Laser-Scan Ltd.

CONVERT PACKAGE

IFFTDST Reference

Issue 1.0 - 17-June-1991

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

Document "IFFTDST REFERENCE", Category "REFERENCE"
Document Issue 1.0   S Townrow (modified 17-Jun-1991)

CONTENTS

IFFTDST - Change Record

Version 1.0 J Barber 14-April-1991

Module IFFTDST - Addition of a /PARAMETER_FILE=file-spec qualifier (to supply a parameter lookup file for TDST2I), with associated messages and data preparation section, to the CONVERT User Guide.

Version 1.1 S Townrow 17-June-1991

Module IFFTDST - Reorganised package documentation.

PREFACE

Intended audience

This manual is intended for users of a specific utility of the Laser-Scan CONVERT package running under the VAX/VMS operating system. Each manual contains the documentation for a particular CONVERT utility and a site will only receive new or updated documentation for those utilities which they have purchased.

Structure of this document

This document is composed of 2 major sections.

The Introduction is an overview of the CONVERT package and its purpose.

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

MODULE - the name of the CONVERT module.

FORMAT DESCRIPTION - a description of the data format written or read by the utility programs in this conversion module.

DATA PREPARATION - guidance on how to digitise or prepare the IFF and other data required by the utility programs in this module.

For each utility program in the module, there will then be the following categories:

UTILITY - the name of the utility.

FUNCTION - a synopsis of what the utility does.

FORMAT - a summary of the utility 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.

RESTRICTIONS - a summary of restrictions on the use of

qualifiers

DESCRIPTION - the definitive description of the utility action.

COMMANDS - for interactive utilities only, a description of all commands. Commands are ordered alphabetically and default argument values are indicated.

EXAMPLES - annotated examples of utility 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.

Conventions used in this document

Convention	Meaning
<CR>	The user should press the carriage control 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>.
$ **IFF2SIF<CR>**	Command examples show all user entered commands in **bold** type.
$ **IFF2SIF<CR>** . . .	Vertical series of periods, or ellipsis, mean either that not all the data that CONVERT would display 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 indicates 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 assisnment statement).

Convention	Meaning
'integer'	An integer number is expected in the specified input or output field. (See "Command line data types" below).
'real'	A real number is expected in the specified input or output field. (See "Command line data types" below).
'file-spec'	A VMS file specification is expected in the specified input or output field.
'device-name'	A VMS device specification (for instance, MTA0:) is expected in the specified input or output field.

CHAPTER 1

TDST FORMAT DESCRIPTION

FORMAT DESCRIPTION

Introduction

TDST is the term used by Laser-Scan to describe SysScan's, otherwise unnamed, ASCII version of DST. DST is the file extension used by SysScan to identify GINIS data files, their main data format. TDST is produced by SysScan program UNLDST and is converted to DST by SysScan program LOADST. A brief comparison of the way IFF and TDST regard data is given below.

The format produced is an ASCII text format, as described in the document **GINIS I/O** of August 1988 (SysScan document "Reference Manual RM-025 Release 4.1").

For more information on the IFF file structure see the **IFF User's Guide** of the Laser-Scan MAPPING package documentation.

File output

TDST output is to disc file. The record length is variable. Each disc file contains the output from one IFF file.

These files may then be output to magnetic tape using either of the VAX programs BACKUP or COPY - the former is intended for transfer between VAXes, and the latter provides an ANSI standard tape format. Consult the relevant Digital manuals for more details of these utilities.

The TDST subset supported

The TDST data expected by TDST2I and produced by I2TDST is a subset of the complete format. I2TDST and TDST2I do not support the following parameters: GEOLINK, AREA, OBJECT and MULTIPOINT. This means that all IFF features are represented by, or converted to, one of four feature types: POLYGON, ARC, TEXT and POINT. The data fields of the unsupported parameters are simply read through and ignored by TDST2I so that the program run is not affected.

This section describes what may be expected in the output.

Summary of TDST format

TDST files start with a master level section. The format for a master level block is:

	(Fortran format)
MASTER	10A1
<name> <user-ID>	10A1,I11
<XLL> <YLL> <XUR> <YUR>	4F16.4
EOS	10A1

Where <name> is the name of the master block,
<user-ID> is the user identification number,
<XLL> <YLL> is the coordinate of the lower left corner and
<XUR> <YUR> is the coordinate of the upper right corner.

