* * * * * * * * * * *

Site Reports

* * * * * * * * * * *

TEX UNDER THE NORTH STAR

Part I

As coordinator for the CDC Cyber implementation of TEX, I have agreed to answer questions from interested Cyber sites about our progress with TEX-in-Pascal and with our implementation of drivers for the output devices to which we have access. When we have succeeded in our implementation, I will make available for distribution, for a charge, a tape with the following information:

1. TEX-in-Pascal source, as modified to compile successfully on a Cyber under the version of Pascal maintained at the University of Minnesota.
2. A file containing the Computer Modern font set.
3. Our device driver(s).
4. Relocatable binary of these files.

I will also send a copy on microfiche of the Pascal compilation and cross reference listing. I will provide sympathy, understanding, and handholding as you put TEX up on your system.

I have also agreed to act as go-between for other matters which may come up from time to time. However, I don't intend to act as a roadblock! Bob Welland, editor of TUGboat, would be delighted to receive articles and letters-to-the-editor directly from you.

My address is in the front of TUGboat; my phone number is (612) 373-4599. This number is always answered and I am reasonably good about returning calls.

In Part II of this report, M. J. Frisch, our TEX implementor, describes our current progress. However, please note that calls should be directed to me, not to him.

Thea D. Hodge

Part II

We recently got a proper device independent (DVI) file output from our Pascal version of TEX. This is a version we received from Stanford in December, 1980. While we are pleased with this success, it is only a beginning.

Several temporary changes had to be made to get TEX to work because our conversion is incomplete. Our font information file had only one font which was converted to 60-bit Cyber words from 36-bit DEC-10 words (containing mixed integer and floating point). The input text was very simple and the output was only a set of numbers, the contents of TEX's DVI file.

We have no device driver programs and our next major task is to write one or more. Potential devices at our site are a Varian 200 dot/inch plotter, a Linotron 202 typesetter, plus a Xerox 9700 page printer at a local service bureau. (The 9700 would be intended for proof copy of text; full math output doesn't seem practical on it at present.)

Further tasks are to finish the TEX conversion by finding out how to remove the temporary changes. We believe these changes are due to incomplete implementation of the system dependent routines. We will also have to convert the remaining font information files to Cyber format.

Our system dependent changes produce a NOS operating system version using 6- and 12-bit ASCII character codes for input and output. If we can find a way to test it, we can build an 8-bits-in-12 ASCII version for NOS/BE sites.

TEX is a very large program, around 98K words. Our site only allows time-sharing for 32K so we run TEX as a batch program. While we realize that TEX is intended to be interactive, our CDC Cyber users will have to live with this problem until (or if) we can find ways to make our version of TEX smaller.

If all goes well, we plan to announce TEX availability for CDC Cyber users at the VIM users group meeting in October. We will need to write installation documentation and we would like to try out our installation procedure at a nearby CDC site.

If there is enough time before October, we will do a conversion of a more recent version of TEX (rather than the December 1980 version). There are a number of attractive features in recent versions which would be helpful. Several recently written Stanford programs are also valuable, particularly one by David Fuchs that prints the contents of a DVI file.

In summary, we really can see some light at the end of the tunnel.

Michael J. Frisch

* * * * * * * * * * *

DECSystem-10/20 IMPLEMENTATION WORKSHOP

Phil Sherrod

An implementation workshop for persons interested in installing TEX on a DECSystem 10 or 20 will be held at Vanderbilt University in Nashville, Tennessee, on September 10 and 11. The course will

discuss the details of connecting TEX output devices to DEC 10s and 20s as well as software issues involved in installing TEX. Conference attendees will have access to a system running TEX. For details, contact

Phil Sherrod
Vanderbilt University
Box 1577, Station B
Nashville, TN 37235
615-322-2951

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

AMS SITE REPORT

Barbara Beeton

Since our last report (TUGboat Vol. 2, No. 1, page 48), an integrated spooling system has been developed and installed to control output to both the Varian and the Florida Data. This spooler will print straight text (ASCII) files, using the rather primitive fonts supplied with the output devices, as well as TEX output. Output defaults to the "local" device, using device-appropriate fonts (Varian in Providence, Florida Data in Ann Arbor), but may be overridden; output page positioning may be adjusted either interactively or from a predefined option (.OPT) file, allowing each column of a multicolumn page to be treated by TEX as a separate page to conserve memory space, then overlaid on output; multiple copies, output of selected pages, deletion (by submitter) of a job from the queue, and other similar features are also supported.

