SET User Guide

CADDS® 5 Revision 6.1.2

DOC38108-005

Parametric Technology Corporation

Parametric Technology Corporation, 140 Kendrick Street, Needham, MA 02494-2714
8 January 2001

Table of Contents

Preface

Introducing the SET Processor

The PUT SET Command

The GET SET Command

Using The CONVENTION File

Using the LIBRARY Function

Supported Entities

System Messages

Preface

The *SET User Guide* contains information about how to translate geometric data to and from the Standard Exchange and Transfer (SET) and CADDS format. This book provides instructions for use in the CADDS environment.

This book is for users who have working knowledge of their specific application environment and UNIX. Users should also be familiar with the contents and scope of the SET specification.

This manual supports Norm Z68-300 (AFNOR-June 89) of SET as it applies to CADDS.

Book Conventions

The following table illustrates and explains conventions used in writing about Optegra applications.

Convention	Example	Explanation
EPD_HOME	`cd` $EPD_HOME`/install` (UNIX) `cd` %EPD_HOME%`\install` (Windows)	Represents the default path where the current version of the product is installed.
Menu selections	Vault > Check Out > Lock	Indicates a command that you can choose from a menu.
Command buttons and options	Mandatory check box, Add button, Description text box	Names selectable items from dialog boxes: options, buttons, toggles, text boxes, and switches.

User input and code	`Wheel_Assy_details`
	
	`-xvf /dev/rst0`
	
	`Enter command>` **`plot_config`**
		Enter the text in a text box or on a command line.
		
		Where system output and user input are mixed, user input is in bold.

System output	`CT_struct.aename`	Indicates system responses.
Parameter and variable names	`tar -cvf /dev/rst0` filename	Supply an appropriate substitute for each parameter or variable; for example, replace filename with an actual file name.
Commands and keywords	The `ciaddobj` command creates an instance of a binder.	Shows command syntax.
Text string	`"SRFGROUPA"` or `'SRFGROUPA'`	Shows text strings. Enclose text strings with single or double quotation marks.
Integer	n	Supply an integer for *n*.
Real number	x	Supply a real number for *x*.
#	`# mkdir /cdrom`	Indicates the root (superuser) prompt on command lines.
%	`%` `rlogin` remote_system_name `-l root`	Indicates the C shell prompt on command lines.
$	`$` `rlogin` remote_system_name `-l root`	Indicates the Bourne shell prompt on command lines.
>	`> copy` filename	Indicates the MS-DOS prompt on command lines.
Keystrokes	`Return` or `Control-g`	Indicates the keys to press on a keyboard.

Online User Documentation

Online documentation for each Optegra book is provided in HTML if the documentation CD-ROM is installed. You can view the online documentation from an HTML browser or from the HELP command.

You can also view the online documentation directly from the CD-ROM without installing it.

From an HTML Browser:

1. Navigate to the directory where the documents are installed. For example,

   $EPD_HOME/data/html/htmldoc/ (UNIX)

   %EPD_HOME%\data\html\htmldoc\ (Windows NT)
2. Click mainmenu.html. A list of available Optegra documentation appears.
3. Click the book title you want to view.

From the HELP Command:

To view the online documentation for your specific application, click HELP. (Consult the documentation specific to your application for more information.)

From the Documentation CD-ROM:

1. Mount the documentation CD-ROM.
2. Point your browser to:

   CDROM_mount_point/htmldoc/mainmenu.html (UNIX)

   CDROM_Drive:\htmldoc\mainmenu.html (Windows NT)

Printing Documentation

A PDF (Portable Document Format) file is included on the CD-ROM for each online book. See the first page of each online book for the document number referenced in the PDF file name. Check with your system administrator if you need more information.

You must have Acrobat Reader installed to view and print PDF files.

The default documentation directories are:

- $EPD_HOME/data/html/pdf/doc_number.pdf (UNIX)
- %EPD_HOME%\data\html\pdf\doc_number.pdf (Windows NT)

Resources and Services

For resources and services to help you with PTC (Parametric Technology Corporation) software products, see the *PTC Customer Service Guide*. It includes instructions for using the World Wide Web or fax transmissions for customer support.

Documentation Comments

PTC welcomes your suggestions and comments. You can send feedback in the following ways:

- Send comments electronically to `doc-webhelp@ptc.com`.
- Fill out and mail the PTC Documentation Survey located in the *PTC Customer Service Guide.*

Chapter 1

Introducing the SET Processor

This chapter gives a brief history of the SET processor and introduces the entities processed by SET. There are also notes on working with SET data.

- The SET Processorr
- Entities Supported by the SET Processor
- Special Considerations for SET Data
- Special Considerations for Solaris and SunOS

The SET Processor

The SET processor translates geometric data to and from the Standard Exchange and Transfer (SET) format. The data can be transferred to and from non-PTC systems that also support the SET format.

Brief History of the SET Specification

The SET specifications study was done by AFNOR (French Association of Normalization) from a project started by AEROSPATIALE, beginning 1983.

The PTC SET processor supports Norm Z68-300 of SET (AFNOR-June 1989). The basic document, *Data Exchange and Transfer Standard Specification* Z68-300, Version 89-06, contains the complete SET standard format.

SET Conversion Process

The SET conversion process consists of two complementary commands:

PUT SET Uses a CADDS part to create a text file in SET format

GET SET Uses a SET text file to create a CADDS part

Additional Functions

There are two additional functions you can use to specify how data is translated:

CONVENTION Specifies how data is exchanged. This is a document that sets out agreed exchange criteria between several parties.

LIBRARY Defines a list of libraries for use in translating entities.

Entities Supported by the SET Processor

This version of the SET processor processes these entities:

- NURBS curves
- NURBS surfaces
- SOLIDS
- TRIMMED SURFACES
- Constructive solid geometry (CSG)
    - constructive solid
    - constructive geometric point
    - constructive curve
    - constructive surface
    - constructive face

Processing SOLIDS

SOLIDS are usually represented in geometry by the following four methods:

- Primitives
- Boolean operations (CSG), or transformations
- Exact boundaries (Exact BREP)
- Polygons (Approached BREP)

The following SOLIDS entities are also processed:

- All primitive solid geometry
- Topological exact representation
- Polygon representation
- Obtained by a transformation operation or Boolean operation of one of the solids (CSG) described above.

In the CADDS database, SOLIDS and TRIMMED SURFACE entities are defined according to the exact BREP.

Special Considerations for SET Data

Pay special attention to the following when working with SET data.

- SET files in directories with only root permission
- Single-precision and double-precision parts

These are described below.

SET Files in Directories with Only Root Permission

Do not create a SET file in a directory with only root permission. This makes the file unusable by GET SET inside CADDS.

Single-precision and Double-precision Parts

If you create a part using single-precision (PFORMAT SINGLE), you should convert it to SET and then read it into double-precision (PFORMAT DOUBLE) parts. If you do not follow this procedure, unpredictable results may occur.

Special Considerations for Solaris and SunOS

This section gives a brief overview of the special considerations for Solaris and SunOS.

Primitive Solid: @50 SET block type

GET SET treats a new kind of SET primitive Solid:

- Geometric bounded sphere 1st type (#34 SET sub block type)
- Geometric bounded sphere 2nd type (#35 SET sub block type)
- Geometric bounded torus (#36 SET sub block type)
- Generation of parallel thick surface (#140 SET sub block type)
- Generation of translated thick surface (#141 SET sub block type)
- Solid of revolution (#142 SET sub block type)

Constructed Solid (CSG)

```
Treatment @100 SET block type
```

Constructed Entity

```
treatment @110 SET block type
treatment @111 SET block type
treatment @112 SET block type
treatment @113 SET block type
```

Convention

GET SET and PUT SET in the CADDS interface can transfer the sets by a convention file. For example, color table, character font and line font can be transferred by a convention file.

Library

In one SET file GET SET can exchange several assemblies which constitute a Library. PUT SET can handle a directory which contains several CADDS parts.

Repair Surface

When GET SET in CADDS5 interface cannot repair a surface, the new Repair Surface action cuts the face at the discontinuous parameters on several faces and sews the different one on one Tsurface.

Chapter 2

The PUT SET Command

This chapter contains the information you need to convert a CADDS part to a SET text file using the PUT SET command.

The PUT SET command uses a CADDS part to create an 80-column SET text file.

You also need a separate parameter file, containing information to initialize the SET file.

The PUT SET command is valid only outside a part.

- Using PUT SET with a Parameter File
- Contents of the PUT SET Parameter File
- Using Annotations
- Sample PUT SET Parameter File
- Example Using PUT SET
- Explanation of the PUT SET Example

Using PUT SET with a Parameter File

The parameter file `psetparams` is required for PUT SET to execute. There is a default parameter file in `/usr/apl/cadds/data/psetparams`.

Using the Default Parameter File

If you use the default parameter file, or if you modify the file but do not move it to another directory, no special preparation is necessary for the program to locate the parameter file and execute PUT SET.

Modifying the Parameter File

You can modify the parameter file using an editor, for example, `vi`.

If you copy the parameter file to another directory and then modify the copy, you must be sure that:

- The copy of the parameter file is named `psetparams`.
- The CADDSPATH environment variable in your `.caddsrc` file contains the pathname of the new parameter file.
- The pathname to the new parameter file is listed before the `/usr/apl/cadds/data` directory in your search list.

For more information about the `.caddsrc` file and CADDSPATH, see the *Explicit Modeling User Guide and Menu Reference.*

Contents of the PUT SET Parameter File

The parameter file contains the following items:

- SET identification
- Correspondence between user line fonts
- Correspondence between user text fonts
- Trace Flag
- Messages display
- Form draw name

These items are described below.

SET Identification

The sub-block 9900 (identification of the SET file) is generated from the following elements:

- Version number of the SET specification
- Version number of the SET interface software
- Name of the generating software
- Version number of the generating software
- User's identification, consisting of the names of the:
  - Company
  - Division
  - Department
  - Designer

Correspondence Between User Line Fonts

Shows how the font numbers in the line font definition file (`data/fontdefs`) of the CADDS part correspond to the fonts you will obtain in the SET file (parameter 53 of the dictionary).

Correspondence Between User Text Fonts

Shows how the PTC text font number in the CADDS part corresponds to the standard text font number you will obtain in the SET file (parameter 101 of the dictionary).

Message Display

If the DISPLAY.MESSAGES flag is set to `YES`, the SET processor displays a message for each error found during the processing of an entity. Messages are useful when debugging.

Trace Flag

If the trace flag (TRACE.FLAG) is set to `YES`, the SET processor produces a trace for the processing of each entity which displays the:

- Master index pointer (MIPTR)
- The CADDS entity type
- The sequential number of the corresponding SET block created in the SET file.

This allows you to trace the processing of each entity.

Form draw name

If your CADDS part includes only parametric geometry (no drawing and view in explicit environment), the SET processor automatically generates an explicit drawing for correct geometry translation.

