# STREAM FORMAT
# GDSII RELEASE 5.1

Calma Company

# STREAM FORMAT

GDSII RELEASE 5.1

# STREAM FORMAT, GDSII RELEASE 5.1

This document consists of the contents of the file STREAM.DC. Data in this manual is subject to change during future developments. At the time of release, the date in this publication was as accurate and current as possible. However, Calma is not responsible for any damages (including consequential) caused by reliance upon materials provided.

## NOTICE

Calma has prepared this manual for use by Calma personnel and customers. Receipt of this document shall be deemed to be an acceptance of the conditions specified herein.



Printed in U.S.A. August 1984

STREAM FORMAT is a canonical output format for GDS II data. Libraries which are preserved in this format can be easily transferred to other systems for processing, or can be archived since this format is independent of future internal modifications of the GDS II system.

**********

ADDITIONS TO STREAM FORMAT

26 JUNE 1981

PLEX records have been put into each element after the ELFLAGS record.

Two new element types have been added: BOX and NODE.

19 JANUARY 1982

Two new records, TAPENUM and TAPECODE, may be used in conjunction with LIBNAME for putting STREAM format onto multiple reels of tape. See below.

24 SEPTEMBER 1982

The records ELFLAGS, BGNEXTN, ENDEXTN, and STRCLASS have been defined.

12 DECEMBER 1983

For GDSII Release 5.1 and above:
New optional records have been added for Filtered Stream Format enhancement: FORMAT, MASK, ENDMASKS.

13 JULY 1984

Corrected syntax definition of path to include optional extension records.

**********

## STREAM SYNTAX

The following is a psuedo-Bachus Naur representation of the STREAM syntax. Square brackets (that is "[]") denote an entity which can occur zero or one time. Braces (that is "{}") mean "pick one of the entities within the braces." Braces with a star (that is "{}*") mean that the entities within the braces can occur an arbitrary number of times. Braces with a plus (that is "{}+") mean that at least one of the entities within the braces must be present.

<library>::= HEADER BGNLIB LIBNAME [ REFLIBS ] [ FONTS ] [ ATTRTABLE ]
[ STYPTABLE ] [ GENERATIONS ] [ <FormatType> ] UNITS
{ <structure> }* ENDLIB

<FormatType>::= FORMAT | FORMAT { MASK }+ ENDMASKS

<structure>::= BGNSTR STRNAME [ STRCLASS ] [ STRTYPE ] { <element> }*
ENDSTR

<element>::= { <boundary> | <path> | <sref> | <aref> | <text> | <node> |
<box> } [ ELKEY ] { <property> }* ENDEL

<boundary>::= BOUNDARY [ ELFLAGS ] [ PLEX ] LAYER DATATYPE XY

<path>::= PATH [ ELFLAGS ] [ PLEX ] LAYER DATATYPE [ PATHTYPE ]
[ WIDTH ] [ BGNEXTN ] [ ENDEXTN ] XY

<sref>::= SREF [ ELFLAGS ] [ PLEX ] SNAME [ <strans> ] XY

<aref>::= AREF [ ELFLAGS ] [ PLEX ] SNAME [ <strans> ] COLROW XY

<text>::= TEXT [ ELFLAGS ] [ PLEX ] LAYER <textbody>

<node>::= NODE [ ELFLAGS ] [ PLEX ] LAYER NODETYPE XY

<box>::= BOX [ ELFLAGS ] [ PLEX ] LAYER BOXTYPE XY

<textbody>::= TEXTTYPE [ PRESENTATION ] [ PATHTYPE ] [ WIDTH ]
[ <strans> ] XY STRING

<strans>::= STRANS [ MAG ] [ ANGLE ]

<property>::= PROPATTR PROPVALUE

## MULTI-REEL STREAM FORMAT

STREAM format can be put onto multiple reels of tape. The first tape should end with the records TAPENUM, TAPECODE, and LIBNAME, in that order. Each subsequent tape should begin with those same records, in that order, and should end with the record TAPENUM. STREAM tapes must contain only complete STREAM records, i.e. no STREAM record should begin on one tape and continue on the next tape.

