Press File Format

by Bob Sproull, William Newman, and Joe Maleson

DECEMBER 1979

This report describes the format of Press files, a standard file format used to encode documents for printing and editing. This report and associated references cover all the information needed to generate Press files.

Filed on <PrintingDocs>PressFormat.Press.

XEROX
PALO ALTO RESEARCH CENTER
3333 Coyote Hill Road / Palo Alto / California 94304

This memorandum describes the Press file format, an attempt at defining a "universal" format for describing documents. The format permits easy printing on a number of different devices and cooperative editing of documents by various editing and illustrating programs. For an introduction to Press, the reader is invited to consult [1].

This description is in four parts. Part 1 is an explanation of the basic ideas embodied in the Press file format. Part 2 describes the format in detail. Part 3 describes the management of character width information. Part 4 describes current conventions, entity types and the like.

Part 1: Basic organization

The Press file format has been designed to meet the following requirements:

- (a) it should permit documents to contain text, computer graphics and scanned images, arranged freely on the page;
- (b) it should permit straightforward text formatting and page layout by editing programs;
- (c) single-pass generation of files should be possible (e.g. file directory information should be at the end of the file) so that files can be transmitted directly to the printing facility as they are being created;
- (d) the format should permit printing and display on various devices such as the Alto display, a Slot/7000, a Slot/3100, or a Diablo printer;
- (e) it should permit the inclusion of high-level formatting information in addition to the information required specifically for printing and display.

The last of these requirements is an important one, and has had a profound influence on the design. Our intention is to make it possible for Press files to be the *only* representation of documents generated by editors and illustration programs. Files in this format will then be suitable not only for printing, but also for use as input files for later editing or re-formatting. This approach solves the problem of converting files generated by one editor for use by another.

The general organization of Press files is therefore as follows:

- 1. The document is divided into *parts*. The most common type of part is a *printed page*; special parts define such things as fonts used in the document, cross-reference tables, etc.
- 2. The information for each printed page is divided into two lists: a *data list* followed by an *entity list*. The data list may be thought of as the basic text and graphics to appear on the page, while the entity list contains page layout information together with pointers into the data list.
- 3. Following the information for the last page is a *part directory* giving the file address and length of each part. The file is terminated by a *document directory*.

The entity list consists of a number of *entities*; the layout of each entity is described in terms of its x-y coordinates, using 10-micron (1/2540 inch) units named *micas*. Each entity has an absolute position on the page, and within the entity all coordinates are relative to this position. Thus the entity may be readily repositioned on the page without the need for examination of the entity's contents by the page layout program. Entities may include instructions to skip over parts of the data list; this permits inclusion in the data list of high-level formatting instructions for use by text editors.

The page information may include graphical items, or *objects*. These are geometrical shapes, curved and rectilinear, defined by their outlines and by the shade of gray to be used to fill them in. Objects are opaque, and may overlay each other on the page. The order of overlaying is determined by the order in which the objects occur in the entity list: objects that occur late overlay objects that occur early.

The overall effect is to enable printing or display devices to generate page images by examining the entity lists and following the data pointers within each entity. Editing and formatting programs, on the other hand, may rely largely on the high-level information contained in the data lists: for these programs, the entity lists will serve mainly to distinguish the different classes of data (e.g. pure text, illustrations, etc.) so that specialized editors can process some of the data and ignore the rest.

Part 2: Description of the Press file format

In the description that follows, names that are in bold face refer to abstractions that are defined by this memo. Thus part refers to a formal object that is defined below.

A byte is 8 bits; data bytes in the file are numbered starting at 0. A word is two bytes aligned so that the first byte of the word has an even byte address. A record is 512 bytes or 256 words. Records are numbered from the beginning of the file (the first record is numbered 0). A Bcpl string is formatted as follows: the first byte contains the number of characters in the string (e.g. 3 for the string "the"), and the following bytes contain the ASCII codes for the characters in the string.

Numbers are decimal unless followed by a "b," in which case they are octal. Brackets (<...>) are used to delimit commands that appear in the entity list (EL). Double brackets (<<...>) delimit formats of things that appear in the data list (DL). Commands may have arguments: [value n] means that value is specified in n bytes.

The coordinate system for a **printed page** is as follows: the origin is at the lower left-hand corner of the page (assuming a portrait-oriented page, i.e., with the shorter edges horizontal). The x direction extends to the right, the y direction up. The unit of measurement is a mica (equivalent to 10 microns = 1/2540 inches); this means that each of the coordinates of any point on the page can be represented by a 16-bit number.

