```
                        hor_line_length (q);
                        H := H - q
                    END;
                136:    BEGIN
                        overprinting := true
                    END
                137:    BEGIN  (* font *)
                    END;
                138:    BEGIN
                        read_4_bytes (w);
                        H := H + w
                    END;
                        . . .
            END
        END
    ELSE    IF ((154 <= b) AND (b <= 217))
            THEN change_font (f, chr(b-90))
END
END.
```

The header file TEXDIA.H

```
CONST
    hor_spacing = 10;       (* standard HMI *)
    vert_spacing = 8;       (* standard VMI *)
    stack_size = 125;
    mem_size = 3000;
    max_font_no = 5;

TYPE
    byte = 0..255;
    half_word = 0..65535;
    oneoftwo = 1..2;
    oneoffour = 1..4;
    halves2 =   PACKED RECORD
                lhword: half_word;
                CASE oneoftwo OF
                    1:(rhword: half_word);
                    2:(byte2: byte; byte3: byte)
            END;
    bytes4 =    PACKED RECORD
                byte0: byte;
                byte1: byte;
                CASE oneoftwo OF
                    1:(rhword: half_word);
                    2:(byte2: byte; byte3: byte)
            END;
    memoryword =    PACKED RECORD
                CASE oneoffour OF
                    1:(pts: real);
                    2:(int: integer);
                    3:(twohalves: halves2);
                    4:(fourbytes: bytes4)
            END;
    pts = real;
    stack_range = 0..stack_size;
    mem_range = 0..mem_size;
    font_range = 1..max_font_no;
    fnt_store = PACKED FILE OF byte;
    font_type = (rm, it, sy, ex, tt);
    fontfile = FILE OF memoryword;
```

Site Reports

NEWS FROM THE HOME FRONT

David Fuchs
Stanford University

Here's what's going on TeX-wise at the
Department at Stanford. Professor Knuth h
working version of the UNDOC macro proc
written in its own language (DOC). UNDOC
piles itself into a Pascal program, thus UN
is now available in Pascal. DOC is being use
the source language for new versions of TEX
and TeX82. All three programs (both DOC
Pascal sources) are expected to be available for
ing to new machines in early 1982. TeX82
complete rewrite of TeX based on the exper
gained from Ignacio Zabala's translation of
TeX. Portability has been improved by removir
floating point operations. Another sticky portal
problem with the current Pascal TeX is initia
tion. Recall that installing a new TeX involves
ning the program TEXPRE, which makes a
file (called TEXINI.TBL) that represents the i
state of TeX's data structures (about 36K wor
size). On TOPS20, we then run TeX, which
in TEXINI.TBL, at which point we interrupt
process and save the current core image. When
users ask for "TEX", they get a copy of this
image, which continues execution from where w
terrupted the first TeX run. Thus, our users
saved the not-insignificant overhead of data s
ture initialization. The resulting core image is
smaller and faster than if the initialization f
tions of TEXPRE were to be incorporated into
Unfortunately, we have found that the facili
"saving an interrupted job's core image for later
tinuation" is not available in many environm
including VAX VMS, UNIX, and IBM timesh
systems. Consequently, TeX users outside o
DEC 36-bit world have TeX re-read TEXINI.
each time it is run, which is a significant
handicap. To help rectify the situation, TE
data structures will change to require less initia
tion. We also plan to make a program avai
that can read TEXINI.TBL and produce Pa
language initialization code to be inserted int
TeX Pascal source before compiling. Unfortuna
variable initialization is not standard Pasca
there must be different versions of this progra
the Hedrick compiler, Pascal/VS, VMS Pascal,

Of course, the option of reading in a significantly smaller TEXINI.TBL each time TEX is run will still be a possibility, and will only use standard features of Pascal.

Scott Kim's INVERSIONS has appeared in bookstores. The text was typeset with TEX, using fonts leased from the Alphatype Corporation.

A few people in our department spent some of their summers porting Pascal TEX to a number of new machines. Joe Weening has a version working on the Cray, with output to a Versatec. Jeff Rosenschein brought up TEX in Israel on an IBM processor running VM/CMS, with output on a Versatec. I spent some time writing Pascal drivers for the Autologic APS-5 and Mergenthaler Linotron 202. In fact, it's a single program that compiles under either IBM's Pascal/VS or Hedrick's Pascal for DECSystem 10/20, and will convert DVI files to APS-5 DCRTU Input Command Language format, or Linotron CORA-V format, or Linotron Binary Byte format, depending on the setting of some compile time switches. The first version of this program is up and running at two IBM and one DEC20 sites. Each of these installations is running its typesetter with the native fonts (auxiliary programs serve to convert font width information as supplied by Autologic and Mergenthaler into TEX-compatible TFM format). This restricts the number of TEX's features that can be used, however, since information about 'height' and 'depth' of characters is not provided by either company, and can only be guessed at heuristically, and changed manually if the need arises. Also, much of math mode is crippled, since the TEX math fonts are not available. There is still some hope for the future, though. There has been some interest expressed by individuals both at Autologic in LA and Linotype-Paul in England in Knuth's Computer Modern fonts. Similarly, at the instigation of the folks in Wisconsin, Compugraphic seems interested in the possibility of providing the CM fonts to 8600 users.

One other thing I worked on over the summer was PXL files. A PXL file is the raster description of a font at a given size and resolution (such as CMR10 at 200 dots per inch). The documentation for PXL format is contained elsewhere in this issue (p. 8). Our spooling software for Varians/Versatecs has been updated to use PXLs. PXL files supersede the older VNT format and have the following advantages:

(1) They include sufficient information to be used with TEX's magnification features.

(2) I have added a PXL mode to **METAFONT**, so that PXL files can be made directly for **METAFONT** fonts.

(3) The internal layout of PXL files has been improved to allow for more efficient operation of spoolers that must deal with them.

(4) The layout is also such that PXL files can be written sequentially from **METAFONT**, which will aid in writing a transportable version of **METAFONT**.

* * * * * * * * * * *

TEX UNDER THE NORTH STAR

Michael J. Frisch
University of Minnesota

I have made some progress on the CDC Cyber TEX at the University of Minnesota since the last TUGboat, but at this writing (October 1) it is not ready for distribution. There are a few known bugs in my version of TEX and a possibility that more bugs will show up during further testing. (I believe that most of the bugs are in my implementation rather than in the original Stanford version.) As a result, I can't give an estimated release date.

I have a working—though probably not debugged—device driver program for our Varian plotter. The device driver is written in machine-dependent FORTRAN since I can write and debug code much faster in FORTRAN. The driver translates the DVI file into line segments. Each segment has the same baseline and the same size font. At the end of each page, the driver sorts the segments by vertical position and then outputs scan lines to the plotter one at a time. This allows multiple column text input to be processed.

The driver is small enough to run in interactive mode though my version of TEX will not. Luis Trabb-Pardo pointed out that if I can get TEX to run in a large memory area, then I could make a large driver keep an entire page of plotter bits in memory and vastly simplify the driver. For lack of time, I haven't tried this idea. The driver I wrote works and could be improved or rewritten later. My thanks to David Fuchs for his Pascal DVI file printing program which explained a lot of mysteries to me about the file.

