### Output Devices

**OUTPUT DEVICES AND COMPUTERS**

### Table I: Proof-Quality Devices

<table>
<thead>
<tr>
<th>Device</th>
<th>Lasergraphe</th>
<th>Pera/Canon</th>
<th>Quine/Sprint 5</th>
<th>Symbolics</th>
<th>Varian</th>
<th>Versatec</th>
<th>Xerox</th>
<th>Xerox 9700</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amdahl (MTS)</td>
<td>U. British Columbia</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Univ. Michigan</td>
<td></td>
</tr>
<tr>
<td>Apollo</td>
<td>COS Info.</td>
<td>OCLC</td>
<td></td>
<td></td>
<td>OCLC</td>
<td>COS Info.</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Ethernet</td>
<td>Stanford, Image, Inc.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Stanford</td>
<td></td>
</tr>
<tr>
<td>DEC10 *</td>
<td>Vanderbilt*, Talaris*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Vanderbilt*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEC10</td>
<td>Stanford (Sai)</td>
<td>Talaris</td>
<td></td>
<td></td>
<td>G A Technology</td>
<td></td>
<td>Univ. Delaware</td>
<td></td>
</tr>
<tr>
<td>DEC20</td>
<td>Math Reviews</td>
<td>SRI; Columbia</td>
<td>Talaris</td>
<td></td>
<td>AMS</td>
<td>CMU</td>
<td></td>
<td></td>
</tr>
<tr>
<td>DG MV8000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HP1000</td>
<td>JDI Wordware</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>HP3000</td>
<td>TeXt</td>
<td>TeXt</td>
<td>TeXt</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>IBM (MVS)</td>
<td></td>
<td>SLAC</td>
<td></td>
<td></td>
<td></td>
<td>CIT</td>
<td></td>
<td></td>
</tr>
<tr>
<td>IBM (VM)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Prime</td>
<td></td>
<td>Texas A&amp;M</td>
<td></td>
<td></td>
<td>Livermore</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Siemens BS2000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>GMD Bonn</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sun</td>
<td>Textset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>VAX (Unix)</td>
<td></td>
<td>UC Irvine</td>
<td>Talaris</td>
<td>UC* Santa Cruz</td>
<td>Cal. Tech.; Univ. Washington</td>
<td>Stanford</td>
<td></td>
<td></td>
</tr>
<tr>
<td>VAX (VMS)</td>
<td>Inter-</td>
<td>INFN CNAF*</td>
<td>Inter-</td>
<td>Argonne*</td>
<td>Texas A&amp;M</td>
<td>Calma*</td>
<td>Sci. Appl.</td>
<td>Inter-</td>
</tr>
</tbody>
</table>

**Notes:**
- * Still running TeX80
- † Graphics supported

Most of the interfaces listed here are not on the standard distribution tape. Some of them are considered proprietary. Information regarding these interfaces should be obtained directly from the sites listed.

Output device data is being maintained by Rilla Thedford. Anyone desiring more information or relaying new information can now send it to her on the Arpanet:

Rilla_Thedford@UMich-MITSGMIT

### Table II: Typesetters

<table>
<thead>
<tr>
<th>System</th>
<th>Alphatype CRS</th>
<th>APS-5/Micro-5</th>
<th>Compugraphic 9400</th>
<th>Compugraphic 8600</th>
<th>Harris 7500</th>
<th>Linotron 202</th>
</tr>
</thead>
<tbody>
<tr>
<td>Amdahl (MVS)*</td>
<td>Washington St U*</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Apollo</td>
<td>COS Info.</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>CDC Cyber *</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>DEC 20</td>
<td>AMS</td>
<td>Textset</td>
<td></td>
<td></td>
<td></td>
<td>Adept, Inc.*</td>
</tr>
<tr>
<td>HP3000</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Univ. of Sheffield</td>
<td></td>
</tr>
<tr>
<td>IBM 370 *</td>
<td></td>
<td></td>
<td></td>
<td>Info. Handling*</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Sun</td>
<td>Textset</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Univac 1100 *</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Univ. of Wisconsin*</td>
<td></td>
</tr>
<tr>
<td>VAX (UNIX)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>SARA</td>
<td></td>
</tr>
<tr>
<td>VAX (VMS)</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td>Intergraph †</td>
<td></td>
</tr>
</tbody>
</table>
Camera copy for the following items in this issue of TUGboat was prepared on the devices indicated, and can be taken as representative of the output produced by those devices.