Overall file organization

A Press file will typically consist of its parts, its part directory and its document directory, in that order, with the parts arranged in order of ascending page number and terminated by a font directory part. It is not obligatory to follow this order; the only mandatory rules for arranging the information within the file are:

- 1. The document-directory must be at the end of the file:
- Parts, part directory and document directory must each occupy an integral number of records.

2.1 The printed page

A printed page is a specific type of part, and consists of two lists: first comes the data list, followed by the entity list. Below, these are abbreviated as DL and EL respectively. The order of information in the data list as referenced by pointers in the entity list must be sequential to avoid the need to back up during printing or display.

The entity list is constructed as follows: a word that contains zero comes first, followed by the entities. Following the last entity, there may be some entity list padding required to fill out the last

record of the part. Each entity has two pieces: a set of entity commands, described below, followed by an entity trailer. The format of the entity trailer is:

[type 1] [font-set 1] [begin-byte 4] [byte-length 4] [Xe 2] [Ye 2] [left 2] [bottom 2] [width 2] [height 2] [entity-length 2]

An entity describes a region of DL, from begin-byte to (begin-byte)+(byte-length)-1. The type indicates what kind of editing or illustrating program has generated the entity. The font-set byte tells which set of up to 16 fonts should be used to print the entity. The values Xe and Ye define the origin of the coordinate system used for all coordinates in this entity. The four values left, bottom, width and height define the bottom left corner position (relative to this origin) and dimensions of the bounding box that surrounds the image of the entity on the page. The entity-length specifies the number of words of EL occupied by this entity (including the trailer itself).

The entity trailer is preceded, in EL, by entity commands that indicate how to interpret the data to which the entity refers.

Example: In a text editor, an entity could correspond to a paragraph. The begin-byte pointer and byte-length would describe where in **DL** the text for the paragraph lies. Formatting information recorded by the editor for the paragraph may also lie in **DL**.

The *type* of the entity does not affect how a printing facility prints the page, but is included to permit various editing and formatting programs to distinguish entities that they created or can reformat. Entity types will be assigned to software projects on application to the authors of this memo. Current entity type definitions are listed in section 4.

Several rules should be kept in mind when building entities and corresponding commands:

- 1. The commands in EL are processed sequentially, from beginning to end. Thus, characters and objects overwrite each other in that order. A gray character appearing near the end of the EL information may overwrite a black one near the beginning.
- 2. The entity and entity trailer must begin on a word boundary. This convention permits EL to be rearranged easily by copying a word at a time. <Nop> may be used to fill out an odd byte before the entity trailer. Note that a word containing zero must precede the first entity in EL; printing programs will normally determine the positions of entities in the file by scanning back from the end of EL, and this word may then be used to terminate the search.
- 3. The **Set-brightness**, **Set-saturation**, and **Set-hue** settings apply to all characters displayed with **Show-characters** and **Show-character-immediate**, to objects displayed with **Show-object**, and to rectangles displayed with **Show-rectangle**. The settings do *not* apply to information described by **Show-dots**.
- 4. Each entity commences by establishing the following defaults: <Set-brightness> 0; <Set-saturation> 377b; <Set-hue> black; 0; <Reset-space>; <Set-x> 0; <Set-y> 0; <Only-on-copy> 0.

The commands in an entity are comprised of:

 $\langle \text{Set-x} \rangle [x \ 2]$

The current x position is set to x+Xe (Xe and x are signed numbers).

 $\langle \text{Set-y} \rangle [y \ 2]$

The current y position is set to y+Ye (Ye and y are signed numbers).

\langle Show-characters \rangle [n 1]

The next n characters in DL are displayed at the current position (x,y). As each character is displayed, the current position is updated by adding the x- and y-components of the width of the character to the current position (see section 3 for Width information). When a character is displayed, its "(0,0) point" is made coincident with the current (x,y) position. Fonts are usually designed so that (0,0) points of characters lie on the baseline at the left edge of the character.

⟨Show-character-immediate⟩ [char 1]

This command is similar to (Show-characters), but it displays the single character specified as the parameter. This command is intended to be used to add visible characters to the page when doing formatting. An example might be a hyphen at the end of a line, which the programmer may find inconvenient to include in the DL.

$\langle Skip\text{-characters} \mid [n \mid 1]$

The next n bytes in DL are ignored. This command may be used to skip over carriage-returns, tabs and other characters that may have no meaning to the printing device.

$\langle Skip\text{-control-bytes} \rangle [n \ 2] [type \ 1]$

This has the same effect as $\langle Skip\text{-characters} \rangle$ n. Editor and illustrator programs should use this feature to hide special information in **DL** that is not to be displayed when the document is printed. The writer of the control information is identified by the registered *type*, treated like the entity type.