The form name (refer to the *Design and Drafting User Guide and Menu Reference* chapter “Using Figures in Detailing” paragraph “Form Parts”) used for this operation is specified by user using psetparams file key: EXPLICIT.DRAW, the draw name in form must be: ‘`A`’. If the key is not specified the SET processor will use default: FORM.A4-VIEW.

Comment File

The flag COMMENT.FILE sets the name of the text file which contains the annotation for that part. For details regarding the use of annotations, refer to “Using Annotations” on page 2-5.

Using Annotations

The SET rule allows you to add annotations to the different CFAO systems using the #9902 sub-block. An annotation, according to the SET rule, is a sequence of alphanumeric characters. Every character has a value between 32 and 126 in ASCII code.

SET Annotations

The SET annotation is a permanent sequence of characters written into the #9902 sub-block.

This sub-block can be added under all the blocks except the @9997, @9998 and @9999 blocks.

ASCII characters not contained in the SET characters, will be written as blank characters.

Example

```
@9901,1,'TEST ANNOTATION'
#9902,'ZONE NORMES GENERALES'
#9902,'TOLERANCES NON SPECIFIES'
```

Using PUT SET Annotations

All annotation files placed under the part directory, are written in #9902 sub-block of the matching assembly.

You can set the name of the test file containing the annotation under the part, by setting the flag COMMENT.FILE in the `psetparams` file. If the COMMENT.FILE flag is set (COMMENT.FILE = comment) and if comment file exists under the part, the #9902 sub-block is written.

If the COMMENT.FILE flag is not set or if there is no text file under the part, CADDS-SET interface does not write the #9902 sub-block.

If the COMMENT.FILE flag is set (COMMENT.FILE = dircom.comment) and if comment file exists under the dircom directory, the #9902 sub-block is written.

CADDS-SET interface writes the #9902 sub-blocks only under the @9900 or @9901 blocks.

Limitations

- The test length is restricted to 246 characters.
- A #9902 sub-block must be written for every annotation line ended by line feed special character.

Sample PUT SET Parameter File

The following example shows `psetparams`, the default parameter file for PUT SET. Lines starting with an asterisk (*) are comments.

The psetparams File

```
*******************************************************
**
**           PARAMETER FILE FOR THE INTERFACE
**                    CADDS  =====> SET
**
**               CADDS COMMAND : #0N# PUT SET
**
*******************************************************
*
*  SET IDENTIFICATION
*  -----------------
*
SET.SPECIFICATION.VERSION = NF Z68-300
SET.SOFTWARE.INDICE = 3.3
GENERATING.SOFTWARE.NAME = cadds_revision_name
GENERATING.SOFTWARE.VERSION = cadds_revision_number
DIVISION = MCAD
DEPARTMENT = DEX
DESIGNER = Italcad
*
* USER LINE FONT CORRESPONDENCE
* ----------------------------
*
* USER LINE FONT NUMBER --> SET STANDARD LINE FONT NUMBER
* (DATA/FONTDEFS)          (DICTIONARY PARAMETER 53)
*                            0 = invisible line
*                            1 = continuous line
*                            2 = dashed line
*                            3 = long chain line
*                            4 = double-dashed line (phantom)
*                            5 = continuous free hand line
*                            6 = continuous line with zig-zags

USER.LINE.FONT 0 = 1
USER.LINE.FONT 1 = 2
USER.LINE.FONT 2 = 4
*
* USER TEXT FONT CORRESPONDENCE
* ----------------------------
* USER TEXT FONT NUMBER --> SET STANDARD TEXT FONT NUMBER
* (FONT)(DICTIONARY PARAMETER 101)
*     1 = standard
*     2 = ISO
*     3 = DIN
*
```

```
USER.TEXT.FONT 1 = 1
USER.TEXT.FONT 16 = 2
USER.TEXT.FONT 9 = 9

* TRACE FLAG
* -----------
TRACE = NO
*
* MESSAGES DISPLAY
* -----------------
DISPLAY.MESSAGES = NO
*
*  Convention File Name (in CADDS format).
*   ------------------------------------
*
* If present --> Use Convention table to exchange data.
*    (A convention File with this name
*     will be created or updated).
* If not present --> Use Correspondence tables.
CONVENTION.FILE = cv-conv
*
*  Library flag
*  ------------
* If flag is YES -->the PUT SET command will process all
* parts under a given directory into only one SET file.
*           NO --> only one part will be processed.
LIBRARY.FLAG = YES
*
* Form draw name
* --------------
*
* If present  --> Use given form name for explicit draw
* activation if the part to be converted doesn't include one
* (parametric only)
* If not present  --> Use default (FORM.A4-1VIEW)
*
EXPLICIT.DRAW = FORM.A4-1VIEW
COMMENT.FILE=comment
```

Example Using PUT SET

This example shows how you respond to sample system prompts.

```
#n# put set
*******************************************************
**                I.N.T.E.R.F.A.C.E.
**                CADDS ====> SET
**
**                   RELEASE 3.3
*******************************************************
ENTER CADDS PART NAME ? :
$ part.test.set

ENTER SET FILE NAME ? :
$ myset.file
BEGINNING OF THE PART PROCESS : part.test.set
------------------------------
NUMBER OF PROCESSED ENTITIES : 75 OUT OF 75
*******************************************************
**
**             END OF PROCESSING STATUS
**
*******************************************************
**  POINT            : 1 OUT OF 1 ==> 100.00 %
**  LINE             : 8 OUT OF 8 ==> 100.00 %
**  N-SPLINE         : 4 OUT OF 4 ==> 100.00 %
**  N-SPLINE SURFACE : 13 OUT OF 13 ==> 100.00 %
**  SOLID            : 2 OUT OF 2 ==> 100.00 %
**  FACE             : 7 OUT OF 7 ==> 100.00 %
**  EDGE             : 16 OUT OF 16 ==> 100.00 %
**  VERTEX           : 12 OUT OF 12 ==> 100.00 %
*******************************************************
**  TOTAL NUMBER OF PROCESSED ENTITIES :
**    75 OUT OF 75 ==> 100.00%
*******************************************************
S.E.T. CONVERSION OF THE PART: part.test.set
S.E.T. TEXT FILE OUTPUT FILED UNDER THE NAME:myset.file
#n#
```

Explanation of the PUT SET Example

The following dialog shows sample prompts and user responses followed by corresponding explanations.

Specifying the Part Name

Enter the name that identifies the part for conversion. If you press RETURN without specifying a name, the SET processor returns to the CADDS prompt (#*n*#).

```
ENTER CADDS PART NAME ? :
$ part.test.set
```

If there is no part with the name you entered, you see the prompt:

```
CADDS PART DOES NOT EXIST, TRY AGAIN.
ENTER CADDS PART NAME ? :
$
```

Specifying the SET File

Enter a file name for the SET file (without `&bcd` before the last level of the directory).

```
ENTER SET FILE NAME ? :
$ myset.file
```

If you press RETURN without specifying a name, you return to the CADDS prompt (#*n*#). If a file exists with the name you entered, you see this prompt:

```
FILE ALREADY EXISTS, TYPE OK TO OVERWRITE ? :
$
```

If you enter `ok`, the processor converts the CADDS part and overwrites the SET text file with the new information. If you press RETURN, you see the `ENTER SET FILE NAME` prompt again.

Processing the Part

The processor processes the CADDS part for conversion and the screen displays the number of entities to be processed out of total amount of entities.

```
BEGINNING OF THE PART PROCESS : part.test.set
NUMBER OF PROCESSED ENTITIES : 75 OUT OF 75
```

Displaying the Processing Status

The screen shows the processing status for each type of entity, and the number of entities converted in the SET text file.

```
*****************************************************
**
**             END OF PROCESSING STATUS
**
*****************************************************
**  POINT            : 1 OUT OF 1 ==> 100.00 %
**  LINE             : 8 OUT OF 8 ==> 100.00 %
**  N-SPLINE         : 4 OUT OF 4 ==> 100.00 %
**  N-SPLINE SURFACE : 13 OUT OF 13 ==> 100.00 %
**  SOLID            : 2 OUT OF 2 ==> 100.00 %
**  FACE             : 7 OUT OF 7 ==> 100.00 %
**  EDGE             : 16 OUT OF 16 ==> 100.00 %
**  VERTEX           : 12 OUT OF 12 ==> 100.00 %
*****************************************************
**  TOTAL NUMBER OF PROCESSED ENTITIES :
**    75 OUT OF 75 ==> 100.00%
*****************************************************
S.E.T. CONVERSION OF THE PART: part.test.set
S.E.T. TEXT FILE OUTPUT FILED UNDER THE NAME:myset.file
```

Chapter 3

The GET SET Command

This chapter contains the information you need to convert a SET format text file into a CADDS part. The GET SET command can only read formatted files consisting of 80 columns or less.

You also need a parameter file, containing information to initialize the SET file.

The GET SET command is valid within CADDS in a non-active part.

- Using GET SET with a Parameter File
- Contents of the GET SET Parameter File
- Sample GET SET Parameter File
- Example Using GET SET
- Transforming Entities
- Example of Transforming Entities
- Creating a New Text Font
- Using the REPAIR SURFACE Function in GET SET
- Using Annotations
- Sample Report

Using GET SET with a Parameter File

The parameter file `gsetparams` is required for GET SET to execute. There is a default parameter file in `/usr/apl/cadds/data/gsetparams`.

Using the Standard Parameter File

If you use the default parameter file, or if you modify the file but do not move it to another directory, no special preparation is necessary for the program to locate the parameter file and execute GET SET.

Modifying the Parameter File

You modify the parameter file using an editor, for example, `vi`.

If you copy the parameter file to another directory, then modify the copy, you must be sure that:

- The name of the copy of the parameter file is `gsetparams`.
- The CADDSPATH environment variable in your `.caddsrc` file contains the pathname of the new parameter file.
- The pathname to the new parameter file is listed before the `/usr/apl/cadds/data` directory in your search list.

For more information about the `.caddsrc` file and CVPATH, see the *Explicit Modeling User Guide and Menu Reference.*

Contents of the GET SET Parameter File

The parameter file contains the following items:

- Correspondence between SET standard text fonts
- Create text font
- Message display
- Trace level
- SET header display
- Repair surface
- Repair angle tolerance
- Repair layer
- Repair surface tolerance
- Cut layer
- Reject layer
- Comment file

These items are described in detail below:

Correspondence Between SET Standard Text Fonts

This shows how the standard text font numbers in the SET file (parameter 101 of the dictionary) correspond to the numbers for PTC user text fonts you see in the CADDS part.

Create Text Font

If the flag CREATE.TEXT.FONT is set to YES, the specific font description files (a maximum of four files) are created at the same level as the `_pd` file of the resulting part in GET SET. If more than four text fonts are described in the SET file, only the first four are created with the user-available fonts. These are:

- font1cx.F
- font1dx.F
- font1ex.F
- font1fx.F

See “Creating a New Text Font” on page 3-17 for details on creating a New text font.

Message Display

If the flag DISPLAY.MESSAGES is set to YES, the SET processor displays a message for each error found during the processing of a SET block. These messages are useful when debugging.