This master level block is then parent to any number of general logic level blocks. The format for a general logic level block is:

	(Fortran format)
LOGIC	10A1
<name> <user-ID> <HL-name> <HL-ID>	2(10A1,I11)
<ident for 2nd subsection>	10A1
.	
.	
.	
<ident for nth subsection>	10A1
.	
.	
.	
EOS	10A1

Where <name> is the name of the logic block,
<user-ID> is the user identification number of the logic block,
<HL-name> is the name of the higher level block (i.e. the master block, or some other logic block),
<HL-ID> is the user identification number of the higher level block and
<ident> is the identifier of additional subsections.

Four different types of additional subsections are accepted:

1) POLYGON - any linear feature
2) ARC - arc or circle feature
3) TEXT - text feature
4) POINT - point or symbol features

The polygon feature type is used for IFF features with Graphical Types of 1, 6, 11 and 12. The format for this subsection is:

	(fortran format)
POLYGON	10A1
<nlev>	I5
<nobj>	I5
<np> <colour> <thick> <type> <height>	4I5,F16.4
<x1> <y1> <x2> <y2>	4F16.4
<x3> <y3>	
.... <xnp> <ynp>	

Where <nlev> is the number of other logic levels this element belongs to, followed by the level names and ids, (I2TDST and TDST2I DO NOT SUPPORT THIS ENTITY).
<nobj> is the number of other objects/areas this element belongs to, followed by the object names,
<np> is the number of points (there is a maximum of 70 points),
<colour> is the colour number of the polygon,
<thick> is the line thickness in pixels,
<type> is the line type,
<height> is the height value of the line and
<xi> <yi> is the coordinate of point i (where i= 1 to np)

The arc feature type is used for IFF features with Graphical Types of 2, 3, 4 and 5. The format for this subsection is:

	(fortran format)
ARC	10A1
<nlev>	I5
<nobj>	I5
<colour> <thick> <type> <height>	3I5,F16.4
<xc> <yc> <rad> <alpha1> <alpha2>	5F16.4

Where <nlev> is the number of other logic levels this element belongs to, followed by the level names and ids, (I2TDST and TDST2I DO NOT SUPPORT THIS ENTITY).
<nobj> is the number of other objects/areas this element belongs to, followed by the object names,
<colour> is the colour number of the arc,
<thick> is the line thickness in pixels,
<type> is the line type,
<height> is the height value of the arc,
<xc> <yc> is the coordinate of the centre of the arc,
<rad> is the radius of the arc,

<alpha1> is the start angle in degrees relative to the x-axis and
<alpha2> is the angle from start to end point (the arc size)

All angles are in degrees and a positive angle is measured counterclockwise.

The text feature type is used for IFF features with a Graphical Type of 10. The format for this subsection is:

	(fortran format)
TEXT	10A1
<nlev>	I5
<nobj>	I5
<colour> <T-size> <isl> <iun> <nlin> <xt> <yt> <t-alpha>	5I5,3F16.4
<nch1>	I5
<line1>	40A1
.	
.	
.	
<nch nlin>	
<line nlin>	

Where <nlev> is the number of other logic levels this element belongs to, followed by the level names and ids, (I2TDST and TDST2I DO NOT SUPPORT THIS ENTITY).
<nobj> is the number of other objects/areas this element belongs to, followed by the object names,
<colour> is the colour number of the arc,
<T-size> is the size of the text in 1.10mm.,
<isl> is the index of inclination,
<iun> is the index of underlining,
<nlin> is the number of lines in the text (maximum is 5),
<xt> <yt> is the coord of the lower left corner of the first text line,
<t-alpha> is the start angle in degrees relative to the x-axis,
<nch i> is the number of characters in line i (maximum 40) and
<line i> is the text string in line i.

Positive angles are measured counterclockwise.

The point feature type is used for IFF features with Graphical Types of 7, 8 and 9. The format for this subsection is:

	(fortran format)
POINT	10A1
<nlev>	I5
<nobj>	I5
<sno> <colour> <xs> <ys> <height> <alpha>	2I5,4F16.4

Where <nlev> is the number of other logic levels this element belongs to, followed by the level names and ids,

(I2TDST and TDST2I DO NOT SUPPORT THIS ENTITY).

<nobj> is the number of other objects/areas this element belongs to, followed by the object names,

<sno> is the symbol number,

<colour> is the colour number of the point/symbol,

<xs> <ys> is the coord of the point/symbol,

<height> is the height value in metres for the point/symbol and

<alpha> is the rotation of the symbol in degrees relative to the x-axis.

Positive rotation is measured counterclockwise.

Comparison of TDST and IFF

IFF is a feature oriented data format - data is separated into features, and each feature represents one 'thing' on the map. An integer feature code is used to say what type of thing it is that the feature represents. A feature might thus be a contour, or a house, or a river, depending on the type of map being digitised, and the way that the data is to be used.