$\langle Skip\text{-control-bytes-immediate} \rangle [n 1]$

This command is also provided for skipping control information that is recorded in the Press file. This command causes the n bytes following this command in the EL to be skipped.

$\langle Font \rangle + [font 1]$

This command, one byte long with the font number is the low order four bits, changes the current font. Characters displayed subsequently (e.g. with **Show-characters**) will use the specified font. At the beginning of an entity, the font is set to font 0 of the specified font-set.

$\langle \text{Set-space-x} \rangle [s \ 2]$

This command sets the value of the spacing in x. In all subsequent Show-characters and Show-character-immediate commands, if a space character (40b) is encountered, the font pattern for that character (normally blank) will be printed, but s will be used as the width of the character in determining where the next character will appear.

Example: When displaying a line of justified text, it is sufficient to say (Set-space-x> 140 (Show-characters) 72; the actual space width should, of course, be calculated so that the displayed line will have the correct length.

<Set-space-y> [s 2]

Similar to \Set-space-x>.

<Reset-space>

This command cancels the settings of space-x and space-y. Subsequently when a space character (40b) is printed, it is treated like any other character: the character (normally blank) is printed, and the width of the character in the current font is used to update the x and y positions.

<Space>

Insert a space by adding the values of space-x and space-y to the current position. If no values have been defined, or if there has been a $\langle Reset-space \rangle$, use the width and height values of the 40b character in the current font.

$\langle \text{Set-brightness} \rangle [b \ 1]$

This sets the brightness of all subsequent graphical items to be displayed. Black is 0; white is 377b. The scale is intended to correspond to the Munsell value parameter (see [5] for more information): b = 25*value.

$\langle Set-hue \rangle$ [h 1] (unimplemented)

This command sets the hue of all subsequent objects. The scale is intended to correspond to the Munsell hue parameter: there are 100 divisions of the Munsell hue circle, starting at 0R, and moving counter-clockwise through YR, Y, GY, G, BG, B, PB, P, and RP. The h value is simply twice the index of the division. Thus 6.5YR has h=33.

⟨Set-saturation⟩ [s 1] (unimplemented)

This command sets the saturation of the current hue. Lowest saturation is 0; highest is 377b. The scale is intended to correspond to the Munsell *chroma* parameter: s = 10*chroma.

$\langle Show-object \rangle [n 2]$

The next n words of DL are interpreted as a graphical object (see below for a description of the format of objects).

<Show-rectangle> [width 2] [height 2]

This causes a rectangle to be drawn on the page with lower left corner at the current (x,y) position. The parameters specify the size of the rectangle: width (in x) and height (in y).

$\langle Show-dots \rangle [n \ 4]$

This command points to n words in DL that are to be interpreted as an image sampled according to one of various coding formats, and to be displayed on the page (see below for a specification of dots). The image on the page is a rectangle with its lower left corner at the current (x,y) position.

An image described with **Show-dots** will only *add* marks to the page, and will never cause marks previously generated for other graphical items to be erased.

\langle Show-dots-opaque \rangle [n 4]

This command is similar to the **Show-dots** command, but the entire rectangular area of the page is *replaced* with the pattern specified by the image. This may cause marks previously generated for other graphical items to be erased.

⟨Alternative⟩ [EL-types 2] [EL-bytes 4] [DL-bytes 4]

Press permits a graphical effect to be represented in a number of ways. For example, a character "A" could be placed in a document by an appropriate <Show-characters>command, by a <Show-object> command with corresponding object descriptions, or by a <Show-dots> command with an appropriate image description. In some cases, it is useful to include several descriptions in the Press file. Then a printing or display program may select the one it is best able to handle.

The <Alternative> command marks such multiple representations in the EL. Alternatives that appear early in the EL are assumed to be preferred to those that appear late. Each alternative has three arguments: *EL-types*, which is a bit mask describing what sorts of EL commands appear in the fragment of the EL used to describe the alternative; *EL-bytes*, the number of bytes of EL commands used to describe the fragment; and *DL-bytes*, the

number of bytes of DL used by commands within the alternative. Following the last alternative is an Alternative command, with arguments 0. Alternatives may not nest. Example:

```
<a href="#">
<a href="#">
<a href="#">Alternative</a> 4000b 3 286
<a href="#">
<a h
```

The intention is that a program reading the Press file will parse only *one* of the alternative EL fragments, the first one whose *EL-types* argument assures the program it can deal with the alternative. Consequently, entity commands before and after the alternatives must be arranged to make sure that parsing any *one* of the alternatives will produce correct results.

