

[Sinclair Lewis] felt that his [writing] had suffered in the thirties because during his marriage to Dorothy Thompson he had let her talk him into using professional typists to make the final drafts of his scripts. He now [1943] regretted following her advice, because he was convinced that if an author didn't do his own retyping—revising and improving as he went—he lost some control over his work. Maybe he had a good point; William Faulkner and John O'Hara, for example, did their own typing.

*At Random*, the reminiscences  
of Bennett Cerf,  
Random House, 1977, p. 145

# TUGBOAT

THE T<sub>E</sub>X USERS GROUP NEWSLETTER  
EDITOR BARBARA BEETON

VOLUME 6, NUMBER 1 • MARCH, 1985  
PROVIDENCE • RHODE ISLAND • U.S.A.

## Addresses of Officers, Authors and Others

**ADDISON-WESLEY**  
Reading, MA 01867  
617-944-3700 x2677

**AZZARELLO, Ariane**  
I P Sharp Associates  
220 California Ave  
Suite 201  
Palo Alto, CA 94306  
415-327-1700

**BECK, Lawrence A.**  
Grumman Data Systems  
1111 Stewart Ave  
MS B14-111  
Bethpage, NY 11714  
516-575-9838

**BEETON, Barbara**  
American Mathematical Society  
P. O. Box 6248  
Providence, RI 02940  
401-272-9500  
bb@SU-AI

**BLOCK, Neil**  
Hughes Aircraft Co.  
Bldg. A1, M/S 3C923  
P.O. Box 9339  
Long Beach, CA 90810  
213-513-4891

**CARNES, Lance**  
163 Linden Lane  
Mill Valley, CA 94941  
415-388-8853

**CHILDS, S. Bart**  
Dept of Computer Science  
Texas A & M University  
College Station, TX 77843  
409-845-5470

**CODE, Maria**  
Data Processing Services  
1371 Sydney Dr  
Sunnyvale, CA 94087

**CRAWFORD, John M.**  
Computing Services Center  
College of Administrative Science  
Ohio State University  
Columbus, OH 43210  
614-422-1741  
CSNet: Crawford-J@Ohio-State  
BITNet: TS0135@OHSTVMA

**CUMISKEY, James A.**  
Bord Fáilte-Irish Tourist Board  
Baggot Street Bridge  
Dublin 2, Ireland  
+353-1-765871 x1275

**DUPREE, Chuck**  
Datapoint, Inc.  
9725 Datapoint Drive, MS N-22  
San Antonio, TX 78284  
512-699-7200

**EPPSTEIN, David**  
Arpanet: Eppstein@Columbia-20

**EPPSTEIN, Maureen**  
Administrative Publication Manager  
Stanford University  
Encina Hall, Room 200  
Stanford, CA 94305  
415-497-9254  
Arpanet: MVEppstein@SU-Score

**FUCHS, David**  
Department of Computer Science  
Stanford University  
Stanford, CA 94305  
415-497-1646  
DRF@SU-Score

**FURUTA, Richard**  
Univ of Washington  
Computer Science, FR-35  
Seattle, WA 98195  
206-543-7798  
Furuta@Washington

**GOODELL, Gail**  
Educational & Professional Technologies  
Division  
Addison-Wesley Publishing Co  
Reading, MA 01867  
617-944-3700

**GOUCHER, Raymond E.**  
TeX Users Group  
P. O. Box 9506  
Providence, RI 02940  
401-272-9500 x232

**GROPP, William**  
Dept of Computer Science  
Yale University  
Box 2158 Yale Station  
New Haven, CT 06520  
203-436-3761  
Arpanet: Gropp@Yale

**ION, Patrick D.**  
Mathematical Reviews  
611 Church Street  
P. O. Box 8604  
Ann Arbor, MI 48107  
313-763-6829

**JÜRGENSEN, Helmut**  
Dept of Computer Science  
Univ of Western Ontario  
London N6A 5B7, Ontario, Canada  
519-679-3039

**KELLER, Arthur**  
Computer Science Dept  
Stanford University  
408C Margaret Jacks Hall  
Stanford, CA 94305  
415-497-3227  
ARK@SU-AI

**KELLY, Bill**  
Academic Computing Center  
University of Wisconsin, Madison  
1210 W. Dayton Street  
Madison, WI 53706  
608-262-9501

**KNUTH, Donald E.**  
Department of Computer Science  
Stanford University  
Stanford, CA 94305  
DEK@SU-AI

**LØFSTEDT, Benedict**  
RECAU, Buid. 540  
Ny Munksgade  
8000 Aarhus C, Denmark  
06-128355

**LUCARELLA, Dario**  
Istituto di Cibernetica  
Università di Milano  
Via Viotti 3/5  
20133 Milano, Italy  
23.52.93

**LUVISETTO, M. L.**  
Istituto Nazionale de Fisica Nucleare  
Via Mazzini No 2  
40138 Bologna, Italy  
051-307572

**MacKAY, Pierre A.**  
University of Washington  
Department of Computer Science, FR-35  
Seattle, WA 98195  
206-543-2386  
MacKay@Washington

**MALLET, Rick**  
Computing Services  
Room 1208 Arts Tower  
Carleton University  
Ottawa (K1S 5B6), Ontario Can  
613-231-7145

**NICHOLS, Monte C.**  
Exploratory Chemistry Division  
Sandia National Laboratories 8313  
Livermore, CA 94550  
415-422-2906

**PALAIS, Richard S.**  
Department of Mathematics  
Brandeis University  
Waltham, MA 02154  
617-647-2667

**PIZER, Arnold**  
Department of Mathematics  
University of Rochester  
Rochester, NY 14627  
716-275-4428

**PLASS, Susan**  
Information Technology Services  
Cypress Hall, Jordan Quadrangle  
Stanford University  
Stanford, CA 94305  
415-497-3302

**PLATT, Craig**  
Depr of Math & Astronomy  
Machray Hall  
Univ of Manitoba  
Winnipeg R3T 2N2, Manitoba, Canada  
204-474-9832

**SMITH, Barry**  
Kellerman & Smith  
534 SW Third Ave  
Portland, OR 97204  
503-222-4234

**SOUTHALL, Richard**  
83 Eastern Avenue  
Reading RG1 5SQ, UK  
(0734)67267

**SPIVAK, Michael**  
1660 West Alabama, #7  
Houston, TX 77006

**STROMQUIST, Ralph**  
MACC  
University of Wisconsin  
1210 W. Dayton Straet  
Madison, WI 53706  
608-262-8821

**THEDFORD, Rilla**  
Intergraph Corporation  
One Madison Industrial Park  
Huntsville, AL 35807  
205-772-2000

**TOBIN, Georgia K. M.**  
The Metafoundry  
OCLC Online Computer Library Center, Inc  
6565 Frantz Rd  
Dublin, OH 43017  
614-764-6087

**TUTTLE, Joey K.**  
I P Sharp Associates  
220 California Avenue  
Suite 201  
Palo Alto, CA 94306  
415-327-1700

**UGOLINI, E.**  
Istituto Nazionale de Fisica Nucleare  
Via Mazzini No 2  
40138 Bologna, Italy

**WELLAND, Robert**  
Dept of Mathematics  
Northwestern University  
Lunt Hall  
Evanston, IL 60201  
312-492-3298

**WHIDDEN, Samuel B.**  
American Mathematical Society  
P. O. Box 6248  
Providence, RI 02940  
401-272-9500

**ZABALA, Ignacio**  
Centro de Cálculo  
Facultades de Ciencias  
Universidad de Valencia  
Carnetara de Ademuz  
Valencia, Spain  
011-34-6-357-4065

**ZAPF, Hermann**  
Seitersweg 35  
D-6100 Darmstadt, Fed Rep Germany

TUGboat, the newsletter of the TeX Users Group (TUG), is published irregularly by TUG, P. O. Box 9506, Providence, RI 02940. Annual dues for individual members of TUG include one subscription to TUGboat; fees are detailed on the membership form bound into the back of this issue. Applications for membership in TUG should be addressed to the TeX Users Group, P. O. Box 594, Providence, RI 02901; applications must be accompanied by payment.

Manuscripts should be submitted to a member of the TUGboat Editorial Committee, whose names and addresses are listed inside the front cover. Articles of general interest, or not covered by any of the topics listed, as well as all items submitted on magnetic tape, should be addressed to Barbara Beeton, American Mathematical Society, P. O. Box 6248, Providence, RI 02940.

Submissions to TUGboat are for the most part reproduced with minimal editing. Any questions regarding the content or accuracy of particular items should be directed to the authors.

## TUG Membership Dues and Privileges

### *Memberships and Subscriptions*

1985 dues for individual members are as follows:

North America:

- New (first-time) members or subscribers: \$20.
- Membership and subscription renewals: \$30,  
reduced rate of \$20 for renewals received before January 31, 1985.

Outside North America (includes air mail postage):

- New (first-time) members or subscribers: \$25.
- Membership and subscription renewals: \$35,  
reduced rate of \$20 for renewals received before January 31, 1985.

Membership privileges include all issues of TUGboat published during the membership (calendar) year. Anyone inquiring about TUG will be sent a complimentary copy of TUGboat Vol. 1 (1980), No. 1, along with a current copy of the membership list and forms for acquiring T<sub>E</sub>X82, joining TUG and ordering publications available from TUG.

Issues to domestic addresses are mailed third class bulk, which may take up to six weeks to reach their destinations. If you have not received an issue to which you are entitled, write to TUG at the address given below.

### *Institutional Membership*

1985 Institutional Membership dues for educational organizations are \$200; for non-educational, \$300. Membership privileges include:

- designating up to 5 persons as individual members,
- special reduced rates for participation at TUG meetings and  
T<sub>E</sub>X-related courses and for purchase or lease of videotapes.

In addition, institutional members are acknowledged in each issue of TUGboat. For further information, call Ray Goucher at (401) 272-9500, ext. 232.

### Submitting Items for Publication in TUGboat

The deadline for submitting items for Vol. 6 (1985), No. 2, will be May 15, 1985; the issue will be mailed in mid-July, so that, if possible, copies will be in the hands of members before the August meeting. Contributions on magnetic tape or in camera copy form are encouraged; see "Submitting items to TUGboat", page 78, Vol. 5, No. 2. Editorial addresses are given on the inside front cover. For instructions on preparing magnetic tapes or for transferring items directly to the AMS computer, write or call Barbara Beeton at the address given, (401) 272-9500, ext. 299.

### TUGboat Advertising and Mailing Lists

For information about advertising rates or the purchase of TUG mailing lists, write or call the T<sub>E</sub>X Users Group, Attention: Ray Goucher, P. O. Box 9506, Providence, RI 02940, (401) 272-9500, ext. 232.

## General Delivery

### MESSAGE FROM THE PRESIDENT

Pierre MacKay

In the past few weeks there has been a great deal of excitement over the announcements of new printer systems and software, and some of these announcements may provide the T<sub>E</sub>X Users Group with an even clearer role in the solution of one of the most difficult problems we have continually faced—the relation between public domain and proprietary software in a T<sub>E</sub>X-based document production system.

One of the most remarkable things about T<sub>E</sub>X is its absolute character as public domain software. This character is the result of Don Knuth's personal decision, and of the general organization of the site coordinators distribution system. The principal distributions of T<sub>E</sub>X are subject to no licensing restrictions other than those which inhere in the basic target operating system in which they run. If you have a computer at all, you probably have an operating system for it, and that operating system is certain to be protected by some sort of licensing agreement. The existence of public domain distributions of T<sub>E</sub>X does not preclude the development of other versions with better bells and whistles which may be tied to further licensing restrictions. For example, within the UNIX environment there are at least two compilations of T<sub>E</sub>X (same T<sub>E</sub>X, but different binaries) which depend on the use of specially licensed Pascal compilers. Each of these compilations offers significant advantages over the public domain compilation, and any establishment which happens to have access to the compilers may well want to consider reducing the load on its general computing resources by using a tighter and more efficient binary executable file. The majority of academic users, however, will continue to need the slower, public domain version; first, because special licensing inhibits experimentation, and second because many cannot afford anything but public domain software. The heart of the TUG effort must continue to be public domain distributions.

The most serious difficulties in this regard have always come at the output end of T<sub>E</sub>X systems, and recent developments could make the situation worse, or make it a great deal better. In a few months' time, METAFONT will come of age and will, like T<sub>E</sub>X, be made available through the same public

domain distributions as T<sub>E</sub>X. But what machines will it address, and under what restrictions?

This is a very delicate problem. The protection of font designs and font generation software is essential for the very survival of any manufacturer of typesetting equipment. It is entirely reasonable that the producers of such equipment should guard and protect their proprietary interest in machine-specific elements of their product. Such protection is fundamentally analogous to the licensing control over basic operating systems in general purpose computers, and does not, in itself, seriously restrict the user. Any user who can afford the machine can afford the essential proprietary operating system that goes with it. What the TUG community will need is not access to that, but rather to some kind of open interface which will support the public domain features of METAFONT and will make experimentation possible. Such an interface is not at all impossible, and it appears that we have a real hope that it will be provided, for example, in the case of the new PostScript system. The same thing could be done for most, and probably for all digital phototypesetters, in such a way as to retain and enhance the public domain features of METAFONT while also preserving the manufacturer's vital interest in the security of proprietary software.

### ANSI X3V1.8 LIAISON REPORT TO THE T<sub>E</sub>X USERS GROUP

Lawrence A. Beck

This is my first report to TUG as X3V1.8 liaison and I will try to make it as informative as possible. Just a word about the name change—we used to be X3J6, but as you will see, there has been a reorganization.

I'd like to start with a word about my responsibilities within ANSI and ISO. I've been involved with what was formerly known as ANSI X3J6 and with its ISO counterpart TC97/SC5/WG12 since early 1982. I am currently the Vice Chair of X3V1.8 and Secretary of the Task Group that is charged with developing SGML, X3V1.8.1.

There has been much activity since Lynne Price last reported to you in the September 1983 TUGboat. I'll discuss the administrative changes first.

During 1984, both ISO TC97 and ANSI X3 reorganized and consolidated their Subcommittees and Technical Advisory Groups. This was done for two reasons:

1. to take into account converging technologies, and
2. to cut down on the administrative overhead caused by having many small separate groups.

Within ISO we started out as part of TC97/SC5 (Information Processing Systems/Programming Languages). The TC97 reorganization put us into SC18—Office Systems, which was then renamed *Text & Office Systems*. TC97/SC5/WG12 became TC97/SC18/WG8.

Since ANSI X3 tries to mirror the organization of ISO TC97, X3J6 was merged into X3V1, also called *Office Systems*, which was then renamed *Text: Office & Publishing Systems*. X3J6 became X3V1.8 and was named *Text Processing Languages*.

Also, Mr. Charles D. Card who was the ANSI Chair and the ISO Convenor resigned both positions and was replaced by William W. Davis, Jr., of the U.S. Internal Revenue Service as ANSI Chair and William W. Tunnicliffe of the Graphic Communications Association as the ISO Convenor.

Within the area of technical development, we have continued refinement of the standard and have released Working Drafts 6–9. In August of 1983, Part 6 of the Standard (Document Markup Metalanguage—SGML) was released as a GCA Standard and adopted for trial use by the United States Department of Defense.

In September 1983, Parts 1, 2, 3, 5 and 6 were submitted to ISO for ballot to be considered “Draft Proposed”. The ballot failed as most member bodies did not feel that Part 3 (Programming Language) was in any shape to be a standard, and, in addition, some members did not see a need for a new programming language.

Part 3 is now being re-evaluated and will probably be drafted as a set of general language extensions that could be used to make any block structured programming language more friendly to text.

In November 1984, ISO issued ballots covering Parts 2, 5 and 6 of the Standard (Glossary, Formatting Functions, and SGML) which asked the member bodies for authority to issue these parts as a Draft Proposed Standard. We have every expectation that this ballot will succeed.

As the ISO adoption process moves along, we expect that SGML (Standard Generalized Markup Language) will become very widely used as the method for interchanging documents. It provides

a coherent and unambiguous syntax for describing whatever a user chooses to identify within a document. Projects have already begun, especially in Europe, to develop SGML parsers, including one being done by SOBEMAP in Brussels as part of the “ESPRIT” program.

We believe that as the use of SGML grows, various vendors will find it in their interest either to develop SGML front ends to existing text processing systems, or, as has been done in some cases, to develop new integrated systems based on SGML principles. We hope that TUG will become a willing partner in these efforts.

## ACKNOWLEDGMENT OF CONTRIBUTIONS

The Officers and Steering Committee gratefully acknowledge receipt of royalties and other contributions to TUG from several sources:

1. From the sale of Don Knuth’s *T<sub>E</sub>Xbook*, royalties of \$3,268, with at least \$7,000 more still to come.
2. From David Kellerman and Barry Smith, of Kellerman & Smith, distributors of the VAX/VMS version of the *WEB* sources, along with some related proprietary software (output drivers, etc.), royalties of \$1,425. They incorporated \$25 into the price of every copy as a royalty to be transferred to TUG for “continued support of projects of interest to the TeX community”; Barry said they hoped this would “establish a tradition”.
3. More than 600 copies of Arthur Samuel’s *First Grade T<sub>E</sub>X* sold by TUG in 1984 resulted in the addition of \$5,500 to TUG’s treasury.
4. A contribution of \$500 received from David Rodgers on behalf of Textset, Inc.

TUG sincerely appreciates these very generous contributions.

Samuel B. Whidden, Treasurer

First principles of typographic design for document production  
(TUGboat, Vol. 5, No. 2, pp. 79–90): Corrigenda

Richard Southall

Page 83, lines 6 ff.:

What happens in text?

