STREAM FORMAT

GDS II Release 5.2

Stream Format, GDS II Release 5.2

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.

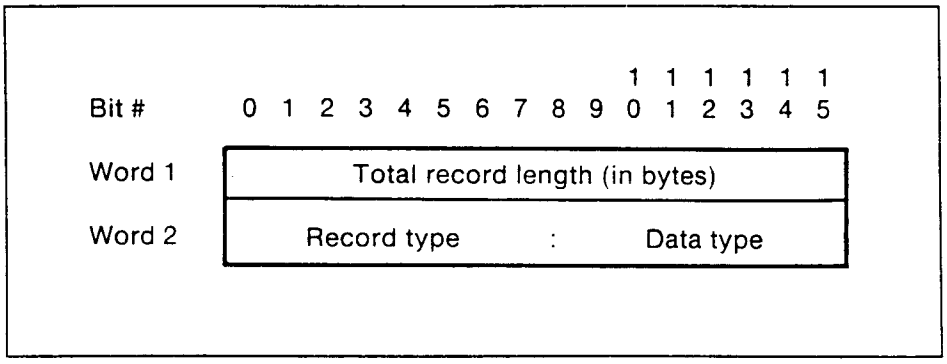
Printed in U.S.A. November, 1985

Stream format is a canonical output format for GDS II data. Stream format is the format written by OUTFORM and STREAMOUT and read by INFORM. Libraries preserved in this format can be easily transferred to other systems for processing. Stream format is upward compatible between releases. Libraries archived under an old release will always be readable by newer releases. For this reason, libraries preserved in Stream format can be archived.

Record Description

The output file is composed of variable length records. The minimum record length is four bytes. Records can be infinitely long. The first four bytes of a record are the header. The first two bytes of the header contain a count (in eight-bit bytes) of the total record length. The count tells you where one record ends and another begins. The next record begins immediately after the last byte included in the count.

The third byte of the header is the record type. The fourth byte of the header describes the type of data contained within the record. The fifth through last bytes of a record are data. The following figure shows a typical record header.



Typical Record Header

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

A null word consists of two consecutive zero bytes. Use null words to fill the space between:

- the last record of a library and the end of its physical block, or
- the last record of a tape in a multi-reel Stream file and the end of its physical block.

Data Type Description

The following table lists the possible data types and their values. You can find the type value in the fourth byte of the record.

Stream Data Types

Data Type	Value
No Data Present	0
Bit Array	1
Two-Byte Signed Integer	2
Four-Byte Signed Integer	3
Four-Byte Real	4 (at present, this data type is not used)
Eight-Byte Real	5
ASCII String	6

The following paragraphs describe the data types listed in the table.

Remember: A word consists of 16 bits, numbered 0 to 15, left to right.

- *Bit Array (1):*

  A bit array is a word which uses the value of a particular bit or group of bits to represent data. A bit array allows one word to represent a number of simple pieces of information.

- *Two-Byte Signed Integer (2)*

  2-byte integer = 1 word 2s-complement representation

  The range of two-byte signed integers is -32,768 to 32,767.

  Following is a representation of a two-byte integer, where S is the sign and M is magnitude.

  SMMMMMMM MMMMMMM

Following are examples of Two-Byte Integers:

```
00000000 00000001 = 1
00000000 00000010 = 2
00000000 10001001 = 137
11111111 11111111 = -1
11111111 11111110 = -2
11111111 01110111 = -137
```

- *Four-Byte Signed Integer (3):*

  4-byte integer = 2 word 2s-complement representation

  The range of four-byte signed integers is -2,147,483,648 to 2,147,483,647.

  Following is a representation of a four-byte integer, where `S` is the sign and `M` is magnitude.

  ```
  SMMMMMMM MMMMMMMM MMMMMMMM MMMMMMMM
  ```

  Examples of four-byte integers:

  ```
  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
  ```

- *Four-Byte Real (4) and Eight-Byte Real (5)*

  4-byte real = 2 word floating point representation

  8-byte real = 4 word floating point representation

  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 (bits 1-7) is in Excess-64 representation. 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. For a four-

byte real, the mantissa is bits 8-31. For an eight-byte real, the mantissa is bits 8-63.

- The binary point is just to the left of bit 8.
- Bit 8 represents the value 1/2, bit 9 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 its left 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 left three bits of a normalized mantissa to be zero. A zero value, also called *true zero*, is represented by a number with all bits zero.

Following are representations and examples of 4-byte and 8-byte reals. 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.

4-byte real:

```
SEEEEEEE MMMMMMMM MMMMMMMM MMMMMMMM
```

8-byte real:

```
SEEEEEEE MMMMMMMM MMMMMMMM MMMMMMMM
MMMMMMMM MMMMMMMM MMMMMMMM MMMMMMMM
```

Examples of 4-byte real:

Note: In the first six lines of the following example, the 7-bit exponent field = 65. The actual exponent is 65-64=1.

```
01000001 00010000 00000000 00000000 = 1
01000001 00100000 00000000 00000000 = 2
01000001 00110000 00000000 00000000 = 3
11000001 00010000 00000000 00000000 = -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
00000000 00000000 00000000 00000000 = 0
```

- *ASCII String (6)*

  A collection of ASCII characters, where each character is represented by one byte. All odd length strings must be padded with a null character (the number zero) and the byte count for the record containing the ASCII string must include this null character. Stream read-in programs must look for the null character and decrease the length of the string by 1 if the null character is present.

Record Types

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.

Following are records and a brief description of each, where the first two numbers in brackets are the record type and the last two numbers in brackets are the data type. All record numbers are expressed in hexidecimal.

