DP - Format of the Drawing Files

Dario Giuse

CMU-RI-TR-85-16 **(3)**

The Robotics Institute
Carnegie-Mellon University
Pittsburgh, Pennsylvania 15213

September 1985

Table of Contents

Abstract

DP is a highly interactive graphics editor that runs on a personal workstation and can produce general purpose illustrations as well as circuit drawings. The main purpose of this document, which is a complet specification of the format and semantics of DP drawing files, is to encourage the development of applicatioi programs that can read and generate drawings in the DP format and thus exchange information in graphic form.

1. Introduction

DP is an interactive graphics editor that can produce arbitrarily complex drawings and runs on a scientifi personal computer, Perq System Corporation's PERQ workstation (see Giuse's DP - Command Set [3] for general description of the program). DP deals with graphics objects such as lines, strings of text, circles, etc as opposed to being purely bitmap-oriented. All the information entered by the user is preserved in th drawing; this makes it possible, for instance, for a program to analyze a drawing and to extract from graphical and semantical information.

As an example of information extraction, DP drawing files can be fed into a set of post-processors that extrac circuit information from drawings (see Giuse for a description of two such post-processors [9] [7] Application programs are free to assign their own semantic interpretation to any drawing; DP only deals wit the syntactic rules that determine the appearance of graphic items in a drawing.

DP drawings are stored on secondary memory as files, where each file corresponds to one drawing. Th document describes the format of drawing files as generated by DP version 6.10; please contact the author you are running a different version of the program.

The main objectives in the design of the DP file format and the technical reasons behind some of th decisions are described in section 2. Portability, completeness, and the ability to input drawing files in on pass were the primary considerations that influenced the design.

The primitive graphics elements used in DP drawing files and the symbol mechanism, which allows drawing to contain hierarchical structures that can be nested arbitrarily, are described in section 3.

The remaining sections of the document describe the coordinate system, the general structure of drawing file and the detailed syntax and semantics of each DP graphics item as represented in drawing files.

Section 12 contains the formal description of the syntax of DP files. This description uses an extended BN grammar.

Finally, presented in section 13 is an example of a simple drawing and the complete listing of th corresponding DP drawing file, illustrating some of the more important points of the format.

2. Main Objectives of the File Definition

The design of the format of DP files evolved from a few objectives that were considered esse objectives were often dictated by observations and experience with previous drawing program Draw [4], Markup [5], and SIL [6]). Most of these programs ran on the Alto, an early bit-mapp workstation developed by the XEROX Palo Alto Research Center.

The main objectives can be summarized as follows:

- **Portability**: drawing files should be portable to different machines and to different ope systems. It should be easy to read or generate drawings using different programming lang the representation of a drawing should be independent of the particular program that gener

- **Completeness**: drawing files should be completely self-contained and should not depend external library. A single file should correspond to a single drawing; the file should not ha external dependency.

- **Simplicity**: drawing files should be simple to interpret and should have a direct correspond the graphics operations they describe. It should be possible for a program to read a drawing just one pass, without the need for elaborate multiple-pass operations that may be consuming and difficult to implement.

- **Separation of concerns**: drawing files should be a purely graphical description, witho embedded semantic knowledge about what the drawing represents. The interpretatio drawing should be left to the user and to appropriate application programs; this achieves co separation between the "meaning" of a drawing and its graphical appearance, thus increasi overall flexibility of the system.

2.1 Portability

The main mechanism to achieve portability of drawing files is the choice of using exclus characters in the format. The requirement that DP files be text files was considered very impor central to the development of a large number of application programs that can read and generate was clear that this choice would cause files to be somewhat larger than binary files, and that the ti to parse a file would also be longer. On the other hand, a number of reasons that justified the c files exist; those reasons are briefly presented here.

A purely ASCII-based file format makes it easy to transfer and store files to different machin changes between successive versions are easier than in the case of binary files. ASCII files are machine independent, while binary files may embed assumptions about machine word size

tandard I/O statements available in most programming languages, without any knowledge of the interna epresentation of graphics objects.

Finally, as a side-effect, DP files may be edited using standard text editors. While this is not a recommende procedure, it has proven useful in a few cases where a simple text substitution in a very large file could fo nstance change the name of a symbol. Moreover, this procedure can be used to salvage drawings that wer lamaged due to file system errors.

The fact that DP files are pure text files was one of the main reasons that made it possible to interface DP t Mint, the Spice document preparation system, in less than two days. The task would undoubtedly have bee nuch harder if binary files had been used.