We have acquired the capability of generating fonts for the Alphatype via **METAFONT** (see articles by W. J. LeVeque, page 39, and Ron Whitney, page 40). The Alphatype is being used for limited internal production. (Another composition system is still being used for our major journals, pending completion of necessary work on symbol and math fonts.) The entire two-column portion of this issue of TUGboat and the separate address list have been generated on the Alphatype. A number of articles were sent in on mag tape, from VAX and Univac 1100 systems as well as from DEC 10s and 20s; this experiment has been most successful, and we hope to receive many articles for future issues on tape. Some files have also been received via a phone file-transfer mechanism (not a network).

We have prepared camera copy on the Alphatype for several other organizations that do not have their own high-quality output device; for these jobs, TEX input was provided to us on tape, and the output proved to be virtually identical to the original. Requests for Alphatype jobs will be considered, on a time-available basis; for price and other details, call or write to

Raymond Goucher
American Mathematical Society
P.O. Box 6248
Providence, RI 02940
401-272-9500

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

TEX AT THE 1981 SPRING DECUS U. S. SYMPOSIUM

Patrick Milligan
BNR Inc.

In keeping with a long standing tradition, there was a TEX Birds-of-a-Feather session at the recent DECUS symposium held May 18-21 in Miami. (In case you didn't know, DECUS stands for "Digital Equipment Computer Users Society.") The panel members were:

Patrick Milligan	Coordinator for DEC-20
Phil Sherrod	Coordinator for DEC-10
Barry Doherty	American Mathematical Society
Bob Phillips	Oregon Software

It comes as no great surprise that DEC users are interested in TEX: Don Knuth's book, *TEX and METAFONT: New Directions in Typesetting* is jointly published by the American Mathematical Society and Digital Press. In addition, versions of TEX exist for the DEC-10, DEC-20, and VAX/VMS computers.

Since the DECUS Symposium was held immediately after the TEX Implementors' Workshop at Stanford, emphasis was placed on the installation of TEX. Output samples from various devices (including Versatec, XGP, Canon LBP-10, and Alphatype) were passed around. A free TEX sample (the TUG membership form) was distributed. The content and utility of recent TUGboat issues was described. In short, a sales pitch was made for TEX, TUG, and TUGboat.

Another session on TEX is planned for the next DECUS Symposium to be held December 7-11 in Los Angeles.

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

TEX AT NIH

Rachel Schwab

TEX exists on the DECsystem-10 at the National Institutes of Health in Bethesda, Maryland. At

present it may be used only by the DECsystem-10 staff. Hopefully we will soon have facilities available so that it may be opened up for general usage.

Currently there are four steps involved in running TEX at NIH. First one runs TEX to translate text into a DVI file. Then the DVI file is used as input to a program called DVIPDP. DVIPDP essentially takes DVI records, extracts all the important information, and outputs new records that contain opcodes along with other information. For example: opcode 3 indicates a rule and the information that goes with a rule are height and width. Opcode 2 indicates a "delta y" item—the vertical coordinate is moved by a certain number of pixels, et cetera.

DVIPDP produces a "PDP" file of records like the ones described above. This file is transferred to a PDP 11/70 by means of a high-speed communications link. A Benson/Varian printer plotter is hooked up to this PDP 11. A driver program, TEX/11, was written for the 11 that takes "PDP" records and translates them into scan lines.

TEX at NIH has the ability to take graphs and figures and place them inside a TEX file. TEX itself has not been modified to do this, but the DVIPDP and TEX/11 driver have the ability. Graphs are created by using MLAB or OMNIGRAPH on the DECsystem-10. A program has been written that will convert a plot file into a file of alphanumeric records which in turn may be translated by DVIPDP.

As one can see by the above description, using TEX at NIH is complex. The four steps (TEX, DVIPDP, CLINK (transfer to the 11), and TEX/11) will need to be compressed before TEX can come up in a production environment. This necessarily involves getting our own small computer and printer to hook up to the DECsystem-10 for usage with TEX.

NIH is a community of scientists and researchers who produce technical papers in abundance. The potential for use of TEX at NIH could be very high. However, TEX is not an easy system to learn. We are hoping that the $\mathcal{AMS}$-TEX macros will make using TEX a little easier, and we are looking into the possibility of writing some type of TEX preprocessor.