My version of TEX now accepts almost all of the standard BASIC file. This is a great improvement over the TEX version I mentioned in the last TUGboat. My TEX doesn't read all the font information files correctly, because some were not properly converted from PDP-10 36-bit floating point to CDC 60-bit. Despite this, and with some temporary changes, I have made a very short TEX input file run and have produced a plot so I have hopes that I will make more progress.

Though my version of TEX is not totally functional, the University of Aarhus in Denmark has one they built that works quite well. Erik Bertelson and I recently made an oral agreement that Minnesota will distribute their version in the U.S. With code modifications and decreased memory size, their version runs in about 38K words. They have printed a couple of books using it so it is quite well debugged. They have a Pascal version of UNDOC which has made it a lot easier to change TEX. When I receive a copy of their TEX, I will install it and then decide on distribution details.

* * * * * * * * * * *

A TUGboat TOUR: EXCERPTS FROM THE TEXNICIAN'S LOG

Barbara Beeton
American Mathematical Society

The morning mail is very exciting around the TUGboat deadline. In it appear manuscripts and mag tapes from all over, bearing who knows what news, but certainly containing something unexpected and interesting. Putting together an issue is also fun; the readers get to see only the printed pages, but I get to grub around in the mud that seeps into all the cracks.

During this tour I'll be referring to various articles that have appeared in TUGboat, so you might want to get out your back issues. In particular, I'll be making a few changes to the previous TUGboat guidebook: *How to Prepare a File for Publication in TUGboat*, vol. 2, no. 1, pp. 53–54.

Although this *How to* ... guide states that `basic.tex` and all the features of $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX are available, as well as some formatting macros especially for TUGboat, some authors build their own macro set from scratch, including formatting details. This may be required by local font and device limitations (see the article by John Sauter, p. 34, and the excerpt from his output, p. 13), or the author may simply have his own idea (different from mine) of how his article should look in print.

Whenever possible, I try to accommodate the author's ideas, so long as they do no violence to the articles which will follow when the issue is put together. Often this can be accomplished very simply: quarantine an article from the rest of the issue by putting `{` braces `}` at the beginning and end. (This was necessary for Brendan McKay's ar-
ticle in vol. 2, no. 2, pp. 46–49; any attempt to r
the format would have taken too long, and w
certainly have resulted in unfortunate errors.)

Another technique is to lay out the pages in
a way that the "singular" item begins on a new p
and deal with it separately, adjusting page num
as required. This is obviously necessary for mat
received as camera copy, but another reason w
be that, between the standard TUGboat set and
author's, there are too many control sequence
fit in the hashtable. (Mike Plass' article on sy
chart macros, p. 39, is an example of this conditi

While I'm on the subject of the hashtab
should note that, here at the Math Society, w
found it necessary to change several of the
values that show up in the error message

```
! TEX capacity exceeded
```

(TEX manual, p. 144). The most important
the following: `hashsize=1009`, `memsize=32`
`varsize=11500`. (One or more of these changes
by now have been made in the "official" SAIL
sion at Stanford.)

In the SAIL implementation of TEX, `memsiz`
restricted to a maximum of 2^{15}; this will be li
in Pascal TEX, but we won't be using that
til the "definitive" version is published. `vars`
is effectively a subset of `memsize`, and there
the actual capacity of `memsize` is really only al
21,200. (All these values represent 36-bit word
our DECSystem 2060.) The original value of v
`size` was 17,000, but much of our work invo
large pages of small type and, being unable to
crease the absolute value of `memsize` to avoid
ceeding it, we had to increase its capacity by re
ing `varsize`.

`hashsize` can be no larger than 2^{10}, and
Knuth recommended that its assigned value b
prime (it used to be 797); he suggested 1,009, e
to roman `MIX`, of which he is particularly f
since he finished TEXing Volume 2 of *The Ar*
Computer Programming (TEX manual, Appendi
pp. 161 *ff.*). These changes must be made in
SAIL (or Pascal) source, and the modules rec
piled.

To provide more facts on just how much
`hashize`, `varsize` and `memsize` are required
troublesome jobs, we've also created a "diagnos
version of TEX, which reports at the end of e
completed (output) page and at each occurrenc
`\ddt` the current and maximum (so far) demand
`varsize` and `memsize`. It also, if requested, pr
each control sequence name as it is loaded into
hashtable, counting down as it goes from the
tial value of `hashsize` (1,009 less however m

control sequences were preloaded). Some of this reporting facility already existed in the SAIL source code obtained from Stanford, requiring only that a switch, `MSTAT`, be turned on before compilation; we've suggested that a similar facility be included in the "definitive" Pascal TeX.

As TeX comes into being on more different kinds of computers, these computers generate tapes containing input files to be processed somewhere else. TUGboat gets a good sampling. We've successfully read tapes from the following computers: VAX (running under both VMS and UNIX), IBM 370, PDP-11 (TeX was not running on this machine; it was used only to prepare the tape), Univac 1100, and DECSystems 10 and 20. The DECSystem tape utilities, of course, generate tapes that are "native" to our 2060. Finding a common format for exchanging tapes with other machines has been more of a challenge.

The tape format we have settled on works quite well, although it is limited to the standard ASCII character set, one character per tape character (8 bits), which does not accommodate font files in internal form. This format is simply a variation on the old "card-image" format, with 80-character records, 100 records per block (8,000 characters per block). When reading one of these tapes, we assume that no carriage return/line feeds are present, and that all terminal spaces can be stripped from each record. We therefore recommend that input lines not end with the \␣ control sequence, since it cannot be distinguished from \⟨cr⟩ after stripping; in practice, this has occurred only rarely, and was easy to correct, although inconvenient.

Another tape problem is what to do about labels. The utility program we use to read "foreign" tapes isn't very clear about label formats, so the easiest thing to do is omit labels. We have just successfully read a IBM 370-generated tape with "ANSI standard labels" (which are not among those defined in the utility manual, and for which Susan Plass had to try long and hard to find a description, and make several attempts before a good tape was actually written); we finally treated the labels as a separate file, which was discarded once the real file was safely on disc. Suggestion: forget about labels. (They're probably really necessary only for multi-volume tapes, and no single TeX file should be that big anyhow—it would probably be big enough to contain the entire *Encyclopædia Britannica.*)

Finally, there is the matter of identifying what is on the tape. It would be appreciated if every tape were accompanied by a transmittal form (one is bound into every issue of TUGboat), and by a list of what files the tape contains. And, just to be sure,

`%` (file name)

as the first line of each file will make it easier to check the disc copy.

Once files are safely on disc, properly identified, they undergo some editing (as little as possible) to ensure that they conform to TUGboat requirements. Of particular importance for compatibility is the use of a "standard" font set. For TUGboat (in fact, for all the Society's TeX work), our standard set is based on the one used for the "book format" (TeX manual, Appendix E, p. 152) except for `\font ?=cmti10` as in `basic.tex`. Only font codes `G` through `Z` are free to be used for job-specific fonts, so if a TUGboat author has special font requirements he should note them in comment lines at the beginning of his file and if possible identify them by letters `G`–`Z`.