As a side remark, a comparison with SUDS [8] drawings files (which are binary files) showed that DP files ar ctually 45% smaller than SUDS files, in spite of the fact that they are entirely ASCII-based. This is becaus he representation used by SUDS is extremely bulky; a careful choice of representation would of course resu n a binary file being shorter than the corresponding text file.

2.2 One-pass Scanning

It was considered essential for the file format to support one-pass scanning: it should be possible to read drawing in just one pass through the file, and no information should be used before being defined. For a example of another format that supports single-pass reading, see the description of the CIF format in [2].

The main device to achieve one-pass scanning is the ordering of symbol definitions. Symbol definitions ar always output before any of the correspondent symbol calls; moreover, symbol definitions cannot contai other symbol definitions. Each definition appears at the top level in the file, and all the symbols that are use nside other symbols are always output first.

This convention requires more work to generate a file, but it makes reading the file much faster. Since a fil may be read several times but is only written once, it was decided to favor reading at the expense of longe writing times. In the case of DP, for instance, the measured overhead for generating files in this order is le than 15 percent of the total time to write a file; this is certainly an acceptable overhead, given the significar savings in reading time.

2.3 Files as Independent Entities

Some drawing systems (see, for instance, [8]) allow drawings that are composed of multiple files, or drawing

It was thus decided that DP drawings should be totally self-contained entities: no reference to ex allowed. This makes it extremely easy to transfer a drawing to a different machine.

Totally self-contained drawings must contain the definitions for all the symbols they use; th somewhat larger files, since commonly used definitions may have to be duplicated in several file however, that this would not constitute an important problem and that the advantages far ou possible drawbacks.

It may seem that self-contained drawings make the problem of change propagation worse, since to change a commonly used definition is to change all the drawings that use it. It should be not that external references do not address the problem of change propagation in a multi-machine anyway, and this is by far the most severe aspect of the problem. It is probably safer to let th about change propagation explicitly; for instance, a program[2] has been written that changes the one or more symbols in a set of DP files. This approach allows selective propagation of changes t files, as opposed to the common-library approach which affects all the drawings that use a library

3. Graphic Elements in DP Drawings

Only a limited set of basic elements can appear in drawing files: such elements are used to create mor complex drawings. All the basic elements represent a geometric concept (a line, a circle, and so on); eac drawing is thought of as a collection of geometric elements.

DP does not use bitmaps as representations for objects: it is impossible, for instance, to describe a curve b tracing it on the screen with the cursor. Smooth curves are defined geometrically, in terms of control point Extensive use of geometric representations makes DP drawings independent of the particular characteristic of the device the drawing was created on.

3.1 Primitive Graphic Elements

The primitive elements used by DP are

- **lines**: finite-length straight line segments,
- **circles** and **arcs**: full circles or arcs of circles,
- **ellipses**: full ellipses or arcs of ellipses,
- **splines**: 3rd order B-splines, i.e., parametric curves that interpolate a set of points,
- **polygons**: filled polygons identified by a set of vertices and a filling pattern,
- **text strings**: sequences of printing ASCII characters in a given font, and
- **pins**: used inside symbol definitions to provide connection points. Each pin has a pin number associated with it.

3.2 Symbols

Primitive elements may be composed through the Symbol mechanism. When a set of objects is made into symbol, the latter becomes equivalent to a primitive element. All the operations that apply to primitiv elements apply to symbols as well; in particular, symbols may be nested inside other symbols.

The following two mechanisms are provided:

- **definition** of a symbol: definition of a group of elements that determine the shape of the symbol. A definition in itself does not add any element to a drawing: it simply defines how to draw a group of elements if required. A definition can be considered as a template that describes how to draw a picture.
- **instance** of a symbol: creation of a copy of a symbol in a drawing; each instance defines the global offset and the transformation parameters. Creating an instance is equivalent to adding to the drawing the whole set of elements that form the symbol; if transformations are used, the elements may appear rotated or scaled.

4. Description of the Format

Please note that in the rest of the document the phrases "the Reader" and "the DP Reader" wil indicate the portion of code in DP that reads drawing files and converts them into graphic items.

4.1 Coordinate System

DP files use a Cartesian coordinate system: abscissas increase to the right, and ordinat.es increas- All coordinates are expressed as pairs of integers in base 10, in the range -32767 through 32767; th the point (0, 0). For consistency, numbers other than coordinates arc also expressed as decimal inl

Angles are always measured in minutes of arc, starting from 0 for the positive X direction z counter-clockwise. For instance, an angle of 90 degrees (corresponding to the north) is expressed 5400 minutes; south is expressed as 270*60 = 16200 minutes. The valid range for an angle is [0.. the first and last values representing the same angle. Angles outside the range are always no adding or subtracting 21600.