The records TAPENUM, TAPECODE, and LIBNAME, used in this manner, are used only for identification of the tapes and are not incorporated into the library in any way. The records TAPENUM and TAPECODE must be used only as described, and must not appear anywhere else in the STREAM file. LIBNAME, of course, also occurs normally as the third record of a STREAM file. Tapes may end after any record in STREAM format.

In the spirit of the psuedo Bachus Naur representation above, here is a representation of multi-reel STREAM tapes:

<multi-reel STREAM tape>::= <tape1> { <continuation tapes> }+

<tape1>::= HEADER { <complete records in STREAM syntax> }*
<tape-id>

<continuation tapes>::= <tape-id> { <complete records continuing in STREAM syntax> }+ TAPENUM

<tape-id>::= TAPENUM TAPECODE LIBNAME

The entire concatenation of STREAM records, without the tape-id groups and TAPENUMs, should conform to the previously described STREAM syntax.

## RECORD DESCRIPTION

The output file is composed of variable length records. The first two bytes contain a count (in eight-bit bytes) of the total record length (count bytes are included in the count). The third byte is the record type, and the fourth byte describes the type of data contained within the record.

If the output file is magnetic tape, then the records of the library are written out in 2048 byte physical blocks. Records may overlap physical block boundaries – i.e., a record is not required to be wholly contained in a single physical block.

A "null word" consists of two consecutive zero bytes. Null words should be used to fill the space between the last record of a library and the end of its physical block, or between the last record of a tape in a multi-reel STREAM file and the end of its physical block.

Here are the possible data types (number for type occurs in fourth byte of the record):

| DATATYPE | VALUE |
|---|---|
| No Data Present | 0 |
| Bit Array | 1 |
| Double Byte Signed Integer | 2 |
| Four Byte Signed Integer | 3 |
| Four Byte Real | 4 |
| Eight Byte Real | 5 |
| ASCII String | 6 |

Records are always an even number of bytes long. If a character string is an odd number of bytes long it is padded with a null character.

The following are record types and a brief description of each:
[rrdd] The record type and data type in Hex.

0 HEADER [0002] — Double Byte Signed Integer

Contains two bytes of data, the version number.
Prior to release 3.0 of GDS II, the version number was 0.
Starting with release 3.0, the version number is 3.

1 BGNLIB [0102] — Double Byte Signed Integer

Contains last modification time (two bytes for year, two bytes for month, two bytes for day, two bytes for hour, two bytes for minute, and two bytes for second) as well as time of last access (same as creation time format) and marks beginning of library.

2 LIBNAME [0206] — ASCII String

Contains a string which is the library name. The library name must adhere to CDOS file name conventions, for length and valid characters.

3 UNITS [0305] — Eight Byte Real

Contains two double precision (eight byte) real numbers. The first is the size of a database unit in user units. The second is the physical size of a data base unit in meters.
Typically, the first number will be less than one, since it is recommended to use more than one database unit per user unit.
To retrieve the physical size of a user unit in meters, divide the second number by the first.

4 ENDLIB [0400] — No Data Present

Marks the end of a library.

5 BGNSTR [0502] — Double Byte Signed Integer

Contains creation time of structure (two bytes for year, two bytes for month, two bytes for day, two bytes for hour, two bytes for minute, and two bytes for second) and contains last modification time of the structure (same format as creation time) and marks the beginning of a structure.

6 STRNAME [0606] — ASCII String

Contains a string which is the structure name. A structure name may be up to thirty-two characters long, and legal characters are upper and lower case letters, digits, underscore (_), and currency ($).

7 ENDSTR [0700] — No Data Present

Marks the end of a structure.

8 BOUNDARY [0800] — No Data Present

Marks the beginning of a boundary element.