Trace Level

If the flag TRACE.LEVEL = `0`, the SET processor displays the following information in the CADDS window:

- Block number being processed
- Type
- Master index pointer (MIPTR) of the CADDS entity created
- Master index pointer of the CADDS Tsurf created
- Error type if processing has failed
- The block number of the surface repaired or rejected
  - The master index pointer (MIPTR)
  - The tag of this surface
  - The maximum deviation between this surface and the original surface

This information enables you to trace the processing of each SET file block.

SET Header Display

If the flag DISPLAY.SET.HEADER is set to YES, the SET processor prints the header of the SET file being processed.

Repair Surface

If the flag REPAIR.SURFACE is set to YES, the surface repair action is used. If REPAIR.SURFACE is set to NO, the repair surface action is not used. In this case, only smooth surfaces are generated inside CADDS.

Repair Angle Tolerance

The flag REPAIR.ANGLE.TOLERANCE sets the tolerance angle between the right tangent and the left tangent at the discontinuity point of the underlying surface of a face.

If the angle between these two tangents is smaller than 0.1 degree the surface is only reparameterized (not repaired).

If the angle between these two tangents is greater than 0.1 degree and smaller than REPAIR.ANGLE.TOLERANCE, the surface is reparameterized and repaired (removing knots at the discontinuous parameter).

If the angle between these two tangents is more than REPAIR.ANGLE.TOLERANCE, the original face is cut at the discontinuous parameters into several faces and then sewn into one trimmed surface and put in CUT.LAYER. If the trimming operation fails, the original underlying surface and the trimming curves are put on REJECT.LAYER if REPAIR.SURFACE=YES.

Repair Layer

The flag REPAIR.LAYER sets the layer where the repaired trimmed surface is put when the repair surface action makes the tangent continuous in the underlying surface (if REPAIR.SURFACE = YES).

Repair Surface Tolerance

The flag REPAIR.SURFACE.TOLERANCE sets the tolerance distance between the original underlying surface and the repaired underlying surface. If the distance between these two surfaces is more than REPAIR.SURFACE.TOLERANCE, the original face is cut at the discontinuous parameters into several faces and then sewn into one trimmed surface and put in the CUT.LAYER. If the trimming operation fails, the original underlying surface and the trimming curves are put on REJECT.LAYER (if REPAIR.SURFACE = YES).

Cut Layer

The flag CUT.LAYER sets the layer where the trimmed surface is put when the action repair surface fails to repair the underlying surface
(if REPAIR.SURFACE =YES).

Reject Layer

The flag REJECT.LAYER sets the layer where the original NSURFACE is put when the trimming operation fails (if REPAIR.SURFACE = YES).

Comment File

The flag COMMENT.FILE sets the name of the text file which contains the annotation for that part. For details regarding the use of annotations, refer to “Using Annotations” on page 3-20.

Sample GET SET Parameter File

The following shows `gsetparams`, the default parameter file for GET SET. Comment lines start with an asterisk (*).

The gsetparams File

```
************************************************************
**
**              PARAMETER FILE FOR THE INTERFACE
**                    SET =====> CADDS
**
**               CADDS COMMAND : #0N# GET SET
**
************************************************************
*
*  SET STANDARD TEXT FONT CORRESPONDENCE
*  -------------------------------------
*  SET STANDARD TEXT FONT NUMBER --> USER TEXT FONT NUMBER
*  (DICTIONARY PARAMETER 101)        (FONT)
*     1 = standard
*     2 = ISO table
*     9 = Table of special SET characters
*     10 = KANJI character font
*
SET.TEXT.FONT 2 = 16
SET.TEXT.FONT 9 = 9
* New text font flag
* ---------------------
*
CREATE.TEXT.FONT = YES
*
* TRACE FLAG
* ---------------------
*     0 -------> Display informations of all blocks
*     1 -------> Display informations of surfaces repaired
*     2 -------> No display
*
* MESSAGES DISPLAY
* ---------------------
DISPLAY.MESSAGES = NO
* SET HEADER DISPLAY
* ---------------------
*
DISPLAY.SET.HEADER = NO
*
* SET reparameterization flag
* ---------------------
REPAIR.SURFACE = YES
*
REPAIR.ANGLE.TOLERANCE = 2
*
```

```
REPAIR.LAYER = 250

*
CUT.LAYER = 251
*
REJECT.LAYER = 252
*
REPAIR.SURFACE.TOLERANCE = 0.01
*
* Default convention File Name (in CADDS format).
* ----------------------------------------------
*
CONVENTION.FILE = cv-conv
*
* Library name and type (in CADDS format) .
* ----------------------------------------
* If present    --> Use this name instead of the library
*            name given in the SET file (#9906 or :11).
* If not present --> Use library name given in the SET file.
*
LIBRARY.NAME = libset
*
* Library flag : to specify the type of library figure.
*
*           1 --> sub-figure library (The default)
*           2 --> nodal figure library
*           3 --> extended nodal fig
LIBRARY.TYPE = 1
COMMENT.FILE = comment
```

Example Using GET SET

The following shows sample prompts and the user's response. Lines of output with only the $ prompt mean that the user pressed RETURN without typing a response. A detail explanation follows the example.

```
#n#  get set
*********************************************************
**
***              I.N.T.E.R.F.A.C.E
****
*****           SET  ===>   CADDS
****
***                RELEASE   3.3
**
*********************************************************
ENTER SET FILE NAME ? :
$  myset.file1
ENTER CADDS PART NAME ? :
$  part.converted.set1
GENERATING OF THE ASSEMBLIES SET FILES
--------------------------------------
NUMBER OF FOUND ASSEMBLIES = 1
PREPROCESSING OF THE ASSEMBLY SET FILE :
----------------------------------------
myset.file
NUMBER OF FOUND BLOCKS = 656

++++++++++++++SET FILE IDENTIFICATION+++++++++++++++++
NUMBER OF SET SPECIFICATION VERSION:   Z68-300.89.06
INDICE OF SET SOFTWARE              : 3200
NAME OF GENERATING SOFTWARE         : CADDS
VERSION OF GENERATING SOFTWARE      : 5.00
COMPANY                             : CAD Ltd
DIVISION                            : CAD/CAM
DEPARTMENT                          : Development
DESIGNER                            : Paul Dennis

CREATION DATE   4-18-1989  9:46:17
++++++++++++PART IDENTIFICATION OF THE ASSEMBLY++++++++
PART NAME : part.test.set
CREATION DATE   12-18-1993  9:14:25
BEGINNING OF PROCESS ON SET FILE : myset.file
---------------------------------
NUMBER OF SET MANAGEMENT BLOCKS :       6 OUT OF 656
TOTAL AMOUNT OF BLOCKS TO BE PROCESSED: 650 OUT OF 656
-----------------------------------------------------
NUMBER OF PROCESSED BLOCKS   : 646 OUT OF 650
PROCESSING OF THE POINTERS
--------------------------
The following Tsurf were repaired:
Nset = 510  Miptr= 430  Tag = FAC33 MAX DEV = 0.000029
```

```
Nset = 522 Miptr = 528  Tag = FAC42 MAX DEV = 6.350e-06
The following Tsurf were cut :
Nset = 517                   Miptr = 498 Tag = FAC39
Nset = 587                   Miptr = 630 Tag = FAC48
The following Tsurf were rejected :
Nset = 78                    Tag = SUR52  MAX DEV = 0.024365

The repair process of surfaces is ON.
Angle tolerance             :2.0000 deg
Geometric tolerance         :0.0100 mm
repair layer                :250
Cut layer                   :251
reject layer                :252
*****************************************************
**
**            END OF PROCESSING STATUS
**
*****************************************************
**  LINE segment              :2 OUT OF 2==>100.00%
**  PARAMETRIC CURVES         :380 OUT OF 380==>100.00%
**  PARAMETRIC SURFACE        :89 OUT OF  89 ==>100.00%
**  GENERALIZED SIMPLE FACE   :88 OUT OF 89 ==>99.00%
**  COMPOSITE CURVE           :88 OUT OF 89 ==>99.0%
**  VIEW BLOCK DEFINITION     :1 OUT OF 1 ==>100.00%
**  DRAW BLOCK DEFINITION     :1 OUT OF 1 ==>100.00%
**  ASSEMBLY                  :1 OUT OF 1 ==>100.00%
*****************************************************
**  TOTAL AMOUNT OF PROCESSED BLOCKS:648 OUT OF 650==>99.00%
*****************************************************
The repair process of surfaces is ON.
Repaired Tsurf               : 2 out of 89
Cut Tsurf                    : 2 out of 89
Rejected Tsurf               : 1 out of 89
S.E.T CONVERSION OF THE SET TEXT FILE : myset.file
NEW PART SAVED UNDER THE NAME  : part.converted.set
FILE NAME ? :
%
```

Specifying the SET File

Enter the SET filename for conversion (without `&bcd`). Pressing RETURN without specifying a name, returns you to the CADDS prompt (#*n*#). If the SET file does not exist, you can try again:

```
ENTER SET FILE NAME ? :
$ myset.file
FILE DOES NOT EXIST, TRY AGAIN.
ENTER SET FILE NAME ? :
```

Specifying the Part Name

Enter a name for the part where GET SET should store the entities of the SET file after conversion. If you press RETURN without specifying a name, you return to the CADDS prompt (#*n*#).

```
ENTER CADDS PART NAME ? :
$ part.converted.set
```

If a CADDS part exists with that name, you see the prompt:

```
CADDS PART ALREADY EXISTS, TYPE OK TO OVERWRITE? :
```

If you enter `ok`, the processor converts the SET text file and overwrites the existing CADDS part. If you press RETURN, the processor repeats the `ENTER CADDS PART NAME` prompt.

Processing the Part

The processor controls the SET block format and indicates the total number of SET blocks read.

```
   GENERATING OF THE ASSEMBLIES SET FILES
   --------------------------------------
   NUMBER OF FOUND ASSEMBLIES = 1
   PREPROCESSING OF THE ASSEMBLY SET FILE :
   ---------------------------------------
   myset.file
   NUMBER OF FOUND BLOCKS = 142
```

Displaying the File Header

The processor displays the SET file header if you requested it in the parameter file.

```
++++++++++++++++SET FILE IDENTIFICATION+++++++++++++++++

NUMBER OF SET SPECIFICATION VERSION: Z68-300.89.06+SOLIDE2
INDICE OF SET SOFTWARE: 3.2
NAME OF GENERATING SOFTWARE: CADDS
VERSION OF GENERATING SOFTWARE: 5.00
COMPANY: CAD Ltd
DIVISION: CAD/CAM
DEPARTMENT: Development
DESIGNER : Paul Dennis

CREATION DATE   4-20-1989  9:46:17
++++++++++++PART IDENTIFICATION OF THE ASSEMBLY+++++++++
PART NAME : part.test.set
CREATION DATE   4-18-1989  9:14:25
```

Processing the SET blocks

The processor indicates the number of SET management blocks (9900, 9901, 9902, 9997, 9998, 9999) and the total number of blocks to be processed.