To permit automatic cross-referencing and insertion of page numbers in the Table of Contents, a reference is added just following the title which allows page numbers to be sent to a separate file. An error in the second T-of-C page in vol. 2, no. 2, should give you an idea how this works (when it's done right), but here are the details anyhow. In each file, a line is inserted:

```
\pagexref{filnam}
```

which invokes the definition:

```
\def\pagexref#1{\send9{~def~#1{\curpage}}}
```

At the beginning of a TUGboat run the page number file from the previous run is read in (`~` is `\chcoded` to type `0` for the duration, so that these definitions really do become control sequences, then back to type `12`="other"). Then a new version of the file is opened for output, to record new page numbers should any changes have taken place. Actually, two different file names are used so that the old file is available after the run for comparison to the new version; also, some items don't ever get run through TeX, and their references are added to the page number file manually. When the page numbers converge, and a final scan of Varian copy turns up no obvious blunders, the .DVI file is shipped off to the Alphatype for camera copy.

The main TUGboat header file is used to format the one- and two-column pages. At present, the two-column routine uses the `\save5\page` … `\box5\hfil\page` technique. I would really like to be able to write out each column as a separate page, but the output drivers require that each "sheet" contain the same number of "pages", and I haven't been able to figure out how to output two "pages" for a single-column page. Suggestions are welcome; I'll submit this as a problem for the next issue if no solution has been found by then.

The address list uses a different header file, which does output each \page (i.e. column) separately to the .DVI file, putting the running heads and footers out as part of the last column. The width of each partial page is the distance from the left boundary of column 1 to the right boundary of the current column. This width is recalculated for every \page; \xcol is the column number, \xcolmax is the total number of columns per page, \xcolwd is the number of points each column is wide, and \intercol is the number of points skipped between two adjacent columns:

```
\def \howwide{\setcount8\xcol
     \setcount3 0
     \sowide}
\def \sowide{\advcount8 by -1
     \advcount3 by \xcolwd
     \ifpos8{\advcount3 by \intercol
             \sowide}
     \else{\xdef\thiswide{\count3 pt}}}
                 . . .
\output{ . . .
     \howwide
     \if \xcolmax\xcol{(output headers)}
     \vbox to size{
            \hbox to \thiswide{\hfil\page}}
     \if \xcolmax\xcol{(output footers)
               \gdef\xcol{1}}
     \else{(add 1 to \xcol)}}
```

I even use this to calculate the total page width (needed for, e.g., running heads; TEX's arithmetic is more reliable than mine):

```
\def \xcol{\xcolmax}
\howwide
\xdef \pagewd{\thiswide pt}
```

I've used this technique to put together pages of up to 6 columns; if \xcolmax gets much larger, Mike Spivak's \result trick (vol. 2, no. 2, p. 50) has to be used to avoid a nesting level error on \sowide.

The .DVI file now contains \xcolmax pieces for each publishable page, each of which has the same reference point (upper left corner). It merely remains for them to be overlaid by the output device driver, which is assumed to have "pasteup" capability.

Yet another header file is used to prepare the Errata list. You may already have noticed that the current one has been TEXed, unlike those sent out with previous issues. There's a good reason for the delay: all the previous schemes for encoding "typewriter-style input" have been rather cumbersome, and I've been waiting for one that's really easy to use. Mike Spivak has made up one for his AMS-TEX manual (*The Joy of TEX*) that makes input strings look just like ordinary math coding; the only difference is that *...* goes around in-line statements, and

```
**
. . .
**
```

goes around displayed material. This will tually become available for use in the main of TUGboat, and Mike has promised to publis macro.

I said I'd be changing the instructions for some of the TUGboat control sequences describ the old *How to ...* guide. Here are the change

A new control sequence, \hpar will give you a dented paragraph like this one. Unlike the hanging indented paragraphs listed, it do have to be \ended.

- The control sequence \endhpar turns out unnecessary, since hanging indentation is sient, disappearing at the end of the cu paragraph.
- The line breaks in \textaddr can now be cated by \\; I got tired of misspelling \ and made life easier (inspired laziness mu the greatest source of creativity known to

Actually, there have been lots more changes t header files as I've learned more about TEX most of these changes are entirely transparent t casual TUGboat author. When I'm entirely pl with the package, I'll submit it for publication i Macros column. (It does depend on the AMS macros, and is distributed with them on tape.

In a later issue I'll report on the timing s tics for TUGboat. There have been many que regarding how much computer time is requir run TEX and prepare output on various de Although TUGboat is probably atypical, it' publication that is readily available to all TEX and thus suitable for an example.

So keep those tapes and letters coming—t *your* newsletter, and only your participation keep it afloat.

* * * * * * * * * * *

TEX FOR THE HP3000

Lance Carnes

Amazingly enough, the mighty TEX-in-I system runs on the modest Hewlett-Packard computer.

The TEX-in-Pascal tape was received Stanford in May 1981 and it took approxin two person-months to do the conversion. were no problems of significant magnitude co ing the Pascal sources. As with every conve the sources had to be edited to weed out the standard" features (e.g. OTHERS:) and many of the SYSDEP modules had to be rewritten. The .TFM font files converted nicely from the FIX representation of reals.

By far the most difficult task was shoe-horning TEX into a 16-bit word, 32K address space, non-virtual memory machine. Accessing the 49152 records of MEM (takes 200K 16-bit words) and the other large arrays was accomplished through liberal use of software-implemented virtual memory. A version of the Pascal P4 compiler was modified so that when a large array is referenced, code is generated to bring in chunks of the array from disc storage.

Naturally, a heavy performance penalty is paid with this implementation. Currently it takes two to three minutes to compile a single, simple page of text. Additional optimizations will be implemented before distributing this version, sometime in the next month (November 1981). Anyone with ideas and/or experience optimizing such an implementation is welcome to write to the address below.

This author has agreed to be the site coordinator for the HP3000. If you are interested in obtaining a copy of TEX for the HP3000, please write to the address below. Indicate which model and MPE release you have, and any output device(s) at your site. The initial release is scheduled for December 1, 1981.

Lance Carnes
163 Linden Lane
Mill Valley, California 94941

* * * * * * * * * * *

TEX FOR THE IBM 370

Susan Plass
Stanford Center
for Information Technology

It's finally up and running—Pascal TEX for the IBM 370 running in the MVS batch. We have successfully produced device independent (DVI) output files which have been correctly printed on the Stanford Computer Science Department Dover printer. We do not have output drivers for printing these files from an IBM 370—we hope to write those soon.

What we do have, however, is a version of PTEX which runs on our 3033 under MVS and under ORVYL, Stanford's timesharing system. The following is a brief outline of the changes we have made to PTEX in order to compile under Pascal/VS, to run on an IBM processor under MVS, and to fix a few known CSD version bugs. We have made no changes to TEX specifically to run under ORVYL; that was achieved merely by compiling TEX with an interactive version of Pascal/VS.