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

AN IMPLEMENTATION REPORT FOR THE UNIVAC 1100

Bill Kelly
University of Wisconsin-Madison

We have TEX running on the Univac 1100 using University of Wisconsin-Pascal. This is a report on some of the problems we encountered in implementing TEX. The difficulties were of several kinds. A major difficulty is in the differing syntax of external compilation between various Pascal compilers, a feature not defined in the original Pascal definition. Differences in I/O on the host machines and in the ways the compilers handle I/O were another problem. Memory limitation problems were encountered, and these were made worse by the fact that we were altering Pascal code rather than the macro language TEX was written in. The typesetter we are using is not addressable as a raster device, and so we were unable to use standard TEX/**METAFONT** fonts. Instead we had to write a pre-processor to convert descriptions of the typesetter fonts into TEX format. Various errors occurred in using TEX, some of which were traced to errors in the TEX code, others of which related to improperly formatted fonts. The 1100 operating system allows a user to end terminal interaction with a program by typing an end-of-file signal—TEX doesn't consider the existence of this concept. Because we have no inexpensive proofing device we had to write a line printer proofing program which has severe limitations due to the limited range of actions and characters possible on a line printer. We have found the overfull box messages to be unhelpful in correcting justification problems and have replaced them with a more informative message. We have found it very difficult to set type in narrow columns with TEX. Details of all these problems are discussed below.

We should all hope that the project of transporting TEX makes future language designers careful to be complete in defining language standards. Hours and days of work could have been saved if the Pascal definition included external compilation, the default case in the CASE statement, etc.

External compilation in UW-Pascal uses a syntax very different from that used at Stanford. Any shared types, variables, procedures, and functions must be declared in a separate environment module. Only the head of procedures and functions is included there: the name, parameters, and result type comprise the head. Each code module includes a list of procedures and functions which correspond to the heads in the environment. To convert TEX into this format required a lot of hand coding.

Differences in I/O handling required a lot of recoding in the system-dependent module, SYSDEP. Of course, it is to be expected that one would have to recode the system-dependent code, but it could be made clearer what part of the Stanford code refers to PDP-10 peculiarities and what is required by TEX. For instance, TEX expects lines to be ended by carriage returns or line feeds, etc., which is the

internal file format of most PDP computers. In the Univac file architecture, carriage returns and so forth are not preserved in the file. This must be the case with many computer systems, and especially in Pascal compilers where such system-specific things are supposed to be transparent. What we did was to read a line from the Univac file, and when Pascal returned true as the value of the end-of-line function *EOLN*, SYSDEP appended a carriage return. This would have been much easier had the SYSDEP code not also dealt with ignoring line feeds and nulls, or rather had it documented and localized these local quirks. To make it easier for future sites to implement TEX, procedures like SkipTrailingStuff to ignore nulls after a line end, and functions like EndOfLine which would check for PDP-10 line end characters, but note that on non-PDP machines a function like *EOLN* might be used.

The fact that TEX expects input files to be page-oriented is a minor nuisance. Apparently files on the PDP-10 are made up of "pages" delimited by a form feed character. This is not the case on our machine and on many others. This causes minor problems in two respects: one is that if a file containing a form-feed character on our system were read by TEX, it would be interpreted in a manner inconsistent with our operating system. The other is that "page numbers" appear along with line numbers in the error messages. This tends to be very confusing since there is no page number in the file structure, but there are page numbers in the formatted output—so a Univac TEX user would assume the page number referred to the output pages. We circumvented this by changing the error message from "p.0, l.50" to "Line 50". This could be built into TEX by providing a Pascal constant "pageOriented" which the TEX implementor could set to true or false.

Implementing TEX is difficult because the program is written in a macro language which is converted by the UNDOC program into Pascal. Unfortunately the UNDOC program is not presently available in Pascal insofar as we know. The macro code serves well as structured documentation, but it is difficult to refer back and forth between the macro code and the Pascal code. This is because the macro code is divided into sub-sections in a structured heirarchy, which expand into linear Pascal code. Also the use of macros to provide alternate names for variables in the macro language means that a single Pascal statement may look entirely different in the two listings. Another drawback is that expressions involving constants are folded into a single constant. This makes compilation efficient but changing array sizes difficult. We ran into this problem in trying to reduce the size of the "mem" array to meet a memory restriction in our compiler. The ultimate solution is to distribute UNDOC with TEX. But admittedly this work should take second priority to work on TEX itself. A difficulty which may be impossible to surmount with this approach is that the output from UNDOC may still be unsuitable for compilation, depending on the features of the compiler. For us this came about because of the environment feature of UW-Pascal; other sites have had to hand code the default case of the Pascal CASE statement. If this situation occurs, it would be impractical to correct errors in the macro source and all fixes will end up being done in the Pascal code anyway. It is unclear whether it is realistic to expect UNDOC to be able to produce compilable code for all compilers automatically.

