TEX USERS GROUP 1982 WINTER MEETING

Monday and Tuesday, January 11–12, 1982
Stouffer's Cincinnati Towers, Cincinnati, Ohio

A TEX Users Group meeting will be held to discuss TEX issues of general interest. The Steering Committee will also meet with the membership to discuss dues and the future development of the Users Group.

The meeting will cover three areas of interest:

- TEX-in-Pascal, with demonstrations of TEX on the Canon Laser Beam Printer,
- macro packages (both development and exchange methods), and
- output devices and interfaces.

Manufacturers of phototypesetters are invited to discuss their equipment and its TEX compatibilities. Seminars for the different computer architecture groups will be mediated by the Site Coordinators.

Donald Knuth will speak on a subject yet to be decided. Other members of the Stanford group will also be present, both as speakers and to participate in computer and output device sessions.

The first copies of "version 0" of *The Joy of TEX* (the AMS-TEX manual), are expected to be available for sale.

Pre-register early as the hotel reservation deadline is December 15, 1981. A meeting registration form is included with this issue. For additional information, contact

Thomas Pierce
'82 Winter Meeting Coordinator
Rohm and Haas Research Laboratories
727 Norristown Road
Spring House, PA 19477

* * * * * * * * * * * *

Software

* * * * * * * * * * * *

PASCAL-CODED TEX ERRATA

Arthur L. Samuel
Stanford University

The rate at which errors are being reported for the "PASCAL-coded version of TEX" has slowed down to the point that we seem to be approaching that famous last bug. All known errors have been corrected in the DOC and PASCAL files available from Stanford and an August 1981 revision of the descriptive documents has been printed. There are a few instances where suggested changes (for example, as in the naming of variables) have not yet made.

One correction, made since August, is repo below. Also reported is a suggested improveme procedure *PrintOctal*.

37. The following procedure should work on 36-bit and 32-bit machines and specifically on t machines where the previous PrintOctal proce caused overflow problems. This may be use replace procedure PrintOctal; in section 37 on 18 of the SYSDEP module.

```
procedure PrintOctal(n : integer); { Prints the
        rightmost 32 bits of an integer in oct
  var i,k : integer;
    s : array [0..10] of asciiCode;
    msb,mbb : boolean;
  begin msb := false; mbb := false;
  if n < 0 then
    begin n := -n; msb := true
    end;
  for k := 10 downto 0 do
    begin i := n mod 8;
    if k = 0 then
      begin if msb then
        begin if 1 > 3 then i := 7 - i
                          { it was a 36-bit wo
        else begin if (i mod 2) = 0 then i := 3
          else i := 2
          end
        end
      else i := (i mod 4)
      end
    else begin if msb = true then
        begin if mbb = true then i := 7 - i
                    { a borrow has propagate
        else begin if i > 0 then
            begin i := 8 - i; mbb := true;
                        { a borrow requir
          end
        end
      end;
    end;
  case i of
    0 : s[k] := zero;
    1 : s[k] := one;
    2 : s[k] := two;
    3 : s[k] := three;
    4 : s[k] := four;
    5 : s[k] := five;
    6 : s[k] := six;
    7 : s[k] := seven
    end;
  n := n div 8;
  end;
Print(" "); k := 0;
while (k < 10) and (s[k] = zero) do
    Increment(k);
for k := k to 10 do Print(s[k])
end;
```

453. An error has been found and fixed in the m operating module of the PASCAL-coded versio

TEX as described in section 453 on page 157 of the August 1981 revision. This error affected the use of *leqno*. The correction involves replacing the word *linq* with *link* and adding a missing line of *shift* := 0.0; as shown below:

```
453. ⟨Attach equation number 453⟩ =
  begin q := getnode(gluenodesize);
  typ(q) := gluenode; gluelink(q) := fillglue;
  if leqno then
    begin link(q) := b; link(eqnobox) := q;
    b := hpack(eqnobox, dw − shift, false);
        { eqno will be left-justified }
    shift := 0.0;
    end
  else begin link(q) := eqnobox; link(b) := q;
    b := hpack(b, dw − shift, false)
        { eqno will be right-justified }
    end
  end
This code is used in section 444.
```

