OUTPUT DEVICES: A NEW COLUMN

Barry Doherty

To aid the development of output device interfaces, we would like to collect and report information on work in progress on interfaces. It would greatly help both those who are considering installing an output device and those who are working on interfaces to have information on who is working on interfaces for which devices. If you are involved in such a project, please send TUGboat the following information for this column:

- host computer (and operating system);
- output device;
- status of the work;
- your name and phone number and your organization's name and address.

Please send all information to TUGboat. We'll start reporting it with the next issue.

* * * * * * * * * * *

Site Reports

* * * * * * * * * * *

STATUS OF TEX ON THE STANFORD IBM 370/3033 SYSTEMS

Eagle Berns

After obtaining a copy of the IBM PASCAL/VS system, work was begun in conjunction with Luis Trabb-Pardo and Ignacio Zabala, on the implemetation of TEX at Stanford. After a bit of debugging of both the PASCAL/VS and the TEX systems, I was able to get as far as generating a device independent output file from TEX which was sent on tape over to the Computer Science Department. The next day I received back my output from Luis with a note saying "looks good!".

However, the version of TEX which was working was far from being the end product. This was due mainly to the fact that the version of PASCAL being used permitted only 8-byte REAL variables. TEX, on the other hand, required only 4-byte REALs. Since these reals overlayed some integer 4-byte variables in an array which is 32K words long I was essentially using 128K bytes more than necessary. (This number was even larger in total as there were other reals in the system.) The PASCAL/VS group came through, and on a prerelease of the next version of PASCAL/VS, the variable type SHORTREAL existed as a 4-byte real variable. However, due to other changes, my original running version of TEX no longer would run! (Sound familiar?) Along with this came other smaller problems, including the problem of making PASCAL run in an interactive mode in ORVYL (Stanford's timesharing system). My primary objective was to get PASCAL/VS up and running for the Stanford community. Hence, this is where the main burst of activity is taking place. Of course, TEX will be the first major program to be run using interactive PASCAL. Projections about when a working version of TEX will be available run in the neighborhood of the end of February to mid-March (1981), although no fixed commitment is being made at this time.

Some initial looking into the hardware to be used for printing TEX at Stanford has begun, but this is a separate project and nothing substantial can be said at this time.

Those interested in information on the availability of the TEX implementation at C.I.T. can send me a letter (no phone calls—I have a terrible time not losing phone messages) with an address to forward information to, and when firm dates for its availability can be made, a letter will be sent out.

* * * * * * * * * * *

AMS SITE REPORT

Barry Doherty and Barbara Beeton

AMS equipment consists of a DEC 2060 running under TOPS-20, a Varian 9211 printer/plotter, an Alphatype CRS and a Florida Data dot-matrix printer (the latter located at the Math. Reviews office in Ann Arbor). Both the Varian and the Florida Data are driven by Monolithic Z80s configured according to the Stanford specification.

We have been using the SAIL version of TEX for well over a year. Most effort so far has been devoted to converting various administrative publications (*Combined Membership List, Administrative Directory, Catalogue of Publications*) and the issue indexes for *Math. Reviews* and *Current Mathematical Publications* to TEX format. Certain regular departments of the *Notices* have been converted to TEX, and an experiment is underway to typeset material for the *Abstracts*, using *AMS*-TEX. Some of the TEX files are program-generated and maintained, while others have been created and maintained manually.

Varian output is used primarily for proofing, although some camera copy is still generated on it (primarily due to lack of a complete set of fonts for the Alphatype); we expect that most camera copy will soon be produced by the Alphatype. The Florida Data is set up to produce output identical

in size to that from the Varian, using fonts constructed especially for this purpose; a compatible set of fonts is being developed by the Ann Arbor staff (an article on this project will appear in a later issue of TUGboat). The Florida Data, with its ability to print on card stock and preprinted forms, will be used to generate records (via TEX) for Math. Reviews; the Florida Data is also being used as a line printer.

Both the Varian and the Florida Data are currently run by separate spoolers, although we have designed an integrated spooling system and are in the process of implementing it.

We expect to extend our use of TEX to mathematical journals, relying to a great extent on $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX, perhaps beginning some time this year. The timing of the transition depends on the completion of necessary, but currently nonexistent fonts, particularly of mathematical symbols and non-roman alphabets.

We also have a strong interest in **METAFONT**, and have managed to get a version (modified from the Stanford original) running on the DEC20, although it is currently unable to prepare fonts for the Alphatype. So far we have used **METAFONT** mainly to generate fonts for the Florida Data, using Knuth's existing Computer Modern designs with different size parameters; a similar technique has been used to generate some Varian fonts in sizes that were unavailable. We are also developing a special compact ("telephone book") font based on CMSS6, which in turn was adapted from CMR6 and CMSS10.