Written language contains elements that are not alphabetic or numeric characters: punctuation signs, and space. It also has features that seem to operate at a higher level than the words of the text: capitalization, changes in type style and size, and the presence of vertical and horizontal space in varying amounts. These elements and features evidently have some part to play in written language, and some (but surprisingly few) attempts have been made to identify their functions (Mountford, 1980; Walker, 1979; Walker, 1983). The following list of functions is taken from Walker's thesis:

Distinction

    particularization (including emphasis)

    quotation

    interpolation

Abbreviation

Introduction

Omission

Separation and connection

Presentation of numbers

The most significant of these functions seems to be that of distinction/particularization.

References

Page 89, lines 20–22:

Walker, S.F.

Descriptive techniques for studying verbal graphic language  
Unpublished thesis: Department of Typography & Graphic  
Communication, University of Reading, England (1983)

## Low T<sub>E</sub>X

Maureen Eppstein  
 Controller's Office  
 Encina 200  
 Stanford University  
 Stanford, CA 94305

MVEppstein@SU-Score

The purpose of this new column is to offer a forum for discussion of design and layout of material using T<sub>E</sub>X. It is aimed at the beginning or low-tech T<sub>E</sub>X user, although we hope T<sub>E</sub>Xperts will look in occasionally to help answer questions and contribute to discussions about design.

I thought we'd start out with a discussion of letterheads or standard headings you use in your work. I invite you to send me a copy of the standard heading you use in your work, along with a listing of the output routine you use to create it, and a few words about how you designed it. If you have questions, send those too, and we'll try to get them answered for you. I also hope to persuade some typographic designers to give us pointers on how to improve our layouts.

To kick off, I'd like to tell you a little about my own work. At Stanford University I am editor of the Administrative Guide, a weighty manual covering administrative organization, policies, and procedures. Sections of the manual, known as guide memos, are updated on a piecemeal basis.

The previous editors had manually typeset these guide memos. I inherited their format (shown below), which I translated into a T<sub>E</sub>X output routine. The design constraints I had were that the margin at the top of the page should be about half an inch, and I needed a three-string running head that contained the date, page number information, and the Guide Memo number.

The input file for my example is:

```
\input guidemac
```

December 15, 1984

Page 1 of 5

Guide Memo 54.4

```
\input guideoutput
\def\date{December 15, 1984}
\def\pgnm{5}
\def\mnum{54.4}
\centerline{\titlefont Rapid
Purchase Orders }
\vskip 1.2cm
\noindent
This Guide Memo outlines ...
```

My *guidemac* file contains font information, page specifications, and a few macros for recurring layout elements. I use an 11-point computer modern roman font for the body of the text, with a 14-point bold title. *Guideoutput* contains modifications to the `\plainoutput` definitions found on page 255 of the T<sub>E</sub>Xbook.

My first problem was the half-inch margin I needed at the top of the page; the output device I use insisted on giving me a more generous allowance. To trick it, I increased the negative `\vskip` in the `\plainoutput` definition for `\makeheadline` to `-40pt`, a number I found by trial and error.

The `\plainoutput` routine gives a blank top of the page and a centered page number (in 10-point CMR) at the bottom. To obtain my top running head and blank bottom, I redefined `\headline` and `\footline` as follows:

```
\headline={\elevenrm\line
{\date\hfil Page \folio\ of \pgnm
\hfil Guide Memo \mnum}}
\footline={\hfil}
```

Then at the beginning of each Guide Memo file I set `\date` to the revision date, `\pgnm` to the number of pages in the piece, and `\mnum` to the Guide Memo's assigned number. The result is a serviceable format that matches the existing format of the manual.

Now it's your turn. Please send your comments, questions, and contributions to the address at the head of this column, either via paper or network mail.

## Rapid Purchase Orders

This Guide Memo outlines basic procedures for purchases made directly by departments of goods or services totaling less than \$500 per transaction. The objectives of this procedure are to:

- Assist departments to purchase goods and services through an accelerated ordering, receipt, and payment process by allowing departments to order supplies and expendable supplies directly from vendors.

Provide more rapid

department

## Fonts

### GENERIC FONT FILE FORMAT

This is a revised version of the GF file format description published in TUGboat Vol. 5, No. 1, and completely replaces that document. The present version is an extract from the WEB METAFONT documentation.

**1. Generic font file format.** The most important output produced by a typical run of METAFONT is the “generic font” (GF) file that specifies the bit patterns of the characters that have been drawn. The term *generic* indicates that this file format doesn’t match the conventions of any name-brand manufacturer; but it is easy to convert GF files to the special format required by almost all digital phototypesetting equipment. There’s a strong analogy between the DVI files written by T<sub>E</sub>X and the GF files written by METAFONT; and, in fact, the file formats have a lot in common.

A GF file is a stream of 8-bit bytes that may be regarded as a series of commands in a machine-like language. The first byte of each command is the operation code, and this code is followed by zero or more bytes that provide parameters to the command. The parameters themselves may consist of several consecutive bytes; for example, the ‘*boc*’ (beginning of character) command has six parameters, each of which is four bytes long. Parameters are usually regarded as nonnegative integers; but four-byte-long parameters can be either positive or negative, hence they range in value from  $-2^{31}$  to  $2^{31} - 1$ . As in TFM files, numbers that occupy more than one byte position appear in BigEndian order, and negative numbers appear in two’s complement notation.

A GF file consists of a “preamble,” followed by a sequence of one or more “characters,” followed by a “postamble.” The preamble is simply a *pre* command, with its parameters that introduce the file; this must come first. Each “character” consists of a *boc* command, followed by any number of other commands that specify “black” pixels, followed by an *eoc* command. The characters appear in the order that METAFONT generated them. If we ignore no-op commands (which are allowed between any two commands in the file), each *eoc* command is immediately followed by a *boc* command, or by a *post* command; in the latter case, there are no more characters in the file, and the remaining bytes form

the postamble. Further details about the postamble will be explained later.

Some parameters in GF commands are “pointers.” These are four-byte quantities that give the location number of some other byte in the file; the first file byte is number 0, then comes number 1, and so on.

**2.** The GF format is intended to be both compact and easily interpreted by a machine. Compactness is achieved by making most of the information relative instead of absolute. When a GF-reading program reads the commands for a character, it keeps track of two quantities: (a) the current column number,  $m$ ; and (b) the current row number,  $n$ . These are 32-bit signed integers, although most actual font formats produced from GF files will need to curtail this vast range because of practical limitations. (METAFONT output will never allow  $|m|$  or  $|n|$  to get extremely large, but the GF format tries to be more general.)

How do GF’s row and column numbers correspond to the conventions of T<sub>E</sub>X and METAFONT? Well, the “reference point” of a character, in T<sub>E</sub>X’s view, is considered to be at the lower left corner of the pixel in row 0 and column 0. This point is the intersection of the baseline with the left edge of the type; it corresponds to location (0, 0) in METAFONT programs. Thus the pixel in GF row 0 and column 0 is METAFONT’s unit square, comprising the region of the plane whose coordinates both lie between 0 and 1. The pixel in GF row  $n$  and column  $m$  consists of the points whose METAFONT coordinates  $(x, y)$  satisfy  $m \leq x \leq m + 1$  and  $n \leq y \leq n + 1$ . Negative values of  $m$  and  $x$  correspond to columns of pixels left of the reference point; negative values of  $n$  and  $y$  correspond to rows of pixels below the baseline.

Besides  $m$  and  $n$ , there’s also a third aspect of the current state, namely the *paint\_switch*, which is always either *black* or *white*. Each *paint* command advances  $m$  by a specified amount  $d$ , and blackens the intervening pixels if *paint\_switch* = *black*; then the *paint\_switch* changes to the opposite state. GF’s commands are designed so that  $m$  will never decrease within a row, and  $n$  will never increase within a character; hence there is no way to whiten a pixel that has been blackened.

**3.** Here is a list of all the commands that may appear in a GF file. Each command is specified by its symbolic name (e.g., *boc*), its opcode byte (e.g., 67), and its parameters (if any). The parameters are followed by a bracketed number telling how many bytes they occupy; for example, ‘ $d[2]$ ’ means that parameter  $d$  is two bytes long.



- paint\_0* 0. This is a *paint* command with  $d = 0$ ; it does nothing but change the *paint\_switch* from *black* to *white* or vice versa.
- paint\_1* through *paint\_63* (opcodes 1 to 63). These are *paint* commands with  $d = 1$  to 63, defined as follows: If *paint\_switch* = *black*, blacken  $d$  pixels of the current row  $n$ , in columns  $m$  through  $m + d - 1$  inclusive. Then, in any case, complement the *paint\_switch* and advance  $m$  by  $d$ .
- paint1* 64  $d[1]$ . This is a *paint* command with a specified value of  $d$ ; **METAFONT** uses it to paint when  $64 \leq d < 256$ .
- paint2* 65  $d[2]$ . Same as *paint1*, but  $d$  can be as high as 65535.
- paint3* 66  $d[3]$ . Same as *paint1*, but  $d$  can be as high as  $2^{24} - 1$ . **METAFONT** never needs this command, and it is hard to imagine anybody making practical use of it; surely a more compact encoding will be desirable when characters can be this large. But the command is there, anyway, just in case.
- boc* 67  $c[4]$   $p[4]$   $min\_m[4]$   $max\_m[4]$   $min\_n[4]$   $max\_n[4]$ . Beginning of a character: Here  $c$  is the character code, and  $p$  points to the previous character beginning (if any) for characters having this code number modulo 256. (The pointer  $p$  is  $-1$  if there was no prior character with an equivalent code.) The values of registers  $m$  and  $n$  defined by the instructions that follow for this character must satisfy  $min\_m \leq m \leq max\_m$  and  $min\_n \leq n \leq max\_n$ . (The values of  $max\_m$  and  $min\_n$  need not be the tightest bounds possible.) When a GF-reading program sees a *boc*, it can use  $min\_m$ ,  $max\_m$ ,  $min\_n$ , and  $max\_n$  to initialize the bounds of an array. Then it sets  $m\_min\_m$ ,  $n\_max\_n$ , and *paint\_switch.white*.
- boc1* 68  $c[1]$   $del\_m[1]$   $max\_m[1]$   $del\_n[1]$   $max\_n[1]$ . Same as *boc*, but  $p$  is assumed to be  $-1$ ; also  $del\_m = max\_m - min\_m$  and  $del\_n = max\_n - min\_n$  are given instead of  $min\_m$  and  $min\_n$ . The one-byte parameters must be between 0 and 255, inclusive. (This abbreviated *boc* saves 19 bytes per character, in common cases.)
- eoc* 69. End of character: All pixels blackened so far constitute the pattern for this character. In particular, a completely blank character might have *eoc* immediately following *boc*.
- skip0* 70. Decrease  $n$  by 1 and set  $m\_min\_m$ , *paint\_switch.white*. (This finishes one row and begins another, ready to whiten the leftmost pixel in the new row.)
- skip1* 71  $d[1]$ . Decrease  $n$  by  $d + 1$ , set  $m\_min\_m$ , and set *paint\_switch.white*. This is a way to produce  $d$  all-white rows.
- skip2* 72  $d[2]$ . Same as *skip1*, but  $d$  can be as large as 65535.
- skip3* 73  $d[3]$ . Same as *skip1*, but  $d$  can be as large as  $2^{24} - 1$ . **METAFONT** obviously never needs this command.
- new\_row\_0* 74. Decrease  $n$  by 1 and set  $m\_min\_m$ , *paint\_switch.black*. (This finishes one row and begins another, ready to blacken the leftmost pixel in the new row.)
- new\_row\_1* through *new\_row\_164* (opcodes 75 to 238). Same as *new\_row\_0*, but with  $m\_min\_m + 1$  through  $min\_m + 164$ , respectively.
- xxx1* 239  $k[1]$   $x[k]$ . This command is undefined in general; it functions as a  $(k + 2)$ -byte *no\_op* unless special GF-reading programs are being used. **METAFONT** generates *xxx* commands when encountering a **special** string; this occurs in the GF file only between characters, after the preamble, and before the postamble. However, *xxx* commands might appear anywhere in GF files generated by other processors. It is recommended that  $x$  be a string

having the form of a keyword followed by possible parameters relevant to that keyword.

*xxx2* 240 *k*[2] *x*[*k*]. Like *xxx1*, but  $0 \leq k < 65536$ .

*xxx3* 241 *k*[3] *x*[*k*]. Like *xxx1*, but  $0 \leq k < 2^{24}$ . **METAFONT** uses this when sending a **special** string whose length exceeds 255.

*xxx4* 242 *k*[4] *x*[*k*]. Like *xxx1*, but *k* can be ridiculously large; *k* mustn't be negative.

*yyy* 243 *y*[4]. This command is undefined in general; it functions as a 5-byte *no\_op* unless special GF-reading programs are being used. **METAFONT** puts *scaled* numbers into *yyy*'s, as a result of **numspecial** commands; the intent is to provide numeric parameters to *xxx* commands that immediately precede.

*no\_op* 244. No operation, do nothing. Any number of *no\_op*'s may occur between GF commands, but a *no\_op* cannot be inserted between a command and its parameters or between two parameters.

*char\_loc* 245 *c*[1] *dx*[4] *dy*[4] *w*[4] *p*[4]. This command will appear only in the postamble, which will be explained shortly.

*char\_loc0* 246 *c*[1] *dm*[1] *w*[4] *p*[4]. Same as *char\_loc*, except that *dy* is assumed to be zero, and the value of *dx* is taken to be  $65536 * dm$ , where  $0 \leq dm < 256$ .

*pre* 247 *i*[1] *k*[1] *x*[*k*]. Beginning of the preamble; this must come at the very beginning of the file. Parameter *i* is an identifying number for GF format, currently 131. The other information is merely commentary; it is not given special interpretation like *xxx* commands are. (Note that *xxx* commands may immediately follow the preamble, before the first *boc*.)

*post* 248. Beginning of the postamble, see below.

*post\_post* 249. Ending of the postamble, see below.

Commands 250–255 are undefined at the present time.

```
define gf_id_byte = 131 { identifies the kind of GF files described here }
```

4. Here are the opcodes that **METAFONT** actually refers to.

```
define paint_0 = 0 { beginning of the paint commands }
```

```
define paint1 = 64
```

```
{ move right a given number of columns, then black ↔ white }
```

```
define boc = 67 { beginning of a character }
```

```
define boc1 = 68 { short form of boc }
```

```
define eoc = 69 { end of a character }
```

```
define skip0 = 70 { skip no blank rows }
```

```
define skip1 = 71 { skip over blank rows }
```

```
define new_row_0 = 74 { move down one row and then right }
```

```
define xxx1 = 239 { for special strings }
```

```
define xxx3 = 241 { for long special strings }
```

```
define yyy = 243 { for numspecial numbers }
```

```
define char_loc = 245 { character locators in the postamble }
```

```
define pre = 247 { preamble }
```

```
define post = 248 { postamble beginning }
```

```
define post_post = 249 { postamble ending }
```

5. The last character in a GF file is followed by '*post*'; this command introduces the postamble, which summarizes important facts that **METAFONT** has accumulated.

The postamble has the form

```
post p[4] ds[4] cs[4] hppp[4] vppp[4] min_m[4] max_m[4] min_n[4] max_n[4]
```

```
( character locators )
```

```
post_post q[4] i[1] 223's[≥4]
```

Here  $p$  is a pointer to the byte following the final *eoc* in the file (or to the byte following the preamble, if there are no characters); it can be used to locate the beginning of *xxx* commands that might have preceded the postamble. The  $ds$  and  $cs$  parameters give the design size and check sum, respectively, which are exactly the values put into the header of the TFM file that **METAFONT** produces (or would produce) on this run. Parameters  $hppp$  and  $vppp$  are the ratios of pixels per point, horizontally and vertically, expressed as *scaled* integers (i.e., multiplied by  $2^{16}$ ); they can be used to correlate the font with specific device resolutions, magnifications, and "at sizes." Then come  $min_m$ ,  $max_m$ ,  $min_n$ , and  $max_n$ , which bound the values that registers  $m$  and  $n$  assume in all characters in this GF file. (These bounds need not be the best possible;  $max_m$  and  $min_n$  may, on the other hand, be tighter than the similar bounds in *boc* commands. For example, some character may have  $min_n = -100$  in its *boc*, but it might turn out that  $n$  never gets lower than  $-50$  in any character; then  $min_n$  can have any value  $\leq -50$ . If there are no characters in the file, it's possible to have  $min_m > max_m$  and/or  $min_n > max_n$ .)

6. Character locators are introduced by *char\_loc* commands, which specify a character residue  $c$ , character escapements ( $dx$ ,  $dy$ ), a character width  $w$ , and a pointer  $p$  to the beginning of that character. (If two or more characters have the same code  $c$  modulo 256, only the last will be indicated; the others can be located by following backpointers. Characters whose codes differ by a multiple of 256 are assumed to share the same font metric information, hence the TFM file contains only residues of character codes modulo 256. This convention is intended for oriental languages, when there are many character shapes but few distinct widths.)

The character escapements ( $dx$ ,  $dy$ ) are the values of **METAFONT**'s **chardx** and **chardy** parameters; they are in units of *scaled* pixels; i.e.,  $dx$  is in horizontal pixel units times  $2^{16}$ , and  $dy$  is in vertical pixel units times  $2^{16}$ . This is the intended amount of displacement after typesetting the character; for DVI files,  $dy$  should be zero, but other document file formats allow nonzero vertical escapement.

The character width  $w$  duplicates the information in the TFM file; it is a *fix\_word* value relative to the design size, and it should be independent of magnification.

The backpointer  $p$  points to the character's *boc*, or to the first of a sequence of consecutive *xxx* or *yyy* or *no\_op* commands that immediately precede

the *boc*, if such commands exist; such "special" commands essentially belong to the characters, while the special commands after the final character belong to the postamble (i.e., to the font as a whole). This convention about  $p$  applies also to the backpointers in *boc* commands, even though it wasn't explained in the description of *boc*.