TDST format shares this concept of a 'feature', however the features are not uniquely feature coded. Each individual feature has coding relating to its graphical appearance such as colour and line type. The features are notionally coded by grouping similar features in a logic block which has a name describing its contents. If this grouping is maintained by the conversion, by forcing all the features in a logic block to have the same feature coding, then graphic differences between features within the block will be lost, and also there is no guaranteeing that this feature coding will be unique. Alternatively each separate logic block could be put into a separate IFF layer. However, this would mean the possible creation of an unmanageable number of IFF layers.

Therefore all output from TDST2I is written into one IFF layer with a feature code representing the graphical information as specified in the TDST file. It is unlikely that an existing FRT file will be able to cope with the information presented in a TDST file because of non-compatibility of symbol numbers or line types etc. between the two formats. Also, every combination of all the attributes would have to be anticipated. The coding is therefore controlled by the conversion program to comply with a standard FRT table: LSL$FRT:TDST.FRT. It will ensure that all features having the same attribute information as each other in the TDST file will have the same feature code in the IFF file. However, it will not, for example, ensure that all red lines in a TDST file will appear as red when converted to IFF because there is no way of knowing what symbol numbers or line types etc. refer to in the TDST definition. If the colour or the symbology of the features is particularly important then some post-conversion editing will have to be done using LITES2 to restore the features to their intended appearance.

CHAPTER 2

TDST DATA PREPARATION

DATA PREPARATION

In order to facilitate a flexible transfer of data between TDST and IFF formats, a lookup table may be used. This determines how TDST entities are translated into IFF feature codes.

The TDST2I parameter file

The parameter file contains one lookup table :-

Each line of the table is prefixed with a command which identifies the entity to which the IFF feature code corresponds. The lines are free-format, so the order of entries matters, but the actual position of entries on the line is not important. Commands may be in upper or lower case.

A maximum of 1000 entries will be allowed per entity type per parameter file.

Any text starting with the "!" character is regarded as a comment, and is ignored. Comments can appear on the same lines as commands, or on separate lines.

There must always be a FRT file given with the parameter file to be able to determine the graphical type of the requested feature code, so that the IFF feature entries and values may be set up correctly.

If the FC could not be found in the FRT file, a warning message will be given and certain default graphical types will be used for the particular entity: GT2 (clockwise arc) or GT 3 (anticlockwise arc) for ARC entities, depending on whether the angle difference is negative (GT2) or positive (GT3); GT 1 (linear) for POLY entities; GT 8 (oriented symbol) for POINT entities and GT 10 (text) for TEXT entities.

If the FC is found in the FRT file, its graphical type is checked against the allowed GT's for the particular entity type, and if the GT is not allowed, the same default graphical types as if the FC were not in the FRT file (above) will be imposed. The allowed graphical types for each entity type are listed below in the mapping restrictions below.

The parameter table format

The parameter file determines the translation from TDST entities into IFF features. The entity types allowed for conversion to IFF are ARC, POINT, POLYGON and TEXT, and only these are allowed in the parameter file. AREA, OBJECT and MULTIPOINT entities are ignored by TDST2I, and just read through. It is assumed that the component features of AREA and OBJECT entities are repeated elsewhere in the TDST file under their respective entity types. There is no IFF equivalent to the MULTIPOINT entity.

Each line is prefixed with the entity type command, which must be one of ARC, POINT, POLY or TEXT as illustrated below.

A line has one of the following forms :-

```
!
!
! Entity type      colour             style         IFF feature code
!
ARC             colour number       line type          FC
POINT           colour number       symbol number      FC
POLY            colour number       line type          FC
TEXT            colour number       text inclination   FC
```

where the fields are :-

line type - the line type number in the TDST file, corresponding to the IFF pattern index in the FRT file matching the parameter file.

symbol number - the TDST symbol number, coresponding to the IFF symbol FC in the FRT (and SRI files) associated with the parameter file.

inclination index - the TDST inclination (equivalent to text font) corresponding to the IFF text FC in the FRT (and TRI files) associated with the parameter file.

FC - the IFF feature code.

All numerical entries must be non-negative, and all fields must be present.

Text heights will be written to IFF TH entries (in mm./100) from the TDST text height field, (ENABLE HEIGHT will be required in LITES2).

Mapping restrictions between TDST entities and IFF features

The mapping between TDST entities and IFF features is restricted by the nature of the TDST entity types and their associated IFF graphical types. The following is a list of TDST entity types that may correspond to IFF grapical types.

