

*** ERROR *** You must specify a floating point argument to the DATUM command
*** WARNING *** You must specify a floating point argument to the DATUM command

---

Examples:

TRIEDIT> **DATUM 8.2<CR>**
TRIEDIT>

DEASSIGN BREAKLINE_FC

Deassigns specified IFF feature codes from the list of feature codes identifying breaklines to be included in the triangulation.

FORMAT: DEASSIGN BREAKLINE_FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature
codes may be specified by separating the range start and stop values by a colon
e.g. DEASSIGN BREAKLINE_FC 2:6 will result in the deassignment of feature codes
2,3,4,5 and 6.

If an FRT file has been read into TRIEDIT any valid feature code group names may be used as arguments to the DEASSIGN BREAKLINE_FC command.

DESCRIPTION:

The DEASSIGN BREAKLINE_FC command complements the ASSIGN BREAKLINE_FC command. DEASSIGN BREAKLINE_FC enables the user to remove specified feature codes from the list of feature codes that have been assigned for interpretation as breaklines.

By default no feature codes are assigned for breaklines, rivers etc. and all IFF data that lie within the triangulation bounds are included in the triangulation.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the DEASSIGN BREAKLINE_FC command:

*** WARNING *** You must have read an FRT file to be able to use group names

*** WARNING *** No groups have been defined in the FRT

*** WARNING *** Illegal feature code 'integer'

*** WARNING *** Bad group name 'group-name'

Examples:

TRIEDIT> **DEASSIGN BREAKLINE_FC 6:9,CLIFFS,126<CR>**
TRIEDIT>

DEASSIGN BREAKLINE_LAYER

Deassigns specified IFF layer numbers from the list of layer numbers identifying breaklines to be included in the triangulation.

FORMAT: DEASSIGN BREAKLINE_LAYER layer[,...]

Command parameters:

layer

An IFF layer number which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. DEASSIGN BREAKLINE_LAYER 2:6 will result in the deassignment of layers 2,3,4,5 and 6.

DESCRIPTION:

The DEASSIGN BREAKLINE_LAYER command complements the ASSIGN BREAKLINE_LAYER command. DEASSIGN BREAKLINE_LAYER enables the user to remove specified layers from the list of layers that have been assigned for interpretation as breaklines.

By default no feature codes are assigned for breaklines, rivers etc. and all IFF data that lie within the triangulation bounds are included in the triangulation.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the DEASSIGN BREAKLINE_LAYER command:

*** WARNING *** Too many layer arguments in one command

*** WARNING *** Illegal layer number 'integer'

Examples:

TRIEDIT> **DEASSIGN BREAKLINE_LAYER 21:29,126<CR>**
TRIEDIT>

DEASSIGN RIDGE_FC

Deassigns specified IFF feature codes from the list of feature codes identifying ridgelines to be included in the triangulation.

FORMAT: DEASSIGN RIDGE_FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. DEASSIGN RIDGE_FC 2:6 will result in the deassignment of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIEDIT any valid feature code group names may be used as arguments to the DEASSIGN RIDGE_FC command.

DESCRIPTION:

The DEASSIGN RIDGE_FC command complements the ASSIGN RIDGE_FC command. DEASSIGN RIDGE_FC enables the user to remove specified IFF feature codes from the list of those assigned for use as ridgelines within the triangulation.

By default no feature codes are assigned for breaklines, rivers etc. and all IFF data that lie within the triangulation bounds are included in the triangulation.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the DEASSIGN FC command:

*** WARNING *** You must have read an FRT file to be able to use group names

*** WARNING *** No groups have been defined in the FRT

*** WARNING *** Illegal feature code 'integer'

*** WARNING *** Bad group name 'group-name'

Examples:

```
TRIEDIT> DEASSIGN RIDGE_FC 6:9,WATER,126<CR>
TRIEDIT>
```

DEASSIGN RIDGE_LAYER

Deassigns specified IFF layer numbers from the list of layer numbers identifying rivers to be included in the triangulation.

FORMAT: DEASSIGN RIDGE_LAYER layer[,...]

Command parameters:

layer

An IFF layer which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. DEASSIGN RIDGE_LAYER 2:6 will result in the deassignment of layers 2,3,4,5 and 6.

DESCRIPTION:

The DEASSIGN RIDGE_LAYER command complements the ASSIGN RIDGE_LAYER command. DEASSIGN RIDGE_LAYER enables the user to remove specified layers from the list of layers that have been assigned for interpretation as ridgelines during triangulation formation.

By default no feature codes are selected for breaklines, rivers etc. and all IFF data that lie within the triangulation bounds are included in the triangulation.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the DEASSIGN RIDGE_LAYER command:

*** WARNING *** Too many layer arguments in one command

*** WARNING *** Illegal layer number 'integer'

Examples:

TRIEDIT> **DEASSIGN RIDGE_LAYER 21:29,126<CR>**
TRIEDIT>

DEASSIGN RIVER_FC

Deassigns specified IFF feature codes from the list of feature codes identifying rivers to be included in the triangulation.

FORMAT: DEASSIGN RIVER_FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. DEASSIGN RIVER_FC 2:6 will result in the deassignment of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIEDIT any valid feature code group names may be used as arguments to the DEASSIGN RIVER_FC command.

DESCRIPTION:

The DEASSIGN RIVER_FC command complements the ASSIGN RIVER_FC command. DEASSIGN RIVER_FC enables the user to remove specified IFF feature codes from the list of those assigned for use as rivers within the triangulation.

By default no feature codes are assigned for breaklines, rivers etc. and all IFF data that lie within the triangulation bounds are included in the triangulation.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the DEASSIGN FC command:

*** WARNING *** You must have read an FRT file to be able to use group names

*** WARNING *** No groups have been defined in the FRT

*** WARNING *** Illegal feature code 'integer'

*** WARNING *** Bad group name 'group-name'

Examples:

TRIEDIT> **DEASSIGN RIVER_FC 6:9,WATER,126<CR>**
TRIEDIT>

DEASSIGN RIVER_LAYER

Deassigns specified IFF layer numbers from the list of layer numbers identifying rivers to be included in the triangulation.

FORMAT: DEASSIGN RIVER_LAYER layer[,...]

Command parameters:

layer

An IFF layer which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. DEASSIGN RIVER_LAYER 2:6 will result in the deassignemnt of layers 2,3,4,5 and 6.

DESCRIPTION:

The DEASSIGN RIVER_LAYER command complements the ASSIGN RIVER_LAYER command. DEASSIGN RIVER_LAYER enables the user to remove specified layers from the list of layers that have been selected for interpretation as rivers during triangulation formation.

By default no feature codes are assigned for breaklines, rivers etc. and all IFF data that lie within the triangulation bounds are included in the triangulation.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the DEASSIGN RIVER_LAYER command:

*** WARNING *** Too many layer arguments in one command

*** WARNING *** Illegal layer number 'integer'

Examples:

TRIEDIT> **DEASSIGN RIVER_LAYER 21:29,126<CR>**
TRIEDIT>

DELETE

Delete the selected string of nodes.

FORMAT: DELETE

Command parameters: None.

DESCRIPTION:

The DELETE command enables the user to delete a whole string of selected nodes from the triangulation. After issuing the DELETE command, the user must position the cursor over a node in the string and press the rightmost function button. TRIEDIT will draw a symbol over all the nodes deleted from the string.

The "hole" left in the triangulation by the deletion will be automatically repaired with the most equilateral triangulation possible given the shape of the hole and the distribution of surrounding nodes.

It is not possible to delete imaginary or cliffline nodes.

Messages:

The following warning messages are specific to the DELETE command:

*** WARNING *** You cannot DELETE an imaginary node

*** WARNING *** You cannot DELETE a cliffline node

Examples:

TRIEDIT> **ENABLE DTUPDATE<CR>**
TRIEDIT> **CLEAR<CR>**
TRIEDIT> **DELETE<CR>**
TRIEDIT> **CLEAR<CR>**
TRIEDIT>

DESELECT FC

Deselects specified IFF feature codes. Any IFF features having the specified feature codes will be excluded from the triangulation.

FORMAT: DESELECT FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. DESELECT FC 2:6 will result in the deselection of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIEDIT any valid feature code group names may be used as arguments to the DESELECT FC command.

DESCRIPTION:

The DESELECT FC command complements the SELECT FC command. DESELECT FC enables the user to prevent TRIEDIT from reading in any IFF features which have the specified feature codes.

On program startup all FCs are selected for input.

Specific FC selections may then be made with the SELECT FC command. Only the sepcified FCs will be used for input. All FCs not explicitly specified in a SELECT FC command will then be excluded from input.

All FC and layer selections are cancelled by the SELECT ALL command; i.e. all layers and FCs are reselected for input.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers.

Messages:

The following warning messages are specific to the DESELECT FC command:

*** WARNING *** You must have read an FRT file to be able to use group names

*** WARNING *** No groups have been defined in the FRT

*** WARNING *** Illegal feature code 'integer'

*** WARNING *** Bad group name 'group-name'

Examples:

TRIEDIT> **DESELECT FC 6:9,WATER,126<CR>**
TRIEDIT>

DESELECT LAYER

Deselects specified IFF layers. Any IFF features lying within the specified layers will be excluded from the triangulation.

FORMAT: DESELECT LAYER layer[,...]

Command parameters:

layer

An IFF layer number which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. DESELECT LAYER 2:6 will result in the deselection of layers 2,3,4,5 and 6.

DESCRIPTION:

The DESELECT LAYER command complements the SELECT LAYER command. DESELECT LAYER enables the user to prevent TRIEDIT from reading in any IFF features which lie within the specified layers.

On program startup all layers are selected for input.

Specific layer selections may then be made with the SELECT LAYER command. Only the sepcified layers will be used for input. All layers not explicitly specified in SELECT LAYER commands will then be excluded from input.

All FC and layer selections are cancelled by the SELECT ALL command; i.e. all layers and FCs are reselected for input.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. The use can use the SHOW SELECTIONS command to examine current input selections and SHOW ASSIGNMENTS to examine current assignments.

Messages:

The following warning messages are specific to the DESELECT LAYER command:

*** WARNING *** Too many layer arguments in one command

*** WARNING *** Illegal layer number 'integer'

Examples:

TRIEDIT> **DESELECT LAYER 21:29,126<CR>**
TRIEDIT>

DISABLE BITPAD

Disables a previous ENABLE BITPAD command.

FORMAT: DISABLE BITPAD

Command parameters: None.

DESCRIPTION:

DISABLE BITPAD allows the user to disable a previous ENABLE BITPAD command. The bitpad will no longer be available as a graphics input device.

Messages: None.

Examples:

TRIEDIT> **DISABLE BITPAD<CR>**
TRIEDIT>

DISABLE DCUPDATE

Disables a previous ENABLE DCUPDATE command.

FORMAT: DISABLE DCUPDATE

Command parameters: None.

DESCRIPTION:

DISABLE DCUPDATE allows the user to disable a previous ENABLE DCUPDATE command. Contours will no longer be automatically drawn on the graphics screen after every CLEAR command.

Messages: None.

Examples:

TRIEDIT> **ENABLE DCUPDATE<CR>**
TRIEDIT> **DISABLE DCUPDATE <CR>**
TRIEDIT>

DISABLE DIAGNOSTICS

Disables a previous ENABLE DIAGNOSTICS command.

FORMAT: DISABLE DIAGNOSTICS

Command parameters: None.

DESCRIPTION:

DISABLE DIAGNOSTICS allows the user to disable a previous ENABLE DIAGNOSTICS command.

Messages: None.

Examples:

```
TRIEDIT> ENABLE DIAGNOSTICS<CR>
TRIEDIT> SHOW ENABLE<CR>
DIAGNOSTICS ..................................................... On
DIVIDEBY ........................................................ Off
INTEGER_HEIGHT .................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ......................................................... Off
MULTIPLY ........................................................ Off
PME ............................................................. Off
TOFEET .......................................................... Off
TOMETRES ........................................................ Off
TRIEDIT>
```

DISABLE DIVIDEBY

Disables a previous ENABLE DIVIDEBY command.

FORMAT: DISABLE DIVIDEBY

Command parameters: None.

DESCRIPTION:

DISABLE DIVIDEBY allows the user to disable a previous ENABLE DIVIDEBY command. Heights will no longer divided on read-in using the FILEIN command.

Messages: None.

Examples:

TRIEDIT> **DISABLE DIVIDEBY<CR>**
TRIEDIT>

DISABLE DLUPDATE

Disables a previous ENABLE DLUPDATE command.

FORMAT: DISABLE DLUPDATE

Command parameters: None.

DESCRIPTION:

DISABLE DLUPDATE allows the user to disable a previous ENABLE DLUPDATE command. Labels will no longer be automatically drawn on the graphics screen after every CLEAR command.

Messages: None.

Examples:

TRIEDIT> **ENABLE DLUPDATE<CR>**
TRIEDIT> **DISABLE DLUPDATE <CR>**
TRIEDIT>

DISABLE DNUPDATE

Disables a previous ENABLE DNUPDATE command.

FORMAT: DISABLE DNUPDATE

Command parameters: None.

DESCRIPTION:

DISABLE DNUPDATE allows the user to disable a previous ENABLE DNUPDATE command. Node locations will no longer be automatically drawn on the graphics screen after every CLEAR command.

Messages: None.

Examples:

TRIEDIT> **ENABLE DNUPDATE<CR>**
TRIEDIT> **DISABLE DNUPDATE <CR>**
TRIEDIT>

DISABLE DTUPDATE

Disables a previous ENABLE DTUPDATE command.

FORMAT: DISABLE DTUPDATE

Command parameters: None.

DESCRIPTION:

DISABLE DTUPDATE allows the user to disable a previous ENABLE DTUPDATE command. Triangles will no longer be automatically drawn on the graphics screen after every CLEAR command.

Messages: None.

Examples:

TRIEDIT> **ENABLE DTUPDATE<CR>**
TRIEDIT> **DISABLE DTUPDATE <CR>**
TRIEDIT>

DISABLE INTEGER_HEIGHT

Disables the effect of a previous ENABLE INTEGER_HEIGHT command.

FORMAT: **DISABLE INTEGER_HEIGHT**

Command parameters: None.

DESCRIPTION:

In IFF files height values are transmitted via AC (Ancillary Code) or ZS (3D string) entries. By default contour and spot height Z values are read from type 3 ACs as floating point values. By use of the ENABLE INTEGER_HEIGHT command, integer heights may be read from type 2 ACs.

The DISABLE INTEGER_HEIGHT command disables the effect of a previous ENABLE INTEGER_HEIGHT command and heights will be read from type 3 (floating point) ACs.

Messages: None.

Examples:

TRIEDIT> **DISABLE INTEGER_HEIGHT<CR>**
TRIEDIT>

DISABLE INVERSE

Disables a previous ENABLE INVERSE command.

FORMAT: DISABLE INVERSE

Command parameters: None.

DESCRIPTION:

The DISABLE INVERSE command disables a previous ENABLE INVERSE command.

By default heights are not inverted.

Messages: None.

Examples:

TRIEDIT> **DISABLE INVERSE <CR>**
TRIEDIT>

DISABLE JOYSTICK

Disables a previous ENABLE JOYSTICK command.

FORMAT: DISABLE JOYSTICK

Command parameters: None.

DESCRIPTION:

DISABLE JOYSTICK allows the user to disable a previous ENABLE JOYSTICK command. The JOYSTICK will no longer be available as a graphics input device.

Messages: None.

Examples:

TRIEDIT> **DISABLE JOYSTICK<CR>**
TRIEDIT>

DISABLE MULTIPLYBY

Disables a previous ENABLE MULTIPLYBY command.

FORMAT: DISABLE MULTIPLYBY

Command parameters: None.

DESCRIPTION:

DISABLE MULTIPLYBY allows the user to disable a previous ENABLE MULTIPLYBY command. Heights will no longer by multiplied on read-in using the FILEIN command.

Messages: None.

Examples:

TRIEDIT> **DISABLE MULTIPLYBY<CR>**
TRIEDIT>

DISABLE MOUSE

Disables a previous ENABLE MOUSE command.

FORMAT: DISABLE MOUSE

Command parameters: None.

DESCRIPTION:

DISABLE MOUSE allows the user to disable a previous ENABLE MOUSE command. The MOUSE will no longer be available as a graphics input device.

Messages: None.

Examples:

TRIEDIT> **DISABLE MOUSE<CR>**
TRIEDIT>

DISABLE MUART_TABLE

Disables a previous ENABLE MUART_TABLE command.

FORMAT: **DISABLE MUART_TABLE**

Command parameters: None.

DESCRIPTION:

DISABLE MUART_TABLE allows the user to disable a previous ENABLE MUART_TABLE command. The MUART controlled digitising table will no longer be available as a graphics input device.

Messages: None.

Examples:

TRIEDIT> **DISABLE MUART_TABLE<CR>**
TRIEDIT>

DISABLE PME

Disables the effect of a previous ENABLE PME command.

FORMAT: DISABLE PME

Command parameters: None.

DESCRIPTION:

The ENABLE PME and DISABLE PME commands are reserved for Laser-Scan use. PME is a code optimisation tool and should be invoked by LSL software personnel only.

DISABLE PME disables the effect of a previous ENABLE PME command and causes the PME_EXIT routine to be invoked.

Message:

The following warning message is specific to the DISABLE PME command:

*** WARNING *** You were not using PME anyway!

Examples:

TRIEDIT> **ENABLE PME<CR>**
TRIEDIT> **ENABLE DIAGNOSTICS<CR>**
TRIEDIT> **FRT CHASTITY<CR>**
TRIEDIT> **ASSIGN BREAKLINE_FC 1:7,WATER,80,102<CR>**
TRIEDIT> **DISABLE PME<CR>**
TRIEDIT>

DISABLE TABLE

Disables a previous ENABLE TABLE command.

FORMAT: DISABLE TABLE

Command parameters: None.

DESCRIPTION:

DISABLE TABLE allows the user to disable a previous ENABLE TABLE command. The digitising table will no longer be available as a graphics input device.

Messages: None.

Examples:

TRIEDIT> **DISABLE TABLE<CR>**
TRIEDIT>

DISABLE TOFEET

Cancels the effect of a previous ENABLE TOFEET command.

FORMAT: DISABLE TOFEET

Command parameters: None.

DESCRIPTION:

DISABLE TOFEET cancels the effect of a previous ENABLE TOFEET command. Heights will no longer be converted to feet on read-in using the FILEIN command.

N.B. DISABLE TOFEET will not cancel an explicit ENABLE MULTIPLYBY 3.2808455 command.

Messages: None.

Examples:

TRIEDIT> **DISABLE TOFEET<CR>**
TRIEDIT>

DISABLE TOMETRES

Cancels the effect of a previous ENABLE TOMETRES command.

FORMAT: DISABLE TOMETRES

Command parameters: None.

DESCRIPTION:

DISABLE TOMETRES cancels the effect of a previous ENABLE TOMETRES command. Heights will no longer be converted to metres on read-in using the FILEIN command.

N.B. DISABLE TOMETRES will not cancel an explicit ENABLE DIVIDEBY 3.2808455 command.

Messages: None.

Examples:

TRIEDIT> **DISABLE TOMETRES<CR>**
TRIEDIT>

DISABLE THUMBWHEELS

Disables a previous ENABLE THUMBWHEELS command.

FORMAT: DISABLE THUMBWHEELS

Command parameters: None.

DESCRIPTION:

DISABLE THUMBWHEELS allows the user to disable a previous ENABLE THUMBWHEELS command. The thumbwheels will no longer be available as a graphics input device.

Messages: None.

Examples:

TRIEDIT> **DISABLE THUMBWHEELS<CR>**
TRIEDIT>

DISABLE TRACKERBALL

Disables a previous ENABLE TRACKERBALL command.

FORMAT: DISABLE TRACKERBALL

Command parameters: None.

DESCRIPTION:

DISABLE TRACKERBALL allows the user to disable a previous ENABLE TRACKERBALL command. The trackerball will no longer be available as a graphics input device.

Messages: None.

Examples:

TRIEDIT> **DISABLE TRACKERBALL<CR>**
TRIEDIT>

DRAW BREAKLINES

Specifies that all nodes of data type breakline are to be drawn on the graphics screen.

FORMAT: DRAW BREAKLINES

COMMAND PARAMETERS: None.

DESCRIPTION:

Specifies that all nodes of data type breakline are to be drawn on the graphics screen.

Messages: None.

Examples:

TRIEDIT> **DRAW BREAKLINES<CR>**
TRIEDIT>

DRAW CONTOURS

Calculate and display contours from the currently windowed triangles.

FORMAT: DRAW CONTOURS [iff-file-spec]

COMMAND PARAMETERS:

[iff-file-spec]

The specification of an IFF file to which contour drawing is to be directed.

If no file-spec is supplied output is directed to the graphics screen.

If a partial file-spec is supplied, missing parts will be taken from the default 'LSL$IF:IFF.IFF;0'.

DESCRIPTION:

Calculate contours from the currently windowed triangles.

If the optional IFF file-spec argument is supplied, output is directed to the IFF file only.

If no file-spec argument is supplied, output is directed to the graphics screen.

On the graphics screen contours are displayed in two colours. This enables the user to differentiate between index contours (of height specified by the INDEX_INTERVAL command) and intermediate contours (of a height specified by the INTERVAL command).

By default 10% of the triangulation Z range is used as the contour interval. This value is displayed in the status area. If no index interval is specified, no index contours will be drawn. If the intermediate contour interval is zero no intermediate contours will be drawn.

A crude but very fast linear interpolation algorithm is used on each triangle facet plane in turn. The quality of the contouring around the edge of the triangulation can sometimes be improved by raising the level of triangle accuracy with the SET TRIANGLE_ACCURACY command. This may, however, slow other edit operations.

An automatic DRAW CONTOURS command may be selected to occur after every CLEAR command by using the ENABLE DCUPDATE option. Automatic contour output will be directed to the graphics screen only.

The effect of the ENABLE DCUPDATE command will remain active until explicitly disabled with the DISABLE DCUPDATE command.

If output is directed to an IFF file for hardcopy plotting purposes two different feature codes are used to differentiate the contour types.

By default these feature codes are:

Contour type	FC
Intermediate	1
Index	2

These defaults may be altered with the following commands:

Contour type	SET command for FC
Intermediate	SET CONTOUR_FC
Index	SET INDEX_CONTOUR_FC

By default the IFF features will be created in layer 1, or the layer number set with the most recent SET LAYER command.

When plotting to an IFF file only contours for that portion of the triangulation that would be displayed on the graphics screen are sent to the IFF file. If contours for the whole triangulation are to be sent to IFF file use the WINDOW command to set the screen window to cover the whole triangulation area.

IFF coordinates are in the user units supplied to TRIANG.

The IFF file is designed for plotting purposes only. Contour sections are, wherever possible, joined together for efficient storage and plotting. The IFF features have no height tagging.

Messages:

The following warning message is specific to the DRAW CONTOURS command:

*** WARNING *** error reading IFF file-spec

Examples:

TRIEDIT> **DRAW CONTOURS<CR>**
TRIEDIT>

DRAW LABELS

Draw all selected labels on the graphics screen.

FORMAT: DRAW LABELS

COMMAND PARAMETERS: None.

DESCRIPTION:

Specifies that any labels selected with LABEL HEIGHT, LABEL SEQUENCE or LABEL SIGNS commands are to be drawn on the graphics screen.

The current label selections are displayed in the status area.

A label is drawn in a colour which reflects the feature flag of the node to which it relates.

The size of the labels may be altered with the LABEL SMALL (default) and LABEL BIG commands.

If no LABEL options are active (e.g. on program startup or after a LABEL NONE command), a DRAW LABELS command will have no effect.

An automatic DRAW LABELS command may be selected to occur after every CLEAR command by using the ENABLE DLUPDATE option.

The effect of the ENABLE DLUPDATE command will remain active until explicitly disabled with the DISABLE DLUPDATE command.

Messages: None.

Examples:

TRIEDIT> **DRAW LABELS<CR>**
TRIEDIT>

DRAW NODES

Specifies that all nodes of all data types are to be drawn on the graphics screen.

FORMAT: DRAW NODES

COMMAND PARAMETERS: None.

DESCRIPTION:

Specifies that all nodes of all data types are to be drawn on the graphics screen.

An automatic DRAW NODES command may be selected to occur after every CLEAR command by using the ENABLE DNUPDATE option.

The effect of the ENABLE DNUPDATE command will remain active until explicitly disabled with the DISABLE DNUPDATE command.

Messages: None.

Examples:

TRIEDIT> **DRAW NODES<CR>**
TRIEDIT>

DRAW RIDGELINES

Specifies that all nodes flagged as ridgelines are to be drawn on the graphics screen.

FORMAT: DRAW RIDGELINES

COMMAND PARAMETERS: None.

DESCRIPTION:

Specifies that all nodes flagged as ridgelines are to be drawn on the graphics screen. Nodes flagged as being part of ridgelines will be drawn in a different colour to unflagged and river nodes.

Messages: None.

Examples:

TRIEDIT> **DRAW RIDGELINES<CR>**
TRIEDIT>

DRAW RIVERS

Specifies that all nodes flagged as rivers are to be drawn on the graphics screen.

FORMAT: DRAW RIVERS

COMMAND PARAMETERS: None.

DESCRIPTION:

Specifies that all nodes flagged as rivers are to be drawn on the graphics screen. Nodes flagged as being part of rivers will be drawn in a different colour to unflagged and ridgeline nodes.

Messages: None.

Examples:

TRIEDIT> **DRAW RIVERS<CR>**
TRIEDIT>

DRAW STRINGS

Display the connectivity between nodes that is inherited from their position within input strings.

FORMAT: DRAW STRINGS

COMMAND PARAMETERS: None.

DESCRIPTION:

Display the connectivity between nodes that is inherited from their position within input strings.

N.B. Due to the nature of the triangulation constraint process, and of subsequent TRIEDIT edit sessions, original input strings may be fragmented.

Messages: None.

Examples:

TRIEDIT> **DRAW STRINGS<CR>**
TRIEDIT>

DRAW TRIANGLES

Display the triangular connectivity between nodes.

FORMAT: DRAW TRIANGLES [iff-file-spec]

COMMAND PARAMETERS:

[iff-file-spec]

The specification of an IFF file to which triangle drawing is to be directed.

If no file-spec is supplied output is directed to the graphics screen.

If a partial file-spec is supplied, missing parts will be taken from the default 'LSL$IF:IFF.IFF;0'.

DESCRIPTION:

Display (or write to an optional IFF file), the triangular connectivity between nodes.

If the optional IFF file-spec argument is supplied, output is directed to the IFF file only.

If no file-spec argument is supplied, output is directed to the graphics screen.

On the graphics screen triangle links are displayed in three colours. This enables the user to differentiate triangle links that connect normal nodes from nodes that represent locations of slope discontinuity (breaklines and cliffs) and imaginary points (which have location but undefined attributes).

If output is directed to an IFF file for hardcopy plotting purposes three different feature codes are used to differentiate the triangle links.

By default these feature codes are:

Link type	FC
Normal	1
Breakline	2
Imaginary	3

These defaults may be altered with the following commands:

Link type	SET command for FC
Normal	SET LINK_FC
Breakline	SET BREAKLINE_LINK_FC
Imaginary	SET IMAGINARY_LINK_FC

By default the IFF features will be created in layer 1, or the layer number set with the most recent SET LAYER command.

When plotting to IFF file, only that portion of the triangulation that would be displayed on the graphics screen is sent to the IFF file. If the whole triangulation is to be sent to IFF file use the WINDOW command to set the screen window to cover the whole triangulation area.

IFF coordinates are in the user units supplied to TRIANG.

The IFF file is designed for plotting purposes only. Triangle links are, wherever possible, joined together for efficient storage and plotting. The IFF features have no height tagging.

An automatic DRAW TRIANGLES command may be selected to occur after every CLEAR command by using the ENABLE DTUPDATE option. Automatic triangle output will be directed to the graphics screen only.

The effect of the ENABLE DTUPDATE command will remain active until explicitly disabled with the DISABLE DTUPDATE command.

Messages:

The following warning message is specific to the DRAW TRIANGLES command:

*** WARNING *** error reading IFF file-spec

Examples:

TRIEDIT> **DRAW TRIANGLES<CR>**
TRIEDIT>

DUMP

Specifies that new triangulation .DTA and .NOD files are to be created, but that the editing session is to be resumed after file output.

FORMAT: DUMP [file-spec]

COMMAND PARAMETERS:

[file-spec]

The generic specification of the file to be opened for data output.

If a partial file-spec is supplied, missing parts will be taken from the defaults SYS$DISK:[].DTA;0 and SYS$DISK:[].NOD;0

DESCRIPTION:

The DUMP command specifies that new triangulation .DTA and .NOD files are to be created, without leaving the edit session.

Optionally, a generic file-spec may be supplied as an argument to the DUMP command and this will be used to form the output file specifications. If no file-spec command argument is supplied then output will be to files having the same specifications as those used for input at the start of the edit session but with the file version numbers incremented by one.

Messages:

The following message is specific to the DUMP command:

*** ERROR *** reading DUMP file-spec

Examples:

TRIEDIT> **DUMP DUA3:[DEMONSTRATION]IDAHO<CR>**
DUA3:[DEMONSTRATION]IDAHO.DTA;1 opened for write
DUA3:[DEMONSTRATION]IDAHO.NOD;1 opened for write

TRIEDIT>

ENABLE BITPAD

Selects and enables a Sigmex bitpad as the graphics input device.

FORMAT: ENABLE BITPAD

Command parameters: None.

DESCRIPTION:

ENABLE BITPAD allows the user to select and enable a Sigmex bitpad as the graphics input device.