We are hoping to develop full **METAFONT** capability, so that we can produce in-house any new fonts needed for full journal production.

* * * * * * * * * * *

THE STATUS OF VAX†/TEX AT BROWN

Janet Incerpi

At Brown University we are implementing Pascal TEX using a virtual memory approach without any special purpose hardware to drive the printer/plotter. Pascal TEX has been compiled under Berkeley UNIX‡ on a VAX 11/780. Also, a scan conversion program that sends output to a Benson-Varian 9211 has been completed.

Those who are working on implementing Pascal TEX know that there is more involved than just getting a very large Pascal program to run. The four components necessary for a working system are the Pascal TEX program, font metric files, font files and a scan conversion program. The font metric files contain character measurements (e.g. height, width and depth), ligature and kerning information, and various information used in setting mathematical formulas. TEX uses the font metric files to do the typesetting. TEX outputs a description of the pages being typeset by specifying font and character placement. The font files contain the actual bit rasters (dot-by-dot pictures) for the characters. A scan conversion program is necessary to convert from the ASCII characters in the file output from TEX to the bit raster format (from the font files) needed to print the characters. The scan conversion program uses the output of TEX and the bit rasters from the font files to produce a large bit array which defines the printed page.

Using Virtual Memory for TEX

In standard implementations the TEX output, called a DVI file, is passed through a program, DVIVER, which creates an intermediate file. This is passed to a second program, VERSER, which communicates through a low-speed line with a microprocessor that does the scan conversion to a printer/plotter. Our system does not use a microprocessor; instead it uses the direct memory access (DMA) facility of the printer/plotter interface when doing the scan conversion.

On our system, the DVI file is used as input to a position-sort program that is similar to DVIVER. This generates an intermediate file that contains the necessary packed information, including the character's horizontal (x) position, the change in vertical (y) position, the font to use, the character and the part number. The part number is necessary if the character is too large and therefore must be broken into parts. This intermediate file is constructed a page at a time with each page sorted on the y coordinate before the information is placed in the file. This file is used as input to the scan conversion program which sends the output to the printer/plotter.

The virtual memory facility provided by Berkeley UNIX enables us to handle a large number of font files, while requiring only the pages used to be in main memory. The way we handle the large number of font files is to have an array of font file pointers. Space is allocated for a font file only if that font number is to be used. Although the space is allocated, only those pages of the font file which are referenced using the font file pointer are actually read in. It is no longer necessary to switch font files in and out of memory when the space allocated has been used. This approach does, however, make

†Insiders pronounce the X of VAX as an "x", not as a Greek chi, so that VAX rhymes with the word "hacks".

‡UNIX is a trademark of Bell Laboratories.

the system harder to export to non-virtual memory machines. For example, with the VNT font files, which are used for output to the printer/plotter, the pages containing the character information (i.e., pixel height, width, etc.) are read in, as well as any pages that contain the bit rasters of any characters that are used from the font. The scan conversion program uses the information from the intermediate file and the VNT font files to send bits to the printer/plotter. It uses a wraparound buffer into which it places the bit rasters for characters or vertical and horizontal rules. When the buffer must be emptied (if a character is too tall or a change in y is too large), the scan conversion program writes the appropriate rows to the printer/plotter. To minimize the number of transfers, each write sends the maximum number of rows, taking advantage of the printer/plotter interface's DMA feature.

When the scan conversion program gets the bit rasters from a VNT font file, it is important for it to know that each row of the raster starts on a word boundary. Any bits of the last word in a raster row which are not used are zero. The scan conversion program skips over these trailing zeroed bytes since the buffer is always cleared after its contents are written out.

Pascal and Word-Size Problems

As mentioned earlier, we have compiled Pascal TEX. This was only possible after manually fixing the **case** statements in the TEX source code since our Pascal adheres to the standard of no default (OTHERS) case. The remaining problems that must be fixed to get TEX running are system dependent.

One such problem is the conversion of the 36-bit-word oriented TFX files, the TEX font metric files, to our 32-bit machine. TEX uses these files to do the typesetting. We obtained both the VNT font files and the TFX font metric files from AMS, which runs TEX on a 36-bit machine. The transition onto our 32-bit machine was straightforward for VNT files, since these files only use the low-order 32 bits of each word. Unfortunately TFX files are not as easily adaptable. The TFX file descriptions we have seen use all 36 bits of a word, but in two sources there are references to possible solutions.

The two possible solutions deal with ignoring bits in the TFX file words. The words in TFX files are divided into various fields. Two such fields are the device-width (6 bits) and the ligature (9 bits) fields. One source claims the device-width field is not used, while another says that only 5 bits are necessary for the ligature field. Either way our problem would be solved; unfortunately we do not know whether these are really two solutions to the problem. Once we have our TFX files set up we need only be concerned about TEX's accessing them.