1. TEX

a) All integer subsets were changed to PACKED integer subsets.

b) All REALs were changed to SHORTREAL.

c) EXTERN was changed to EXTERNAL throughout.

d) OTHERS: was changed to OTHERWISE throughout.

e) INITPROCEDURE was changed to PROCEDURE INIT and a call to INIT was added as the first executed statement.

f) In DEFINEFONT, the label 0 on the statement

```
LABEL 50, 31, 0
```

is never referenced, causing a warning from Pascal/VS; this label was removed.

g) In the loop from N+1 to 30 within LEXICALORER, the initialization of TRUNCWORD[I]:=0 was added to the existing initialization of HYPHENATIONWORD.

2. TEXPRE

Changes (a) through (e) of TEX above are identical for the TEXPRE module.

f) There are a few lines longer than 72 characters. These were broken up rather than expand the MARGINS option to 80.

g) At the start of XENT the statement SWORD := NULWORD was added.

h) The following were added to INITSUF:

```
VAR I : INTEGER;
FOR I:=0 TO 115
          DO SUFFIX[I].ALPHASET:=[];
```

i) The following were added to INITPREF:

```
VAR I : INTEGER;
FOR I:=0 TO 108
          DO PREFIX[I].ALPHASET:=[];
```

3. SYSDEP

Changes (a) through (d) of TEX and TEXPRE are identical for SYSDEP; there is no INITPROCEDURE, so CONSTANT VALUE variables were changed to STATIC data.

e) The TYPE CHAR9 was changed to STRING(9).

f) All files were changed to type TEXT.

g) ASCII translation in and out and to and from EBCDIC was added via the chrX and ordX arrays.

h) Terminal I/O was modified so that files would not be RESET for each reference (necessary for a BATCH environment).

i) RESET and REWRITE statements were modified for PASCAL/VS.

j) The routine INTOUT was replaced with an algorithm more suitable for HEXadecimal arithmetic.

k) Octal computations and constants were changed to `HEX`.

l) A `POSTAMBLE` file called `PST` is created in `CLOSEOUT` to make this information easier to access using standard IBM access methods.

m) The `DVI` file is always named `DVI` so that the same DDname may be used in batch JCL.

n) Code relating to `DIRECTORY` references was commented out; it is not needed in PASCAL/VS.

o) The `FONT` file directory was changed to a VS partitioned data set.

A tape containing the source for our version of PTEX, TEX/370, will be available soon. For details on obtaining a copy write to:

Susan Plass
C.I.T. Systems
Polya Hall, Room 203
Stanford University
Stanford, CA 94305

When available, TEX/370 will be supplied on an IBM standard-labeled 1600 bpi tape—please don't send us a tape.

We plan to fix bugs, implement new releases, and incorporate comments and criticisms into TEX/370 and will publish those changes periodically to users who have ordered TEX/370. No promises are made or implied about responses outside of such newletters, but we do welcome feedback and will try to act on it. We also plan to implement output drivers for several output devices attached to our 3033/3081. These will be announced as they are implemented.

* * * * * * * * * * *

TEX IN ISRAEL

Jeffrey S. Rosenschein
Stanford University

Over the past summer, TEX was brought up at the Weizmann Institute of Science in Rehovot, Israel. It is running there on an IBM 4341 under CMS, with the Imperial College Pascal P4 compiler, and producing output on a Versatec 1200-A.

Many of the issues addressed in this implementation of TEX have been treated (repeatedly) in previous issues of TUGboat, with regard to other machines and other versions of Pascal; nevertheless, for the sake of completeness, I will briefly outline the major points of interest. It should be noted, however, that this was an old version of TEX, received from Stanford in January 1981.

Editor's note: David Fuchs comments, "It is unfortunate that Jeff had to use an old version of TEX-Pascal. Most of the following points had already
been cleared up by Ignacio Zabala and Eagle
while Eagle was working on compiling TEX
Pascal/VS. Because Pascal/VS packs records a
an OTHERS construct, it seems more suitable
current TEX than the P4 compiler. TEX82 i
rently planned to require OTHERS, but even th
rent TEX makes no assumptions regarding pa
The routine READFONTINFO is system-depe
and the documentation in TEX82 will be more e
about exactly which bits go where.

(1) In Imperial College Pascal, there is no d
case for CASE statements; instead, a
THEN-ELSE construction was used to pe
its function.

(2) CASE statement selector variables being
range caused a Pascal crash (this is not th
in Stanford's Pascal). An IF-THEN con
tion made sure CASE statements were ac
only when the selector variable was in ra

(3) Labels that were declared but not used h
be removed.

(4) The INITPROCEDURE construct does n
ist in Imperial College Pascal; instead, a
dure called INITIALIZE was introduced i
place.

(5) Overly large procedures had to be split f
compilation to succeed. In the TEXPRE
module, these procedures were INITIALI
INITMATHCODES, INITFONTCODES,
INITSUF and INITPREF. In the TEX
module, the procedures JUSTIFICATION
MLISTTOHLIST had to be split.

(6) Imperial College Pascal does not allow a
ment between variables of differently
(though identically defined) types.
the TYPE declarations of PCKDHYPH
PCKDCONSPAIR and TBLREADOUT
in the TEX and TEXPRE modules
changed, respectively, to declarations of
PACKEDHYPHENBITS,
PACKEDCONSONANTPAIRENTRY a
TABLEREADOUTTYPE so as to be
patible with the corresponding SYSDEP
rations.

(7) The name INPUTFILE was used in TEX
as a procedure name and as an identifier
enumerated type. To allow compilatio
identifier name was changed.

(8) FILES OF ASCII had to be changed to
OF CHAR.

(9) The ORD and CHR functions in Im
College Pascal map to and from the EB
character encoding scheme. This con
with TEX's expectations of an internal

encoding of all characters. Two translation arrays were utilized to convert characters to and from ASCII, thus satisfying TEX's needs.

(10) PRINTOCTAL was altered so as to work on a 32 bit machine.

(11) The procedure CONNECT was used to link internal Pascal file names to real-world files, replacing TEX's multiple parameter RESET and REWRITE procedures.

(12) All code that looked for an "end-of-line" character (usually a carriage return) was changed to utilize the EOLN function. This was necessary due to the record-oriented structure of IBM files. Likewise, instead of writing a carriage return onto a file to signify an end-of-line, the procedure WRITELN was used to finish off a record and transfer it to a file.

(13) Imperial College does not pack records as expected in the SYSDEP module code. To overcome this, the PTEXINI.TBL file was changed from a file of INTEGER to a file of MEMORYWORD, extra routines were introduced to build correct font data structures, and bytes were explicitly packed into integers for the DVI file.