0 HEADER
[0002]

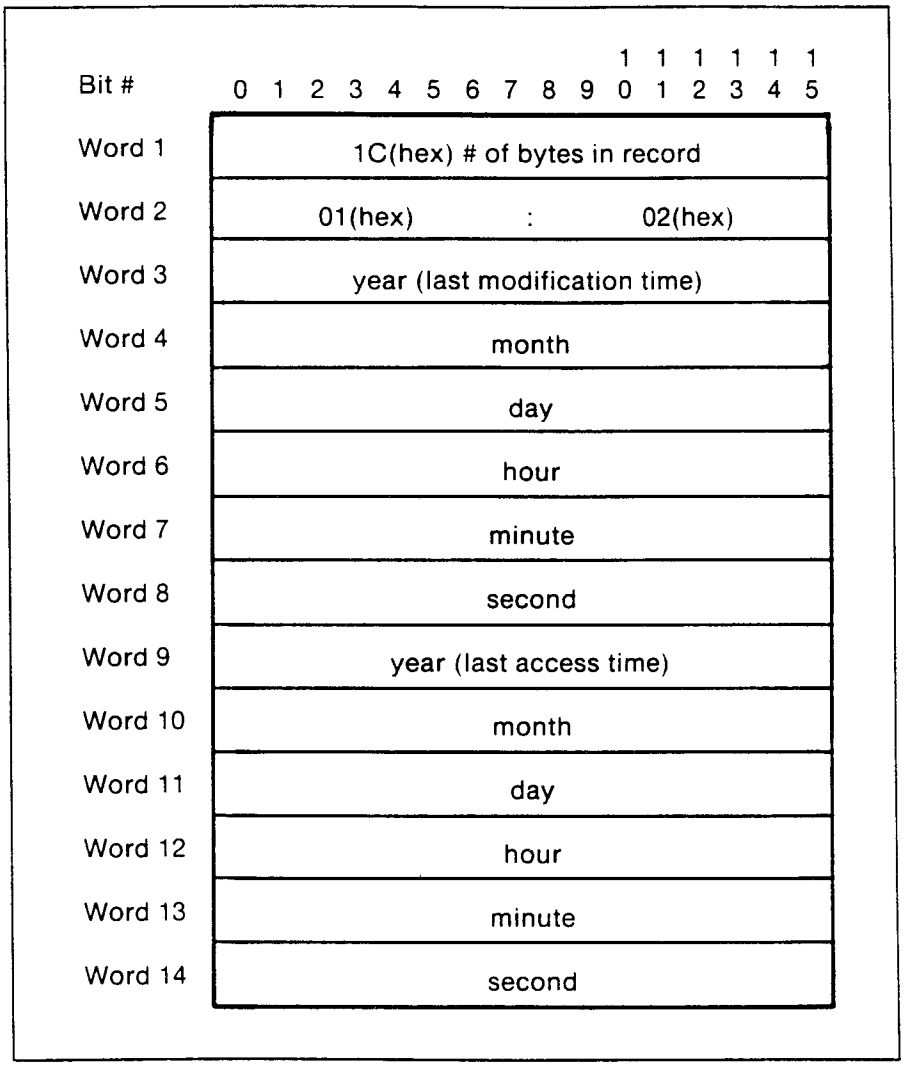
Two-Byte Signed Integer

Contains two bytes of data which represents the version number. Prior to GDS II Release 3.0, the version number was zero. For Release 3.0, the version number is three. Likewise, for Release 4.0 the version number is four, and for Release 5.0, the version number is five.

1 BGNLIB
[0102]

Two-Byte Signed Integer

Contains last modification time of library (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 format) and marks beginning of library. Refer to the figure.



A BGNLIB Record

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. The library name may include the file extension (.DB in most cases).

3 UNITS
[0305]

Eight-Byte Real

Contains two eight-byte real numbers. The first is the size of a database unit in user units. The second is the size of a database unit in meters. For example, if your library was created with the default units (user unit = 1 micron and 1000 database units per user unit), then the first number would be .001 and the second number would be 1E-9. Typically, the first number is less than one, since you use more than one database unit per user unit.

To calculate the 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]

Two-Byte Signed Integer

Contains creation time and last modification time of a structure (in the same format as for the BGNLIB record) 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. Legal structure name characters are:

- A through Z,
- a through z,
- 0 through 9,
- underscore (_),

- question mark (?), and
- dollar sign ($).

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]

Two-Byte Signed Integer

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

14 DATATYPE
[0E02]

Two-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 or text lines in data base units. 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. If omitted, 0 is assumed.

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.

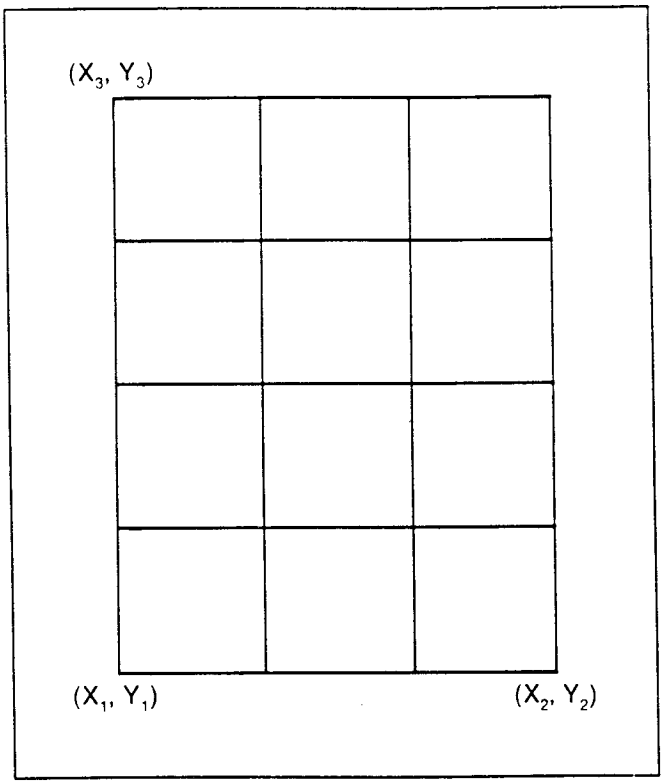
Path and boundary elements may have up to 200 pairs of coordinates. A path must have at least two, and a boundary at least four pairs of coordinates. The first and last point of a boundary must coincide.

