* * * * * * * * * * *

Interface Software

* * * * * * * * * * *

ERRATUM: THE FORMAT OF TEX'S DVI FILES

David Fuchs

Editor's note: Several erroneous values were given in a command description which appeared in TUGboat Vol. 1, No. 1, *page* 18. *The full corrected entry is given below.*

Command	Description
129 BOP n<4> p<4>	Beginning of page n, with pointer p to the BOP command of the *previous* page. By "pointer" is meant the relative byte number within the DVI file, where the first byte (the BOP of the first page) is byte number zero. (ex.: If the first page had only a BOP and EOP, the *third* page's pointer would be 10, because the BOP command takes bytes 0 to 8, the EOP is 9, so the *second* page's BOP is in byte 10. Get it?). The *first* page has a —1 for a pointer; the second, a zero. Start the H- and V-coordinates out at 0, as well as the w-, x-, y-, and z-amounts. The stack should be empty, and no characters will be set before a FONT(NUM) command occurs. Remember that n can be < 0, if the page was Roman Numbered. Also the pages need not come in the proper order in the file, depending on who's doing the TEXing.

* * * * * * * * * * *

TEX-PASCAL AND PASCAL COMPILERS (A STATUS REPORT)

Ignacio Zabala

Following the original SAIL program, TEX-Pascal has suffered some modifications since our previous report of September 1980 (see TUGboat Vol. 1, No. 1, p. 16), but the compiler requirements stated then are still perfectly valid.

The information that has reached us about the latest release of the program indicates three sources of difficulties for some installations:

- The TFX (font information files) that were distributed contained packed information in units of 36 bits.
- The program uses packed records that have to be stored in a single machine word.
- CASE statements contain a default case which is not standard in Pascal.

The first problem is solved in the next release, because TEX now uses TFM (TEX font metrics) files, which have only 32 bits of information per word.

The second problem, which really affects the amount of memory employed by the program, is not easy to solve without slowing TEX down. Essentially, the TEX-Pascal program expects that a structure of the type:

```
  memoryword = PACKED RECORD CASE 1..4 OF
    1: (pnts: REAL);
    2: (int: INTEGER);
    3: (twohalves: halves2);
    4: (fourbytes: bytes4)
    END;
where
  halves2 = PACKED RECORD
    lword: 0..65535;
    CASE 1..2 OF
      1: (rhword: 0..65535);
      2: (byte2: 0..255;
          byte3: 0..255)
    END;
and
  bytes4 = PACKED RECORD
    byte0: 0..255;
    byte1: 0..255;
    CASE 1..2 OF
      1: (rhword: 0..65535);
      2: (byte2: 0..255;
          byte3: 0..255)
    END;
```

is stored in a single 32-bit word. It does not make any assumption on the order in which fields are stored: if they are put in the structure and retrieved from it in exactly the same order, the result should be correct.

If the compiler does not pack as expected, it does not necessarily mean that the program will not run correctly, but that it may require an enormous amount of memory (and this may cause the program not to run at all!!).

Some installations that found the third problem (CASE statements) have solved it by hand-editing the program, and it is not yet clear whether it will be fixed once and for all in the sources.

In all, the results are encouraging. Besides Stanford, there are reports of the program running at four other sites:

- On an IBM 370/3033 with Pascal/VS at Stanford CIT (Eagle Berns).
- On a VAX (VMS) at Oregon Software (Barry Smith).
- On an IBM 370/3022 (VM-CMS) with SLAC-Pascal at the University of Pisa (Gianfranco Prini). They printed the DVI files on a Versatec.
- On a Univac 1100/82 at the University of Wisconsin (Ralph Stromquist). Output is to a Compugraphic 8600. (See report, p. 51.)

From the information sent to Stanford, we gather that the Pascal compilers being employed in the installations of TEX are:

IBM 370: Pascal-VS, SLAC-Pascal, Pascal-8000
UNIVAC: U. of Wisconsin Pascal, Pascal-8000
PDP-10: Hamburg Pascal
VAX: (See report by Janet Incerpi, p. 49.)

Note: Charles Lawson (Jet Propulsion Lab.—Caltech) has produced two short reports that can help in reprogramming the SYSDEP module of TEX-Pascal. (Both are reprinted in this issue, pp. 20 and 32.)

* * * * * * * * * * *

TEX FONT METRIC FILES
— or —
What happens when you say "\font A=CMR10"

David Fuchs

When you tell TEX that you will be using a particular font, it has to find out information about that font. It gets this information from what are known as .TFM files. For instance, when you say `\font A=CMR10` to TEX (`\:A=CMR10` in the old TEX lingo), TEX looks around for a file called CMR10.TFM, and reads it in. If CMR10.TFM is not to be found, TEX will give you the error message "`Lookup failed on file CMR10.TFM`", and you will be out of luck as far as using CMR10 is concerned.

What does TEX want with the TFM files? Generally speaking, a font's TFM file contains information about the height, width and depth of all the characters in the font, plus kerning and ligature information. So, CMR10.TFM might say that the lower-case "d" in CMR10 is 5.55 points wide, 6.94 points high, etc. This is the information that TEX uses to make its lowest-level boxes—those around characters. See the TEX manual (p. 41) for information about what TEX does with these boxes. Note that TFM files do NOT contain any device-dependent description of the font (such as the raster description of the characters at a certain resolution). Remember that the program TEX does not deal with pixels. Only device-drivers that read TEX's DVI output files use that sort of information.

Where do .TFM files come from? The best way to get a TFM file is with **METAFONT**. Font designers should include the **METAFONT** instructions that specify the width, height, etc. of each character they design. The **METAFONT** manual contains details and examples of how to do this—see the index entries for **charwd**, **charht**, **chardp**, etc. If this is done, then when **METAFONT** is run on CMR10, it produces CMR10.TFM. (Depending on what "mode" it is run in, it also makes CMR10.FNT, CMR10.ANT, CMR10.VNT, or CMR10.OC. These are all different formats for files containing the raster description of the font. Drivers for various devices require one or another of these files.)

Whatever happened to the TFX format that the TEX and **METAFONT** manuals actually refer to? Is this just a misprint for TFM? No—TFM files take the place of TFX files. The differences are conceptually small; they both contain more or less the same information. The main reason for changing TEX and **METAFONT** from using/making TFX files to TFM files is that TFX files were based on 36-bit words. This proved to be a real problem for people running Pascal TEX, especially on 32-bit machines. The format of TFM files assumes 8-bit bytes, packed four to a 32- or 36-bit word. They are readily adapted for use on 16-bit machines, too. While the format was being changed, a few new bits of information were added, too.

What if I have fonts that I want TEX to know about that were not made with **METAFONT**? Don't despair—we have two programs, TFTOPL and PLTOTF, that convert TFM files to readable, editable format, and back again. For instance, if we run TFTOPL on CMR10.TFM, it makes CMR10.PL, an excerpt of which follows ("`R`" means a floating-point number is coming up (all dimensions given in this form are in terms of the `DESIGNSIZE`); "`C`" means that a character is next; "`O`" means an octal value for a character that isn't an ASCII printing character is next):

```
(FAMILY CMR)
(DESIGNSIZE  R  10.0000000)
(CODINGSCHEME TEX TEXT)
(TEXINFO
        (SPACE R   .3333330)
        (STRETCH R   .1666670)
        (SHRINK R   .1111107)
        (XHEIGHT R   .4444447)
        (QUAD R   1.0000000)
        (EXTRASPACE R   .1111107)
        )
```