4.2 Bounding Boxes

The bounding box of an object is defined as the smallest rectangle that completely encloses the edges of the rectangle are parallel to the cartesian axes.

The bounding box of an instance is always computed after the necessary transformations have b< for this reason, the dimensions of the box may not coincide with the size of the symbol definition.

4.3 Genera! Format

In general, a text line in a DP file describes one graphic element; some lines are used differently, as comments or as non-graphical information. Lines are terminated by the normal end-of-line and different fields of a line are separated by one blank.

The first character in a line determines what kind of object the line represents. The first charac may be one of the following:

- **Capital letter:** the line describes a basic element or a symbol definition.
- Semicolon: the line is a comment.
- @: the line contains special non-graphical information.

```
; DP ver. 6.9
```

The DP Reader requires this line at the beginning, and will abort the Read operation if the first character not match exactly the string "; DP ver.". This line has the double function of rejecting non-DP files informing DP of what version of the program created the file;[4] all the characters after the first ten considered part of the version number.

Except for this special convention, comment lines may appear anywhere in the file.

4.4 Structure of a DP File

The following order is used in all DP files. Since the DP Reader depends on this particular order in a places, files that do not follow this order are considered illegal.

- **Version number**: the special comment line mentioned above.
- **Time stamp**: an optional comment line with the date and time of creation of the file.
- **Tree structure**: an optional set of comments that describe the tree of symbol definitions and specify dependencies on other symbols. This information is only used by the "i" (Read Symbol From File) command.
- **Font information**: definition of all the fonts used in the drawing. This section is absent if no strings are used.
- **Layer information**: definition of all the layers that constitute the drawing.
- **Marks information**: list of all the special position markers in the drawing. This section is absent if no mark is used.
- **Grids information**: list of mouse and display grids used when creating the drawing.
- **Symbol definitions**: list of all the symbol definitions used within other symbols and at the top level. The definitions are ordered so that lower-level definitions precede higher-level definitions: a symbol is always defined before it is used. This section is absent if the drawing is "flat", i.e., it contains no symbols.
- **Top-level items**: all the objects (primitive graphics elements and symbol instances) that appear at the top level in the drawing, i.e., are not nested inside symbols.

5. Tree Structure

The purpose of this section is to describe dependencies among symbols. This information is used
single symbol is read from a file by the Reader. Application programs are free to ignore
altogether; all the lines in the section are preceded by a semicolon and as such they may be c
comments.

The special command that reads one symbol from a DP file uses this information to avoid mu
over the file. When a symbol is read, all the nested symbols must be read as well; this procedure i
recursive, since it is impossible to know what symbols are needed before reaching the top-leve
The Tree Structure provides the Reader with this information: in particular, each symbol is lis
transitive closure of all the symbols it uses. This information is precomputed when the file is firs
that each symbol contains the names of all the symbols it uses either directly or indirectly.

During the "Input One Symbol" command, the DP Reader simply scans the file and enters the s
were listed in the Tree Structure as used by the required symbol. Everything else is discarded;
stops after the required symbol has been read.

5.1 Format

For each symbol definition two comment lines are used in the Tree Structure section:

- ;SYMBOL: *SymbolName*
- ;CALLS: *list of all the nested symbols, separated by blanks*

All the symbols have an entry in the Tree Structure entry. Leaves of the tree, i.e., symbols that
composed of primitive elements, have an empty ';CALLS: ' field.

5.2 Example

Imagine a drawing that contains symbol TOP, which imports the two other symbols SYM
SYMBOL2; imagine also that both SYMBOL1 and SYMBOL2 import symbol LEAF. The Tr
section of the drawing would then look like this:

```
;SYMBOL: TOP
;CALLS: SYMBOL1 SYMBOL2 LEAF
;SYMBOL: SYMBOL1
;CALLS: LEAF
;SYMBOL: LEAF
;CALLS:
;SYMBOL: SYMBOL2
;CALLS: LEAF
```

3. Font Information

This section of the file describes the fonts used by the strings in the drawing. The main purpose of the sectio is to define local font numbers, i.e., numbers that uniquely identify a particular font. This is the only place i file where font name, size, etc., are explicitly mentioned; all further references are through the unique loca ont number.

Local font numbers are only meaningful within one file: the same font entry may well have a different loca ont number in a different file. Whenever the file is read, the DP Reader automatically converts the local fon umber into a global font specification.