We had difficulty making variant records in Pascal as clean as in standard Pascal TEX, because our current compiler makes packing and unpacking information in file words more difficult. For example, in Pascal TEX, to output a word to the DVI file, four bytes are packed into a word using a variant record which is one of two possible cases. The record is either four scalars (ranging from 0 to 255) or is an integer, so the TEX procedure, *dvi*, uses scalars (bytes) to pack the information and then writes the integer to the DVI file.

However, complications arise because our Pascal implementation does not overlay the scalars and the integer as one might expect. For our compiler, we found a way to correct this but the solution is far from ideal. We expect the same type of revisions to be necessary to access the TFX file information.

The other changes that must be made to Pascal TEX are those dealing with the file system and user interaction. These routines will be rewritten.

The Status of Scan Conversion

Though we haven't yet run Pascal TEX on the VAX (to generate DVI files) we have tested the scan conversion program. (We were able to obtain DVI files from AMS through Sail TEX; in fact, this paper has been printed using our VAX scan conversion program.) The scan conversion program generated the bits for the correct output. Therefore we do have both the position-sort program and the scan conversion program working. Both of these programs are written in C because it allows easy bit manipulation and the use of virtual memory facilities. Later, we will make these programs more transportable.

Future Plans

Our work will be concentrated in three major areas once we have Pascal TEX up and running. First, we plan a detailed performance study to help evaluate our implementation approach for the scan conversion program (i.e., relying on virtual memory as opposed to special purpose hardware). Second, we intend to get Pascal TEX working under different system environments, including VAX/VMS, various IBM systems and implementing scan conversion for experimental graphics and printing devices. Third, we plan to eventually build high-level facilities on top of TEX for document production.

Please direct any questions or suggestions to either Prof. Robert Sedgewick or the author, at The Department of Computer Science, Box 1910, Brown University, Providence, RI 02912.

* * * * * * * * * * *

REPORT FROM THE NORTH STAR
— or —
TEX AT THE UNIVERSITY OF MINNESOTA

T. D. Hodge

In December, 1980, we received a new tape copy of TEX-in-Pascal from Stanford. Working with this tape, we have gotten the TEX preprocessor to read the text file and have partially converted the font files (TFX files) on our CDC Cyber 74.

The main problem we have encountered is the fact that the TFX files contain DEC10 floating point numbers. Ignacio Zabala has promised us that the font files on the next version we receive will all be in integer.

Meanwhile, we will go ahead with the font files we have and try to get the preprocessor and TEX itself to produce workable output.

Initially, we expect to run TEX as a batch program because of its apparent large size. We hope to have more specifics about size in the CDC version and about other factors by the next TEX Users Group meeting in the spring.

Minneapolis, 30 January 1981

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

Editor's note: The following item has been reproduced directly from copy submitted by the author.

TEX IS AVAILABLE FOR UNIVAC 1100 SYSTEMS

Ralph Stromquist
1100 TEX Coordinator

The Academic Computing Center at the University of Wisconsin-Madison (MACC) has implemented TEX on a Univac 1100/82 computer using the UW-Pascal compiler. Testing had been limited by lack of a typesetter until the last two weeks, but output to a line printer and initial typesetter testing indicate that only minor problems remain.

We have tested most of TEX's features successfully: font changing, macro definitions, ligatures, footnotes, fractions, exponents, subscripts, variable size parentheses, matrices, alignments -- to name a few. We do not yet have extension fonts, so have not extensively tested mathematical typesetting. Tests on uncomplicated documents show TEX uses about 0.14 seconds of CPU time per 6.5 by 9 inch page with memory usage of 119,000 words.

Aside from being the first Univac implementation of TEX, our site is also unusual because we have a new Compugraphic 8600 phototypesetter. The 8600 produces high-quality 1300 scan lines per inch output. The model we have can hold about 100 digitized fonts on disk storage. The 8600 is operated offline via a magnetic tape drive. This has the advantage of easy control and backup if reruns are necessary for any reason.

TEX-1100 is available for distribution. Documentation is currently in preliminary form, but should be adequate for installation. Our package includes TEX, a font preprocessor program that converts readable font descriptions into TEX font files, our program for line printer proofing, the 8600 driver program, useful macros, and documentation. Although we don't expect most installations to have an 8600, our driver should still be useful as a model or framework.

To divide part of the cost of TEX-1100 development, distribution and maintenance among the sites using the package, we are charging $500.00 for the package described above. The charge also covers program maintenance by MACC with distribution of enhancements and updates for a period of one year. We will also do limited telephone and written consulting.