On the Univac 1100, a user at a terminal can type `@eof` to force an end-of-file condition in the terminal input to the current program. The program must terminate without further intervention. This possibility is not allowed for in TEX, most likely because this concept of a terminal end-of-file does not exist on many computers. We have circumvented this problem by having the InLnTer procedure in SYSDEP pretend that the user typed `\end` if he indeed types an `@eof`. This is not the ultimate solution, since there are circumstances where TEX will not accept a `\end` command, as for instance in the middle of a paragraph. To help out with this problem (and to make TEX generally more friendly), our basic macro file redefines the `\end` command to be `\par\vfill\end`. This avoids the problem of an unfinished paragraph at the end, but not of an unfinished `\halign`, etc. The suggested solution to this is to have a function TerEof in SYSDEP to detect a terminal end-of-file condition. On detecting this, a clean-up procedure would terminate any current levels of processing, possibly issuing warning messages. Without having examined the code closely, we don't know if the TerEof idea would be the best approach to modifying the existing TEX code, but the general idea would be the same no matter what the implementation. This added code would not impact sites that do not have a terminal eof concept, because the TEX installation documentation would say to have TerEof always return false at these sites.

Using TEX with a non-raster device

Our primary output device is a Compugraphic 8600 typesetter. The 8600 does use digitized data to create its characters, but it will not accept user-defined character shapes. Therefore our TEX font files reflect the shapes of font characters available

from Compugraphic, and our device driver outputs characters and commands to the 8600 rather than bit rasters. We get an excellent quality of output this way, but the approach is not without its difficulties.

One problem was in converting character shapes into TEX-readable format. The data had to be manually typed, and a program was written to transform the raw data into a pair of files, one for TEX and one for the 8600 driver. Until all the bugs were worked out of this program, mysterious TEX errors resulted from improperly formatted font files. For example, parentheses (and other delimiters) of varying size form a linked list in the font information. At first, our font preprocessor did not correctly indicate the end of the linked list. This problem was difficult to diagnose, because it caused a Pascal error; TEX assumed the font information to be correct and so did not issue any warnings. This is as it should be for maximum efficiency of TEX, but it points up the usefulness of a program such as the one described in TUGboat Vol. 2, No. 1 which automatically (and correctly!) converts font files into readable data and vice versa.

Using a device like the 8600 limits the portability of our TEX-produced documents because the fonts do not correspond exactly to **METAFONT** fonts. We have similar fonts, for instance our English Times resembles Computer Modern Roman, but slightly different widths can lead to different paragraphing and pagination, overfull entries in tables, etc. We cannot use **METAFONT** with the 8600 because the manufacturer provides no method of addressing the typesetter as a graphics device, and the cost of having fonts custom-made by Compugraphic is prohibitive.

An interesting problem encountered in using a micro-computer driven typesetter like the 8600 in conjuction with TEX is the lack of cooperation between the two devices. The typesetter itself has a fair degree of intelligence, but it seems impossible to use any of it with TEX because of the need to make the DVI file device-independent: TEX must assume it is dealing with an entirely stupid device! The 8600 does justification, tabulation, automatic accent placement, box-drawing, etc. The commands we pass to the typesetter bypass all these features! The question of cooperation between smart devices is a difficult one, and it seems a shame that the "smarts" of the typesetter can't be better utilized. This is a general question, not one addressed specifically at TEX, that of portability vs. efficiency for a particular device. One approach would be to have a heirarchy of commands, for instance a 'box' command in TEX, which could also be expressed in terms of more primitive commands like HORZRULE and VERTRULE. A user with an intelligent device could configure TEX to output the 'box' command while other configurations could output 4 separate line commands. To maintain transportability, a standard skeleton for a device driver would have to be distributed which could break a box command down into 4 line commands at that stage. The user with the intelligent machine would bypass this procedure. Alternatively, all configurations of a program like TEX could output the box command and distribute the rest of the work to be divided between the device driver program and the device as the programmer sees fit.