(14) The SCANNUMBER routine in TEX and TEXPRE makes no check for overflow as it reads in a number from the user's TEX input file. If the hapless user includes too large a number, Pascal crashes, and there is no way of knowing that the overflow was not internal to TEX (i.e. some previously undiscovered bug). A check was introduced in the SCANNUMBER routine so that if overflow is about to occur, the ERROR procedure is called. This gives a standard dump of the buffer and allows the user a graceful recovery.

Due mainly to Imperial College Pascal's lack of record packing facilities (which causes MEMORYWORDs to each occupy 4 words of memory), it is necessary to have a runtime storage allocation of approximately 2000K to run TEX. Production of raster files for the Versatec takes about 700K. As of this writing, the system has been put through a series of relatively small tests, and (so far) seems to be working without difficulty.

As with all those who have brought TEX up at various installations, I have several suggestions for changes to the code; these are intended solely to aid portability, and increases in portability may, of course, be purchased at a cost to some other important consideration (such as efficiency). Nevertheless, if TEX-in-Pascal is really intended to be a portable program, there ought to be more consideration of standard Pascal features, and sensitivity to differing machines. The two main suggestions are:

(1) The OTHERS construct should be removed once and for all; it has no place in code that is advertised as portable, especially not in the actual TEXPRE and TEX modules (as opposed to the SYSDEP module). Its removal was time consuming, and it should not have to be carried out repeatedly at various installations. At times, finding the correct values to use in the IF-THEN construct was non-trivial due to nesting of CASE statements and the appearance of labels within procedures. In addition, there was an abnormally high chance of an error creeping into the code; such an error would be extremely difficult to track down.

(2) Assumptions about packing should be removed from the code; experience has shown that this restructuring is quite feasible. Although this will result in slight degradation of TEX's efficiency at some installations, it will cause TEX itself to be implemented with much greater ease. This is especially crucial in the READFONTINFO routine, where insufficiently explained assumptions about packing led, initially, to a serious implementation error. To help increase efficiency at those installations with "correctly" packing compilers, helpful hints on how to convert certain code should be included with the documentation; the default, however, ought to be code that will run even in a non-packing environment.

* * * * * * * * * * *

TEX AT THE UNIVERSITY OF MICHIGAN SUMMARY OF PROGRESS

David Rodgers

The University of Michigan Computing Center has installed TEX on an IBM 370/148 running under VM(CMS). Work continues on converting the system-dependent module to run under the MTS operating system on an Amdahl machine. The installation process would have gone unhindered, except for our inexperience with Pascal, the VM/CMS operating system, and the system editor. The entire installation process required about three weeks of full-time effort (spread over six weeks) and probably could have been done in half the time by an experienced Pascal/VM/CMS programmer.

A device driver has been installed for a Linotron 202 phototypesetter in the Ann Arbor area and we are evaluating options for supporting other proof-

and final-quality output devices available in the University community or through local vendors.

We are especially interested in exploring the feasibility of using intelligent terminals (e.g., ONTEL) as a source of local intelligence for driving printing devices. If anyone has had any experience with this sort of output configuration or wants to express a prejucide, we would welcome an opportunity to discuss them. Please write

David L. Rodgers
University of Michigan
Computing Center
1075 Beal Avenue
Ann Arbor, MI 48109

* * * * * * * * * * *

VAX ON UNIX

Bob Morris
UMASS/Boston

The Computer Science Department at Cal Tech has succeeded in running TEX compiled under the Berkeley VAX UNIX Pascal Compiler. By the time you read this, I hope we will have CIT's work at test sites and be working toward a clean public distribution through one or another traditional UNIX distribution arrangement. The work at CIT was done by Calvin Jackson, whose report I have included below, slightly edited. Following Calvin's report is one on device driver work at Brown University.

* * * * * * * * * * *

TEX AT CALTECH

Calvin Jackson

We have PTEX running on the VAX/780 at the Computer Science Department of CalTech. The operating system is Berkeley UNIX Version 4.0. We have compiled with Version 4.1 and corrected a minor problem—an octal constant > maxint.

We acquired a version of PTEX via the ARPANET in May 1981 after the TEX workshop at Stanford. In July we had DVI output consistent with Stanford test cases. A new version of TEX was retrieved from Stanford in July, and approximately 6 hours was required to make the local modifications and produce acceptable test results.

The Department has a DEC-20 and a VAX/780. Output devices are a Trilog C-100 (100bpi, VAX), XGP (200bpi, DEC-20), Applicon plotter (125bpi, tape), and various HP plotters. Some specialized graphics terminals are also available.

Our experiences with PTEX are similiar to t reported in previous issues of TUGboat. We a with many published suggestions regarding b tutorial information about SYSDEP concepts. found that most of our problems were due to la experience with Pascal and machine dependenci Pascal data representation. The DOC listings essential; we are impressed with their quality completeness.

Compilation Problems, Source Configuration

After reviewing the DOC listing and some samples in the UNDOC version we performed s tests to determine how some potential problem a are handled by the Berkeley compiler. After analysis we made some basic decisions. A prepro sor would be developed to perform some des source transformations, separate compilation u were desired, and the SYSDEP program woul divided into smaller compilation units. A sin "lex" program was prepared to do the source tr formations; it is referred to as "pedit" in the lowing discussion. The decision to use a prepro sor was strongly influenced by the regularity the UNDOC version; if some other source f is used the preprocessor might not be so sim The separate compilation-unit decision was base compilation time and the expectation that we w be doing a lot of compilations of areas of SYSD

Following are the observations and the appro adopted to cope.

The type allocation model used by Berkeley is following:

char: 8bits. This type will perform as expecte long as the value range used is 0..127.

real: 64bits.

integer: 8, 16, or 32bits. Integers are signed stated in the Report). The signed attri means that a subrange of 0..255 require bits.

Variant records are optimal within this definitio

This model allocates 64bits for the MEMORYWORD. Note that 64bits would be quired if reals were shorter. Storage economy quires short reals and signed subrange types (or treatment of certain integers as unsigned).

The DVI file is byte oriented and is defined as subrange type 0..255. We changed this definitio be —128..127. The DVI output routine and o routines that interface to a DVI file convert va > 127 to value —256 on output and convert va < 0 to value +256 on input. TFM files are han on a similar basis.

In case-statements, if none of the case-const is equal to the case-index, a runtime error is s times reported. My experience is that the results are unpredictable.

PTEX uses a compiler specific feature for default (OTHERS) case-constants. We decided to use the following construct.

```
  if expression in set-constructor then
    case case-index of
    ...
    end else begin
    { ...  OTHERS code follows}
    ...
    end
```

This process is performed semi-automatically by **pedit**. A program scans the source for case-statements, inserts the skeleton if statements, and generates the set-constructor on a auxiliary listing. We then manually edit in the set-constructor, the program also provides a line number directory of all case-statements.

We did some after the fact timing studies and determined that the preferred transformation is

```
        CASEARG:=case-index;
          if not (CASEINDEX in
              [LOWERLIMIT,UPPERLIMIT])
          then CASEARG:=LOWERLIMIT-1;
          case CASEARG of
          ...
          LOWERLIMIT-1,otherslist:
          ...   {OTHERS code}
          end
```

This provides acceptable efficiency, does not make an exception of the case-index being a function call, may require a few extra editing keystrokes, and is compatible with the program that scans and alters the source.