A font entry specifies two distinct pieces of information: the abstract font specification (e.g. TimesRoman 1 oldface) and the name of a Perq font used to display that font on the Perq screen. Giuse [3] and Sproull [1 ive more details on font specifications. Two lines are used for each font entry: the first line contains th bstract font specification, the second line identifies the Perq font file.

The format of the first line (abstract font definition) is the following:

@font *FontNumber Face Size Rotation Family*

FontNumber: a small positive integer that uniquely identifies this font. This number will be used in a the strings that use this font, and is meaningless outside the file. Local font numbers ar unique, but they are not guaranteed to be contiguous or monotonically increasing.

Face: one or two lower-case characters that identify the font face (see Giuse [3] for more details The characters **{r b i}** are currently used to indicate Roman, Boldface, and Italic Characters may be combined when this is meaningful (for instance, **bi** stands for boldfac italics).

Size: a positive integer that specifies the size of the font (in points). Big numbers indicate larg fonts; see Sproull [1] for more details.

Rotation: the font rotation, in minutes of an arc (see section 4.1); normally 0.

Family: the name of the font family (e.g., "Bodoni" or "NewHelvetica"). Case distinctions in th family name are ignored, i.e., "Gacha" and "GACHA" are equivalent.

The format of the second line (Perq font) is the following:

@perqfont *FontNumber FileName*

FontNumber: must match one of the local font numbers that appear in a @font statement. Note that i files generated by DP this line follows immediately the corresponding @font line, and th FontNumber is thus redundant.

FileName: the file name of a valid Perq font. The font must be present on the Perq disk when this lin

7. Layers Information

This section defines the layers used in the drawing and specifies the value of the parameters for ea
Giuse [3] for more details on layers). Note that if a layer name is already in use before the file is r
new layer are merged and old and new items appear on one layer. In other words, layer names
all DP files; it is impossible to define two different layers with the same name.

In the case of symbol instances, layers act as filters. Imagine for example that a symbol on laye
items that are on layer B: if both layers A and B are visible, the items will be displayed. If laye
invisible, though, the items will not be displayed even if their own layer (B) is still visible. Since t
the upper level is invisible, none of the objects inside it are visible: layer A is "filtering" a
contained in symbols that appear on it.

Each line in the Layers Information section defines one layer. The format of each line is the follo

`@layer` *LayerNumber Name Options*

LayerNumber: a small positive integer that uniquely identifies this layer. This number will b
the following items, and is meaningless outside the file. Local layer number
but are not guaranteed to be contiguous or monotonically increasing.

Name: the ASCII name of the layer, converted to all upper case. No blanks are all
name.

Options: a sequence of characters encoding the options setting for the layer; all characte
case. If an option is ON, the corresponding character is present. The cu
characters is: R (the layer is Readable), W (the layer is Writable), and O (the
output when the drawing is written to a file). For example, RO means that the
displayed and output but cannot be altered.

8. Marks Information

Marks provide a convenient mechanism to position a drawing around meaningful reference points. Ea
mark has a number associated with it; this number is currently unused (see Giuse [3] for more details
marks).

The format of a mark entry is the following:

@pageMark *x y number*

x, y:	position of the mark, in absolute coordinates.
number:	a unique integer associated with the mark (currently unused).

9. Grids Information

This section contains information about the mouse and display grid settings that were in us
drawing was created. This information is not needed for the correct interpretation of the drawing
provided as a convenience to the user. It is often the case that a drawing is edited using a non-st
causing potential alignment problems if a different grid is used during subsequent editing.

When the Reader encounters the Grids Information section, it sets the current mouse and disp
match the ones specified in the drawing.

The format of the grid entry is the following:

@grids *mouse-grid display-grid*

mouse-grid: value of the mouse grid, as a positive integer. This indicates the distance betw
nearest points at which the mouse can be located.

display-grid: value of the display grid, as a positive integer. This indicates the distance bet
of the grid that DP uses to facilitate item alignment.

Note that only one grid entry is present in a DP file, even though many mouse grids can be used
editing session. Only the current mouse grid is saved in the file.

10. Symbol Definitions

lymbol definitions arc collections of items enclosed by a begin-end pair and identified by a unique name iymbol definitions describe how to draw symbol instances and can be thought of as templates; simila dcturcs can be generated from the same template (symbol definition) by scaling, rotating, or mirroring th< tasic definition.

?oordinatcs within definitions arc relative, i.e., they need a translation before they can be displayed; th< ranslation is specified at the time the symbol is instantiated. Coordinates within a symbol definition ar torcd so that the *center of the symbol* is the point (0, 0); the "center of the symbol" is defined as the center o he bounding box of the symbol. In other words, all the items appearing in symbol definitions arc centers iround die point (0,0).