9 PATH [0900] — No Data Present

Marks the beginning of a path element.

10 SREF [0A00] — No Data Present

Marks the beginning of an sref (structure reference) element.

11 AREF [0B00]

No Data Present

Marks the beginning of an aref (array reference) element.

12 TEXT [0C00]

No Data Present

Marks the beginning of a text element.

13 LAYER [0D02]

Double Byte Signed Integer

Contains two bytes which specify layer. The value of the layer must be in the range of 0 to 63.

14 DATATYPE [0E02]

Double Byte Signed Integer

Contains two bytes which specify datatype. The value of the datatype must be in the range of 0 to 63.

15 WIDTH [0F03]

Four Byte Signed Integer

Contains four bytes which specify the width of a path in data base units, or the width of text lines. A negative value for width means that the width is absolute, i.e. it is not affected by the magnification factor of any parent reference.

16 XY [1003]

Four Byte Signed Integer

Contains an array of XY coordinates in database units. Each X or Y coordinate is four bytes long.

PATH and BOUNDARY may have up to 200 pairs of coordinates. A PATH must have at least two, and a BOUNDARY at least four pairs of coordinates. The first and last point of a BOUNDARY must coincide.

A TEXT or SREF element must have only one pair of coordinates.

An AREF has exactly three pairs of coordinates, which specify the orthogonal array lattice. In an AREF the first point is the array reference point. The second point locates a postion which is displaced from the reference point by the inter-column spacing times the number of columns. The third point locates a position which is displaced from the reference point by the inter-row spacing times the number of rows.

A NODE may have from one to fifty pairs of coordinates.

A BOX currently must have 5 pairs of coordinates with the first and last points coinciding.

17 ENDEL [1100] — No Data Present

Marks the end of an element.

18 SNAME [1206] — ASCII String

Contains the name of a referenced structure. See also STRNAME.

19 COLROW [1302] — Double Byte Signed Integer

Contains four bytes. The first two bytes contain the number of columns in the array. The third and fourth bytes contain the number of rows. The number of columns and the number of rows may not exceed 32,767 (decimal) and are non-negative.

20 TEXTNODE [1400] — No Data Present

Marks the beginning of a text node.
[This record type is not currently used.]

21 NODE [1500]

No Data Present

Marks the beginning of a node.

22 TEXTTYPE [1602]

Double Byte Signed Integer

Contains two bytes representing texttype. The value of the texttype must be in the range 0 to 63.

23 PRESENTATION [1701]

Bit Array

Contains one word (two bytes) of bit flags for text presentation. Bits eleven and twelve, taken together as a binary number, specify the font (00 means font 0, 01 means font 1, 10 means font 2, and 11 means font 3). The thirteenth and fourteenth bits are used to specify the vertical presentation (00 means top, 01 means middle, and 10 means bottom). The fifteenth and sixteenth bits specify the horizontal presentation (00 means left, 01 means center, and 10 means right). Bits one through ten are reserved for future use and must be cleared.

24 SPACING

(Discontinued)

25 STRING [1906]

ASCII String

Contains a character string for text presentation, up to five-hundred-twelve (512) characters long.

26 STRANS [1A01]

Bit Array

Contains two bytes of bit flags for graphic presentation. The first (high order, or leftmost) bit specifies reflection. If it is set, then reflection over the X-axis is applied before angular rotation. For AREFs, the entire array lattice is

reflected, with the individual array elements rigidly attached.
The fourteenth bit flags absolute magnification.
The fifteenth bit flags absolute angle.
The sixteenth (low order, or rightmost) bit and all remaining bits are reserved for future use and must be cleared.

27 MAG
[1B05]

Eight Byte Real

Contains a double precision real number (eight bytes) which is the magnification factor.

28 ANGLE
[1C05]

Eight Byte Real

Contains a double precision real number (eight bytes) which is the angular rotation factor, measured in degrees and in counter clockwise direction.
For an AREF, the ANGLE rotates the entire array lattice (with the individual array elements rigidly attached) about the array reference point.