IFF Graphical Types	TDST Entities
1 = LINEAR	POLY
2 = CLOCKWISE ARC	ARC
3 = ANTICLOCKWISE ARC	ARC
4 = CIRCUMCIRCLE ARC	ARC
5 = FULL CIRCUMCIRCLE	ARC
6 = INTERPOLATED CURVE	POLY
7 = UNORIENTED SYMBOL	POINT
8 = ORIENTED SYMBOL	POINT
9 = SCALED SYMBOL	POINT

10 = TEXT	TEXT
11 = SYMBOL STRING	POLY
12 = FILL AREA	POLY

The following is the summary of the mapping restrictions from TDST entity types to IFF graphical types :-

TDST Entities	**IFF Features (GT)**
ARC	CLOCKWISE ARC (2)
	ANTICLOCKWISE ARC (3)
	CIRCUMCIRCLE ARC (4)
	FULL CIRCUMCIRCLE (5)
POINT	UNORIENTED SYMBOL (7)
	ORIENTED SYMBOL (8)
	SCALED SYMBOL (9)
POLY	LINEAR (1)
	INTERPOLATED CURVE (6)
	SYMBOL STRING (11)
	FILL AREA (12)
TEXT	TEXT (10)

Reading of parameter file

TDST2I utilises the library, DSTPARLIB, for reading the parameter file. Messages output from this library are prefixed by '%DSTPAR_'. These messages are documented here.

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.

ERRFCGT, FC 'integer' has wrong GT 'integer' in FRT file for 'string' entity

Explanation: The graphical type (GT) from the FRT file for the indicated feature code (FC) in the parameter file is inappropriate for the associated entity. The program will use default graphical types dependent on the entity type.

User action: Check the FC in the entity entry in the parameter file, and the specification of the FRT file.

ERRFNDFC, error finding FC 'number' in FRT file

Explanation: Error encountered when looking for IFF feature code (FC), read from the parameter file, in the FRT file. This error may be caused either by entering an incorrect FC entry in the parameter file, or by specifying an incorrect FRT file. The program will use default graphical types dependent on the entity type.

User action: Check the FC in the entity entry in the parameter file, and the specification of the FRT file.

UNKPRICMD, unknown primary command parameter at line 'number'

Explanation: The primary command read from the parameter file is not recognised. Valid primary parameter entries are ARC, POINT, POLY and TEXT. The program will continue, ignoring the current line.

User action: Check the parameter file.

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.

ERROPNFRT, error reading FRT file 'filename'

Explanation: There was an error reading the FRT file.

User action: Check the FRT file and correct the error.

ERROPNPAR, error opening parameter file: 'filename'

Explanation: The parameter file could not be opened. The program will not be able to continue upon the detection of this error.

User action: Check the existence and status of the parameter file.

ERRRDPAR, error reading parameter file at line 'number'

Explanation: There was an error reading the parameter file at the indicated line number. The program may not be able to continue upon the detection of this error, or may give unexpected results.

User action: Check the parameter file at the indicated location.

NOPARFC, no valid entries in parameter file

Explanation: There were no valid entity or FC entries in the given parameter file.

User action: Check the parameter file and correct the errors.

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 3

I2TDST UTILITY

UTILITY I2TDST

FUNCTION

I2TDST is an IFF to SysScan DST ASCII format converter. It produces a TDST file composed of POLYGON, ARC, TEXT and POINT features. Output is directed to disc file.

FORMAT

$ I2TDST input-IFF-file-spec output-TDST-file-spec

Command qualifiers	**Defaults**
/FRT = file-spec	LSL$FRT:FRT.FRT
/BNT = file-spec	LSL$BNT:BNT.BNT

PROMPTS

Input IFF filename:	input-IFF-file-spec
Output TDST filename:	output-TDST-file-spec

PARAMETERS

input-IFF-file-spec

- specifies the IFF file which is to be converted into TDST format. Any part of the file name which is not specified will be taken from the default specification 'LSL$IF:IFF.IFF'.

output-TDST-file-spec

- specifies the TDST file which is to be created. Any part of the file name which is not explicitly given will be taken from the default specification 'LSL$DST:TDST.TDST'.

COMMAND QUALIFIERS

/FRT=file-spec

- specifies the FRT file which is to be used to describe the feature codes within the IFF file. It will be the same FRT file that was used when the file was created, or when it was edited with LITES2. Any part of the file name which is not explicitly given will be taken from the default specification 'LSL$FRT:FRT.FRT'. This qualifier is mandatory.