TRIEDIT supports many different graphics displays and graphics input devices; trackerballs, thumbwheels, digitising tables etc. However, there are severe compatibility restrictions between different graphics displays and graphics input devices. When DTMCREATE is installed by Laser-Scan personnel at your site, the available combinations of graphics display and graphics input devices will be set up for you in the lookup files referenced by TRIEDIT on program initialisation. Do not alter these default combinations without first consulting Laser-Scan.

TRIEDIT can only accept input from one graphics input device at a time. TRIEDIT will not allow you to ENABLE a new graphics input device until any currently active is disabled.

Messages: None.

Examples:

TRIEDIT> **ENABLE BITPAD<CR>**
TRIEDIT>

ENABLE DCUPDATE

ENABLE automatic contour drawing on the graphics screen after every CLEAR command.

FORMAT: ENABLE DCUPDATE

Command parameters: None.

DESCRIPTION:

ENABLE DCUPDATE allows the user to force contours to be automatically drawn on the graphics screen after every CLEAR command.

On the graphics screen contours are displayed in two colours. This enables the user to differentiate between index contours (of height specified by the INDEX_INTERVAL command) and intermediate contours (of a height specified by the INTERVAL command).

For further details of the contour drawing process, see the DRAW CONTOURS command.

Messages: None.

Examples:

TRIEDIT> **ENABLE DCUPDATE<CR>**
TRIEDIT>

ENABLE DIAGNOSTICS

Allows the user to enable diagnostic printout.

FORMAT: ENABLE DIAGNOSTICS

Command parameters: None.

DESCRIPTION:

ENABLE DIAGNOSTICS allows the user to enable diagnostic printout.

Diagnostic printout is useful if the user wishes to receive indications of the effect of selections on data input, using the FILEIN commnad.

It should be noted that if DIAGNOSTICS are enabled, TRIEDIT can produce voluminous printout during the input phase in conjunction with selections from IFF files.

Messages: None.

Examples:

```
TRIEDIT> ENABLE DIAGNOSTICS<CR>
TRIEDIT> SHOW ENABLE<CR>
DIAGNOSTICS ..................................................... On
DIVIDEBY ........................................................ Off
INTEGER_HEIGHT .................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ......................................................... Off
MULTIPLY ........................................................ Off
PME ............................................................. Off
TOFEET .......................................................... Off
TOMETRES ........................................................ Off
TRIEDIT> DISABLE DIAGNOSTICS<CR>
TRIEDIT> SHOW ENABLE<CR>
DIAGNOSTICS ..................................................... Off
DIVIDEBY ........................................................ Off
INTEGER_HEIGHT .................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ......................................................... Off
MULTIPLY ........................................................ Off
PME ............................................................. Off
TOFEET .......................................................... Off
TOMETRES ........................................................ Off
TRIEDIT>
```

ENABLE DIVIDEBY

Allows the user to enable division of input file heights by a specified floating point constant.

FORMAT: ENABLE DIVIDEBY denominator

Command parameters:

denominator

The value by which all heights from the FILEIN input file are to be divided.

DESCRIPTION:

ENABLE DIVIDEBY enables the user to divide all incoming heights by a specified (floating point) constant. For example, the command ENABLE DIVIDEBY 2.0 will cause all incoming heights to be divided by 2.0. An ENABLE DIVIDEBY 3.2808455 command has the same effect as an ENABLE TOMETRES command.

Messages:

The following messages are specific to the ENABLE DIVIDEBY command:

*** WARNING *** You are already planning to multiply by 'constant'
ENABLE DIVIDEBY command now overrides ENABLE MULTIPLYBY command

*** WARNING *** You must specify a value for DIVIDEBY

Examples:

TRIEDIT> **ENABLE DIVIDEBY 5.0<CR>**
TRIEDIT>

ENABLE DLUPDATE

Enable automatic label drawing on the graphics screen after every CLEAR command.

FORMAT: ENABLE DLUPDATE

Command parameters: None.

DESCRIPTION:

ENABLE DLUPDATE allows the user to force any selected labels to be automatically drawn on the graphics screen after every CLEAR command.

Messages: None.

Examples:

TRIEDIT> **ENABLE DLUPDATE<CR>**
TRIEDIT>

ENABLE DNUPDATE

Enable automatic node location drawing on the graphics screen after every
CLEAR command.

FORMAT: ENABLE DNUPDATE

Command parameters: None.

DESCRIPTION:

ENABLE DNUPDATE allows the user to force node locations to be automatically drawn on the graphics screen after every CLEAR command.

Messages: None.

Examples:

TRIEDIT> **ENABLE DNUPDATE<CR>**
TRIEDIT>

ENABLE DTUPDATE

Enable automatic triangle drawing on the graphics screen after every CLEAR command.

FORMAT: ENABLE DTUPDATE

Command parameters: None.

DESCRIPTION:

ENABLE DTUPDATE allows the user to force triangles to be automatically drawn on the graphics screen after every CLEAR command.

Messages: None.

Examples:

TRIEDIT> **ENABLE DTUPDATE<CR>**
TRIEDIT>

ENABLE INTEGER_HEIGHT

Take feature heights from type 2 (integer) AC (Ancillary Code) entries in
an IFF input file.

FORMAT: ENABLE INTEGER_HEIGHT

Command parameters: None.

DESCRIPTION:

In IFF files, height values are transmitted via AC (Ancillary Code) or ZS (3D string) entries. By default contour and spot height Z values are read from type 3 ACs as floating point values. By use of the ENABLE INTEGER_HEIGHT command, integer heights may be read from type 2 ACs.

ENABLE INTEGER_HEIGHT causes TRIEDIT to take heights from type 2 (integer) AC (Ancillary Code) entries in an IFF input file.

Messages: None.

Examples:

TRIEDIT> **ENABLE INTEGER_HEIGHT<CR>**
TRIEDIT>

ENABLE INVERSE

Enable height inversion.

FORMAT: ENABLE INVERSE

Command parameters: None.

DESCRIPTION:

Incoming IFF heights may be inverted if the ENABLE INVERSE command is specified prior to reading an IFF file with a FILEIN command. This enables modelling of hydrographic data where sea depths are often stored as positive heights, drying zone heights as negative and land heights as positive!

By default heights are not inverted.

If the DATUM command has been used to specify a change of height datum the heights are inverted before the datum value is added.

Messages: None.

Examples:

TRIEDIT> **DATUM 8.2<CR>**
TRIEDIT> **ENABLE INVERSE <CR>**
TRIEDIT>

ENABLE JOYSTICK

Selects and enables a Tektronix
4100 series joystick as the graphics input device.

FORMAT: ENABLE JOYSTICK

Command parameters: None.

DESCRIPTION:

ENABLE JOYSTICK allows the user to select and enable a Tektronix
4100 series joystick as the graphics input device.

TRIEDIT supports many different graphics displays and graphics input devices; trackerballs, thumbwheels, digitising tables etc. However, there are severe compatibility restrictions between different graphics displays and graphics input devices. When DTMCREATE is installed by Laser-Scan personnel at your site, the available combinations of graphics display and graphics input devices will be set up for you in the lookup files referenced by TRIEDIT on program initialisation. Do not alter these default combinations without first consulting Laser-Scan.

TRIEDIT can only accept input from one graphics input device at a time. TRIEDIT will not allow you to ENABLE a new graphics input device until any currently active is disabled.

Messages: None.

Examples:

TRIEDIT> **ENABLE JOYSTICK<CR>**
TRIEDIT>

ENABLE MULTIPLYBY

ENABLE MULTIPLYBY allows the user to enable multiplication of input file heights by a specified floating point constant.

FORMAT: ENABLE MULTIPLYBY multiplicand

Command parameters:

multiplicand

The value by which all heights from the FILEIN input file are to be multiplied.

DESCRIPTION:

The ENABLE MULTIPLYBY enables the user to multiply all incoming heights by a specified (floating point) constant. For example, the command ENABLE MULTIPLYBY 2.0 will cause all incoming heights to be multiplied by 2.0. An ENABLE MULTIPLYBY 3.2808455 command has the same effect as an ENABLE TOFEET command.

Messages:

The following messages are specific to the ENABLE MULTIPLYBY command:

*** WARNING *** You are already planning to divide by 'constant'
ENABLE MULTIPLY command now overrides ENABLE DIVIDEBY command

*** WARNING *** You must specify a value for MULTIPLYBY

Examples:

TRIEDIT> **ENABLE MULTIPLYBY 10.0<CR>**
TRIEDIT>

ENABLE MOUSE

Selects and enables a VAXstation MOUSE as the graphics input device.

FORMAT: ENABLE MOUSE

Command parameters: None.

DESCRIPTION:

ENABLE MOUSE allows the user to select and enable a VAXstation mouse as the graphics input device.

TRIEDIT supports many different graphics displays and graphics input devices; trackerballs, thumbwheels, digitising tables etc. However, there are severe compatibility restrictions between different graphics displays and graphics input devices. When DTMCREATE is installed by Laser-Scan personnel at your site, the available combinations of graphics display and graphics input devices will be set up for you in the lookup files referenced by TRIEDIT on program initialisation. Do not alter these default combinations without first consulting Laser-Scan.

TRIEDIT can only accept input from one graphics input device at a time. TRIEDIT will not allow you to ENABLE a new graphics input device until any currently active is disabled.

Messages: None.

Examples:

TRIEDIT> **ENABLE MOUSE<CR>**
TRIEDIT>

ENABLE MUART_TABLE

Selects and enables a Laser-Scan supported MUART controlled digitising table as the graphics input device.

FORMAT: ENABLE MUART_TABLE

Command parameters: None.

DESCRIPTION:

ENABLE MUART_TABLE allows the user to select and enable a Laser-Scan supported MUART controlled digitising table as the graphics input device.

TRIEDIT supports many different graphics displays and graphics input devices; trackerballs, thumbwheels, digitising tables etc. However, there are severe compatibility restrictions between different graphics displays and graphics input devices. When DTMCREATE is installed by Laser-Scan personnel at your site, the available combinations of graphics display and graphics input devices will be set up for you in the lookup files referenced by TRIEDIT on program initialisation. Do not alter these default combinations without first consulting Laser-Scan.

TRIEDIT can only accept input from one graphics input device at a time. TRIEDIT will not allow you to ENABLE a new graphics input device until any currently active is disabled.

Messages: None.

Examples:

TRIEDIT> **ENABLE MUART_TABLE<CR>**
TRIEDIT>

ENABLE PME

ENABLE PME enables the PME performance monitor.

FORMAT: ENABLE PME

Command parameters: None.

DESCRIPTION:

The ENABLE PME and DISABLE PME commands are reserved for Laser-Scan use. PME is a code optimisation tool and should be invoked by LSL software personnel only.

ENABLE PME causes the PME_INIT routine to be invoked.

Message:

The following warning message is specific to the ENABLE PME command:

*** WARNING *** You are already using PME!

Examples:

TRIEDIT> **ENABLE PME<CR>**
TRIEDIT> **FILEIN ATILLA<CR>**

5626 nodes added, total now 27289

TRIEDIT> **DISABLE PME<CR>**
TRIEDIT>

ENABLE TABLE

Selects and enables a Laser-Scan supported TABLE MONITOR controlled digitising table as the graphics input device.

FORMAT: ENABLE TABLE

Command parameters: None.

DESCRIPTION:

ENABLE TABLE allows the user to select and enable a Laser-Scan supported TABLE MONITOR controlled digitising table as the graphics input device.

TRIEDIT supports many different graphics displays and graphics input devices; trackerballs, thumbwheels, digitising tables etc. However, there are severe compatibility restrictions between different graphics displays and graphics input devices. When DTMCREATE is installed by Laser-Scan personnel at your site, the available combinations of graphics display and graphics input devices will be set up for you in the lookup files referenced by TRIEDIT on program initialisation. Do not alter these default combinations without first consulting Laser-Scan.

TRIEDIT can only accept input from one graphics input device at a time. TRIEDIT will not allow you to ENABLE a new graphics input device until any currently active is disabled.

Messages: None.

Examples:

TRIEDIT> **ENABLE TABLE<CR>**
TRIEDIT>

ENABLE TOFEET

The ENABLE TOFEET command enables the conversion of input file heights from metres to feet.

FORMAT: ENABLE TOFEET

Command parameters: None.

DESCRIPTION:

It is possible that different input files may have heights recorded in different measurement systems. The model must be relative to one system only. Two height conversion options are available: ENABLE TOMETRES and ENABLE TOFEET.

The ENABLE TOFEET command enables the conversion of input file heights from feet to metres. It has the same effect as an explicit ENABLE MULTIPLYBY 3.2808455 command. It is possible to read in one file with heights in feet with one FILEIN command and then use the ENABLE TOFEET command and read in another file with heights in metres converting to feet during read-in.

Messages: None.

Examples:

TRIEDIT> **ENABLE TOFEET<CR>**
TRIEDIT>

ENABLE TOMETRES

The ENABLE TOMETRES command enables the conversion of input file heights from feet to metres.

FORMAT: ENABLE TOMETRES

Command parameters: None.

DESCRIPTION:

It is possible that different input files may have heights recorded in different measurement systems. The model must be relative to one system only. Two height conversion options are available: ENABLE TOFEET and ENABLE TOMETRES.

The ENABLE TOMETRES command results in the conversion of heights held in the IFF file in feet to metres. It has the same effect as an explicit ENABLE DIVIDEBY 3.2808455 command. It is possible to read in one file with heights in metres with one FILEIN command and then use the ENABLE TOMETRES command and read in another file with heights in feet converting to metres during read-in.

Messages: None.

Examples:

TRIEDIT> **ENABLE TOMETRES<CR>**
TRIEDIT>

ENABLE THUMBWHEELS

Selects and enables Tektronix
4000 series thumbwheels as the graphics input device.

FORMAT: ENABLE THUMBWHEELS

Command parameters: None.

DESCRIPTION:

ENABLE THUMBWHEELS allows the user to select and enable Tektronix 4000 series thumbwheels as the graphics input device.

TRIEDIT supports many different graphics displays and graphics input devices; trackerballs, thumbwheels, digitising tables etc. However, there are severe compatibility restrictions between different graphics displays and graphics input devices. When DTMCREATE is installed by Laser-Scan personnel at your site, the available combinations of graphics display and graphics input devices will be set up for you in the lookup files referenced by TRIEDIT on program initialisation. Do not alter these default combinations without first consulting Laser-Scan.

TRIEDIT can only accept input from one graphics input device at a time. TRIEDIT will not allow you to ENABLE a new graphics input device until any currently active is disabled.

Messages: None.

Examples:

TRIEDIT> **ENABLE THUMBWHEELS<CR>**
TRIEDIT>

ENABLE TRACKERBALL

Selects and enables a Sigmex 7000 series trackerball as the graphics input
device.

FORMAT: ENABLE TRACKERBALL

Command parameters: None.

DESCRIPTION:

ENABLE TRACKERBALL allows the user to select and enable a Sigmex 7000 series trackerball as the graphics input device.

TRIEDIT supports many different graphics displays and graphics input devices; trackerballs, thumbwheels, digitising tables etc. However, there are severe compatibility restrictions between different graphics displays and graphics input devices. When DTMCREATE is installed by Laser-Scan personnel at your site, the available combinations of graphics display and graphics input devices will be set up for you in the lookup files referenced by TRIEDIT on program initialisation. Do not alter these default combinations without first consulting Laser-Scan.

TRIEDIT can only accept input from one graphics input device at a time. TRIEDIT will not allow you to ENABLE a new graphics input device until any currently active is disabled.

Messages: None.

Examples:

TRIEDIT> **ENABLE TRACKERBALL<CR>**
TRIEDIT>

EXIT

Specifies that the edit session is to be terminated and new triangulation .DTA and .NOD files are to be created.

FORMAT: EXIT [file-spec]

COMMAND PARAMETERS:

[file-spec]

The generic specification of the file to be opened for data output.

If a partial file-spec is supplied, missing parts will be taken from the defaults SYS$DISK:[].DTA;0 and SYS$DISK:[].NOD;0

DESCRIPTION:

The EXIT command specifies that the edit session is to be terminated and new triangulation .DTA and .NOD files are to be created.

Optionally, a generic file-spec may be supplied as an argument to the EXIT command and this will be used to form the output file specifications.

If no file-spec command argument is supplied then output will be to files having the same specifications as those used for input at the start of the edit session but with the file version numbers incremented by one.

Messages:

The following messages are specific to the EXIT command:

*** ERROR *** reading file-spec

Examples:

```
TRIEDIT> EXIT DUA3:[DEMONSTRATION]IDAHO<CR>
+++ File DUA3:[DEMONSTRATION]IDAHO.DTA;1 opened for write
+++ File DUA3:[DEMONSTRATION]IDAHO.NOD;1 opened for write

 ELAPSED:  00:05:25.84 CPU:  0:00:05.71 BUFIO:  281 DIRIO:  46 FAULTS:  263
$
```

FACET

Generate an IFF file to contain the triangles as heighted 3 point features.

FORMAT: FACET iff-file-spec

COMMAND PARAMETERS:

iff-file-spec

The specification of an IFF file in which triangle features are to be created.

If a partial file-spec is supplied, missing parts will be taken from the default 'LSL$IF:IFF.IFF;0'.

DESCRIPTION:

The FACET command enables the user to output the triangles defined by the inter-node relationships stored within the editor, as individual IFF features. Each IFF feature defines the vertices of a triangle. Each vertex has a height (Z) and flag (ZB). The flag indicates whether the vertex is a normal or breakline node within the triangulation, and whether the vertex is of special geomorphological significance.

The IFF triangle features may be used as input to triangle based visualisation software, or may be plotted using FPP or LITES2.

Triangle features are generated for that portion of the triangulation that lies within the current screen window. If triangle features are to be generated for the whole triangulation use the WINDOW command to set the screen window to cover the whole triangulation area.

The IFF file differs from that optionally produced by the DRAW TRIANGLES command as it stores each inter-node link twice; once for each of the two triangles of which it forms a part. The IFF file optionally created by the DRAW TRIANGLES command contains each inter-node link only once. A single feature within the DRAW TRIANGLES IFF file contains links from many triangles. The IFF features within the DRAW TRIANGLES IFF file do not form triangles but are arranged for compact storage and efficient plotting; i.e. purely for graphical reproduction purposes.

One result of this difference is that the FACET command can only create an IFF file containing a maximum of 65535 triangle features. If the triangulation contains more than this number the WINDOW command should be used in conjunction with multiple FACET commands to subdivide the triangulation into several IFF output IFF files. If the user attempts to output more than 65535 triangles to a single IFF file, a warning message is issued and the IFF file is closed in an incomplete state.

The algorithm used to output the triangles to IFF file ensures that when a triangulation is subdivided for output to multiple FACET IFF files a continuous cover of triangles is maintained if IFF files containing adjacent windows are plotted side by side. Triangles which represent links to nodes that lie outside of the user defined window are included in the IFF output file. The algorithm ensures that no duplicate triangles are output, either within a single IFF output file or between IFF files which share a common window boundary.

If triangles do have links which lie outside of the user specified WINDOW limits, the IFF output file RA (RAnge entry) reflects the true data range, not merely the WINDOW limits. The IFF file CP (Control Point) entry does, however, reflect the user specified WINDOW limits.

To ensure that multiple IFF output files do share **exactly** common WINDOW boundaries it is recommemded that the WINDOW command is specified with the optional command arguments. Do not use the WINDOW command in conjunction with the GIN cursor. That is:

```
TRIEDIT> ! Left hand window<CR>
TRIEDIT> WINDOW 100.0 130.0 270.0 340.0<CR>
TRIEDIT> FACET TRIPLOT1<CR>

IFF file LSL$IF:TRIPLOT1.IFF opened for write

TRIEDIT> ! Right hand window<CR>
TRIEDIT> WINDOW 270.0 130.0 470.0 340.0<CR>
TRIEDIT> FACET TRIPLOT2<CR>

IFF file LSL$IF:TRIPLOT2.IFF opened for write

TRIEDIT>
```

Another important difference between the IFF file created by the DRAW TRIANGLES and FACET commands is their respective use of IFF CB (Coordinate Block) entries. The features in IFF files created by the FACET command wil always contain CB entries regardless of the IFF output revision level active at the user's site. Features in IFF files created by the DRAW TRIANGLES command will only contain CBs if the IFF output revision level is set to 1 (i.e. CBs are to be produced and more importantly can be handled by all IFF utility programs at that site).

The reason for the FACET command IFF files containing CBs is to enable both a height (Z) and a vertex type flag to be transmitted to any visualisation software reading the IFF file. The vertex type flag indicates whether each vertex is a normal or breakline node within the triangulation, and conveys any associated geomorphological attributes. The vertex type flag is the fourth dimension of each (X,Y,Z,flag) coordinate defining the triangle and is defined by ACD ZB (94) using the Laser-Scan preset ACD definitions. (For further information about ACDs see the FRT User Guide in the MAPPING Reference Manual).

Possible vertex flag (ZB) values are:

Vertex type	Flag value
Breakline ridgeline node	-3
Breakline river node	-2
Breakline unflagged node	-1
Normal unflagged node	1
Normal river node	2
Normal ridgeline node	3

Imaginary nodes within the triangulation have no Z value. Triangles which have imaginary vertices are omitted from FACET command IFF output files.

IFF coordinates are in the user units supplied to TRIANG.

By default the IFF features will be created in layer 1, or the layer number set with the most recent SET LAYER command.

By default the triangle features are given feature code 1, or the feature code set with the most recent SET LINK_FC command.

Messages:

The following messages are specific to the FACET command:

*** ERROR *** The FACET command requires an IFF file-spec argument

*** ERROR *** opening IFF file

*** WARNING *** error reading IFF file-spec

Examples:

TRIEDIT> **WINDOW 100.0 130.0 270.0 340.0<CR>**
TRIEDIT> **FACET VIS3.IFF<CR>**

IFF file LSL$IF:VIS3.IFF opened for write

TRIEDIT>

FILEIN

Specifies an IFF file that is to be opened and used for data input.

FORMAT: FILEIN IFF-file-spec

COMMAND PARAMETERS:

IFF-file-spec

The specification of the file to be opened for data input.

Any parts of the IFF-file-spec not supplied for the FILEIN command will be taken from the default specification LSL$IF:IFF.IFF;0

DESCRIPTION:

The FILEIN command causes the specified IFF file to be opened and used as an input file to TRIEDIT. The SELECT, DESELECT, ASSIGN and DEASSIGN commands are provided to enable the user to selectively extract and flag IFF features for inclusion in the triangulation.

If a node is inserted which is coincident with an existing node, TRIEDIT will always replace the existing node with the one that you have just inserted. A coincident node is one which lies within the distance defined by the maximum triangulation extent multiplied by 7.0/300000.0.

Messages:

The following messages are specific to the FILEIN command:

*** WARNING *** You must specify a file-spec argument to the FILEIN command

*** ERROR *** Unable to interpret input file-spec

Examples:

TRIEDIT> **FILEIN DUA3:[DEMONSTRATION]IDAHO<CR>**
IFF file DUA3:[DEMONSTRATION]IDAHO.IFF;1 opened for read

1841 nodes added, total now 56900

TRIEDIT>

FRT

Specifies an FRT file which contains feature code group definitions.

FORMAT: FRT file-spec

Command parameters:

file-spec

The specification of an FRT (Feature Representation Table) file required to define feature code groups.

Missing parts from the FRT file-spec argument are taken from the default specification LSL$FRT:FRT.FRT;0.

DESCRIPTION:

The FRT command allows the user to specify an FRT file which contains feature code group definitions. The availability of feature code groups simplifies the specification of complex feature code selections for breaklines etc.

Messages:

The following messages are specific to the FRT command:

*** ERROR *** reading FRT file-spec

*** ERROR *** unable to open specified FRT

Examples:

TRIEDIT> **SELECT FC OUTCROPS,7,COAST<CR>**
*** WARNING *** You must have read an FRT file to be able to use group names
TRIEDIT> **FRT HOVER<CR>**
FRT file LSL$FRT:HOVER.FRT;8 opened for read
TRIEDIT> **SELECT BREAKLINE_FC OUTCROPS,7<CR>**
TRIEDIT>

HEIGHT

Display the height of the specified location.

FORMAT: HEIGHT

Command parameters: None.

DESCRIPTION:

The HEIGHT command enables the user to display the height of a specified location. After issuing the HEIGHT command place the cursor over the selected location (which need not be at a node) and press the rightmost function button. The height will be displayed next to the cursor position.

The characteristics of the label are determined by the current LABEL selections.

The height displayed is calculated using linear interpolation across the facet plane of the triangle within which the cursor lies. Clearly, using this approach, if the cursor lies at a node location the height displayed is the node height. The same technique is used for contour generation with the DRAW CONTOURS command.

It is not possible to use the HEIGHT command to estimate the height of the triangulated surface within a triangle defined by one or more imaginary nodes. Within TRIEDIT imaginary nodes have location but no Z value.

Messages:

The following messages are specific to the HEIGHT command:

*** WARNING *** Tried to calculate height in a perimeter triangle

---

Examples:

TRIEDIT> **HEIGHT<CR>**
TRIEDIT>

HELP

Give help on a subject

FORMAT: HELP subject

Command parameters:

subject

The subject on which help is required

Description:

The HELP command looks the rest of the line up in the DTMCREATE HELP library. This library contains a brief summary of the operation of each command.

The information is looked up in the TRIEDIT section of the DTMCREATE help library, LSL$HELP:DTMCREATE.HLB.

Messages:

Where required, warning messages are output via the VMS LBR$OUTPUT_HELP utility.

Examples:

TRIEDIT> **HELP DESELECT RIVER_FC<CR>**

ENABLE

DESELECT_RIVER

The DESELECT RIVER_FC command complements the SELECT RIVER_FC command. DESELECT RIVER_FC enables the user to remove specified IFF feature codes from the list of those selected for use as rivers within the triangulation.

By default no feature codes are selected for breaklines, rivers etc. and all IFF data that lie within the triangulation bounds are included in the triangulation.

Press <RETURN> to continue**<CR>**

TRIEDIT>

IFF

Generate an IFF file from the triangulated data.

FORMAT: IFF [iff-file-spec]

Command parameters:

iff-file-spec

The specification of the IFF file which is to be created. If no file-spec is supplied then the following default file-spec is constructed: "LSL$IF:'original-input-DTA-filename'.IFF".

If a partial file-spec is supplied as an argument to the IFF command then the missing parts of the file specification are taken from the default "LSL$IF:IFF.IFF;0".

DESCRIPTION:

The IFF command enables data to be extracted from a triangulation and written back to an IFF file.

By default the following feature codes are applied in the IFF file:

Triangulation feature attribute	FC in IFF file
Unflagged string - multi node	1
Unflagged string - single node	2
River string - multiple node	3
River string - single node	4
Ridge string - multiple node	5
Cliff string	6

Data types are differentiated by IFF layer:

Triangulation data type	IFF layer
Normal (non-break/cliff)	1
Breaklines	2
Clifflines	3

It can be seen from the table that, by default, triangulation feature attributes are differentiated by IFF feature code and that feature type is differentiated by IFF layer number.

Using this coding scheme in the output IFF file it may be readily inferred that a feature which has a feature code of 3 and which lies within layer 2 is a river that has been used in the triangulation as a breakline.

Cliffline features are separated from other data by layer number. They are also given a feature code that enables differentiation between cliffs and other data.

The complete suit of SET commands required to override all these defaults are shown in the following typical example TRIEDIT command file:

```
!
! ASCII file to be used as an indirect TRIEDIT command file (invoked using
! the TRIEDIT @file-spec facility)
!
! This file contains a complete set of commands to override the default
! feature code assignments used in IFF files created using the IFF command.
!
! Note use of "!" character as a comment delimiter. All text which lies to the
! right of a "!" character is ignored.
!
SET STRING_FC  1001
SET POINT_FC 1002
SET RIVER_STRING_FC  201
SET RIVER_POINT_FC  202
SET RIDGE_STRING_FC  305
SET RIDGE_POINT_FC  670
SET CLIFF_STRING_FC  671
SET LAYER  34
SET BREAKLINE_LAYER  2
SET CLIFF_LAYER  8
SET FRAME_FC 19
!
! RETURN control to terminal input ...
!
RETURN
```

Note that use of the FACET or IFF commands within TRIEDIT results in the generation of IFF files containing CB (Coordinate Block) entries. CB entries replace the use of ST (STring) and ZS (3D string) entries.

IFF features which represent clifflines will have the X,Y and two attribute fields set for each coordinate in their CBs. The attributes are 80 ("cliff left height") and 81 ("cliff right height"), both of which contain a real (floating point) height value.

Messages:

The following messages are specific to the IFF command:

*** ERROR *** opening IFF file

*** ERROR *** reading IFF file-spec

Examples:

TRIEDIT> **IFF TRIANGULATION_3.IFF<CR>**

IFF file TRIANGULATION_3.IFF opened for write

TRIEDIT>

INDEX_INTERVAL

Specifies the height interval between successive index contours drawn by the DRAW CONTOURS command.

FORMAT: INDEX_INTERVAL interval

Command parameters:

interval

The height interval between successive index contours (floating point or integer).

DESCRIPTION:

The INDEX_INTERVAL argument specifies the height interval between successive index contours. By default an index contour interval of 0.0 is used.

If the index contour interval is zero TRIEDIT will not generate any index contours.

If drawing contours to IFF file index contours may be given a different feature code (specified by the SET INDEX_CONTOUR_FC command) in the IFF output file to enable them to be differentiated from other (or "intermediate") contours.

When drawing contours on the graphics screen, the index contours will be given a different colour to intermediate (ordinary) contours.