The bulk of this issue, as usual, has been prepared (using $\TeX$X82 for pages 4-11 and $\TeX$X80 otherwise) on the DEC 2060 and Alphatype CRS at the American Mathematical Society.

- Autologic APS Micro-5: Description of Metafont Generic Font file format and Gray fonts for Metafont proofs, p. 31; DEC 2060.
- Compugraphic 8600: Dean Guenther et al., $\TeX$ at Washington State University, p. 24; Amdahl V/8 (MVS).
- Compugraphic 8600: Goffredo Haus, How to Tame Your Phototypesetter by $\TeX$, p. 17; Univac 1100/80.
- Harris 7500: Han Noot, DVI-Code to the Harris 7500, p. 18; $\TeX$ on CDC Cyber 175-750, typesetter driven by VAX 11/780 (Unix) and/or PDP 11/45.
- HP 2688A Laser Printer (300 dpi): Susan Daniels, The HP $\TeX$ Macros, p. 49; HP 3000.
- Imagen Imprint-10 (240 dpi): G.M.K. Tobin, The OCLC Roman Family of Fonts, p. 36; Apollo.

HOW TO TAME YOUR PHOTOTYPESETTER BY $\TeX$

Goffredo Haus


viale Pisa, 10

I-20146 Milano (Italy)

I have implemented $\TeX$X-1100 from MACC on a UNIVAC 1100/80 mainframe computer at C.I.L.E.A. (Segrate, Milano).

I have used a Compugraphic 8600 phototypesetter as a quality output device and various low-cost graphic devices to edit documents.

I have read the CG8600 User Manual and I have found a lot of features which seem to be very useful to produce my documents but, unfortunately, $\TeX$ is not able to control them.

So, I have successfully tried to deceive $\TeX$: I have built (by means of the FONTPRE program from MACC) a TFM file in which ASCII codes correspond to particular CG8600 commands to be treated as zerowidth, zerodepth, zeroheight characters by $\TeX$.

The CG8600 driver program SETTEX (from MACC) recognizes the font code and produces a command string suitable for the phototypesetter according to the character code.

For example, I can write white-on-black or I can get SmallCapsItalic from SmallCaps font.

This simple trick, allows me to completely control CG8600 directly from the $\TeX$ source of my document by means of $\TeX$ processor and SETTEX driver program using my special TFM-commands file.
DVI-CODE TO THE HARRIS 7500

Han Noot

Introduction

In 1981 it was decided that at SARA (Stichting Academisch Rekencentrum Amsterdam) \TeX would be installed on a Cyber 175-750 under NOS/BE 552. SARA is a computing center which provides its service to Amsterdam's two universities and to Stichting Mathematisch Centrum (the author's institute) among others. \TeX's DVI-code was to be processed at SMC, where a Harris 7500 phototypesetter, driven by a VAX 11/780 and a PDP 11/45 are available. (On both computers the UNIX operating system runs.) The filter which converts DVI-code to Harris 7500 code as well as the various programs for the creation and maintenance of font metric files would be implemented and run under UNIX at SMC.

At SARA, one obtained a Cyber version of \TeX from Erik Bertelsen, RECAU, Aarhus, Denmark. We at SMC obtained a \TeX version for the VAX under UNIX from Robert Morris, University of Massachusetts, Boston, USA.

The Harris 7500 phototypesetter

The Harris 7500 phototypesetter is a CRT machine with typefonts stored in digital form on an internal system disc. It produces output on photographic paper or film. Its setting speed highly depends on character size and the number and size of moves between characters; the estimated speed is between 200 and 300 characters/second when the typesetter is driven through a 9600 Baud serial interface. Characters can be explicitly positioned with an accuracy of 0.05 points in both the horizontal and vertical directions.

When used in so-called 'slave mode', the Harris 7500 accepts a set of commands which is an almost ideal target language for the translation of DVI-code. Its main features that interest us here are:

- Horizontal- and vertical move commands which perform displacements relative to the current position in the film plane. They come in two flavors: utilizing absolute machine units (0.05 points) or relative units (1/72 of an 'em' at the current point size).