/BNT=file-spec

- specifies the BNT (Block-Name Table) file which is to be used to give a name to each logic block. The BNT file is a text file with format as described below. Any part of the file name which is not explicitly given will be taken from the default specification 'LSL$BNT:BNT.BNT'. This qualifier is mandatory.

DESCRIPTION

I2TDST converts IFF files into TDST format. The output is compatible with SysScan program LOADST which converts data from TDST format into their DST format used by their utilities. It forms part of an exchange flowline between Laser-Scan and SysScan formats. The opposite path from DST to IFF is carried out by SysScan's UNLDST and TDST2I which is the complement of I2TDST.

I2TDST expects all IFF features with the same feature code to be contiguous within the IFF file. The Laser-Scan IMP utility ISORT/FC must therefore be used as the first stage in the IFF to TDST conversion process. ISORT/FC groups together IFF features with the same code within a layer, and so, to achieve contiguous feature codes, the input IFF file should not contain features with the same code in more than one layer.

I2TDST and logical names

I2TDST requires that the logical name LSL$IF: should point to the directory containing the IFF file which is to be converted into TDST format. It also requires that the logical name LSL$DST: be set up to point to a directory to which the TDST file is to be written. It also requires that logical name LSL$BNT: be set up to point to a directory to which the BNT file is to be read. Finally, logical name LSL$FRT: should be set up to point to the directory which holds the required FRT file. Any of these logical names may be overridden by explicit specification of the directory concerned. This is not recommended as the location of files will quickly become confusing.

The BNT file format

The BNT file format permits the specification of a logic block tree structure of unlimited extent, such that a logic block may have either the MASTER block or any other logic block as a parent.

The format is as follows:

```
! All blank lines and everything following '!' on a line will be ignored
!
! IFF feature code = <integer 0:32767>
! Logic block name = <alphanumeric name up to 10 characters long>
!
! IFF feature code      Logic block name
! ----------------      ----------------
feature code            Logic block name
feature code            Logic block name
...                     ...
BEGIN
        feature code            Logic block name
        ...                     ...
        BEGIN
                feature code            Logic block name
                ...                     ...
                ...                     ...
        END
        feature code            Logic block name
        ...                     ...
        ...                     ...
END
feature code            Logic block name
...                     ...
...                     ...
BEGIN
        feature code            Logic block name
        ...                     ...
        ...                     ...
END
feature code            Logic block name
...                     ...
...                     ...
```

The IFF feature code will be used for the user-ID. Any feature whose feature code is not in the BNT file, or whose block name is missing, will not appear in the TDST file.

All occurrences of duplicate feature codes will be ignored.

By simply specifying feature codes with their corresponding logic block name, i.e. without the BEGIN-END blocks, all logic blocks will have the unique MASTER block as parent.

An example follows:

```
! level 1       2       3       4
! ---------------------------------------------------------------
        10      one
        begin
                30      thirty
                50      fifty
        end

        40      forty
        begin
                60      sixty
                begin
                        70      seventy
                end
                80      eighty
                20      twenty
```

EXAMPLES

```
$ ISORT/FC CONTOURS CONTOURS.TMP
 ELAPSED: 00:01:57.24  CPU: 0:01:22.45  BUFIO: 32  DIRIO: 156  FAULTS: 123
$ I2TDST CONTOURS.TMP ATDST/FRT=TDSTFRT/BNT=TDSTBNT
 ELAPSED: 00:01:32.06  CPU: 0:00:53.65  BUFIO: 29  DIRIO: 126  FAULTS: 108
$
```

This example demonstrates the use of I2TDST. The IFF file LSL$IF:CONTOURS.IFF has been successfully processed and a TDST format disk file LSL$TDST:ATDST.TDST has been produced. The BNT file LSL$BNT:TDSTBNT.BNT was used to name the TDST logic blocks. The FRT file LSL$FRT:TDSTFRT.FRT was used to interpret the contents of the IFF file.

Note that ISORT must always be used to sort IFF features by feature code before I2TDST can be used successfully.

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.

BADFCLAYER, Feature FSN='integer' occurs in more than one layer

Explanation: I2TDST will not work unless all features have previously been sorted into order by feature code. ISORT/FC will only order feature codes within layer, and so features with the same feature code should not appear in more than one layer.

User action: Ensure features with the same feature code do not appear in more than one layer before using ISORT/FC to sort features by FC.

BADFCORDER, Feature FSN='integer' is not in order by feature code

Explanation: I2TDST will not work unless all features have previously been sorted into order by feature code, using ISORT/FC, so that all features within a layer with equal feature code are contiguous.

User action: Use ISORT/FC to sort features by FC before running I2TDST.