Messages:

The following messages are specific to the INDEX_INTERVAL command:

*** ERROR *** Missing interval argument

Only one value please !

Taking first value only ('real')

Examples:

TRIEDIT> **INTERVAL 10.0<CR>**
TRIEDIT> **INDEX_INTERVAL 200.0<CR>**
TRIEDIT>

INSERT

Insert a new node, or nodes, into the triangulation having the attributes defined by the SET HEIGHT, SET FEATURE_FLAG and SET TYPE commands.

FORMAT: INSERT

Command parameters: None.

DESCRIPTION:

The INSERT command enables the user to interactively insert a new node, or nodes, into the triangulation.

The attributes of nodes inserted with the INSERT command are defined by the SET HEIGHT, SET FEATURE_FLAG and SET TYPE commands.

Nodes may be entered singly or as strings of nodes. These strings may have a fixed Z value or may have a continuously varying Z value along their length.

Nodes may be flagged as unflagged, rivers or ridgelines. They may be assigned as either the NORMAL or BREAKLINE data type.

Clifflines may be inserted with the INSERT command.

NOTE

If a node is inserted which is coincident with an existing node, TRIEDIT will always replace the existing node with the one that you have just inserted. A coincident node is one which lies within the distance defined by the maximum triangulation extent multiplied by 7.0/300000.0.

How to use the INSERT command:

- Use SET commands to define the correct insertion defaults for node height (single floating point value or multivalued), node feature flag (either unflagged, river or ridgeline) and data type (either normal or breakline). The current INSERT attribute defaults are displayed in the status area.
- Issue the INSERT command
- Move the cursor the the location of the first (or only) node in the string to be inserted.

o Then:

Inserting a single node

If inserting a single node press the rightmost function button ("end"). If you had specified that the string was multivalued (SET HEIGHT MULTIVALUED) you will be prompted for the height of the node. (It would have been better to have set the height beforehand with the SET HEIGHT 'height' command). Type the height and press carriage return. TRIEDIT will return to the TRIEDIT> prompt. The insertion is complete. The symbol drawn on the screen represents the location of the new node.

Inserting a multi-node string with varying Z values

If inserting a multi-node string with varying Z values press the middle function button ("master node"). As you specified that the string was multivalued (SET HEIGHT MULTIVALUED) you will be prompted for the height of the first node. Type the height and press carriage return. A symbol will be drawn on the screen to represent the location of the new node.

Add additional nodes using the leftmost function button ("node") until you want to put in another master node for which you will supply a Z value. When prompted, type the height and press carriage return. A symbol will be drawn on the screen to represent the location of the new node. You will notice also that other symbols may be drawn between the nodes that you have specified. These represent nodes that TRIEDIT has had to generate to ensure that existing triangle links are cut by your new inserted string. The nodes that you inserted explicitly with the leftmost function button and those automatically generated by TRIEDIT will be given Z values derived using linear interpolation between the "master" nodes that you inserted with the centre function button.

Continue inserting nodes and "master" nodes until you have inserted the penultimate node.

Insert the last node by using the rightmost function button ("end"). When prompted for the final height type in the height and press carriage return. TRIEDIT will return to the TRIEDIT> prompt. The insertion is complete. The symbols drawn on the screen represent the location of the new string.

Inserting a multi-node string with constant Z value

If inserting a multi-node string with constant Z value press the middle function button ("master node"). A symbol will be drawn on the screen to represent the location of the new node.

Add additional nodes using either the leftmost function button or the centre function button. Both are interpreted as simply "node" insertions, there is no "master" node concept in constant Z value string insertion.

A symbol will be drawn on the screen to represent the location of each new node. You will notice also that other symbols may be drawn between the nodes that you have specified. These represent nodes that TRIEDIT has had to generate to

ensure that existing triangle links are cut by your new inserted string. The nodes that you inserted explicitly with the leftmost function button and those automatically generated by TRIEDIT will be given the Z value set with the SET HEIGHT command.

Continue inserting nodes until you have inserted the penultimate node.

Insert the last node by using the rightmost function button ("end").

TRIEDIT will return to the TRIEDIT> prompt. The insertion is complete. The symbols drawn on the screen represent the location of the new string.

If you fail to move the cursor between successive function button presses within the same insertion, TRIEDIT will complain about superimposed nodes with the message "Nodes too close". Move the cursor and continue the insertion.

Messages:

The following messages are specific to the INSERT command:

Nodes too close

*** WARNING *** First node must be a master node - command abandoned

*** WARNING *** Illegal node - operation abandoned

*** WARNING *** Backlog buffer full - enter the height of this node

---

Examples:

TRIEDIT> **SET HEIGHT 134.9<CR>**
TRIEDIT> **SET TYPE BREAKLINE<CR>**
TRIEDIT> **INSERT<CR>**

211 nodes added, total now 52628

TRIEDIT> **SET HEIGHT MULTIVALUED<CR>**
TRIEDIT> **SET TYPE NORMAL<CR>**
TRIEDIT> **SET FEATURE_FLAG RIVER<CR>**
TRIEDIT> **INSERT<CR>**
Height: **60<CR>**
Height: **68<CR>**
Final height: **92<CR>**

51 nodes added, total now 52679

TRIEDIT>

INTERVAL

Specifies the height interval between successive contours drawn by the DRAW CONTOURS command.

FORMAT: INTERVAL interval

Command parameters:

interval

The height interval between successive contours (floating point or integer).

DESCRIPTION:

The INTERVAL argument specifies the height interval between successive contours. By default a contour interval of 10% of the triangulation Z range is used.

If the contour interval is zero TRIEDIT will not generate any contours.

If drawing contours to an IFF file, intermediate (ordinary) contours may be given a different feature code (specified by the SET CONTOUR_FC command) in the IFF output file to enable them to be differentiated from "index" contours.

When drawing contours on the graphics screen, index contours will be given a different colour to intermediate (ordinary) contours.

Messages:

The following messages are specific to the INTERVAL command:

*** ERROR *** Missing interval argument

Only one value please !

Taking first value only ('real')

Examples:

TRIEDIT> **INTERVAL 10.0<CR>**
TRIEDIT> **INDEX_INTERVAL 200.0<CR>**
TRIEDIT>

LABEL BIG

Specifies that labels are to be displayed using large characters for clarity on a zoomed screen.

FORMAT: LABEL BIG

Command parameters: None.

DESCRIPTION:

LABEL BIG enables the user to specify that labels are to be displayed using large characters.

By default TRIEDIT will generate labels using small characters.

LABEL BIG should only be specified for use with windows containing few nodes, otherwise excessive cluttering of the screen with text can result.

After issuing a LABEL BIG command labelling with large characters will persist until a LABEL SMALL command is issued.

The effect of ENABLE BIG and ENABLE SMALL commands applies to labels drawn automatically under ENABLE DLUPDATE control.

Messages: None.

Examples:

TRIEDIT> **LABEL BIG<CR>**
TRIEDIT>

LABEL FLOAT

Specifies that labels are to be displayed as floating point numbers.

FORMAT: LABEL FLOAT

Command parameters: None.

DESCRIPTION:

LABEL FLOAT enables the user to specify that contour labels are to be displayed as floating point numbers. TRIEDIT generates labels to 5 significant figures.

By default TRIEDIT will generate integer labels.

Where all digits to the right of the decimal point are zero, TRIEDIT will always display the label as an integer to reduce cluttering of the graphics display.

Messages: None.

Examples:

TRIEDIT> **LABEL FLOAT<CR>**
TRIEDIT>

LABEL HEIGHT

Specifies that input node heights are to be displayed by the DRAW LABELS command (or automatically after a CLEAR command if the ENABLE DLUPDATE option is active).

FORMAT: LABEL HEIGHT

Command parameters: None.

DESCRIPTION:

LABEL HEIGHT enables the user to specify that node heights are to be displayed when a DRAW LABELS command is issued.

Node heights will be displayed as integer values, unless the ENABLE FLOAT command is issued.

LABEL HEIGHT, LABEL SEQUENCE and LABEL SIGN selections may be cancelled using the LABEL NONE command.

Messages: None.

Examples:

TRIEDIT> **LABEL HEIGHT<CR>**
TRIEDIT>

LABEL INTEGER

Specifies that contour labels are to be displayed as integer numbers.

FORMAT: LABEL INTEGER

Command parameters: None.

DESCRIPTION:

LABEL INTEGER enables the user to specify that labels are to be displayed as integer numbers.

By default TRIEDIT will generate integer labels. Floating point labels can be selected with the LABEL FLOAT command.

Messages: None.

Examples:

TRIEDIT> **LABEL INTEGER<CR>**
TRIEDIT>

LABEL NONE

Specifies that any LABEL HEIGHT, LABEL SEQUENCE or LABEL SIGNS labelling selections are to be cancelled.

FORMAT: LABEL NONE

Command parameters: None.

DESCRIPTION:

Specifies that any LABEL HEIGHT, LABEL SEQUENCE or LABEL SIGNS labelling selections are to be cancelled.

This will result in nothing being displayed by the DRAW LABELS command (nor automatically after a CLEAR command if the ENABLE DLUPDATE option is active).

Messages: None.

Examples:

TRIEDIT> **LABEL NONE<CR>**
TRIEDIT>

LABEL SEQUENCE

Specifies that input node sequence numbers are to be displayed by the DRAW LABELS command (or automatically after a CLEAR command if the ENABLE DLUPDATE option is active).

FORMAT: LABEL SEQUENCE

Command parameters: None.

DESCRIPTION:

DTMCREATE assigns each node within a triangulation a unique node sequence number.

LABEL SEQUENCE enables the user to specify that input node sequence numbers are to be displayed when a DRAW LABELS command is issued.

Node sequence numbers are allocated on the basis of data entry order. Due to the nature of the triangulation constraint process, and of subsequent TRIEDIT edit sessions, original input strings may be fragmented. As a result, node sequence numbers may appear to be randomly scattered!

LABEL HEIGHT, LABEL SEQUENCE and LABEL SIGN selections may be cancelled using the LABEL NONE command.

Messages: None.

Examples:

TRIEDIT> **LABEL SEQUENCE<CR>**
TRIEDIT>

LABEL SIGNS

Specifies that input string signs are to be displayed by the DRAW LABELS command (or automatically after a CLEAR command if the ENABLE DLUPDATE option is active).

FORMAT: LABEL SIGNS

Command parameters: None.

DESCRIPTION:

LABEL SIGNS enables the user to specify that input string sign flags are to be displayed when a DRAW LABELS command is issued.

DTMCREATE preserves the concept of inter node connectivity inherent in strings of input nodes by flagging in memory the nodes of successive input strings with a change of sign. These sign flags '+' and '-' will be drawn as labels next to each node if the LABEL SIGNS option is selected and a DRAW LABELS command issued.

A change of string is signified wherever a change of sign occurs.

Due to the nature of the triangulation process, and of subsequent TRIEDIT edit sessions, original input strings may be fragmented. Many more changes of sign may occur than would be expected if all the original input strings were preserved intact.

LABEL HEIGHT, LABEL SEQUENCE and LABEL SIGNS selections may be cancelled using the LABEL NONE command.

Messages: None.

Examples:

TRIEDIT> **LABEL SIGNS<CR>**
TRIEDIT>

LABEL SMALL

Specifies that labels are to be displayed using small characters to reduce screen cluttering.

FORMAT: LABEL SMALL

Command parameters: None.

DESCRIPTION:

LABEL SMALL enables the user to specify that labels are to be displayed using small characters.

LABEL SMALL is the TRIEDIT startup default.

After issuing an explicit LABEL SMALL command labelling with small characters will persist until a LABEL BIG command is issued.

The effect of ENABLE BIG and ENABLE SMALL commands applies to labels drawn automatically under ENABLE DLUPDATE control.

Messages: None.

Examples:

TRIEDIT> **LABEL SMALL<CR>**
TRIEDIT>

PAUSE

Pauses TRIEDIT execution.

FORMAT: PAUSE

Command parameters: None.

DESCRIPTION:

Pauses TRIEDIT execution and issues a prompt for a carriage return to continue execution. This command is designed for use in software demonstration situations.

Messages: None.

Examples:

TRIEDIT> **PAUSE<CR>**

Press <RETURN> to continue**<CR>**
TRIEDIT>

POSITION

Display the (x,y) position of the specified location.

FORMAT: POSITION

Command parameters: None.

DESCRIPTION:

The position command enables the user to display the (x,y) position of a specified location. After issuing the position command place the cursor over the selected location (which need not be at a node) and press the rightmost function button. The position will be displayed at the top of the graphics screen.

Messages: None.

Examples:

TRIEDIT> **POSITION<CR>**
TRIEDIT>

QUIT

Quit from TRIEDIT saving none of the edits made.

FORMAT: QUIT

Command parameters: None.

Description:

The QUIT command causes TRIEDIT to exit immediately, closing all input files. No edits will be saved.

Messages: None.

Examples:

```
TRIEDIT> QUIT<CR>
 ELAPSED:  00:05:25.84 CPU:  0:00:05.71 BUFIO:  281 DIRIO:  46 FAULTS:  263
$
```

REMOVE

Delete the selected node, leaving the remaining nodes in the string untouched.

FORMAT: REMOVE

Command parameters: None.

DESCRIPTION:

The REMOVE command enables the user to delete a single selected node from the triangulation. After issuing the DELETE command, the user must position the cursor over the node and press the rightmost function button. TRIEDIT will draw a symbol over the node deleted.

The "hole" left in the triangulation by the deletion will be automatically repaired with the most equilateral triangulation possible given the shape of the quadrilateral hole and the distribution of surrounding nodes.

It is not possible to REMOVE imaginary or cliffline nodes.

Messages:

The following warning messages are specific to the REMOVE command:

*** WARNING *** You cannot REMOVE an imaginary node

*** WARNING *** You cannot REMOVE a cliffline node

Examples:

TRIEDIT> **ENABLE DTUPDATE<CR>**
TRIEDIT> **CLEAR<CR>**
TRIEDIT> **REMOVE<CR>**
TRIEDIT> **CLEAR<CR>**
TRIEDIT>

RETURN

Restores command input from an indirect file to SYS$COMMAND.

FORMAT: RETURN

Command parameters: None.

DESCRIPTION:

Restores command input from an indirect file to SYS$COMMAND.

A typical application is to allow the user to use an indirect command file to set up those run time defaults which are constant within a flowline and then return to input from the terminal (or batch stream) for the run specific commands. To do this RETURN must be the last command in the indirect command file.

Messages:

The following messages are specific to the RETURN command:

RETURN command detected - returning to terminal input

RETURN command ignored - command input is already from terminal

---

Examples:

```
TRIEDIT> @FLOW2<CR>
TRIEDIT> ENABLE DIAGNOSTICS
TRIEDIT> FRT FLOW2 FRT file LSL$FRT:FLOW2.FRT;8 opened for read
TRIEDIT> SELECT FC OUTCROPS,7,COAST
TRIEDIT> RETURN
TRIEDIT>
```

SELECT ALL

Resets all feature selections made with the SELECT and DESELECT commands.

FORMAT:######SELECT ALL

Command parameters: None

DESCRIPTION:

This command resets all feature input selections. If features are subsequently selected using the other SELECT commands then all features are first implicitly deselected.

Messages:

The following message is specific to the SELECT command.

*** ERROR *** Specifying command SELECT

Examples:

TRIEDIT>**SELECT ALL <CR>**

At this point all features are selected.

TRIEDIT>**SELECT FC 7:10,56:78**

Here only features with the specified feature codes are selected

TRIEDIT>**SELECT FC 11:20**

At this point the specified feature codes are added to the currently selected features code, i.e. features with FC 7-20 and 56-78 are now selected.

TRIEDIT>

SELECT FC

Selects all features with specified IFF feature codes for inclusion in the triangulation.

FORMAT: SELECT FC feature-code[,...]

Command parameters:

feature-code

An IFF feature code which must lie in the range 0 to 32767. Multiple feature codes may be specified separated by commas or spaces. Ranges of feature codes may be specified by separating the range start and stop values by a colon e.g. SELECT FC 2:6 will result in the selection of feature codes 2,3,4,5 and 6.

If an FRT file has been read into TRIEDIT any valid feature code group names may be used as arguments to the SELECT FC command.

DESCRIPTION:

The SELECT FC command complements the DESELECT FC command.

SELECT FC selects all features with specified IFF feature codes for inclusion in the triangulation.

On program startup all FCs are selected for input. IFF features which are not required for input must be specifically excluded using the appropriate DESELECT FC and DESELECT LAYER commands. For example, the command DESELECT FC 0:7 11:300 302:32767 will leave only features with FCs 8,9,10, and 301 selected for input.

The first SELECT FC command has the effect of deselecting all FCs from input except those explicitly specified as the arguments to the SELECT FC command. Subsequent SELECT FC commands have the effect of adding the specified FCs to the list of FCs selected for input.

All layers and FCs may be reselected for input by specifying the SELECT ALL command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the SELECT FC command:

*** WARNING *** You must have read an FRT file to be able to use group names

*** WARNING *** No groups have been defined in the FRT

*** WARNING *** Illegal feature code 'integer'

*** WARNING *** Bad group name 'group-name'

Examples:

TRIEDIT> **SELECT FC 6:9,WATER,126<CR>**
TRIEDIT>

SELECT LAYER

Selects all features which lie within the specified IFF layers for inclusion in the triangulation.

FORMAT: SELECT LAYER layer[,...]

Command parameters:

layer

An IFF layer number which must lie in the range 0 to 32767. Multiple layers may be specified separated by commas or spaces. Ranges of layers may be specified by separating the range start and stop values by a colon e.g. SELECT LAYER 2:6 will result in the selection of layers 2,3,4,5 and 6.

DESCRIPTION:

The SELECT LAYER command complements the DESELECT LAYER command. SELECT LAYER enables the user to select any IFF features which are contained within the specified layers.

By default TRIEDIT will input all features within an IFF file, regardless of the layer in which they lie.

On program startup all layers are selected for input. IFF features which are not required for input must be specifically excluded using the appropriate DESELECT FC and DESELECT LAYER commands. For example, the command DESELECT LAYER 0:7 11:300 302:32767 will leave only features which lie within layers 8,9,10, and 301 selected for input.

The first SELECT LAYER command has the effect of deselecting all layers from input except those explicitly specified as the arguments to the SELECT LAYER command. Subsequent SELECT LAYER commands have the effect of adding the specified layers to the list of FCs selected for input.

All layers and FCs may be reselected for input by specifying the SELECT ALL command.

Note that selections made with the DESELECT and SELECT commands will override input data assignments (e.g. ASSIGN BREAKLINE_FC) which share the same feature code or layer numbers. Thus even though FC 9 has been assigned breakline status by an ASSIGN BREAKLINE_FC 9 command, all features with FC 9 will be excluded from input if the user has specified a DESELECT_FC 9 command. Use the SHOW SELECTIONS command to examine current input selections and the SHOW ASSIGNMENTS command to examine current assignments.

Messages:

The following warning messages are specific to the SELECT LAYER command:

*** WARNING *** Too many layer arguments in one command

*** WARNING *** Illegal layer number 'integer'

Examples:

TRIEDIT> **SELECT LAYER 21:29,126<CR>**
TRIEDIT>

SET BREAKLINE_LAYER

Set the IFF output file breakline layer number to the specified value. The breakline layer is only used by the IFF command.

FORMAT: SET BREAKLINE_LAYER layer-number

Command parameters:

layer-number

The number of the layer to contain the breakline features. This must lie in the range 0 to 32767.

DESCRIPTION:

SET BREAKLINE LAYER enables the user to specify the number of the IFF layer which is to contain the breakline features generated by the IFF command.

By default, TRIEDIT uses layer 2

Messages:

The following message is specific to the SET BREAKLINE_LAYER command:

*** WARNING *** Layer numbers must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET BREAKLINE_LAYER 1008<CR>**
TRIEDIT>

SET BREAKLINE_LINK_FC

Set the IFF feature code of features representing triangle links between breakline nodes to the specified value. Used by the DRAW TRIANGLES 'iff-file-spec' command.

FORMAT: SET BREAKLINE_LINK_FC feature-code

Command parameters:

feature-code

The feature code to be used for IFF features representing triangle links between breakline nodes. This must lie in the range 0 to 32767.

DESCRIPTION:

The feature code set up by the SET BREAKLINE_LINK_FC command is used by the DRAW TRIANGLES 'iff-file-spec' command.

SET BREAKLINE_LINK_FC enables the user to specify the feature code to be used for IFF features representing triangle links between breakline nodes. By default a feature code of 2 is assumed for triangle link features between breakline nodes.

When choosing a feature code for triangle links between breakline nodes, ensure that the feature code is in the FRT file which is to be used for plotting the links and that the feature code has an appropriate graphical type, i.e. line.

Messages:

The following message is specific to the SET BREAKLINE_LINK_FC command:

*** WARNING *** Feature codes must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET BREAKLINE_LINK_FC 112<CR>**
TRIEDIT>

SET CLIFF_LAYER

Set the IFF output file cliff layer number to the specified value. The cliff layer is only used by the IFF command.

FORMAT: SET CLIFF_LAYER layer-number

Command parameters:

layer-number

The number of the layer to contain cliffline features. This must lie in the range 0 to 32767.

DESCRIPTION:

SET CLIFF_LAYER enables the user to specify the number of the IFF layer which is to contain the cliff features generated by the IFF command.

By default, TRIEDIT uses layer 3.

Note that use of the FACET or IFF commands within TRIEDIT results in the generation of IFF files containing CB (Coordinate Block) entries. CB entries replace the use of ST (STring) and ZS (3D string) entries.

IFF features which represent clifflines will have the X,Y and two attribute fields set for each coordinate in their CBs. The attributes are 80 ("cliff left height") and 81 ("cliff right height"), both of which contain a real (floating point) height value.

Messages:

The following message is specific to the SET CLIFF_LAYER command:

*** WARNING *** Layer numbers must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET CLIFF_LAYER 1008<CR>**
TRIEDIT>

SET CLIFF_STRING_FC

Set the IFF feature code of features representing cliff strings. The cliff feature code is only used by the IFF command.

FORMAT: SET CLIFF_STRING_FC feature-code

Command parameters:

feature-code

The feature code to be used for IFF features representing cliff features. This must lie in the range 0 to 32767.

DESCRIPTION:

The feature code set up by the SET CLIFF_STRING_FC command is used by the IFF 'iff-file-spec' command. It provides a mechanism to enable cliffline features to be differentiated from other features.

When choosing a feature code for closed cliff strings, ensure that the feature code is in the FRT file which is to be used for plotting the IFF file and that the feature code has an appropriate graphical type, i.e. line.

Note that use of the FACET or IFF commands within TRIEDIT results in the generation of IFF files containing CB (Coordinate Block) entries. CB entries replace the use of ST (STring) and ZS (3D string) entries.

IFF features which represent clifflines will have the X,Y and two attribute fields set for each coordinate in their CBs. The attributes are 80 ("cliff left height") and 81 ("cliff right height"), both of which contain a real (floating point) height value.

Messages:

The following message is specific to the SET CLIFF_STRING_FC command:

*** WARNING *** Feature codes must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET CLIFF_STRING_FC 112<CR>**
TRIEDIT>

SET CONTOUR_FC

Set the IFF feature code of intermediate contour features to the specified
value.

FORMAT: SET CONTOUR_FC feature-code

Command parameters:

feature-code

The feature code to be used for intermediate contours. This must lie in the range 0 to 32767.

DESCRIPTION:

SET CONTOUR_FC enables the user to specify the feature code to be used for intermediate (i.e. not index) contours. By default, a feature code of 1 is assumed for intermediate contour features.

The feature code set up by the SET CONTOUR_FC command is used by the DRAW CONTOURS 'iff-file-spec' command.

When choosing a feature code for intermediate contours, ensure that the feature code is in the FRT file which is to be used for plotting the contours and that the feature code has an appropriate graphical type, i.e. line.

Messages:

The following message is specific to the SET CONTOUR_FC command:

*** WARNING *** Feature codes must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET CONTOUR_FC 2<CR>**
TRIEDIT>

SET FEATURE_FLAG

Sets the feature flag for nodes to be inserted with the INSERT command.

FORMAT: SET FEATURE_FLAG feature-flag

Command parameters:

feature-flag

The feature flag to be used for nodes inserted with the INSERT command. Valid feature flags are:

- RIDGELINE - insert ridgeline nodes
- RIVER - insert river nodes
- UNFLAGGED - insert nodes that are neither a river or ridge

On program startup the default feature flag is "UNFLAGGED".

DESCRIPTION:

Sets the feature flag for nodes to be inserted with the next INSERT command. This setting will remain active for all subsequent INSERT commands until an alternative SET FEATURE_FLAG command is issued.

The current feature flag setting is displayed in the status area.

Messages:

The following message is specific to the SET FEATURE_FLAG command:

*** WARNING *** Unexpected end of SET FEATURE_FLAG command
Specify either UNFLAGGED, RIVER or RIDGELINE

Examples:

TRIEDIT> **SET FEATURE_FLAG RIVER<CR>**
TRIEDIT>

SET FRAME_FC

Set the IFF feature code of bounding frame features to the specified value. Used by the DRAW TRIANGLES 'iff-file-spec' command.

FORMAT: SET FRAME_FC feature-code

Command parameters:

feature-code

The feature code to be used for bounding frame features. This must lie in the range 0 to 32767.

DESCRIPTION:

SET FRAME_FC enables the user to specify the feature code to be used for bounding frame features. By default, a feature code of 0 is assumed for bounding frame features.

The feature code set up by the SET FRAME_FC command is used by the DRAW TRIANGLES 'iff-file-spec' command.

When choosing a feature code for bounding frame features, ensure that the feature code is in the FRT file which is to be used for plotting the frame and that the feature code has an appropriate graphical type, i.e. line.

Messages:

The following message is specific to the SET FRAME_FC command:

*** WARNING *** Feature codes must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET FRAME_FC 2<CR>**
TRIEDIT>

SET HEIGHT_FC

Sets the height for nodes to be inserted with the INSERT command.

FORMAT: SET HEIGHT height

Command parameters:

height

The height to be used for nodes inserted with the INSERT command. This can be either an integer or floating point value.

If the INSERT command is to be used to insert strings which have varying Z values then specify the keyword MULTIVALUED as the SET HEIGHT argument instead of a height value.

DESCRIPTION:

Sets the height for nodes to be inserted with the next INSERT command. This setting will remain active for all subsequent INSERT commands until an alternative SET HEIGHT command is issued.

The current height setting is displayed in the status area.

On program startup the height is set to 0.0.

If strings with varying heights are to be inserted the keyword MULTIVALUED should be specified as the argument to SET HEIGHT instead of a height value. The user will be prompted for the node heights during the INSERT command sequence. For further details see the INSERT command.

Messages:

The following message is specific to the SET HEIGHT command:

*** ERROR *** Unexpected end of SET HEIGHT command
Either supply a floating point value or the keyword MULTIVALUED

Examples:

TRIEDIT> **SET HEIGHT 134.9<CR>**
TRIEDIT> **INSERT<CR>**

52 nodes added, total now 53266

TRIEDIT> **SET HEIGHT MULTIVALUED<CR>**
TRIEDIT>

SET IMAGINARY_LINK_FC

Set the IFF feature code of features representing triangle links between imaginary nodes to the specified value. Used by the DRAW TRIANGLES 'iff-file-spec' command.

FORMAT: SET IMAGINARY_LINK_FC feature-code

Command parameters:

feature-code

The feature code to be used for IFF features representing triangle links between imaginary nodes. This must lie in the range 0 to 32767.

DESCRIPTION:

The feature code set up by the SET IMAGINARY_LINK_FC command is used by the DRAW TRIANGLES 'iff-file-spec' command.

SET IMAGINARY_LINK_FC enables the user to specify the feature code to be used for IFF features representing triangle links between imaginary nodes. By default a feature code of 3 is assumed for imaginary triangle link features.

When choosing a feature code for triangle links between imaginary nodes, ensure that the feature code is in the FRT file which is to be used for plotting the links and that the feature code has an appropriate graphical type, i.e. line.

Messages:

The following message is specific to the SET IMAGINARY_LINK_FC command:

*** WARNING *** Feature codes must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET IMAGINARY_LINK_FC 112<CR>**
TRIEDIT>

SET INDEX_CONTOUR_FC

Set the IFF feature code of index contour features to the specified value. Used by the DRAW CONTOURS 'iff-file-spec' command.

FORMAT: SET INDEX_CONTOUR_FC feature-code

Command parameters:

feature-code

The feature code to be used for index contours. This must lie in the range 0 to 32767.

DESCRIPTION:

SET INDEX_CONTOUR_FC enables the user to specify the feature code to be used for index contours. By default, a feature code of 2 is assumed for index contour features.

The feature code set up by the SET INDEX_CONTOUR_FC command is used by the DRAW CONTOURS 'iff-file-spec' command.

When choosing a feature code for index contours, ensure that the feature code is in the FRT file which is to be used for plotting the contours and that the feature code has an appropriate graphical type, i.e. line.

Messages:

The following message is specific to the SET INDEX_CONTOUR_FC command:

*** WARNING *** Feature codes must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET INDEX_CONTOUR_FC 2<CR>**
TRIEDIT>

SET LAYER

Set the IFF output file layer number to the specified value.

FORMAT: SET LAYER layer-number

Command parameters:

layer-number

The number of the layer to contain the contour, triangle link or normal node features. This must lie in the range 0 to 32767.

DESCRIPTION:

SET LAYER enables the user to specify the number of the IFF layer which is to contain the contour features generated by the DRAW CONTOURS 'file-spec' command, triangles generated by the DRAW TRIANGLES 'file-spec' command and normal nodes output by the IFF command.