The maintenance support may be continued after the first year by paying an annual fee of $400.00. If you have questions concerning TEX-1100, contact

Ralph Stromquist
MACC
1210 W. Dayton St.
Madison, WI 53706
(608) 262-8821

Orders can be placed directly with the MACC Program Librarian at the above address.

This report was typeset by TEX on the Univac 1100/82.

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

REPORT ON THE USE OF TEX AT COMPUTAS A/S, NORWAY NORD 100 COMPUTER 16 bit "big mini"

Helge Totland
Computas A/S
Box 310, N-1322 Høvik, Norway

Hello, all of you far away in the wilderness somewhere. This is a report on two people's (Tore Østensen and Helge Totland) exciting work with TEX. Well, perhaps we are not doing TEX work at all, some would say. Temporarily, we call what

we are working with FTEX (which means Formula part of TEX). We have a phototypesetting system (Nortext), and we are cooperating with the computer firm Norsk Data (they make "Nord" computers, sell Nortext systems and are funding the TEX activity). What we missed most in Nortext is the ability to typeset formulas. While sometimes it might be advantageous to start a project from scratch, we started with the existing typesetting system Nortext (originally intended for newspaper work, the system might drive different phototypesetters). What we decided to do was to grab hold of the math part of TEX, write formulas in TEX code and other text in Nortext.

We started to work on this project last Spring (the "thinking" started November 1979). Our first aim was to have a compiler version of FTEX, with input in TEX math code and output in Nortext code. Later we will integrate this compiler into Nortext, which will be turned into something TEXlike when writing some appropriate "start code" (working like $ or $$ in TEX). The first main user of this system will probably be CERN in Geneva (Switzerland) (Derek Ball). "Det Norske Veritas" (DnV) will also be a user, we expect. COMPUTAS A/S is the data division of DnV, which is an institution concerned with ship classification and other activities (so many technical documents are produced).

Our main source of information has, of course, been Donald Knuth's description of TEX, which we have found very useful. Compared with other systems, the macro feature is especially useful. We had a few beneficial talks with Mike Bennett, while he was staying in Aarhus (Denmark). During our implementation we probably have invented the wheel (again) several times, but then at least we know what's going on. Alternatively, we would have to make rather significant changes in the Pascal code (which is not yet available anyway), if we should convert TEX (or rather extract/convert the math part, change the output part and squeeze this into a 16-bit machine). Our FTEX compiler, by the way, is written in PLANC, which is a rather new systems programming language to Nord (not yet released—this has occasionally caused some new-language troubles).

In order not to stir up the TEX wizards far, far away, we have tried to implement the math mode very close to what we imagine it works like at Stanford. As a side effect, this makes user documentation much less of a problem to us than it would otherwise be. We have a few special debug macros, which are not of any great concern. Our special macros begin normally with *, which is not normally defined as a letter (these macros are not available to users unless using `\chcode`). Example: `\*help` (list macro names on one or more levels), `\*list` (list macro definition), `\*status` (miscellaneous status info). One of the macros we have introduced in our system is `\kdef` (keydef), giving identical results to setting `\chcode` to 13 and defining a one-symbol macro (as described in the TEX errata, TUGboat, October 1980, page 11). This makes definitions similar to `\def` and `\xdef`. It is of special interest to us as the Nortext terminals use 8-bit TET code and then more codes are available than in the 7-bit ASCII code.

We have a few other changes as well:

- cc means cicero (1 cc = 12 dd, ref. dimensions page 40 in TEX manual. In Europe we often use cicero instead of pica (7% bigger (and better)).
- The category code table is extended (Special key, NUL and comment (end-of-line (category 5) is not (!) identical to comment (%))).
- Our internal precision is "only" 1/500 mm (to save space).
- We produce Nortext code instead of DVI-file (DVI-file would be a bit easier to make, but then we would still have one more chain the formulas had to pass before finally being typeset).
- The font files contain left- and right-space in addition (?) to height, depth and width (and we assume the symbol reference point is identical to an assumed focus point in centre of the lens—or similar on CRT's). We use 8 bits on each measure and multiply the measures to different sizes when in use.

The structure of the FTEX compiler is mainly as others have described in TUGboat Vol. 1 No. 1. We have a macro-expansion-section, a formula-tree-builder-section and a formula-tree-traverse-output-section. The first two of these sections work in parallel. We have not segmented the programs or made external intermediate files, as this would only delay the compiling process. When testing we have successfully produced code for typesetting rather complicated formulas (and flags, to test our rule code routine), but there are still features we have not implemented.

Thanks for the TUGboat newsletter. We enjoyed it, and the accompanying TEX errata list will certainly be followed by some actions by us (where practically possible).

Oslo, 8 January 1981