10.1 Beginning of Symbol Definition

Phis line starts the definition of a new symbol. The format of the line is the following:

D *width height name*

vidth, height: width and height of the bounding box of the symbol in its non-transformed definition, i.e when the rotation is 0 and the scaling is 1.0,1.0.

lame: the unique name of the symbol, in upper-case characters. This name is the only way thi symbol can be referred to in the future.

\fter this line the primitive elements that constitute the body of the symbol definition are listed, each one i ts normal format (see section 11). Every basic element can appear here, including instances of other symbol: symbols may be nested at any level. The end of the symbol is marked by the End of Symbol Definition.

10.2 End of Symbol Definition

Fhe format of this line is the following:

F

Fhis line marks the end of a symbol definition. The symbol being defined is closed and entered in the list < definitions. This line will typically be followed by either the beginning of a new definition, or by the fir top-level item in the drawing.

11. Top-level Items

By default, all items that are not part of a symbol definition are considered to be at the top drawing. All top-level items use absolute coordinates, unlike items that are part of a symbol defini

Several fields are common to various item types; their meaning is described only once and repeated for individual items. The common fields are:

Thickness: a positive integer that indicates the thickness of lines, circles, ellipses, and convention, the thinnest item has thickness 1; in the current implementation, item has thickness 7. Numbers greater than 7 are currently interpreted as 7.

Color: a positive integer that specifies the color of an item. By convention, color 1 black item. Colors other than 1 are preserved and used for devices that suppor the current implementation all items except Polygons are displayed in black regardless of the color specified in the file. *The meaning of the color informatio change in future versions.*

Layer: a local layer number, i.e., a reference to an **@layer** statement. This indicates wh item belongs to. In the case of a symbol instance, this is the layer of the **instan** act as a filter for nested items.

Line-Style: a small integer that determines the line style to be used for lines, circles, splines. The default line style is *solid* and is indicated by line-style 0. Line-styl *dotted*, which is a pattern of equally spaced short dashes. Line style 2 indic which is a pattern of equally spaced long dashes. Line style 3 indicates *dot-and-* is a repeating pattern of short and long dashes. Line styles greater than 3 a undefined.

The following sections contain the description of all the different types of items and the fo corresponding entries in a DP file. Remember that all numbers are integers, unless otherwise spec

11.1 Straight Lines

L *x1 y1 x2 y2 thickness color layer line-style*

x1,y1: coordinates of the first endpoint of the line.
x2,y2: coordinates of the second endpoint of the line.

11.2 ASCII Strings

S *x1 y1 x2 y2 font color layer string*

he size of a string is computed under the assumption that the specified font is used to display the string; sing a different font would result in a different size. The reference point is, at any rate, the **lower-left** corner f the string: the upper-right point can always be computed from the lower-left corner. If a different font ust be used for a different device, the string should be positioned in such a way that its lower-left corner nds up at the point (*x1*, *y1*).

1.3 Arcs and Circles

A *x y radius angle1 angle2 thickness color layer line-style*

,y:	coordinates of the center of the circle.
adius:	radius of the circle.
ngle1:	first angle of the arc, i.e., first angle encountered on the arc when scanning i counterclockwise; the starting point is on the positive X axis, at (*radius*, 0) from the center The angle follows the usual convention explained in section 4.1.
ngle2:	second angle of the arc. If angle2 is equal to angle1 the arc is a full circle.

Note that a full circle is normally indicated by angle1 = 0, and angle2 = 21600.

11.4 Ellipses

E *x y radius1 radius2 angle1 angle2 thickness color layer line-style*

,y:	coordinates of the center of the ellipse.
adius1:	horizontal radius of the ellipse, i.e., half the horizontal diameter.
adius2:	vertical radius of the ellipse.
ngle1:	first angle of the arc of ellipse, i.e., first angle encountered on the arc when scanning counterclockwise; the starting point is on the positive X axis, at (*radius1*, 0) from the cente The angle follows the usual convention explained in section 4.1.
ngle2:	second angle of the arc; same conventions as before. If angle2 is equal to angle1 the arc is full ellipse.

A full ellipse is normally indicated by angle1 = 0, and angle2 = 21600. Note that this representation do not allow an ellipse whose major and minor axes are not parallel to the cartesian axes, but such an ellipse ca be represented by nesting it into a symbol and then rotating the symbol.