The bits of *EL-types* are defined as follows (16-bit word):

100000ь	EL fragment contains (Show-characters), (Show-character-immediate), (Show-characters-short) or (Show-characters-and-skip) commands.
040000ь	EL fragment contains command.
020000b	EL fragment contains (Set-brightness) command.
010000b	EL fragment contains (Set-hue) or (Set-saturation) commands.
004000b	EL fragment contains (Show-object) command.
002000b	EL fragment contains (Show-rectangle) command.
001000р.	EL fragment contains (Show-dots) or (Show-dots-opaque) commands, with coding type 0.
000400ь	EL fragment contains Show-dots or Show-dots-opaque commands, with coding type non-zero.

$\langle \text{Only-on-copy} \rangle \begin{bmatrix} n & 1 \end{bmatrix}$

This command is included to permit information to be included selectively on certain copies of the document. Following an $\langle Only\text{-on-copy}\rangle$ command, entity commands will be ignored unless copy number n is being printed. If n=0, the subsequent entity commands apply to all copies.

<Nop>

A null operation.

<..available..>

These command codes are not assigned an interpretation by Press. They may be used by application programs to insert one-byte information into the EL. If more than one byte is required, use *Skip-control-bytes-immediate>*.

<..spare..>

These command codes are presently unused by Press, but are reserved for future expansion.

Various short forms of the above commands are available to permit tighter coding:

$\langle Show-characters-short \rangle + n-1$

A command to show n characters; $1 \le n \le 32$.

$\langle Skip\text{-characters-short} \rangle + n-1$

A command to skip n characters; $1 \le n \le 32$.

\langle Show-characters-and-skip \rangle + n-1

Equivalent to $\langle Show\text{-characters} \rangle$ n, $\langle Skip\text{-characters} \rangle$ 1. $1 \leq n \leq 32$. Useful for displaying short strings separated by spaces.

$\langle \text{Set-space-x-short} \rangle + [x \ 2]$

A short form of $\langle \text{Set-space-x} \rangle$; $0 \le x \le 2047$.

$\langle \text{Set-space-y-short} \rangle + [y \ 2]$

A short form of $\langle \text{Set-space-y} \rangle$; $0 \le y \le 2047$.

Objects

Graphical objects can be described by information contained in DL, and identified by the **Show-object**> command in EL. The information in DL records the outlines of the objects to be shown, in a fashion similar to PICO [2]. As mentioned earlier, objects may overlay each other, and the order of overlaying is determined by the order of pointers to objects from EL: the last object is always visible. All objects are described by outlines: there are no lines as such. The area enclosed by an object's outline is filled in with the color specified in **Set-brightness**>, **Set-hue>** and **Set-saturation>** commands. A single object may consist of a number of separate closed outlines; in this way, holes may be defined within objects. Further details of the semantics of the operators may be found in the PICO manual.

$\langle Moveto \rangle [x \ 2] [y \ 2]$

This command specifies the first point on one of the outlines of an object. The x and y coordinates are relative to Xe and Ye of the controlling entity. This command is equivalent to the PICO function call MOVETO(x+Xe,y+Ye).

$\langle\langle Drawto \rangle\rangle$ [x 2] [y 2]

This command extends the outline by drawing a straight line from the point defined by the previous command to (x+Xe,y+Ye). It is therefore equivalent to DRAWTO(x+Xe,y+Ye) in PICO.

$\langle\langle Drawcurve \rangle\rangle$ [Cx 4] [Cy 4] [Bx 4] [By 4] [Ax 4] [Ay 4]

This command draws a curved outline. It is similar to the PICO DRAWCURVE command, but the arguments are the coefficients of the parametric cubic equation for x and y. The equation for x, for example, is $X = Ax^*t^3 + Bx^*t^2 + Cx^*t + current-x$ -position. The coefficients are floating-point numbers (2 words; [4]).

Dots

This section describes how dots referred to by the $\langle Show-dots \rangle$ or $\langle Show-dots-opaque \rangle$ commands may be organized. The $\langle Show-dots \rangle$ or $\langle Show-dots-opaque \rangle$ commands point to a region of DL that describes a scanned image or some other form of bit-oriented data that is to be displayed in a rectangular region on the page. The lower-left hand corner of the region is the current (x,y) position at the point the $\langle Show-dots \rangle$ or $\langle Show-dots-opaque \rangle$ command is encountered in the EL. The region of DL cited by the EL commands contains commands that together describe the bits to be displayed.