29 UINTEGER

User Integer (No longer used)

In Release 2.0.0 only, User Integer data could be made part of any element. This consists of a sequence of up to 32 two-byte signed integers. In the stream format it was an optional block of data immediately after the USTRING block. Discontinued in Release 2.0.1, in anticipation of the more general user-property capability of Release 3. See also PROPATTR and PROPVALUE below.

30 USTRING

Character String (No longer used)

Contains character string data. If this record is not present then it is the null string. Formerly called CSD (character string data).
User string data was used in release 1 and 2. Starting in

release 3.0, user string data was replaced by the more general user properties, see PROPATTR and PROPVALUE.

31 REFLIBS [1F06]

ASCII String

Contains names of the reference libraries. The name for the first reference library starts the data and the name of the second library starts at byte number forty-five (45 decimal). If either library is not named its place is filled with nulls.

32 FONTS [2006]

ASCII String

Contains names of textfont definition files. The name of Font Zero starts the record, followed by the remaining three fonts. Each name is forty-four bytes long and is null if there is no corresponding textfont definition. Each name is padded with nulls if it is shorter than forty-four bytes.

33 PATHTYPE [2102]

Double Byte Signed Integer

This record contains a value of 0 for square-ended paths that end flush with their endpoints, 1 for round-ended paths, and 2 for square-ended paths that extend a half-width beyond their endpoints. Pathtype 4 (for the STICKS product only) signifies a path with variable square-end extensions (see records 48 and 49). If not specified, a Pathtype of 0 is assumed.

34 GENERATIONS [2202]

Double Byte Signed Integer

This record contains a positive count of the number of copies of deleted or backed-up structures to retain. This number must must be at least 2 and not more than 255. If the GENERATIONS record is not present, a value of 3 is assumed.

35 ATTRTABLE [2306]
ASCII String

Contains the name of the attribute definition file. Maximum size is 44 bytes.

36 STYPTABLE [2406]
ASCII String (Unreleased feature)

37 STRTYPE [2502]
Double Byte Signed Integer (Unreleased feature)

38 ELFLAGS [2601]
Bit Array

Contains two bytes of bit flags. Bit 16 (the rightmost bit) specifies TEMPLATE data. Bit 15 specifies EXTERNAL data. All other bits are currently unused and must be cleared to 0.

39 ELKEY [2703]
Four Byte Signed Integer (Unreleased feature)

40 LINKTYPE [28]
Two Byte Signed Integer (Unreleased feature)

41 LINKKEYS [29]
Four Byte Signed Integer (Unreleased feature)

42 NODETYPE [2A02]
Double Byte Signed Integer

Contains two bytes which specify nodetype. The value of the nodetype must be in the range of 0 to 63.

43 PROPATTR
[2B02]

Double Byte Signed Integer

Contains two bytes which specify the attribute number. The attribute number is an integer from 1 to 127. Attribute number 127 is reserved for the "user string" or "CSD" property, which existed previous to release 3.0. Attribute number 126 is reserved for the "user integer" property, which existed previous to release 3.0. (User string and user integer data in previous releases becomes converted to property data having attribute number 127 and 126 by the stream format input program INFORM.)

44 PROPVALUE
[2C06]

ASCII String

Contains the string value associated with the attribute named in the preceding PROPATTR record. Maximum length is 126 characters. The attribute-value pairs associated with any one element must all have distinct attribute numbers. Also, there is a limit on the total amount of property data that may be associated with any one element: the total length of all the strings, plus twice the number of attribute-value pairs, must not exceed 128 (or 512 if the element is an SREF or AREF or NODE).

45 BOX
[2D00]

No Data Present

Marks the beginning of a box element.

46 BOXTYPE
[2E02]

Double Byte Signed Integer

Contains two bytes which specify boxtype. The value of the boxtype must be in the range of 0 to 63.