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 position 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. For an example of an array latice, see Figure B-3.

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

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



An Array Lattice

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]

Two-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. Neither the number of columns nor the number of rows may exceed 32,767 (decimal) and are non-negative.

20 TEXTNODE
[1400]

No Data Present

Marks the beginning of a text node. (Not currently used.)

21 NODE
[1500]

No Data Present

Marks the beginning of a node.

22 TEXTTYPE
[1602]

Two-Byte Signed Integer

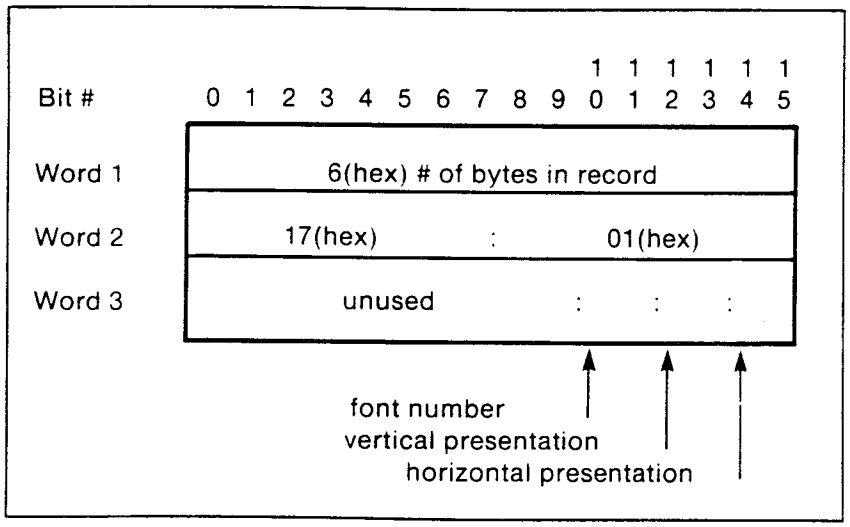
Contains two bytes representing texttype. The value of the texttype must be in the range 0 to 63.

23 PRESENTATION
[1701]

Bit Array

Contains one word (two bytes) of bit flags for text presentation. Bits ten and eleven, 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 twelfth and thirteenth bits are used to specify the vertical presentation (00 means top, 01 means middle, and 10 means bottom). The fourteenth and fifteenth bits specify the horizontal presentation (00 means left, 01 means center, and 10 means right). Bits zero through nine are reserved for future use and must be cleared. If this record is omitted, then top-left justification and font 0 are assumed.

The following figure shows a presentation record.



A PRESENTATION Record

24 SPACING

Discontinued

25 STRING
[1906]

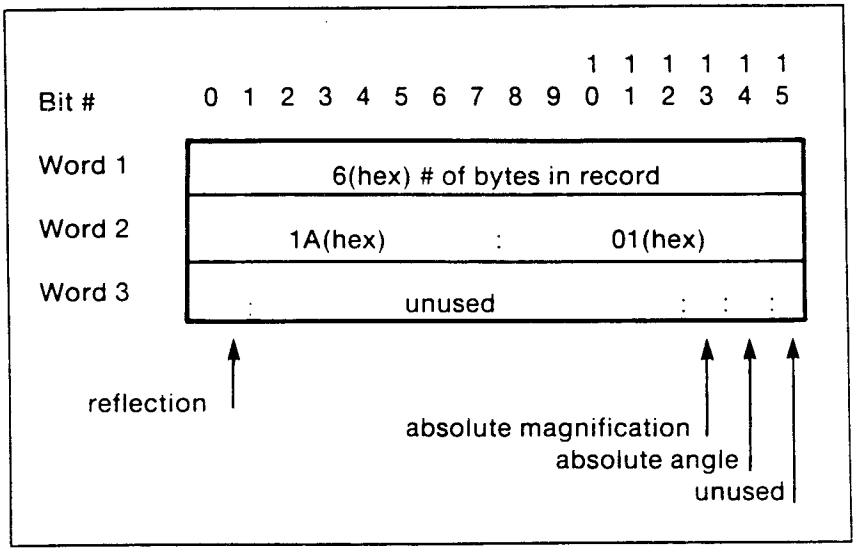
ASCII String

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 SREF, AREF, and text transformation. The zeroth (or leftmost) bit specifies reflection. If it is set, then reflection about the X-axis is applied before angular rotation. For AREFs, the entire array lattice is reflected, with the individual array elements rigidly attached. The thirteenth bit flags absolute magnification. The fourteenth bit flags absolute angle. The fifteenth (or rightmost) bit and all remaining bits are reserved for future use and must be cleared. If this record is omitted, then the element is assumned to have no reflection and its magnification and angle are assumed to be non-absolute. The following figure shows an **STRANS** record.