11.5 Splines

B *x y n k thickness color layer line-style x1 y1 x2 y2 ... xn yn*

...	global offset of the spline.

11.6 Polygons

Ry convention, an n-sided polygon is represented by n vertices; the first and last vertices arc considerc ;onncctcd.

Y *x y thickness color layer xl yl x2 y2... xn yn*

c,y:	global offset of the polygon.
:olor:	this parameter is used to select a pattern that fills the polygon. Polygon patterns ar predefined and arc intended to emulate different shades of gray. Only colors in the range through 17 are supported; color 1 corresponds to solid black, color 17 is solid whit< Numbers in between represent intermediate shades of gray, color 8 being an intermedia! "solid" gray.
d,yl:	coordinates of the first vertex of the polygon, relative to (x,y).
t2,y2:	coordinates of the second vertex, relative to (x,y).
..	
cn,yn:	coordinates of the last vertex of the polygon, relative to (x,y).

Note that the *thickness* parameter is ignored.

11.7 Pins

P *xl yl number position color layer*

d,yl:	abscissa and ordinatc of the pin.
lumber:	an integer corresponding to the pin number.
)osition:	an integer in the range 0 through 3 that specifies the quadrant the pin number should b displayed in.[6] Position 0 corresponds to the first (upper-right) quadrant, position corresponds to the upper-left quadrant, and so on.

^in numbers are ignored by many programs that process DP drawings. A pin contained in a symbol acts as gravity point in DP, independent of its pin number.

11.8 Instance of a Symbol

Phis line creates an instance of a symbol. Instantiating a symbol is equivalent to calling a procedure that *hz*)een defined and stored away. The instance specifies the global offset and transformations for the symbo he transformations are applied first, and the offset is applied later. If nested symbols are contained in th iefinition, each offset/transformation is applied in order, from the innermost to the outermost levels.

transformations are additive: if a symbol has rotation Rl and calls another symbol with rotation R2, the fin< •esuit is the same as if the nested symbol were directly called with rotation CR14-R2V The order c

he format of a symbol instance is the following:

C *x y angle scale-x scale-y layer name*

y:	abscissa and ordinate of the center of the symbol, i.e., global offset.
ıgle:	rotation of the instance (see 4.1 for angles conventions).
:ale-x,scale-y:	(these are two real numbers): scaling factors for the instance in the X and Y directions. A scaling factor of 1.0 means the same scale as the definition; a negative factor implies a mirroring operation. Note that because of the order of application of symbol transformations, these factors indicate the scaling of the symbol *before* the rotation is applied, i.e., they indicate the scaling factors relative to the symbol's own X and Y axes.
ame:	reference to a symbol definition. This string must match exactly the name in a symbol definition, and thus it must be in all upper-case characters. The symbol must be already defined; forward references are not allowed.

12. BNF Description of the Format

This is the description of the format of DP drawing files in an extended BNF syntax. The fol additional notations have been used:

{ <item> }$^{+}$ <item> may appear from 1 to infinite times.
{ <item> }* <item> may appear any number of times, including 0.

<char>, not defined here, is essentially the whole ASCII printing set minus EOL (End-Of-Line).

<DP file> ::= "; DP ver." <string> {<line>}*

<comment> ::= ";" {<char>}* EOL

<string> ::= {<char>}* EOL

<number> ::= " " <integer> | " -" <integer>

<integer> ::= {<digit>}$^{+}$

<real> ::= <number> "." <integer> | <number>

<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9

<point> ::= <number> <number>

<parameters> ::= <number> <number> <number> <number>

<line> ::= <symbol definition> | <ENV line> | <DP line>

<ENV line> ::= "@font" <number> {<char>}$^{+}$ <number> <number> <string> |
"@perqFont" <number> <string> | "@layer" <number> {<char>}$^{+}$ <string> |
"@pageMark" <number> <number> <number> EOL |
"@grids" <number> <number> EOL

<DP line> ::= "L" <point> <point> <parameters> EOL |
"A" <point> <number> <number> <number> <parameters> EOL |
"E" <point> <number> <number> <number> <number> <parameters> EOL |
"B" <point> <number> <number> <parameters> {<point>}$^{+}$ EOL|
"Y" <point> <number> <number> <number> {<point>}$^{+}$ EOL|
"S" <point> <point> <number> <number> <number> <string> |

13. Example of a Drawing File

The present section contains an example of an actual drawing file. The drawing in the example is shown fig. 13-1.