- Horizontal and vertical position commands relative to the origin. (A user defined point in the left margin.)

- Characters can be typeset with- or without automatic updating of the writing beam position.

- Two character size definition commands. The first fixes the overall pointsize of the characters (in units of 0.1 points), the second can override the pointsize in the horizontal direction as determined by the first. As a result, horizontal- and vertical pointsize can be set independently.

- A rule command for hardware generated rules.

- Slanting hardware to produce slanted versions of non-slanted characters. There are three possibilities: rotation through 9, 12 and 15 degrees. (This feature only gives typographically good results with sans-serif characters.)

- Characters are grouped into fonts of at most 128 characters each.

A naive approach

For every \TeX font we need two different font information files: the familiar \TeX font metric file and a file to be used by the DVI-code translator. This last file contains information on the correspondence between \TeX characters and Harris characters and the width of every character. (Width information is included because the code translator must be able to update the horizontal position after typesetting a character.)

We have designed a symbolic human readable font description from which both font information files can be generated by program. These programs use a data base of height-, depth-, and width values for all the characters that are available on our typesetter. (We obtained this information through the help of the Harris corporation.)

The symbolic font description consists of:

- a header section,
- the character mapping section,
- ligature specifications,
- kern specifications,
- lists of characters of ascending size,
- extensible character definitions,
- a font parameter part.

The header section just contains the font name, design size, character coding scheme and whether the font is a 'NOS-\TeX' font or a 'UNIX-\TeX' font (see below).

The character mapping section contains one line for every character in the \TeX font. Such a line has the form:
Nint Fname Cint Sdig Ifrac Pint Hint
in which N, F, C, S, I, P and H are fixed key-letters.
The meaning is the following:
Noct: We are talking about the character with octal
number oct (0 <= oct <= 177) from the \TeX font being defined here. (We use octal numbers
because they are used in the font examples in
appendix F of the \TeX manual too.)
Fname: This character is represented by a character
taken from the Harris font indicated by name.
Cdec: We use character number dec (0 <= dec <= 127) from this Harris font. (That we have a
decimal number here is for easy correspondence
with Harris documentation.)
Sdig: The character must be set slanted by 9, 12 or
15 degrees (dig = 0, 1 or 2).
Ifrac: The italic correction is frac \ast character-width,
where frac is a decimal fraction.
Pint: The typesetter must set this character at
pointsize int (specified in units of 0.1 points).
Hint: The horizontal pointsize must be int, instead
of the value specified by the P-field.
The ligature specification consists of one line of
the form:
Noct1 Noct2 Loct3
for each ligature, indicating that \TeX chars oct1 and
oct2 are to be combined to ligature oct3.
Kern specifications are of the form:
Noct1 Noct2 Wfrac
indicating that the kerning value for \TeX characters
oct1 and oct2 is decimal fraction frac of the width of
char oct1.
Next come lists of characters of increasing size
(e.g. a list of left braces). There is one list per line.
Extensible characters are specified by lines of the
form:
Noct1 Toct2 Moct3 Boct4 Eoct5
saying that \TeX character oct1 is extensible, having
as top part character oct2, as middle part character
oct3, oct4 as bottom part and oct5 as extension component.
This symbolic \TeX-font description together
with the data base of Harris character dimensions
was thought to contain all the information needed
to generate \TeX font metric files by program, but
a few somewhat unexpected problems cropped up.
They required an extension of the character mapping
definition. We deal with them in the next section.
Meanwhile we mention that while generating
font metric files, our program performs various con-
istency checks. For instance, it is checked whether
a character is not both declared to be the first of a
ligature pair and the first of a kern pair. (By the
way, why does \TeX not allow such - albeit theoretical-
possibilities?)

The device font files for the DVI-code translator
are generated using only the header- and character
mapping parts of the symbolic font specification.
They contain a machine readable form of the N, F,
C, S, P and H fields of the character mapping lines.
Furthermore, they contain the width of the charac-
ters, both in \TeX's rsu's and in typesetter units.
These twofold width values are used to eliminate
rounding errors. As soon as the theoretical (correct)
position on a page, measured in rsu's differs from the
physical (rounded) position measured in typesetter
units (0.01 point) by more than one typesetter unit,
the typesetter is instructed to make a compensating
move.