An STRANS Record

27 MAG
[1B05]

Eight-Byte Real

Contains a double precision real number (eight bytes) which is the magnification factor. If omitted, a magnification of 1 is assumed.

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. If this record is omitted, an angle of 0 degrees is assumed.

29 UINTEGER

User Integer (No longer used)

In Release 2.0 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 3.0. See also **PROPATTR** and **PROPVALUE.**

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 character string data.

User string data was used in Release 1.0 and 2.0. Starting with Release 3.0, **PROPATTR** and **PROPVALUE** replace user string data.

31 REFLIBS
[1F06]

ASCII String

Contains names of the reference libraries. This record must be present if there are any reference libraries bound to the current library. This record cannot be present otherwise.

The name for the first reference library starts at byte zero and the name of the second library starts at byte number forty-five (45 decimal). The reference library names may include directory

specifiers (separated with ":") and an extension (separated with "."). If either library is not named, its place is filled with nulls.

32 FONTS
[2006]

ASCII String

Contains names of textfont definition files. This record must be present if any of the 4 fonts have a corresponding textfont definition file. This record must not be present if none of the fonts have a texfont definition file. 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. The texfont definition file names may include directory specifiers (separated with ":") and an extension (separated with ".").

33 PATHTYPE
[2102]

Two-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 CustomPlus product only) signifies a path with variable square-end extensions (see records 48 and 49). If not specified, a Pathtype of 0 is assumed. Figure B-6 shows the pathtypes.

34 GENERATIONS
[2202]

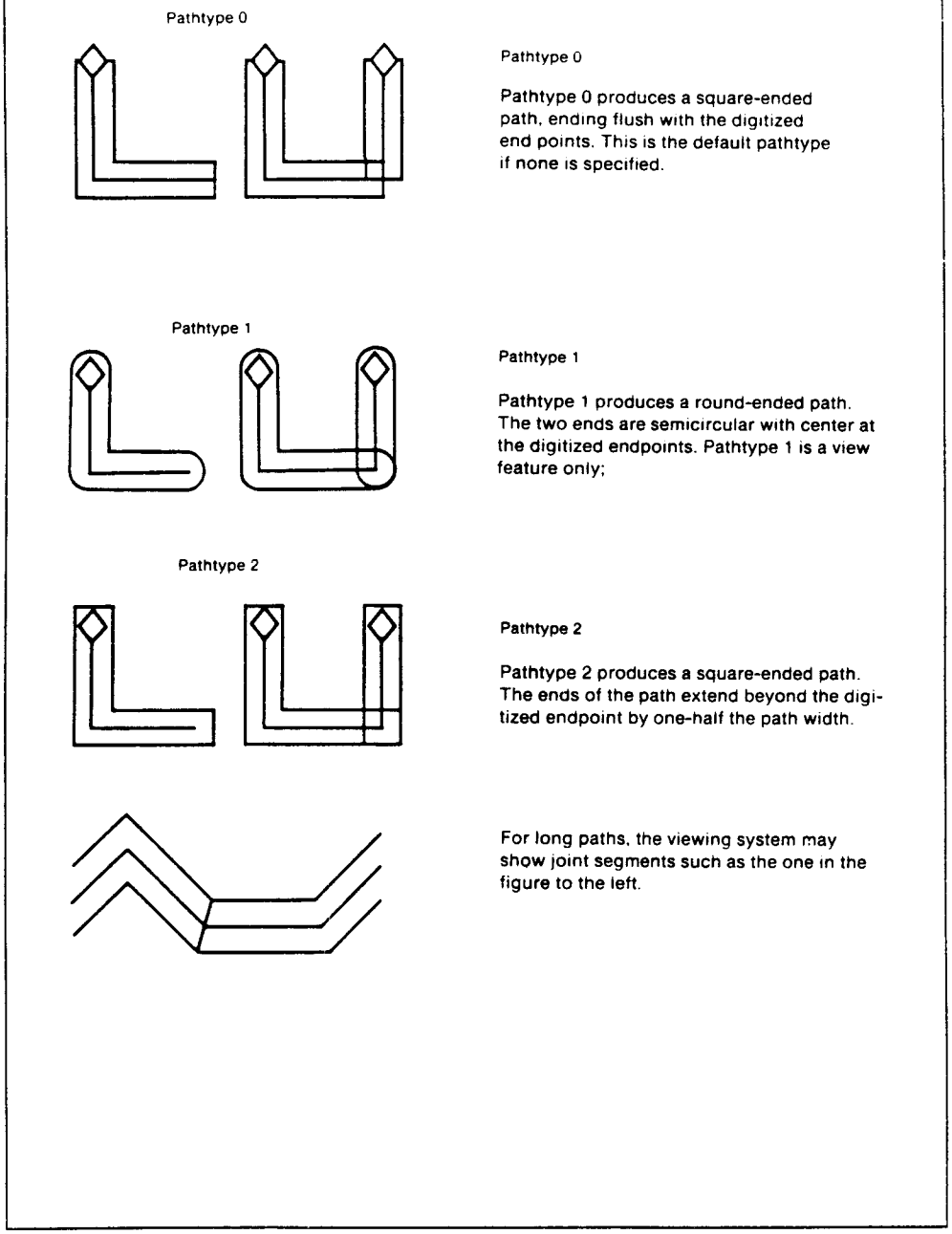
Two-Byte Signed Integer

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 99. 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. This record is present only if there is an attribute definition file bound to the library. The attribute definition file name may include directory specifiers (separated with ":") and an extension (separated with "."). Maximum size is 44 bytes.