Pointer  $p$  might be  $-1$  if the character exists in the TFM file but not in the GF file. This unusual situation can arise in **METAFONT** output if the user had *proofing*  $< 0$  when the character was being shipped out, but then made *proofing*  $\geq 0$  in order to get a GF file.

7. The last part of the postamble, following the *post\_post* byte that signifies the end of the character locators, contains  $q$ , a pointer to the *post* command that started the postamble. An identification byte,  $i$ , comes next; this currently equals 131, as in the preamble.

The  $i$  byte is followed by four or more bytes that are all equal to the decimal number 223 (i.e., '337 in octal). **METAFONT** puts out four to seven of these trailing bytes, until the total length of the file is a multiple of four bytes, since this works out best on machines that pack four bytes per word; but any number of 223's is allowed, as long as there are at least four of them. In effect, 223 is a sort of signature that is added at the very end.

This curious way to finish off a GF file makes it feasible for GF-reading programs to find the postamble first, on most computers, even though **METAFONT** wants to write the postamble last. Most operating systems permit random access to individual words or bytes of a file, so the GF reader can start at the end and skip backwards over the 223's until finding the identification byte. Then it can back up four bytes, read  $q$ , and move to byte  $q$  of the file. This byte should, of course, contain the value 248 (*post*); now the postamble can be read, so the GF reader can discover all the information needed for individual characters.

Unfortunately, however, standard PASCAL does not include the ability to access a random position in a file, or even to determine the length of a file. Almost all systems nowadays provide the necessary capabilities, so GF format has been designed to work most efficiently with modern operating systems. But if GF files have to be processed under the restrictions of standard PASCAL, one can simply read them from front to back. This will be adequate for most applications. However, the postamble-first approach would facilitate a program that merges two GF files, replacing data from one that is overridden by corresponding data in the other.

# A bit of doggerel


*composed upon being asked, for the n<sup>th</sup> time  
how Meta-fonts compare to digitized & scaled fonts*

G.K.M. Tobin

Shall I compare thee to a scaled font?

Thou art more *flexible* by any measure,

Canst GROW, and SHRINK, and **BEND** the way I want,

Thou turn'st  into *prints*, for my pleasure.

Thou makest Æschylus easy as  $\pi$ ;

Thou comest ПО РУССКИИ, with accents equipt;

Thou let'st my words be plain, or go awry,

Or let'st them run *in flowing florid script*.

Thou'rt never cross, although **my mood grows black**.

Thou sayst “√—” with not a fret or ◡.

When I need help, thou never turn'st thy БАКЪ —

Without thee, my world would turn ОБЪИДЕ ДОМЪ.

*And so, as long as on my disk there's space,*

*I'll mold new forms of thy sweet META-face.*

Every character in this document is from a meta-font designed by G.K.M. Tobin using one, more-or-less identical *base.mf*. It was printed on a Canon LBP-CX.

## Output Devices

### Output Devices and Computers

Table I: Proof-Quality Devices

	Amdahl (MTS)	Apollo	CDC Cyber	DEC10	DEC20	DG MV 8000	Ethernet	HP 1000	HP 3000	HP 9000	IBM (MVS)	IBM (VM)	IBM PC	PERQ	Prime	Siemens (BS2000)	Sun	VAX (Unix)	VAX (VMS)	
C Itoh																				LSU
Canon											GMD			GMD		GMD		Canon		
DEC LN01																		UWash	LSU	
DEC Ltr Ptr 100					OSU <sup>d</sup>															
DEC VT125																				INFN
Diablo									Text						OSU <sup>P</sup>					
Epson								JDJW					A-W							
Facit 4542																				INFN
Fla.Data					MR													Textset		
GE3000		COS																		
HP2680							Stnfd		Text											
HP2688A										HP; CaTch										
IBM APA 6670												SLAC								
Imagen	UBC	OCLC		Stnfd; Vndblt	SRI; Climbia		Imagen					SLAC	OCLC <sup>‡</sup>				Sun	UCIrv	K&S <sup>†</sup>	
NDK 7700												IAM								
QMS Lasrgrfx	Textset	ScnLsr; Textset		Textset	Textset	TAMU					Textset	Textset		GMD	TAMU		Textset	Textset	TAMU	
Qume Sprint 5									Text											
screen prevue		Yale; Textset										GMD		GMD		GMD	Textset			Adid
Symbolics					UWash													UWash	Calma	
Talaris				Talrs <sup>†</sup>	Talrs <sup>†</sup>													Talrs <sup>†</sup>	Talrs <sup>†</sup>	
Tektronix 4014											UMilan									Adid; INFN
Varian					AMS															SciAp
Versatec			UKöln	GATch; Vndblt	UWash						UMilan	Wzmn			Lvmr			UWash	K&S <sup>†</sup>	
Xerox Dover					CMU		Stnfd												Stnfd	
Xerox 2700II					OSU <sup>d</sup>															
Xerox 9700	UMich; Textset	COS		UDel								UDel					Textset			

**Notes:**

\* Still running T<sub>E</sub>X80

† Graphics supported

‡ Computer used only to support output device,  
not to run T<sub>E</sub>X at this installation.

Table II: Typesetters

	Amdahl (MTS)	Amdahl (MVS)	Apollo	CDC Cyber	DEC20	HP3000	HP9000	IBM (MVS)	IBM (VM)	Sun	Univac 1100	VAX (Unix)	VAX (VMS)
Agfa P400									IAM				
Alphatype CRS					AMS								
APS-5/Micro-5	Textset	WashStU	COS; Textset		Textset	Textset	HP	Textset	Textset	Textset		Textset	Intergraph†; Textset
Compugraphic 8400						USheffield							K&S†
Compugraphic 8600		WashStU		RECAU*							UWis*		
CRTronic													Eire
Harris 7500												SARA	
Linotron 202					Adapt								

Most of the interfaces listed in these charts are not on the standard distribution tapes. Some are considered proprietary. Information regarding these interfaces should be obtained directly from the sites listed.

Output device data is being maintained by Rilla Thedford. Anyone desiring more information or relaying new information can send it to her at the address given on the reverse of the title page or via the Arpanet:

Rilla.Thedford%UMich-MTS@MIT

The codes used in the charts are interpreted below, with a person's name given for a site when that information could be obtained and verified. If a contact's name appears in the current TUG membership list, no further information beyond a phone number is given. If the contact is not a current TUG member, the most recent address, and its source, are shown.

**Adapt** (Adapt, Inc): Marc Berkowitz, 415-393-9500

**Aldd** (Adelaide University, Australia): Andrew Trevorrow, (08) 228 5984

**AMS** (American Math Society): Ron Whitney, 401-272-9500

**A-W** (Addison-Wesley): 617-944-3700, ext. 2677

**Eire** (Bord Fáilte - Irish Tourist Board): James Cumiskey, Dublin 353-1-765871, ext. 1275

**Calma**:

**CalTech**: Glen Gribble, 818-356-6988

**Canon** (Tokyo): Masaaki Nagashima, (03)758-2111

**Cmbia** (Columbia): Frank da Cruz, 212-280-5126

**CMU** (Carnegie-Mellon University): Howard Gayle, 412-578-3042

**COS** (COS Information, Montreal): Kevin Small, 514-738-2191

**GATech** (G A Technologies): Phil Andrews

**GMD** (Gesellschaft der Math und Datenfabrik, Bonn, Germany): Dr. Wolfgang Appelt

**HP** (Hewlett-Packard): Stuart Beatty, 303-226-3800, ext. 2067

**IAM** (Institut für Angewandte Math, Univ of Bonn, Germany): Bernd Schulze, 0228-733427

**Imagen**: Dan Curtis, 408-986-9400

**INFN** (INFN/CNAF, Bologna, Italy): Maria Luisa Luvisetto, 051-307572

**Intgrph** (Intergraph): Mike Cunningham, 205-772-2000

**JDJW** (JDJ Wordware): John D. Johnson, 415-965-3245

**K&S** (Kellerman & Smith): Barry Smith, 503-222-4234

**LSU** (Louisiana State University): Neal Stoltzfus, 504-388-1570

**Lvmr** (Lawrence Livermore Lab):

**MR** (Math Reviews): Dan Lattner, 313-764-7228

**OCLC**: Tom Hickey, 616-764-6075

**OSU** (Ohio State University): DEC 20: John Gourlay,

614-422-6653; Prime: John Crawford, 614-422-1741

**RECAU** (Aarhus University, Regional Computer Center): Benedict Løfstedt, 06-128355

**SARA** (Stichting Acad Rechenzentrum Amsterdam):

Han Noot, Stichting Math Centrum, Tweede Boerhaavestraat 49, 1091 AL Amsterdam (TUGboat 5#1)

**ScnLsr** (Scan Laser, England): John Escott

**SciAp** (Science Applications): L. E. Fields

**SLAC**: Alan Spragens, 415-854-3300 x2849

**SRI**:

**Stnfd** (Stanford):

**Sun** (Sun, Inc):

**TAMU** (Texas A&M): Bart Childs, 415-965-3245

**Text**: Lance Carnes, 415-388-8853

**Textset** (Ann Arbor, Mich.): Bruce Baker, 313-996-3566

**Talrs** (Talaris): Sonny Burkett, 619-454-3363

**UBC** (Univ of British Columbia): Afton Cayford, 604-228-3045

**UCIrv** (Univ of California, Irvine):

**UDel** (Univ of Delaware): Daniel Grim, 302-451-1990

**UKöln** (Univ of Köln, Germany): Jochen Roderburg, 0221-/478-5372

**UMich** (Univ of Michigan): Hal Varian, 313-764-2364

**UMilan** (Università Degli Studi Milan, Italy):

Tektronix: Dario Lucarella, 02/23.62.441 (329);

Versatec: Giovanni Canzii, 02/23.52.93

**USheffield** (Univ of Sheffield, England): Ewart North, (0742)-78555, ext. 4307

**UWash** (Univ of Washington): Richard Furuta, 206-543-7798

**UWis** (Univ of Wisconsin): William Kelly, 608-262-9501

**Vndblt** (Vanderbilt University): H. Denson Burnum, 615-322-2357

**WashStU** (Washington State University): Dean Guenther, 509-335-0411

**Wzmn** (Weizmann Institute, Rehovot, Israel):

Malka Cymbalista, 08-482443

**Yale**: Bill Gropp, 203-436-3761

## Device drivers included on the UNIX $\TeX$ distribution tape

Richard Furuta

Editor's note: An effort will be made to obtain similar information for other distribution tapes for the next issue of TUGboat.

These device drivers are currently on the Unix  $\TeX$  tape. The authors may no longer be where I say they are.

Device	Source language	Authors
Imagen 10/240 and 5/480	C	Pavel Curtis, Cornell Mike Urban, TRW Ralph Campbell, Berkeley
Imagen 8/300	C	Chris Torek, U. Maryland
DEC LN01 and LN01S	C	Samuel Bent, U. Wisconsin; some parts by Scott Jones, MIT
Versatec/Varian	C	Janet Incerpi, Brown Robert Morris, UMass, Boston Richard Furuta, U. Washington Carl Binding, U. Washington Chris Torek, U. Maryland
Symbolics LGP-1	WEB (Pascal)	Richard Furuta, Pierre MacKay, U. Washington

### DVI previewers

BBN Bitgraph	C	Mark Senn, Purdue James Schaad, U. Washington
SUN II	C	based on dvibit Norm Hutchinson, U. Washington

We expect to have a new version of the SUN II previewer sometime in the foreseeable future. The current version does not run under Sun's window system. The new version uses the window system. In fact, it runs within a window.

We also hope to have a QMS 1200 (and maybe QMS 800) driver soon. Three have been contributed, and one is in testing—if no major bugs surface, it will be added.

We also can distribute, on request, a copy of our Symbolics LGP-1 driver for for Tops-20:

Device	Source language	Authors
Symbolics LGP-1	WEB (Pascal)	Richard Furuta, Pierre MacKay, U. Washington

## Index to Sample Output from Various Devices

Camera copy for the following items in this issue of TUGboat was prepared on the devices indicated, and can be taken as representative of the output produced by those devices. The bulk of this issue, as usual, has been prepared (all with  $\TeX$ 82) on the DEC 2060 and Alphatype CRS at the American Mathematical Society.

- Canon CX (300 dpi): G. K. M. Tobin, "A bit of doggerel", p. 12.
- Epson FX-80 (240×216dpi): Addison-Wesley's Micro $\TeX$  announcement, p. 27, and advertisement, p. 42; IBM PC using Micro $\TeX$ .
- HP 2688A Laser Printer (300 dpi): Lance Carnes, "small  $\TeX$ ", p. 26; HP 3000.
- Mergenthaler CRTronic 150 (50 lines/mm): James Cumisky,  *$\TeX$  for tourists!* p. 21; VAX 11/750 (VMS).
- QMS Lasergrafix 800 (300 dpi): Textset advertisements, pp. 41 and 44.

## A $\TeX$ 82 SPOOLER FOR VT AND DOT MATRIX PRINTERS

M. L. Luvisetto, E. Ugolini  
Istituto Nazionale di Fisica Nucleare  
Bologna (Italy)

We have been running  $\TeX$ 80 on VAX/VMS since fall 1982. Last February we installed  $\TeX$ 82 and converted our spooler to DVI version 2. Our institute has different sites all over Italy equipped with VAX/VMS linked through a private network; only some of the sites are equipped with VERSATEC to produce the final output, while most of the sites have graphic terminals and/or dot matrix printers. In order to enable  $\TeX$  proofreading at each site, we have developed a general-purpose spooler with the aim of supporting a wide range of graphic devices with special attention to VT displays of any type and price and structured in such a way as to enable easy upgrading of the system if a new device is needed. At present the supported devices are:

1. Tektronix series 4000 (PLOT10)
2. Facit 4542 (Native mode)
3. Digital VT125 (REGIS)

The spooler is written in FORTRAN 77 mainly for two reasons: our institute is concerned with High Energy Physics research and FORTRAN is the official language (and practically the only well known one) and in this way the package can achieve a high degree of modularity through routine libraries both for general purpose routines and for drivers. Therefore it should be easy for other users to develop and include new drivers for output devices different from the distributed ones. In addition our software should have a high degree of portability, as the non-ANSI code is confined in a well defined and small number of routines. Furthermore some debug utilities that were developed at the early stage of the spooler implementation (i.e. programs to print DVI files and to display PXL contents) could be used as already debugged nucleus for the spooler itself and can be distributed with the whole package.

The spooler reads the DVI file keeping always a record ahead in storage and creates a full bit map for a standard  $\text{\TeX}$  page at a maximum of 300 dots/inch. The bit map is stored in a common array. As PXL information is variable in size and number of PXL files required per run, we have developed a simple method to allocate virtual memory through VAX/VMS facilities. The method takes advantage of the directory structure used to store the same fonts at different magnification and for different resolution devices. For this to work in a most convenient way, we have created a pilot file with names and sizes of font information.

The spooler can accept a certain amount of options (i.e. page selection, output device, magnification, etc.) in the form of VMS qualifiers, both lower and upper case allowed for file names and options. A list of valid qualifiers is printed when HELP is typed at prompt. To change magnification the qualifier /MAG=value is provided (ex. /MAG=1500 will produce a page at 300 dots/inch). If  $\text{\TeX}$  was run setting the magnification to some value different from 1000, then this value is used, and the /MAG qualifier is ignored. The default value for MAG is 1000, that is 200 dots/inch. By default the output is produced on a disk file (file type .BIT), then the user can plot the page on any of the forseen devices, using the stand-alone version of the related driver. As already pointed out, the main idea of keeping the driver inside and outside the spooler itself is to enable other users to implement their own drivers for new devices with small effort.

If a device allows two different resolutions (as Tektronix 4014), the driver is able to handle both, with default set to high resolution; if the device

offers a sufficiently high resolution, a complete A4 page width can be displayed, thus providing a reasonable proofreading feature. In the case of VT displays, to speed up the drawing of one page the bit map is plotted using vectors, and the driver signals the end of one page using the bell and waits for the return key before going to the next one. For the printer the driver sets no limit to page length. As VT125 vertical resolution is 480 with odd y emulation (i.e. 240 pixels), the output may seem strange as some lines may look taller than others. The driver for the dot matrix printer has been written making use of the pin adressability feature. To enable the use of the program as a guide line for other similar devices, this driver makes use of one single pin.

## Site Reports

### HOW TO GET THE LATEST NEWS VIA $\text{\TeX}$ HAX

Between issues of TUGboat,  $\text{\TeX}$  users who have access to the Arpanet, BITnet or CSNet can get up-to-the-minute news of what's happening in the  $\text{\TeX}$  community via  $\text{\TeX}$ hax. A number of sites, particularly universities with active local populations of  $\text{\TeX}$  users, have  $\text{\TeX}$  bulletin boards that are already plugged into  $\text{\TeX}$ hax, and most, although not all, of the TUG Site Coordinators are also linked in. To get your name added to the distribution list, send a network message to

**TeXhax-request@SU-Score.ARPA**

giving your name, full net address, and perhaps an indication of the particular sorts of things you're interested in.

If you have a query, some news, or a response to an earlier  $\text{\TeX}$ hax message that you want to announce to everyone on the list, send it to

**TeXhax@SU-Score.ARPA**

Your message will be digested, along with other communications, and broadcast, with occasional editorial comments in [square brackets], by the acting  $\text{\TeX}$ hax editor, David Fuchs. An attempt will be made to report items of general interest in the next issue of TUGboat.