A complication with device font files is that we
have to generate different files depending on whether
we use \TeX running under NOS or \TeX under UNIX.
This is so, because our 'NOS-TeX' and 'UNIX-TeX'
use different rsu's, namely a 2⁻¹₆ point rsu respec-
tively a 2⁻¹⁸ meter rsu. Hence our DVI-code trans-
lator unfortunately has to be aware of which type of
DVI-code it is translating. This is moreover so be-
cause the two \TeX versions produce slightly different
postamble formats in their DVI-code.

Complications

The font description scheme discussed so far
would work fine, if not for a number of practical
complications, which we will discuss now.

In the first place, Harris characters are generated
by digitizing hand drawn art work. The result is that
even in a straightforward typefont like Times Roman,
not all capitals are of exactly the same height nor do
they have equal depth. The same applies to lower
case characters: 'p' does not always have its height
exactly equal to that of 'r' or its depth equal to that
of 'g'. As a consequence, font metric files cannot
be generated by directly using character dimensions
from the data base (after some unit conversions).
That way we would exceed by far the limit of six-
ten different depth- and height- values per font as
allowed by \TeX. We considered automatic round-
ing to sixteen different values but had to discard this
for the following reason: In the extension font, ex-
tension components should have an (almost) exact
depth, while the height and depth of for instance the
'+' operator may be wrong by quite a lot. A symbol
like 'Σ' is an intermediate case. So rounding must
be done under explicit human control.
To make that possible, we introduced into the character mapping lines an optional A-field (A comes from as) which can take one of the following forms:

Aarg, Adarg, Aharg, or Ahargdarg

in which arg is either an octal number or a period. If a line reads:

N157 ... A141

it means: take the height and depth of character 157 equal to the height and depth of character 141 from this same \TeX{} font. Ahidj means: take the height equal to that of character i and the depth equal to that of character j. Adj means: give the character its own height but take its depth equal to that of character j. Finally, a period instead of a character number (e.g. A. or Ah. etc.) means: take the character dimension indicated by the period equal to zero.

The next complication arose from the fact that in \TeX{}'s extension font, components of extensible characters must have a height of zero while the corresponding Harris characters intersect the base line.* Furthermore, symbols with limits (like $\Sigma$) only come out right when they have a height of zero too. As a remedy, we introduced an optional U (up) field in the character mapping lines. It is more general than is strictly needed, but can be turned to good use in other circumstances. This U field comes in four forms:

\text{Ud, Uu, Udfrac or Uufrac}

meaning the following: Ud pushes the Harris character representing the \TeX{} character so much down, that its top touches the base line. Udfrac pushes the character down by an amount equal to frac * (character-height + character-depth), in which frac is a decimal fraction. Uu and Uufrac work like Ud and Udfrac, but now in the upward direction.

When all this was done, two problems remained: there were (just visible) gaps between occurrences of some extensible character components and between objects like root signs and rules connected to them. The problem with extension characters seems entirely due to the fact that there apparently are minor discrepancies between character height and -depth as given in the typesetter documentation and their real physical dimensions. We introduced an E (epsilon) field for that. It tells the font metric file generating program to take the height and depth of a character a fraction 'epsilon' (in our case 0.02) smaller than

the values contained in the character dimension data base. This produces just enough overlap between extensible character components to make the extensible character come out fine.

The problem with roots (and horizontal braces) is due to the fact that there is sometimes a bit of white space at the left and right of certain typesetter characters where we do not want it. Solution: the R- (reduce white space) field. It has the form:

Rfrac1frac2

(The order of the lfrac- and rfrac parts may be reversed and one of them may be omitted.) The meaning is: reduce the white space to the left of a character by frac1 * character-width and to the left by frac2 * character-width. (frac1 and frac2 are signed decimal fractions.) This is done by having the typesetter perform small horizontal moves before and after setting the character. It solves the problem with roots and furthermore it can be used more generally to make differently spaced fonts out of existing ones.

To sum up: apart from the fields discussed in the previous section, a character mapping line may contain:

- an A field to specify that character dimensions must be taken equal to those of other characters,
- an U field to specify vertical displacement of the character,
- a R field to specify changes in the amount of white space surrounding the character,
- an E field to specify that the character height and -depth must be taken somewhat smaller than the physical values.