Pathtypes

36 STYPTABLE ASCII String (Unreleased feature)
[2406]

37 STRTYPE
[2502]

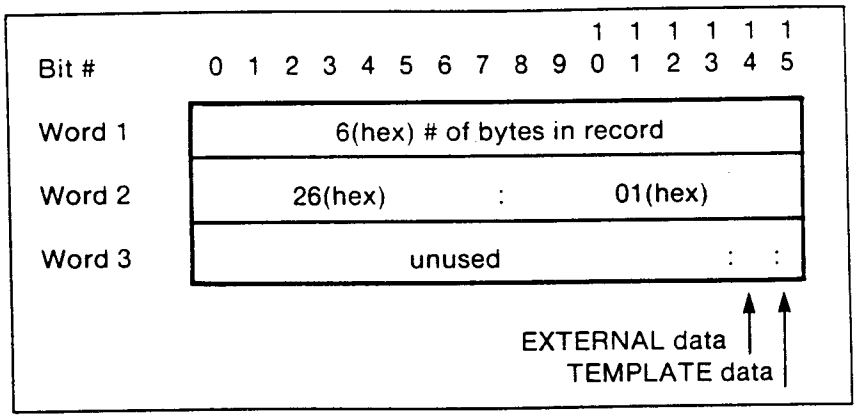
Two-Byte Signed Integer (Unreleased feature)

38 ELFLAGS
[2601]

Bit Array

Contains two bytes of bit flags. Bit 15 (the rightmost bit) specifies Template data. Bit 14 specifies External data (also referred to as Exterior data). All other bits are currently unused and must be cleared to 0. If this record is omitted, then all bits are assumed to be 0.

For additional information on Template data, consult the *GDS II Reference Manual.* For additional information on External data, consult the *CustomPlus User's Manual.* The following figure shows an **ELFLAGS** record.



An ELFLAGS Record

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]

Two-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]

Two-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 prior to release 3.0. Attribute number 126 is reserved for the "user integer" property, which existed prior to release 3.0. (User string and user integer data in previous releases are 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, AREF, or node).

For example, if a boundary element used property attribute 2 with corresponding property value "metal" and property attribute 10 with corresponding property value "property" then the total amount of property data would be 18 bytes. This is 6 bytes for "metal" (odd-length strings must be padded with a null) + 8 for "property" + 2 times the 2 attributes (4) = 18.

45 BOX
[2D00]

No Data Present

Marks the beginning of a box element.

46 BOXTYPE
[2E02]

Two-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 elements of

the plex to which this element belongs. The head of the plex is flagged by setting the 7th bit. Thus, plex numbers should be small enough to occupy only the rightmost 24 bits. If this record is omitted, then the element is not a plex member.

48 BGNEXTN
[3003]

Four-Byte Signed Integer. Only occurs in CustomPlus.

Applies to Pathtype 4. Contains four bytes which specify in database units the extension of a path outline beyond the first point of the path. Value can be negative.

49 ENDEXTN
[3103]

Four-Byte Signed Integer. Only occurs in CustomPlus.

Applies to Pathtype 4. Contains four bytes which specify in database units the extension of a path outline beyond the last point of the path. Value can be negative.

50 TAPENUM
[3202]

Two-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]

Two-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. Verifies that the correct reels are being read in.

52 STRCLASS
[3401]

Two-Byte Bit Array. Only for Calma internal use with CustomPlus 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 was used for **NUMTYPES** but was not required.

54 FORMAT
[3602]

Two-Byte Signed Integer. (Optional)

Defines the format type of a Stream tape in two bytes. The two possible values are: 0 for Archive format, 1 for Filtered format.

An Archive Stream file contains elements for all the layers and data types. It is created with OUTFORM. 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 STREAMOUT. The list of layers and data types specified for STREAMOUT 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 STREAMOUT.

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 STREAMOUT. At least one **MASK** record must follow the **FORMAT** record. More than one **MASK** record 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 semicolon. 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. **ENDMASKS** is immediately followed by the **UNITS** record.

See **FORMAT** and **MASK** above.

Stream Syntax

Following is a Bachus Naur representation of the Stream syntax. Square brackets ([]) denote an entity which can occur zero or one time. Braces ({}) mean "pick one of the entities within the braces". Braces with an asterisk ({}*) mean that the entities within the braces can occur zero or more times. Braces with a plus ({}+) mean that the entities within braces must occur one or more times. Terminal are in capital letters. Non-terminals are in angle brackets (<>). The or bar (|) means "or".

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

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

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

<element>::=            {<boundary> | <path> | <SREF> | <AREF>
                        | <text> | <node> | <box>} {<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

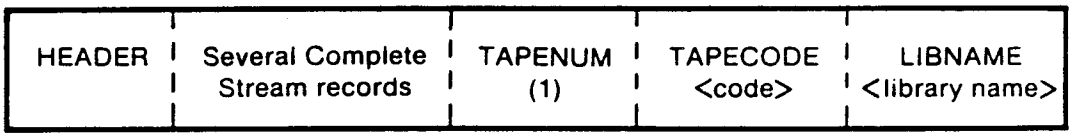
You can put Stream format onto multiple reels of tape. The first tape must end with the records **TAPENUM**, **TAPECODE**, and **LIBNAME**, in that order. Each subsequent tape must begin with the same records, in the same order, and must end with the record **TAPENUM**. Stream tapes must contain only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

> **Note:** Use **TAPENUM** and **TAPECODE** only as described. These records cannot appear anywhere else in the Stream file.

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. **LIBNAME** occurs normally as the third record of a Stream file. Tapes may end after any record in Stream format.