Recent  $\TeX$ hax communications have dealt with the following topics:

- Use of the variable name `first` in  $\TeX$  version 1.3; this name doesn't meet ISO Pascal standards, and will be changed.
- $\LaTeX$ , both inquiries and announcements of additional style files, including one for viewgraphs and several for memos.
- The possibility of using a `tty` for "preview" output or on-line documentation.
- Requests for more information about, or explanation of, various macros defined in the  $\TeX$ book, such as `\raggedbottom`, `\obeylines`, `\obeyspaces`.
- Persuading  $\TeX$  to consider hyphenating the second part of an already-hyphenated word.
- Inquiries about ports of  $\TeX$  to particular computers or drivers for particular output devices. (In this regard, see the charts and associated list of contacts on pages 13-14; these include the latest information from both  $\TeX$ hax communications and the TUG records.)
- Announcement of a macro package, `BoTeX`, designed for formatting Lisp manuals; inquiries to George J. Carrette, `GJC@MIT-MC`.
- Announcement of a shell script (for use with UNIX 4.2) to automatically generate a preloaded version, equivalent to using `INITEK` and `VIRTEX` to generate  $\TeX$  from `plain.tex`, or  $\LaTeX$  from `lplain.tex`; inquiries to Stephan Bechtolsheim, `SVB@Purdue`.
- Inquiries about fonts, including questions about bugs in existing fonts, whether particular fonts exist (e.g. cyrillic, and text fonts with smaller capital letters for typesetting German), and a request for help with sample input for the new **METAFONT** to test a new port. (Sample MF input is on the distribution tapes beginning with the set containing  $\TeX$  1.3.)

## PRIME SITE REPORT

John M. Crawford  
Ohio State University

Things here have been pretty lively with respect to  $\TeX$ . Over the past several months, I have enjoyed discussing  $\TeX$  with many Prime users. Several

sites in the United States, as well as England, Ireland and Australia have received the OSU Primos implementation of  $\TeX$ . While on vacation, I was pleased to follow-up a tape delivery with a personal visit to a Prime site in Paris. Professor Laurent Breyton (E.S.I.E.E.) and I spent a very pleasant afternoon discussing some of the general problems one encounters with text processing French. We also examined " $\TeX$  in Pascal" implementation concerns (his group was working with a different compiler). Although I can not visit all Prime  $\TeX$  sites, I am interested in knowing how things are progressing and welcome feedback.

My  $\TeX$  distribution tape has matured. I now have  $\TeX$  1.1 available. This latest Primos port of  $\TeX$  also contains some general improvements; existing sites will likely want to obtain the latest tape. The current  $\TeX$  (dated 12/17/84) has been enlarged for  $\LaTeX$ . It also has return code hooks tied to the  $\TeX$  `history` variable, and removes an accidental "feature" pertaining to the level of file protection required by  $\TeX$  system files. Some of the  $\TeX$  utilities have been improved as well. If you are interested in "REVing up", please me know. One unambiguous way to express your interest is to send a tape, which will be returned in short order with the goodies.

I spent a very cold January weekend working on implementing the new **METAFONT**. Following further testing, I expect **METAFONT** in its present form will be included on my distribution tape. This should occur, fingers crossed, by the time this TUGboat hits the newsstands.

On the device driver scene, I have operational a driver which can be used on Diablo-like devices (limitations accepted). I plan to use it to drive the Texas Instruments TI855 matrix printer, with enhancements permitting the dynamic utilization of the various font modules supported by the printer.

I have not vigorously tried to obtain a device driver for the Printronix 300 LPM matrix printer, but expect one would generate a lot of interest in  $\TeX$  at other Prime sites. (The printer is marketed with many Prime systems.) If any friendly  $\TeX$  users would like to contribute a driver for this lumpy device, please consider me interested. (I can read most any tape format, but a nicely blocked tape is best).

## UNIX T<sub>E</sub>X SITE REPORT

Richard Furuta

We have made some administrative adjustments in our handling of Unix T<sub>E</sub>X orders. Requests for information about Unix T<sub>E</sub>X and orders for Unix T<sub>E</sub>X now are sent to Pierre MacKay, in his role as the other Unix T<sub>E</sub>X Site Coordinator. His address and telephone number are:

Pierre MacKay  
 Computer Science, FR-35  
 University of Washington  
 Seattle, WA 98195  
 Telephone: (206) 545-2386  
 Arpanet/CSNet: MacKay@Washington  
 UUCP: ihnp4!uw-beaver!uw-june!mackay

Pierre will only rarely be in on Tuesdays and Thursdays because of his obligations with other departments on campus.

We now have Unix T<sub>E</sub>X for 4.1 bsd and 4.2 bsd on the VAX and 4.2 bsd on the Sun. Some Ultrix sites have ordered the T<sub>E</sub>X tape and since they haven't complained, I believe that the distribution also runs there. Still no System III or System V versions. The distribution is available on a 2400 foot magnetic tape reel and is written in tar format at 1600 bpi. To get it, send Pierre a check for \$75 made out to the University of Washington and a copy of your 4.1 bsd or 4.2 bsd source license. If you don't have the source license, please note that in a cover letter and instead of sending you the entire tape, we'll send you a partial distribution that will work for the 4.2 bsd (and Ultrix) computers. We cannot accept purchase orders—if your company won't issue a check, please call for alternate instructions.

As mentioned, the distribution is on standard magnetic tape. There is a possibility that we can write parts of it onto the Sun's streamer tapes for those sites that cannot read magnetic tapes. However, the ordering details will certainly be different from those involving the standard media. If you require this, contact Pierre MacKay first to make arrangements.

So much for administrative matters—on to the technical status of the distribution. Our previous report (in the November TUGboat) covered through the beginning of October, 1984. This report is being written near the beginning of February, 1985.

In October, Chris Torek of the University of Maryland contributed another whiz-bang device driver—this time for the Imagen 8/300. The new driver is built using most of the modules developed for his previously submitted Versatec driver and is written in C. Reports are that it is significantly

faster than the previous Imagen driver although it does not yet have some of the niceties provided in the previous driver such as extraction of individual pages for printing. Both of the Imagen drivers are now on the tape. Also in October, Samuel Bent of the University of Wisconsin sent us his device driver for Digital's LN01 laser printer. Scott Jones of MIT submitted changes that make this driver usable on the LN01S. (A revised version of the LN01/LN01S driver went onto the tape in late January.) We still don't have a QMS driver on the distribution tape but we do have a couple of experimental ones that we have been making available to people who ask. One, by Karl Berry of Dartmouth, is a port of the WEB VMS driver written by Jane Colman of Lawrence Berkeley Laboratory. Another QMS driver, written in C, was provided to us some time ago by Bill White of Megatest.

T<sub>E</sub>X 1.2 and L<sup>A</sup>T<sub>E</sub>X 2.07 arrived at the beginning of November. At the same time, we merged in the changes needed to get T<sub>E</sub>X, **tangle**, and **weave** running on the Sun II. More discussion about our T<sub>E</sub>X on the Sun will be at the end of this report. As mentioned in the last report, these changes were provided by Steven Correll (Lawrence Livermore National Lab), Rusty Wright (U.C. San Diego), and Charles Perkins and Mike Harrison (U.C. Berkeley). Thanks also to Howard Trickey for taking time out from dissertation writing to track down a bug in the port.

T<sub>E</sub>X version 1.3 was placed on the distribution at the end of December. At that time, there were also some updates to L<sup>A</sup>T<sub>E</sub>X although the version number remained at 2.07.

Van Jacobson of Lawrence Berkeley Laboratory sent along a Scribe to L<sup>A</sup>T<sub>E</sub>X conversion program. This program, called **s2latex**, is expected to get about 90% of the translation right the first time it is run. You can then add translation rules and generally get almost all of the way to a document that L<sup>A</sup>T<sub>E</sub>X will accept. Some hand editing will get you the rest of the way there. Jacobson has translated the Gosling Emacs manual with this program. **s2latex** is written using **lex** and **yacc** in C. It was added to the tape in late December.

During this period, there have also been small bugfixes to **detex** and **delatex**. Some missing font magnifications were included. A new version of Lou Salkind's programs to convert PXL format files to RST format was acquired. A fix to Chris Torek's Versatec and 8/300 drivers, which fixes a bug which T<sub>E</sub>X doesn't encounter, will be installed soon. We also included a copy of the **patch** program in the distribution. **patch**, written by Larry Wall and

acquired via Usenet, recognizes the various output formats produced by the Unix `diff` program and merges the changes into the original copy. This makes it a lot more convenient for people on our electronic mailing list to merge in the bugfixes that we send out.

L<sup>A</sup>T<sub>E</sub>X 2.08 was released in late January and was put onto the Unix T<sub>E</sub>X distribution tape.

Since there is usually a lag of about a month between when I write the site report and when you see it, I'd like to close this segment of the report with some predictions about what might be in the Unix T<sub>E</sub>X distribution by publication time.

I expect that we will be getting a Unix version of **WEB METAFONT** in the very near future. Paul Richards of the University of Illinois has been doing the development and the reports I have received indicate that all is going well. It's another question, of course, as to when we will have font descriptions for the new Metafont, but I imagine other articles in this issue of TUGboat will tell you about that!

I also expect that an improved version of the Sun DVI previewer will become available. Jeff McCarrell of U.C. Berkeley has been working to produce a version that works within a window on the Sun (previous versions have taken over the entire screen for their display).

We have been using a prerelease of Oren Patashnik's `bibtex`, which will be L<sup>A</sup>T<sub>E</sub>X's bibliography processor. The `bibtex` user will prepare bibliography files in a format that is very similar to that used by Scribe. Our perspective is a little different, however, as potential developers of bibliography descriptions, we are also interested in using the provided implementation language. I have been very impressed with the power and flexibility of the language (much more powerful than the equivalent language in Scribe) and look forward to its eventual release.

### T<sub>E</sub>X on the Sun

I began by merging together the changes to the VAX distribution submitted by the gentlemen named above. There were surprisingly few changes—the primary one was a change to a variable declaration to avoid problems created by the different ordering of bytes in the two machines. The Sun we used for this purpose is located in the Computer Science Department and is running version 1.2 of Sun's software. It is a diskless workstation with two megabytes of physical memory. Compilation and loading was uneventful—the primary trick needed was to break up the source into multiple pieces.

(This approach was suggested by both Steven Correll and by the people at Berkeley.) When the source wasn't divided up, the assembler seemed like it would be perfectly happy to run forever (more about this later in an even more hostile environment—a one megabyte Sun). I have noticed similar effects on VAXes with small amounts of physical memory. When we broke the source up in four pieces, the process took a much more reasonable amount of time—on the order of two to three hours, a time in the same range as that on the VAX (but note, below, how this time increases for the one megabyte Sun).

I have been very pleased with the speed of the implementation—astonished, actually. On an unloaded Sun, the implementation seems to run at about the same speed as on an unloaded VAX 11/780 (both running 4.2 bsd). While it is clear that with appropriate compilers, the VAX implementation could run at least several times faster than it does now, this comparison indicates to me that the Sun T<sub>E</sub>X is quite serviceable.

T<sub>E</sub>X's library areas take up about one megabyte of disk space (with L<sup>A</sup>T<sub>E</sub>X but without any font bitmaps). The binaries for `initex`, `virtex`, a preloaded T<sub>E</sub>X, and a preloaded L<sup>A</sup>T<sub>E</sub>X require another three megabytes of disk storage.

At the present time, we've only tried compiling **tangle**, **weave**, **undump**, and the various forms of T<sub>E</sub>X.

Pierre's Sun is a one megabyte configuration. He sends along some notes on his experiences with T<sub>E</sub>X:

### T<sub>E</sub>X on a Small Sun

In addition to the growing network of Sun workstations located in the Department of Computer Science, and linked in the appropriate manner through the Ethernet, there is an isolated installation at the Department of Near Eastern Languages, where "T<sub>E</sub>X for Arabic Script" will one day become a reality. This unit has an 84-megabyte disk and only one megabyte of on-board memory, and is linked by a slow 1200-baud UUCP line to Computer Science. Even if the challenge of compilation had not been attractive, there would be very good reasons for not attempting to pass T<sub>E</sub>X binaries across such a channel. But trying to compile on a 1-megabyte Sun is not a simple matter.

I began by taking the makefile just as it is distributed with our Unix T<sub>E</sub>X tapes. This breaks up the output from the `pxp` prettyprinter into four modules, and the result is successful on a 2-megabyte Sun. On a 1-megabyte Sun, the attempt

to compile even a quarter of  $\TeX$  produces what has been referred to as pathological paging. I let an unloaded machine work on `initex1.p` for 36 clock hours before I stopped it, and determined that the CPU (judging from the times shown in a `ps` listing) was working about 60% of its time on  $\TeX$ . When I shut it off, the assembler had accounted for about 12 hours, and the page daemon for about 11 hours. There was no indication that the job would ever be finished. The percentage of time used by the page daemon was growing, and I suspect that 12 hours later the page daemon would have been taking the larger share of time.

So the answer was to break down  $\TeX$  into even smaller units, and the obvious units were the individual procedures and functions. I now have `sed` scripts which separate  $\TeX$  into all 333 procedures and functions for individual compilation, and it looks as if this may be the best way to handle the problem of compiling  $\TeX$  on a large variety of smaller Unix systems. The `sed` scripts depend on the complete reliability of the `pxp` reformatting, so they are as yet rather dependent on some 4.2 bsd features. Separate compilation is of course a must, and the loader must be reasonably sophisticated since it is impossible to get the full dependency statement onto one "line" in a makefile. (In case you have ever wondered, the limit for continuation lines in `make` appears to be about 50 medium length lines—2500 characters seems to be a good guess.)

Total time for "make `initex`" was 11 hours, and for "make `virtex`" about 10 hours. That includes the laborious working of the `sed` script to do the separation.

The intermediate stages of compilation require a large amount of disk space. Ten megabytes seems to be safe, and 8 may just be enough. The villains are the `*.o` files, which take a minimum of 23 KBytes each. Considerable savings would be possible by loading the first half of the executable file and ruthlessly eliminating all the related `*.o` files before compiling the second half. Even so, I should imagine that five megabytes of free disk is the absolute lowest limit for compilation. The ultimate executable file is quite reasonable: 696 KBytes for `initex`, and 324 KBytes for `virtex`. A fully loaded  $\TeX$ , with `plain.fmt` preloaded, occupies 900 KBytes of disk space and `size` shows that about the same amount of virtual memory will be occupied when it runs. But, since the vast preponderance of that space is the huge general

purpose working storage, I suspect that I will rarely need to page in the last 300 KBytes in any actual run. On short files, even my *small* Sun is FAST.

## VAX/VMS SITE REPORT

Barry Smith

VAX/VMS version 4.0 is the current hot topic for VMS users. If your copy of  $\TeX$  doesn't run, and complains about a bad parameter, give me a call—there's a one-byte patch that will correct the problem. What does it do? Well, embedded in the Pascal run time library is an undocumented `Open()` parameter that would have been written as `'prompt := always'`, and caused output to that file to be written immediately, with no buffering. This is useful for keeping terminal output up-to-date, as when  $\TeX$  is displaying the current output page number. This internal parameter has been removed from VMS 4.0 (DEC? Remove a feature?) and the easiest fix is to remove its use as well. (There is a better solution that runs on all current VMS versions, but it can't be patched into existing systems.)

Andrew Trevorrow, Adelaide University, Australia, reports that he has a screen driver for several common graphic terminals including the Tektronix 4010/4014 series. This has been requested by several people. He said he'll be sending it on to us to add to the tape; contact Andrew directly if you're in a hurry.

The Macintosh  $\TeX$  implementation project is going well; we've solved (twice) the addressing limitations of the Apple Pascal compiler and should be running code "in a couple of weeks". Meanwhile, Apple has introduced the LaserWriter and the AppleTalk network; should be easily the best cost/performance  $\TeX$  system available.

One final note—we've moved. Our new address is

Kellerman & Smith  
534 SW Third Avenue  
Portland, OR 97204  
503-222-4234

## **T<sub>E</sub>X for Tourists!**

*James A. Cumiskey  
Irish Tourist Board*

### **Introduction**

In November, 1982 Bord Failte – Irish Tourist Board received Oregon Software's release tape of T<sub>E</sub>X (the Pascal version as existed in 1980). We installed the T<sub>E</sub>X, WEB, WEAVE and TANGLE systems and the T<sub>F</sub>toPL and PLtoT<sub>F</sub> programs on a Vax 11/750 running under VMS. Our principal interest was to produce camera-ready copy in-house for our range of tourism information guides on accommodation and catering premises and other tourism facilities.

After some months of experimentation (mainly on an Anadex and a Diablo) and having done an in-depth analysis of the market, we purchased a Mergenthaler (Linotype) CRTronic 150 phototypesetter and also prepared a device driver (DviPtu).

### **Mergenthaler CRTronic 150 phototypesetter**

The CRTronic is a CRT phototypesetting device with its fonts stored in digitized form on disk — a total of sixteen fonts can be available on-line (8 in memory, 8 on floppy disk). The CRTronic's output resolution is 50 scan lines per mm and estimated speed is 40 news-column lines per minute.

Output is produced on photographic paper which is then chemically processed to produce high-quality, high-resolution, high-contrast output for printing.

The CRTronic includes a complete editing system of its own, but we have generally ignored this in favour of editing on the VAX and producing blocks of output which are then copied downline to the CRTronic's floppy disk.

The range of type sizes available is from 4.5pt to 72pt. A very large range of fonts is available from the manufacturer's own "font-shop". To cater for our range of requirements, we purchased Times Roman, Italic and Boldface, Univers 45, 55, 65, and 75, and a set of special character fonts including all the general range of symbols needed in text and four "tourist-symbols" fonts.

For each font we prepared a property-listing file and then converted each to TFM format using the PLtoT<sub>F</sub> program.

### **Device Driver program – DviPtu**

Having gone through the TUG users list and found that no installation had yet prepared a device driver for the CRTronic, we wrote our DviPtu program to translate each DVI command to CRTronic readable commands. Our earlier experience with the Diablo and Anadex and study of other drivers helped a lot. We built a set of checks into DviPtu to ensure that all device-dependent parameters are correct before the job is sent to the CRTronic (for instance, page width and depth must be within certain limits).