One procedure (VARSYMBOL) contained some inaccessible code due to the omission of a case-constant. The instance was reported by the compiler. We edited in the appropriate case constant. This condition did not exist in the second set of sources we retrieved (July 1981).

Equivalence of type requires that each variable be defined by the same type declaration. This was significant because of our decision to use the separate compilation feature. We manually reviewed the programs for common type declarations and placed them on an "include" file. Our first copy of PTEX (May 1981) contained instances of the same types with different spellings. A subsequent copy (July 1981) had resolved these problems. It was interesting that we had guessed wrong on the eventual spellings. This process was not very time consuming, there are surprisingly few type definitions and the lexical order of the program simplifies the task.

The empty-statement is incorrectly handled in the following construct:

```
if boolean-expression then empty-statement
else statement
```

A diagnostic is provoked. A "begin end" or system procedure "null" in place of the empty-statement will satisfy the compiler. This task was assigned to the program "**pedit**".

Labels that occur in the label-declaration-part must occur in the statement-part. Labels that did not occur in the statement-part were edited out of the statement-declaration-part. The diagnostics provided by the compiler were the cues.

The standard compilation mode treats upper-case as different from lower-case. In the standard mode keywords are lower-case. A compile option permits this treatment to be suppressed; the option specifies that warnings should be produced for non-standard Pascal. However, we could not use the separate compilation capability if the "standard" Pascal option was selected. We assigned the task of converting keywords and standard identifiers to **pedit**.

Separate compilation units are supported by the compiler. The compile system rigidly enforces the concept that the fragmented program be equivalent to its composite model. This enforcement is primarily at the compilation and linkage stages.

We manually reviewed the compilation text and constructed the following compilation units. The choice was primarily based on the DOC version, experience has suggested better fragmentation.

texpre.p: Vanilla TEXPRE.PAS except for the deletion of common subroutines and types.

tex.p: Vanilla TEX.pas except for the deletion of common subroutines and types.

sysdep.p: SYSDEP.PAS (DOC) Section 16, 64 – max, min and output routines.

basicio.p: SYSDEP.PAS Section 23 – Basic I/O procedures.

string.p: SYSDEP.PAS Section 17 – The String Handler.

filename.p: SYSDEP.PAS Section 51 – Scanning File Names.

infont.p: SYSDEP.PAS Section 57 – Reading Font Information.

fetchdata.p: SYSDEP.PAS Section 71 – Retrieving Data Structures.

storedata.p: SYSDEP.PAS Section 69 – Storing Data Structures.

globsysdep.h: SYSDEP.PAS Global variable and procedure declarations for SYSDEP.

globconst.h: SYSDEP.PAS Global constant-definition-part.

globtype.h: SYSDEP.PAS Global type-definition-part.

globexternal: SYSDEP.PAS Global procedure and function definitions.

The program-declaration must include the file **"output"**.

Program declarations were edited to conform to the Report.

Variables declared to be structured types that require more than 64,536bytes of allocated space provoke a diagnostic. Berkeley was contacted regarding the limitation on the size of structured-types. My understanding is that the limitation should only apply to the use of those variables, not to their allocation. The diagnostic was not present in Version 4.1; we were provided a fix for Version 4.0. The fix required the deletion of the diagnostic and rebuild of the compiler.

PTEX uses the compiler-specific initialization feature INITPROCEDURE not supported by the Berkeley compiler. INITPROCEDURE was changed to a proper procedure and a call placed in the statement-part of the program.

Goto-statements in procedures ERROR and QUIT provoked a diagnostic during the assembly phase. The statements were sufficiently displaced from their target that the preferred instruction could not be used (its displacement field is too short). An assembly option is available that requests the longer jump. We decided to move procedure QUIT adjacent to the program-statement-part and change the goto-statement in ERROR to an invocation of QUIT.

Support Tools

Stanford and Berkeley were extremely helpful when support was requested. We particularly appreciate the efforts of I. Zabala (Stanford) and P. Kessler (Berkeley).

We used a screen editor based on EMACS and many of the standard UNIX tools. A "lex" program was invaluable for performing simple source analysis and transformation. The Stanford programs DVITYP, PLTOTF, and TFTOPL were also invaluable for understanding the interfaces and as roots for other programs. Following is a list of auxiliary programs and their function.

DVITYP: Stanford program that provides a formatted dump of a DVI file. Converted to VAX without the random access feature. Essentially no program changes required.

DVIDEC: Stanford program that provides a raw dump of a DVI file. No program changes required.

tfmdec: Program based on TFTOPL but lacking robustness and elegance of output. Produc raw dump of a TFM. Operates on the DE and VAX; primarily used to prepare TFM for transfer to the VAX.

tfmbin: Program based on PLTOTF but lacking robustness. Produces a TFM file from the put produced by tfmdec.

tfmprt: Program based on TFTOPL but lacking robustness and elegance of output. Produc satisfactory dump of a TFM file.

textoucb: Converts a VNT file to the Berk "vfont" format.

ucbtfm: Generates a partial TFM file fron Berkeley "vfont" file.

dvitriplot: Converts a DVI file to the format requ for plot mode printing on the Trilog prin Font files are in the Berkeley "vfont" forma page of text is set and output to the stand printer spooler (raw mode option).

dvitrilog: Converts a DVI file to the format requ for standard printing on the Trilog. The used must be a typewriter style defined for Trilog. This program is unsatisfactory.

pedit: Lex program that scans a .PAS source and cards blank lines. Inserts code to assist edi of case-statements with OTHERS. Transfo certain empty statements to "begin e Changes keywords and standard identi to lower case. Produces a directory case-statements with a complementary constructor. Produces a directory of proced and function declarations. This program m significant assumptions about the nature of input, i.e., it is not general purpose. If t are significant changes in the format of distributed version, the program would be carded or changed.

Plans

We are thoroughly dissatisfied with what we h for user interface with respect to file specificat Better user control of the name of the output file is desired. This work is local to two of SYSDEP compilation units.

The system response to error conditions is satisfactory. I believe some of this is due to response to runtime language violations. Som due to the way the system is designed. Experie and experimentation will, hopefully, provide s guidance.

The Versatec printer is being interfaced to VAX. Scan conversion will occur on the V Software has to be found or developed for device. *Some from Brown is reported elsewher* *this issue...ram* A similar condition exists for our Applicon plotter. The Applicon is off line, data is transferred via magnetic tape.

The device interfaces are not very efficient or robust. Improvements in these areas will be pursued.

Currently, there are a number of pieces that must be used to perform the typesetting task. They have not been integrated into a reasonable user oriented capability. This has to be corrected.

Various macro packages look interesting and should be available to our users. Acquisition and installation will be pursued. (We do not yet have any TEX users.) The system is still in a pre-release state. Release and user support/encouragement are key items.

We desire to have equivalent facilities on the DEC-20, an effort has been started to host PTEX on that machine.