EMPTFEAT, Empty feature 'integer' has no coordinates

Explanation: No locating information was available with the feature, therefore it is invalid.

User action: Edit the IFF file to find the source of the error.

FAIFINFC, FC 'integer' in feature 'integer' not found in FRT

Explanation: The FC value specified does not exist in the FRT table.

User action: Check that the specified FRT table is the correct one for dealing with this IFF file.

INVALARC, Invalid arc - points cannot represent an arc in feature 'integer'

Explanation: The points given to represent the circle cannot feasibly do so, because e.g. they are collinear.

User action: Respecify the feature in an editing session or edit a text version of the IFF file to give the correct coordinates.

NOEFENT, No EF entry found in feature 'integer' - one assumed

Explanation: The current feature wasn't properly closed before the next feature was opened. An EF entry has therefore been presumed and the entry processed as well as possible.

User action: None.

NOFSENT, No FS entry found in feature 'integer', feature aborted

Explanation: The FS entry of the feature in hand was not correctly located immediately after the NF entry.

User action: Check the IFF file by using IMP utility ITOTEXT.

NONEXIGT, Non-existent GT 'integer' found in FC 'integer' in feature 'integer'

Explanation: The GT specified does not exist.

User action: Check the definition of the relevant FC in the specified FRT table.

NONTEXTS, TS entry found in non-text feature 'integer'

Explanation: TS entries should only occur in text features - in this instance it has been found in a non-text feature.

User action: Inspect the IFF file to resolve the conflict of entries.

NOROTDAT, No rotation data in oriented or scaled symbol 'integer'

Explanation: The current feature is an oriented or scaled symbol which needs rotation information either as an RO entry or by a second defining point. Neither of these have been found. A default rotation of 0 has been applied.

User action: Either amend the original feature to contain rotation information or edit the final TDST file if a rotation of 0 is unacceptable.

NOROTTEX, No rotation value in text feature 'integer' - default applied.

Explanation: Text features should normally have a rotation value associated with them. This feature does not have, so a default of 0 degrees has been applied.

User action: Amend in Lites2 or edit a text version of the IFF file.

NOTXTENT, No text entry in feature 'integer'

Explanation: The specified feature has all of the correct entries except for the text itself.

User action: Insert a text value into a text version of the file.

TXTTOOLON, Text is too long in feature 'integer'

Explanation: TDST files have a maximum text string length of 40 characters. This has been exceeded in this feature.

User action: The text feature should be split into more than one text feature.

UNXENTTYP, Unexpected entry 'code' in feature 'integer'

Explanation: An unknown or unexpected feature type has been found in the IFF file. This message may sometimes be superfluous when an entry is not expected but still valid.

User action: Decide whether the message is valid or not, and if it is edit the IFF file to remove the offending entry.

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.

BADBNTLIN, Bad format in BNT file, line 'integer'

Explanation: Illegal format on specified line in BNT file.

User action: Use text editor to correct BNT file.

NOPARENT, Missing parent block before BNT line 'integer'

Explanation: BNT file contains a BEGIN line on the line indicated for which there is no corresponding parent logic block. This BEGIN line will be ignored.

User action: Correct the BNT file.

NORANENT, No range entry in 'file-spec'

Explanation: The conversion has encountered a feature before the range has been declared.

User action: Amend the source IFF file to contain suitable range values.

TOOMNYLEVS, Too many levels in BNT tree, line 'integer' (max is 'integer')

Explanation: The BNT file contains a tree structure which has more than the maximum permitted number of levels.

User action: Reduce the number of levels in the BNT file tree.

TOOMNYTXCOR, Too many text coordinates in feature 'integer'

Explanation: Text features should only have one locating point.

User action: Amend the IFF file and rerun the program.

UNACREDST, Unable to create DST file 'file-spec'

Explanation: It was not possible to open the DST file as specified. Further information should be obtainable from the adjoining system message.

User action: Repeat the specification after taking notice of the LSLLIB error messages.

UNAOPNBNT, Unable to open specified BNT file 'file-spec'

Explanation: The file that you specified does not exist.

User action: Check the spelling and path name of the file as specified, and check that LSL$BNT points to where you expect it to.

UNAOPNFRT, Unable to open specified FRT file 'file-spec'

Explanation: The file that was specified does not exist.

User action: Check the spelling and path name of the file as specified, also check that LSL$FRT points to where you expect it to.

UNAOPNIFF, Unable to open specified IFF file 'file-spec'

Explanation: The file that was specified does not exist.

User action: Check the spelling and path name of the file as specified, also check that LSL$IF points to where you expect it to.