Once DviPtu was ready, we then designed and developed a system for the computerization of our principal tourism information for publishing. This included screen-entry forms, proofreading programs and programs to output verified data with embedded T<sub>E</sub>X commands (macros, control sequences or primitives as required). Sets of macros were also prepared for each accommodation and catering guide to allow for differing formats and fonts.

### **Operation of the Device Driver**

Experience has shown us that the size of batches which can be set on the CRTronic depends to a large extent on the complexity of the material sent from T<sub>E</sub>X. For each job, (for instance, a guide-book), we tell the program the size of the batch that is required. As DviPtu reads the DVI file, it prints batches of pages to an output directory with the same filename as the input file and a file extension from .AA - .ZZ in alphabetical order of sets of pages output. We have prepared a command procedure to ease the use of T<sub>E</sub>X and DviPtu and to queue the output batches in a "typesetting queue" on the VAX. Each job is then released in turn on to the CRTronic and typeset.



If you feel a restaurant is outstanding, please fill in this form and send it to:

**Accommodation and Catering Dept.  
Bord Failte  
Baggot Street Bridge  
Dublin 2.**

As a result of my personal experience, I nominate the following restaurant from "Dining in Ireland" and/or "Tourist Menu" Scheme for a Bord Failte Distinction.

Name of Restaurant . . . . .

Address . . . . .

I had Lunch/Dinner/Snack there on . . . . .

I am not connected directly or indirectly by . . . . .

"Dining in Ireland"   
Please indicate

"Tourist Menu"

Si

My Name and Home Address (block capital letters) . . . . .

I consider this restaurant outstanding because . . . . .

. . . . .

. . . . .

. . . . .

. . . . .

. . . . .

. . . . .

. . . . .

(Please give reasons including food, service, atmosphere, etc. Nominations must be submitted in writing to the Bord Failte, Baggot Street Bridge, Dublin 2, on or before September 30th of the year of nomination.)

**CO. CARLOW**

**CARLOW**

**REDDY'S GRILLROOM, 67 Tullow Street, Carlow.** Telephone (0503)42224. *Special Value Menu Period: 1 May-30 Sep. Closed: Good Friday, Christmas Day & St Stephen's Day.*  
★ 12:30-14:30 ■ 17:30-20:30

**ROYAL HOTEL, Dublin Street, Carlow.** Telephone (0503)31621/33728. *Special Value Menu Period: 1 Jan-23 Dec. Closed: Sundays-Limited service.*  
★ 12:30-14:00 ■ 18:00-20:30

**CO. CAVAN**

**CAVAN**

**THE GALLEY, Main Street, Cavan.** Telephone (049)32205. *Special Value Menu Period: 1 Jan-31 Dec.*  
★ 12:30-14:30; 19:30-24:00

**KILLESHANDRA**

**LOUGH BAWN HOTEL, Killishandra, Co Cavan.** Telephone (049)34404/34423. *Special Value Menu Period: 1 Jan-31 Dec. Closed: 24 to 27 December.*  
■ 12:45-15:00; 17:45-20:30

**CO. CLARE**

**SEAFOOD HOUSE, Fanore, Coast Road, Lisdoonvarna-Ballyvaughan, Co Clare.** Telephone Craggagh 205. *Special Value Menu Period: 1 May-30 Sep. Closed: 1 Jan-14 May, 16 Oct-31 Dec.*  
■ 12:00-22:00

**COROFIN**

**BOFEY QUINN'S, Corofin, Co Clare.** Telephone (065)27627. *Special Value Menu Period: 1 Jan-31 Dec.*  
★ 12:00-16:00 ■ 16:00-22:00

**COROFIN**

**MARYSE & GILBERT'S RESTAURANT, Corofin, Co Clare.** Telephone (065)27660. *Special Value Menu Period: 18 Apr-31 Aug. Closed: 1 Jan-19 Apr, 1 Oct-31 Dec; Sundays & Mondays (Mondays*

only in July & August).

■ 19:00-22:00

**DOOLIN**

**BRUACH NA HAILLE RESTAURANT, Roadford, Doolin, Co Clare.** Telephone (065)74120. *Special Value Menu Period: 1 Jun-30 Sep. Closed: 1 Jan-31 May, 1 Nov-31 Dec.*

■ 13:00-16:30; 18:00-21:30

**ENNIS**

**AUBURN LODGE HOTEL, Galway Road, Ennis, Co Clare.** Telephone (065)21247. *Special Value Menu Period: 1 May-31 Oct. Closed: Christmas Day & St Stephen's Day.*

■ 12:45-14:30; 18:30-20:30

**GOLDEN MOUNTAIN, 16 O'Connell St, Ennis, Co Clare.** Telephone (065)29491. *Special Value Menu Period: 1 Jan-31 Dec. Closed: Christmas Day & New Year's Day.*

★ 12:00-14:30 ■ 18:00-22:00

**LADY GREGORY'S RESTAURANT, Abbey Arcade, Abbey Street, Ennis, Co Clare.** Telephone (065)24240. *Special Value Menu Period: 1 Jan-31 Dec. Closed: Sundays & Bank Holidays..*

★ 12:15-14:30 ■ 12:15-14:30

**QUEENS HOTEL, Abbey Street, Ennis, Co Clare.** Telephone (065)28963. *Special Value Menu Period: 1 Jan-30 Sep. Closed: Christmas Day.*

★ 12:30-15:00; 17:00-21:00 ■ 12:30-15:00; 17:00-21:00

**SADDLE ROOM STEAK HOUSE, Vandeleur Street, Kilrush, Co Clare.** Telephone Kilrush 258. *Special Value Menu Period: 1 May-30 Sep. Closed: 25 & 26 December.*

★ 12:45-14:30; 18:00-20:30

**LAHINCH**

**ATLANTIC HOTEL, Lahinch, Co Clare.** Telephone (065)81049, Telex 28136. *Special Value Menu Period: 1 Jun-31 Oct. Closed: 1 Jan-31 May, 1 Nov-31 Dec.*

★ 12:30-14:30 ■ 18:00-19:00

**McINERNEYS SANCTA MARIA HOTEL, Lahinch, Co Clare.** Telephone (065)81041. *Special Value Menu Period: 1 Jun-30 Sep. Closed: 1 Jan-29*

## CORPORATE SERVICES DIVISION

## Administrative Services Department

Name and Title	Phones	Room	Job Description
MURRAY, NIALL Clerical Assistant	1180	G12-B	Admin Assistance
MCGARRY, ROBERT Clerical Assistant Postal	1204	B6-B	Post
HENNESSY, LIAM Messenger	1204	B6-B	Motor-cycle Messenger
DOYLE, GERARD Driver/Maintenance Assistant	1286	C	Van Driver/Security Maint - Chancery
<b>OFFICE SERVICES</b>			
BARRINGTON, NANCY (OSS) Office Services Supervisor	1390	G7-B	Office Services Telephone Faults
KING, BREDA Services Supervisor	1228	B2-B	Sec Unit/Reception Switch/Telex
DILLON, ROSEMARY Typist	1332	B1-B	Typing/Reception & Switch Reliefs
FITZPATRICK, ROSALEEN Typist	1332	B1-B	Typing/Telex & Reception Pembroke R.
HAYDEN, MAUREEN Typist	1332	B1-B	Typing/Alternate Supervisor
KEOGH, NOREEN Typist	1332	B1-B	Typing/Reception & Switch Reliefs
KELLY, MARIE Typist	1332	B1-B	Typ/W-Proces/Typesetting Recep P Row
KILMARTIN, MARGARET Typist	1332	B1-B	Typing and Reception Pembroke Row
MCCABE, MARIE Typist	1391	B1-B	Typing & Word Processing
MCKEVITT, CATHERINE	1332	B1-B	Typing/W-Proces/Switch &



# Digital Equipment Computer Users Society

## VAX Special Interest Group 1983 Annual General Meeting Panel Session with DEC Management 16 September 1983

### Timetable

September 16, 1983

Morning	10:25	VAX Special Interest Group – Introduction
	10:30	“All-in-One: End User Computing in the Bank of Ireland” by Charles Williams
	11:30	Open Forum for news and views of VAX users
	12:30	Lunch
Afternoon	2:30	AGM
	3:00	DEC New Products Presentation and Panel Session
	4:30	End of Meeting

### Location

Room 128/129  
Physics Building  
University College  
Belfield, DUBLIN 4

(parking available)

### Programme

There are two parts to the meeting, a VAX Special Interest Group meeting in the morning and a chance to air your views to DEC management in the afternoon. Senior managers of Digital Ireland Ltd will form a panel to accept views (and complaints) from DECUS members. Each of the major functional areas in DIGITAL will be represented.

Before the panel session there will be a short presentation on new DEC products and services.

### Annual General Meeting

The 1983 Annual General Meeting will be held at 2:30 pm. The following positions will be open for election:

Chairman – DECUS Ireland (the holder of this office is also the representative on the council of DECUS UK and Ireland)

Deputy Chairman

Treasurer

Newsletter Editor

Program Library Co-Ordinator

The following positions are automatically filled:

DECUS UK and Ireland Chapter Administrator (Ray Woodyear)

VAX Sig Chairman (Ken O'Brien)

CMG Sig Chairman (Rory O'Connor)

Nominations to the vacant positions may be made at the AGM or, in writing, to the DECUS Ireland secretary, *Elizabeth Crossan, c/o Digital Equipment Ltd, Park House, North Circular Road, DUBLIN 7*

Please bring this notice to the attention of all DECUS members and any others who might be interested in your organisation. Membership of the Digital Equipment Computer Users Society is open to any user of DEC computers and is free of charge.

\* \* \* \* \*

"small" T<sub>E</sub>X

Lance Carnes

In the previous issue, I discussed several IBM PC T<sub>E</sub>X versions that were in progress. By October, 1984, two versions were running, one mine, and one by David Fuchs. By the time you read this, the marketing groups supporting both versions will have sent you at least one piece of mail describing their products. Additional articles and advertisements on these two versions are found elsewhere in this issue.

The two versions are similar in that they execute on an 8088- or 8086-based computer running MS-DOS. Each requires at least 512Kb of memory

(640Kb to run LaTeX) and a 10Mb hard disk. David's version was done with Lattice-C and requires an 8087 coprocessor. My implementation was done with MS-Pascal and does not need an 8087 coprocessor. Both versions compile the T<sub>E</sub>Xbook at an average of 25 seconds per page.

Barry Smith of Kellerman and Smith reports that the Apple Macintosh version is well underway, and they expect to have a running version by the time you read this. They intend to use the "Fat Mac" (512Kb memory version) and the Macintosh XL, and direct output to the recently announced Laserwriter.

Below is the grid for known small T<sub>E</sub>X implementations. All versions are T<sub>E</sub>X82 unless otherwise noted. Let me know if you have additional information.

"small" T <sub>E</sub> X implementations			
Computer	Processor	Contact	Organization, Address
Hewlett-Packard 3000	16-bit	Lance Carnes	TeX <sub>E</sub> L, 163 Linden Lane, Mill Valley, CA 94941; 415-388-8853
Hewlett-Packard 1000	16-bit	John Johnson	JDJ Wordware, Box 354, Cupertino, CA 95015; 415-965-3245
DEC PDP-11/44 Plexus, Onyx IBM PC	16-bit <sup>1</sup> Z8000 <sup>1</sup> 8086/88 <sup>1</sup>	Dick Gauthier	TYX, 11250 Roger Bacon Dr., Suite 16, Reston, VA 22090; 703-471-0233
Apollo	MC68000	Thom Hickey Bill Gropp Pierre Clouthier	OCLC, Box 7777, Dublin OH 43017; 614-764-6075 Dept. of Computer Science, Yale University, Box 2158, Yale Station, New Haven, CT 06520; 203-436-3761 COS Information, 6272 Notre Dame West, Montreal, H4C 1V4, P.Q.; 514-935-4222
Hewlett-Packard 9836	MC68000	Jim Crumly	Hewlett-Packard, Box 15, Boise, ID 83707; 208-376-6000 x2869
Sun Microsystems	MC68000	Jim Sterken Rich Furuta	Textset, Box 7993, Ann Arbor, MI 48107; 313-996-3566 University of Washington, Computer Science, FR-35, Seattle, WA 98195; 206-543-7798
Cyb	MC68000 <sup>2</sup>	Norman Naugle	Mathematics Dept., Texas A&M University, College Station, TX 77843; 409-845-3104
Apple Macintosh, Lisa	MC68000 <sup>3</sup>	Barry Smith Dave Kellerman	Kellerman & Smith, 2343 SE 45th Av., Portland, OR 97215; 503-232-4799
Masscomp	MC68000	Bart Childs	Dept. of Computer Science, Texas A&M University, College Station, TX 77843; 409-845-5470
Synapse	MC68000	Dick Wallenstein	Comcon, 5 Underwood Ct., Delran, NJ 08075; 609-764-1720
PERQ/ICL		Jaap van 't Ooster	Océ, St. Urbanusweg 43, 5900 MA Venlo, Holland
IBM PC, XT, AT	8088, 80286	Lance Carnes	TeX <sub>E</sub> L, 163 Linden Lane, Mill Valley, CA 94941; 415-388-8853
IBM XT, AT	8088, 80286	David Fuchs	Dept. of Computer Science, Stanford University, Stanford, CA 94305
IBM XT, AT	8088, 80286 <sup>3</sup>	Ronny Bar-Gadda	446 College Av., Palo Alto, CA 94306; 415-326-1275

<sup>1</sup> not T<sub>E</sub>X82<sup>2</sup> currently unimplementable<sup>3</sup> in progress or recently completed

## TeX Now on Microcomputers

At the January meeting of AMS, Addison-Wesley Publishing Company demonstrated a full implementation of TeX82 for the IBM PC/XT, or IBM PC with hard disk. This implementation, called MicroTeX<sup>TM</sup>, was developed by David Fuchs, who had to work quite a few tricks to shoehorn the system into the PC/XT and then to achieve high performance. The MicroTeX package also includes Addison-Wesley's driver for IBM and Epson dot matrix printers (IBM Graphics and Matrix, Epson MX-, RX-, and FX-80 and 100).

MicroTeX is an exact translation of TeX82. David first wrote a translator to convert Pascal code to C, added overlay facilities to allow TeX to operate effectively with its fonts in 512K of memory, and then compiled this program for an MS-DOS environment. In the process, David had to work around the limitations, for his purposes, of the best available C compiler. Among other things, he rewrote its math, I/O, and memory management runtimes, plus a post-compiler code optimizer and FMT file preloader.

The resulting MicroTeX includes all the commands of TeX and plain TeX, as well as all the fonts of plain TeX. Since MicroTeX is TeX, it produces .DVI files identical to those produced by any implementation of TeX82. As proof, MicroTeX-generated files that were printed locally by Addison-Wesley on an Epson printer were sent to Textset, Inc., in Ann Arbor, Michigan, and printed without a hitch on their QMS laser printer and APS5 phototypesetter.

MicroTeX runs under DOS 2.1 and requires 512K of memory, much of which is required to hold macros; the 8087 co-processor is not required. With this configuration, it has been timed to process *The TeXbook* in 3 hours 37 minutes, or 26.4 seconds per page for difficult and dense text. This speed is more than acceptable for most purposes, and even compares favorably with processing times on larger machines in actual timeshared environments. For less dense text, you can decrease the processing time to under 20 seconds.

When running TeX, you will get the "\*\*\*" prompt 12 seconds after you invoke TeX on the XT. This is a fully pre-loaded "plain TeX." In the 512K of memory, you have a 30,000 word "mem" array, and a 20,000 word "font" array; 640K will make

those arrays 60,000 and 22,000 words, respectively. All other arrays are full-size, even in 512K; that is, the arrays are the same size as on mainframe and minicomputer implementations of TeX. All data for TeX are kept in RAM; overlays are used only for infrequently used code, to minimize the cost of these overlays.

The 16 plain TeX typefaces included in the MicroTeX package are each provided at 2 dot densities, 240 dpi and 120 dpi, and each of these 32 densities, in turn, is provided at 6 magsteps: 1200, 1315, 1440, 1728, 2074, and 2488 for the 240 dpi fonts, and 600, 657, 720, 864, 1037, and 1244 for 120 dpi. The complete MicroTeX package, including TeX, fonts, and system utilities, resides in 4 megabytes of hard disk space. The package is installed from 8 floppy disks by an installation program.

When running Addison-Wesley's driver to print a .DVI file, the user can specify values for numerous processing and printing options. These options include: print quality, magstep, horizontal and vertical offsets, level of error analysis to display on the screen, where to find font files, and which pages or portions of pages in the file to process. The values hold only for that run of the driver program; the program uses default values for any options not specified. Addison-Wesley's configuration program allows the user to set values for all but one of these options, and thereby make them the new default values.

The user can select from 5 levels of print quality and can also specify the definitions of any or all print quality levels. A draft quality level provides a very readable output of a .DVI file at almost the normal speed of the dot matrix printer; highest resolution print quality takes longer, because it requires more passes of the printer, but the results are well worth it. The user may also choose to process only a portion of a .DVI file and can specify the starting position of the starting page, and the ending position of the ending page.

The MicroTeX package includes several other programs. PXL-EPF converts font files from a pixel format to a column-oriented format more useful to Epson type printers. Another program expands font files that had been stored in compressed form on the floppy disks; the program then erases the compressed font file from the hard disk, to reclaim storage space. The installation program transfers MicroTeX, the font files and the other programs in the package to the hard disk; it then invokes the pro-

gram to expand the font files.