By default, TRIEDIT uses layer 1.

Messages:

The following message is specific to the SET LAYER command:

*** WARNING *** Layer numbers must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET LAYER 1008<CR>**
TRIEDIT>

SET LINK_FC

Set the IFF feature code of features representing triangle links between normal nodes to the specified value. Used by the DRAW TRIANGLES 'iff-file-spec' command.

FORMAT: SET LINK_FC feature-code

Command parameters:

feature-code

The feature code to be used for IFF features representing triangle links between normal nodes. This must lie in the range 0 to 32767.

DESCRIPTION:

The feature code set up by the SET LINK_FC command is used by the DRAW TRIANGLES 'iff-file-spec' command.

SET LINK_FC enables the user to specify the feature code to be used for IFF features representing triangle links between ordinary nodes. By default a feature code of 1 is assumed for ordinary triangle link features.

When choosing a feature code for triangle links between ordinary nodes, ensure that the feature code is in the FRT file which is to be used for plotting the links and that the feature code has an appropriate graphical type, i.e. line.

Messages:

The following message is specific to the SET LINK_FC command:

*** WARNING *** Feature codes must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET LINK_FC 112<CR>**
TRIEDIT>

SET POINT_FC

Set the IFF feature code of features representing isolated unflagged nodes. Used by the IFF 'iff-file-spec' command.

FORMAT: SET POINT_FC feature-code

Command parameters:

feature-code

The feature code to be used for IFF features representing isolated unflagged node features. This must lie in the range 0 to 32767.

DESCRIPTION:

The feature code set up by the SET POINT_FC command is used by the IFF 'iff-file-spec' command. It provides a mechanism to enable single point IFF features representing isolated unflagged nodes to be differentiated from similar features representing isolated river and ridgeline nodes. A different IFF feature code has to be used for single point IFF features to that used for multi-point "string" features to ensure that the feature is plotted correctly. It is normal practice to plot single point IFF features using a symbol (e.g. a dot, cross or a box).

By allowing the user to distinguish between single point features, it is possible to read the IFF file created by the TRIEDIT IFF command into the triangulation module TRIANG for retriangulation. TRIANG has to be told what the isolated nodes represent.

All single point IFF features representing non-breakline nodes (including river and ridgeline nodes) are placed in the same IFF layer (set with the SET LAYER command).

All single point IFF features representing breakline nodes (including river and ridgeline nodes) are placed in a different IFF layer (set with the SET BREAKLINE_LAYER command).

Clifflines can never occur as single nodes.

When choosing a feature code for single point IFF features representing isolated unflagged nodes, ensure that the feature code is in the FRT file which is to be used for plotting the IFF file and that the feature code has an appropriate graphical type, i.e. unoriented symbol.

Note that use of the FACET or IFF commands within TRIEDIT results in the generation of IFF files containing CB (Coordinate Block) entries. CB entries replace the use of ST (STring) and ZS (3D string) entries.

Messages:

The following message is specific to the SET POINT_FC command:

*** WARNING *** Feature codes must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET POINT_FC 152<CR>**
TRIEDIT>

SET RIDGE_POINT_FC

Set the IFF feature code of features representing isolated ridgeline nodes. Used by the IFF 'iff-file-spec' command.

FORMAT: SET RIDGE_POINT_FC feature-code

Command parameters:

feature-code

The feature code to be used for IFF features representing isolated ridgeline node features. This must lie in the range 0 to 32767.

DESCRIPTION:

The feature code set up by the SET RIDGE_POINT_FC command is used by the IFF 'iff-file-spec' command. It provides a mechanism to enable single point IFF features representing isolated ridgeline nodes to be differentiated from similar features representing isolated unflagged and ridgeline nodes. A different feature code has to be used for single point IFF features to that used for multi-point "string" features to ensure that the feature is plotted correctly. It is normal practice to plot single point IFF features using a symbol (e.g. a dot, cross or a box).

By allowing the user to distinguish between single point features, it is possible to read the IFF file created by the TRIEDIT IFF command into the triangulation module TRIANG for retriangulation. TRIANG has to be told what the isolated nodes represent.

All single point IFF features representing non-breakline nodes (including river and ridgeline nodes) are placed in the same IFF layer (set with the SET LAYER command).

All single point IFF features representing breakline nodes (including river and ridgeline nodes) are placed in a different IFF layer (set with the SET BREAKLINE_LAYER command).

Clifflines can never occur as single nodes.

When choosing a feature code for single point IFF features representing isolated ridgeline nodes, ensure that the feature code is in the FRT file which is to be used for plotting the IFF file and that the feature code has an appropriate graphical type, i.e. unoriented symbol.

Note that use of the FACET or IFF commands within TRIEDIT results in the generation of IFF files containing CB (Coordinate Block) entries. CB entries replace the use of ST (STring) and ZS (3D string) entries.

Messages:

The following message is specific to the SET RIDGE_POINT_FC command:

*** WARNING *** Feature codes must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET RIDGE_POINT_FC 152<CR>**
TRIEDIT>

SET RIVER_POINT_FC

Set the IFF feature code of features representing isolated river nodes. Used by the IFF 'iff-file-spec' command.

FORMAT: SET RIVER_POINT_FC feature-code

Command parameters:

feature-code

The feature code to be used for IFF features representing isolated river node features. This must lie in the range 0 to 32767.

DESCRIPTION:

The feature code set up by the SET RIVER_POINT_FC command is used by the IFF 'iff-file-spec' command. It provides a mechanism to enable single point IFF features representing isolated river nodes to be differentiated from similar features representing isolated unflagged and ridgeline nodes. A different feature code has to be used for single point IFF features to that used for multi-point "string" features to ensure that the feature is plotted correctly. It is normal practice to plot single point IFF features using a symbol (e.g. a dot, cross or a box).

By allowing the user to distinguish between single point features, it is possible to read the IFF file created by the TRIEDIT IFF command into the triangulation module TRIANG for retriangulation. TRIANG has to be told what the isolated nodes represent.

All single point IFF features representing non-breakline nodes (including river and ridgeline nodes) are placed in the same IFF layer (set with the SET LAYER command).

All single point IFF features representing breakline nodes (including river and ridgeline nodes) are placed in a different IFF layer (set with the SET BREAKLINE_LAYER command).

Clifflines can never occur as single nodes.

When choosing a feature code for single point IFF features representing isolated river nodes, ensure that the feature code is in the FRT file which is to be used for plotting the IFF file and that the feature code has an appropriate graphical type, i.e. unoriented symbol.

Note that use of the FACET or IFF commands within TRIEDIT results in the generation of IFF files containing CB (Coordinate Block) entries. CB entries replace the use of ST (STring) and ZS (3D string) entries.

Messages:

The following message is specific to the SET RIVER_POINT_FC command:

*** WARNING *** Feature codes must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET RIVER_POINT_FC 152<CR>**
TRIEDIT>

SET RIDGE_STRING_FC

Set the IFF feature code of features representing ridgeline node strings. Used by the IFF 'iff-file-spec' command.

FORMAT: SET RIDGE_STRING_FC feature-code

Command parameters:

feature-code

The feature code to be used for IFF features representing ridgeline node string features. This must lie in the range 0 to 32767.

DESCRIPTION:

The feature code set up by the SET RIDGE_STRING_FC command is used by the IFF 'iff-file-spec' command. It provides a mechanism to enable IFF features representing ridgeline node strings to be differentiated from similar features representing unflagged and river node strings. A different feature code has to be used for multi-point IFF features to that used for single point features to ensure that the feature is plotted correctly. It is normal practice to plot single point IFF features using a symbol (e.g. a dot, cross or a box). Multi-point features are plotted using lines which join the nodes together.

By allowing the user to distinguish between multi-point string features, it is possible to read the IFF file created by the TRIEDIT IFF command into the triangulation module TRIANG for retriangulation. TRIANG has to be told what the strings of nodes represent.

All IFF features representing non-breakline strings (including river and ridgeline nodes) are placed in the same IFF layer (set with the SET LAYER command).

All IFF features representing breakline strings (including river and ridgeline nodes) are placed in a different IFF layer (set with the SET BREAKLINE_LAYER command).

Although all IFF features representing cliffline strings are placed in a third IFF layer (set with the SET CLIFF_LAYER command), it is not possible to have a cliffline string flagged as a river or ridgeline!

When choosing a feature code for multi-point IFF features representing strings of ridgeline nodes, ensure that the feature code is in the FRT file which is to be used for plotting the IFF file and that the feature code has an appropriate graphical type, i.e. line.

Note that use of the FACET or IFF commands within TRIEDIT results in the generation of IFF files containing CB (Coordinate Block) entries. CB entries replace the use of ST (STring) and ZS (3D string) entries.

Messages:

The following message is specific to the SET RIDGE_STRING_FC command:

*** WARNING *** Feature codes must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET RIDGE_STRING_FC 152<CR>**
TRIEDIT>

SET RIVER_STRING_FC

Set the IFF feature code of features representing river node strings. Used by the IFF 'iff-file-spec' command.

FORMAT: SET RIVER_STRING_FC feature-code

Command parameters:

feature-code

The feature code to be used for IFF features representing river node string features. This must lie in the range 0 to 32767.

DESCRIPTION:

The feature code set up by the SET RIVER_STRING_FC command is used by the IFF 'iff-file-spec' command. It provides a mechanism to enable IFF features representing river node strings to be differentiated from similar features representing unflagged and ridgeline node strings. A different feature code has to be used for multi-point IFF features to that used for single point features to ensure that the feature is plotted correctly. It is normal practice to plot single point IFF features using a symbol (e.g. a dot, cross or a box). Multi-point features are plotted using lines which join the nodes together.

By allowing the user to distinguish between multi-point string features, it is possible to read the IFF file created by the TRIEDIT IFF command into the triangulation module TRIANG for retriangulation. TRIANG has to be told what the strings of nodes represent.

All IFF features representing non-breakline strings (including river and ridgeline nodes) are placed in the same IFF layer (set with the SET LAYER command).

All IFF features representing breakline strings (including river and ridgeline nodes) are placed in a different IFF layer (set with the SET BREAKLINE_LAYER command).

Although all IFF features representing cliffline strings are placed in a third IFF layer (set with the SET CLIFF_LAYER command), it is not possible to have a cliffline string flagged as a river or ridgeline!

When choosing a feature code for multi-point IFF features representing strings of river nodes, ensure that the feature code is in the FRT file which is to be used for plotting the IFF file and that the feature code has an appropriate graphical type, i.e. line.

Note that use of the FACET or IFF commands within TRIEDIT results in the generation of IFF files containing CB (Coordinate Block) entries. CB entries replace the use of ST (STring) and ZS (3D string) entries.

Messages:

The following message is specific to the SET RIVER_STRING_FC command:

*** WARNING *** Feature codes must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET RIVER_STRING_FC 152<CR>**
TRIEDIT>

SET STRING_FC

Set the IFF feature code of features representing unflagged node strings. Used by the IFF 'iff-file-spec' command.

FORMAT: SET STRING_FC feature-code

Command parameters:

feature-code

The feature code to be used for IFF features representing unflagged node string features. This must lie in the range 0 to 32767.

DESCRIPTION:

The feature code set up by the SET STRING_FC command is used by the IFF 'iff-file-spec' command. It provides a mechanism to enable IFF features representing unflagged node strings to be differentiated from similar features representing river and ridgeline node strings. A different feature code has to be used for multi-point IFF features to that used for single point features to ensure that the feature is plotted correctly. It is normal practice to plot single point IFF features using a symbol (e.g. a dot, cross or a box). Multi-point features are plotted using lines which join the nodes together.

By allowing the user to distinguish between multi-point string features, it is possible to read the IFF file created by the TRIEDIT IFF command into the triangulation module TRIANG for retriangulation. TRIANG has to be told what the strings of nodes represent.

All IFF features representing non-breakline strings (including river and ridgeline nodes) are placed in the same IFF layer (set with the SET LAYER command).

All IFF features representing breakline strings (including river and ridgeline nodes) are placed in a different IFF layer (set with the SET BREAKLINE_LAYER command).

All IFF features representing cliffline strings are placed in a third IFF layer (set with the SET CLIFF_LAYER command).

When choosing a feature code for multi-point IFF features representing strings of unflagged nodes, ensure that the feature code is in the FRT file which is to be used for plotting the IFF file and that the feature code has an appropriate graphical type, i.e. line.

Note that use of the FACET or IFF commands within TRIEDIT results in the generation of IFF files containing CB (Coordinate Block) entries. CB entries replace the use of ST (STring) and ZS (3D string) entries.

Messages:

The following message is specific to the SET STRING_FC command:

*** WARNING *** Feature codes must lie in the range 0 to 32767.

Examples:

TRIEDIT> **SET STRING_FC 152<CR>**
TRIEDIT>

SET TRIANGLE_ACCURACY

Sets the level of triangle plotting accuracy required.

FORMAT: SET TRIANGLE_ACCURACY level

Command parameters:

level

The level of triangle plotting accuracy required. This must lie in the range 0-2 (inclusive).

DESCRIPTION:

Triangle links can be plotted on the screen with a variable level of accuracy.

The SET TRIANGLE_ACCURACY command enables the user to set the level of triangle plotting accuracy required.

Triangle accuracy level 0 plots all links that have one end inside the window, and level 1 those links emanating from points outside the window but with neighbours inside it. Level 2 gives maximum performance with full clipping of all links traversing the window. Time rises dramatically, in terms of calculating which points should be considered to be in the window, as the level rises.

Messages:

The following message is specific to the SET TRIANGLE_ACCURACY command:

*** WARNING *** Level must lie in range 0 to 2 - try again

Examples:

TRIEDIT> **SET TRIANGLE_ACCURACY 1<CR>**
TRIEDIT>

SET TYPE

Sets the data type for nodes to be inserted with the INSERT command.

FORMAT: SET TYPE type

Command parameters:

type

The data type to be used for nodes inserted with the INSERT command. Valid data types are:

- NORMAL - slopes are continuous at this node
- BREAKLINE - slopes are discontinuous at this node

DESCRIPTION:

Sets the data type for nodes to be inserted with the next INSERT command. This setting will remain active for all subsequent INSERT commands until an alternative SET TYPE command is issued.

On program startup the INSERT data type is set to NORMAL.

The current data type setting is displayed in the status area.

The TRIEDIT INSERT command does not support the CLIFFLINE data type

---

Messages:

The following message is specific to the SET DATA_TYPE command:

*** WARNING *** Unexpected end of SET TYPE command
Specify either NORMAL or BREAKLINE

Examples:

TRIEDIT> **SET TYPE NORMAL<CR>**
TRIEDIT>

SHOW

Shows current status of TRIEDIT option and parameter settings.

FORMAT: SHOW subject

Command parameters:

subject

The subject that is to be displayed, chosen from:

ASSIGNMENTS	BREAKLINES	DATUM	ENABLE	FC	FILES
FRT	HEIGHTS	IFF_OUTPUT	LAYER	RIDGELINES	
RIVERS	SELECTIONS	UNITS	WINDOW		

DESCRIPTION:

SHOW enables the user to examine the current status of TRIEDIT options and parameter settings.

Messages:

TRIEDIT issues the following message if the SHOW command is specified without an argument:

*** ERROR *** Specifying command SHOW

Available SHOW command qualifiers are:

ASSIGNMENTS	BREAKLINES	DATUM	ENABLE	FC	FILES
FRT	HEIGHTS	IFF_OUTPUT	LAYER	RIDGELINES	
RIVERS	SELECTIONS	UNITS	WINDOW		

This feature can be used to advantage if the user wishes to quickly determine for which items the SHOW facility is available.

Examples:

```
TRIEDIT> SHOW<CR>
*** ERROR *** Specifying command SHOW

Available SHOW command qualifiers are:

ASSIGNMENTS    BREAKLINES     DATUM      ENABLE    FC        FILES

FRT            HEIGHTS        IFF_OUTPUT LAYER     RIDGELINES

RIVERS         SELECTIONS     UNITS      WINDOW

TRIEDIT> SHOW BREAKLINES<CR>

BREAKLINES:
No layers assigned for breaklines
No feature codes assigned for breaklines

TRIEDIT>
```

SHOW ASSIGNMENTS

Shows current status of TRIEDIT feature code and layer assignments made using ASSIGN and DEASSIGN commands.

FORMAT: SHOW ASSIGNMENTS

Command parameters: None.

DESCRIPTION:

Shows current status of TRIEDIT feature code and layer assignments made using ASSIGN and DEASSIGN commands. The SHOW ASSIGNMENTS command has the effect of issuing SHOW BREAKLINES, SHOW RIDGELINES and SHOW RIVERS commands sequentially.

Messages: None.

Examples:

$ **TRIEDIT<CR>**
TRIEDIT> **SHOW ASSIGNMENTS<CR>**

BREAKLINES:
No layers assigned for breaklines
No feature codes assigned for breaklines

RIDGELINES:
No layers assigned for ridgelines
No feature codes assigned for ridgelines

RIVERS:
No layers assigned for rivers
No feature codes assigned for rivers

TRIEDIT>

SHOW BREAKLINES

Shows current status of TRIEDIT feature code and layer breakline assignments made using ASSIGN and DEASSIGN commands.

FORMAT: SHOW BREAKLINES

Command parameters: None.

DESCRIPTION:

Shows current status of TRIEDIT feature code and layer breakline assignments made using ASSIGN and DEASSIGN commands.

Messages: None.

Examples:

TRIEDIT> **SHOW BREAKLINES<CR>**
No layers assigned for breaklines
No feature codes assigned for breaklines
TRIEDIT>

SHOW DATUM

Shows current status of of the height datum which is to be added to all incoming IFF and DTI heights.

FORMAT: SHOW DATUM

Command parameters: None.

DESCRIPTION:

Shows current status of of the height datum which is to be added to all incoming IFF and DTI heights.

Messages: None.

Examples:

TRIEDIT> **SHOW DATUM<CR>**
Height datum 0.0 to be added to all incoming heights
TRIEDIT>

SHOW ENABLE

Shows current status of all TRIEDIT processing options.

FORMAT: SHOW ENABLE

Command parameters: None.

DESCRIPTION:

Shows current status of all TRIEDIT processing options.

Messages: None.

Examples:

```
TRIEDIT> SHOW ENABLE<CR>
DIAGNOSTICS ..................................................... Off
DIVIDEBY ........................................................ OFF
INTEGER_HEIGHT .................................................. Off
(Incoming IFF heights expected in type 3 AC entries)
INVERSE ......................................................... Off
MULTIPLY ........................................................ Off
PME ............................................................. Off
TOFEET .......................................................... Off
TOMETRES ........................................................ Off
TRIEDIT>
```

SHOW FC

Shows current status of TRIEDIT feature code input selections.

FORMAT: SHOW FC

Command parameters: None.

DESCRIPTION:

Shows current status of TRIEDIT feature code input selections.

Messages: None.

Examples:

TRIEDIT> **SHOW FC<CR>**
Feature codes selected for input:
0-32767
TRIEDIT>

SHOW FILES

Shows current status of TRIEDIT input file useage.

FILES: SHOW FILES

Command parameters: None.

DESCRIPTION:

Shows current status of TRIEDIT input files.

SHOW FILES enables the user to determine the names of the triangulation input files. It also enables the user to keep track of which IFF files have been successfully read into TRIEDIT using FILEIN commands. Up to 20 input files can be displayed using the SHOW FILES command.

Messages: None.

Examples:

```
TRIEDIT>SHOW FILES<CR>
.DTA file:   LSL$DATA_ROOT:[DEMONSTRATION]IDAHO.DTA;9
.NOD file:   LSL$DATA_ROOT:[DEMONSTRATION]IDAHO.NOD;9

FILEIN FILES:
No FILEIN input files sucessfully read yet.
```

SHOW FRT

Shows current status of TRIEDIT FRT selection.

FRT: SHOW FRT

Command parameters: None.

DESCRIPTION:

Shows current status of TRIEDIT FRT selection.

Messages: None.

Examples:

TRIEDIT> **SHOW FRT<CR>**
No FRT file selected
TRIEDIT>

SHOW HEIGHTS

Shows current status of TRIEDIT height modification and datum options.

HEIGHTS: SHOW HEIGHTS

Command parameters: None.

DESCRIPTION:

Shows current status of TRIEDIT height modification and datum options.

Messages: None.

Examples:

```
TRIEDIT> SHOW HEIGHTS<CR>
Incoming heights expected in IFF type 3 AC entries
MULTIPLYBY and DIVIDEBY are disabled.
No imperial or metric conversion to be applied to incoming heights
Height datum     0.000 to be added to all incoming heights
No inversion to be applied to incoming heights
TRIEDIT>
```

SHOW IFF_OUTPUT

Shows current status of TRIEDIT IFF output file FC and layer settings.

IFF_OUTPUT: SHOW IFF_OUTPUT

Command parameters: None.

DESCRIPTION:

Shows current status of TRIEDIT IFF output file FC and layer settings.

Messages: None.

Examples:

TRIEDIT> **SHOW IFF_OUTPUT<CR>**

OUTPUT IFF FILE CHARACTERISTICS:

Layer for triangles (DRAW TRIANGLES 'iff-file-spec',
re-contouring (DRAW CONTOURS 'iff-file-spec' and normal node
output (IFF command)..... 1
Layer for breakline node output..... 2
Layer for cliffline node output..... 3
Feature code for plot frame..... 4
Feature code for strings of normal nodes (IFF command) 8
Feature code for individual normal nodes (IFF command) 0
Feature code for strings of river nodes (IFF command) 11
Feature code for individual river nodes (IFF command) 2
Feature code for strings of ridge nodes (IFF command) 19
Feature code for individual ridge nodes (IFF command) 0
Feature code for cliff strings (IFF command) 9
Feature code for normal triangle links
DRAW TRIANGLES 'iff-file-spec' command) 78
Feature code for breakline triangle links (DRAW TRIANGLES
'iff-file-spec' command) 21
Feature code for imaginary triangle links (DRAW TRIANGLES
'iff-file-spec' command) 7
Feature code for intermediate contours (DRAW CONTOURS
'iff-file-spec' command) 6
Feature code for index contours (DRAW CONTOURS
'iff-file-spec' command) 1
TRIEDIT>

SHOW LAYER

Shows current status of TRIEDIT layer input selections.

FORMAT: SHOW LAYER

Command parameters: None.

DESCRIPTION:

Shows current status of TRIEDIT layer input selections.

Messages: None.

Examples:

TRIEDIT> **SHOW LAYER<CR>**
Layers selected for input:
0-32767
TRIEDIT>

SHOW RIDGELINES

Shows current status of TRIEDIT feature code and layer ridgeline assignments made using ASSIGN and DEASSIGN commands.

FORMAT: SHOW RIDGELINES

Command parameters: None.

DESCRIPTION:

Shows current status of TRIEDIT feature code and layer ridgeline assignments made using ASSIGN and DEASSIGN commands.

Messages: None.

Examples:

```
TRIEDIT> SHOW RIDGELINES<CR>
No layers assigned for ridgelines
No feature codes assigned for ridgelines
TRIEDIT>
```

SHOW RIVERS

Shows current status of TRIEDIT feature code and layer river assignments made using ASSIGN and DEASSIGN commands.

FORMAT: SHOW RIVERS

Command parameters: None.

DESCRIPTION:

Shows current status of TRIEDIT feature code and layer river assignments made using ASSIGN and DEASSIGN commands.

Messages: None.

Examples:

TRIEDIT> **SHOW RIVERS<CR>**
No layers assigned for rivers
No feature codes assigned for rivers
TRIEDIT>

SHOW SELECTIONS

Shows current status of TRIEDIT layer and feature code input selections.

FORMAT: SHOW SELECTIONS

Command parameters: None.

DESCRIPTION:

Shows current status of TRIEDIT layer and feature code input selections.

SHOW selections has the same effect as issuing SHOW FC and SHOW LAYER commands sequentially.

Messages: None.

Examples:

TRIEDIT> **SHOW SELECTIONS<CR>**
Feature codes selected for input:
0-32767
Layers selected for input:
0-32767
TRIEDIT>

SHOW UNITS

Shows current status of TRIEDIT window units as set using the UNITS command.

FORMAT: SHOW UNITS

Command parameters: None.

DESCRIPTION:

Shows current status of TRIEDIT window units as set using the UNITS command.

Messages: None.

Examples:

TRIEDIT> **SHOW UNITS<CR>**
Window to be specified in metres
TRIEDIT>

SHOW WINDOW

Shows current triangulation window values.

FORMAT: SHOW WINDOW

Command parameters: None.

DESCRIPTION:

Shows current triangulation window values.

The units of the WINDOW command parameters are set using the UNITS command. By default metre units are assumed. If it is more convenient to specify the window in latitude and longitude the sexagesimal lat long values may be supplied after specifying a UNITS LATLONG command. This assumes that the data read in from the DTI and IFF files are in units of tenths second of arc. A similar assumption is made for UNITS SECONDS.

The SHOW WINDOW command displays the window values in the units currently selected by the UNITS command. No projection transformation is performed, so the unwise user could easily specify that the window be shown as latitude longitude despite the fact that his data are in metres!

Messages: None.

Examples:

TRIEDIT> **SHOW WINDOW<CR>**
Window units are metres

Triangulation coverage SW: 494600.00 171000.00 NE: 496100.00 173400.00

Triangulation window SW: 494599.19 170999.20 NE: 496100.84 173400.98
TRIEDIT>

SPAWN

The SPAWN command enables you to create a subprocess while within TRIEDIT.

FORMAT: SPAWN command-line

Command parameters:

command-line

Specifies a DCL command string to be executed as if typed in response to a '$' prompt. When the command completes, the subprocess terminates and control is returned to TRIEDIT. The command string cannot exceed 80 characters.

DESCRIPTION:

The SPAWN command enables you to create a subprocess while within TRIEDIT. When the subprocess terminates control is returned to TRIEDIT.

Messages:

The following warning messages are specific to the SPAWN command:

*** WARNING *** SPAWN requires a valid DCL command line

*** ERROR *** Unable to spawn command, returning to TRIEDIT

Examples:

TRIEDIT> **SPAWN DIRECTORY *.DTA;*<CR>**

```
Directory DUA3:[DTMCREATE.ACCEPTANCE_TESTS]

TEST1.DTA;1              8/8       18-AUG-1987 07:56  [LSL,TIM]
TEST2.DTA;2              7/8       18-AUG-1987 17:17  [LSL,TIM]
TEST2.DTA;1              7/8       18-AUG-1987 17:07  [LSL,TIM]

Total of 3 files, 22/24 blocks.
```

TRIEDIT>

SWAP

Specifies that the selected inter-node link is to be swapped diagonally within the quadrilateral formed by the four nodes defining the two triangles which share the selected link.

FORMAT: SWAP

Command parameters: None.

DESCRIPTION:

The SWAP command enables the user to specify that the selected inter-node link is to be swapped diagonally within the quadrilateral formed by the four nodes defining the two triangles which share the selected link.

After typing the SWAP command, place the cursor on the inter-node link that is to swapped. Press the rightmost function button. If the cursor is too far from the nearest link a warning message will be issued. If it is possible to swap the link the old link will be obliterated with symbols and the path of the new swapped link indicated with a line of open symbols.

There are instances when it is not possible to swap an inter-node link within a quadrilateral. If the quadrilateral is re-entrant, crossing internode links may occur if the swap is allowed to take place. Checks are performed and warning messages issued if a harmful SWAP has been attempted. The triangulation will not be changed.

It is usually possible to overcome the problem of a re-entrant quadrilateral by swapping other inter-node links around it first until the distribution of triangle links is better.

Messages:

The following warning messages are specific to the SWAP command:

*** WARNING **** Bad intersection with adjacent triangle would occur

*** WARNING **** Cliffline or zero width triangle detected.

*** WARNING **** Unable to swap diagonal - polygon is not quadrilateral

*** WARNING **** Unable to swap diagonal - polygon is re-entrant

---

Examples:

TRIEDIT> **SWAP<CR>**
TRIEDIT>

UNITS

Specifies the units of measurement that will be used when defining the display window using the WINDOW command.

The command also controls the units of measurement which will be used when displaying output from the POSITION command.

FORMAT: UNITS units

Command parameters:

units

A keyword defining the measurement units, chosen from:

METRES	Metres on the ground
LATLONG	Latitude and Longitude (in degrees, minutes and seconds)
SECONDS	Seconds of arc
PROJECTION	Projection units

DESCRIPTION:

The UNITS command enables the user to specify in what units of measurement he wishes to define the triangulation window using the WINDOW command, or in what units of measurement details from the POSITION command are displayed.

By default metre units are assumed.
The UNITS command should be given before specifying the display window if the user wishes to specify the window in non-metre units.

If UNITS SECONDS or UNITS LATLONG are used it is assumed that the triangulation data are in units of tenth seconds of arc.

UNITS METRES (the default) or UNITS PROJECTION assume that the the data read in using FILEIN commands are in metres or projection units. In a future release of DTMCREATE coordinate consistency checks between input files and units specified with the UNITS command will be implemented.

Messages:

The following error messages are specific to the UNITS command:

*** ERROR *** Specifying command UNITS
Command qualifiers are METRES,PROJECTION,SECONDS or LATLONG

Examples:

```
TRIEDIT> UNITS LATLONG<CR>
TRIEDIT> WINDOW 52 00 00N 08 30 00 E 52 30 00 N 09 00 00 E<CR>
TRIEDIT>
```

WAIT

Suspend processing for the specified number of seconds.

FORMAT: WAIT seconds

Command parameters:

seconds

The number (floating point) of seconds for which TRIEDIT processing is to be suspended.

DESCRIPTION:

The WAIT command causes processing to be suspended for a specified number of seconds. It is designed for use in software demonstration situations and is of no value in a production flowline.