It processes the SET file and indicates, in groups of 10, the number of blocks it has processed out of the total number to be processed, as shown on the following page:

```
BEGINNING OF PROCESS ON SET FILE : myset.file
---------------------------------
NUMBER OF SET MANAGEMENT BLOCKS: 16 OUT OF 142
TOTAL AMOUNT OF BLOCKS TO BE PROCESSED: 126 OUT OF 142
---------------------------------------
NUMBER OF PROCESSED BLOCKS :  126 OUT OF 126
```

The processor then processes all the associativity relations.

```
PROCESSING OF THE POINTERS
--------------------------
```

Displaying the Processing Status

The screen displays an end of processing status indicating, for each type of block, the number of blocks converted in the CADDS part.

```
*****************************************************
**          **
**             END OF PROCESSING STATUS**
**
*****************************************************
** POINT                      : 1 OUT OF  1==>100.00%
** LINE                       : 8 OUT OF  8==>100.00%
** BEZIER OR BSPLINE CURVE    :44 OUT OF 44==>100.00%
** BEZIER OR BSPLINE SURFACE  :13 OUT OF 13==>100.00%
** TOPOLOGICAL SOLID          : 1 OUT OF  1==>100.00%
** FACE COMPOSITE             : 2 OUT OF  2==>100.00%
** GENERALIZED SIMPLE FACE    : 7 OUT OF  7==>100.00%
** COMPOSITE CURVE            : 7 OUT OF  7==>100.00%
** VIEW BLOCK DEFINITION      : 1 OUT OF  1==>100.00%
** DRAW BLOCK DEFINITION      : 1 OUT OF  1==>100.00%
** TOPOLOGICAL VERTEX         :28 OUT OF 28==>100.00%
** TOPOLOGICAL EDGE           :12 OUT OF 12==>100.00%
** ASSEMBLY                   : 1 OUT OF  1==>100.00%
*****************************************************
** TOTAL AMOUNT OF PROCESSED BLOCKS :
** 126 OUT OF 126 ==>100.00%
*****************************************************
```

```
S.E.T CONVERSION OF THE SET TEXT FILE : myset.file
NEW PART SAVED UNDER THE NAME  : part.converted.set
#n#
```

Transforming Entities

Transforming Block @100

The SET processor translates the constructed solid as follows:

When running GET SET, a constructed solid is generated by the SET to CADDS interface. However, you have to execute certain operations outside the interface. For example:

- UNION
- INTERSECTION
- SUBTRACT
- SCULPT
- SWEEP
- MATRIX TRANSFORMATION

In these cases, the constructed solid is generated by an EXECUTE file. This EXECUTE file is generated by the SET to CADDS interface at the same time as the CADDS part.

The name of this EXECUTE file is：

```
construction_file
```

and it is stored at the same level as “_pd” file of the resulting part.

Transforming Sub-block #101

Where the transformation matrix sub-block is #101 the entity is transformed by the TRANSFORM ENTITY command in CADDS. This is executed by the “construction-file” EXECUTE file.

Transforming CADDS Super-type Entities

The following entities with a CADDS super-type of:

- POINT
- CURVE
- SURFACE
- SOLID

are transformed using the following transformation:

$$\begin{Bmatrix} Xt \\ Yt \\ Zt \end{Bmatrix} = \begin{bmatrix} A1 & A4 & A7 \\ A2 & A5 & A8 \\ A3 & A6 & A9 \end{bmatrix} \begin{Bmatrix} Xi \\ Yi \\ Zi \end{Bmatrix} + \begin{Bmatrix} A10 \\ A11 \\ A12 \end{Bmatrix}$$

Where:

Xi, Yi, Zi are the coordinates of the initial entity

Xt, Yt, Zt are the coordinates of transformed entity

Ai (i=1,9) are the coefficients of the transformation matrix:

- Rotation
- Mirror
- Scaling
- General Transformation

Ai (i=10,12) are the Translation Parameters.

Degenerated Operators

In the case of degenerated operator, the TRANSFORMED SOLID is not generated.

Example of Transforming Entities

The projection of the solid box on the plane is not generated.

Valid Mode

- The part must be active.
- The mode of the part must be mode MODEL
- The SET license must be active

Syntax

```
#n# MTRANSFORM ENTITY [modifiers] : Model ent
```

Modifiers

Coefficients of the transformation matrix:

```
A <A1>
B <A2>
C <A3>
D <A4>
E <A5>
F <A6>
G <A7>
H <A8>
I <A9>
```

Coefficients of translation vector:

```
J <A10>
K <A11>
L <A12>
```

Name of the Transformed ENTITY:

```
TAG <name>
```

Layer of the Transformed ENTITY:

```
LAY <n>
```

Creating a New Text Font

Follow these steps to create a new text font corresponding to @252 SET block type:

Compiling the Text Font

1. Start CADDS.
2. Set the flag of the new fonts in `gsetparams` to YES.
3. Execute GET SET.
4. Exit from CADDS.
5. Compile the new text font using the following commands:
   a. `cp font1cx.F /usr/apl/cadds/src/graphics/ggs/textfont/font1cx.F`
   b. `cd /usr/apl/cadds/src/graphics/textfont`
   c. `f77 -c font1cx.F`
   d. `maketextfont font1cx`
   e. `cd /usr/apl/cadds/bin`

Building the Text Font

You use the command `BUILD_text_font USER` to build the new text font. The following dialog shows the prompts and responses:

```
Enter FONT number (2 digit hexadecimal value)   -->1c
Type "OK" to enable PRIMARY graphics            --OK
Type "OK" to enable SECONDARY graphics          --OK
Type "OK" to enable TAG graphics                --OK
Compilation in progress.....
Output file: /usr/apl/cadds/data/textfont/efont1c
Text Font Compilation complete
Type "OK" update the text font "directory" file --OK
```

Using the REPAIR SURFACE Function in GET SET

Surfaces from another CAD/CAM system can create discontinuous tangents, and using these surfaces in CADDS may cause some problems with CADDS commands such as INTERSECT ENTITY.

Using the repair surface functionality, the surfaces are repaired inside the GET SET interface before being used in CADDS.

REPAIR SURFACE creates the trimmed surface inside the SET interface, even if this surface had tangent discontinuities. The trimmed surface is then constructed in the same way as in the CADDS command TRIM SURFACE.

Please note: This repair creates extra topology, namely curves and edges where the surface discontinuity occurred, and it is placed on the layer (CUT.LAYER) specified in the `gsetparams` file.

Repairing a Surface

The REPAIR SURFACE process is as follows:

- Change the parameterization of surfaces (REPARAMETERIZATION) and remove small segments. This does not change the geometry of the surface; Tsurfaces and Nsurfaces are taken into account.
- Treat the trimmed surfaces which have discontinuous tangents where the angle between the right and left tangents at the point of discontinuity is greater than 0.1 degree. This process may change the geometry.

Trimmed Surfaces

The TRIM SURFACE command subdivides surfaces if the angle between the right and left tangents at the discontinuity points is more than an internal tolerance (is equal to 0.1 degree). The repair surface functionality repairs surfaces only if the angle between the right and left tangents at the discontinuity points is more than 0.1 degree.

When these surfaces cannot be repaired, instead of putting the underlying Nsurface and the trimming curves on the repair layer (REPAIR.LAYER), the trimmed surfaces are created as described in the following algorithm:

Algorithm

Trimmed surface with discontinuous tangents generated by SET interface = TS

The elements required to create this trimmed surface are:

- 1 Nsurface S
- n contours Ci, i=0,n-1

  where

  C0 is the external contour,
  Ci, i=1,n-1, are the internal contours

  Nu and Nv are respectively the number of u discontinuities and v discontinuities where the angle between the right and left tangents is more than 0.1 degree.

Apply the phases of this algorithm:

- Subdivide the trimmed surface ***TS*** at each tangent discontinuity parameter where the angle between the right and left tangents is more than 0.1 degree. (Trim the first trimmed surface TS with the u iso line (resp. v iso line) at the u discontinuity parameter (resp. at the v discontinuity parameter). The boundary for each sub trimmed surface TSi created, is a part of the first boundary C, and a part of the u iso line or the v iso line.
- Sew all these trimmed surfaces and create one composite trimmed surface.

Please note: TRIM SURFACE subdivides surfaces at each discontinuity if the angle between the right and left tangents at the discontinuity point is more than 0.1 degree. This tolerance can be decreased if you have problems with other commands such as INTERSECT ENTITY. This decreases the repair surface tolerance.

Using Annotations

The SET rule allows you to add annotations to the different CFAO systems using the #9902 sub-block. An annotation, according to the SET rule, is a sequence of alphanumeric characters. Every character has a value between 32 and 126 in ASCII code.

SET Annotations

The SET annotation is a permanent sequence of characters written into the #9902 sub-block.

This sub-block can be added under all the blocks except the @9997, @9998 and @9999 blocks.

ASCII characters not contained in the SET characters, will be written as blank characters.

Example

```
@9901,1,'TEST ANNOTATION'
#9902,'ZONE NORMES GENERALES'
#9902,'TOLERANCES NON SPECIFIES'
```

Using GET SET Annotations

The CADDS-SET interface writes one file of annotation for every assembly or sub-assembly having more than one #9902 sub-block.

You can set the name of the test file containing the annotation under the part, by setting the flag COMMENT.FILE in the `gsetparams` file.

If the #9902 sub-block is a part of the @9902 block, its annotation is written in the annotation file of the sub-assembly.

If the #9902 sub-block is a part of @9900 or @9901 blocks, its annotation is written in the annotation file of the part.

Also, if the #9902 sub-block is related to an entity, the SET entity number and the MIPTR of the same entity will be written in the annotation file.

The following tables summarizes this application:

Table 1

The annotation file name is assigned by name. In `gsetparams` set: COMMENT.FILE=comment

parameter SET	structure
setdir/_bcd/setname	parts/partname
@9902	parts/partname/_pd
#9902,...	parts/partname/comment
@9901,1,1	
#9902,...	
@x,...	
#9902	
@9902,27	parts/partname/fig/27/_pd
#9902 ...	parts/partname/fig/27/_sfig
@y,	parts/partname/fig/27/comment
#9902 ...	
@9997	
...	
@9998	
@9999	

Table 2

The annotation file name is assigned by pathname.

In `gsetparams` set: COMMENT.FILE=dircom.comment

The pathname must be in CADDS format (CGOS).

parameter SET	structure
setdir/_bcd/setname	parts/partname
@9900	parts/partname/_pd
#9902,...	parts/partname/comment
@9901,1,1	
#9902,...	
@x,...	
#9902	
@9902,27	parts/partname/fig/27/_pd
#9902 ...	parts/partname/fig/27/_sfig
@y,	parts/partname/fig/27/dircom/comment
#9902 ...	
@9997	
...	
@9998	
@9999	

Table 3

The annotation file name is assigned by name.

parameter SET	structure
setdir/_bcd/setname	parts/partname
@9900	mylib/comment
#9902,...	
#9906, ‘mylib	
@9901,1,’modelname’	mylib/modelname/_pd
#9902,...	mylib/modelname/_sfig
@x,...	mylib/modelname/comment
#9902	
@9998	
@9901,1, ‘drawname’, ...	mylib/drawname/_pd
#9902 ...	mylib/drawname/_sfig
@y,	mylib/drawname/comment
#9902 ...	
@9998	
@9999	

SET file holds more assemblies.