Addison-Wesley also expects to have a version of Micro $\TeX$  for the IBM PC/AT available soon. Enhancements to the package, such as L $\TeX$ , the availability of additional drivers, and the publication of related software and books will be announced at appropriate times in the coming months. Addison-Wesley welcomes suggestions from TUG members on what is needed to make  $\TeX$  even more useful and broadly available, particularly in the microcomputer environment, and welcomes discussion with you on work you may be doing in this area yourself. Contact Addison-Wesley Publishing Company, Reading, MA 01867. 617-944-3700, extension 2677.

(NOTE: This article was printed, on an Epson FX-80 printer, at 240 by 216 dpi.)

## PC $\TeX$

Lance Carnes  
Personal  $\TeX$ , Inc.

Having successfully ported  $\TeX$  to the HP3000, I wanted to bring it to a wider market. Last summer, I formed a business with Scott Guthery and Michael Ballantyne, two  $\TeX$  users from Austin, Texas, to bring up  $\TeX$  on the IBM PC.

I started the project in September, 1984. Getting a version running on the PC, however, was not so simple a matter as merely compiling the  $\TeX$ .PAS source from TANGLE. The Microsoft Pascal compiler (version 3.20) would not compile the "vanilla" Pascal code as written, and I had to create a translation program to modify the code so that it would compile cleanly, let alone run. By October, I had a version of  $\TeX$  running that could at least completely process Don Knuth's TRIP test, though not correctly. (I have since eliminated the eight or ten minor problems.)

This first, inefficient version of PC  $\TeX$  ran agonizingly slowly, taking 40 to 60 seconds to compile each page of a document. I recognized that not only would I have to eliminate the TRIP bugs, I must optimize the run-time performance.

I spent the next months debugging and optimizing. I was able to make use of much of the considerable amount of data I had amassed about my HP3000 version of  $\TeX$ , particularly run-time characteristics. Since  $\TeX$  is not optimized for particular machines, I was not surprised to find

that a majority of the program's execution time on the PC was spent in file I/O, memory management, and in the modules identified as "inner loop" in the index of the  $\TeX$ 82 Manual.

I was able to improve file I/O performance through direct DOS calls rather than using the Microsoft Pascal run-time package. This speeded things by 10 to 15%. For example, one improvement was in the way  $\TeX$  reads source input files. In the original code, these files are read one character at a time. The optimization consisted of reading the file in large blocks and de-blocking these into records by dedicated routines.

The most significant problem in optimizing memory management appeared in addressing the two arrays most often used by  $\TeX$ , *mem* and *font.info*. In most implementations of  $\TeX$ , these arrays are assigned storage greater than 64K. (*mem* is usually allotted 120 to 240K, and *font.info* 80 to 120.) However, Microsoft Pascal (in common with most other compilers for the PC) cannot address arrays larger than 64K. To properly implement  $\TeX$  on the PC, I had to devise methods of getting around the 64K limitation. This I did by judicious use of segmented pointer types and by writing efficient machine-level address calculation routines. These memory management optimizations speed up run time by 16 to 20%.

Other miscellaneous optimizations included replacing unnecessary division and multiplication operations with other implementation-dependent operations, and changing data types to those more efficient for the particular machine. Each reduced run time by a few percent.

My last step was to eliminate the final few remaining bugs. As an interesting sidelight, during the entire development period, I discovered only one bug in the Microsoft Pascal compiler (32-bit integer division).

PC  $\TeX$  is now a completed product. It compiles the  $\TeX$ book at an average rate of 25 seconds per page. We timed it on an IBM PC/XT with 512K memory (without an 8087 coprocessor). It should run three times as fast on an AT, that is, about eight seconds per page.

During the time that I was writing Pascal code, and learning all about the 8088 processor and MS-DOS, my partners and I met frequently to discuss marketing and distributing PC  $\TeX$ . Dana Ballantyne, Michael's brother, brought his business expertise to the company, joining us as a partner in December.

Personal  $\TeX$ , Inc., offers a full implementation of the  $\TeX$  document compiler. PC  $\TeX$  includes

L<sup>A</sup>T<sub>E</sub>X, AM<sup>S</sup>T<sub>E</sub>X, and our own macro package that lets beginners quickly begin producing documents themselves, plus drivers for the IBM Graphics printer, and Epson RX, FX, and LQ series printers.

Elsewhere in this issue is an advertisement for PC<sup>T</sup>E<sub>X</sub> that tells you how to order and how much it costs.

Future revisions will include a preview screen driver, and drivers for QMS, Imagen, Apple's Laser-Writer, and other popular output devices. We will also offer customized macro packages, written by Michael Spivak, each aimed at specific needs, such as publishing, business, and education.

If you have questions about PC<sup>T</sup>E<sub>X</sub>, you can reach me weekdays from 9 a.m. until at least 6 p.m. (PST), or leave a message with my answering service.

Lance Carnes  
 Personal T<sub>E</sub>X, Inc.  
 20 Sunnyside, Suite H  
 Mill Valley, CA 94941 USA  
 (415)388-8853. TELEX 910-481-0421

## Macros

### MACROS FOR TWO-COLUMN FORMAT

Craig Platt  
 University of Manitoba

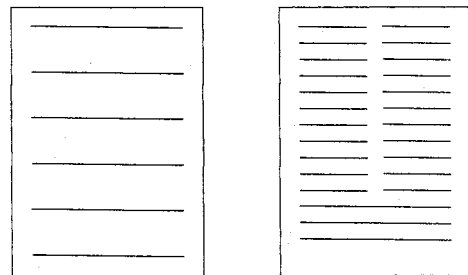
In Appendix E of the T<sub>E</sub>Xbook, Don Knuth presents the macros that were used for two-column formatting in Appendix I, the index. For their intended purposes they seem to have worked well enough, but there are a couple of circumstances under which they may fail. I discovered the first of these while trying to adapt the macros to another context, and in the process of working out a fix, Don came across the other.

The first problem can arise when switching from single-column to double-column mode near the end of a page, and then back to single-column mode "too soon" on the next page. Referring to page 417 of The T<sub>E</sub>Xbook, the `\begindoublecolumns` macro operates by first saving the current `\box255` in `\partialpage`, changing the output routine to `\doublecolumnout`, changing `\hsize` to `\colwidth`, and changing `\vsize` to `\bigcolheight`, which is a bit more than twice the original `\vsize`. This allows

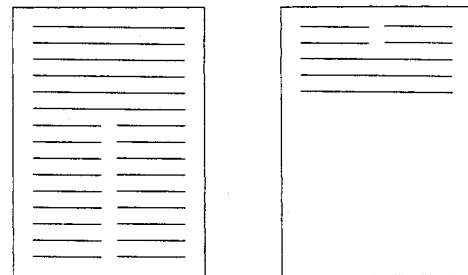
a very tall column to accumulate, after which, when `\doublecolumnout` is invoked, `\vsplit` is used to extract columns of the correct height.

When `\enddoublecolumns` occurs, the output routine `\balancecolumns` uses a `\loop` in an attempt to split the current `\box255` into two columns of equal height. Then `\pagesofar` packages these boxes side by side and contributes the result (along with the `\partialpage`, if any) to the current vertical list, and normal processing resumes.

The problem is that the alteration of `\vsize` by `\begindoublecolumns` doesn't take the height of `\partialpage` into account. This can allow `\box255` to grow too large, creating a situation that `\balancecolumns` can't handle. Consider a case where `\begindoublecolumns` has occurred on a given page and more than enough material has accumulated in the main vertical list to fill the remainder of the page, if split equally. Since `\begindoublecolumns` has set `\vsize` for a full double-column page, this might not be enough material to cause the output routine to be invoked. If `\enddoublecolumns` occurs at this point, `\balancecolumns` will split the *entire* `\box255` into two equal parts, and the resulting columns won't fit onto the page along with `\partialpage`. The result is that `\partialpage` gets put onto the current page as a badly underfull `\vbox`, and the two columns get held over for the next page. You get something like this:



instead of this:



Of course, this situation never occurs in Appendix I because before `\enddoublecolumns` finally occurs, several full-size double-column pages have intervened, so that `\partialpage` is empty.

The cure is to set a value of `\vsize` which ensures that the `\doublecolumnout` output routine gets “triggered” at the right time. In the modified versions of the macros below, the changes have been marked by `%%n` for easy reference. The `\save size` register is used to hold the original `\vsize`, so it can be restored by `\enddoublecolumns`. In `\begindoublecolumns`, the new line `%%5` adjusts `\vsize` to compensate for `\partialpage`, and line `%%8` resets it to `\bigcolheight`, since a fresh page will start after `\doublecolumnout` operates.

A second problem with the macros in Appendix E is that under certain circumstances it is possible for `\balancecolumns` to be invoked twice in a row. Note that the final act of `\balancecolumns`, after splitting `\box255`, is to invoke `\pagesofar`,

which contributes its results to the current vertical list. It is possible for the resulting list to be big enough to trigger the output routine, which at this point is still `\balancecolumns`. Since `\box255` no longer contains a “column” to be split, the results will tend to be chaotic. In this kind of situation it is likely that the page *cannot* be properly balanced, so the best thing to do is warn the user and go on. This is accomplished by changing the output routine *inside* `\balancecolumns` to simply produce an error message. This happens in line `%%9`, and then line `%%6` is required to restore normal output in `\enddoublecolumns`.

Note that `\colwidth` and `\bigcolheight` here replace the T<sub>E</sub>Xbook values of 14pc and 89pc respectively, and that `\dimen0` replaces `\dimen@`.

```

\newdimen\colwidth \newdimen\bigcolheight           %%%1
\colwidth=14pc \bigcolheight=89pc                   %%%2
\output{\onepageout{\unvbox255}}
\newbox\partialpage
\newdimen\save size                                  %%%3
\def\begindoublecolumns{\begingroup
  \save size=\vsize                                  %%%4
  \output={\global\setbox\partialpage=\vbox{\unvbox255}}\eject
  \output={\doublecolumnout} \hsize=\colwidth \vsize=\bigcolheight
  \advance\vsize by -2\ht\partialpage}              %%%5
\def\enddoublecolumns{\output={\balancecolumns}}\eject
\global\output={\onepageout{\unvbox255}}           %%%6
\global\vsize=\save size                             %%%7
\endgroup \pagegoal=\vsize}
\def\doublecolumnout{\dimen0=\pageheight
  \advance\dimen0 by-\ht\partialpage \splittopskip=\topskip
  \setbox0=\vsplit255 to\dimen0
  \setbox2=\vsplit255 to\dimen0
  \onepageout\pagesofar
  \global\vsize=\bigcolheight                       %%%8
  \unvbox255 \penalty\outputpenalty}
\def\pagesofar{\unvbox\partialpage
  \wd0=\hsize \wd2=\hsize \hbox to\pagewidth{\box0\hfil\box2}}
\def\balancecolumns{\setbox0=\vbox{\unvbox255} \dimen0=\ht0
  \advance\dimen0 by\topskip \advance\dimen0 by-\baselineskip
  \divide\dimen0 by2 \splittopskip=\topskip
  {\vbadness=10000 \loop \global\setbox3=\copy0
    \global\setbox1=\vsplit3 to\dimen0
    \ifdim\ht3>\dimen0 \global\advance\dimen0 by1pt \repeat}
  \setbox0=\vbox to\dimen0{\unvbox1}
  \setbox2=\vbox to\dimen0{\unvbox3}
  \global\output={\balancingerror}                  %%%9
  \pagesofar}
\newhelp\balerrhelp{Please change the page          %%%10
  into one that works.}                             %%%11
\def\balancingerror{\errhelp=\balerrhelp           %%%12
  \errmessage{Page can't be balanced}              %%%13
  \onepageout{\unvbox255}}                          %%%14

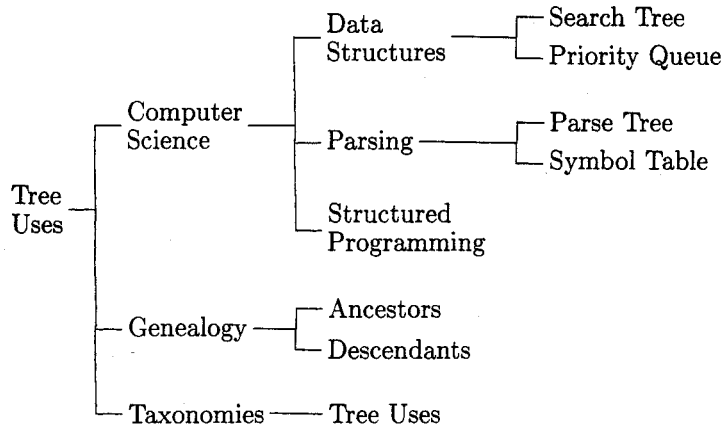
```

**TREES IN T<sub>E</sub>X**

David Eppstein

**Introduction**

There are many possible uses for trees in typeset text. The following taxonomy illustrates some of them.



Unfortunately T<sub>E</sub>X provides no easy way to typeset such trees. One possible method is given in exercise 22.14 of the T<sub>E</sub>Xbook: using T<sub>E</sub>X's alignment primitives by hand. This method becomes very clumsy as the trees grow, however. A more general technique is to write a set of tree construction macros; that is the approach taken in this paper. The taxonomy above was typeset with the following input:

```

\tree
  Tree
  Uses
  \subtree
    Computer
    Science
    \subtree
      Data
      Structures
      \leaf{Search Tree}
      \leaf{Priority Queue}
    \endsubtree
  ...
\endsubtree
...
\endtree

```

It turns out that T<sub>E</sub>X's alignment primitives are not very well suited to automatic generation of trees. The left edges of the trees at each level can easily be made to line up, but it is difficult to center lines of text for the root of a tree in vertical relation to its subtrees. Instead, the macros described here construct trees from boxes and glue, doing the alignment themselves. This is not quite as simple as it sounds—it would be incorrect to set subtrees independently of each other, because then the edges would not line up. For instance, in the taxonomy above, the text “Search Tree” should line up with “Parse Tree”. A one-pass algorithm would set the former somewhat to the right of the latter.

To solve this problem, the macros described here set a tree using three passes. First, a data structure is built up from the tree definition. Second, that data structure is used to calculate the width of each level of the tree, so that the subtrees can be aligned with each other. Finally, the data structure and the calculated list of widths are used to set the system of boxes, glue, and rules that make up the tree.

**Pass 1: Internal storage of the tree structure**

There are several possible ways to store the structure defined by the tree macros. Since we want to remember already-set text (the words at the roots of each tree or subtree) we will use a nested structure of boxes. Each subtree is stored in an `\hbox`, so that pieces of it can be pulled off easily using `\lastbox` and `\unskip`. To distinguish it from another subtree, the text at the root of a subtree is stored in a `\vbox`. To make this clearer, let us return to our original taxonomy. We shall ignore for the moment the details inside the text `\vboxes`, and the glue between boxes. After the first pass, the tree as a whole would look like the following set of boxes:

```
\hbox{\vbox{Tree Uses}
  \hbox{\vbox{Computer Science}
    \hbox{\vbox{Data Structures}
      \hbox{\vbox{Search Tree}}
      \hbox{\vbox{Priority Queue}}}}
  ...}
...}
```

Now we can begin defining the tree macros. We start defining a tree with the `\tree` macro; this merely sets up the assignment of the boxed tree structure into a box called `\treebox`. Starting a subtree is similar, but there is no assignment; also, if it is the first `\subtree` of its tree or subtree, we must stop making the `\vbox` containing the root text. A leaf is merely a subtree without any sub-subtrees.

```
\newbox\treebox
\def\tree{\global\setbox\treebox=\boxtree}
\def\subtree{\etext \boxtree}
\def\leaf#1{\subtree#1\endsubtree}
```

Finishing a subtree merely involves first making sure the root text is complete, and second completing the box that was started in the expansion of `\subtree`. Finishing a whole tree involves both of those steps, but then after the box is completed the remaining two passes must be run.

```
\def\endsubtree{\etext \egroup}
\def\endtree{\endsubtree \settreesizes \typesettree}
```

Now all that remains to be defined of the first pass is the construction of the `\vbox` containing the root text. The difficulty here is convincing `TEX` to make the `\vbox` only as wide as the widest line of text, rather than the width of the entire page. One solution is to put the text in an `\halign`, with `\crcr` implicit at the end of each line. The `\iftreetext` test is used to tell whether we are still inside the `\halign` and `\vbox`, so that `\etext` can tell whether it should do anything. It is globally false, but within the `\vbox` it gets set to true.

```
\newif\iftreetext\treetextfalse           % Whether still aligning text
\def\boxtree{\hbox\bgroup                  % Start outer box of tree or subtree
  \baselineskip 2.5ex                      % Narrow line spacing slightly
  \tabskip Opt                             % No spurious glue in alignment
  \vbox\bgroup                             % Start inner text \vbox
  \treetexttrue                            % Remember for \etext
  \let\par\crcr \obeylines                 % New line breaks without explicit \cr
  \halign\bgroup##\hfil\cr               % Start alignment with simple template
\def\etext{\iftreetext                    % Are we still in inner text \vbox?
  \crcr\egroup \egroup \fi}              % Yes, end alignment and box
```



**Pass 2: Calculation of widths at each level**

Here we calculate a list of the dimensions of each level of the tree; that is, the widths of the widest `\vbox` at each level. To do this, we need to be able to maintain lists of things. Since these are dimensions rather than boxes of text it will be most convenient to use macros like the ones given on page 378 of the *TEXbook*. However, it turns out that we need to set our lists both locally to some grouping and also globally. Therefore, we will use a stripped down version of those list macros that can handle the `\global` flag. To implement this feature, we have to lose some others; the chief losses are that the contents of the lists will be macro-expanded by various of the list manipulation macros, and that we can't use redefinitions of `\` to perform some operation on the whole list.