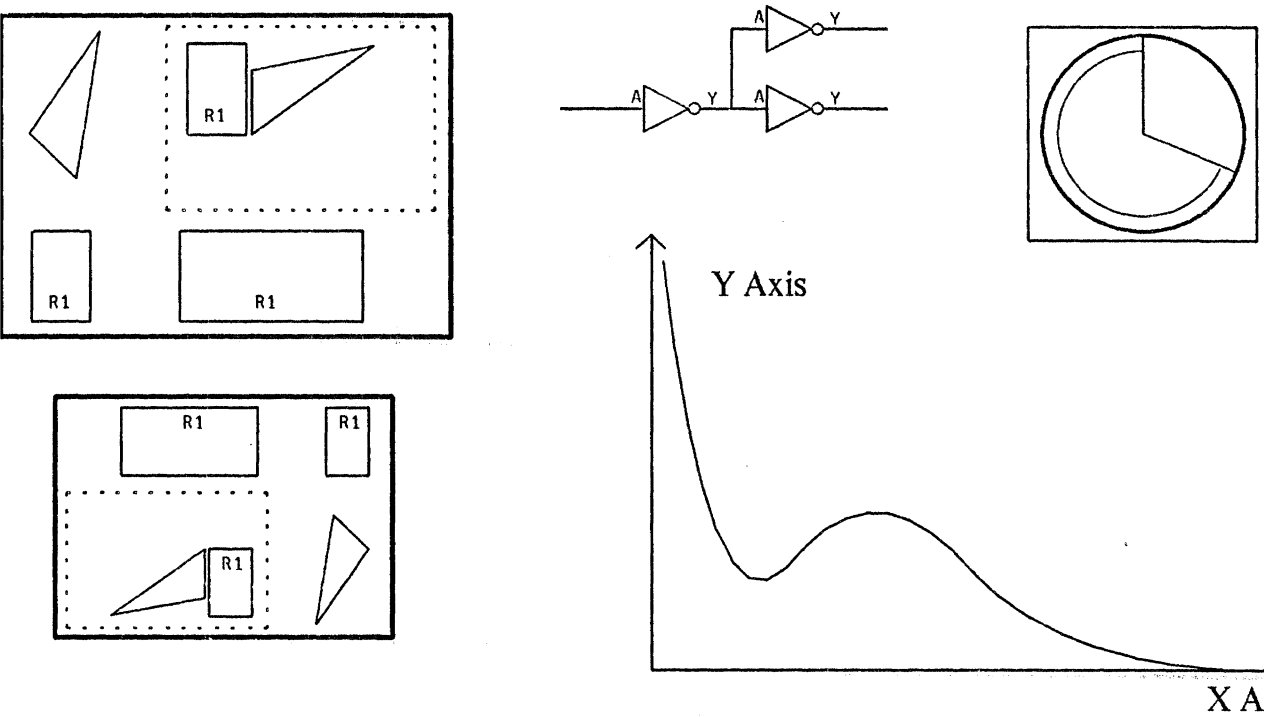

Figure 13-1: The drawing in our example

The complete text of the DP file is also shown. Several points are worth discussing:

- Most of the drawing is on the STANDARD layer, except for the axes system in the graph at the right which is on the FRAME layer instead. This layer includes the two axes and the two strings "Y Axis" and "X Axis"; the layer is currently non-writable, as indicated by the absence of the W parameter from the last field of the line "@layer 2 FRAME RO".

- Two instances of the symbol named PICTURE are used; the two instances appear in the left half of the drawing. The instance at the bottom is scaled down (the scaling parameters are equal to 0.75 and 0.75) and rotated 180 degrees counterclockwise (as indicated by a rotation angle of 10800 minutes)

• The graph at the bottom-right is drawn as a single 3rd-ordcr B-splinc, described by the

B 2 -168 6 3 1 1 1 0 311 1 204 0 108 112 24 6 0 218

The spline is defined by the 5 control points (311,1) (204,0) (108,112) (24,6) (0,218) ai in thickness 1.

```
;DPvcr.6.10
; 09-M-85 12:54:17
SYMBOL: 7404
;CALLS:
SYMBOL: PICTURE
-.CALLS: COMPOSITE RECT´TRIANGLE
SYMBOL: COMPOSITE
;CALLS: RECTTRIANGLE
SYMBOL: TRIANGLE
;CALLS:
SYMBOL: RECT
;CALLS:
©font 1 r 7 0 Gacha
©pcrqFont 1 gacha7.kst
©font 4 r 12 0 TimesRoman
©perqFont 4 gacha7.kst
©layer 1 STANDARD RWO
©layer 2 FRAME RO
©pageMark 41 -1211
©pageMark -18 7 3
©pageMark 41 -121 2
©grids 16
D 30 48 RECT
S -7 -18 5 -9 1 1 1 Rl
L 15 -24 15 24 1 1 1 0
L15 -24 -15 -24 1 1 1 0
L -15 -24 -15 24 1 1 1 0
L 15 24 -15 24 1 1 1 0
F
D 36 78 TRIANGLE
L18 39 6 -39 1 1 1 0
L6-39-18-15 1 1 1 0
L -18 -15 18 39 1 1 1 0
F
D 96 80 COMPOSITE
C 8 0 18900 1 1 1 TRIANGLE
C -33 -8 0 1 1 1 RECT
```