In gsetparams set:COMMENT.FILE=comment

*LIBRARY.NAME=libset (commentary)

A file must be assigned a library name using the #9906 sub-block. For example:

```
#9906,'mylib','ensembles1','ensembles2'
```

The annotations of the @9901 assembly will be written in ../mylib/ensemble1/comment.

The annotations following @9900 block will be written in ../parts/partname/comment.

Limitations

- The test length is restricted to 246 characters.
- Special characters like @ or # are not allowed.

Sample Report

SET report gives the information about the repaired, cut, and rejected Tsurfs. The status contains the following information:

- SET block number of the Tsurfs (Nset)
- Master index pointer of the Tsurfs (Miptr)
- Tag of the Tsurf
- Maximum deviation between the original underlying surface and the repaired underlying surface (MAX DEV)
- Angular tolerance
- Geometric tolerance
- Cut layer
- Reject layer

A sample report follows.

```
The following Tsurf were repaired
Nset =  8 Miptr = 185 Tag = FAC33 MAX DEV = 0.000029
Nset = 43 Miptr = 528 Tag = FAC42 MAX DEV = 6.350788e-13
The following Tsurf were cut
Nset = 28 Miptr = 245 Tag = FAC37
Nset = 31 Miptr = 28  Tag = FAC38
The following Tsurf were rejected
Nset= 78  Tag = FAC52 MAX DEV = 0.013841
The repair process of surfaces is ON.
Angle tolerance : 5.0000 deg
Geometric tolerance : 0.0100 mm
Cut layer : 251
Reject layer : 252
************************************************************
      END OF PROCESSING STATUS
************************************************************
** Line segment : 2 out of 2 ===> 100.00%
** Parametric curve : 380 out of 380 ===> 100.00%
** Parametric surface : 89 out of 89 ===> 100.00%
** Simple face : 88 out of  89 ===> 99.00%
** Composite curve : 88 out of  89 ===> 99.00%
** Group : 1 out of 1 ===> 100.00%
** Assembly :  1 out of 1 ===> 100.00%
************************************************************
Total amount of processed blocks:648 out of 650===>   99.50%
************************************************************
Repaired Tsurfaces : 2 out of  89
Cut Tsurfaces : 2 out of  89
Rejected Tsurfaces : 1 out of  89
```

Chapter 4

Using The CONVENTION File

This section contains information on how to use the CONVENTION file with the GET SET and PUT SET commands.

- Description of the CONVENTION File
- Defining Reference Tables
- Sample CONVENTION File
- PUT SET and the CONVENTION File
- GET SET and the CONVENTION File

Description of the CONVENTION File

The CONVENTION file is a document agreed upon by all parties wishing to exchange data. It details how data is exchanged.

For example, the CONVENTION file may include the name of “table” for the dictionary entry ’:18’ or ’:25’ which contains the definition of character font, line font, or attributes like color table, crosshatching and so on.

If you need to use the CONVENTION file, you must set the corresponding flags in the `gsetparams` and `psetparams` files.

New Features

The CADDS to SET interface did not use the CONVENTION file in previous releases. It means that the following items are now translated as follows:

Item	Translated by
Text fonts	SET block type @252
Line fonts	SET block type @250
Attributes	SET sub-block type #521

This concept applies even if you do not use the CONVENTION file.

Line Length

In the CONVENTION file, a line cannot have more than 256 characters, but you can create continuation lines by using the “>” character as the last character on the line.

Defining Reference Tables

Using the CONVENTION file allows you to define tables to set up a relationship between SET list references and CADDS list references.

In the SET file, you set up a reference to a CONVENTION file by setting the following dictionary entries:

Entry	Description
:25	Defines the name of the CONVENTION file to use
:18	Specifies the table reference to use. This is specified in the CONVENTION file. The following dictionary entries refer to the table elements: 101 Indicates the name of the character font which is defined in the TEXT_FONT table. 53 Indicates the name of the line font which is defined in the LINE_FONT TABLE of the CONVENTION file.

Defining the Conventions

These conventions are defined in the CONVENTION file, according to the following rules:

- Lines beginning with an asterisk (*) are comments.
- The table type is defined with the keyword in the first column:
  - LINE_FONT
  - TEXT_FONT
  - ATTRIBUTES
- The table name is given by table_name ='TABLE_NAME'

  The default table name is the name of the table type. For example; LINE_FONT, TEXT_FONT or ATTRIBUTE.

Example of Dictionary Entries

The following example shows how you define the dictionary entries:

```
:25,'file-conv'
:18,'attribute table'
```

where “attribute_table” is the attribute table defined in the CONVENTION file ’file_conv’.

Note on Table Names

The PUT SET interface in CADDS generates only one TABLE_NAME in the CONVENTION file, so only the default case is treated in the CONVENTION file which comes from CADDS.

The next lines (until the next key word) are used to define the table.

- The first word defines the SET reference
- The next word defines the CADDS reference
- The last words are comments

Example 1:

```
*SET code/CADDS code/ CADDS name
LF12                   9
DASH
...BB,4,265,41122,8442,250
...TT,2,0.,Y,25.,N,50.
```

`BB` Binary definition of the line. The first number gives the number of the words in the binary definition.

`TT` Trace definition of the line. The first integer gives the number of segments and the next number is the shift:

`Y` Trace until the next code

`N` Jump to the next code

Please note: If the two lines exist, the `BB` line has a higher priority. This line describes the binary definition of the line font which is used only by CADDS, but the `TT` line describes the trace of the font and then it is interpretable by the user of the font.

For users who work on other systems as well as CADDS but using outgoing CADDS SET files, only the `TT` line is useful.

Example 2:

This is a more complex example:

```
"...TT,2,0.0,Y,25.0,N,50.0,S12,3.0E14,4.0"
```

`Si` Indicates a symbol number i, and it is followed by the symbol scaling

`Ei` Indicates a symbol of extremity

Sample CONVENTION File

This example is a convention file created by the PUT SET command from the CADDS to SET interface.

```
*Mon Apr 5 16:06:31 1993 SET processor; Created convention
*to exchange data for the interface CADDS5 ==> SET
*********************************************************
*CADDS5 command : PUT SET
*********************************************************
*
ATTRIBUTE TABLE
*
* SET code / CADDS code / CADDS name
*
112                        4102            CHANNEL
103                        36936           VARIANT
194                        40962           ORGHIST
*
TEXT_FONT_TABLE
*
* SET code / CADDS code / CADDS name
*
TF2                     1                  STANDARD (PSPACE)
TF35                   34                  LEROY (USPACE)
*
LINE FONT TABLE
* SET code / CADDS code /  CADDS name
LF121                  9                   DASH
...BB, 4, 265, 41122, 8442, 250
...TT, 2, 0.0000,Y, 25.00000, N, 50.0000
LF122                  42                  ORCHARD
...BB, 4, 298, 40962, 300, 17164
...TT, 1, 0.000, N, 30.000, S12, 3.0000
```

PUT SET and the CONVENTION File

The PUT SET command uses a CADDS part to create an 80-column SET text file. You also need a separate parameter file (with information to initialize the SET file) to execute PUT SET. In a CONVENTION file used by the PUT SET command, observe the following format:

- SET reference must be coded in the first 7 columns using following format:
  - A number for the attribute
  - 'LFi' where i is a number for the line fonts
  - 'TFi' where i is a number for the text fonts
- CADDS reference must be coded between columns 9 and 15
- Comments can be coded from column 17

Naming the CONVENTION File

In the `psetparams` file, the keyword CONVENTION.FILE gives the convention name to be used in the generation of the SET file:

```
CONVENTION.FILE = cv-conv
```

No Keyword in the File:

If this keyword does not appear, no conventions are used, and the SET file is written as in the previous versions:

Block @252 Defines the character font or uses a standard SET number

Sub-block #521 Defines the attributes

Block @250 Defines the line font or uses a standard SET number.

Using the CONVENTION Name

If a convention name is given, there are two possible results:

- If the `cv-conv` file exists in the CVPATH directory and if the SET codes mentioned are already used in the SET file, the `cv-conv` file is not updated. But if new codes are needed, the CONVENTION file is updated.
- If `cv-conv` does not exist, it is created in the first directory mentioned in the CVPATH.

The consequences for the SET file are:

- The character string `:25,'cv-conv'` appears in sub-block #9920.
- The entity properties are transferred by sub-block #520

Example 1:

```
#520, :18, 'ATTRIBUTE', 'TERMTYPE', 1, 89, 125
```

Instead of referencing the character font block type or the standard principal font (as in the previous versions, where there was no CONVENTION file), the corresponding name of the character font is found in the CONVENTION file.

Example 2:

```
:101,'TF6'
```

This example uses line fonts. Every time a line type is referenced or standard line number is referenced, the corresponding name of the line font is found in the CONVENTION file.

```
@9900, ....
#9906,LIB,[,NOM1,.....,NOMi,...]
```

GET SET and the CONVENTION File

In the `gsetparams` file, the keyword CONVENTION.FILE gives the default convention name. The default is used when the CONVENTION file specified does not exist in the CVPATH directories.

Example

The SET file contains a reference to a CONVENTION file `conv1`.

In the `gsetparams` file, CONVENTION.FILE is set to `cv-conv`.

If `conv1` and `cv-conv` exist in the CVPATH directories, the GET SET command uses the `conv1` convention file.

If `conv1` does not exist in the CVPATH directories, the GET SET command uses the `cv-conv` file.

If neither `conv1` nor `cv-conv` exist in the CVPATH directories, the GET SET command cannot use any conventions.

Chapter 5

Using the LIBRARY Function

This section contains information on how to use LIBRARY in the GET SET and PUT SET commands.

- Description of the INTERNAL LIBRARY Function
- Description of the EXTERNAL LIBRARY Function
- PUT SET and the External LIBRARY Function
- GET SET and the External LIBRARY Function
- Using the CALL Definition

Description of the INTERNAL LIBRARY Function

The SET standard can exchange several sub-assemblies (an INTERNAL LIBRARY) in one SET file.