To initialize a control sequence to the empty list, we do `\let\curname\nil`. Then to add an element to the start of the list we do `\cons{tokens}\curname`, and to remove that element we do `\cdr\curname`. Both `\cons` and `\cdr` can be prefixed with `\global`. The first element on the list can be expanded into the token stream by doing `\car\curname`. There is no error checking, so giving `\car` or `\cdr` the empty list will cause mysterious errors later on. Because of the macro expansion performed by `\cons` and `\cdr`, the token used to separate list elements expands to itself, and unlike the *TEXbook* macros cannot be redefined to do anything useful.

```

\def\cons#1#2{\edef#2{\xmark #1#2}} % Add something to start of list.
\def\car#1{\expandafter\docar#1\docar} % Take first element of list
\def\docar\xmark#1\xmark#2\docar{#1} % ..by ignoring rest in expansion.
\def\cdr#1{\expandafter\docdr#1\docdr#1}% Similarly, drop first element.
\def\docdr\xmark#1\xmark#2\docdr#3{\def#3{\xmark #2}}
\def\xmark{\noexpand\xmark} % List separator expands to self.
\def\nil{\xmark} % Empty list is just a separator.

```

We calculate the level widths by calling `\setsizes` on the tree; it will in turn call itself recursively for each of its subtrees. The tree being sized will be in `\box0`, which is used as scratch in this macro, and the list of widths already found for this level and below will be in `\treesizes` (initially `\nil`). When the macro exits, `\treesizes` will be updated with the widths found in the various levels of the given tree or subtree. A new `\dimen`, `\treewidth`, is used within the macro to remember the previous maximum width at the level of the tree's root.

```

\def\settreesizes{\setbox0=\copy\treebox \global\let\treesizes\nil \setsizes}
\newdimen\treewidth % Width of this part of the tree.
\def\setsizes{\setbox0=\hbox\bgroup % Get a horiz list as a workspace.
  \unhbox0\unskip % Take tree, unpack it into horiz list.
  \inittreewidth % Get old width at this level.
  \sizedsubtrees % Recurse through all subtrees.
  \sizelevel % Now set width from remaining \vbox.
  \egroup} % All done, finish our \hbox.

```

The first thing `\setsizes` does is to find out what the previous maximum at this level was, and store it in `\treewidth`. If `\treesizes` is `\nil`, we haven't seen anything this deep in the tree before, so the previous size is zero. Otherwise, it is `\car\treesizes`, and we also do `\cdr\treesizes` to prepare for later recursive calls to `\setsizes`.

```

\def\inittreewidth{\ifx\treesizes\nil % If this is the first at this level
  \treewidth=0pt % ..then we have no previous max width.
\else \treewidth=\car\treesizes % Otherwise take old max level width
  \global\cdr\treesizes % ..and advance level width storage
\fi} % ..in preparation for next level.

```

At this point, we have a horizontal list (the `\hbox` in `\setsizes`) containing the `\vbox` for the text at the root of this subtree, followed by the `\hboxes` for all of its sub-subtrees. We loop pulling boxes from the end of the list with `\lastbox` until we find the text `\vbox`, calling `\setsizes` recursively for each `\hbox` we come across.

```

\def\sizesubtrees{\loop           % For each box in horiz list (subtree)
  \setbox0=\lastbox \unskip      % ..pull it off list and flush glue.
  \ifhbox0 \setsizes             % If hbox, it's a subtree - recurse
  \repeat}                       % ..and loop; end loop on tree text.

```

Now all that remains to do in this call to `\setsizes` is to update `\treewidth` if the text box, which can be found in `\box0`, is wider than the previous maximum. Then we add the (possibly updated) value of `\treewidth` as a text string back onto the head of `\treesizes`.

```

\def\sizelevel{\ifdim\treewidth<\wd0 % If greater than previous maximum
  \treewidth=\wd0 \fi                % Then set max to new high
\global\cons{\the\treewidth}\treesizes}% In either case, put back on list

```

### Pass 3: Typesetting the tree

We are now ready to begin actual construction of the tree. This is done by calling `\maketree`, which like `\setsizes` calls itself recursively for all subtrees. It adds an `\hbox` containing the given subtree (which it finds in `\treebox`) to the current horizontal list; thus the outer call to `\maketree` sends the whole tree into `TEX`'s output stream.

```

\newdimen\treeheight             % Height of this part of the tree.
\newif\ifleaf                    % Tree has no subtrees (is a leaf).
\newif\ifbotsub                  % Bottom subtree of parent.
\newif\iftopsub                  % Top subtree of parent.
\def\typesetree{\medskip \maketree \medskip} % Make whole tree with spacing.
\def\maketree{\hbox{\treewidth=\car\treesizes % Get width at this level.
  \cdr\treesizes                 % Set up width list for recursion.
  \makesubtreebox\unskip         % Set \treebox to text, make subtrees.
  \ifleaf \makeleaf             % No subtrees, add glue.
  \else \makeparent \fi}}       % Have subtrees, stick them at right.

```

After `\maketree` sets `\treewidth` from `\treesizes`, it calls `\makesubtreebox`. This opens up the horizontal list describing this level of the tree, and checks whether it has subtrees. If not, `\ifleaf` is set to true; otherwise it is set to false, and `\box0` is set to contain a `\vbox` of them with their connecting rules, except for the horizontal rule leading from the tree text to the subtrees. In any case `\treebox` is set to the `\vbox` containing the tree text.

```

{\catcode'\@=11                 % Be able to use \voidb@x.
\gdef\makesubtreebox{\unhbox\treebox % Open up tree or subtree.
  \unskip\global\setbox\treebox\lastbox % Pick up very last box.
  \ifvbox\treebox               % If we're already at the \vbox
    \global\leaftrue \let\next\relax % ..then this is a leaf.
  \else \botsubtrue              % Otherwise, we have subtrees.
    \setbox0\box\voidb@x        % Init stack of processed subs
    \botsubtrue \let\next\makesubtree % ..and call \maketree on them.
  \fi \next}}                   % Finish up for whichever it was.

```

If this tree or subtree itself has subtrees, we need to put them and their connections in `\box0` for `\makesubtreebox`. We come here with the bottom subtree in `\treebox`, the remaining list of subtrees in the current horizontal list, and the already processed subtrees stacked in `\box0`. The `\ifbotsub` test will be true for the first call, that is, the bottom subtree. Here we process the subtree in `\treebox`. If this is the top subtree, we return; otherwise we tail recurse to process the remaining subtrees. We use `\box1` as another scratch variable; this is safe because the `\hbox` in `\maketree` puts us inside a group, and also because we are not changing the output list.

```

\def\makesubtree{\setbox1\maketree % Call \maketree on this subtree.
  \unskip\global\setbox\treebox\lastbox % Pick up box before it.
  \treeheight=\ht1 % Get height of subtree we made
  \advance\treeheight 2ex % Add some room around the edges
  \ifhbox\treebox \topsubfalse % If picked up box is a \vbox,
    \else \topsubtrue \fi % ..this is the top, otherwise not.
  \addsubtreebox % Stack subtree with the rest.
  \iftopsub \global\leaffalse % If top, remember not a leaf
    \let\next\relax \else % ..(after recursion), set return.
    \botsubfalse \let\next\makesubtree % Otherwise, we have more subtrees.
  \fi \next} % Do tail recursion or return.

```

Each subtree in the list is processed and stacked in `\box0`; this is done by `\addsubtreebox`, which calls `\subtreebox` to add connecting rules to the subtree in `\box1`, and appends to that the old contents of `\box0`. The vertical connecting rules in the tree are made with tall narrow `\hrules` rather than a more simple calls to `\vrule`, because they are made inside a `\vbox`.

```

\def\addsubtreebox{\setbox0=\vbox{\subtreebox\unvbox0}}
\def\subtreebox{\hbox\bgroup % Start \hbox of tree and lines
  \vbox to \treeheight\bgroup % Start \vbox for vertical rules.
    \ifbotsub \iftopsub \vfil % If both bottom and top subtree
      \hrule width 0.4pt % ..vertical rule is just a dot.
    \else \treehalfrule \fi \vfil % Bottom gets half-height rule.
    \else \iftopsub \vfil \treehalfrule % Top gets half-height the other way.
    \else \hrule width 0.4pt height \treeheight \fi\fi % Middle, full height.
  \egroup % Finish vertical rule \vbox.
  \treectrbox{\hrule width 1em}\hskip 0.2em\treectrbox{\box1}\egroup}

```

The last line of the definition of `\subtreebox` calls `\treectrbox` twice: once for the horizontal connecting rule, and once for the subtree box itself. This macro centers its argument in a `\vbox` the height of this subtree and surrounding space. We also define here `\treehalfrule`, the macro called to make an `\hrule` half the height of the subtree (with half the height of the horizontal connection added to make the corners come out square).

```

\def\treectrbox#1{\vbox to \treeheight{\vfil #1\vfil}}
\def\treehalfrule{\dimen0=\treeheight % Get total height.
  \divide\dimen0 2\advance\dimen0 0.2pt % Divide by two, add half horiz height.
  \hrule width 0.4pt height \dimen0} % Make a vertical rule that high.

```

That completes `\makesubtree`. If this subtree has no sub-subtrees under it, `\maketree` will now run `\makeleaf`; this merely adds the tree text to the `\hbox` opened in `\maketree`. Otherwise we call `\makeparent` to attach the sub-subtrees and connecting rules to the text at the root of the subtree.

```

\def\makeleaf{\box\treebox} % Add leaf box to horiz list.
\def\makeparent{\ifdim\ht\treebox>\ht0 % If text is higher than subtrees
  \treeheight=\ht\treebox % ..use that height.
  \else \treeheight=\ht0 \fi % Otherwise use height of subtrees.
  \advance\treewidth-\wd\treebox % Take remainder of level width
  \advance\treewidth 1em % ..after accounting for text and glue.
  \treectrbox{\box\treebox}\hskip 0.2em % Add text, space before connection.
  \treectrbox{\hrule width \treewidth}\treectrbox{\box0}} % Add \hrule, subs.

```

## RECIPES AND FRACTIONS

Donald E. Knuth

Pages 233, 236, and 237 of *The T<sub>E</sub>Xbook* contain examples of alignment based on excerpts from the well-known book *Mastering the Art of French Cooking*, by Julia Child et al. Several of the measurements in those examples involve fractions like  $\frac{1}{2}$ , and this caused unpleasant interference between adjacent lines when I first looked at proofs of the tables for pages 236–237. The fractions on different lines didn't actually touch each other, but they came close enough to be visually disturbing. That's why I increased the distance between baselines by 2pt in those examples.

Since writing *The T<sub>E</sub>Xbook* I've had several opportunities to typeset recipes for various social occasions, and I learned something that I should have realized long ago: The typographer's  $\frac{1}{2}$  works better than a mathematician's  $\frac{1}{2}$  in such texts. Hence I recently added a new exercise 11.6 to *The T<sub>E</sub>Xbook*, explaining how to make fractions like  $\frac{1}{2}$  when they aren't already present in a font; I also changed the examples on pages 233, 236, and

237 so that they would use this idea. (See the current errata list for details.)

Last December, my wife and I made a keepsake for the Associates of the Stanford University Libraries: My grandmother's recipe for "Stollen" was used to bake some of the goodies at their annual Christmas Tea, and we provided copies of the recipe as an example of digital typography. I was glad to find that the members of this group were pleased not only by the delicious cake; they also liked the quality of the typesetting, even though it was done by a computer! If I hadn't used an appropriate style of fractions, I'm sure we wouldn't have gotten such a favorable response.

Here is a copy of the keepsake, and the T<sub>E</sub>X code that produced it, in case the reader is interested in seeing another small but complete example of T<sub>E</sub>X usage (based only on the plain T<sub>E</sub>X macros). The final output was printed in such a way that we could easily fold the two pages, making essentially a 3" × 5" card that could be filed with other recipes. Since the recipe is so short, I didn't use any fancy macros to do the double-column formatting of the list of ingredients.

```

\hsize=4.5in
\vsizer=2.3in
\nopagenumbers

\font\ninerm=amr9 % someday this will be "cmr9" instead!

\def\frac#1/#2{\leavevmode\kern.1em
  \raise.5ex\hbox{\the\scriptfont0 #1}\kern-.1em
  /\kern-.15em\lower.25ex\hbox{\the\scriptfont0 #2}}

\parskip=3pt
\parindent=0pt

{\bf Christmas Stollen}
\medskip

\tabskip=20pt plus 1fil
\halign{&#\hfil\cr
1 pint milk, scalded and cooled&
\frac1/2 teaspoon nutmeg\cr
1 ounce compressed or dry yeast&
1\frac1/2 teaspoons salt\cr
1 cup butter&
8 cups flour\cr
1 cup sugar&
1 pound mixed candied fruit\cr
4 eggs&

```

```

\frac3{4} pound candied cherries\cr
grated rind of 1 lemon&
1 cup nuts\cr
}

```

```
\smallskip
```

Dissolve yeast in scalded, cooled milk. Add 1 cup of the flour. Let it rise  $\frac{1}{2}$  hour.

Cream butter and sugar. Beat in eggs, one at a time. Stir in yeast mixture. Add lemon rind, nutmeg and salt. Dredge the fruit in a little flour to keep the pieces from sticking together. Add the rest of the flour to the dough, and finally stir in the fruit and nuts. Knead the dough until smooth. Put in a warm place in a covered bowl and let rise until doubled in bulk. \ (Because the fruit makes the dough heavy, it may take two or three hours to rise.) \ Divide the dough into three parts. Roll each portion out to about 1 inch thick, then fold over in thirds to form a long, loaf shape. Place on a greased cookie sheet, cover and let rise until doubled again. Bake at  $325^{\circ}\text{F}$ . for 45 minutes.

Stollen is traditionally frosted with thin powdered-sugar-and-butter icing. Decorate each loaf with red and green candied cherries.

Vary the fruit and nuts to suit your taste. You may use cherries alone, mixed fruit, and/or dates; almonds, pecans, walnuts, or no nuts at all.

```
\medskip \ninerm \baselineskip=11pt
```

This is the recipe that was used each Christmas by Don's grandmother, Pauline Ehlert Bohning, Cleveland, Ohio. Don's mother, Louise Bohning Knuth, still makes over 20 loaves each year, and when we were married she passed the recipe on to us. We hope you enjoy it.

```
\vskip-\baselineskip
```

```
\rightline{Don and Jill Knuth, Stanford, 1984}
```

```
\eject
```

```
\end
```

### Christmas Stollen

1 pint milk, scalded and cooled	1/2 teaspoon nutmeg
1 ounce compressed or dry yeast	1 1/2 teaspoons salt
1 cup butter	8 cups flour
1 cup sugar	1 pound mixed candied fruit
4 eggs	3/4 pound candied cherries
grated rind of 1 lemon	1 cup nuts

Dissolve yeast in scalded, cooled milk. Add 1 cup of the flour. Let it rise 1/2 hour. Cream butter and sugar. Beat in eggs, one at a time. Stir in yeast mixture. Add lemon rind, nutmeg and salt. Dredge the fruit in a little flour to keep the pieces from sticking together. Add the rest of the flour to the dough, and finally stir in the fruit and nuts. Knead the dough until smooth. Put in a warm place in a covered bowl and let rise

until doubled in bulk. (Because the fruit makes the dough heavy, it may take two or three hours to rise.) Divide the dough into three parts. Roll each portion out to about 1 inch thick, then fold over in thirds to form a long, loaf shape. Place on a greased cookie sheet, cover and let rise until doubled again. Bake at 325° F. for 45 minutes.

Stollen is traditionally frosted with thin powdered-sugar-and-butter icing. Decorate each loaf with red and green candied cherries.

Vary the fruit and nuts to suit your taste. You may use cherries alone, mixed fruit, and/or dates; almonds, pecans, walnuts, or no nuts at all.

This is the recipe that was used each Christmas by Don's grandmother, Pauline Ehlert Bohning, Cleveland, Ohio. Don's mother, Louise Bohning Knuth, still makes over 20 loaves each year, and when we were married she passed the recipe on to us. We hope you enjoy it. Don and Jill Knuth, Stanford, 1984

## News & Announcements

### TEX: The Program

Addison-Wesley announces the publication of *TEX: The Program*, Donald Knuth's "almost final draft" of the program listing for TEX. It is printed in an 8 1/2 x 11 inch, 3-hole punched format. The price is \$24.95. To order, call or write Gail Goodell, Educational & Professional Technologies Division, Addison-Wesley Publishing Co., Inc., Reading, MA 01867, Phone: 617-944-3700.

**Calendar**

Plans are still underway to conduct Beginning T<sub>E</sub>X courses and other T<sub>E</sub>X-related courses at various locations in the U.S. A complete listing of locations, dates and courses to be offered will not be available until early April, at which time copies will be mailed to all TUG members.

**1985**

- April 1 T<sub>E</sub>X Users Group Annual Meeting, Stanford University: Deadline for members to contact Program Committee for presentations. (See below.)
- May 16-17 T<sub>E</sub>X for Scientific Documentation, Como, Italy. (See below.)
- \* May 15 TUGboat Volume 6, No. 2: Deadline for submission of manuscripts.

**Stanford University, Stanford, Calif.:**

- Aug. 5-9 Beginning T<sub>E</sub>X (for individuals with technical backgrounds)
- Aug. 6-7 METAFONT Short Course
- Aug. 8-9 Short Course: Internal Workings of T<sub>E</sub>X82
- \* Aug. 12-14 T<sub>E</sub>X Users Group Annual Meeting
- \* Aug. 15-16 T<sub>E</sub>X Short Course (subject to be announced)

\* \* \* \* \*

- Sep. 30 TUGboat Volume 6, No. 3: Deadline for submission of manuscripts (tentative).

**1986**

- Jan. 9-12 T<sub>E</sub>X exhibits, American Mathematical Society Annual Meeting, New Orleans, La.
- \* Change from previously announced dates.