<<Set-coding>> [code 1] [dots 2] [lines 2]

This command describes the coding form used, and also indicates where the scan-line breaks occur in the information. In all forms of coding, words are read sequentially as dots or runs, starting with the high-order bit of the first word. *Dots* defines the number of dots

per scan line; *lines* indicates the number of scan lines. The following interpretations are assigned to the *code* byte:

0. Bit map. The DL words specify a bit map. The interpretation of "0" bits in the data varies: a <Show-dots> command interprets such a dot as transparent; a <Show-dots-opaque> command interprets such a dot as white. "1" bits are interpreted as black.

1-16. Intensity samples. The data stream is interpreted as a sequence of n-bit bytes (n=code) defining intensity samples $(0=black, 2\uparrow n-1=white)$. Half-tone techniques may be used to print such an image. (If so, zero bits after halftoning will be transparent or opaque according to whether the dots were described by $\langle Show-dots \rangle$ or $\langle Show-dots - opaque \rangle$ commands.)

...additional types will be defined as needed...

$\langle\langle \text{Set-sampling-properties}\rangle\rangle$ [n 2] [data 2n]

This command is provided for careful control of printing parameters of continuous-tone images. The parameter n specifies how many data words of properties are given. At present, three properties are defined:

<<SSP-input-intensity>> [min 2] [max 2]. Set input intensity range. This property gives a range of sample values in the input image; the range will usually be mapped so that a value of min corresponds to black (<Set-brightness> 0); and max to white (<Set-brightness> 377b).

<<SSP-output-intensity>> [min-brightness 1] [min-hue 1] [min-saturation 1] [max-brightness 1] [max-hue 1] [max-saturation 1]. This property gives the output intensity range into which the input range should be mapped. The coding is the same as for <Set-brightness>, <Set-hue> and <Set-saturation>.

<SSP-screen>> [angle 2] [amplitude 2] [frequency 2]. This property sets parameters to be used when half-toning the image. The angle is the angle of inclination of the main axis of the halftone screen. Amplitude, expressed as a percentage of the image amplitude, controls the amount of screen signal that will be used. And frequency, expressed in halftone dots per inch, controls the screen frequency.

$\langle\langle Set\text{-window}\rangle\rangle$ [pd 2] [dd 2] [pl 2] [dl 2]

This command sets a window into the *definition* dots that is actually to be displayed. Of the lines in the encoding, *pl* lines are passed up, and then *dl* lines are displayed. On each *line* of the encoding, *pd* dots are passed up, then *dd* dots are displayed.

$\langle\langle \text{Set-mode}\rangle\rangle$ [m 1]

This command describes how the lines and dots of the encoding are to relate to the screen on which we are drawing. The value in m is dot-direction-description*4 + line-direction-description. Λ direction-description is:

0= to the right on the page

1= to the left on the page

2= to the top on the page

3= to the bottom on the page

Thus, if the dot-direction-description is 2, the dots in the input stream are to be windowed, and then placed starting at the bottom of the destination rectangle and proceeding successively up the page; if the line-direction-description is 0, the successive lines in the

input stream are to be placed at the left of the destination and successive positions to the right. In effect, the input stream can be rotated by any multiple of 90 degrees and mirrored, etc.

<<Set-size>> [width 2] [height 2]

This command sets the size, on the page, of the image of the "windowed" dots. Width and height are measured in micas. The lower-left corner of the image on the page is determined by the current (x,y) coordinate at the time the $\langle Show-dots \rangle$ or $\langle Show-dots-opaque \rangle$ command appeared in the EL. Note that this command may imply scaling the dots so that some number of dots, not unity, is printed for each dot specified in the file.

<< Dots-follow>>

The following words in DL contain the encoding.

<<Get-dots-from-file>> [fn arb] (unimplemented)

The dots are recorded on the file named by fn, a Bcpl string. The contents of this file will be treated exactly as if they had been referenced in DL, using the <<Dots-follow>> command.

<<Get-dots-from-Press-file>> [page 2] [fn arb] (unimplemented)

This command is similar to the $\langle\langle Get\text{-dots-from-file}\rangle\rangle$ command, but the file is assumed to be a Press format file. The dots are assumed to be contained in the DL for the specified page. As a result, the DL of the referenced file may contain commands such as those given immediately above which specify the coding. A $\langle\langle Dots\text{-follow}\rangle\rangle\rangle$ command should appear in that DL.

Note that all necessary window, mode and size settings should be made before the $\langle \langle \text{Dots-follow} \rangle \rangle$ or $\langle \langle \text{Get-dots-from-Fress-file} \rangle \rangle$, commands may appear either in the original Press file or in the referenced DL. For example, the size command might be in the original file and the window command in the referenced file.) Default values will be assumed if commands are omitted: these are (pl=0, dl=lines, pd=0, dd=dots) for window, and 8 for mode.