47 PLEX
[2F03]

Four Byte Signed Integer.

A unique positive number which is common to all the elements of the PLEX of which this element is a member.
The head of the plex is flagged by setting the 8th bit (the leftmost bit is the first and the rightmost bit is the 32nd). Thus, plex numbers should be small enough to occupy only the rightmost 24 bits.

48 BGNEXTN
[3003]

Four Byte Signed Integer

Applies to Pathtype 4 (currently in the STICKS product only). Contains four bytes which specify in data base units the extension of a path outline beyond the first point of the path.

49 ENDEXTN
[3103]

Four Byte Signed Integer

Applies to Pathtype 4 (currently in the STICKS product only). Contains four bytes which specify in data base units the extension of a path outline beyond the last point of the path.

50 TAPENUM
[3202]

Double Byte Signed Integer

Contains two bytes which specify the number of the current reel of tape for a multi-reel STREAM file. For the first tape, the TAPENUM is 1; for the second tape, the TAPENUM is 2; etc.

51 TAPECODE
[3302]

Double Byte Signed Integer

Contains twelve bytes. This is a unique six-integer code which is common to all the reels of a multi-reel STREAM file. It is used to verify that the correct reels are being read in.

52 STRCLASS
[3401]

Two Byte Bit Array

This is currently only for Calma internal use with STICKS structures. If Stream tapes are produced by non-Calma programs, then this record should either be omitted or cleared to 0.

53 RESERVED
[3503]

Four Byte Signed Integer.

Reserved for future use. This record type previously used for NUMTYPES which was never used and not required.

54 FORMAT
[3602]

Double Byte Signed Integer.
Optional.
Defines the format type of a stream tape in two bytes.
The two possible values are:
0 ARCHIVE format,
1 FILTERED format.
An ARCHIVE stream file contains elements for all the layers and data types. It is created with the OUTFORM command.
In an ARCHIVE stream file, the FORMAT record is followed immediately by the UNITS record. A file which does not have the FORMAT record is assumed to be an ARCHIVE file.
A FILTERED stream file contains only the elements on the layers and with the data types specified by the user during execution of the STREAMOUT command. The list of layers and data types specified for the STREAMOUT command follows the FORMAT record, in MASK records. The mask records are terminated with the ENDMASKS record. At least one MASK record must immediately follow the FORMAT record. The FILTERED stream file is created with the STREAMOUT command.
See MASK and ENDMASKS, below.

55 MASK
[3706]

ASCII String.
Required for FILTERED format.
Present only in FILTERED stream file.

Contains the list of layers and data types specified by the user for the STREAMOUT command. At least one MASK record must follow the FORMAT record. More than one MASK records may follow the FORMAT record. The last MASK record is followed by the ENDMASKS record. See FORMAT above and ENDMASKS below. In the list, data types are separated from the layers with a semi-colon. Individual layers or data types are separated with a space. A range of layers or data types is specified with a dash. Example list:

1 5-7 10 ; 0-63

56 ENDMASKS
[3800]

No Data Present.
Required for FILTERED format.
Present only in FILTERED stream file.

Terminates the MASK records. The ENDMASKS record must follow the last MASK record. It is immediately followed by the UNITS record.
See FORMAT and MASK above.

/*

# APPENDIX

/* STREAM RECORDS HAVE TWO BYTES WITH TOTAL BYTE LENGTH OF
RECORD, THEN ONE BYTE FOR RECORD TYPE AND ONE BYTE OF
DATA TYPE. THE REST OF THE RECORD IS DATA. RECORD
LENGTHS ARE ALWAYS AN EVEN NUMBER OF BYTES.
AND REMEMBER...STREAM RECORDS SPAN PHYSICAL BLOCKS.*/

/* Stream Format Types. These are used to implement the two versions of
Stream Format: Archive (OUTFORM) and Filtered (STREAMOUT).
The Archive Stream Format is generated by OUTFORM.
The Filtered Stream Format is generated by STREAMOUT. */