* * * * * * * * * * *

THE FORMAT OF PXL FILES

David Fuchs

A PXL file is a raster description of a single font at a particular resolution. These files are used by driver programs for dot matrix devices; TEX itself knows nothing about PXL files. Let's say a user creates a file called FOO.MF, which is the **METAFONT** language description of a new font, called FOO. In order for everyone to be able to run TEX jobs that use this font and get their output on our 200-dot-per-inch proof device, we must first run the **METAFONT** program on FOO.MF, and ask it to make both a TFM file and a PXL file for it. These files (called FOO.TFM and FOO.PXL) are then put in a public directory so that anyone using TEX may access them. Now, whenever a TEX job is run that refers to FOO (`\font A=FOO`), the TEX program reads in FOO.TFM to get all the width, height, depth, kerning, ligature, and other information it needs about any font. To get output on the proof device, the DVI file produced by TEX must now be processed by a device-driver program. This program reads the postamble of the DVI file to find out the names of all the fonts referred to in the job, and for each font, it opens the corresponding PXL file. In our example, the driver would find the file FOO.PXL, which it would then use along with the main body of the DVI file to produce the actual output. The DVI file tells where to put all the characters on each page, while the PXL file(s) tell which pixels to turn 'on' in order to make each character.

In fact, there is a little lie in the preceding paragraph. The actual name of the PXL file would be something like FOO8.1000PXL. This means that the PXL file represents the font FOO in an 8-point face for a 200-dot-per-inch device with a magnification of 1. (If you don't fully understand the term 'magnification' as it is used in the TEX world, the rest of this paragraph might not make a lot of sense. The end of this document contains more information on magnified fonts.) If we also had a 100-dot-per-inch device, we would also want to have the file FOO8.0500PXL (which we can get by asking **METAFONT** nicely). This PXL file could also be used by the higher resolution device's driver for any TEX job that asked for `\font B=FOO8 at 4pt`; or one that used `\font C=FOO8`, but then got TEXed with magnification 500; or one that used `\font C=FOO8`, but then got spooled with magnification 500. Note that we are assuming that the font FOO is like the CM family in that it does not scale proportionately in different point sizes—we are only talking about the 8-point face. If it turns out that 8-point FOO magnified by 1.5 is exactly the same as 12-point FOO, then we can also use FOO12.1000PXL in place of FOO8.1500PXL, and so forth. For fonts that scale proportionately like this, a point-size should not be included as part of the font name, and FOO.1000PXL is by convention the 10-point size of FOO for a 200-dot-per-inch machine.

Now for an explanation of where the bits go. A PXL file is considered to be a series of 32-bit words (on 36-bit machines, the four low-order bits of each word are always zero). In the discussion below, "left half word" means the highest-order 16 bits in a word, and "right half word" means the 16 next-highest-order bits in the word (which are exactly the lowest-order 16 bits on 32-bit machines).

Both the first and last word of a PXL file contain the PXL ID, which is currently equal to 1001 (decimal). The second-to-last word is a pointer to the first word of the Font Directory. (All pointers are relative to the first word in the file, which is word zero.)

The general layout of a PXL file looks like this:

PXL ID	[1 word long – First word of the PXL file]
RASTER INFO	[many words long – begins at second word]
FONT DIRECTORY	[512 words – 517th-to-last through 6th-to-last word]
CHECKSUM	[1 word – fifth-to-last word]
MAGNIFICATION	[1 word – fourth-to-last word]
DESIGNSIZE	[1 word – third-to-last word]
DIRECTORY POINTER	[1 word – second-to-last word]
PXL ID	[1 word – Last word of the PXL file]