INTERNAL LIBRARY Structure

An INTERNAL LIBRARY or data sub-assembly consists of a sequence of data blocks. The sequence starts with block @9902 (start of data sub-assembly), and ends with block @9997 (end of data sub-assembly). SET files with an INTERNAL LIBRARY structure contain sub-assembly blocks @9902 and @9997 between SET blocks @9901 and @9998. Block @9902 in the SET file shows the file is transmitting members of an internal LIBRARY:

```
@9900, ....
@9901,
@9902,..........@9997,
@9902...........@9997
@9998
@9999
```

Example of Translating Data

Here, the CADDS to SET interface does not write the SET file with SET block @9902, but the SET to CADDS interface reads the SET file which contains one or several data sub-assemblies.

```
@9900,...
@9901,1,......
@9902,101, 'name1'
....
@9997,...
@9902,320, 'name2'
....
@9997,...
@9902,980,'name3'
...
@9997,...
@9998,1500
@9999,1502
```

All the INTERNAL LIBRARIES are translated in CADDS as sub-figures (Sfigs). The translation of the SET file which contains data sub-assemblies is as follows:

```
partname/_pd
partname/fig/name1/_sfig
partname/fig/name2/_sfig ...........
```

where:

`partname` Is the part name requested by GET SET

`name1,name2,.....` Is the name of each SET block @9902 in the assembly.

The result of the data translation from the last example is:

```
partname/_pd
partname/fig/name1/_sfig
partname/fig/name2/_sfig
```

Summary of Data Translation

The translation of the INTERNAL LIBRARY from SET to CADDS is summarized in the following table:

SET Data	CADDS Data
@9900....	--------
@9901,1,	partname/_pd
@9902, n1,'name1',@9997,	partname/fig/name1/_sfig
@9902, n2,'name2', @9997,	partname/fig/name2/_sfig
@9902, nk,@9997	partname/fig/nk/_sfig
@9998,	--------
@9999,	--------

Description of the EXTERNAL LIBRARY Function

The SET standard is able to exchange several assemblies (which form an EXTERNAL LIBRARY) in one SET file.

EXTERNAL LIBRARY Structure

The SET files which have an EXTERNAL LIBRARY structure contain the SET sub-block #9906 under the @9900 block type. Having this sub-block at the start of the SET file block shows that:

- The file transmits members of a LIBRARY named LIB.
- All or only part of the data assemblies in the SET file belong to the LIBRARY.

For example:

```
@9900,....
#9906,LIB,[,NAME1,.....,NAME2,...]
```

If the NAMEn list is empty, all assemblies in the file belong to the LIBRARY.

If the NAMEn list is not empty, all the names referenced in the list are the assembly names which belong to the LIBRARY.

Example 1

In this example the LIBRARY `lib` contains all the assemblies: `draw`, `model` and `assembly`.

```
@9900,....#9906,'lib'
@9901,1,'draw' ........@9998,100
@9901,1,'model',........@9998,250
@9901,1,'assembly',....@9998,130
@9999,385
```

Example 2

Here, LIBRARY `lib` contains two assemblies: `draw` and `model`. The third assembly, `assembly`, does not belong in LIBRARY `lib`.

```
@9900, ....#9906,'lib','draw','model'
@9901,1,'draw'...........@9998,100
@9901,1,'model'..........@9998,250
@9901,1,'assembly'.......@9998,130
@9999,385
```

PUT SET and the External LIBRARY Function

LIBRARY in CADDS is a directory which can contain several CADDS parts. The result is one SET file with several assemblies (the number of assemblies generated is the same number as the CADDS parts which exist under the LIBRARY directory).

Writing a SET file with SET sub-block #9906 depends on the LIBRARY flags in the `psetparams` file. The flag can be `YES` or `NO`.

Library Flag	Description
LIBRARY.FLAG=NO	The program asks for the part name to transfer. Sub-block #9906 is not generated.
LIBRARY.FLAG=YES	The PUT SET command asks for a LIBRARY name instead of a partname. The following files found under this directory are treated as members of the LIBRARY: _pd _sfig _nfig

Example

The LIBRARY `lib1` contains three CADDS parts:

- lib1/draw
- lib1/model
- lib1/assembly

When the LIBRARY.FLAG is set to YES in the `psetparams` file, the PUT SET command asks for:

```
LIBRARY name ?
```

The result is one SET file with 3 assemblies:

```
@9900,.........#9906,'lib1'
@9901,1,'draw',...........,@9998,100
@9901,1,'model',..........,@9998,250
@9901,1,'assembly',.......,@9998,130
@9999,385
```

Summary of Data Translation

The translation of CADDS parts, Sfigs or Nfigs, from CADDS to SET is summarized in the following table. In this example, all the parts are under the directory, `lib1`:

CADDS Data	SET Data
-------	@9900,#9906,'lib1'
lib1/drawname/_sfig	@9901,1,'drawname',,@9998,
lib1modelname/_nfig	@9901,1,'modelname',,@9998,
lib1/assemblyname/_pd	@9901,1,'assemblyname',,@9998,
-------	@9999,

GET SET and the External LIBRARY Function

The command GET SET generates Sfigs, Nfigs or Extended Nfigs from a SET file which contains sub-block #9906.

The LIBRARY name given in the `gsetparams` LIBRARY.NAME flag has higher priority than a given LIBRARY name in the SET file.

Every assembly member of the LIBRARY name of the SET file is transferred as an Sfig, Nfig or an Extended Nfig as shown in the following table:

LIBRARY.TYPE	Type	Result
1	_sfigs	library_name.assembly_name.sfig
2	_nfigs	library_name.assembly_name.nfig
3	_nfigs	library_name.assembly_name.nfig

Please note: LIBRARY.TYPE is a `gsetparams` flag.

The other assemblies which do not belong to the LIBRARY are created under the partname requested by the GET SET command:

`part_name.ass.`*n* (n is the assembly range in the SET file)

Example 1

If LIBRARY.TYPE is set to 1 and the LIBRARY.NAME is not set in the `gsetparams` file, the GET SET command generates the following files:

- lib.draw._sfig
- lib.model._sfig
- lib.assembly._sfig

```
@9900, ..... #9906,'lib'
@9901,1,'draw',.....@9998,100
@9901,1,'model',.....@9998,250
@9901,1,'assembly', .....@9998,130
@9999,385
```

If the LIBRARY.TYPE is set to 2, and the LIBRARY.NAME is not set, the GET SET command generates the following files:

- lib.draw._nfig
- lib.model._nfig
- lib.assembly._nfig

If the LIBRARY.TYPE is set to 1, and the LIBRARY.NAME is libset, the GET SET command generates the following files:

- libset.draw._sfig
- libset.model._sfig
- libset.assembly._sfig

Example 2

If LIBRARY.TYPE is set to 1, and the LIBRARY.NAME is not set in the `gsetparams` file, the GET SET command generates the following files:

- lib.draw._sfig
- lib.model._sfig
- part.ass.3._sfig

where part is the partname requested by GET SET.

```
@9900, ..... #9906,'lib','draw','model'
@9901,1,'draw',.............@9998,100
@9901,1,'model',...........@9998,250
@9901,1,'assembly', ......@9998,130
@9999,385
```

If the LIBRARY.TYPE is set to 2, and the LIBRARY.NAME is not set, the GET SET command generates the following files:

- lib.draw._nfig
- lib.model._nfig
- part.ass.3._nfig

where part is the partname requested by GET SET.

If the LIBRARY.TYPE is set to 1, and the LIBRARY.NAME is libset, the GET SET command generates the following files:

- libset.draw._sfig
- libset.model._sfig
- part.ass.3._sfig

where part is the partname requested by GET SET.

Summary of Data Translation with an External Library

The translation of a SET file containing an EXTERNAL LIBRARY is summarized in the following table.

SET Data	Library Type	CADDS Data
@9900...#9906,'libname', @9901,1,'drawname', @9901,1,'modelname', @9999	1	-- libname/drawname/_sfig libname/modelname/_sfig --
@9900...#9906,'libname', @9901,1,'drawname', @9901,1,'modelname', @9999	2	-- libname/drawname/_nfig libname/modelname/_nfig --
@9900 #9906,'libame','drawname', @9901,1,'drawname', @9901,1,'modelname', @9999	1	-- libname/drawname/_sfig partname/ass/2/_sfig (partname = the part name requested by GET SET) --
@9900. #9906,'libname','drawname,' @9901,1,'drawname', @9901,1,'modelname', @9999	2	-- libname/drawname/_nfig partname/ass/2/_nfig -- --

Summary of Data Translation with Internal and External Libraries

The translation of a SET file containing EXTERNAL and INTERNAL LIBRARIES is summarized in the following table.

SET Data	Library Type	CADDS Data
@9900..#9906,'libname' @9901,1,'drawname' @9902,2,.....@9997 @9902,251,.....@9997 @9902,531,.....@9997 @9998 @9901,1,'modelname' @9998 @9999	1	-- partname/_pd libname/drawname/_sfig libname/drawname/fig/2/_sfig libname/drawname/fig/251/_sfig libname/drawname/fig/531/_sfig libname/modelname/_sfig --
@9900..#9906,'libname' @9901,1,'drawname' @9902,2,.....@9997 @9902,251,.....@999 @9902,531,.....@9997 @9998 @9901,1,'modelname' @9998 @9999	2	-- partname/_pd partname/drawname/_pd libname/drawname/_nfig libname/drawname/fig/2/_sfig libname/drawname/fig/251/_sfig libname/drawname/fig/531/_sfig libname/modelname/_nfig --

Using the CALL Definition

A call defines the inclusion of a LIBRARY element or a previously defined sub-set in the current data SET.

- If you call a module of an INTERNAL LIBRARY, the SET block number of the corresponding sub-assembly is referenced.
- If you call a module of an EXTERNAL LIBRARY, its name is given by dictionary entry:11.

PUT SET

When Sfig, Nfig, or Nfig All are inserted into one part, the PUT SET command on this part generates a calling block (SET block@500) in the resulting SET file. This references the called LIBRARY (in this case; Nfig, Sfig or Nfig All) by dictionary entry:11 in the SET sub-block #500.

GET SET

The SET to CADDS interface calls an INTERNAL LIBRARY using the SET block number of the corresponding sub-assembly. Transferring the calling assembly generates a CADDS part, where the called INTERNAL LIBRARY is referenced as an Sfig.

When you call a module of an EXTERNAL LIBRARY by dictionary entry:11 in the resulting transferred calling assembly, the referenced LIBRARY is inserted as an Sfig or Nfig (depending on the LIBRARY TYPE flag in the `gsetparams` file.)

Appendix A

Supported Entities

This Appendix lists the entities supported by the PUT SET and GET SET commands.