UNAREADBNT, Unable to read /BNT qualifier

Explanation: Unable to read the qualifier /BNT filename from the user command line.

User action: Repeat the specification after taking notice of the LSLLIB error messages.

UNAREADCMD, Unable to read command line

Explanation: Unable to read the user command line.

User action: Repeat the specification after taking notice of the LSLLIB error messages.

UNAREADDST, Unable to read DST filename

Explanation: Unable to read the output DST filename from the user command line.

User action: Repeat the specification after taking notice of the LSLLIB error messages.

UNAREADFRT, Unable to read /FRT qualifier

Explanation: Unable to read the qualifier /FRT filename from the user command line.

User action: Repeat the specification after taking notice of the LSLLIB error messages.

UNAREADIFF, Unable to read IFF filename

Explanation: Unable to read the input IFF filename from the user command line.

User action: Repeat the specification after taking notice of the LSLLIB error messages.

UNASELWRI, Unable to select write file 'file-spec'

Explanation: The specified file could not be assigned for write.

User action: Check the LUN specified matches the one relating to the intended file.

UNBALNCDTREE, Too many END lines in BNT file, at line 'integer'

Explanation: BNT file contains an END line on the line indicated for which there is no corresponding BEGIN line. This END line will be ignored.

User action: Correct the BNT file.

UNEXPEOF, Unexpected end-of-file 'file-spec'

Explanation: The IFF file was improperly closed - no EJ entry could be found.

User action: Mend the file using IMP utility IMEND and rerun I2TDST.

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 4

TDST2I UTILITY

UTILITY TDST2I

FUNCTION

TDST2I is a SysScan DST ASCII format to IFF converter. It produces a standard IFF disc file.

FORMAT

$ TDST2I input-TDST-file-spec output-IFF-file-spec

Command qualifiers	**Defaults**
/FRT = file-spec	/NOFRT
/OBJECT_AC = integer	/NOOBJECT_AC
/PARAMETER_FILE = file-spec	/NOPARAMETER_FILE

PROMPTS

Input TDST filename : input-TDST-file-spec
Output IFF filename : output-IFF-file-spec

PARAMETERS

input-TDST-file-spec

- specifies the TDST file which is to be converted into IFF format. Any part of the file name which is not specified will be taken from the default specification 'LSL$DST:TDST.TDST'. Only one filename may be specified for each run of the program.

output-IFF-file-spec

- specifies the IFF file which is to be created. Only one filename may be specified for each run of the program. Any part of the file name which is not explicitly given will be taken from the default specification 'LSL$IF:IFF.IFF'.

COMMAND QUALIFIERS

/FRT [= file-spec]

- This qualifier is required if a parameter file is supplied with the /PARAMETER_FILE qualifier to define a TDST to IFF feature code translation, and should not be given if no parameter file is used. It is required to determine the graphical type of the requested feature codes to correctly set the IFF entries and values from the TDST entries.

  Any parts of the file-spec not supplied with this qualifier will be taken from the default of 'LSL$FRT:TDST.FRT'.

/OBJECT_AC [= integer]

- If this qualifier is given, the names of the objects or areas to which feature elements belong are output to the text field of an AC entry, the AC type being the number given with this qualifier. If no number is specified with the qualifier a default AC type 1 will be used.

/PARAMETER_FILE [= file-spec]

- If this qualifier is given, a user definable translation from TDST entity, colour number and line style/text inclination/symbol number to IFF feature code may be specified.

  If there is no /PARAMETER_FILE qualifier given, the feature codes will be derived in a simple combination of colour number and pattern number.

  Any parts of the file-spec not supplied with this qualifier will be taken from the default of 'LSL$LOOKUP:TDST.PAR'.

DESCRIPTION

Command line

The symbol TDST2I is normally set up as:

```
TDST2I == "$lsl$exe:tdst2i"
```

and the program may then be used as if it were a normal VMS utility.

TDST2I converts TDST files into IFF format. The input is compatible with the Syscan program UNLDST which converts data from the DST format used by their utilities. It forms part of an exchange flowline between Laser-Scan and SysScan formats. The reverse path from IFF to DST is carried out by SysScan's LOADST, and I2TDST which is the complement of TDST2I.

The feature codes used for the output IFF file are derived from a combination of the TDST style code and colour code for each feature, unless a parameter file is supplied. Not using a parameter file may result in features of different graphical types, or different entity types, being given the same FC.

The parameter file allows a user definable translation from TDST entity, colour number and line style/text inclination/symbol number to IFF feature code. The line types, symbol numbers and text inclinations should map onto line patterns, symbols and text fonts defined in the user's FRT, SRI and TRI files.