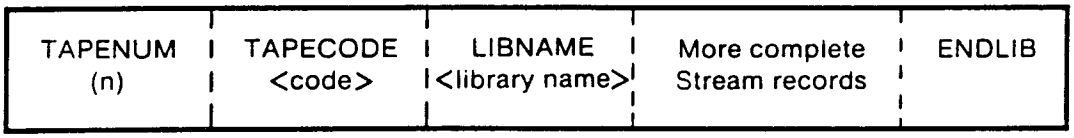
Following are illustrations of multi-reel Stream tapes.

Tape 1:



Intermediate Tape (i):

Last Tape (n):



Following is a Bachus Naur 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 **TAPENUM**s, should conform to the previously described Stream syntax.

Example of a Stream Format File

The following figure shows an FPRINT of a Stream format file. An explanation follows the example.

```
? FPRINT
Source File Name: EXAMPLE.SF
Format (Octal): HEX
Output File: $TTO
000 0006 0002 0005 001C 0102 0055 0009 0003 ..........U....
008 0000 0000 0000 0055 0009 0003 000A 0010 .......U........
010 0000 000E 0206 4558 414D 504C 452E 4442 ......EXAMPLE.DB
018 005C 1F06 4744 5349 493A 5245 4631 2E44 ...GDSII:REF1.D
020 4200 0000 0000 0000 0000 0000 0000 0000 B...............
028 0000 0000 0000 0000 0000 0000 0000 0000 ................
****
040 0000 0000 0000 0000 0000 0000 00B4 2006 ..............  .
048 4744 5349 493A 4341 4C4D 4146 4F4E 542E GDSII:CALMAFONT.
050 5458 0000 0000 0000 0000 0000 0000 0000 TX..............
058 0000 0000 0000 0000 0000 0000 4744 5349 ............GDSI
060 493A 5445 5854 2E54 5800 0000 0000 0000 I:TEXT.TX.......
068 0000 0000 0000 0000 0000 0000 0000 0000 ................
070 0000 0000 0000 0000 4744 5349 493A 464F ........GDSII:FO
078 4E54 2E54 5800 0000 0000 0000 0000 0000 NT.TX...........
080 0000 0000 0000 0000 0000 0000 0000 0000 ................
088 0000 0000 4744 5349 493A 5047 464F 4E54 ....GDSII:PGFONT
090 2E54 5800 0000 0000 0000 0000 0000 0000 .TX.............
098 0000 0000 0000 0000 0000 0000 0000 0000 ................
0A0 0012 2306 4744 5349 493A 4154 5452 532E ..#.GDSII:ATTRS.
```

Sample Stream Format File (Page 1 of 2)

```
0A8 4154 0006 2202 0003 0014 0305 3E41 8937 AT..".......>A.7
0B0 4BC6 A7EF 3944 B82F A09B 5A51 001C 0502 K...9D./..ZQ....
0B8 0055 0007 000C 0011 001D 000A 0055 0007 .U...........U..
0C0 0011 0011 003A 0014 000C 0606 4558 414D .....:......EXAM
0C8 504C 4532 0004 0B00 000C 1206 4558 414D PLE2........EXAM
0D0 504C 4531 0006 1A01 8000 000C 1C05 425A PLE1.........BZ
0D8 0000 0000 0000 0008 1302 0002 0002 001C ................
0E0 1003 0000 4E20 0000 4E20 0000 4E20 0001 ....N ..N ..N ..
0E8 4FF0 0001 3880 0000 4E20 0004 1100 0004 O...8...N ......
0F0 0700 001C 0502 0055 0007 000C 000B 001C .......U........
0F8 0009 0055 0008 001C 000F 0039 003A 000C ...U.......9.:..
100 0606 4558 414D 504C 4531 0004 0C00 0006 ..EXAMPLE1......
108 0D02 0000 0006 1602 0000 0006 1701 0005 ................
110 0006 1A01 8006 000C 1B05 4120 0000 0000 ..........A ....
118 0000 000C 1003 0000 4E20 0000 4E20 000E ........N ..N ..
120 1906 4920 414D 2048 4552 450D 0004 1100 ..I AM HERE.....
128 0004 0800 0006 2601 0001 0006 0D02 0002 ......&.........
130 0006 0E02 0003 0024 1003 0000 1388 0000 .......$........
138 6D60 0000 2EE0 0000 6D60 0000 1F40 0000 m`......m`...@..
140 84D0 0000 1388 0000 6D60 0004 1100 0004 ........m`......
148 0900 0006 0D02 0004 0006 0E02 003F 0006 ..............?..
150 2102 0001 0008 0F03 0000 03E8 0024 1003 !............$..
158 0000 3A98 0000 36B0 0000 6590 0000 36B0 ..:...6...e...6.
160 0000 84D0 0000 2328 0000 55F0 0000 1770 ......#(..U....p
168 0006 2B02 0002 000A 2C06 4D45 5441 4C00 ..+.....,.METAL.
170 0006 2B02 000A 000C 2C06 5052 4F50 4552 ..+.....,.PROPER
178 5459 0004 1100 0004 0700 0004 0400 TY............
```

Sample Stream Format File (Page 2 of 2)

The database that produced this stream format output had only two structures. They were called EXAMPLE1 and EXAMPLE2. EXAMPLE2 contained only one element, a 2 by 2 AREF of EXAMPLE1. EXAMPLE1 contained a boundary that was template data, a path with two properties, and a middle-center justified text element containing the string I AM HERE.