- List of Entities

List of Entities

Table A-1 Entities Supported by GET and PUT SET

CADDS Entity Type	CADDS Entity Number	SET Entity	SET Block Number	SET Sub Block Number	Read	Write
POINT	2	POINT	1	 1	y y	y y
LINE	3	LINE	2	 2	y y	y y
STRING	70	LINEAR STRING	3	 3	y y	y y
VECTOR	22	VECTOR	4	 4	y y	n n
POINT	2, 87	SET OF POINTS	5	 5	y y	n n
ARC	5	CIRCULAR ARC	10	 10	y y	y y
CONIC	6	ELLIPTICAL ARC	11	 11	y y	y y
CONIC	6	PARABOLIC ARC	12	 12	y y	y y
6	6	HYPERBOLIC ARC	13	 13	y y	y y
SPLINE	7	PARAMETRIC CURVE	20	 20 21 9911	y y y y	y y y y
B-SPLINE	8			 20 21 9911	y y y y	y y y y
CURVE POLE	9			 20 21 9911	y y y y	y y y y
NSPLINE if degree > 7	12	PARAMETRIC CURVE	20	 20 21 9911	y y y y	n n n n
CURVE POLE	9	GENERAL B-SPLINE CURVE	21	 22 5 23 24 9911	y y y y y y	n n n n n n

CADDS Entity Type	CADDS Entity Number	SET Entity	SET Block Number	SET Sub Block Number	Read	Write
NSPLINE	12				y	y
				22	y	y
				5	y	y
				23	y	y
				24	y	y
				9911	y	y
PLANE/FACE/HOLE	88	PLANE	30		y	n
				30	y	n
PLANE/FACE/HOLE	88	PARAMETRIC PLANE	31		y	n
				31	y	n
TABULATED CYLINDER	16	LIMITED CYLINDRICAL SURFACE	32		y	n
				32	y	n
SURFACE OF REVOLUTION	17	LIMITED CONICAL SURFACE	33		y	n
				33	y	n
SURFACE OF REVOLUTION	17	BOUNDED SPHERICAL SURFACE	34		y	n
				34	y	n
				35	n	n
SURFACE OF REVOLUTION	17	SURFACE OF REVOLUTION	36		y	y
				130	y	y
TABULATED CYLINDER	16	TABULATED CYLINDER	37	131		
					y	y
					y	y
RULED SURFACE		RULED SURFACE		132		
	18		38		y	y
					y	y
SPLINE SURFACE	19	PARAMETRIC SURFACE	40		y	y
				40	y	y
				41	y	y
				9911	y	y
SURFACE POLE	20				y	y
				40	y	y
				41	y	y
				9911	y	y
NSURFACE if degree>7	14				y	n
				40	y	n
				41	y	n
				9911	y	n
NSURFACE	14	GENERAL B-SPLINE SURFACE	41		y	y
				42	y	y
				5	y	y
				23	y	y
				24	y	y
				991	y	y

CADDS Entity Type	CADDS Entity Number	SET Entity	SET Block Number	SET Sub Block Number	Read	Write
SURFACE POLE	20				y	n
SOLID	91	PRIMITIVE SOLID	50	 32 33 34 35 36 60 61 140 141 142 503 320	y y y y y y y y y y y y y	y y y y n y y n n n n y n
SOLID	91	CONSTRUCTED SOLID	100	 100 101 139 143 144 145 146 503 320	y y y y y y y y y y	y n n n n n n n y n
SOLID	91	BOUNDARY SOLID	101	 123 505 506 320	y y y n y	y y n n y
TRIMMED SURFACE Only if each edge belongs to no more than 2 faces.	91	COMPOSITE FACE	102	 121 502 502 123 122 125 510	y y y y n n n n	y y n y n n n n
FACE	92	SIMPLE FACE	103	 123 505 507 3 10 11 21 120	y y y n y y y y y	y y n n n n n n n

CADDS Entity Type	CADDS Entity Number	SET Entity	SET Block Number	SET Sub Block Number	Read	Write
FACE	92	FLAT FACET	104		y	n
				123	y	n
				505	y	n
				507	n	n
				3	y	n
				4	y	n
NO ENTITY		COMPOSITE SET OF POINTS	105		y	n
				502	y	n
NO ENTITY		WIREFRAME MODEL	106		y	n
				120	y	n
				502	y	n
SOLID Only if each edge belongs to no more than 2 faces.	91	SURFACE MODEL	107		y	n
				121	y	n
				502	y	n
				502	y	n
POINT	2	CONSTRUCTED GEOMETRIC POINT	110		y	n
				100	y	n
				101	y	n
				503	y	n
CURVE	3	CONSTRUCTED GEOMETRIC CURVE	111		y	n
				100	y	n
	7			101	y	n
	8			103	y	n
	9			105	y	n
	12			503	y	n
SURFACE	14	CONSTRUCTED SURFACE	112		y	n
	16			101	y	n
	17			104	y	n
	18			106	y	n
	19			503	y	n
	20					
FACE	92	CONSTRUCTED FACE	113		y	n
				100	y	n
				101	y	n
				104	y	n
Entity group when pointed to by a face.		COMPOSITE CURVE	120		y	y
				120	y	y
				502	y	y
View		VIEW BLOCK DEFINITION	215		y	y
				310	y	y
				216	y	y
				401	n	n
				251	n	n
				511	y	y

CADDS Entity Type	CADDS Entity Number	SET Entity	SET Block Number	SET Sub Block Number	Read	Write
Symbol Line Font		PURE GRAPHICAL SYMBOL	251	 257	 n n	 n n
Text Font		CHARACTER FONT DEFINITION	252	 324 255	y n y	n n y
		CONSTRUCTED HATCHING	260		n	n
		COLOR TABLE	265	 266 267	y y y	y n y
Construction Plane		COORDINATE SYSTEM	301	 301 317	y y n	y y y
Geometric transformation		GEOMETRIC TRANSFORMATION	302	 301 313 314 315 316 317	y y n n n n n	n n n n n n n
		LIST OF HOMOGENEOUS SCALARS	303		n	n
		LIST HETEROGENEOUS SCALARS	304		n	n
		GENERAL MATRIX	305		n	n
		ARRAY OF ORDER n	306		n	n
		Association between physical values and geometry	307		n	n
Layers List	List	LIST OF LAYERS	310		n	n
		PURE TEXT	311		n	n
		TABULATED FUNCTION	315		n	n
		ANALYTIC FUNCTION	316		n	n
		REVISION TRACKING	321		n	n
		PHYSICAL QUANTITY TABLE	325		n	n
		VALUES ASSIGNED TO PHYSICAL QUANTITY	326		n	n
Drawing		DRAWING BLOCK DEFINITION	220	 220 221 251	y y y n	y y y n

CADDS Entity Type	CADDS Entity Number	SET Entity	SET Block Number	SET Sub Block Number	Read	Write
Line Font		LINE STYLE	250		y	y
				324	n	n
				253	y	y
				254	n	n
TEXT NODE	82	TEXT NODE	400		y	y
				1	y	y
				400	y	y
				510	y	y
				409	y	n
TEXT	85	GRAPHIC TEXT	410		y	y
				401	y	y
NODAL TEXT	86				y	y
				401	y	y
				510	y	y
					y	y
					y	y
Feature Control Symbol	32	FRAME GRAPHIC TEXT	402		y	y
				401	y	y
				409	y	y
Linear Dimension	33	LINEAR DIMENSION	412		y	y
				408	y	y
				406	y	y
				402	y	y
				409	y	n
Angular Dimension	34	ANGULAR DIMENSION	413		y	y
				408	y	y
				407	y	y
				402	y	y
				409	y	n
				510	y	y
Radial Dimension	35	RADIAL DIMENSION	414		y	y
				406	y	y
				402	y	y
				409	y	n
				10	y	y
				510	y	n
Diameter Dimension	37	DIAMETER DIMENSION	415		y	y
				406	y	y
				402	y	y
				409	y	n
				510	y	y

CADDS Entity Type	CADDS Entity Number	SET Entity	SET Block Number	SET Sub Block Number	Read	Write
Label	36	LABEL	416		y	y
				401	y	n
				406	y	y
				409	y	y
				402	y	y
				510	y	n
Line	3	CENTER LINE	421	4	y	n
				10	y	n
Line	3	COORDINATE AXIS SYMBOL	422		y	n
				410	y	n
Crosshatch	31	CROSSHATCHING	430	430		
					y	n
					y	n
Subfigure	80	CALLING BLOCK	500		y	y
				500	y	y
				313	y	n
				314	y	y
				315	y	n
				316	y	n
				317	y	n
				510	y	n
No Entity		HOMOGENEOUS GROUP	509		y	n
				323	n	n
				502	y	n
				508	y	n
Entities Group		GROUP	510		y	y
				323	n	n
				510	y	y
		ATTRIBUTE ASSIGNMENT	520		y	n
				520	y	n

CADDS Entity Type	CADDS Entity Number	SET Entity	SET Block Number	SET Sub Block Number	Read	Write
Vertex	94	TOPOLOGICAL VERTEX	529		y	y
				529	y	y
				515	y	y
				503	n	n
Edge	93	TOPOLOGICAL EDGE	530		y	y
				530	y	y
				515	y	y
				503	n	n
Connect Node	81	CONNECTION NODE	601		y	y
				519	n	n
				510	y	y
				510	y	y
				515	y	y
				520	y	y
				521	y	y
Nodal Line	71	CONNECTIVE LINK	602		y	y
				602	y	y
				510	y	y
				515	y	y
				519	n	n
				510	y	y
				520	y	y
				521	y	y
Nodal Subfigure	83	CONNECTIVE COMPONENT	605		y	y
				519	n	n
				510	y	y
				510	y	y
				510	y	y
				515	y	y
				520	y	y
				521	y	y
		FINITE ELEMENT MODELLING	701		n	n
		MODEL ELEMENT	702		n	n
		SUPER ELEMENT	703		n	n
		FINITE ELEMENT	704		n	n
		CALCULATION	705		n	n
		CALCULATION CASE	706		n	n
		LOADING BOUNDARY CONDITION	707		n	n
		DAMPING	708		n	n
		EIGEN ELEMENT	709		n	n
		NODE RENUMBERING	710		n	n

CADDS Entity Type	CADDS Entity Number	SET Entity	SET Block Number	SET Sub Block Number	Read	Write
SET header display		SET HEADER	9900		y	y
				9900	y	y
				9901	y	y
				9902	y	n
				9903	n	n
				9906	y	y
Definition of assembly parameters		ASSEMBLY HEADER	9901		y	y
				9905	y	y
				9920	y	y
				9930	y	y
				9901	n	n
				9902	y	y
				520	y	y
Definition of assembly part parameters		SUB-ASSEMBLY HEADER	9902		y	n
				9905	y	n
				9900	n	n
				9901	y	n
				9902	y	n
				520	y	y
Closing the assembly part		END OF SUBASSEMBLY	9997		y	n
Closing the assembly		END OF ASSEMBLY	9998		y	y
End of SET file processing		END OF SET	9999		y	y

Appendix B

System Messages

This appendix lists the system messages you may see when running PUT SET and GET SET.

An error message indicates that an entity cannot be converted.