L -115 -89 -115 81 3 1 1 0
L 116 -89 -115 -89 3 1 1 0
L 116 -89 116 81 3 1 1 0
C -84 -57 0 1 1 1 RECT
C -82 34 0 1 1 1 TRIANGLE
C 29 50 0 1 1 1 COMPOSITE
F
D 45 24 7404
S -22 2 -16 11 1 1 1 A
P -15 0 1 1 1 1
S 17 1 23 10 1 1 1 Y
A 11 0 3 9900 9150 1 1 1 0
L -15 -12 8 0 1 1 1 0
L -15 12 -15 -12 1 1 1 0
L -15 12 8 0 1 1 1 0
P 14 0 2 4 1 1
F
C -223 99 0 1 1 1 PICTURE
A 248 118 44 5400 20056 1 1 1 0
L 189 174 189 61 1 1 1 0
L 295 98 248 118 1 1 1 0
L 248 170 248 118 1 1 1 0
L 306 174 306 61 1 1 1 0
L 306 174 189 174 1 1 1 0
L 306 61 189 61 1 1 1 0
A 248 118 52 0 21600 3 1 1 0
C 7 131 0 1 1 1 7404
L 55 131 21 131 1 1 1 0
C 70 131 0 1 1 1 7404
L 84 131 117 131 1 1 1 0
L -9 131 -51 131 1 1 1 0
L 37 173 37 131 1 1 1 0
C 70 173 0 1 1 1 7404
L 55 173 37 173 1 1 1 0
L 84 173 117 173 1 1 1 0
B 2 -168 5 3 1 1 1 0 311 1 204 0 108 112 24 6 0 218
C -223 -89 10800 0.75 0.75 1 PICTURE
L -4 64 -4 -167 1 1 2 0
L 338 -167 -4 -167 1 1 2 0
L 3 57 -4 64 1 1 2 0
L -12 57 -4 64 1 1 2 0

14. Compatibility with Previous Versions

The present document describes the format of DP files for internal purposes only; no guarantee is to the stability of the format itself. Although the format has remained substantially unchang periods of time, changes and extensions to DP have required various adaptations and additions to t

In particular, it should be noted that changes and extensions may make new formats unreadab versions of DP. It is typically impossible to read files whose version number is higher (newe version of DP one is using. The DP Reader, on the other hand, is written in such a way as to be with older versions; all versions of DP can read files whose version number is less than or e particular version being used.

This is believed to apply to all existing versions of the program, including local modifications that using. If this is not the case, please contact the author reporting the version number of the instance are using, the version number and creation date of the drawing file, and information about how y your version of the program.

References

[1] Robert F. Sproull.
Font Representations and Formats.
tèchnical Report, XKROX Palo Alto Research Center, October, 1980.

[2] R.W. Hon, CM. Sequin.
/(*Guide to LSI Implementation,*
Technical Report, XKROX Palo Alto Research Center, January, 1980.

[3] Dario Giusc.
DP - Command Set.
Technical Report CMU-RI-TR-82-11, Carnegie-Mellon University, October, 1982.

[4] Patrick C Baudelaire.
Draw Manual.
Technical Report, XEROX Palo Alto Research Center, 1978.

[5] William M. Newman.
*Markup User*s Manual.*
Technical Report, XEROX Corporation, Palo Alto, 1978.

[6] C.P. Thackcr, R.F.Sproull, R.D.Bates.
SIL, ANALYZE, GOBBLE, BUILD - Reference Manual
Technical Report, XEROX Corporation, Palo Alto, Ca., 1979.

[7] Dario Giuse.
SL: a hierarchical wire-listerfor DP drawings.
Technical Report, Carnegie-Mellon University, March, 1982.

[8] Joseph M. Newcomer.
SUDS - User's Manual
Carnegie Mellon University, 1980.

[9] Dario Giuse.
DP: post-processing a circuit drawing
Carnegie-Mellon University, 1981.