---

## MACHINE REPRESENTATION OF DATA TYPES

A word consists of 16 bits, numbered 1 to 16, left to right (high to low order)

2-byte integer (DGL Integer ) = 1 word twos-complement representation
4-byte integer (DGL Integer (2)) = 2 word twos-complement representation
4-byte real (DGL Real ) = 2 word floating point representation
8-byte real (DGL Real (4) ) = 4 word floating point representation

Two-byte integer, Four-byte integer, Four-byte real, and Eight-byte real
all represent the value zero by setting all bits to zero.

For all non-zero values:
A floating point number is made up of three parts: the sign, the exponent, and
the mantissa. The value of a floating point number is defined to be:
(MANTISSA) X (16 RAISED TO THE TRUE VALUE OF THE EXPONENT FIELD).
The exponent field is held in Excess-64 representation, i.e the seven-bit
field shows a number that is 64 greater than the actual exponent.
The mantissa is always a positive fraction >=1/16 and <1. The "binary point"
can be thought of as being just to the left of bit 9. Bit 9, then, represents
the value 1/2, bit 10 represents 1/4, etc.
In order to keep the mantissa in the range of 1/16 to 1, the results of

floating point arithmetic are "normalized". Normalization is a process whereby the mantissa is shifted left one HEX digit at a time until the high-order FOUR bits represent a non-zero quantity. For every hex digit shifted, the exponent is decreased by one. Since the mantissa is shifted four bits at a time, it is possible for the high-order three bits of a normalized mantissa to be zero.

Representation:

2-byte integer:
SMMMMMMM MMMMMMM
4-byte integer:
SMMMMMMM MMMMMMMM MMMMMMMM MMMMMMMM
4-byte real:
SEEEEEEE MMMMMMMM MMMMMMMM MMMMMMMM
8-byte real:
SEEEEEEE MMMMMMMM MMMMMMMM MMMMMMMM
MMMMMMMM MMMMMMMM MMMMMMMM MMMMMMMM

Examples:

2-byte integer:
00000000 00000001 = 1
00000000 00000010 = 2
00000000 10001001 = 137
11111111 11111111 = -1
11111111 11111110 = -2
11111111 01110111 = -137

4-byte integer:
00000000 00000000 00000000 00000001 = 1
00000000 00000000 00000000 00000010 = 2
00000000 00000000 00000000 10001001 = 137
11111111 11111111 11111111 11111111 = -1
11111111 11111111 11111111 11111110 = -2
11111111 11111111 11111111 01110111 = -137

4-byte real:
01000001 00010000 00000000 00000000 = 1
01000001 00100000 00000000 00000000 = 2
01000001 00110000 00000000 00000000 = 3
11000001 00010000 00000000 00000000 = -1

Note that the 7-bit exponent field = 65, meaning that the actual exponent is 65-64=1

11000001 00100000 00000000 00000000 = -2
11000001 00110000 00000000 00000000 = -3

01000000 10000000 00000000 00000000 = .5
01000000 10011001 10011001 10011001 = .6
01000000 10110011 00110011 00110011 = .7
01000001 00011000 00000000 00000000 = 1.5
01000001 00011001 10011001 10011001 = 1.6
01000001 00011011 00110011 00110011 = 1.7

01000001 00010000 00000000 00000000 = 1
01000001 10100000 00000000 00000000 = 10
01000010 01100100 00000000 00000000 = 100
01000011 00111110 10000000 00000000 = 1000
01000100 00100111 00010000 00000000 = 10000
01000101 00011000 01101010 00000000 = 100000

The representation of the negative values of real numbers is EXACTLY the same as the positive, EXCEPT that the highest order bit is 1, not 0.

In the eight-byte real representation, the first four bytes are EXACTLY the same as in the four-byte real representation. The last four bytes contain additional "binary places" for more resolution.

---