The records contained in this stream file are as follows:

```
0006 0002 0005
```

The first word says that this record is 6 bytes long. The second word says that this is the **HEADER** record (00 hex), and that it contains data of type two byte signed integer (02 hex). The information in the third word is the GDSII version number which in this case is version 5.

```
001C 0102 0055 0009 0003 0000 0000 0000 0055 0009 0003 000A 0010
0000
```

This record is 28 (1C hex) bytes long. It is the **BGNLIB** (01 hex) record and it contains data of type two byte signed integer (02). The 24 bytes of information following the first two header words contain the modification and last access date and time. The last 6 words of information, for example, contain: the year - 85 (55 hex), the month - September (9 hex), the day - 3 (3 hex), the hour - 10 a.m. (A hex), the minute - 16 (10 hex) and the seconds - 0. All together it says that this library was last modified on September 3, 1985 at 10:16:00 a.m.

```
000E 0206 4558 414D 504C 452E 4442
```

This record is 14 (E hex) bytes long. It is the **LIBNAME** (02 hex) record and it contains data of type ASCII string (06). The 5 words of information contain the library name **EXAMPLE.DB**.

```
005C 1F06 4744 5349 493A 5245 4631 2E44 4200 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000
```

This record is 92 (5C hex) bytes long. It is the **REFLIBS** (1F hex) record and it contains data of type ASCII string (06). All 92 bytes of this record must be present if there are any reference libraries bound to the working library. In this example, the library **GDSII:REF1.DB** is the reference library bound. The library name is padded with nulls out to 44 bytes. There is no second reference library, so the last 44 bytes are filled with nulls.

```
00B4 2006 4744 5349 493A 4341 4C4D 4146 4F4E 542E 5458 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 4744 5349
493A 5445 5854 2E54 5800 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 0000 0000 0000 0000 4744 5349 493A 464F 4E54 2E54
5800 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
0000 0000 0000 4744 5349 493A 5047 464F 4E54 2E54 5800 0000 0000
0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000 0000
```

This record is 180 (B4 hex) bytes long. It is the **FONTS** (20 hex) record and it contains data of type ASCII string (06). All 180 bytes of this record must be present if there are any

textfont definition files bound to this library. In this example, there are four (the maximum possible) text font definition files bound to this library. They are **GDSII:CALMAFONT.TX**, **GDSII:TEXT.TX**, **GDSII:FONT.TX**, and **GDSII:PGFONT.TX**. Notice that each string is padded with nulls out to 44 bytes.

0012 2306 4744 5349 493A 4154 5452 532E 4154

This record is 18 (12 hex) bytes long. It is the **ATTRTABLE** (23 hex) record and it contains data of type ASCII string (06). This record is only present if an attribute table is bound to the library. The name of the attribute table is **GDSII:ATTRS.AT**.

0006 2202 0003

This record is 6 bytes long. It is the **GENERATIONS** (22 hex) record and it contains data of type two byte signed integer (02). In this example, 3 generations of a structure are retained in the library.

0014 0305 3E41 8937 4BC6 A7EF 3944 B82F A09B 5A51

This record is 20 (14 hex) bytes long. It is the **UNITS** (03 hex) record and it contains data of type eight byte real (05). In this example, 3E41 8937 4BC6 A7EF is 1E-3. This implies that a database unit is 1 thousandth of a user unit. 3944 B82F A09B 5A51 is 1E-9. This implies that a database unit is 1E-9 meters (1E-3 microns).

001C 0502 0055 0007 000C 0011 001D 000A 0055 0007 0011 0011 003A 0014

This record is 28 (1C hex) bytes long. It is the **BGNSTR** (05 hex) record and it contains data of type two byte signed integer (02). The information in this record is the creation time and last modification time of the structure and is in the same format as in the **BGNLIB** record. This structure was created July 12, 1985 at 5:29:10 p.m. and last modified July 12, 1985 at 5:48:20 p.m.

000C 0606 4558 414D 504C 4532

This record is 12 (C hex) bytes long. It is the **STRNAME** (06 hex) record and it contains data of type ASCII string (06). Here the structure name is **EXAMPLE2**.

0004 0B00

This record is 4 bytes long. It is the **AREF** (0B hex) record and it contains no data (00). It marks the start of an AREF.

000C 1206 4558 414D 504C 4531

This record is 12 (C hex) bytes long. It is the **SNAME** (12 hex) record and it contains data of type ASCII string (06). This element contains an sname of structure **EXAMPLE1**.

0006 1A01 8000

This record is 6 bytes long. It is the **STRANS** (1A hex) record and it contains bit array data (01). In this example the only bit set is the zeroth bit which implies that this AREF is reflected. Since the 13th and 14th bits are not set, this structures magnification and angle, respectively, are not absolute.

```
000C 1C05 425A 0000 0000 0000
```

This record is 12 (C hex) bytes long. It is the **ANGLE** (1C hex) record and it contains eight byte real data (05). The data (425A 0000 0000 0000) represents 90.0 which implies that this AREF was placed at an angle of 90 degrees.

```
0008 1302 0002 0002
```

This record is 8 bytes long. It is the **COLROW** (13 hex) record and it contains two byte signed integer data (02). In this example, we have a 2 X 2 AREF.

```
001C 1003 0000 4E20 0000 4E20 0000 4E20 0001 4FF0 0001 3880 0000
4E20
```

This record is 28 (1C hex) bytes long. It is the XY (10 hex) record and it contains data of type four byte signed integer (03). The data, taken two words at a time, can be translated to decimal as: 20000, 20000, 20000, 86000, 80000, 20000. Multiply this by 1 thousandth (because a data base unit is 1 thousandth of a user unit) and we get the coordinates: (20, 20), (20, 86), and (80, 20). The first coordinate is the array reference point. The second coordinate is a point which is displaced from the array reference point in the y-direction by the number of columns times the inter column spacing. In this example, the second point was displaced 66 (86 - 20) units from the array reference point. Since there are 2 columns, this implies that the inter-column spacing was 33 units. A similar calculation could be carried out to verify that the inter-row spacing was 30 units.

```
0004 1100
```