Messages:

The following warning message is specific to the WAIT command:

*** WARNING *** You must specify the number of seconds to wait

---

Examples:

TRIEDIT> **WAIT 4.0<CR>**
TRIEDIT>

WINDOW

Specifies the limits of the data area to be displayed.

FORMAT: WINDOW [xmin ymin xmax ymax]

Command parameters:

The window command parameters are optional. If omitted, the user is expected to define the WINDOW using the GIN cursor.

xmin ymin

The coordinates of the bottom left hand corner of the defining rectangle.

xmax ymax

The coordinates of top right hand corner of the defining rectangle.

The units used to specify the WINDOW command parameters are determined by the UNITS command. By default UNITS METRES is assumed.

In the case of UNITS LATLONG, window values should be expressed as south-west **Latitude Longitude** north-east **Latitude Longitude**, **not Longitude Latitude**. The SHOW WINDOW command will reflect the window extent in both LATLONG units and tenth seconds arc.

DESCRIPTION:

The command is used to define rectangular limits to the area of the triangulation to be displayed on the graphics screen.

The WINDOW command differs from the ZOOM command in that the area of the triangulation currently displayed is shown as a box on the graphics display. The whole area of triangulation coverage is portrayed by an outer box. If the current window covers the whole triangulation only a single, full sized box will be displayed.

The window command parameters are optional. If omitted, the user is expected to define the WINDOW using the cursor.

If WINDOW command parameters are specified the WINDOW command may be used in conjunction with the NOGRAPHICS graphics device option, i.e. at an ordinary VT100 compatible terminal.

If the optional WINDOW command parameters are omitted, the WINDOW limits must be specified by moving the cursor to one corner of the desired window and pressing the rightmost function button. An arrow head symbol will appear on the screen. Move the cursor to the corner of the desired window which is diagonally opposite to the first selected corner (which lies at the location of the arrow head

symbol). Press the rightmost function button. A progress bar will appear in the top left corner of the graphics screen. When data windowing is complete, the screen will be cleared and a new window border will be displayed. You may now issue DRAW commands and start editing data within the new window.

If the two window definition nodes are superimposed (either deliberately or by careless function button presses) TRIEDIT will restore the window to cover the whole triangulation.

The current WINDOW limits can be examined using the SHOW WINDOW command.

Note that the geographic limits defined by the WINDOW command are not used by the FILEIN command. If data within an IFF file used for input lies within the triangulation area it will be inserted into the triangulation.

Messages:

The following warning messages are specific to the WINDOW command:

*** WARNING **** Superimposed window corners,
returning to full window.

*** WARNING **** Zero width window specified,
returning to full window.

*** WARNING **** Zero height window specified,
returning to full window.

*** ERROR **** Unable to read WINDOW arguments

*** ERROR **** You must give a value for all four geographic bounds for
the WINDOW in the order XMIN YMIN XMAX YMAX
or use the cursor

*** ERROR **** window values must be given in the order:
XMIN YMIN XMAX YMAX

*** ERROR **** window values define a zero width window

*** ERROR **** window values define a zero height window

Examples:

TRIEDIT> **WINDOW<CR>**
TRIEDIT> **DRAW TRIANGLES<CR>**
TRIEDIT> **WINDOW 0.0 0.0 120.0 120.0<CR>**
TRIEDIT>

ZOOM

Redraws the area around the cursor enlarged by the specified zoom factor.

FORMAT: ZOOM [zoom-factor]

Command parameters:

zoom-factor

The zoom factor to be applied in the range 0.01 to 100.0. The default factor is 5.0. A zoom factor that is less than 1.0 will have the effect of zoom reduction i.e. unzooming. Thus a ZOOM .2 command will increase the area of triangulation displayed on the screen by 5 times.

DESCRIPTION:

The centre of the ZOOM must be specified by moving the cursor to the position which is to be at the centre of the zoomed screen and pressing the rightmost function button.

The aspect ratio of the zoomed screen window will be taken from the current screen window, limited by triangulation availability.

Messages:

The following warning message is specific to the ZOOM command:

*** WARNING **** Argument out of range
ZOOM command ignored

Examples:

TRIEDIT> **ZOOM<CR>**
TRIEDIT> **DRAW TRIANGLES<CR>**
TRIEDIT> **ZOOM 3.0<CR>**
TRIEDIT> **DRAW TRIANGLES<CR>**
TRIEDIT> **ZOOM 0.333<CR>**
TRIEDIT> **DRAW TRIANGLES<CR>**
TRIEDIT>

MESSAGES (WARNING)

These messages are output when an error has occurred that can be corrected immediately by the user or that the program will attempt to overcome.

NOV2MD, IFF map descriptor in %S is not version 2

> **Explanation:** TRIEDIT expects input files to have type 2 map descriptors as it offers offset merging functionality based on the contents of the map descriptor. TRIEDIT is downwards compatible with old pattern IFF files which have type 1 map descriptors, but no origin offset facility is supported for the earlier pattern files.
>
> **User action:** If origin offsetting is required use ITRANS/DESCRIPTOR to create a copy of the IFF file having a type 2 map descriptor.

TOMNYTRI, Too many triangles in window for IFF file

> **Explanation:** IFF triangle features are generated by the FACET command for that portion of the triangulation that lies within the current screen window. The FACET command can only create an IFF file containing a maximum of 65535 triangle features.
>
> **User action:** If the triangulation contains more than this number the WINDOW command should be used in conjunction with multiple FACET commands to subdivide the triangulation into several IFF output IFF files.

UNSETMD, map descriptor in %S is unset

> **Explanation:** The Map Descriptor in the specified IFF input file is unset.
>
> **User action:** If origin offsetting is required or other input files have characteristics which require the map descriptor to be set, use ITRANS/DESCRIPTOR to set up the map descriptor. Re-run TRIEDIT.

MESSAGES (ERROR)

These messages indicate an error in processing which will cause the program to terminate. The most likely causes are a corrupt or otherwise invalid input file, or an error related to command line processing and file manipulation.

OPNSCR, error opening scratch file SYS$DISK:[]TRIEDIT.TMP

> **Explanation:** The data set read in has exceeded the current TRIEDIT memory quota and a random access disk file was going to be opened to contain the surplus. However, TRIEDIT has failed to open this random access file.
>
> **User action:** The supplementary message given after this error should help you to decide what has gone wrong (e.g. disk full, file protection error, etc.). Correct this problem and then re-run TRIEDIT.

RDDTA, error reading from .DTA file

> **Explanation:** An error has occurred while reading the .DTA file on disk. This error message will be accompanied by a supplementary RMS message which will indicate the cause of the error.
>
> **User action:** This depends on the cause of the error. Correct the cause of the error (e.g. insufficient privilege or file protection violation) before attempting to re-run TRIANG.

STACKOVR, stack overflow - triangulation will be destroyed

> **Explanation:** TRIEDIT has only a finite amount of space available to store new points that have been inserted into the data structure. You have now used up all this space and TRIEDIT will have to give up, losing the triangulation in the process. It is very unlikely that you will get this message as a minimum of 33% of workspace is reserved for storage of points generated during constraint. The data supplied to TRIEDIT must have been very poorly distributed to have caused this failure.
>
> **User action:** Use LITES2 to check the distribution of data points, or run TRIEDIT in graphics mode and note the distribution of triangles immediately prior to failure. If the problem persists please submit an SPR to Laser-Scan.

TOOMNYNODW, too many nodes in window

> **Explanation:** While inserting new points TRIEDIT may have to replace an old window node with a new one supplied by the user. This has now happened too may times and there is no room left to store the changes.
>
> **User action:** Select the SAVE option from the QUIT/SAVE alternative offered after the crash which will allow TRIEDIT to invoke the garbage collection routine to tidy up the loose ends and write out new .NOD and .DTA files.

UNEXPEOF, unexpected end of IFF file

Explanation: This message indicates there is something seriously wrong with the IFF file which has caused immediate termination of the program. TRIEDIT has detected the end of the IFF file, but has not detected an IFF 'EJ' entry.

User action: Use IMEND on the file, which will correctly position the EOF marker and insert an EJ entry at the end of the file. Re-run TRIEDIT on the corrected file.

WRTDTA, error writing to .DTA file

Explanation: An error has occurred while writing to the .DTA file on disk. This error message will be accompanied by a supplementary RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient disk space) before attempting to re-run TRIEDIT.

WRTNOD, error writing to .NOD file

Explanation: An error has occurred while writing to the .NOD file on disk. This error message will be accompanied by a supplementary RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient disk space) before attempting to re-run TRIEDIT.

WRTSCR, error writing to scratch file

Explanation: An error has occurred while writing to the a temporary scratch file on disk. This error message will be accompanied by a supplementary RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient disk space) before attempting to re-run TRIEDIT.

MESSAGES (FATAL)

These messages indicate a severe error in processing, or some form of system failure, which has caused the program to terminate.

ARITHMETIC, arithmetic exception detected

Explanation: TRIEDIT contains an arithmetic exception handler which invokes the "QUIT or SAVE" user escape route in the event of arithmetic error. Such an error has just occurred, the precise nature of which is defined by the accompanying message.

User action: Select the SAVE option from the QUIT/SAVE alternative offered after the crash which will allow TRIEDIT to invoke the garbage collection routine to tidy up the loose ends and write out new .NOD and .DTA files. Please save all the data used for input and then submit an SPR to Laser-Scan.

BOXOVR, too little space for boxes

Explanation: TRIEDIT uses the box structure created by TRIANG. Your current version of TRIEDIT is insufficiently dimensioned to cope with the number of boxes just read in.

User action: This should never happen as the DTMCREATE modules are dimensioned to be mutually compatible! Please report this error to Laser-Scan. Until TRIEDIT can be redimensioned divide up your original IFF file and re-run TRIANG on the resulting sub-areas. TRIEDIT will then probably be able to cope with the reduced data set size. The resulting sub-DTMs can be joined to form the whole DTM area using DTITILE.

BUFFOVR, buffer overflow detected in PLOTQ

Explanation: If an extremely long string insertion is attempted then the buffer used to hold the inserted point coordinates for height interpolation may be filled. Normally the buffer automatically flushes before overflow can occur. The buffer has overflowed. This should not occur.

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

DUFFNOD, duff node found in neighbour list - GARBAG

Explanation: When internal space becomes full TRIEDIT garbage collects as much as possible by removing all duff points from the data set, and from the memory/disk file. Duff points are points which have been deleted by the DELETE command, or which have been replaced by another point as a result of an INSERT or FILEIN command. This may make a vital difference for complex much overwritten data sets. Neighbour list for each node are obtained to allow the gaps for duff nodes in the data list to be removed. There appears (incorrectly) to be a duff node still within a neighbour list. There is no hope of continuing the editing session.

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

LIST, NODB not in list for NODA

Explanation: This error should never occur! If it does it will be associated with an INSERT or FILEIN command. As new points are added to the data set new inter-node relationships are formed in memory which will be added to the node/neighbour file. A new node has been formed within an existing triangulation but now that the inter-node links are being formed, TRIEDIT is unable to locate one of the original triangle vertices (node B) in the neighbour list of node A.

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

LOST, TRIEDIT is lost - 4th point believed outside the imaginary point frame!

Explanation: This error should never occur! If it does it will be associated with an INSERT or FILEIN command. As new points are added to the data set new inter-node relationships are formed in memory which will be added to the node/neighbour file. However, the search for the neighbours for one of the new points has failed and TRIEDIT has begun looking outside of the imaginary point frame!

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

LOST4TH, GTFRTH could not find a 4th point

Explanation: This error should never occur! If it does it will be associated with an INSERT or FILEIN command. As new points are added to the data set new inter-node relationships are formed in memory which will be added to the node/neighbour file. A new node has been formed within an existing triangle but now that the inter-node links are being formed, TRIEDIT is unable to locate one of the original triangle vertices.

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

NODEOF, unexpected end of node file

Explanation: TRIEDIT has read off the end of the node file.

User action: Unless another problem has occurred during this TRIEDIT session run this error should not occur. Please save all the data used for input and then submit an SPR to Laser-Scan.

NODOVR, node has more than 150 neighbours

Explanation: TRIEDIT can currently can only handle nodes with less than 150 neighbours. The data is almost certainly very corrupt if this message appears, possibly as a result of over enthusiastic data insertion in TRIEDIT. Normally a node will only have up to about 9 neighbours.

User action: The triangulation is irrevocably damaged. Re-run TRIANG and try running TRIEDIT again. If the problem persists please contact Laser-Scan.

NODPAIR, NODA/NODB pair not found in routine INJOIN

Explanation: Another error that should never occur! If it does it will be associated with an INSERT or FILEIN command. As new points are added to the data set new inter-node relationships are formed in memory which will be added to the node/neighbour file. TRIEDIT is unable to locate a link between two triangle vertices.

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

NOLD, NOLD not found in RECONN

Explanation: When performing node substitution, TRIEDIT has to reconnect the neighbours of the old node to the new node. Hovever, TRIEDIT cannot find the old node in the neighbour list to substitute with the new node.

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

NOPOINT, closest point does not exist - RIPPLE

Explanation: TRIEDIT can find no point near the cursor position.

User action: Select the SAVE option from the QUIT/SAVE alternative offered after the crash which will allow TRIEDIT to invoke the garbage collection routine to tidy up the loose ends and write out new .NOD and .DTA files.

NOSPAWN, no spawn space left in GETTRY

Explanation: TRIEDIT has run out of workspace while rippling out through potential neighbours looking for the current point.

User action: Select the QUIT option, do not re-use the .NOD and .DTA files. Please save all the data used for input and then submit an SPR to Laser-Scan.

NOTRI, no triangle chosen in DELPNT

Explanation: A triangle having the current point as a vertex cannot be determined. This should never occur.

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

PTLOST, point cannot be found by GETTRY

Explanation: TRIEDIT checks that there are some nodes to check - none can be found. Something quite awful has happened, as the point IX,IY cannot be found in any of the existing triangles!

User action: Select the QUIT option, do not re-use the .NOD and .DTA files. Please save all the data used for input and then submit an SPR to Laser-Scan.

RANDRD, error during random read

Explanation: An error has occurred reading from a random access disk file used when there are too many nodes to hold in memory. This error message will be accompanied by a supplementary FORTRAN or RMS message which will indicate the cause of the error.

User action: Please save all the data used for input and then submit an SPR to Laser-Scan.

RANDWRT, error during random write in routine %S

Explanation: An error has occurred writing to a random access disk file used when there are too many nodes to hold in memory. This error message will be accompanied by a supplementary FORTRAN or RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient disk space) before attempting to re-run TRIEDIT

RDNOD, error reading .NOD file

Explanation: TRIEDIT has suffered a read error when reading back from the .NOD file.

User action: Unless another problem has occurred during this TRIEDIT run this error should not occur. Please save all the data used for input and then submit an SPR to Laser-Scan.

STACKRET, too many returns to stack

Explanation: While removing superceded nodes eg after an insertion, TRIEDIT follows the affected string through the structure and updates the neighbour lists for the affected points. For each update it must return the record to the stack and update the stack pointer, which is now off the bottom of the usable area of the stack! The data has somehow become corrupted.

User action: Select the SAVE option from the QUIT/SAVE alternative offered after the crash which will allow TRIEDIT to invoke the garbage collection routine to tidy up the loose ends and write out new .NOD and .DTA files.

TOMNYNEIB, too many neighbours

Explanation: This is a problem which only arises when the source data is very unevenly distributed. The addition of some additional formlines to the source data-set will always solve this problem.

User action: Use LITES2 to add formlines in the IFF input file in the area of uneven data distribution and the re-run TRIEDIT.

UNRECREC, unrecognised record number

Explanation: Each node entry in workspace is identified by a positive record number, if there are more records than can be held in memory then the remainder are written to a random access disk file. Somehow TRIEDIT has found a record with an identification that is either less than 1 or greater than the current maximum recorded number of records.

User action: Unless another problem has occurred during this TRIEDIT run this error should not occur. Please save all the data used for input and then submit an SPR to Laser-Scan.

MESSAGES (OTHER)

In addition to the above messages which are generated by the program itself, other messages may be produced by the command line interpreter (CLI) and by Laser-Scan libraries. In particular, messages may be generated by the IFF library and by the Laser-Scan I/O library, LSLLIB. IFF library messages are introduced by '%IFF' and are documented in the IFF library users' guide. In most cases IFF errors will be due to a corrupt input file, and this should be the first area of investigation. If the cause of the error cannot be traced by the user, and Laser-Scan are consulted, then the output file should be preserved to facilitate diagnosis. LSLLIB messages are introduced by '%LSLLIB' and are generally self-explanatory. They are used to explain the details of program generated errors.

CHAPTER 7

MODULE TRIGRID

MODULE TRIGRID

REPLACES PANACEA module PANDORA.

FUNCTION

TRIGRID takes the triangulation node and data files created by TRIANG (or edited output from TRIEDIT) and the slope derivative file generated by TRIDER, and produces a DTI (Digital Terrain Image) output file containing a regular grid DTM.

FORMAT

$ TRIGRID

COMMAND QUALIFIERS

None, TRIGRID is command driven.

DESCRIPTION

General

TRIGRID takes the triangulation node and data files created by TRIANG (or edited output from TRIEDIT) and the slope derivative file generated by TRIDER, and produces a DTI (Digital Terrain Image) output file containing a regular grid DTM.

Once the triangulation and derivative input files are available, TRIGRID may be used repeatedly to produce DTM grids from the same triangulation area at differing resolutions.

TRIGRID offers 2 types of interpolation from the triangular form to the DTM grid.

1. a linear facet option, (see DISABLE SMOOTH command)

2. a SMOOTH patch option, (see ENABLE SMOOTH command)

The smooth patch option is enabled by default.

The advantage of using the linear FACET approach is that it is faster and will definitely produce no values outside the z-range of the data set. If a surface is well defined by contours and spot heights or a quick "look-see" model is required then the FACET option is strongly recommended.

The SMOOTH patch option is the most complicated as it generates a fully edge continuous interpolation across the triangulated area. The smooth patch option uses a quintic surface patch fitted to the vertices and estimated derivatives of the triangles to interpolate a grid node location falling within the triangle. The option should be used with caution, as in areas of very sparse data points the results are those of a mathematical patch, NOT necessarily those of a physical environment function! There are however modifying triangle limits (see Commands section) which may be applied in association with the smooth patch option, which can be used to prevent undershoot and overshoot outside a specified z-range. There are no triangle limits to be set up for the linear FACET option.

The size of the DTM generated by TRIGRID is controlled by either:

1. specifying the geographical extent of the model with the WINDOW command (an obligatory command) and then specifying the DTM cell side length in X and Y, using the SIDELENGTH command. TRIGRID then calculates the resulting number of DTM rows and columns, or,

2. specifying the geographical extent of the model with the WINDOW command and then explicitly stating the number of columns and rows using the SIZE command.

OBLIGATORY COMMANDS

For the simplest possible gridding run TRIGRID requires that the following obligatory commands are given:

1) WINDOW
2) SIZE (or SIDELENGTH)
3) FILEIN
4) FILEOUT
5) GO

1. **Obligatory command WINDOW**

WINDOW 'xmin' 'ymin' 'xmax' 'ymax' - determines the geographical extent of the DTM to be gridded, for example:

WINDOW 40.0 20.0 100.0 100.0

defines the area of the DTM as lying between forty and one hundred units in X and twenty and one hundred units in Y. Data in the triangulation files which lie outside the specified area will be ignored.

The specified extent need not cover the whole area contained in the structured data files. It is possible to "window" out sub-areas of DTM and patch them together to form a complete model using the MATRIX DTITILE utility at a later date.

2. **Obligatory command SIZE or SIDELENGTH**

EITHER:

SIZE 'ncols' 'nrows' - enables the user to explicitly force the model size to be **ncol** columns and **nrow** rows. This an alternative to the SIDELENGTH command.

OR:

SIDELENGTH 'x_step' 'y_step' - enables the user to explicitly set the sidelength in X and Y for the individual DTM cells. Using SIDELENGTH the number of rows and columns is calculated for you.

Failure to define the grid WINDOW will result in the message:

You must set the grid SIDELENGTH (or use SIZE) and WINDOW before gridding can begin.

when you attempt to issue a GO command.

When you use SIZE the minimum possible number of rows and columns is 3. If you try to specify less than this the message:

You must have at least 3 rows and 3 columns in your DTM

will appear.

Failure to specify a size for both rows and columns will result in TRIGRID assuming that you wish both sides to be equal and the message:

Number of columns and rows assumed equal ('integer' 'integer')

will appear.

If you have already issued a SIDELENGTH command then the sidelength values that you specified will be replaced by the values calculated from WINDOW divided by the number of rows and number of columns and the message:

Over-riding SIDELENGTH settings

will appear.

If you have already specified values for the WINDOW then TRIGRID will calculate the cell sidelength required to give you the the number of rows and columns specified by the SIZE command and the results of this calculation will be displayed:

Window units are metres

Triangulation coverage SW: 'real' 'real' NE: 'real' 'real'
Triangulation window SW: 'real' 'real' NE: 'real' 'real'

Which with a side length of 10.00 in X and 10.00 in Y

gives 'integer' rows and 'integer' columns in the dtm

You may use the SIDELENGTH, SIZE and WINDOW commands in combination as often as you wish until the desired model characteristics are achieved.

3. **Obligatory command FILEIN**

FILEIN 'file-spec' - This command causes the specified triangulation files (.NOD, .DTA and .DER) to be read as input.

4. **Obligatory command FILEOUT**

FILEOUT 'file-spec' - create this DTI file for output.

5. **Obligatory command GO**

GO - start processing.

Typical Command Sequence

A typical command sequence is:

```
TRIGRID>  WINDOW 0.0 0.0 80.0 160.0! determine area of DTM
TRIGRID>  SIDELENGTH 10.0 12.0     ! cell sidelength in X and in Y
TRIGRID>  ZLIMITS 0.0 456.0        ! overall DTM interpolation limits
TRIGRID>  TRIANGLE_LIMITS 3.0 9.0  ! individual triangle interpolation limits
TRIGRID>  FILEIN FRED              ! read in FRED_.DTA, FRED_.NOD, and FRED_.DER
TRIGRID>  ENABLE TRACE             ! trace along original data strings for
TRIGRID>                           ! up-hill/down-hill side of line information
TRIGRID>  DATA_TYPE REAL           ! output DTM posts as real (floating point)
TRIGRID>  FILEOUT TEST3.DTI        ! create DTI file LSL$DTI:TEST3.DTI a
TRIGRID>                           ! the output file
TRIGRID>  GO                       ! go!
```

TRIGRID input files

TRIGRID expects as input the binary structured data files (the matched pair of .NOD and .DTA files) produced by TRIANG or TRIEDIT. It also needs the slope derivative (.DER) file produced by TRIDER. All three files must share the same generic filename and must have the same version number. The single TRIGRID FILEIN command supplies a file-spec which is used as a generic file-spec for all three input

(.NOD, .DTA and .DER) files.

Any parts missing from the generic file-specification are taken from the defaults SYS$DISK:[].DTA;0 and SYS$DISK:[].NOD;0.

Since it is essential that the file version numbers of the .NOD, .DTA and .DER file always match, TRIGRID performs checks on file version numbers. If mismatches are found, TRIGRID complains and aborts execution.

Never use the VAX/VMS RENAME or COPY commands to alter the version numbers of .NOD, .DTA files or .DER files to make them into a matched set. TRIDER will have to be re-run every time that the .NOD and .DTA files are modified using TRIEDIT.

The output files from TRIANG, TRIEDIT and TRIDER will be scaled to lie between 0 and the value defined by logical, LSL$DTMCREATE_RESOLUTION, or 300000 if the logical is not defined. The valid range for this resolution is between 300000 and 10000000.

TRIGRID uses this logical value to determine the internal resolution of the .NOD, .DTA and .DER files.

IMPORTANT

It is therefore essential that the logical value LSL$DTMCREATE_RESOLUTION remains the same when going from TRIANG all the way through to TRIGRID on a particular dataset. If the resolution is altered between any of the stages, unpredictable results will occur and programs may fail.

TRIGRID output files

TRIGRID generates output files in DTI (Digital Terrain Image) format, (for a description of DTI format see the MATRIX Reference Manual). TRIGRID offers the choice of output of 3 DTI data types:

- WORD
- LONGWORD
- REAL

These may be set using the DATA_TYPE command.

By default a DTI file of data type WORD is generated.

TRIGRID also offers the user a choice of 3 DTI header types:

- TED4
- UHL1
- LSLA

By default the DTI file is given the LSLA type header.

If a set type 2 IFF MD (Map Descriptor) or a set DTI file projection record were available in the first file read into TRIANG using a FILEIN command, TRIANG puts this information into the .NOD and .DTA files. If TRIGRID is creating a DTI file with an LSLA type header this projection information is copied from the .DTA amd .NOD files into the TRIGRID DTI file header after appropriate modifications of extent, origin and gridstep.

The output DTI file header type may be set using the HEADER_TYPE command.

For details of the characteristics of the different DTI file header types see the Matrix Reference Manual. For information about which of the header characteristics are set by TRIGRID see the HEADER_TYPE command below.

TRIGRID graphics output option

TRIGRID offers the user graphics output of the type offered by TRIANG and TRIDER. The convoluted process of grid height estimation may be observed! The graphics option is a useful aid for instruction or for the analysis of troublesome data sets.

TRIGRID commands

@

Take command input from the specified file.

FORMAT: @file-spec<CR>

Command parameters:

file-spec

The file to be opened and used for command input.

Any parts of the file-spec not supplied for the @ command will be taken from the default specification 'SYS$DISK:[].COM;0'.

DESCRIPTION:

TRIGRID offers the facility of command input from an indirect command file. The '@' character preceding a file-spec will cause TRIGRID to open and read commands from the specified file until:

1. a RETURN command is detected and command input is returned to SYS$COMMAND.

2. a GO command is detected - after completion of grid generation TRIGRID exits.

3. end-of-file is detected. This provokes an error message and command input is returned to SYS$COMMAND.

Nested command files are not supported (i.e. a command file containing an '@' command), although sequential '@' commands are supported when read from SYS$COMMAND.

As an aid to batch log interpretation TRIGRID will echo all commands read from an indirect command file.

Messages:

The following messages are specific to the @ command:

*** WARNING *** "@" must precede a file-spec

*** WARNING *** Indirect file error - returning to terminal input

*** ERROR *** Can't open indirect command file 'file-spec'

Examples:

```
$ TRIGRID<CR>
DTMCREATE module TRIGRID of 16:30:12 20-NOV-87
TRIGRID> @FLOW2<CR>
TRIGRID> ENABLE DIAGNOSTICS
TRIGRID> ENABLE GRAPHICS
TRIGRID> RETURN
TRIGRID>
```

!

Treat all text to the right of the '!' as a comment.

FORMAT: ! [comment text]

Command parameters:

comment text

text that is to be treated as a comment and which will be excluded from command interpretation.

DESCRIPTION:

An exclamation mark is the standard DTMCREATE package comment delimiter. All text (and numbers) which lie to the right of a '!' character are excluded from command interpretation. Comments are useful for annotating command procedures used in batch processing etc.

Messages: None.

Examples:

```
TRIGRID> ! a comment for the sake of it<CR>
TRIGRID> ENABLE GRAPHICS !  turn graphics on<CR>
TRIGRID>
```

DATA_TYPE

The DATA_TYPE command specifies the data type of the posts in the output DTI file.

FORMAT: DATA_TYPE type

Command parameters:

type

A data type keyword chosen from:

- WORD - generate 16 bit integer DTM posts (default)
- LONGWORD - generate 32 bit integer DTM posts
- REAL - generate 32 bit real (floating point) DTM posts

DESCRIPTION:

The DATA_TYPE allows the user to specify the data type of the output DTI file which contains the DTM grid.

By default TRIGRID generates a 'MIKE' type DTI file having WORD (16 bit integer) post values.

When choosing the data type of the DTI file, consideration must be given to the Z range of the data to be output. TRIEDIT will output the DTMCREATE default null value (-32767) if data underflow or overflow occurs when writing to the DTI file.

Consideration must also be given to the nature of the output DTM. Remember that integer DTI data types (WORD and LONGWORD) quantise the Z data values. By default Z values are truncated to the next lower integer value. If the NINT option is enabled, Z values in integer DTI files are rounded to the nearest whole integer.

Messages: None.

Examples:

```
TRIGRID> DATA_TYPE REAL<CR>
TRIGRID> SHOW DATA_TYPE<CR>
DTI data type ..................................... REAL
```

TRIGRID>

DISABLE AUTO_LIMITS

Disables a previous ENABLE AUTO_LIMITS command.

FORMAT: DISABLE AUTO_LIMITS

Command parameters: None.

DESCRIPTION:

DISABLE AUTO_LIMITS allows the user to cancel the effect of a previous ENABLE AUTO_LIMITS command.

When the AUTO_LIMITS option is active, TRIGRID applies interpolation limits based on the feature flag information attached to each node in the triangulation. It is then impossible, for example, for a grid height to be set below the height of a triangle vertex that is flagged as being part of a river string.

By default, the AUTO_LIMITS option is enabled during program startup.

Messages: None.

Examples:

```
TRIGRID> ENABLE AUTO_LIMITS<CR>
TRIGRID> SHOW ENABLE<CR>
AUTO_LIMITS ..................................... On
DIAGNOSTICS ..................................... Off
GRAPHICS ........................................ Off
SMOOTH .......................................... On
(Smooth patch interpolation)
NINT ............................................ On
ORTHOGONAL ...................................... Off
TRACE ........................................... Off
TRIGRID>
```

DISABLE DEBUG

Disables a previous ENABLE DEBUG command.