There are four DST entity types supported: ARC, POINT, POLY and TEXT. Processing of AREA, MULTIPOINT and OBJECT entities in the input DST file is not supported, and these entities are simply read through.

The parameter file should contain for the supported entity types:

```
! DST Entity   Colour number  Line style         Feature code
!
     ARC          Colour      Line type             FC
     POINT        Colour      Symbol number         FC
     POLY         Colour      Line type             FC
     TEXT         Colour      Text inclination      FC
```

If a parameter file is given, only the supplied colour and pattern combinations for the given entities will be output to the IFF file, and any not matching will be ignored. In this way, for example, all red features in the DST file may be omitted from the translation to IFF.

If the DST range is inconsistent with the coordinate values encountered in the main body of the DST file, (ie. there were coordinate values encountered outside the DST range), a warning message is given. The correct range is written to the RA entry at the end of the program run.

All OBJECT and AREA entity fields are read through and ignored, as are MULTIPOINT features as there is no satisfactory way of conversion to IFF. The feature elements comprising an OBJECT or AREA are assumed to be repeated in their respective entities elsewhere in the DST file. It is left to the user to join AREA feature elements into the total area feature.

The names of the OBJECTS or AREAS to which an entity belongs can be output to the text field of an AC entry for the IFF feature if the /OBJECT_AC=ac_type is given.

TDST2I and logical names

TDST2I requires that logical name LSL$DST: should point to the directory containing the TDST file which is to be converted into IFF format. It also requires that logical name LSL$IF: be set up to point to a directory to which the IFF file is to be written. Either of these two logical names may be overridden by explicit specification of the directory concerned. This is not recommended as the location of files will quickly become confusing.

EXAMPLES

```
$ TDST2I/OBJECT_AC=2 CONTOURS CONTOURS2
 ELAPSED: 00:01:32.06  CPU: 0:00:53.65  BUFIO: 29  DIRIO: 126  FAULTS: 108
$
```

This example demonstrates the use of TDST2I. The TDST file LSL$DST:CONTOURS.DST has been successfully processed and an IFF format disk file LSL$IF:CONTOURS2.IFF has been produced. All object names will have been output to the text fields of AC type 2 entries.

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.

CIRCANGERR, ARC data angle inconsistent with full circle at line 'integer'

Explanation: The parameter file has specified that this ARC feature should be a full circle, but the incremental angle given at the line shown is not 360 degrees. This angle will be ignored.

User action: Check the FRT file or parameter file for consistency.

ENTIGNORE, DST entity ignored at line 'integer'

Explanation: The combination of colour and line pattern, symbol number or text inclination at the line shown was not found in the supplied parameter file for the current entity type, and will be omitted from the translation to IFF.

User action: Check the parameter file if this an error.

RANGEWRONG, DST Range inconsistent with coordinate data

Explanation: The range information in the header of the DST file does not agree with the coordinates in the main body of the file.

User action: None; the IFF range will be calculated from the coordinate information.

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.

ERRINTLIN, Error interpreting contents of DST file on line 'integer'

Explanation: An error occured whilst trying to read the contents of TXTBUF. This could be due to the presence of the wrong type of variable e.g. encountering a character string when trying to read an integer.

User action: Inspect the ASCII DST file around the indicated line and look for any deviations from the expected format.

NOLOGSEC, Logic section not found at line 'integer'

Explanation: A relevant start to a new section could not be found in the DST file where it was expected the file is therefore corrupt.

User action: Edit the source DST file at the indicated line and amend to a correct format.

NOMASEOS, End of master section not found

Explanation: Initial scanning of the input file could not find the obligatory EOS entry in the master section of the master section.

User action: Check the DST file and add an EOS entry into the master section.

NOMASSEC, Master section not found

Explanation: Initial scanning of the input file could not find the obligatory MASTER section of the file.

User action: Check the DST file and amend with a dummy master record if it appears to be missing.

UNACREIFF, Unable to create specified IFF file 'file-spec'

Explanation: It was not posible to open the IFF file as specified. Further information should be obtainable from the adjoining system message.

User action: Repeat the specification after taking notice of the IFFLIB error messages.

UNAOPNDST, Unable to open DST file 'file-spec'

Explanation: The DST file specified does not exist, or could not be opened.

User action: Check the spelling and path name of the file as specified, also check that LSL$DST points to where you expect it to.

UNKGEOIDN, Unknown geometry type at line 'integer'

Explanation: The program has encountered a geometry type definition that is unknown to it.

User action: Edit the DST source file at the indicated line and correct any obvious errors (such as incorrect geometry name).

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.