This record is 4 bytes long. It is the **ENDEL** (11 hex) record and it contains no data (00). It marks the end of an element.

```
0004 0700
```

This record is 4 bytes long. It is the **ENDSTR** (07 hex) record and it contains no data (00). It marks the end of a structure.

```
001C 0502 0055 0007 000C 000B 001C 0009 0055 0008 001C 000F 0039
003A
```

This is another **BGNSTR** record. This structure was created July 12, 1985 at 11:28:09 a.m. and last modified August 28, 1985 at 3:57:58 p.m.

```
000C 0606 4558 414D 504C 4531
```

This is another **STRNAME** record. It contains the string **EXAMPLE1**.

0004 0C00

This record is 4 bytes long. It is the **TEXT** (0C hex) record and it contains no data (00). It marks the start of a text element.

0006 0D02 0000

This record is 6 bytes long. It is the **LAYER** (0D hex) record and it contains data of type two byte signed integer (02). This text element is on layer 0.

0006 1602 0000

This record is 6 bytes long. It is the **TEXTTYPE** (16 hex) record and it contains data of type two byte signed integer (02). This text element is of text type 0.

0006 1701 0005

This record is 6 bytes long. It is the **PRESENTATION** (17 hex) record and it contains bit array data (01). The hex number 0005 in binary has all bits set to zero except bits 13 and 15. Since bits 10 and 11 are 00, the text element used font 0. Since bits 12 and 13 are 01, the text has a middle vertical presentation. And since bits 14 and 15 are 01, the text has a center horizontal presentation.

0006 1A01 8006

This is another **STRANS** record. This text is reflected and has an absolute magnification and absolute angle.

000C 1B05 4120 0000 0000 0000

This record is 12 (C hex) bytes long. It is the **MAG** (1B hex) record and it contains data of type eigth byte real (05). The data in this record represents 2.0, therefore, this text is magnified 2 times.

000C 1003 0000 4E20 0000 4E20

This is another XY record. The text is placed at coordinate (20, 20).

000E 1906 4920 414D 2048 4552 450D

This record is 14 (E hex) bytes long. It is the **STRING** (19 hex) record and it contains data of type ASCII string (06). The text string is **I AM HERE**.

0004 1100

This is another **ENDEL** record.

0004 0800

This record is 4 bytes long. It is the **BOUNDARY** (08 hex) record and it contains no data (00). It marks the start of a boundary element.

```
0006 2601 0001
```

This record is 6 bytes long. It is the **ELFLAGS** (17 hex) record and it contains bit array data (01). Since bit 15 is set, this element is template data. However, since bit 14 is not set, it's not external data.

```
0006 0D02 0002
```

This is another **LAYER** record. The boundary is on layer 2.

```
0006 0E02 0003
```

This record is 6 bytes long. It is the **DATATYPE** (0E hex) record and it contains data of type two byte signed integer (02). This boundary is of data type 3.

```
0024 1003 0000 1388 0000 6D60 0000 2EE0 0000 6D60 0000 1F40 0000
84D0 0000 1388 0000 6D60
```

This is another XY record. The coordinates are (5, 28), (12, 28), (8, 34), (5, 28).

```
0004 1100
```

This is another **ENDEL** record.

```
0004 0900
```

This record is 4 bytes long. It is the **PATH** (09 hex) record and it contains no data (00). It marks the start of a path element.

```
0006 0D02 0004
```

This is another **LAYER** record. The path is on layer 4.

```
0006 0E02 003F
```

This is another **DATATYPE** record. The path is data type 63 (3F hex).

```
0006 2102 0001
```

This record is 6 bytes long. It is the **PATHTYPE** (21 hex) record and it contains data of type two byte signed integer (02). This path is of path type 1.

```
0008 0F03 0000 03E8
```

This record is 8 bytes long. It is the **WIDTH** (0F hex) record and it contains data of type four byte signed integer (03). 03E8 hex is 1000 in decimal. Multiply this by 1 thousandth (because a data base unit is 1 thousandth of a user unit) and we get 1. Therefore, the

width of this path is 1.

```
0024 1003 0000 3A98 0000 36B0 0000 6590 0000 36B0 0000 84D0 0000
2328 0000 55F0 0000 1770
```

This is another XY record. This path's coordinates are (15, 14), (26, 14), (34, 9), (22, 6).

```
0006 2B02 0002
```

This record is 6 bytes long. It is the **PROPATTR** (2B hex) record and it contains data of type two byte signed integer (02). This path has a property with attribute number 2.

```
000A 2C06 4D45 5441 4C00
```

This record is 10 (A hex) bytes long. It is the **PROPVALUE** (2C hex) record and it contains data of type ASCII string (06). The property value for the property attribute described in the **PROPATTR** record is **METAL**. Note that this odd length string is padded with a null.

```
0006 2B02 000A
```

This is another **PROPATTR** record. This path has another property associated with it and it has attribute number 10 (A hex).

```
000C 2C06 5052 4F50 4552 5459
```

This is another **PROPVALUE** record. Property attribute 10 (above) has the property **PROPERTY**.

```
0004 1100
```

This is another **ENDEL** record.

```
0004 0700
```

This is another **ENDSTR** record.

```
0004 0400
```

This record is 4 bytes long. This record is the **ENDLIB** (04 hex) record and it contains no data (00). **ENDLIB** marks the end of a stream format file.