FORMAT: DISABLE DEBUG

Command parameters: None.

DESCRIPTION:

DISABLE DEBUG allows the user to cancel the effect of a previous ENABLE DEBUG command.

When the DEBUG option is active, TRIGRID produces copious debugging information. The DEBUG option is reserved for use by Laser-Scan personnel.

By default, the DEBUG option is disabled during program startup.

Messages: None.

Examples:

TRIGRID> **ENABLE DEBUG<CR>**
TRIGRID>

DISABLE DIAGNOSTICS

Disables a previous ENABLE DIAGNOSTICS command.

FORMAT: DISABLE DIAGNOSTICS

Command parameters: None.

DESCRIPTION:

DISABLE DIAGNOSTICS allows the user to cancel the effect of a previous ENABLE DIAGNOSTICS command.

Messages: None.

Examples:

```
TRIGRID> ENABLE DIAGNOSTICS<CR>
TRIGRID> SHOW ENABLE<CR>
AUTO_LIMITS ..................................... On
DIAGNOSTICS ..................................... On
GRAPHICS ........................................ Off
SMOOTH .......................................... On
(Smooth patch interpolation)
NINT ............................................ On
ORTHOGONAL ...................................... Off
TRACE ........................................... Off
TRIGRID>
```

DISABLE GRAPHICS

Disables any previous ENABLE GRAPHICS command.

FORMAT: DISABLE GRAPHICS

Command parameter: None.

DESCRIPTION:

The DISABLE GRAPHICS command cancels the effect of a previous ENABLE GRAPHICS command.

Messages: None.

Examples:

```
$ TRIGRID<CR>
DTMCREATE module TRIGRID of 13:30:39  20-AUG-87
TRIGRID> SHOW ENABLE<CR>
AUTO_LIMITS ..................................... On
DIAGNOSTICS ..................................... Off
GRAPHICS ........................................ Off
SMOOTH .......................................... On
(Smooth patch interpolation)
NINT ............................................ On
ORTHOGONAL ...................................... Off
TRACE ........................................... Off
TRIGRID> ENABLE GRAPHICS<CR>
TRIGRID> SHOW ENABLE<CR>
AUTO_LIMITS ..................................... On
DIAGNOSTICS ..................................... Off
GRAPHICS ........................................ On
SMOOTH .......................................... On
(Smooth patch interpolation)
NINT ............................................ On
ORTHOGONAL ...................................... Off
TRACE ........................................... Off
TRIGRID> DISABLE GRAPHICS<CR>
TRIGRID> SHOW ENABLE<CR>
AUTO_LIMITS ..................................... On
DIAGNOSTICS ..................................... Off
GRAPHICS ........................................ Off
SMOOTH .......................................... On
(Smooth patch interpolation)
NINT ............................................ On
ORTHOGONAL ...................................... Off
TRACE ........................................... Off
TRIGRID>
```

DISABLE NINT

Disables a previous ENABLE NINT command.

FORMAT: DISABLE NINT

Command parameters: None.

DESCRIPTION:

DISABLE NINT allows the user to cancel the effect of a previous ENABLE NINT command.

When output is directed to a word or longword DTI file it is necessary to convert the real height value interpolated within TRIGRID to an integer value. This conversion is normally done using the FORTRAN NINT intrinsic function (round to nearest integer). However, some customers prefer to use integer truncation (FORTRAN INT intrinsic function).

The ENABLE NINT and DISABLE NINT commands allow the user to specify which of these two functions is to be used, i.e.

ENABLE NINT - use NINT, round to nearest integer value (Default)
DISABLE NINT - use INT, truncate to integer value

By default, the NINT option is enabled during program startup.

Messages: None.

Examples:

```
TRIGRID> ENABLE NINT<CR>
TRIGRID> SHOW ENABLE<CR>
AUTO_LIMITS ..................................... On
DIAGNOSTICS ..................................... Off
GRAPHICS ........................................ Off
SMOOTH .......................................... On
(Smooth patch interpolation)
NINT ............................................ On
ORTHOGONAL ...................................... Off
TRACE ........................................... Off
TRIGRID>
```

DISABLE ORTHOGONAL

Disables a previous ENABLE ORTHOGONAL command.

FORMAT: DISABLE ORTHOGONAL

Command parameters: None.

DESCRIPTION:

DISABLE ORTHOGONAL allows the user to cancel the effect of a previous ENABLE ORTHOGONAL command.

When the ORTHOGONAL option is active, individual triangle interpolation under-shoot and over-shoot limits are applied orthogonal to the triangle facet plane. The default action is to apply the limits vertically regardless of the slope of the triangle facet plane. ENABLE ORTHOGONAL is not recommended for use with data sets which have regularly spaced contours of regular interval but will be beneficial if the contour data have an irregular interval and have irregular spacing.

By default, the ORTHOGONAL option is disabled during program startup.

Messages: None.

Examples:

```
TRIGRID> ENABLE ORTHOGONAL<CR>
TRIGRID> SHOW ENABLE<CR>
AUTO_LIMITS ..................................... On
DIAGNOSTICS ..................................... Off
GRAPHICS ........................................ Off
SMOOTH .......................................... On
(Smooth patch interpolation)
NINT ............................................ On
ORTHOGONAL ...................................... Off
TRACE ........................................... Off
TRIGRID>
```

DISABLE PME

DISABLE PME disables the effect of a previous ENABLE PME command.

FORMAT: DISABLE PME

Command parameters: None.

DESCRIPTION:

The ENABLE PME and DISABLE PME commands are reserved for Laser-Scan use. PME is a code optimisation tool and should be invoked by LSL software personnel only.

DISABLE PME disables the effect of a previous ENABLE PME command and causes the PME_EXIT routine to be invoked.

Message:

The following warning message is specific to the DISABLE PME command:

*** WARNING *** You were not using PME anyway!

Examples:

$ **TRIGRID<CR>**
DTMCREATE module TRIGRID of 13:30:39 20-AUG-87
TRIGRID> **ENABLE PME ! turn PME on<CR>**
TRIGRID> **DISABLE PME ! turn it off again!!<CR>**
TRIGRID>

DISABLE SMOOTH

Disables a previous ENABLE SMOOTH command.

FORMAT: DISABLE SMOOTH

Command parameters: None.

DESCRIPTION:

DISABLE SMOOTH allows the user to cancel the effect of a previous ENABLE SMOOTH command.

By default, the SMOOTH option is enabled during program startup.

The DISABLE SMOOTH command disables grid point estimation by smooth patches. Grid heights are instead estimated using linear interpolation across the triangle facet planes. Smooth patch estimation tends to give a more rounded smooth appearance to a DTM surface than that given by linear facets.

If smooth patch interpolation is selected, individual triangle interpolation limits are applied. Linear facet interpolation, by definition, does not need to apply interpolation limits as interpolation is restricted to the triangle facet plane.

For further details of interpolation limits, see the LIMITS, TRIANGLE_LIMITS and ENABLE AUTO_LIMITS commands.

Messages:

The following messages are specific to the DISABLE SMOOTH command:

```
*** WARNING *** Linear facet interpolation selected. No smooth patch
                interpolation limits will be applied
                and the ENABLE TRACE command will be ignored

*** WARNING *** Linear facet interpolation selected.
                The ENABLE TRACE command and any smooth patch
                interpolation will be ignored
```

Examples:

```
TRIGRID> DISABLE SMOOTH<CR>
*** WARNING *** Linear facet interpolation selected.
                The ENABLE TRACE command and any smooth patch
                interpolation will be ignored

TRIGRID>
```

DISABLE TRACE

Disables a previous ENABLE TRACE command.

FORMAT: DISABLE TRACE

Command parameters: None.

DESCRIPTION:

DISABLE TRACE allows the user to cancel the effect of a previous ENABLE TRACE command.

The TRACE option causes TRIGRID to calculate up-hill/down-hill side of line context for the original data strings input to (and optionally constrained) in TRIANG. These up-hill/down-hill data are then used to apply further interpolation limits to each individual triangle. These additional limits ensure that triangles whose vertices indicate a flat plane do bulge in the correct direction relative to surrounding hill slopes even when the slope derivatives estimated for the triangle vertices conflict due to complex surrounding relief.

It is strongly recommended that the trace option is selected, particularly if the resulting DTM is to be re-contoured.

By default, the TRACE option is disabled during program startup.

Messages: None.

Examples:

```
TRIGRID> ENABLE TRACE<CR>
TRIGRID> SHOW ENABLE<CR>
AUTO_LIMITS ..................................... On
DIAGNOSTICS ..................................... Off
GRAPHICS ........................................ Off
SMOOTH .......................................... On
(Smooth patch interpolation)
NINT ............................................ On
ORTHOGONAL ...................................... Off
TRACE ........................................... Off
TRIGRID>
```

ENABLE AUTO_LIMITS

Enables the application of automatic limits to grid interpolation.

FORMAT: ENABLE AUTO_LIMITS

Command parameters: None.

DESCRIPTION:

ENABLE AUTO_LIMITS enables the user to activate the application of automatic limits to grid interpolation.

When the AUTO_LIMITS option is active, TRIGRID applies interpolation limits based on the feature flag information attached to each node in the triangulation. It is then impossible, for example, for a grid height to be set below the height of a triangle vertex that is flagged as being part of a river string.

By default, the AUTO_LIMITS option is enabled during program startup.

The AUTO_LIMITS option may be cancelled by the DISABLE AUTO_LIMITS command.

Messages: None.

Examples:

TRIGRID> **ENABLE AUTO_LIMITS<CR>**
TRIGRID>

ENABLE DEBUG

Enables output of debug information. The ENABLE DEBUG command is reserved
for use by Laser-Scan personnel.

FORMAT: ENABLE DEBUG

Command parameters: None.

DESCRIPTION:

When the DEBUG option is active, TRIGRID produces copious debugging information. The DEBUG option is reserved for use by Laser-Scan personnel.

By default, the DEBUG option is disabled during program startup.

Messages: None.

Examples:

TRIGRID> **ENABLE DEBUG<CR>**
TRIGRID>

ENABLE DIAGNOSTICS

ENABLE DIAGNOSTICS allows the user to enable diagnostic printout.

FORMAT: ENABLE DIAGNOSTICS

Command parameters: None.

DESCRIPTION:

ENABLE DIAGNOSTICS allows the user to enable diagnostic printout.

Because it is usually used in a batch processing environment, by default TRIGRID produces minimal diagnostic printout. If however, the user wishes to receive indications of processing progress and of the effect of selections on data input, diagnostic printout may be selected with the ENABLE DIAGNOSTICS command.

On a heavily loaded computer it may be reassuring to ENABLE DIAGNOSTICS to indicate progress through the data set. If SYS$OUTPUT is directed to a video screen terminal, messages indicating percentage progress are issued.

Messages: None.

Examples:

```
TRIGRID> ENABLE DIAGNOSTICS<CR>
TRIGRID> SHOW ENABLE<CR>
AUTO_LIMITS ..................................... On
DIAGNOSTICS ..................................... On
GRAPHICS ........................................ Off
SMOOTH .......................................... On
(Smooth patch interpolation)
NINT ............................................ On
ORTHOGONAL ...................................... Off
TRACE ........................................... Off
TRIGRID>
```

ENABLE GRAPHICS

Enable TRIGRID graphics output.

FORMAT: ENABLE GRAPHICS

Command parameters: None.

DESCRIPTION:

TRIGRID offers the option to generate graphic output to indicate processing progress. By default graphic output is disabled. To prevent a user selecting graphics when it is inappropriate to the current terminal, TRIGRID uses a lookup table of terminal characteristics associated with all available terminal lines (see Appendix 1). An invalid graphics selection will result in a warning message and the GRAPHICS option being deselected.

Graphics selection may be cancelled with the DISABLE GRAPHICS command.

Messages:

The following messages are specific to the ENABLE GRAPHICS command:

*** ERROR *** reading lookup file at line 'integer'

*** WARNING *** Unable to open "LSL$LOOKUP:TERMTYPE.DAT"

Sorry 'name' terminal 'terminal-ident' isn't in the lookup table

Sorry 'name' terminal 'terminal-ident' can't support graphics

*** ERROR *** translating logical name LSL$DTMCREATETERMINAL

Examples:

$ **TRIGRID<CR>**
DTMCREATE module TRIGRID of 13:30:39 20-AUG-87
TRIGRID> **ENABLE GRAPHICS<CR>**
TRIGRID>

ENABLE NINT

Enable rounding of grid z-values to the nearest integer, i.e. used for word and longword DTI data types.

FORMAT: ENABLE NINT

Command parameters: None.

DESCRIPTION:

When output is directed to a word or longword DTI file it is necessary to convert the real height value interpolated within TRIGRID to an integer value. This conversion is normally done using the FORTRAN NINT intrinsic function (round to nearest integer). However, some customers prefer to use integer truncation (FORTRAN INT intrinsic function).

The ENABLE NINT and DISABLE NINT commands allow the user to specify which of these two functions is to be used, i.e.

ENABLE NINT - use NINT, round to nearest integer value (Default)
DISABLE NINT - use INT, truncate to integer value

By default, the NINT option is enabled during program startup.

Messages: None.

Examples:

```
TRIGRID> ENABLE NINT<CR>
TRIGRID> SHOW ENABLE<CR>
AUTO_LIMITS ..................................... On
DIAGNOSTICS ..................................... Off
GRAPHICS ........................................ Off
SMOOTH .......................................... On
(Smooth patch interpolation)
NINT ............................................ On
ORTHOGONAL ...................................... Off
TRACE ........................................... Off
TRIGRID>
```

ENABLE ORTHOGONAL

Causes individual triangle interpolation under-shoot and over-shoot limits to be applied orthogonal to the triangle facet plane.

FORMAT: ENABLE ORTHOGONAL

Command parameters: None.

DESCRIPTION:

When the ORTHOGONAL option is active, individual triangle interpolation under-shoot and over-shoot limits are applied orthogonal to the triangle facet plane. The default action is to apply the limits vertically regardless of the slope of the triangle facet plane. ENABLE ORTHOGONAL is not recommended for use with data sets which have regularly spaced contours of regular interval but will be beneficial if the contour data have an irregular interval and have irregular spacing.

The effect of an ENABLE ORTHOGONAL command may be cancelled with a DISABLE ORTHOGONAL command.

By default, the ORTHOGONAL option is disabled during program startup.

Messages: None.

Examples:

```
TRIGRID> ENABLE ORTHOGONAL<CR>
TRIGRID> SHOW ENABLE<CR>
AUTO_LIMITS ..................................... On
DIAGNOSTICS ..................................... Off
GRAPHICS ........................................ Off
SMOOTH .......................................... On
(Smooth patch interpolation)
NINT ............................................ On
ORTHOGONAL ...................................... On
TRACE ........................................... Off
TRIGRID>
```

ENABLE PME

ENABLE PME enables the PME performance monitor.

FORMAT: ENABLE PME

Command parameters: None.

DESCRIPTION:

The ENABLE PME and DISABLE PME commands are reserved for Laser-Scan use. PME is a code optimisation tool and should be invoked by LSL software personnel only.

ENABLE PME causes the PME_INIT routine to be invoked.

Message:

The following warning message is specific to the ENABLE PME command:

*** WARNING *** You are already using PME!

Examples:

```
$ TRIGRID<CR>
DTMCREATE module TRIGRID of 13:30:39 20-AUG-87
TRIGRID> ENABLE PME<CR>
TRIGRID> FILEIN DUA3:[DEMONSTRATION]IDAHO<CR>
.DTA file DUA3:[DEMONSTRATION]IDAHO.DTA;15 opened for read
.NOD file DUA3:[DEMONSTRATION]IDAHO.NOD;15 opened for read
.DER file DUA3:[DEMONSTRATION]IDAHO.DER;15 opened for read
TRIGRID>
```

ENABLE SMOOTH

Enables grid point estimation by smooth patches.

FORMAT: ENABLE SMOOTH

Command parameters: None.

DESCRIPTION:

The ENABLE SMOOTH command enables grid point estimation by smooth patches. Smooth patch estimation tends to give a more rounded smooth appearance to a DTM surface than that given by linear facets.

By default, the SMOOTH option is enabled during program startup.

If smooth patch interpolation is selected, individual triangle interpolation limits are applied. Linear facet interpolation, by definition, does not need to apply interpolation limits as interpolation is restricted to the triangle facet plane.

The DISABLE SMOOTH command disables grid point estimation by smooth patches. Grid heights are instead estimated using linear interpolation across the triangle facet planes.

For further details of interpolation limits, see the LIMITS, TRIANGLE_LIMITS and ENABLE AUTO_LIMITS commands.

Messages: None.

Examples:

TRIGRID> **ENABLE SMOOTH<CR>**
TRIGRID>

ENABLE TRACE

Causes calculation of up-hill/down-hill side of line context for the original data strings input to (and optionally constrained) in TRIANG or modified in TRIEDIT.

FORMAT: ENABLE TRACE

Command parameters: None.

DESCRIPTION:

The TRACE option causes TRIGRID to calculate up-hill/down-hill side of line context for the original data strings input to (and optionally constrained) in TRIANG. These up-hill/down-hill data are then used to apply further interpolation limits to each individual triangle. These additional limits ensure that triangles whose vertices indicate a flat plane do bulge in the correct direction relative to surrounding hill slopes even when the slope derivatives estimated for the triangle vertices conflict due to complex surrounding relief.

It is strongly recommended that the trace option is selected, particularly if the resulting DTM is to be re-contoured.

By default, the TRACE option is disabled during program startup.

Messages: None.

Examples:

TRIGRID> **ENABLE TRACE<CR>**
TRIGRID>

FILEIN

Specifies the generic (.DTA, .NOD and .DER) file-spec that is to be used for input.

FORMAT: FILEIN file-spec

COMMAND PARAMETERS:

file-spec

The generic specification of the .NOD .DTA and .DER files to be opened for data input.

All components of the supplied file-spec are used to form the input file specifications but with the substitution of the extensions .NOD, .DTA and .DER and version number ';0', i.e. latest version (shared by all three files).

The default file-spec used to make up missing parts of the FILEIN file-spec parameter is dependent on the status of logical name LSL$DTMCREATE_WORK.

If logical name LSL$DTMCREATE_WORK is defined, DTMCREATE utilities translate the logical name to get the default file-spec for input and output of triangulation files. The logical name should be defined to provide a device and directory name only. The DTMCREATE programs themselves provide the default filename and extension fields of the specification. For example, a valid definition of logical name LSL$DTMCREATE_WORK is:

$ DEFINE LSL$DTMCREATE_WORK LSL$DATA_ROOT:[LSL.DTMCREATE]

This mechanism allows all DTMCREATE triangulation files to be stored in a central directory, rather than scattered in many different user directories. It thus mimics the use of logical names LSL$IF for IFF files and LSL$DTI for DTI files.

If the logical name is not defined, any parts of the file-spec not supplied for the FILEIN command will be taken from the defaults 'SYS$DISK:[].NOD;0' 'SYS$DISK:[].DTA;0' and 'SYS$DISK:[].DER;0'. These defaults result in the files being searched for in your current default directory, set using the VMS SET DEFAULT or Laser-Scan SD commands.

DESCRIPTION:

TRIGRID expects as input the 2 binary structured data files (the matched pair of .NOD and .DTA files) produced by TRIANG or TRIEDIT, and the .DER file containing the slope derivatives estimated by TRIDER.

The TRIGRID input file specification is used as a generic file-spec for all three input (.NOD, .DTA and .DER) files.

Any parts missing from the generic file-specification are taken from the from the defaults SYS$DISK:[].DTA;0 and SYS$DISK:[].NOD;0.

Since it is essential that the file version numbers of the .NOD, .DTA and .DER file always match, TRIGRID performs checks on file version numbers. If mismatches are found, TRIGRID complains and aborts execution.

Messages:

The following messages are specific to the FILEIN command:

```
*** WARNING *** You must specify a file-spec argument to the FILEIN command
                For example FILEIN SWAREA.DTA

*** ERROR *** Unable to interpret input file-spec

*** ERROR *** opening input file

.DTA file 'file-spec' opened for read
.NOD file 'file-spec' opened for read
.DER file 'file-spec' opened for read
```

Examples:

```
TRIGRID> FILEIN DUA3:[DEMONSTRATION]IDAHO<CR>
.DTA file DUA3:[DEMONSTRATION]IDAHO.DTA;15 opened for read
.NOD file DUA3:[DEMONSTRATION]IDAHO.NOD;15 opened for read
.DER file DUA3:[DEMONSTRATION]IDAHO.DER;15 opened for read
TRIGRID>
```

FILEOUT

Specifies the name of the DTI file to be created and opened for output.

FORMAT: FILEOUT file-spec

COMMAND PARAMETERS:

file-spec

The specification of the DTI file to be created and opened for output.

Any parts missing from the file-specification are taken from the default LSL$DTI:DTI.DTI;0.

DESCRIPTION:

The FILEOUT command is used to specify the DTI file which is to be created and opened for data output.

The DTI file is not actually created and opened until a GO command is accepted, as the user may redefine the DTI header type (HEADER_TYPE command) or the DTI data type (DATA_TYPE command) after issuing the FILEOUT command.

By default TRIGRID creates a DTI file having an LSLA type header and word data type.

The characteristics of the output DTI file are determined by the HEADER_TYPE and DATA_TYPE commands, and its size by combinations of WINDOW, SIZE or SIDELENGTH commands.

If a set type 2 IFF MD (Map Descriptor) or a set DTI file projection record were available in the first file read into TRIANG using a FILEIN command, TRIANG puts this information into the .NOD and .DTA files. If TRIGRID is creating a DTI file with an LSLA type header this projection information is copied from the .DTA amd .NOD files into the TRIGRID DTI file header after appropriate modifications of extent, origin and gridstep.

In contrast, it is left to the user to set DTED DTI file header contents using DTEDIFF and MCEHED.

Messages:

The following messages are specific to the FILEOUT command:

*** WARNING *** You must specify a file-spec argument to the FILEOUT command
For example FILEOUT SWAREA.DTI

*** ERROR *** Unable to interpret input file-spec

Examples:

TRIGRID> **FILEOUT IDAHO<CR>**
TRIGRID>

GO

Initiates the processing of the data read in using the FILEIN command.

FORMAT: GO

Command parameters: None.

DESCRIPTION:

When all necessary files have been read in the GO command will cause the DTI file specified with the FILEOUT command to be created and opened. TRIGRID will then commence the gridding process. Unless relatively small data-sets are being handled (say less then 50,000 data points) it is strongly recommended that TRIGRID is run in batch mode at an off-peak time.

When grid creation is complete, TRIGRID closes the output DTI file and then exits.

Before the GO command will be accepted, TRIGRID performs checks that the following obligatory commands have been issued:

1) WINDOW
2) SIZE (or SIDELENGTH)
3) FILEIN
4) FILEOUT

If one (or more) of these commands has not yet been issued and accepted, TRIGRID will ignore the GO command and prompt for further commands at the terminal. The SHOW ENABLE command can be used to discover the current command option settings.

Messages:

*** WARNING *** The following commands must be given
before processing can begin:
1) WINDOW
2) SIZE (or SIDELENGTH)
3) FILEIN
4) FILEOUT

Once the GO command has been accepted no more conversational messages of the *** WARNING *** format will be issued. Any messages will relate to serious processing problems and will normally result in abnormal TRIGRID termination. The messages relating to non-interactive processing problems are presented at the end of this document.

Examples:

TRIGRID> **GO<CR>**

```
DTI file LSL$DTI:TESTDTM.DTI opened for write

File    :  LSL$DTI:TESTDTM.DTI
Header  :  LSLA  Data:  WORD

Units are Metres

Matrix Coverage   SW:       0.00       0.00    NE:     200.00     199.50
Matrix Interval    E:       2.00                N:       1.50
Value Range         :     0   to      0



 ELAPSED:    0 00:05:21.82  CPU: 0:00:01.40  BUFIO: 51  DIRIO: 15  FAULTS: 170
$
```

HEADER_TYPE

The HEADER_TYPE command specifies the header type of the output DTI file.

FORMAT: HEADER_TYPE type

Command parameters:

type

A DTI header type keyword chosen from:

- TED4 - generate a TED4 type DTI
- UHL1 - generate a UHL1 type DTI
- LSLA - generate a LSLA type DTI

DESCRIPTION:

The HEADER_TYPE allows the user to specify the header type of the output DTI file which contains the DTM grid.

By default TRIGRID generates a 'LSLA' type DTI file having WORD (16 bit integer) post values.

For a detailed description of the layout and content of DTI file header types see the DTILIB section of the MATRIX Reference Manual.

The default LSLA type DTI header contains only the matrix X and Y extent, Z range and the X and Y grid interval.

Due to the general purpose nature of TRIGRID it is not possible to collect from the user, and validate, all the complex information required to complete a military TED4 or UHL1 DTI header. Instead TRIGRID puts into a TED4 or UHL1 header the same information as for a LSLA type DTI. If output is to a TED4 type DTI file, the Latitude and Longitude origin and the Latitude and Longitude bounding rectangle values are inserted into the DSI record. These values are taken from the WINDOW command arguments. It is assumed that the triangulation units are tenths of seconds of arc.

The user is left to use the MATRIX package utility DTIPATCH to set up missing information.

Messages: None.

Examples:

```
TRIGRID> HEADER_TYPE TED4<CR>
TRIGRID>
```

HELP

Give help on a subject

FORMAT: HELP subject

Command parameters:

subject

The subject on which help is required

Description:

The HELP command looks the rest of the line up in the DTMCREATE HELP library. This library contains a brief summary of the operation of each command.

The information is looked up in the TRIGRID section of the DTMCREATE help library, LSL$HELP:DTMCREATE.HLB.

Messages:

Where required, warning messages are output via the VMS LBR$OUTPUT_HELP utility.

Examples:

TRIGRID> **HELP RETURN<CR>**

RETURN

Restores command input from an indirect file to SYS$COMMAND.

A typical application is to allow the user to use an indirect command file to set up those run time defaults which are constant within a flowline and then return to input from the terminal (or batch stream) for the run specific commands. To do this RETURN must be the last command in the indirect command file.

TRIGRID>

PAUSE

Pauses TRIGRID execution.

FORMAT: PAUSE

Command parameters: None.

DESCRIPTION:

Pauses TRIGRID execution and issues a prompt for a carriage return to continue execution. This command is designed for use in software demonstration situations.

Messages: None.

Examples:

TRIGRID> **PAUSE<CR>**

Press <RETURN> to continue**<CR>**
TRIGRID>

QUIT

Quit from TRIGRID.

FORMAT: QUIT

Command parameters: None.

Description:

The QUIT command causes TRIGRID to exit immediately, closing all input files and closing and deleting all output files.

<CTRL/Z> (pressing the Ctrl and Z keys together) may also be used to quit from the program.

Messages: None.

Examples:

```
TRIGRID> QUIT<CR>
 ELAPSED:  00:05:25.84 CPU:  0:00:05.71 BUFIO:  281 DIRIO:  46 FAULTS:  263
$
```

RETURN

Restores command input from an indirect file to SYS$COMMAND.

FORMAT: RETURN

Command parameters: None.

DESCRIPTION:

Restores command input from an indirect file to SYS$COMMAND.

A typical application is to allow the user to use an indirect command file to set up those run time defaults which are constant within a flowline and then return to input from the terminal (or batch stream) for the run specific commands. To do this RETURN must be the last command in the indirect command file.

Messages:

The following messages are specific to the RETURN command:

RETURN command detected - returning to terminal input

RETURN command ignored - command input is already from terminal

---

Examples:

```
TRIGRID> @FLOW2<CR>
TRIGRID> ENABLE DIAGNOSTICS
TRIGRID> RETURN
TRIGRID>
```

SHOW

Shows current status of TRIGRID option and parameter settings.

FORMAT: SHOW subject

Command parameters:

subject

The subject that is to be displayed, chosen from:

DATA_TYPE ENABLE FILES HEADER_TYPE LIMITS SIDELENGTH

SIZE UNITS WINDOW

DESCRIPTION:

SHOW command enables the user to examine the current status of TRIGRID options and parameter settings.

Messages:

TRIGRID issues the following message if the SHOW command is specified without an argument:

*** ERROR *** Specifying command SHOW

Available SHOW command qualifiers are:

DATA_TYPE ENABLE FILES HEADER_TYPE LIMITS SIDELENGTH

SIZE UNITS WINDOW

This feature can be used to advantage if the user wishes to quickly determine for which items the SHOW facility is available.

Examples:

$ **TRIGRID<CR>**
DTMCREATE module TRIGRID of 14:02:31 3-FEB-89
TRIGRID> **SHOW<CR>**

*** ERROR *** Specifying command SHOW

Available SHOW command qualifiers are:

```
DATA_TYPE    ENABLE       FILES    HEADER_TYPE     LIMITS     SIDELENGTH

SIZE         UNITS        WINDOW

TRIGRID> SHOW ENABLE ! examine current option settings<CR>
AUTO_LIMITS ..................................... On
DIAGNOSTICS ..................................... Off
GRAPHICS ........................................ Off
SMOOTH .......................................... On
(Smooth patch interpolation)
NINT ............................................ On
ORTHOGONAL ...................................... Off
TRACE ........................................... Off
TRIGRID>
```

SHOW DATA_TYPE

SHOW DATA_TYPE enables the user to examine the current setting for the DATA_TYPE command.

FORMAT: SHOW DATA_TYPE

Command parameters: None.

DESCRIPTION:

SHOW DATA_TYPE enables the user to examine the current setting for the DATA_TYPE command. This determines the data type of the output DTI file.

Messages: None.