A Limitation of TEX

We have found one apparent limitation to TEX: it seems impossible to set text in narrow columns without extensive manual hyphenation. One customer wanted to set 9 point type in 2 inch columns. When we ran TEX, we got many overfull box messages. This leaves you to figure out the cause of the error. It may well be that there is a word that TEX does not know how to hyphenate. We have developed a non-standard message for overfull horizontal boxes that prints the box as a single line on the terminal, representing characters as their corresponding ascii character (or a question mark if the character is non-printing), glue as a space, and hyphenation nodes as dashes. This gives you an idea of whether TEX knows how to hyphenate a word at the beginning or end of the line, and you can insert discretionary hyphens as needed. However, if the word that is giving TEX trouble occurs on the previous or next line, you must consult TEX's hyphenation rules to find whether it knows how to hyphenate the word. Thus I have found myself hyphenating words at random, which is a distinct inconvenience in itself, and often find that there are still overfull boxes.

In addition to poor diagnostics, we had trouble in getting TEX to set the narrow columns at all without overfull boxes. We tried adjusting the `\jpar` parameter to have TEX look at more possibilities, and it still gave overfull boxes. Manual hyphenation did not work. We eventually had to increase the `\spaceskip` (or word spacing) parameter to allow a very large space between words, and create typographically bad output. In some cases even this did not work, and a `\linebreak` command had to be used.

At MACC we charge customers as closely as possible for the resources they use, and we have found TEX to be very expensive in both memory and CPU time. This is not really a surprise to anyone, of

course. But it means that most users cannot afford to run a document through TEX four or five times to find the right discretionary hyphens needed to avoid an overfull box. Nor is this user-friendly.

There may be no easy way to solve this justification problem. It may be an unavoidable side effect of the justification algorithm used. One possibility would be having a "paragraph debugger" that would help to diagnose the problem in a given paragraph: e.g., point out a large word that cannot be hyphenated that is at the root of the problem. This would be a difficult program to write. An alternate hyphenation routine would be another possibility. Allowing TEX to violate its hyphenation rules if necessary to justify a paragraph, meantime issuing a warning, would at least focus on a specific problem, i.e. a questionable hyphenation. rather than leaving the user guessing.

Usefulness of macro sharing

TEX in itself seems to provide a standardized language for typesetting. However, a non-computer programmer will find this language difficult to use, and will not have a ready facility for writing his own macros. TEX users need to do much more macro-sharing as was done in TUGboat Vol. 2, No. 1. Good macros put the typesetting capability in the reach of most users. There are still problems when an error occurs through improper use of a macro. The diagnostics are geared toward one with a knowledge of programming. One aid in this would be to provide a way for a macro to send a message directly to the terminal. This was shown in TUGboat Vol. 2, No. 1 as an extension added by one site; it seems to be a useful enough feature to incorporate into the standard TEX.

* * * * * * * * * * *

AVAILABILITY OF OREGON SOFTWARE IMPLEMENTATION OF TEX FOR THE VAX/VMS

Monte C. Nichols

This VAX/TEX site report consists of an abbreviated version of a memo recently sent to all VAX/TEX users. The information contained here should allow anyone who obtains the VMS version from Oregon Software to implement same on their VAX/VMS system. The only output device supported at this time is a Versatec 1200 A or V80 printer having a DMA interface to the VAX.

A preliminary VAX/VMS implementation of TEX, with auxiliary programs to support a Versatec printer, is now available from Oregon Software. (*Further progress on a* UNIX/TEX *at Brown awaits the arrival of the U. of Washington compiler.*) The accompanying article by Barry Smith explains how you can obtain a copy of the VMS version. You should understand that, in spite of the efforts made by Oregon Software, this is not the final version (in fact, Stanford has just released a new version of TEX). Barry has overcome numerous bugs in DEC Pascal to get us to our present state, but more remains to be done. You can help in this effort by carefully documenting any bugs that you encounter after making sure (insofar as possible) that they are bugs and not errors you are making during your TEX learning process. We have encountered our share of both in the short while the system has been up and running at Sandia. If you fix a bug, please send your fix to Barry!

The distribution tape is 9 track at 1600 bpi and consists of about 6500 blocks. The tape was made using the command `MCR BCK MTA0: TEX.BCK=*.*;*`. The data can be recovered by creating a directory "[TEX]", setting your default to that directory, mounting the tape (using `MOU/OVER=ID MTA0:`), and then recovering the files using

```
MCR RST *.*;* = MTA0:TEX.bck
```

It is suggested that you put all the files on the tape in the [TEX] directory and initially use the program there.