Command definitions

This section assigns numbers to all commands mentioned above. Each assignment is followed by "b" if it is one byte long, and by "w" if it is one word long.

Entity list commands:

<show-characters-short></show-characters-short>	0ь	b			
<skip-characters-short></skip-characters-short>	40b	b			
Show-characters-and-skip>	100b	b			
<set-space-x-short></set-space-x-short>	140b	b	(in	first	byte)
<set-space-y-short></set-space-y-short>	150b	b			byte)
	160b	b	`.		• ,
<available></available>	200b-237b	b			
<spare></spare>	240b-352b	b			
⟨Skip-control-bytes-immediate⟩	353b	b			
<alternative></alternative>	354b	b			
<only-on-copy></only-on-copy>	355b	b			
⟨Set-x⟩	356b	b			
<set-y></set-y>	357b	b			
- <show-characters></show-characters>	360b	b			
⟨Skip-characters⟩	361b	b			

<skip-control-bytes></skip-control-bytes>	362b	b
<show-character-immediate></show-character-immediate>	363b	b
<set-space-x></set-space-x>	364b	Ď
⟨Set-space-y⟩	365b	b
<reset-space></reset-space>	366b	Ď
<space></space>	367b	b
<set-brightness></set-brightness>	370b	b
⟨Set-hue⟩	37:b	b
⟨Set-saturation⟩	372b	b
<show-object></show-object>	373b	Ď
<show-dots></show-dots>	374b	b
<show-dots-opaque></show-dots-opaque>	375b	b
<show-rectangle></show-rectangle>	376b	b
<nop></nop>	377b	b
Data list commands:		
< <moveto>></moveto>	0	W
< <drawto>></drawto>	1	W
< <drawcurve>></drawcurve>	2	W
< <set-coding>></set-coding>	1	b
< <set-window>></set-window>	1	w
< <set-mode>></set-mode>	1 2 2 3	b
< <set-size>></set-size>	2	W
< <dots-follow>></dots-follow>	3	W
< <get-dots-from-file>></get-dots-from-file>	4	W
< <get-dots-from-press-file>></get-dots-from-press-file>	5	W
< <set-sampling-properties>></set-sampling-properties>	6	W
< <ssp-input-intensity>></ssp-input-intensity>	0	w
< <ssp-output-intensity>></ssp-output-intensity>	1	W
< <ssp-screen>></ssp-screen>	2	w

2.2 The font directory part

The font directory part consists of a number of entries, each entry containing instructions to the printing or display program for selection of fonts. These fonts may be drawn from standard source descriptions (see reference 4); some printing facilities will be capable of scaling and rotating such source descriptions by arbitrary amounts prior to printing. Fonts that are not available in source form, such as special symbols, may be defined graphically within the font directory. An entry in the font directory will typically define an entire font within one font set; it may, however, define only certain characters within the font, in which case a number of entries will be needed to specify the whole font.

The format of an entry in the font directory is as follows:

```
[entry-length 2] [font-set 1] [font 1] [m 1] [n 1] [fam 20] [face 1] [source 1] [size 2] [rotation 2]
```

The length of the entry in words is given in *entry-length* (this includes the length word). The two bytes *font-set* and *font* specify the set and font defined by this entry. The bytes m and n define the first and last characters to be defined; for example, to define the first 128 characters of a font, m should be zero, n should be 127, while to define a single character Λ , both m and n should be 101b. The 20 fam bytes contain an upper-case **Bcpl string** defining the family (METEOR, NONIE, etc.) to be used; unused bytes should contain zero. The face byte contains an encoding of the face (bold, italic,

etc.) to be used; this byte should be generated using the EncodeFace routine described below. Size and rotation define the size and rotation of the font; size is interpreted as a point size if positive, and as a size in micas if negative (72 points = 2540 micas). Rotation is in minutes of arc, measured anticlockwise (thus fonts should be rotated 90 degrees for landscape). The source byte indicates where to start reading characters from the source description; normally this will be the same as m.

A second format may be employed for entries that define characters not to be drawn from existing defined fonts, but from information contained within the entry:

[entry-length 2] [font-set 1] [font 1] [m 1] [all-ones 1] [zero 2] (objects)

The shape of character m of the font is specified by the (objects) commands, using the same commands as described above under **Objects**. The width of the character is defined by starting and finishing the shape description with $\langle Moveto \rangle$ commands. The two zero bytes should both contain zero to indicate a graphically defined font. The all-ones byte should contain 377b.