Examples:

```
$ TRIGRID<CR>
DTMCREATE module TRIGRID of 17:02:12 23-JAN-89
TRIGRID> SHOW DATA_TYPE ! examine current data_type settings<CR>
DTI data type ..................................... WORD
TRIGRID>
```

SHOW ENABLE

Shows current status of TRIGRID option settings.

FORMAT: SHOW ENABLE

Command parameters: None.

DESCRIPTION:

SHOW ENABLE enables the user to examine the current status of TRIGRID processing options that are set or unset using the ENABLE and DISABLE commands.

Messages: None.

Examples:

$ **TRIGRID<CR>**
DTMCREATE module TRIGRID of 17:02:12 23-JAN-89
TRIGRID> **SHOW ENABLE ! examine current option settings<CR>**
TRIGRID>

SHOW FILES

Shows current TRIGRID input and output files.

FORMAT: SHOW FILES

Command parameters: None.

DESCRIPTION:

SHOW FILES enables the user to examine the current status of TRIGRID input files.

Messages: None.

Examples:

$ **TRIGRID<CR>**
DTMCREATE module TRIGRID of 17:02:12 23-JAN-89
TRIGRID> **SHOW FILES ! examine files<CR>**

Input files:
The input files are undefined

Output file:
The DTI output file is undefined

TRIGRID> **FILEOUT EXAMPLE ! open output DTI file<CR>**
TRIGRID> **SHOW FILES ! examine input file-specs<CR>**

Input files:
The input files are undefined

Output file:
LSL$DATAROOT:[LSL.DTI]EXAMPLE.DTI;0
(DTI file window or size are currently undefined)

TRIGRID> **FILEIN SW100230 ! get input files<CR>**
.DTA file DUA3:[DEMONSTRATION]SW100230.DTA;6 opened for read
.NOD file DUA3:[DEMONSTRATION]SW100230.NOD;6 opened for read
.DER file DUA3:[DEMONSTRATION]SW100230.DER;6 opened for read
TRIGRID> **SHOW FILES ! examine input file-specs<CR>**

Input files:
.DTA file: DUA3:[DEMONSTRATION]SW100230.DTA;6
.NOD file: DUA3:[DEMONSTRATION]SW100230.NOD;6
.DER file: DUA3:[DEMONSTRATION]SW100230.DER;6

Output file:
LSL$DATAROOT:[LSL.DTI]EXAMPLE.DTI;0

```
            (DTI file window or size are currently undefined)

TRIGRID>
```

SHOW HEADER_TYPE

SHOW HEADER_TYPE enables the user to examine the current setting for the HEADER_TYPE command.

FORMAT: SHOW HEADER_TYPE

Command parameters: None.

DESCRIPTION:

SHOW HEADER_TYPE enables the user to examine the current setting for the HEADER_TYPE command. This determines the header type of the output DTI file.

Messages: None.

Examples:

```
$ TRIGRID<CR>
DTMCREATE module TRIGRID of 17:02:12 23-JAN-89
TRIGRID> SHOW HEADER_TYPE ! examine current data_type settings<CR>
DTI header type ..................................... LSLA
TRIGRID>
```

SHOW LIMITS

Shows current status of TRIGRID Z-limits.

FORMAT: SHOW LIMITS

Command parameters: None.

DESCRIPTION:

SHOW LIMITS enables the user to examine the current status of TRIGRID Z-limits parameters set using the LIMITS commands.

Messages: None.

Examples:

$ **TRIGRID<CR>**
DTMCREATE module TRIGRID of 17:02:12 23-JAN-89
TRIGRID> **SHOW LIMITS ! examine LIMITS settings<CR>**
No individual triangle interpolation limits will be applied
No whole DTM Z-limits will be applied
TRIGRID>

SHOW SIDELENGTH

Shows current status of TRIGRID DTI file sidelength.

FORMAT: SHOW SIDELENGTH

Command parameters: None.

DESCRIPTION:

SHOW SIDELENGTH enables the user to examine the current status of TRIGRID DTI file sidelength parameters set using the SIDELENGTH or the combination of SIZE and WINDOW commands.

Messages: None.

Examples:

```
$ TRIGRID<CR>
DTMCREATE module TRIGRID of 17:02:12 23-JAN-89
TRIGRID> SHOW SIDELENGTH ! examine sidelength settings<CR>
X-sidelength ..................................... Unset
Y-sidelength ..................................... Unset
TRIGRID> SIDELENGTH 4.0 4.0 ! set x and y sidelength <CR>
TRIGRID> SHOW SIDELENGTH ! examine sidelength settings<CR>
X-sidelength ..................................... 4.0
Y-sidelength ..................................... 4.0
TRIGRID>
```

SHOW SIZE

Shows current status of TRIGRID output DTI file SIZE.

FORMAT: SHOW SIZE

Command parameters: None.

DESCRIPTION:

SHOW SIZE enables the user to examine the current status of TRIGRID output DTI file SIZE parameters set using the SIZE or the combination of SIDELENGTH and WINDOW commands.

Messages: None.

Examples:

```
$ TRIGRID<CR>
DTMCREATE module TRIGRID of 17:02:12 23-JAN-89
TRIGRID> SHOW SIZE ! examine size settings<CR>
Number of columns ..................................... Unset
Number of rows ........................................ Unset
TRIGRID> SIZE 801 401 ! set x and y sidelength <CR>
TRIGRID> SHOW SIZE ! examine sidelength settings<CR>
Number of columns ..................................... 801
Number of rows ........................................ 401
TRIGRID>
```

SHOW UNITS

Shows current status of TRIGRID window units as set using the UNITS command.

FORMAT: SHOW UNITS

Command parameters: None.

DESCRIPTION:

Shows current status of TRIGRID window units as set using the UNITS command.

Messages: None.

Examples:

$ **TRIGRID<CR>**
DTMCREATE module TRIGRID of 13:53:27 3-FEB-89
TRIGRID> **SHOW UNITS<CR>**
Window to be specified in metres
TRIGRID>

SHOW WINDOW

Shows current triangulation and DTM window values.

FORMAT: SHOW WINDOW

Command parameters: None.

DESCRIPTION:

Shows current triangulation and DTM window values.

The units of the WINDOW command parameters are set using the UNITS command. By default metre units are assumed. If it is more convenient to specify the window in latitude and longitude the sexagesimal lat long values may be supplied after specifying a UNITS LATLONG command. This assumes that the data read in from the DTI and IFF files uaing TRIANG were in units of tenths second of arc. A similar assumption is made for UNITS SECONDS.

The SHOW WINDOW command displays the window values in the units currently selected by the UNITS command. No projection transformation is performed, so the unwise user could easily specify that the window be shown as latitude longitude despite the fact that his data are in metres!

Messages: None.

Examples:

$ **TRIGRID<CR>**
DTMCREATE module TRIGRID of 13:53:27 3-FEB-89
TRIGRID> **SHOW WINDOW<CR>**
Window units are metres
DTM WINDOW currently undefined
Triangulation coverage currently unknown
TRIGRID>

SIDELENGTH

Specifies the spacing between the columns and rows of the output DTM.

FORMAT: SIDELENGTH x-sidelength y-sidelength

Command parameters:

x-sidelength

The spacing between DTM posts along the X axis (columns).

y-sidelength

The spacing between DTM posts up the Y axis (rows). If this parameter is omitted TRIGRID assumes equal X and Y sidelength values.

DESCRIPTION:

The SIDELENGTH command enables the user to specify the spacing between the columns and rows of the output DTM. The SIDELENGTH command arguments should be specified in the units used for the triangulation.

If a SIZE command has already been issued then the SIZE values will be overridden by new size values calculated when WINDOW is set using:

NUMBER OF COLUMNS = X WINDOW LENGTH/ X_SIDELENGTH +1.5

NUMBER OF ROWS = Y WINDOW LENGTH/ Y_SIDELENGTH +1.5

and the message:

Over-riding SIZE settings

will appear.

If the WINDOW is set, the number of rows and columns to cover completely (possibly with some overlap) the specified area is calculated. The results of this calculation will be displayed:

Window units are metres

Triangulation coverage SW: 'real' 'real' NE: 'real' 'real'
Triangulation window SW: 'real' 'real' NE: 'real' 'real'

Which with a side length of 10.00 in X and 10.00 in Y
gives 'integer' rows and 'integer' columns in the dtm

You may use the SIDELENGTH and WINDOW commands in combination as often as you wish until the desired model characteristics are achieved.

Messages:

The following messages are specific to the SIDELENGTH command:

*** WARNING *** You must have at least 3 rows and 3 columns in your DTM

*** WARNING *** You must specify at least one argument

Examples:

TRIGRID> **SIDELENGTH 40.0 50.0<CR>**
TRIGRID>

SIZE

Specifies the number of columns and rows of the output DTM.

FORMAT: SIZE ncols nrows

Command parameters:

ncols

The number of columns (X-axis) in the output DTM.

nrows

The number of rows (Y-axis) in the output DTM. If this parameter is omitted TRIGRID assumes equal X and Y SIZE values.

DESCRIPTION:

The SIZE command enables the user to explicitly set the number of columns and rows in the output DTM.

If you have already issued a SIDELENGTH command then the sidelength values that you specified will be replaced by the values calculated from WINDOW divided by the number of rows and number of columns and the message:

Over-riding SIDELENGTH settings

will appear.

If you have already specified values for the WINDOW then TRIGRID will calculate the cell sidelength required to give you the the number of rows and columns specified by the SIZE command and the results of this calculation will be displayed:

```
Window units are metres

Triangulation coverage SW:  'real' 'real' NE:  'real' 'real'
Triangulation window   SW:  'real' 'real' NE:  'real' 'real'

Which with a side length of    10.00 in X and     10.00 in Y
gives 'integer' rows and 'integer' columns in the dtm
```

You may use the SIZE and WINDOW commands in combination as often as you wish until the desired model characteristics are achieved.

Messages:

The following messages are specific to the SIZE command:

*** WARNING *** You must have at least 3 rows and 3 columns in your DTM

*** WARNING *** You must specify at least one argument

Examples:

TRIGRID> **SIZE 101 201<CR>**
TRIGRID>

SPAWN

The SPAWN command enables you to create a subprocess while within TRIGRID.

FORMAT: SPAWN command-line

Command parameters:

command-line

Specifies a DCL command string to be executed as if typed in response to a '$' prompt. When the command completes, the subprocess terminates and control is returned to TRIGRID. The command string cannot exceed 80 characters.

DESCRIPTION:

The SPAWN command enables you to create a subprocess while within TRIGRID. When the subprocess terminates control is returned to TRIGRID.

Messages:

The following warning messages are specific to the SPAWN command:

*** WARNING *** SPAWN requires a valid DCL command line

*** ERROR *** Unable to spawn command, returning to TRIGRID

Examples:

TRIGRID> **SPAWN DIRECTORY *.DTA;*<CR>**

```
Directory DUA3:[DTMCREATE.ACCEPTANCE_TESTS]

TEST1.DTA;1              8/8       18-AUG-1987 07:56  [LSL,TIM]
TEST2.DTA;2              7/8       18-AUG-1987 17:17  [LSL,TIM]
TEST2.DTA;1              7/8       18-AUG-1987 17:07  [LSL,TIM]

Total of 3 files, 22/24 blocks.

TRIGRID>
```

TRIANGLE_LIMITS

The TRIANGLE_LIMITS arguments set limits for the interpolation function selected for DTM post estimation within the individual triangles.

FORMAT: TRIANGLE_LIMITS real1 real2

COMMAND PARAMETERS:

real1

Undershoot limit for individual triangles. This is the amount (measured in the model Z value units) that an interpolation within the triangle may fall under the range of heights given by the values at the vertices of the triangle. A suggested limit is approx 1/3 contour interval. Default is 300000, i.e. no effectively limit!

real2

Overshoot limit for individual triangles. This is the amount (measured in the model Z value units) that an interpolation within the triangle may exceed the range of heights given by the values at the vertices of the triangle. A suggested limit is approx 1 contour interval. Default is 300000, i.e. no effectively limit!

Description:

By default TRIGRID applies no limits to post height estimation.

The undershoot and overshoot limits for individual triangles define how far an interpolation within the triangle may fall below or above the range of heights given by the values at the vertices of the triangle. These undershoot and overshoot limits are automatically modified by the ENABLE AUTO_LIMITS command when appropriate node feature flags are detected. For example, a triangle which has one or more nodes flagged as "river" nodes will be allowed no interpolation undershoot at all.

No triangle limits are applied if the linear facet interpolation option is selected as the interpolation is restricted to the triangle facet plane itself. The same effect can be achieved (rather extravagantly) by setting both the undershoot and overshoot triangle limits to 0.0 and then using the SMOOTH patch interpolation option!

Messages:

The following messages are specific to the TRIANGLE_LIMITS command:

*** WARNING *** You must specify lower and upper limit arguments
Lower limit, upper limit?

Examples:

```
TRIGRID> SHOW LIMITS<CR>
No individual triangle interpolation limits will be applied
No whole DTM Z-limits will be applied
TRIGRID> TRIANGLE_LIMITS 3.0 10.0<CR>
TRIGRID> SHOW LIMITS<CR>
Individual triangle undershoot limit .................  3.000
Individual triangle overshoot limit ..................  10.00
Whole DTM lower z-value limit ........................  Unset
Whole DTM upper z-value limit ........................  Unset
TRIGRID>
```

UNITS

Specifies the units of measurement that will be used when defining the DTM window using the WINDOW command.

The command also controls the units of measurement which will be used when displaying DTI file header details.

FORMAT: UNITS units

Command parameters:

units

A keyword defining the measurement units, chosen from:

METRES	Metres on the ground
LATLONG	Latitude and Longitude (in degrees, minutes and seconds)
SECONDS	Seconds of arc
PROJECTION	Projection units

DESCRIPTION:

The UNITS command enables the user to specify in what units of measurement he wishes to define the DTM window using the WINDOW command, or in what units of measurement details from the header of the DTI file are displayed.

By default metre units are assumed.
The UNITS command should be given before specifying the DTM window the user wishes to specify the window in non-metre units.

If UNITS SECONDS or UNITS LATLONG are used it is assumed that the data read in from the .DTA and .NOD files using FILEIN commands are in units of tenth seconds of arc.

UNITS METRES (the default) or UNITS PROJECTION assume that the the data read in using FILEIN commands are in metres or projection units.

Remember that TRIGRID assumes that if UNITS SECONDS or UNITS LATLONG are selected the basic units in the .DTA snd . NOD files are tenth seconds arc.

The UNITS command does not affect the values specified as arguments to the SIDELENGTH command. These should be expressed in terms of the basic units held in the .NOD and .DTA files (i.e. tenth second arc units in the case of geographical SECONDS or LATLONG). In the case of UNITS LATLONG, window values should be expressed as south-west **Latitude Longitude** north-east **Latitude Longitude**, **not Longitude Latitude**. The SHOW WINDOW command will reflect the window extent in both LATLONG units and tenth seconds arc.

Messages:

The following error messages are specific to the UNITS command:

*** ERROR *** Specifying command UNITS
Command qualifiers are METRES,PROJECTION,SECONDS or LATLONG

Examples:

$ **TRIGRID<CR>**
DTMCREATE module TRIGRID of 14:28:22 20-JAN-89
TRIGRID> **FILEIN WEEPY<CR>**

.DTA file LSL$SOURCEROOT:[DTMCREATE.TRIGRID]WEEPY.DTA;8 opened for read
.NOD file LSL$SOURCEROOT:[DTMCREATE.TRIGRID]WEEPY.NOD;8 opened for read
.DER file LSL$SOURCEROOT:[DTMCREATE.TRIGRID]WEEPY.DER;8 opened for read
TRIGRID> **UNITS LATLONG<CR>**
TRIGRID> **SHOW WINDOW<CR>**

Window units are latlong (degrees, minutes and seconds)

Triangulation coverage SW: 4 45 00N 13 44 20E NE: 4 49 00N 13 46 50E
(SW: 494600.00 171000.00 NE: 496100.00 173400.00)
DTM WINDOW currently undefined.

TRIGRID> **WINDOW 4 45 00N 13 45 00 E 4 50 00N 13 46 00E<CR>**
TRIGRID> **SIDELENGTH 10<CR>**

X and Y call side lengths assumed equal (10.000 10.000)

Window units are latlong (degrees, minutes and seconds)

Triangulation coverage SW: 4 45 00N 13 44 20E NE: 4 49 00N 13 46 50E
(SW: 494600.00 171000.00 NE: 496100.00 173400.00)

Triangulation window SW: 13 45 00N 4 45 00E NE: 13 46 00N 4 50 00E
(SW: 171000.00 495000.00 NE: 174000.00 495600.00)

Which with a side length of 10.000 in X and 10.000 in Y
gives 61 rows and 301 columns in the dtm

TRIGRID> **SHOW WINDOW<CR>**

Window units are latlong (degrees, minutes and seconds)

Triangulation coverage SW: 4 45 00N 13 44 20E NE: 4 49 00N 13 46 50E
(SW: 494600.00 171000.00 NE: 496100.00 173400.00)

Triangulation window SW: 13 45 00N 4 45 00E NE: 13 46 00N 4 50 00E
(SW: 171000.00 495000.00 NE: 174000.00 495600.00)
TRIGRID>

WAIT

Suspend processing for the specified number of seconds.

FORMAT: WAIT seconds

Command parameters:

seconds

The number (floating point) of seconds for which TRIGRID processing is to be suspended.

DESCRIPTION:

The WAIT command causes processing to be suspended for a specified number of seconds. It is designed for use in software demonstration situations and is of no value in a production flowline.

Messages:

The following warning message is specific to the WAIT command:

*** WARNING *** You must specify the number of seconds to wait

---

Examples:

TRIGRID> **WAIT 4.0<CR>**
TRIGRID>

WINDOW

Specifies the limits of the triangulation area to be gridded.

FORMAT: WINDOW xmin ymin xmax ymax

Command parameters:

xmin ymin

The coordinates of the bottom left hand corner of the defining rectangle.

xmax ymax

The coordinates of the top right hand corner of the defining rectangle.

DESCRIPTION:

The WINDOW command is used to define rectangular limits to the area of triangulation to be gridded into a DTI file. The limits must be specified in the order bottom left hand (or south west) corner, then top right hand (or north east) corner.

The WINDOW command can be used to clip data from a triangulation.

There is only one restriction on the WINDOW grid coverage allowed relative to the triangulation extent. The grid DTM must have a minimum of 3 columns and 3 rows. The user may specify that the grid DTM WINDOW extends beyond the triangulation area. Of course there are no original data points for interpolation in the resulting marginal areas and the proclivities of a poorly controlled mathematical surface may not truly represent the real world!

If the user is unsure of the triangulation (X,Y) extent a SHOW WINDOW command will supply this information assuming that the triangulation files have previously been opened with the FILEIN command.

The WINDOW command is obligatory.

Messages:

The following warning messages are specific to the WINDOW command:

*** WARNING **** Unable to read WINDOW arguments

*** WINDOW **** is still unset, please respecify the WINDOW command

*** ERROR **** window values define a zero width window

*** ERROR **** window values define a zero height window

Examples:

TRIGRID> **WINDOW 0.0 0.0 120.0 120.0<CR>**
TRIGRID>

ZLIMITS

The ZLIMITS arguments set overall model top and bottom z-limits for the interpolation function selected for DTM post estimation.

FORMAT: ZLIMITS real1 real2

COMMAND PARAMETERS:

real1

The absolute lower range limit for the whole DTM. No posts in the DTM will be allowed to have Z values below this value. The default is 0.0

real2

The absolute upper range limit for the whole DTM. No posts in the DTM will be allowed to have Z values above this value. The default is 300000.0 i.e. effectively no limit!

Description:

By default TRIGRID applies no limits to post height estimation.

The ZLIMITS command specifies lower and upper absolute range limits to enable the user to force a base and ceiling value for the whole model which the interpolation cannot exceed or fall under. This is particularly useful for models containing areas of sea as the lower limit can be set to zero ensuring that the sea cannot receive negative post values under any circumstances. A model whose highest area forms a plateau can be similarly controlled.

Messages:

The following messages are specific to the ZLIMITS command:

*** WARNING *** You must specify lower and upper limit arguments
Lower limit, upper limit?

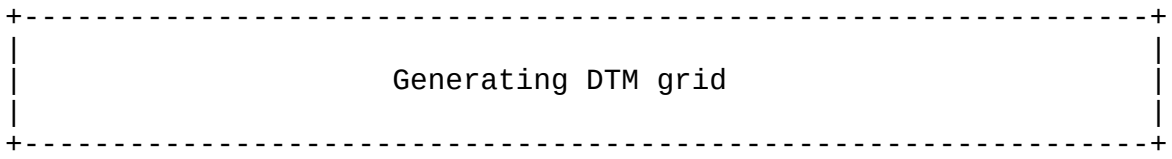
Examples:

```
TRIGRID> ZLIMITS 150.0 500.0<CR>
TRIGRID> SHOW LIMITS<CR>
Individual triangle undershoot limit .................  Unset
Individual triangle overshoot limit ..................  Unset
Whole DTM lower z-value limit ........................   150.000
Whole DTM upper z-value limit ........................   500.000
TRIGRID>
```

EXAMPLES

```
$ TRIGRID<CR>
DTMCREATE module TRIGRID of 14:02:31  3-FEB-89
TRIGRID> @TESTCMD<CR>
Command input now being read from SYS$DISK:[]TESTCMD.COM;0
TRIGRID> FILEIN WEEPY
.DTA file LSL$SOURCEROOT:[DTMCREATE.TRIGRID]WEEPY.DTA;9 opened for read
.NOD file LSL$SOURCEROOT:[DTMCREATE.TRIGRID]WEEPY.NOD;9 opened for read
.DER file LSL$SOURCEROOT:[DTMCREATE.TRIGRID]WEEPY.DER;9 opened for read
TRIGRID> ZLIMITS 150 500
TRIGRID> TRIANGLE
TRIGRID> ENABLE DIAGNOSTICS
TRIGRID> WINDOW 494600.00  171000.00 496100.00  173400.00
TRIGRID> SIDELENGTH 10 10

Window units are metres

Triangulation coverage SW:  494600.00  171000.00    NE:  496100.00  173400.00

Triangulation window   SW:  494600.00  171000.00    NE:  496100.00  173400.00

Which with a side length of    10.00 in X and     10.00 in Y
gives 241 rows and 151 columns in the dtm

TRIGRID> SHOW ENABLE
AUTO_LIMITS ..................................... On
DIAGNOSTICS ..................................... On
GRAPHICS ........................................ Off
SMOOTH .......................................... On
(Smooth patch interpolation)
NINT ............................................ On
ORTHOGONAL ...................................... Off
TRACE ........................................... Off
TRIGRID> SHOW LIMITS
Individual triangle undershoot limit .................     5.000
Individual triangle overshoot limit ..................    15.000
Whole DTM lower z-value limit ........................   150.000
Whole DTM upper z-value limit ........................   500.000
Input files:
.DTA file:   LSL$SOURCEROOT:[DTMCREATE.TRIGRID]WEEPY.DTA;9
.NOD file:   LSL$SOURCEROOT:[DTMCREATE.TRIGRID]WEEPY.NOD;9
.DER file:   LSL$SOURCEROOT:[DTMCREATE.TRIGRID]WEEPY.DER;9

Output file:
The DTI output file is undefined
TRIGRID> SHOW FILES
Input files:
.DTA file:   LSL$SOURCEROOT:[DTMCREATE.TRIGRID]WEEPY.DTA;9
.NOD file:   LSL$SOURCEROOT:[DTMCREATE.TRIGRID]WEEPY.NOD;9
.DER file:   LSL$SOURCEROOT:[DTMCREATE.TRIGRID]WEEPY.DER;9

Output file:
```

```
The DTI output file is undefined
TRIGRID> SHOW WINDOW
Window units are metres

Triangulation coverage SW:  494600.00  171000.00    NE:  496100.00  173400.00

Triangulation window   SW:  494600.00  171000.00    NE:  496100.00  173400.00
TRIGRID> GO
*** WARNING *** The following commands must be given
                before processing can begin:
                1) WINDOW
                2) SIZE (or SIDELENGTH)
                3) FILEIN
                4) FILEOUT


*** WARNING ***  Indirect file error - returning to terminal input
TRIGRID> FILEOUT TESTDTM<CR>
TRIGRID> GO<CR>

+-----------------------------------------------------------------+
|                                                                 |
|             Reading in data and point in box markers            |
|                                                                 |
+-----------------------------------------------------------------+


 WARNING  No projection information available for DTI header

+-----------------------------------------------------------------+
|                                                                 |
|                  Reading slope information                      |
|                                                                 |
+-----------------------------------------------------------------+



+-----------------------------------------------------------------+
|                                                                 |
|                   Opening and initialising DTI file             |
|                                                                 |
+-----------------------------------------------------------------+


DTI file LSL$DATAROOT:[LSL.DTI]TESTDTM.DTI;0 opened for write

File    :  LSL$DATAROOT:[LSL.DTI]TESTDTM.DTI;0
Header  :  LSLA  Data:  WORD

Units are metres

Matrix Coverage   SW:       0.00       0.00    NE:     199.50     199.50
Matrix Interval    E:       2.10                N:       1.50
Value Range         :     0   to      0
```




```
 ELAPSED:    0 00:00:10.96  CPU: 0:00:00.94  BUFIO: 14  DIRIO: 25  FAULTS: 347
```

This example shows a sequence of TRIGRID commands used to generate a DTI output file having a 'LSLA' type header and word integer post values.

TRIGRID introduces itself by giving the full name and creation date of the current version of the program. The user has chosen to issue the TRIGRID commands by use of an indirect command file using the @file-spec facility. After successfully opening the specified command file, SYS$DISK:[]TESTCMD.COM; TRIGRID issues a message indicating the file-spec used. All command input will be taken from this file until end of file is detected, a command error is detected, a RETURN command is detected, or a GO command results in successful termination of the run.

The first command specifies that input is to be from a file called WEEPY. Only a file-name is supplied and TRIGRID parses this against the user's current default device and directory specification. This results in the file specification LSL$SOURCE_ROOT:[DTMCREATE.TRIGRID]WEEPY.DTA This is used as a generic specification for all three input files required by TRIGRID, a .DTA file containing the internode relationships, a .NOD file containing the nodes themselves and a .DER file containing slope derivatives.

These files are opened in turn and a message indicating successful opening issued.

The ZLIMITS command specifies that no DTM post values are to be generated below 100 or above 500 height units. The height units were originally set in TRIANG.

The TRIANGLE_LIMITS command is used to set the undershoot (5) and overshoot limits (15) for individual triangles. These limits define how far an interpolation within the triangle may fall below or above the range of heights given by the values at the vertices of the triangle. The undershoot and overshoot are automatically modified by the ENABLE AUTO_LIMITS command when appropriate node feature flags are detected. For example, a triangle which has one or more nodes flagged as "river" nodes will be allowed no interpolation undershoot at all. AUTO_LIMITS are enabled by default on program startup and must be explicitly disabled if not required.

As this program run is to be used as an example, the user has used the ENABLE DIAGNOSTICS command to cause TRIGRID to output diagnostic messages. Where a processing stage may take a long time, the selection of diagnostics will cause messages indicating percentage progress to be output. In the interests of compactness, these percentage progress messages are not shown in this example.

The WINDOW command specifies the area of the triangulation that is to be gridded into a DTI file. There is only one restriction on the grid coverage allowed relative to the triangulation extent. The grid DTM must have a minimum of 3 columns and 3 rows. The user may specify that the grid DTM WINDOW extends beyond the triangulation area. Of course there are no original data points for interpolation in the resulting marginal areas and the proclivities of a poorly controlled mathematical surface may not truly represent the real world!

If the user is unsure of the triangulation (X,Y) extent a SHOW WINDOW command will supply this information assuming that the triangulation files have previously been opened with the FILEIN command.

With the SIDELENGTH command the user has elected to specify explicit (X,Y) DTM inter-post sidelengths rather than specify the WINDOW and then a definitive grid size. TRIGRID calculates the DTM grid size to be 241 rows and 151 columns.

The SHOW commands enable the user to examine the current settings for all the TRIGRID run time options. What the user has failed to notice from this listing is the fact that he has yet to specify an output file specification with the FILEOUT command.

The user, believing that all is set up for this run, specifies the GO command to commence TRIGRID processing. TRIGRID performs checks that all the obligatory commands have been specified and discovers an omission. The warning message indicates the 4 obligatory commands that must be issued before TRIGRID processing can be started. The GO command is ignored. As command input was from indirect command file, TRIGRID issues a further warning message, closes the indirect file and prompts for further commands at the terminal.

The user, realising that he has not specified a FILEOUT command, now does so. The missing parts from the file-spec supplied are taken from the default LSL$DTI:DTI.DTI;0. Satisfied with this, he reissues the GO command, which is this time accepted. TRIGRID processing begins. Messages indicating the start of each major processing stage are output.

The warning that no projection information is available for the DTI header is issued because TRIANG could find no projection information to put into the input .NOD and .DTA files when forming the triangulation. Input must have been from IFF files with unset type 2 MDs (Map Descriptor) or DTI files without projection records. This will not affect the TRIGRID run, but the DTI file header information should be checked carefully and the DTI file projection record possibly set up using DTITRANS.

After opening the output DTI file, its header is printed to the terminal. From this it is evident that the default file header type is 'LSLA' and the default DTI file post data type is word (16 bit) integer.

The run completed successfully. DCL symbol $STATUS is set to SS$_NORMAL, normal successful completion. The DTM grid in the DTI file may now be viewed using DTIVIEW.

MESSAGES (INFORMATIONAL)

These messages give information only, and require no immediate action by the user. They are used to provide information on the current state of the program, or to supply explanatory information in support of a warning or error message.