A character mapping line must contain a N, F and C field; it may contain S, I, P, H, U, R, E or A fields. When font metric files and device font files are generated from the symbolic font description, the order of processing of the various fields of a character-mapping line is crucial. It is the order in the summary of the fields just given. First the character height, -depth and -width are obtained from the data base. Height and depth are updated according to the ratio of the character pointsize (P-field) and the pointsize for which the data base values are applicable. The same is done for the width, using the horizontal pointsize if a H field is there, otherwise using the P-field. Next, a width value may be affected by a R-field; height- and depth values are modified by U-, A- and E-field values, in that order. Finally all values are converted to proper units and inserted in the device font- and font metric files.

The rationale for this processing order is the following: First the precise dimensions of the physical character as a result from pointsize specification,
up/down moves etc. are calculated. Only thereafter, deliberate errors may be introduced by E- and A-field values.

The specification of a symbolic font should proceed in corresponding stages. First, the font is specified without R-, E- and A-fields. After inspection of a human-readable version of a font metric file (probably still containing too many height or depth values), A-fields are introduced as needed. Thereafter, on evidence from the typeset output, R- and E fields can be added. E-fields may only be added to character specifications not containing A-fields or to all characters which have the same height and depth value (taking into account their A-fields). If this restriction is violated, extra height- and depth values may again appear.

Finally, it may be useful to sum up which information from character mapping lines is reflected in font metric files and which in device font files. First the font metric file: The F and C fields together with the character data base determine initial character dimensions. These are then acted upon by eventual P, H, U, R, E and A field values, so all these fields may influence the contents of a font metric file. Furthermore, the I- (italic correction) field is used in this file too. On the other hand, only the F, C, P, H, I, U and R fields determine the contents of the device font file. F, P, H and I specify the state in which the typesetter must be while typesetting the character, the U and R fields describe moves to be performed in the film plane before and after setting the character.

Miscellaneous topics

To assist in the task of creating and maintaining symbolic font description files, Gertjan Vinksteijn has implemented a kind of special purpose editor. This program interactively asks for the values of the various font description fields, checks the response for correct format, range in which values may lie, etc.

The translator program, which generates typesetter code from DVI code can be called with a number of optional arguments which are used to specify:

- whether 'NOS-DVI' or 'UNIX-DVI' code must be processed,
- which pages from the document must be typeset (all of them, only individual ones, ranges of pages) and in which order,
- whether some special symbol must be typeset in the margin in order to demarcate pages (to assist in cutting them later on) and if so which one,
- whether all pages must be set to the length of the largest one or to their individual length (which saves some quite expensive photographic material but which may annoy a printer).

As already stated, all programs are implemented under UNIX in the programming language C. There is no fundamental reason for this however. Everything could have been equally well coded in say PASCAL, but we quite heavily use the UNIX system call which makes possible to move freely forward or backward to any position in a file.

Conclusions

The claim made by some that DVI-code can be used to generate code for almost any reasonable output device has not been falsified by our experience. On the contrary, it was quite straightforward to transform DVI-code to code for the Harris 7500. On the other hand, there is quite a lot of 'device dependence' (METAFONT dependence?) hidden in the limitations on font metric files and especially in the peculiarities of \LaTeX{}'s extension font. In particular, the 'zero-height' requirement for certain characters considerably expands the amount of code to be generated for our typesetter. To typeset one extension component seven bytes must be sent across the typesetter-interface. The first three generate a downward move (one opcode, two operand bytes), the fourth byte is the actual character and next come three bytes to move up again. (Some of these moves could be optimized away, but only at the cost of complicating the device font files with otherwise unnecessary height- and depth-values of characters.) We wonder whether the reason (unknown to us) for the zero-height requirement is compelling enough to justify the complications it introduces in driving output devices and generating font metric files. Furthermore, could there not be two types of font metric files: one of the current compacted kind and one allowing 128 different values for every character dimension? In spite of its cost in memory, we certainly would have used the latter type for the extension font. Meanwhile we are curious to know, whether there are others who have encountered problems analogous to the ones described here.

To end with a note of optimism, when everything finally worked, we indeed got typeset output which looked quite attractive.