- PUT SET Messages
- GET SET Messages

PUT SET Messages

The following system messages may appear when PUT SET is executed.

```
Actually none processed font width PCWIDTH
Assembly name to be put inside SET file ? :
CADDS part format is incompatible. (please select part format)
CADDS reading error
Dynamic memory overflow :
Error in activation of the part to be processed
Error reading solid
FACE Mip [I] : invalid boundary
No edge referenced
Problem in a line font definition
SET writing error
Subroutine PSETBLCLOSE : File close error
Subroutine PSETCVENT08 : Too big degree
Subroutine PSETCVENT16/17/18 : Error
Subroutine PSETCVENT19 : Degree is too big
Subroutine PSETCVENT19 : Not enough space in the chunk
Subroutine PSETDEALLOC : Invalid PTR pointer
Subroutine PSETDYNENT12 : Error loading curve in dynamic memory
Subroutine PSETDYNENT14 : Error loading surface in dynamic memory
Subroutine PSETDYNENT91 : Error loading solid in dynamic memory
Subroutine PSETGENC2D : Error reading uv representation of an edge
Subroutine PSETGENBOUNDARY : Error reading boundary
Subroutine PSETGENBOUNDARY : No boundary referenced
Subroutine PSETGENEDGE : Error reading edge
Subroutine PSETGENFACE : No face referenced
Subroutine PSETGENFACE : Error reading face
Subroutine PSETGENSHELL : No shell referenced
Subroutine PSETGENSHELL : Error reading shell
Subroutine PSETGENSOL : Invalid solid
Subroutine PSETGENSOL : No solid in CADDS dynamic memory
Subroutine PSETGENVERTEX : Error reading vertex
Subroutine PSETGETCVFONT : Invalid length
Subroutine PSETRDLSUBR : Not enough space in ZWORK
Subroutine PSETSBLPARM : Invalid PRMLEVEL
Subroutine PSETSETPOS : Error
Subroutine PSETSRCHPARM : Invalid data type
Subroutine PSETXSBLPARM : Invalid PRMLEVEL
Subroutine PSETWPARMS : Unknown data type
Tcyl/Srev/Rsur with none processed offset
Unknown unit: millimeter by default
WARNING : full tables, the line font definition will not be
complete
WARNING : primitive solid without 4064 subrecord
```

Messages for Convention Processsing

```
ERROR : cannot read Line Font Definition File
ERROR : cannot read Properties File
ERROR : cannot write Convention File
Property [I] not in table
```

Messages for LIBRARY Processing

```
Enter Library name ?
Maximum number of library part reached
No part in the library, try again
```

GET SET Messages

The following system messages may appear when GET SET executes.

```
Block type 215 : Invalid structure
Block type 220 : Invalid structure
Block type 301 : Invalid structure
Boundary not closed
Call block type 500 : Invalid structure
Call by name : Name is too long
Call by pointer : Not yet processed
Cpole with offset not yet processed
Curve not determined, changed in Bspline
Curve not determined, changed in Cpole
Curve not determined, changed in Nspline
Curve with offset different from zero
Curves changed in many Cpoles
Dynamic memory overflow
End of sub-assembly
ERROR : Boundary not closed (Face block number [I])
Error generating Surfacic Curve (Block number [I])
Error in activating the part to be processed
Error in the activation of the sub-assembly to be processed
Error in the interval calculation between a bspline and a SET
curve
Error loading curve in dynamic memory
Error loading surface in dynamic memory
Error opening sub-block set
Error reading data from underlying surface
Error reading definition matrix
Error reading SET
Error reading sub-assembly header block
Error reading sub-block text (401)
Error set parameters ==> QQXXXX
ERROR Subroutine GSETW96TP1 Processing
Error type of coordinates
Error writing database CADDS
Error writing edge
Error writing FACE (block [I]) : only surface and curves created
Error writing mesh subrecord
Face without underlying surface
Font 5 or 6 ==> string too big
Full tables (coordinate system)
Incomplete block or not compatible data
Invalid block type referenced
Invalid sub-assembly structure
Invalid sub-block type 3 (String)
Invalid vector
Irregular topology
Number of coefficients in the polynomial in X, Y or Z too big for
a PTC curve ( > 8 )
Number of columns or rows of the coefficient matrix in X, Y or Z
too big for a PTC surface ( > A8)
```

```
Number of rows or lines of the coefficients matrix
X,Y or Z too big for a surface (> 8)
One too many sub-blocks
Part did not get deleted, try again
Problem in the generating of the surface
Reading error assembly header block
Reading error SET header block
Relative coordinate system not yet processed
Solid not closed, create a TSURF instead of a solid
String too big
Sub-block 31 not found ==> Plane boundaries not defined
Subroutine GSETBL0020 : Curve type not yet processed
Subroutine GSETBL0020 : Problem in the curve generation
Subroutine GSETBL0101 : The number of solid shells exceeds the
limit 40
Subroutine GSETBL0102 : The number of shell edges exceeds the
limit 100 000
Subroutine GSETBL0102 : The number of the shell boundaries exceeds
the limit 100
Subroutine GSETBL0102 : The number of shell faces exceeds the
limit 10 000
Subroutine GSETBL0103 : The number of face boundaries exceeds
thelimit 101
Subroutine GSETBL0120 : The number of edges per boundary exceeds
the limit 500
Subroutine GSETBL0120 : The number of vertices exceeds the limit
500
Subroutine GSETBL0529 : The number of edges per vertex exceeds
the limit 30
Subroutine GSETBL0530 : The number of faces per edge exceeds the
limit 100
Subroutine GSETBL9901 : Not complete block
Subroutine GSETBSPLUNIF: Warning ==> CONT different of degree - 1
Subroutine GSETBSPLUNIF : Warning ==> Maximum number of segments
reached
Subroutine GSETBSPLUNIF    : Warning ==> MG different from 2
Subroutine GSETBSPLUNIF    : Warning ==> Too big degree
Subroutine GSETBSURFACE : Not enough space in the chunks
Subroutine GSETBSURFACE : Warning, CONT different from 2
Subroutine GSETBSURFACE : Warning, MG different from 2
Subroutine GSETCOOR3D : Z parameter not found or invalid
Subroutine GSETDEALLOC : PTR invalid
Subroutine GSETDL9902 : Not complete block
Subroutine GSETDYNENT12 : error filling Nspline data into
C structure
Subroutine GSETDYNENT14 : Error filling Nsurface data into
C structure
Subroutine GSETENTETE : Invalid class
Subroutine GSETENTETE : Invalid data (TYPE 9)
Subroutine GSETENTETE : Invalid parameter number or not processed
Subroutine GSETENTETE : PRMLEVEL invalid
Subroutine GSETEEVAL : Cannot be called for the time being
Subroutine GSETENTETE : Too many parameters
Subroutine GSETGENBOUNDARY : Error writing boundary
```

```
Subroutine GSETGENBOUNDARY : No boundary referenced
Subroutine GSETGENEDGE: No edge referenced
Subroutine GSETGENEDGE : No underlying curve
Subroutine GSETGENEDGE : No UV representation for an edge
Subroutine GSETGENEDGE : Warning ==> UV representation of a curve
without edge
Subroutine GSETGENEDGE_SIMPLE : Warning, No underlying curve
Subroutine GSETGENFACE : Error writing face
Subroutine GSETGENFACE : No face referenced
Subroutine GSETGENFACE : No underlying surface
Subroutine GSETGENSHELL : Error writing shell
Subroutine GSETGENSHELL : No shell referenced
Subroutine GSETGENSOL : Error writing solid
Subroutine GSETGENVERTEX : Error writing vertex
Subroutine GSETGENVERTEX : No coordinates for the vertex
Subroutine GSETGENVERTEX : No vertex referenced
Subroutine GSETGETUVBOX : Warning==> Error computing sphere
extents
Subroutine GSETLEDIM : Warning ==> Too big interpolation matrix
dimension
Subroutine GSETPUTCOLOR : error reading color parameter
Subroutine GSETPUTCOLOR : invalid color number
Subroutine GSETPUTFONT : Font invalid or not yet processed
Subroutine GSETPUTFONT : Font parameter not found
Subroutine GSETPUTPARM : Called with PRMLEVEL < = 1
Subroutine GSETPUTPARM : Full parameters stack
Subroutine GSETRDSBL21 : NCMAX invalid
Subroutine GSETRDSBL41 : NCMAX invalid
Subroutine GSETRDSBL253 : Warning, Maximum number of segments
reached
Subroutine GSETREADES : invalid block number
Subroutine GSETSBLPARM : Invalid data (TYPE 9)
Subroutine GSETSBLPARM : Invalid parameter
Subroutine GSETSBLPARM : PRMLEVEL invalid
Subroutine GSETSBL9905 : Incomplete or invalid sub-block
Subroutine GSETSBL9905B : Sub-block not complete or invalid
Subroutine GSETSBL9920 : PRMLEVEL invalid
Subroutine GSETSBL9920 : Syntax error
Subroutine GSETSBL9930 : Error sub block 9930
Subroutine GSETSBL9930 : Full TBCLNUM
Subroutine GSETSBL9930 : Unknown class
Subroutine GSETSETDR50 : Block overtakes the end of file
Subroutine GSETSETDR51S : Alphanumeric sub block, valued as not
integer
Subroutine GSETSETDR51S : Reading error
Subroutine GSETSETDR52 : Reading error
Subroutine GSETSRCHPARM : Warning ==> Data type 9 invalid
Subroutine GSETSRCHPLEV : Empty parameters stack
Subroutine GSETSRCHPLEV : Invalid level
Subroutine GSETTRSPAR : Warning, Full tables, the definition will
not be complete
Subroutine GSETTRSPAR : Warning, Offset too big
Subroutine GSETZALLOC : No more space
Surface changed in many Cpoles
```

```
Surface not determined, changed in Spoles
Surface of irregular topology, changed in Spoles
Surface type not processed yet
Too big surface for PTC
Too much solid for handle
Warning : Appearance mask on a bspline
WARNING: boundary not closed with regard to CADDS tolerance
WARNING : Boundary not closed with regard to SET tolerance
(FACE block number [I])
Warning: Invalid sub-block type
Warning : Maximum number of text line reached in sub-block 401
Warning : Not enough space for storing the text (SUB-BLOCK 401)
Warning : Parametric curve, offset not yet processed
Warning : Problem in the interval calculation between a bspline
and a SET curve
Warning : too many text fonts
Writing error from dynamic memory to CADDS
```

Messages for CSG Processing

```
Block number
ERROR : cannot open construction file
ERROR : generator curve does not lie on the XZ plane
ERROR : no other sub-block
Error opening SET block
ERROR : Problem in the curve generation
ERROR : Problem in the face generation
ERROR : Problem in the point generation
ERROR : Problem in the solid generation
ERROR : Problem in the surface generation
Error reading [I] type sub-block
Error sewing faces
WARNING : too many [I] type sub-block
WARNING : [I] type sub-block not processed
```

Messages for Repair Surface II

```
ERROR : Face block number [I] couldn't be cut, the Tsurf is
rejected
WARNING : FACE block number [I] is cut into several faces Tsurface
is generated Miptr : [I]
```

Messages for Convention Processing

```
ERROR : Attributes [I] not in convention
ERROR : Cannot read Line Font Definition File
ERROR : Cannot read Properties File
ERROR : Invalid attribute type
ERROR : Property [I] not in table
```

```
ERROR : Reference [A] not in convention
ERROR : SET code multiply defined in [A] Convention File
WARNING : Cannot read [A] Convention File
WARNING : Maximum number of conventions is reached
```

Messages for LIBRARY Processing

```
WARNING : Invalid library name, using default name
WARNING : Invalid library name, using default part name
WARNING : Invalid project name
WARNING : maximum number of library part reached
```

Index

I

L

M

P

R

S

T