We plan to retrieve **METAFONT** and attempt to host it on the DEC-20.

* * * * * * * * * * *

THE STATUS OF VAX/TEX AT BROWN

Janet Incerpi

We received a tape from Calvin Jackson containing some TFM files and the TEX source and binary. We were able to get TEX running simply by reading the tape and putting the files in the directories specified. Since we had already installed VNT files and debugged software to take DVI files and send output to the Benson Varian (see TUGboat vol. 2 no. 1 p. 49) we were able to print papers within a few days. (It was necessary to change the programs DVIVER and VERSER to handle the new DVI file format.) We have not recompiled TEX, we are just running the binary we received but don't expect recompiling to be a problem.

* * * * * * * * * * *

VAX/VMS SITE REPORT

M. C. Nichols
Sandia National Laboratories

There are now 40 sites that have received the Oregon Software VAX/VMS version of TEX. The VMS group is presently in a holding pattern while deciding whether to bring up an intermediate version of TEX (the current VMS version is the one which existed at Stanford as of 11/80; it uses tfx fonts rather than tfm ones) or to wait for the most recent Pascal version. Rumor has it that the most recent version will go a long way toward eliminating the 15–20 sec. delay now experienced at the start of every TEX user's run.

Until now, the Versatec was the only output device for TEX on the VMS system, but an accompanying article by Jim Mooney (see p. 13) discusses the birth of a Varian driver written for the VAX in FORTRAN 77 (*FORTRAN STRIKES AGAIN!*). It is hoped that the Varian driver will appear on the VAX/TEX distribution tape soon along with any other drivers etc. that have yet to be publicized. Several people have called expressing interest in spoolers for the Linotron 202 and APS-5 for the VAX.

The current version of TEX for the VAX is still available from Oregon Software for a $50 reproduction charge (see TUGboat vol. 2 no. 2, page 33). Their address is

Oregon Software
2340 SW Canyon Road
Portland, OR 97201

You are to be encouraged to share this tape with other VAX users in your area, but please contact Barbara Beeton (AMS) or myself if you obtain a VMS version in this manner so others in your area will be able to find out that you have TEX up and running.

I would like to solicit news specific to VAX/VMS for possible inclusion in a TEX/VAX/VMS memo to be sent to VAX/VMS users early next year. Especially welcome would be problems, comments, programs, or macros that bring to light or solve any of the difficulties with TEX that are peculiar to the VAX/VMS system.

* * * * * * * * * * *

ENHANCEMENTS TO VAX/VMS TEX AT CALMA

John Blair
Calma

Calma has had TEX running on its VAX for about four months now, using a Versatec V–80 as an output device. While we were happy to have TEX inhouse, there were a few inconveniences to the VAX version of TEX which prompted me to rework some of the VAX-dependent code in Barry Smith's latest release of the VAX Pascal version of TEX.

The biggest problem was the fact that DEC Pascal can only open files for exclusive use, while TEX requires shared, read-only access for the initialization and font files, as well as common source files such as `basic.tex`. Thus if two people tried to run at the same time, the second (and third ...) were informed that the initialization file TEXINI.TBL was locked by another user. The quick and dirty fix for this problem (well, it wasn't really that quick, and all-in-all isn't really that dirty, considering the alternatives) was to write a set of FORTRAN subroutines and functions: FOPEN, FCLOSE, FRESET, FGET, ... to simulate the Pascal I/O calls for all input files. Since DEC FORTRAN supports shared, read-only access, the problem was solved. The only negative factor in the conversion is that initialization now takes 30 seconds elapsed time on an otherwise idle system, rather than the previous 20 seconds. I feel that this time could be shortened dramatically if I could find a free week to re-write the entire initialization procedure (using an assembler routine to block-load the data directly into the program arrays).

Previous to doing this rewrite, we worked around the problem of not being able to run concurrent images of TEX by creating a batch job queue which ran TEX jobs one at a time. To do this we changed TEX to scan the command line and automatically \input the file name found there. For example,

```
TEX foo
```

is equivalent to running TEX and then entering \input foo as the first command. For the batch system to run properly, foo.tex should contain everything from the \input basic to the \end.

A side effect of this organization is that once the TEX job is submitted (a command file handles all of the details), the terminal can immediately be used for other work, rather than having to wait for TEX to finish.

When the background job has finished, a message is broadcast to the user's terminal, and, if TEX exited normally, the .DVI file is automatically sent to the spooler. (Note that any TEX error which prompts the user for terminal input will cause TEX to exit abnormally, since the batch SYS$INPUT will indicate an end-of-file condition to the program).

The names TEXOUT.DVI and ERRORS.TMP which were hard-coded into the source were changed to DVIFILE and ERRFILE so that the command procedure could set the name through

```
assign/user foo.dvi DVIFILE
assign/user foo.err ERRFILE
```

which gives the output files the same name (but different extensions) as the input file.

Now that multiple images of TEX can run concurrently, we no longer need the batch system, but, in fact, most people still use it instead of the interactive version due to the savings in time. We could also use subprocesses to run TEX, but the batch system allows us the flexibility of specifying how m jobs can run concurrently (three or four TEX j running together tend to slow the system dow bit).

We had a problem using the Versatec printer to the fact that normal print jobs and TEX j would both try to write to the device at the sa time. This was solved without writing a symbi manager by running both types of jobs in a sir batch job queue which only let one job at a t have access to the device.

The file name parsing routines contained the ror of assuming that the first period "." encounte was the delimiter between the file name and the tension. This resulted in the inability to use s directories in file name specifications. Fixing t allowed us to organize the [TEX] directory a bit moving all of the fonts to [TEX.FONTS], and us [TEX.INPUT] as the default directory to try if input file is not found in the user's directory.

As an aid to the interactive user, I put in hooks to the VAX help utility. If the user ent Control-H in his input line, the user is passed the help routine. When the user exits help, command line is automatically reinstated up to Control-H, and the user can continue the line. T usefulness of this can be appreciated by anyone w has tried to look up the syntax for a control seque in the TEX manual. Trying to find the right p can be frustrating. Though we have the hooks help, we currently lack the content. If anyone ha machine-readable base of information on TEX s tax and usage, I would certainly appreciate hear from you.

Another problem which I rectified was the f that all fonts which were defined were written i the postamble, whether they were actually u or not. Since basic.tex defines about a do fonts, the spooler spent a very long time load in fonts which were not used. This was easily c rected, and was the only modification which volved TEX.PAS itself, all other changes being c tained in SYSDEP.PAS.

Barry's programs LVSPOOL (which I renam TEXSPOOL) and READDVI were also modi to open "DVIFILE" rather than "TEXOUT.DV TEXSPOOL was changed to refer to the Versatec "LVA0:" rather than "LPA0:" since we had alre assigned LPA0: to our line printer. TEXSPOOL a minor error in that it would crash if you tried print an undefined character from a font. In gen this doesn't really matter, but it makes the prod tion of a font book quite difficult. A trivial fix t care of the problem.