The last entry in the font directory is followed by a word containing zero.

2.3 The part directory

Each part is described by a four-word entry in the part directory:

word 0	Part	type,	interpreted	as	follows:
--------	------	-------	-------------	----	----------

0: Part is a printed page.

1: Part is the font directory part. Every printed document must

contain one and only one font directory part.

>1: ...more types to be defined as needed...

40: Part types less than zero may be used by applications programs to

include their own information in Press files. See the registry of

part types, in section 4.

word 1 Record number where the part begins.

word 2 Length in records of the information for this part

word 3 Interpretation depends on part type (word 0):

0: Length in words of the entity list padding (i.e., the number of

unused words at the end of the last record of the printed page).

others Undefined.

2.4 The document directory

This is a single 256-word record, and must be the last record of the file.

word 0	General password (27183)
word 1	Total number of records in this file
word 2	Number of parts, including the font directory part
word 3	Record number where part directory begins
word 4	Number of records occupied by the part directory
word 5	Back-pointer to obsolete document directory
word 6	-1 (unused)
word 7	-1 (unused)

Words 8 and 9 of the **document directory** may be used to indicate how many copies are to be printed; thus to print three copies, numbered from copy 1 to copy 3, word 8 should be 1, and word 9 should be 3.

Part 3: Width information

A key aspect of the Press format is that accurate formatting depends on the use of the same character width information by both the formatting and the printing programs. A public file FONTS.WIDTHS defines the widths of all fonts known to Press facilities. Width information will generally be stored in this file in a size- and rotation-independent form, consistent with "dictionary" conventions of the PrePress program [3]. User programs will be able to compute widths in micas for any character in any font at any size and rotation, using the following three Bcpl routines (in FontWidths.Bcpl):

face = EncodeFace(weight, slope, expansion)

The three arguments are ASCII characters: weight is M, B or L (\$M, \$B or \$L in Bcpl parlance) according to whether the font weight is medium, bold or light (default is medium); slope is R or I for regular or italic (default is regular); expansion is C, R or E for condensed, regular or expanded (default is regular). Arguments that have default values may be omitted or set to zero. This routine returns -1 if no encoding exists for the specified arguments. Otherwise it returns a value suitable for inclusion in the face byte of a file directory entry.

p = DecodeFace(face, 1v weight, 1v slope, 1v expansion)

This routine decodes *face* into three ASCII codes defining the weight, slope and expansion, stored in *weight, slope* and *expansion* respectively. It returns TRUE or FALSE depending on whether it can or cannot decode the *face* value.

p = LookupFontName(stream, "family", face, size, rotation, bufx, bufy, boundbox, [bufferlength, [lvCode]])

This routine requires opening *stream* to read FONTS.WIDTHS in word mode. This routine will look up the font whose family is defined by the **Bcpl** string "family" and whose face is encoded in *face* (as delivered by EncodeFace). *Size* and *rotation* should be specified as in font directory entries.

The width information is returned in two tables: bufx and bufy. The tables will be assumed to be 256 words long unless the argument bufferlength is present and has a different value. The widths in micas of the first bufferlength characters of that font, scaled and rotated as required, are stored in bufx and bufy.

In addition, a "bounding box" is computed in micas and stored in the 4-word vector boundbox. The bounding box can be viewed as the smallest rectangle that can be drawn around all characters of the font if they were positioned with their (0,0) points coincident. The first word of boundbox is the offset (negative to the left) from the (0,0) point of a character to the left-most visible piece of any character in the font. The second word is the offset (negative down) from the (0,0) point of a character to the bottom-most visible piece of any character (this is therefore the negative of the dimension of the largest descender). The third word is the width of the bounding box; the fourth

word is the height.

If no width information can be computed (e.g. if "family" contains an unknown name), this routine returns FALSE; otherwise it returns TRUE. If the optional parameter lvCode is supplied, it is the address of a variable that will be filled with a unique number that describes the font family name.

The format of FONTS.WIDTHS is described in reference 8.

Part 4: Conventions

By convention, an Alto screen dot measures 32 micas square.

Entity Type Registry

The following type codes have been registered to identify entities and control information:

Туре	Use	Registered owners
0	pure	Press. Entities of this type contain no type-dependent control information.
1	Cypress	Tesler. Used for mentioning labels and cross-references.
4	Draw	Baudelaire. Used to label information saved in Press files by Draw.
64-67	OfficeTalk	Newman.

Part Type Registry

The following type codes have been registered to identify special "parts" in a Press file:

Type	Use	Registered owners
0,1	pure	Press. These denote printed pages and font directory parts.
-1	Cypress	Tesler. Used to contain the style sheet, among other things.
-2 to -5	OfficeTalk	Newman. Used to contain information pertinent to the interpretation of the document.