The special Versatec driver (`LVDRIVER.EXE`) must be copied from [TEX] to [SYSEXE] and installed. This is done for Versatecs utilizing a DMA interface by inserting two lines in `[SYSEXE]STARTUP.COM` after `$ RUN SYS$SYSTEM:SYSGEN` but before `AUTOCONFIGURE ALL`. The two lines to add are

```
LOAD LVDRIVER/DRIVER=SYS$SYSTEM:LVDRIVER.EXE
```

and

```
CONNECT LPA0/ADAP=3/NUMVEC=2/VEC=%0174
  /CSR=%0777510/DRIVE=LVDRIVER
```

Reboot the system after these changes have been made.

The programs LVSPOOL and LHSPOOL spool output from the DVI file to the Versatec printer, placing pages of TEX output vertically or horizontally on the printer. To function properly both programs need the privileges ALLSPOOL and PHY_IO. When running from the TEX account the user must have these privileges; if LVSPOOL and LHSPOOL are run in [SYSEXE], the system manager can install them with these privileges so they can be used by any unprivileged user.

Although you could start your first *TEXperience* by using TEX interactively, or with a file built us-

ing your system editor (using Knuth's *TEX and* **METAFONT** as a guide), it is suggested that you might want your first attempt to be with the file TYPO.TEX. To run this example from the [TEX] directory (without using the startup command file provided) enter:

```
R TEX (Wait for the * prompt—
      this may take up to 20 seconds)
\input typo (Note lowercase.)
Watch the TEX comments—wait for system $ )
R LVSPOOL or R LHSPOOL
   (the system should handle the rest)
```

The mailing list for future informational memos regarding VAX/TEX will be the list of DUES PAYING TUG members who indicate a VAX interest, so make sure you join.

* * * * * * * * * * *

TEX FOR VAX/VMS

Barry Smith
Oregon Software

Well, it works—TEX for the VAX running VMS is alive, available, and in production use. (Production use is defined by example—we've just finished a 192 page manual for our optimizing PDP-11 Pascal compiler that is entirely typeset by TEX, including charts and diagrams.)

It's not yet perfect, nor something a "naive user" should be expected to enjoy. To list the major deficiencies:

- There are still annoying bugs, such as the flaming crash that occurs when one tries to insert a footnote. Most of these seem to be due to bugs in Digital's VAX Pascal, which just makes them harder to trace. (There's a new version due out soon, as always.)
- It's the "old" TEX-in-Pascal acquired from Stanford in late December. (Some of the bugs are real TEX bugs, too.) We've just received the recent official release, and as soon as I get a free weekend, ...
- There's absolutely no documentation whatsoever that relates to the VMS version. (Garçon, another weekend, please!) The amazing thing is that this doesn't seem to matter—Knuth's book describes the input format *exactly*, down to subtleties like tracing, and the installation directions are rather concise: "put everything in account [TEX]".

But, when it's good, it's very, very good—we're converting our skeptics to TEXnicians.

Details: TEX-in-Pascal for the VAX (11/750 and 11/780) running VMS, with about 50 Computer Modern fonts in assorted sizes. Comes with two spoolers (horizontal and vertical) for a Versatec 1200 printer/plotter, using the standard Versatec interface. Uses about 7000 blocks of disk space (mostly for fonts) and about a megabyte (whew!) of virtual memory while running. Comes in source and binary/executable forms, sources for spoolers, utilities, etc. (i.e., we'll send you everything we have). If you want to play with the Pascal programs, you'll want the fancy listings available from Stanford. A copy of the Pascal manual mentioned above will be included on request.

Supplied only on magnetic tape (600 foot) in "BCK/RST" format (that's an MCR utility for backup/restore, which works well for binary files). We can write 800 or 1600 b/in tapes—hearing nothing, we'll send 1600.

Fees: fifty dollars will get you a tape and shipment via UPS. To minimize our overhead, please don't send a tape, and do send a check or negotiable securities so we don't have to deal with purchase orders and billing (I'd prefer small unmarked bills). If you (or your friend) are in the "truly needy", just let me know.

Maintenance: We'll be working on bugs and convenience improvements for the perpetual future, and will be pleased to hear comments, suggestions, gripes, bugs—no promises of any response except through the TUGboat and Monte's newsletters.

* * * * * * * * * * *

Fonts

* * * * * * * * * * *

FONT COMMITTEE

Barry C. W. Doherty

Getting TEX up and running is only the first step in producing output. One needs also an output device with an appropriate driver and fonts. Transportable TEX output requires compatible fonts. Those sites which have similiar output devices (Varians etc.) and are willing to use the Stanford Computer Modern family have this problem solved.

There are many, however, who feel limited by the CM family as available from Stanford, either for reasons of design, completeness or merely preference for more traditional fonts (Helvetica, for instance).