BADRANGE, Z value is out of range for DTI data type

Explanation: A Z value has been calculated for the current post which lies outside of the permissible range of the selected DTI file data type. In the case of a DTI file having the WORD data type, Z values must lie in the range -32768 to +32767. TRIGRID assumes a default value of -32767 for the post.

User action: If default post values in the DTI file are unacceptable, re-run TRIGRID and select a DTI file data type which can hold the large Z values that your data contains.

MESSAGES (WARNING)

These messages are output when an error has occurred that can be corrected immediately by the user or that the program will attempt to overcome.

SEAFILE, Unable to find 'file-spec'

Explanation: Before trying to open an input .NOD, .DTA or .DER file TRIGRID searches the disk for the file specification supplied. If TRIGRID finds it then all is well, the file-spec is parsed and the version number extracted to make up the name of the .DER file. Unfortunately you have specified a non-existent file-spec. TRIGRID will allow you to try again.

User action: Respecify the input file-spec.

WRTDTI, Error detected while writing to the DTI file at column 'integer', row 'integer', Z-value 'real''

Explanation: An error has been detected while writing the specified Z value to the DTI file at the specified position. The accompanying message will indicate the precise nature of the error and will determine whether processing is to be terminated, or alternatively if a null value of -32767 is to be applied for this post.

User action: The accompanying message will indicate the severity of this error. If, for example, the Z value was out of range for the DTI data type, processing will continue but a defauly value of -32767 will be applied for the affected post in the DTI file. If default post values in the DTI file are unacceptable, re-run TRIGRID and select a DTI file data type which can hold the large Z values that your data contains. Other errors may result in the termination of processing altogether. The user must decide what action is to be taken on the basis of the information supplied.

MESSAGES (ERROR)

These messages indicate an error in processing which will cause the program to terminate. The most likely causes are a corrupt or otherwise invalid input file, or an error related to command line processing and file manipulation.

OPNDER, error opening 'file-spec' for read

Explanation: The system error return supplied with the message will help you to decide what to do.

User action: Check:

1. That the file exists,
2. that you are in the correct directory,
3. that you have the privilege to read the file.

OPNDTA, error opening 'file-spec' for read

Explanation: The system error return supplied with the message will help you to decide what to do.

User action: Check:

1. That the file exists,
2. that you are in the correct directory,
3. that you have the privilege to read the file.

OPNNOD, error opening 'file-spec' for read

Explanation: The system error return supplied with the message will help you to decide what to do.

User action: Check:

1. That the file exists,
2. that you are in the correct directory,
3. that you have the privilege to read the file.

OPNSCR, error opening scratch file

Explanation: TRIGRID sometimes needs to open a scratch file to store extended neighbours of imaginary points and possibly another to hold side of line relationships created by the TRACE option. However, TRIGRID has failed to open the specified disk file.

User action: The supplementary message given after this error should help you to decide what has gone wrong (e.g. disk full, file protection error, etc.). Correct this problem and then re-run TRIGRID.

RDDTA, error reading from .DTA file

Explanation: An error has occurred while reading the .DTA file on disk. This error message will be accompanied by a supplementary RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient privilege or file protection violation) before attempting to re-run TRIANG.

RDNOD, error reading .NOD file

Explanation: TRIGRID has suffered a read error when reading the .NOD file. This error message will be accompanied by a supplementary RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient privilege or file protection violation) before attempting to re-run TRIANG. If the supplementary RMS error message indicates that the end of file was unexpectedly found check that the TRIANG or TRIEDIT run which created the .NOD file terminated successfully. If it disn't then re-run TRIANG or TRIEDIT and then try TRIGRID agaion on the corrected file. If all appears to be correct and the problem persists, please submit an SPR to Laser-Scan.

RDSCR, error reading from scratch file

Explanation: An error has occurred reading from a temporary scratch file used when there are too many nodes to hold in memory. This error message will be accompanied by a supplementary FORTRAN or RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient privilege or file protection violation) before attempting to re-run TRIANG.

RDSLP, error reading slope information from .DER file

Explanation: An error has occurred while reading the .DER file on disk. This error message will be accompanied by a supplementary RMS message which will indicate the cause of the error.

User action: This depends on the cause of the error. Correct the cause of the error (e.g. insufficient privilege or file protection violation) before attempting to re-run TRIANG.

MESSAGES (FATAL)

These messages indicate a severe error in processing, or some form of system failure, which has caused the program to terminate.

BOXOVR, too little space for boxes

> **Explanation:** TRIGRID uses the box structure created by TRIANG. Your current version of TRIGRID is insufficiently dimensioned to cope with the number of boxes just read in.
>
> **User action:** This should never happen as the DTMCREATE modules are dimensioned to be mutually compatible! Please submit an SPR to Laser-Scan. Until TRIGRID can be redimensioned divide up your original IFF file and re-run TRIANG on the resulting sub-areas. TRIGRID will then probably be able to cope with the reduced data set size. The resulting sub-DTMs can be joined to form the whole DTM area using DTITILE.

NODOVR, node has more than 150 neighbours

> **Explanation:** TRIGRID can currently can only handle nodes with less than 150 neighbours. The data is almost certainly very corrupt if this message appears, possibly as a result of over enthusiastic data insertion in TRIEDIT. Normally a node will only have up to about 9 neighbours.
>
> **User action:** The triangulation is irrevocably damaged. Re-run TRIANG and try running TRIGRID again. If the problem persists please submit an SPR to Laser-Scan.

RANDRD, error during random read

> **Explanation:** An error has occurred reading from a random access disk file used when there are too many nodes to hold in memory. This error message will be accompanied by a supplementary FORTRAN or RMS message which will indicate the cause of the error.
>
> **User action:** Please save all the data used for input and then submit an SPR to Laser-Scan.

STACKOVR, stack overflow

> **Explanation:** TRIGRID has only a finite amount of space available to store all the nodes and thir neighbour relationships. It is very unlikely that you will get this message as all modules in DTMCREATE share the same dimensioning parameters and it should be impossible for TRIANG or TRIEDIT to output .DTA and .NOD files which require such a large stack.
>
> **User action:** Please submit an SPR to Laser-Scan.

TOMNYNEIB, too many neighbours

Explanation: TRIGRID has encountered a node with more neighbours than can be safely stored. This should never happen as all the modules of the DTMCREATE package share common workspace dimensioning parameters and this node should have been rejected by TRIANG.

User action: Please submit an SPR to Laser-Scan.

UNRECREC, unrecognised record number

Explanation: Each node entry in workspace is identified by a positive record number, if there are more records than can be held in memory then the remainder are written to a random access disk file. Somehow TRIGRID has found a record with an identification that is either less than 1 or greater than the current maximum recorded number of records.

User action: Unless another problem has occurred during this TRIGRID run this error should not occur. Please save all the data used for input and then submit an SPR to Laser-Scan.

MESSAGES (OTHER)

In addition to the above messages which are generated by the program itself, other messages may be produced by the command line interpreter (CLI) and by Laser-Scan libraries. In particular, messages may be generated by the DTILIB library and by the Laser-Scan I/O library, LSLLIB. DTILIB library messages are introduced by '%DTILIB' and are documented in the MATRIX package reference manual. In most cases DTI errors will be due to a corrupt input file, and this should be the first area of investigation. If the cause of the error cannot be traced by the user, and Laser-Scan are consulted, then the output file should be preserved to facilitate diagnosis. LSLLIB messages are introduced by '%LSLLIB' and are generally self-explanatory. They are used to explain the details of program generated errors.

APPENDIX A

DTMCREATE GRAPHICS LOOKUP FILES

Construction of DTMCREATE graphics lookup files

General

DTMCREATE module graphics options are controlled by three lookup files which are accessed via logical name LSL$LOOKUP: The files are CONFIGURE.DAT, CONFIGDEF.DAT and TERMTYPE.DAT. The logical name LSL$LOOKUP defines a search list. The search list is constructed such that a site dependent directory LSL$SITE_ROOT:[LSL.LOOKUP] will be accessed before the public directory LSL$PUBLIC_ROOT:[DTMCREATE.LOOKUP]. On delivery of the DTMCREATE package to the customer site, the lookup files will be contained in LSL$PUBLIC_ROOT:[DTMCREATE.LOOKUP] and the lookup files will, by default, contain definitions for use with VAXstations only. Should the user wish to modify the default definitions files supplied with the package, the files should first be copied to LSL$SITE_ROOT:[LSL.LOOKUP] with the command:

$ **COPY/LOG<CR>**
_From: **LSL$PUBLIC_ROOT:[DTMCREATE.LOOKUP]CONFIGDEF.DAT, -<CR>**
_$ **CONFIGURE.DAT,TERMTYPE.DAT<CR>**
_To: **LSL$LOOKUP:<CR>**

This will have the effect of copying the files to LSL$SITE_ROOT:[LSL.LOOKUP]. The user should then edit the copies of the files to make them site dependent, **NOT** the original files contained in LSL$PUBLIC_ROOT:[DTMCREATE.LOOKUP]. Failure to copy the files before editing them for the first time may result in the loss of any site dependent features of the files next time a DTMCREATE package upgrade is installed using the LAMPSINSTALL procedures.

The copied lookup files may be changed using VAX/VMS EDIT at any time to allow for changes in serial line allocation and hardware availability.

Graphics option records in the lookup files are referenced via the ident of the serial line currently in use. If the user has access to a terminal server, the serial line idents vary each time the user logs on, even at the same terminal. To overcome this problem DTMCREATE uses the translation of a logical name, LSL$DTMCREATETERMINAL as the key by which graphics lookup files are referenced. This logical name is set up for the user whenever he logs onto a terminal server line. The logical name translates as "LTXX". Records in the graphics lookup file are then referenced using "LTXX" instead of the "real" serial line ident e.g. LTA125:. If there is a terminal server in use at your site ensure that there is a record in the graphics lookup files for serial line "LTXX". If there is no terminal server in use at your site logical name LSL$DTMCREATETERMINAL will not be defined, unless you choose to explicitly do so.

If a Tektronix 4000 or 4100 series or a SIGMEX 6100 series terminal is to be used, ensure that logical name LSL$TK is defined. At most customer sites this is done in the LITES2 initialisation file LSL$COM:LITES2INI.COM.

See the relevant DTMCREATE Workstation Guide for further details of hardware dependent installation requirements, lookup file options and program useage.

Lookup table for TRIANG TRIDER and TRIGRID graphics

TRIANG, TRIDER and TRIGRID offer the user the option to display the progress of their calculations on a graphics device. The user must be logged on to the serial line associated with the graphics device of his choice. If a graphics option is selected the program opens a lookup file and checks that the current serial line supports a graphics device. If it does then the graphics device is initialised using routines appropriate to the type of graphics device described in the file.

Currently supported graphics devices are:

1. T4010 - Tektronix 4010,

2. T4014 - Tektronix 4014,

3. T4100 - Tektronix 4100 series terminal,

4. S6100 - Sigmex 6100 series terminal

5. S7000 - Sigma ARGS 7000,

6. VT100 - able to clear screen only,

7. VAXstation

8. NONE - as it says (default),

TRIANG TRIDER and TRIGRID share the same lookup file: LSL$LOOKUP:TERMTYPE.DAT

The directory containing the lookup files is pointed to by the system wide logical name LSL$LOOKUP:. The file is organised on the basis of the user's serial line identification. The first stage of lookup file interrogation is to compare the current terminal line identification with the line idents (which are the first item on each active record of the lookup file). If the serial line identification matches then the rest of the record of the lookup file is read. The lookup file may be changed using VAX/VMS EDIT at any time to allow for changes in serial line allocation and hardware availability.

An example lookup file for TRIANG,TRIDER and TRIGRID is listed below:

```
!*********************************************************************
!*                        T E R M T Y P E . D A T                    *
!*********************************************************************
!
! This file contains the hardware configuration for  each terminal line
! likely to be used for graphics generated by  DTMCREATE modules TRIANG,
! TRIDER and TRIGRID.
!
! N.B. Two separate files LSL$LOOKUP:CONFIGURE.DAT and
! LSL$LOOKUP:CONFIGDEF.DAT define the hardware available for
! DTMCREATE module TRIEDIT.
!
! On initial site installation this file should be copied to
! LSL$SITE__ROOT:[LSL.LOOKUP] and modified to suit the site requirements.
!
!
! Available options are:
!
!       Graphics devices
!                               S6100
!                               S7000
!                               T4010
!                               T4014
!                               T4100 - all Tek 4100 series
!                               GPX
!                               VT100
!
! A  "!" is a comment delimiter and the remainder of a line after a "!"
! will be ignored.
!
! The following facilitate use of DTMCREATE with a VAXstation
WTA1:           GPX
WTA2:           GPX
WTA3:           GPX
WTA4:           GPX
WTA5:           GPX
WTA6:           GPX
WTA7:           GPX
WTA8:           GPX
!
! If not running DTMCREATE on a VAXstation, the following example
! lines can be modified/deleted to suit your site requirements.
! Remember to remove the !
!TTA3:          T4100
!TTB2:          S6100
!TTB7:          S7000
!TXA7:          T4010
!
! If a terminal server is in use, LSL$COM:DTMCREATEINI.COM will have set
! up logical name LSL$DTMCREATETERMINAL to be "LTXX:". This overcomes the
! problems of arbitrary LTxxx terminal line idents produced by a terminal
! server.
!
! Remove the ! and alter the definition below to suit your site
! requirement if logging  onto a terminal server:
```

```
!LTXX:          T4010
!
!************************* End of TERMTYPE.DAT **********************
```

Note that a "!" denotes a comment and any text to the right of this flag will be ignored.

Lookup tables for TRIEDIT graphics

TRIEDIT uses a lookup file, LSL$LOOKUP:CONFIGDEF.DAT, to select and initialise the default hardware configuration for your particular terminal line at your site. Alternatively you may specify on the startup command line the hardware options that you wish to use, or select hardware options (but not graphic screen type) while you are running the program by using the ENABLE command. If you choose to select your own hardware options TRIEDIT checks in another site dependent lookup file, LSL$LOOKUP:CONFIGURE.DAT, that the desired options are available (and are in a valid combination) and then initialises the devices accordingly.

The contents of both the default hardware configuration lookup file and the option configuration lookup file are organised on the basis of the users serial line identification. The first stage of lookup file interrogation is to compare the current terminal line identification with the line idents (which are the first item on each active record of the lookup file). If the serial line identification matches then the rest of the record of the lookup file is read.

Records in the lookup files may be in free format, option names being separated by spaces. A comment facility is available; any text to the right of a "!" is ignored. The only restrictions to the layout of the files are:

1. The serial line identification must be the first item in a record.

2. All the options relating to one serial line must be contained on one line of the file

An example default hardware configuration lookup file is shown below:

```
!*********************************************************************
!*                        C O N F I G D E F . D A T                   *
!*********************************************************************
!
! This file contains the default hardware configuration for
! each terminal line likely to be used for DTMCREATE module TRIEDIT
! A complementary file, CONFIGURE.DAT contains the possible range of
! options supported by that line.
!
! On initial site installation this file should be copied to
! LSL$SITE__ROOT:[LSL.LOOKUP] and modified to suit the site requirements.
!
!
! Available options are:
!
!       Graphics devices
!                               S6100
!                               S7000
!                               T4010
!                               T4014
!                               T4105
!                               T4106
!                               T4107
!                               T4109
!                               T4115
!                               MUART_T4014
!                               GPX
!       Supplementary terminal
!                               VT100
!       GIN devices
!                               TRACKERBALL     - S7000 only
!                               MUART_TABLE     - MUART workstations only
!                               TABLE           - TABLE__MONITOR control
!                               THUMBWHEELS     - T4000 series only
!                               JOYSTICK        - T41000 series only
!                               MOUSE           - VAXstation only
!                               BITPAD          - SIGMEX 6100 series only
!       No graphics
!                               NOGRAPHICS
!
! A  "!" is a comment delimiter and the remainder of a line after a "!"
! will be ignored.
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
TERMINAL        AVAILABLE OPTIONS
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! The following facilitate use of DTMCREATE with a VAXstation
WTA1:           GPX MOUSE
WTA2:           GPX MOUSE
WTA3:           GPX MOUSE
WTA4:           GPX MOUSE
WTA5:           GPX MOUSE
WTA6:           GPX MOUSE
WTA7:           GPX MOUSE
WTA8:           GPX MOUSE
!
```

```
! If not running DTMCREATE on a VAXstation, the following example
! lines can be modified/deleted to suit your site requirements:
!TTA0:          NOGRAPHICS
!TTA3:          T4105 JOYSTICK
!TTB1:          MUART_T4014 MUART_TABLE
!TTB7:          S7000 TRACKERBALL
!TXA7:          T4010 THUMBWHEELS
!TXA7:          T4010 VT100 TABLE
!
! If a terminal server is in use, LSL$COM:DTMCREATEINI.COM will have set
! up logical name LSL$DTMCREATETERMINAL to be "LTXX:". This overcomes the
! problems of arbitrary LTxxx terminal line idents produced by a terminal
! server.
!
! Remove the ! and alter the definition below to suit your site
! requirement if logging  onto a terminal server:
!LTXX:          T4105 JOYSTICK VT100
!
!************************* End of CONFIGDEF.DAT **********************
```

And this is an example hardware configuration lookup file, against which user selected options are checked for availability:

```
!*********************************************************************
!*                        C O N F I G U R E . D A T                  *
!*********************************************************************
!
! This file contains the possible range of hardware configuration for
! each terminal line likely to be used for DTMCREATE module TRIEDIT
! A complementary file, CONFIGDEF.DAT contains the DEFAULT options
! supported by that line.
!
! On initial site installation this file should be copied to
! LSL$SITE__ROOT:[LSL.LOOKUP] and modified to suit the site requirements.
!
!
! Available options are:
!
!       Graphics devices
!                               S6100
!                               S7000
!                               T4010
!                               T4014
!                               T4105
!                               T4106
!                               T4107
!                               T4109
!                               T4115
!                               MUART_T4014
!                               GPX
!       Supplementary terminal
!                               VT100
!       GIN devices
!                               TRACKERBALL     - S7000 only
!                               MUART_TABLE     - MUART workstations only
!                               TABLE           - TABLE__MONITOR control
!                               THUMBWHEELS     - T4000 series only
!                               JOYSTICK        - T41000 series only
!                               MOUSE           - VAXstation only
!                               BITPAD          - SIGMEX 6100 series only
!       No graphics
!                               NOGRAPHICS
!
! A  "!" is a comment delimiter and the remainder of a line after a "!"
! will be ignored.
!
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
TERMINAL        AVAILABLE OPTIONS
!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!!
! The following facilitate use of DTMCREATE with a VAXstation
WTA1:           GPX MOUSE
WTA2:           GPX MOUSE
WTA3:           GPX MOUSE
WTA4:           GPX MOUSE
WTA5:           GPX MOUSE
WTA6:           GPX MOUSE
WTA7:           GPX MOUSE
WTA8:           GPX MOUSE
```

```
!
! If not running DTMCREATE on a VAXstation, the following example
! lines can be modified/deleted to suit your site requirements:
!TTA0:          NOGRAPHICS VT100
!TTA3:          T4105 JOYSTICK NOGRAPHICS VT100 TABLE
!TTB1:          MUART_T4014 NOGRAPHICS MUART_TABLE
!TTB7:          S7000 TABLE TRACKERBALL
!TXA7:          T4010 THUMBWHEELS
!
! If a terminal server is in use, LSL$COM:DTMCREATEINI.COM will have set
! up logical name LSL$DTMCREATETERMINAL to be "LTXX:". This overcomes the
! problems of arbitrary LTxxx terminal line idents produced by a terminal
! server.
!
! Remove the ! and alter the definition below to suit your site
! requirement if logging  onto a terminal server:
!LTXX:          T4105 JOYSTICK NOGRAPHICS VT100 TABLE
!
!************************* End of CONFIGURE.DAT **********************
```

APPENDIX B

BIBLIOGRAPHY

BIBLIOGRAPHY

Akima, H., (1974)

A method of bivariate interpolation and smooth surface fitting based on local procedures, *Communications of the ACM,* Vol 17, No 1, 18-20.

Akima, H., (1978)

A method of bivariate interpolation and smooth surface fitting for irregularly distributed data points, *ACM Transactions on Mathematical Software,* Vol 4, No 2, 148-159.

Akima, H. (1978)

ALGORITHM 526. Bivariate interpolation and smooth surface fitting for irregularly distributed data points (E1), *ACM Transactions on Mathematical Software,* Vol 4, No 2, 160-164.

Barnhill, R.E., (1975)

Blending function interpolation: a survey and some new results, *International Series of Numerical Mathematics* 30, Birkauser Verlag, Basel.

Barnhill R.E., (1977)

Representation and Approximation of Surfaces, *Mathematical Software III,* edited by J.R. Rice, Academic Press, 69-118.

Barnhill, R.E. and Farin, G., (1981)

C1 Quintic interpolation over triangles - two explicit representations, *International Journal for Numerical Methods in Engineering,* Vol 17, 1763-1778.

Bhattacharyya, B.K., (1969)

Bicubic spline interpolation as a method for treatment of potential field data, *Geophysics,* Vol 34, No 3, 402-423.

Bolondi, G., Rocca, F. and Zanoletti, S. (1976)

Automatic Contouring of Faulted Sub-Surfaces, *Geophysics,* Vol 34, No 3, 402-423.

Brassel, K.E. and Reif, D. (1979)

Procedure to Generate Thiessen Polygons, *Geographical Analysis,* Vol 11, 289-303.

Campbell, J.B., Davis, J.C. and the Experimental Cartography Unit, (1979)

Northern Part of the Wakarusa Quadrangle, Kansas: Atlas, No 5, Series on Spatial Analysis, Kansas Geological Survey, Kansas, US, 29p.

Crain, I.K., and B.K. Bhattacharyya (1967)

Treatment of non-equispaced two-dimensional data with a digital computer. *Geoexploration,* Vol 5, 173-194

Davis, J.C., (1973)

Statistics and data analysis in geology, John Wiley and Sons, New York, 550p.

Davis, J.C., and McCullagh, M.J., (Eds) (1975)

Display and Analysis of Spatial Data, John Wiley and Sons, London, 383p.

Delaunay, B., (1934)

Sur La Sphere Vide, *Bulletin of the Academy of Sciences of the USSR,* Class Sci Mat Nat, 793-800.

Douglas, D.H. and Peucker, T.K., (1973)

Algorithms for the Reduction of the Number of Points Required to Represent a Digitised Line or its Caricature, *Canadian Cartographer* Vol 10, No 2, (Dec), 112-122.

Ellis, T.M.R., and McLain, D.H. (1977)

ALGORITHM 514: A new method of cubic curve fitting using local data (E2), *ACM Transactions on Mathematical Software,* Vol 3, No 2, 175-178.

Franke, R., (1977)

Locally determined smooth interpolation at irregularly spaced points in several variables, (1977), *Jnl Inst Maths Applics,* Vol 19, 471-482.

Franke, R., (1982a)

Smooth interpolation of scattered data by local thin plate splines, *Computer and Mathematics with Applications,* Vol 8, 273-281.

Franke, R., (1982b)

Scattered data interpolation - tests of some methods, *Mathematics of Computation,* Vol 38, 181-200.

Franke, R., and Neilson, (1980)

Smooth interpolation of large sets of scattered data, *International Journal for Numerical Methods in Engineering,* Vol 15, 1691-1704.

Gold, C., Charters, T. and Ramsden, J. (1977)

Automated Contour Mapping Using Triangular Element Data, *Computer Graphics,* 11, 170-175.

Green, P.J. and Sibson, R. (1978)

Computing Dirichlet Tessellations in the Plane, *Computer Journal,* 21, 168-172.

Hardy, R.L. (1971)

Multiquadric Equations of Topography and Other Irregular Surfaces, *Jour Geophysical Res,* Vol 76, No 8, 1905-15.

Hardy, R.L. (1977)

Least squares prediction, *Photogrammetric Engineering and Remote Sensing,* Vol 43, No 4, 475-492.

Hayes, J.G. and Halliday, J., (1974)

The least squares fitting of cubic spline surfaces to general data sets, *Jnl Inst Maths Applics,* Vol 14, 89-103.

Heap, B.R., (1972)

Algorithms for the Production of Contour Maps Over an Irregular Triangular Mesh, *National Physical Lab Report* NAC 10.

Journel, A.G. and Huijbregts, C.J., (1978)

Mining Geostatistics, Academic Press, London, 600p.

Klucewicz, I.M. (1978)

A piecewise C1 interpolant to arbitrarily spaced data,*Computer Graphics and Image Processing,* Vol 8, 92-112.

Lancaster, P., and Salkauskas, K., (1977)

A Survey of Curve and Surface Fitting, published by the authors, 3052 Conrad Drive, Calgary, Alberta, Canada.

Lodwick, G.D. and Whittle, J. (1970)

A Technique for Automatic Contouring of Field Survey Data,*Australian Computer Journal,* Vol 2, No 3, (Aug), 104-109.

McCullagh, M.J. (1980)

Triangulation Systems in Surface Representation, *Auto-Carto IV,* proceedings of International Symposium on Cartography and Computing, Vol 1, 146-153.

McCullagh, M.J. and Ross, C.G. (1980)

Delaunay Triangulation of a Random Data Set for Isarithmic Mapping, *Cartographic Journal,* Vol 17, No 2, (Dec), 93-99.

McCullagh, M.J., (1981)

Creation of Smooth Contours Over Irregularly Distributed Data Using Local Surface Patches, *Geographical Analysis,* Vol 13, No 1, (Jan), 51-63.

McCullagh, M.J., (1982)

Contouring of Geological Structures by Triangulation, *Geological Society of London,* Miscellaneous Paper No 15 - Computer Applications in Geology III, 47-60.

McCullagh, M.J., (1982b)

Mini/Micro Display of Surface Mapping and Analysis Techniques, *Cartographica,* Perspectives in the Alternate Cartography (Monograph 28, Euro-Carto I), Vol 19, No 2, 136-144.

McCullagh, M.J., (1983a)

Automatic Contouring of Faulted Seismic Data Sets, *Computer Graphics Forum,* Journal of Eurographics Association, (in press).

McCullagh, M.J., (1983b)

Transformation of Contour Strings to a Rectangular Grid Based Digital Elevation Model, *Cartographica,* (Monograph of Euro-Carto II), (in press).

McCullagh, M.J. and Bennett, A.M. (1983c)

An Interactive Surface Modelling System for Thematic Data, *Cartographica,* (Monograph of Euro-Carto II), (in press).

McLain, D.H. (1974)

Drawing Contours from Arbitrary Data Points, *Computer Journal,* Vol 17, 318-324.

Olea, R.A. (1975)

Optimal Mapping Techniques using Regionalised Variable Theory. *Kansas Geological Survey Series on Spatial Analysis,* No 2, 137p.

Peucker, T.K., (1975)

A Theory of the Cartographic Line, *Proc Auto-Carto II,* ACSM and US Bureau of the Census, 508-518.

Peucker, T.K., (1979)

Digital Terrain Models: An Overview, *Proc Auto-Carto IV,* ACSM and ASP,97-107.

Peucker, T.K., (1980)

The impact of different mathematical approaches to contouring, *Cartographica,* Vol 17, 73-95.

Peucker, T.K. and Chrisman, (1975)

Cartographic data structures *The American Cartographer,* Vol 2, No 1, 55-69.

Pfaltz, J.L., (1977)

Computer Data Structures, McGraw-Hill, New York, 446p.

Pouzet, J. (1980)

Estimation of a Surface with Known Discontinuities for Automatic Contouring Programs, *Jour Int Assn Math Geol,* Vol 12, No 6, 559-576.

Powell, M.J.D., (1976)

Numerical methods for fitting functions to two variables, presented at the IMA Conference on the State of the Art in Numerical Analysis, University of York (April 1976).

Rhind, D. (1975)

A Skeletal Overview of Spatial Intepolation Techniques, *Computer Applications,* Vol 2, Nos 3 and 4, 293-309.

Rhynsburger, D. (1973)

Analytic Delineation of Theissen Polygons, *Geographical Analysis,* No 5, 133-144.

Ritchie SIM, (1978)

Surface Representation by Finite Elements, Unpubl Master's thesis, University of Calgary, Alberta, Canada.

Sabin, M.A., (1980)

Contouring - a review of methods for scattered data, in Brodlie, K.W., (Ed), *Mathematical methods in computer graphics and design,* Academic Press, New York, 63-86.

Sampson, R.J (1975)

The Surface II Graphics System, in *The Display and Analysis of Spatial Data,* edited by Davis J C and McCullagh M J, Wiley, 244-266.

Schagen, I.P. (1982)

Automatic Contouring from Scattered Data Points, *Computer Journal,* Vol 25, No 1, 7-11.

Sibson, R., (1981)

A brief description of natural neighbour interpolation, in Barnett, V. (Ed), *Interpreting multivariate data,* John Wiley and Sons, New York, 21-36.

Tipper, J.C., (1980)

Surface Modelling Techniques, *Series on Spatial Analysis,* No 4, Kansas Geological Survey, University of Kansas.

Tobler, W.R., (1979)

Lattice Tuning, *Geographical Analysis,* Vol 11, 1, 36-44.

Watson, D.F., (1981)

Computing the n-dimensional Delauney tesselation with application to Voronoi polytopes, *The Computer Journal,* Vol 24, 167-172.

Watson,D.F., (1982)

ACORD - Automatic contouring of raw data, *Computers and Geosciences,* Vol 8, 97-101.

Zienkeiwicz, O.C., (1971)

The Finite Element Method in Engineering Science, McGraw-Hill, London.

INDEX