History of Changes

This section lists briefly the changes that have occurred to this document.

Version of April 2, 1975. Original version.

Version of June 15, 1975. Deleted: <Show-xx-on-copy> commands. Added: <Show-character-immediate>, <Show-rectangle>, <Show-dots-opaque>, <Only-on-copy>, <<Get-dots-from-Press-file>>, entity type 3. Changed: order of arguments to <<Set-window>> command, lvCode parameter to LookupFontName.

Version of January 26, 1977. Added: Added: <a href="https://docs.nih.google-p

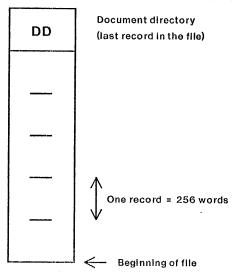
References

1. R.F. Sproull, "Press Overview," Xerox PARC Internal Memo, December 16, 1975. Filed on

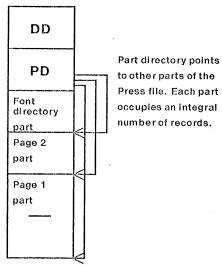
<Gr-Docs>PressOverview.Ears.

- 2. W.M. Newman, R.F. Sproull, "PICO Manual", Xerox PARC Internal Memo, July 1974.
- 3. R.F. Sproull, "PrePress Manual," Xerox PARC Internal Memo, January 1, 1977. Filed on PrePress.Ears">Gr-Docs>PrePress.Ears.
- 4. R.F. Sproull, "Font Representations and Formats," Xerox PARC Internal Memo, Marcl. 5, 1977. Filed on <Gr-Docs>FontFormats.Ears.
- 5. J.A.C. Yule, Principles of Color Reproduction, Wiley, New York, 1967.

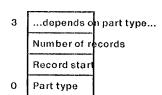
1. Overall Press File Format



3. Font and Page Parts



4. Part Directory Entry



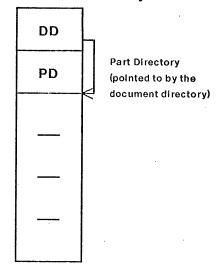
Each entry is 4 words long.

Part type = 0 for a "page part"

Part type = 1 for the font directory part

Part type < 0 for private part types

2. Part Directory



5. Typical Font Directory Entry

0		Rotation (minutes of arc)
10		Size (points if > 0)
1	0	Face source
ххх	xxx	
xxx	xxx	
С	Α	
Т	ı	
٧	E	
E	L	
9	Н	Family name (Bcpl string)
0	127	m n
0	1	Font set font number
16		Entry length in words
	1 xxx xxx xxx xxx T V E 9 0	10 1 0 xxx xxx xxx xxx xxx xxx xxx xxx C A T I V E E L 9 H 0 127 0 1

xxx represents "don't care" value

6. Printed Page Part Tentity list padding xxx(mentioned in part directory) Entity 3 -- any length EL Entity 2 -- any length Entity 1 -- any length A single 0 word divides DL and EL. xxx Arbitrary padding at end of DL XXX DL (Data List) for Entity 3 DL DL for Entity 2 DL for Entity 1

7. Typical Entity Entity length in words, including trailer Height Bounding box of items T Width in entity, measured r in micas **Bottom** a i Left Υe Coordinate offsets to apply е to each x,y value in entity r Byte-Double-precision (32-bit)length starting position and length of region of DL referred Beginto by the entity commands byte 0 10 Entity type | Font-set 32 377b ...Nop C 44 0 0 376b 1 Show rectangle, witch = 300, height = 32 m 13 14 ...Show characters short (15 chars) m a 356b 3 Set x 781 n 360b 72 Show characters (72 chars) d 0 140

Set y 140 (2 byte argument)

Set x 0 (2 byte argument)

Entity commands are parsed from beginning (bottom in this picture). A Nop can be used to fill out odd bytes as needed. Entities are located in the "printed page part" by scanning backwards from the end of the part, using the "entity list padding" information recorded in the part directory to start, and then observing the entity length entries in the trailer.

0

356b

357b

As the entity commands are processed, the pointer to data in the DL may be advanced. (In the example, Show characters 72 advances the pointer 72 bytes). The pointer is initialized by the "Begin-byte" entry in the trailer.

Entities are intended to be self-contained to facilitate copying entities from page to page or from document to document. It is for this reason that the trailer cites the range of DL used. Note that an entity can be "moved" on the page simply by changing Xe and/or Ye.