Future work involves speeding up TeX initialization (perhaps I'll just wait for the new Pascal version from Stanford, which supposedly will eliminate the need for much of the initialization), as well as using Calma's interactive graphics systems to dump raster images in the form of a font so that we can incorporate drawings and shaded images directly into the TeX output for our internal documentation needs.

I'm sending off a tape of my modifications to Barry Smith so that he can implement those that he feels are worthwhile into his future releases. Anyone interested should also feel free to contact me directly.

* * * * * * * * * * *

"POOR MAN'S" TeX

John Sauter

Greetings to all of you in TeX-land, from grand wizard to humble worker in the field. I have joined your ranks, I have become a TeX user!

I am running what must surely be characterized as a "poor-man's" TeX system. Although my host machine is a VAX-11/780, my printer is an IDS-460 with the graphics option, which provides me with 84 dots per inch. The printer was tax-deductible because I also use it to help in the preparation of W2 forms on mag tape (but that's another story). After buying the printer I couldn't afford the $50 for Pascal TeX from Barry Smith, so I convinced a department at DEC that they should order it. They did and, through the magic of DECnet, I got it from them.

Putting up TeX was a real nostalgia trip. My first computer job was with the Stanford A. I. Project in the late 1960's, and when I was there I felt that any project could be completed in 6 weeks, and anything short of a major compiler could be done in a single weekend if one were sufficiently dedicated. Since leaving Stanford I have learned that even a trivial change to a text editor can take a year to get into customer's hands.

It was with great joy, therefore, that I spent last weekend making TeX work on my IDS-460. Actually, TeX itself worked with few problems. The struggle was with the post-processor to send the .dvi file to the printer. Even though I had LVSPOOL before me as an example getting everything straightened out wasn't simple. Here are the most critical things I learned, in case somebody should face a similar problem in the future:

First, TeX seems to believe that the printer operates from high bit to low bit, left to right across the page. This is the PDP-10 convention. The PDP-11 convention is to operate from low bit to high bit. The Versatec seems to follow neither of these conventions, operating from low byte to high byte, but within a byte operating from high bit to low bit. There is some code at INSERTLV to scramble a bit string from PDP-10 format to Versatec format, and my statements above about what the Versatec expects are based on it (I have never seen a Versatec manual).

My printer follows the PDP-11 convention. I considered modifying INSERTLV to convert from PDP-10 format bit strings to PDP-11 format, but decided not to, since it would involve reversing the whole string, one bit at a time. Instead I just ORed the bit string into the scan line using the VAX-11 instructions EXTZV and INSV to operate on all 32 bits at once, and arranged to scan the string in reverse order when sending it to the printer. My code for INSERTLV ended up looking like this:

```
.ENTRY INSERTLV,^M<R2>
SUBL3 @HORIZ(AP),#2080,R2
EXTZV R2,#32,@SCAN(AP),R0
BISL @BITS(AP),R0
INSV R0,R2,#32,@SCAN(AP)
RET
```

Since I don't have Metafont I was unable to convert the fonts for an 84 by 84 resolution printer, so I must pretend that my printer has 200 by 200 resolution and a small page. After some experimenting I found that a design size of 2.5 inches wide by 3 inches high works. Using the ten-point fonts I can print posters, so I have made some copies of "The Typographical Error" and am beginning to distribute them among the writers to try to generate interest in TeX. Maybe I can find someone who is willing to invest in a real printer!

I hear someone asking, "I didn't know you could interface an IDS-460 to a VAX-11." The IDS-460 is certainly not supported by DEC as a printer on the VAX-11 so mine is driven through a 300 bit per second terminal line and my APPLE II. I have constructed a protocol to compress the page image for transmission the way a FAX machine would (during a previous weekend) but even so it takes about half an hour to print a moderately complex page.

I made one change to SYSDEP and LVSPOOL for the sake of convenience. I changed all references to [TEX] to be instead TEX$:. I did this because I didn't want the administrative hassle of setting up [TEX] as a top-level directory, so instead I put TeX into a sub-directory and defined the logical name TEX$ to point to it. I recommend this to all users of TEX on VMS, since it makes using TEX more convenient.

I had one problem with TEX: I compiled SYSDEP with /CHECK, and sometimes when loading a font I get a subrange error in MAKEPTS.

I am looking forward to the "final" version of TEX for the VAX-11, which will let me run the fancy macro packages published in TUGboat, and let me build font tables for my low-resolution printer. Keep up the good work, TEX fans, and, from now on, count me in!

John Sauter
801128 Bates Road
Merrimack, NH 03054
603-424-7637
Computer: VAX-11
Output Device: IDS-460
Application: TEX posters

Editor's note: The hard copy of this report, printed on the aforementioned IDS-460, is too appealing not to share with the readership. One page (of $4\frac{1}{2}$) is reprinted as an exhibit on page 13.

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

Fonts

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

FONT UPDATE

Ronald Whitney

Probably the most important development concerning fonts since the last issue of TUGboat is the limited access that TEX users now have to the Autologic and Mergenthaler fonts (see the article by Fuchs, page 21). While this is very good news to those with APS-5's and Linotron 202's and at least a point of leverage to others, progress is still limited with regard to establishing a library of device independent font descriptions. Work on Euler at Stanford and a Cyrillic alphabet at the AMS is continuing, and other sites (including Autologic) have expressed an interest in **METAFONT**. Of course, not only alphabets but also other special fonts need our attention (see the article by Murphy, page 37). We hope that further developments and other ongoing research concerning fonts and **METAFONT** will be reported to the TEX community via TUGboat and TUG meetings.

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

Warnings & Limitations

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

Don't Just \let TEX Hang—\raise or \low

Beware of repeating a \let statement; TEX hang, leaving you no alternative but to kill th leaving you with no error file to help in your nosis.

These four lines cause TEX to hang:

```
\let\oldsize=\size
\def\size{\oldsize}
\let\oldsize=\size
\size
```

These don't:

```
\let\oldsize=\size
\def\size{\oldsize}
\size
```

Nor do these:

```
\let\oldsize=\size
\xdef\size{\oldsize}
\let\oldsize=\size
\size
```

The above results were originally reporte Mike Spivak using the SAIL version of TEX. have been duplicated by Lynne Price usin Pascal version on a VAX. An especially insi variation has just occurred at the Math Socie job \input a header file wherein

```
\let\italcorr=\/
\def\/{\unskip\italcorr}
```

was lurking. A second header file, essentia copy of the first, was also \input, and somew *much* later on in the data, an italic correction was applied. After questioning whether the sy had crashed (not an impossible occurrence), th came to the local wizards for help. Diagnosis t considerable length of time, and would have even longer had we not been familiar with the lem from Mike's experience.

Not that anyone would try it, but \raisein \lowering boxes by negative amounts (in the version, at any rate), has no different effect using the same positive amounts. The foll example says exactly what it's trying to do.

baseline$^{\text{raise 3pt}}$baseline$^{\text{raise –3pt}}$baselin

baseline$_{\text{lower 3pt}}$baseline$_{\text{lower –3pt}}$baselin

Barbara B