**T<sub>E</sub>X for Scientific Documentation Conference**

As previously announced (TUGboat Vol. 5 (1984), No. 2, page 147), a European Conference will be held in Italy, May 16-17, 1985. The location has been changed, however, from Varenna to Colomo (Como), which is a few miles southwest of Varenna. A very full program is planned, with 18 papers being presented. For a copy of the program, registration information, etc., contact Dario Lucarella, Istituto di Cibernetica, Università di Milano, Via Viotti 3/5, 20133 Milano, Italy, or the TUG office, 401-272-9500, ext. 232. The registration fee will be approximately \$50.00 (U.S.). The Conference Proceedings will be published shortly thereafter; publication details will be announced later.

**TUG Annual Meeting****August 14-16, 1985, Stanford University**

The Program Committee, in an effort to better organize and improve the Annual Meeting overall, has requested that members wishing to give talks at this year's meeting submit a brief description of the subject to be covered along with an estimate of the amount of time needed. The deadline was April 1, which is now upon us. It's still not too late; however, it is imperative that this information be submitted immediately to the Program Committee in order for them to have time to put together a comprehensive and informative program.

Also, this is the time to let the Committee know if there are particular sessions that you would like to have included in the program. At the 1984 Meeting, it was suggested that sessions might be scheduled on parallel tracks. Please make your wishes known. Tell us the kinds of tracks of interest to you plus specific topics which you'd like to see included.

Call Arlene Azzarello today about the presentation you would like to make or suggestions for topics you would like to have presented. She can be reached at 415-327-1700. Her mailing address is:

Arlene E. Azzarello  
I. P. Sharp Associates, Inc.  
220 California Avenue - Suite 100  
Palo Alto, CA 94306-1683, U.S.A.

**T<sub>E</sub>X and METAFONT reports available from the  
Stanford Computer Science Department**

The reports listed below are available from  
Publications, Computer Science Department,  
Stanford University, Stanford, CA 94305  
(415) 497-4776

California residents add 6.5% to total charge.  
M = microfiche only, \$2.00 + tax

CS	824	Tung	"LCCD - a language for Chinese character design"	M
CS	828	Knuth, Plass	"Breaking paragraphs into lines"	M
CS	848	Tang	"On the problem of inputting Chinese characters"	M
CS	870	Plass	"Optimal pagination techniques for automatic typesetting systems"	M
CS	886	Knuth	"Concept of a Meta-font"	2.35
CS	901	Fuchs, Knuth	"Optimal font caching"	2.60
CS	914	Gu, Hobby	"Using string matching to compress Chinese characters"	2.50
CS	960	Zabala	"Interacting with graphic objects"	4.95
CS	965	Ghosh	"An approach to type design and text composition in Indian scripts"	6.45
CS	966	Bigelow, Ghosh	"A formal approach to lettershape description for type design"	3.60
CS	974	Gu, Hobby	"A Chinese meta-font"	2.70
CS	977	Liang	"Word Hy-phen-a-tion by com-put-er"	4.70
CS	978	Knuth	"Lessons learned from METAFONT"	3.05
CS	980	Knuth	"The WEB system of structured documentation"	8.10
CS	981	Knuth	"Literate Programming"	2.45
CS	985	Samuel	"First Grade T <sub>E</sub> X"	3.95
CS	1013	Désarménien	"How to Run T <sub>E</sub> X in French"	3.30
CS	1027	Knuth	"A Torture Test for T <sub>E</sub> X"	9.25

The following preliminary report is not yet available from Stanford  
but can be obtained from

Maria Code, 1379 Sydney Drive,  
Sunnyvale CA 94087 (415) 735-8006

Knuth "T<sub>E</sub>Xware"

Available in a technical bookstore or from Maria Code:

Knuth *The T<sub>E</sub>Xbook*, published by Addison-Wesley (1984)  
Knuth *T<sub>E</sub>X: The Program*, published by Addison-Wesley (1985)



## TEXTSET, T<sub>E</sub>X, and the IBM PC

Two implementations of T<sub>E</sub>X for the IBM PC have been announced—MicroT<sub>E</sub>X™ by Addison-Wesley and PC T<sub>E</sub>X™ by Personal T<sub>E</sub>X. Both implementations produce standard DVI files. To insure quality printing with both of these implementations, Textset's professionally supported DVI-to-printer driver programs are being ported to the IBM PC. Use MicroT<sub>E</sub>X™ or PC T<sub>E</sub>X™ to process your T<sub>E</sub>X documents and use Textset's device drivers to print pages on QMS Lasergrafix, Imagen, and Xerox 9700 printers, or Autologic APS-5 series phototypesetters.

Also, don't forget to send your IBM PC disks containing T<sub>E</sub>X DVI files to Textset for fast, inexpensive typesetting on an Autologic APS-5 phototypesetter.

## TEXTSET, T<sub>E</sub>X, and PostScript

Textset and Adobe Systems, Inc. are pleased to announce that our companies are cooperating to develop PostScript representations for the Computer Modern T<sub>E</sub>X fonts. Adobe has also designated Textset as their official T<sub>E</sub>X-to-PostScript support group. PostScript is the standardized printer language used in the newly released Apple Laserwriter and QMS Lasergrafix 1200A. Other major printer and typesetter manufacturers are also developing PostScript printers.

**TEXTSET, Inc.**

416 Fourth Street, P.O. Box 7993

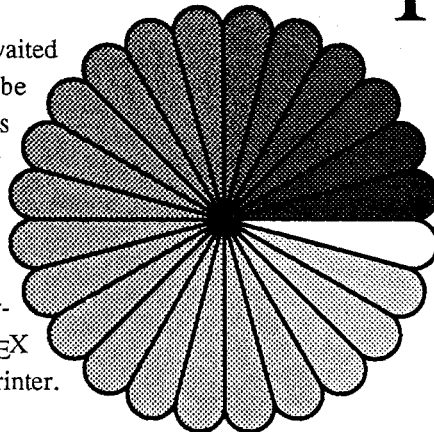
Ann Arbor, Michigan 48107

**(313) 996-3566**

This ad was produced on a QMS Lasergrafix 800 printer. T<sub>E</sub>X is a trademark of the American Mathematical Society.

# PostScript

T<sub>E</sub>X and PostScript are pleased to announce the long awaited marriage of text and graphics. Textset, Inc. and Adobe Systems, Inc. have also announced that our companies are cooperating to develop PostScript representations for the Computer Modern fonts. Adobe has designated Textset as their official T<sub>E</sub>X-to-PostScript support group. PostScript is the standardized printer language used in the newly released Apple LaserWriter and QMS Lasergrafix 1200A printers. This ad was formatted using T<sub>E</sub>X and PostScript and printed on an Apple LaserWriter printer.



**TEXTSET, Inc.**

416 Fourth Street, P.O. Box 7993

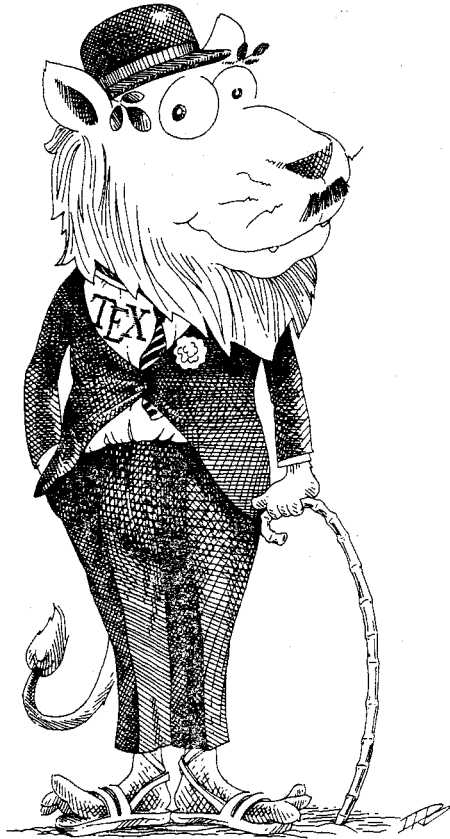
Ann Arbor, Michigan 48107

**(313) 996-3566**

Addison-Wesley is Proud to Present...

# MicroTEX™

MicroTEX is a full microcomputer implementation of Donald Knuth's technical text processing system, TEX, developed for the IBM PC/XT by David Fuchs. MicroTEX contains the power of TEX implementations on larger machines. Both input and output files from MicroTEX are fully compatible with all implementations of TEX82. The MicroTEX package also includes a driver for local output on a dot matrix printer.



### MicroTEX System Requirements:

- \* IBM PC/XT or IBM PC with a hard disk
- \* 512K Memory

### MicroTEX supports these printers:

- \* Epson RX-, FX-, MX-, 80 and 100 Printers
- \* IBM Dot Matrix and Graphics Printers

MicroTEX is available to you now in our preliminary version for only \$495.00 for the complete package.

Use the form below to order now! The preliminary version of MicroTEX may be updated to the published version at a nominal cost.

A leading international publisher of quality scientific books and software, including *The TEXbook*, Addison-Wesley will shortly announce another important publication, *L<sup>A</sup>TEX: A Document Preparation System*, by Leslie Lamport. In addition, TEX source code listings are now available from Addison-Wesley in *TEX: The Program*.

TEX is a trademark of the American Mathematical Society, MicroTEX is a trademark of Addison-Wesley Publishing Company, Inc., IBM is a registered trademark of International Business Machines, Inc., and Epson is a registered trademark of Epson, Inc.

Order Form

35178

Yes, please send me \_\_\_ copy(ies) of MicroTEX at \$495.00 each.

My check is enclosed

Visa  MasterCard (Interbank#

)

American Express

Card No. \_\_\_\_\_ Exp. Date \_\_\_\_\_

Signature \_\_\_\_\_

Please bill my company. (Your company order form and P.O. number must accompany your order. Thank you.)

Name \_\_\_\_\_ Title \_\_\_\_\_

Firm/Institution \_\_\_\_\_ Department \_\_\_\_\_

Street \_\_\_\_\_ City/State/Zip \_\_\_\_\_

Telephone(\_\_\_\_) \_\_\_\_\_ Ext. \_\_\_\_\_

I am interested in MicroTEX, but would like more information.

The best time to phone me is \_\_\_\_\_ A.M. \_\_\_\_\_ P.M.

Please keep me informed about all MicroTEX developments and related products.

Send your order to:



Addison-Wesley Publishing Company  
Educational and Professional Technologies Division  
Reading, MA. 01867  
Telephone: (617) 944-3700 Ext. 2677

## TEX LECTURES ON TAPE

Videotapes of lectures from short courses and some lectures presented at TEX Users Group meetings are available for purchase in all video formats; for lease, only the 3/4" U-Matic format is available.

For additional information, contact Ray Goucher, TEX Users Group, P. O. Box 9506, Providence, RI 02940-9506, U.S.A., (401) 272-9500, ext. 232.

### TEX Users Group Meeting, Stanford University, August 15-17, 1984

The following sessions were videotaped and are available for lease (\$25 each hour or fraction thereof) or purchase (\$50 each hour or fraction thereof):

Don Knuth: Update on TEX82 and general Q & A (1:30); Jacques Désarménien: Running TEX in a French environment (fonts, hyphenation, typography) (1:00); Don Knuth et al.: Macro writing (3:00) and What's new in Metafont and typography (1:30); Leslie Lamport: Improvements in L<sup>A</sup>T<sub>E</sub>X macros (0:30); David Fuchs, John Gourlay and Peter Sih: Output devices and drivers (1:00); Leslie Lamport and Richard Southall: Interfacing conventional design practice with TEX and Metafont: discussion (1:30); Georgia Tobin: Font design using Metafont & discussion (1:00); and Site Coordinators' progress reports - DG MV 8000, Prime 750, HP 3000, IBM Group, UNIX, and VAX (VMS) (1:30).

### Book Design Utilizing TEX - Richard Southall and Leslie Lamport

At the August 1984 TUG meeting a short course, "First Principles of Typographic Design for Document Production", consisting of approximately 11 hours of lectures, was presented. The course establishes some basic principles for the typographic design of simple text. The application of these principles to the design of documents, and the implementation of the resulting designs with TEX, was discussed. Topics include: typographic structures in text; L<sup>A</sup>T<sub>E</sub>X structures; graphic conventions in text; the document designer's tools; making text readable; designing headings; implementing text and heading designs; designing pages; implementing page layouts; list design and other issues. The lease/purchase prices are listed below.

### Introduction to the Internal Workings of TEX82 - Donald Knuth

At the July 1982 TUG meeting a short course of 12 one-hour lectures, "Introduction to TEX82", was presented on the internal workings of TEX82. The WEB source of TEX82 was used as a reference. A reading knowledge of PASCAL was strongly recommended as a prerequisite.

The principal goal of the course was to make the participant familiar with the anatomy of the TEX82 system, so that it will be clear how to make system dependent changes necessary to install it as an effective production tool. For a complete description of this course, see TUGboat Volume 5 (1984), No. 1, Cover 3. The lease/purchase prices are listed below.

### Introduction to AMS-TEX82 - Michael Spivak

At the July 1983 TUG meeting an Introductory AMS-TEX82 Users Course for secretaries and technical typists, consisting of 11 hours of lectures, was presented by Michael Spivak, author of *The Joy of TEX*, which served as the text for this course. For a complete description of this course, see TUGboat Volume 5 (1984), No. 1, Cover 3. The lease/purchase prices are listed below.

	Lease	Purchase*
List	\$375/month	\$750
Institutional member, non-educational	325/month	650
Educational institution	300/month	600

\*The purchase price for VHS and Beta formats is \$100 less in each category.

## Professional T<sub>E</sub>Xware

Textset's T<sub>E</sub>X Package and T<sub>E</sub>X Preview software are available for the Sun Workstation and the Apollo DOMAIN Workstation. The T<sub>E</sub>X Package is an optimized version of T<sub>E</sub>X for Apollo and Sun Workstations. T<sub>E</sub>X Preview is a T<sub>E</sub>X page previewer that uses the complete standard distribution T<sub>E</sub>X fonts to display average size pages on the bit-map screen at the rate of one to three seconds per page depending on the computer system configuration.

Textset's T<sub>E</sub>X DVI-to-printer driver programs produce pages quickly and accurately on QMS Lasergrafix, Imagen, and Xerox 9700 printers. DVIAPS, Textset's T<sub>E</sub>X DVI-to-printer driver for Autologic APS-5 series phototypesetters, is marketed in cooperation with Autologic Inc. Autologic offers the complete Computer Modern T<sub>E</sub>X fonts as a package to its clients. DVIAPS can also produce pages on Autologic's Bit Blaster printers.

All of Textset's software is fully compatible with the most up-to-date versions of T<sub>E</sub>X, and we have a professional commitment to maintain that compatibility.

For organizations producing a high volume of documents and requiring professional T<sub>E</sub>X support, Textset can install complete turnkey T<sub>E</sub>X systems with screen preview facilities, proof quality output on low- and high-end laser printers, and final typesetting on Autologic APS-5 series phototypesetters. Textset also provides T<sub>E</sub>Xpert applications development and support and programmer level consultation.

Textset software has been installed on the following operating systems: UNIX (SUN), Aegis (Apollo), IBM VM/CMS, IBM MVS, TOPS20, VAX/VMS, HP9000, and MTS. *All of Textset's DVI-to-printer driver programs are being ported to the IBM PC.*

## TEXTSET, T<sub>E</sub>X, and the Future

Textset has a strong commitment to research and development. New T<sub>E</sub>X products and services are under development already and scheduled for release later this year. Among these are:

- A T<sub>E</sub>X-to-PostScript driver program. PostScript is the standard printer language developed by Adobe Systems, Inc. The newly released Apple Laserwriter and QMS Lasergrafix 1200A are PostScript printers. PostScript will make it possible to integrate graphics into T<sub>E</sub>X documents.
- A semi-interactive version of T<sub>E</sub>X Preview for high-performance workstations like SUN and Apollo.
- *Getting Started with T<sub>E</sub>X*, a beginners manual that explains how to use T<sub>E</sub>X for standard document types. An easy-to-use macro package will be distributed with the book.

Textset offers courses in beginning T<sub>E</sub>X and Advanced Macro writing, some arranged through TUG. The first offering was presented at the Los Alamos National Laboratory in early March.

Remember to send your T<sub>E</sub>X DVI files on tape or disk to Textset for fast, inexpensive typesetting on an Autologic APS-5 phototypesetter.

### TEXTSET, Inc.

416 Fourth Street, P.O. Box 7993

Ann Arbor, Michigan 48107

**(313) 996-3566**

*Textset has provided professional T<sub>E</sub>Xnical software and support since 1982. We are always pleased to give professional references from our international, installed customer base.*

This ad was produced on a QMS Lasergrafix 800 printer. T<sub>E</sub>X is a trademark of the American Mathematical Society.



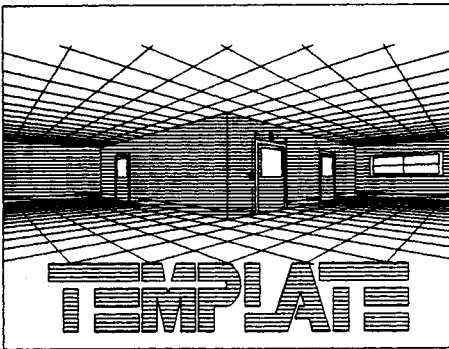
SYSTEMS INC.

## TALARIS TALKS T<sub>E</sub>X

We talk your language. Talaris Systems Inc. offers T<sub>E</sub>X users full T<sub>E</sub>X support through its line of T<sub>E</sub>X-compatible text and graphics software, its family of high quality laser printers, and its extensive library of METAFONT-generated T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X fonts.

### T<sub>E</sub>X AND GRAPHICS

We offer QDRIVE™, a T<sub>E</sub>X-compatible text and graphics merging tool. QDRIVE processes T<sub>E</sub>X-formatted text pages and merges them with automatically scaled and positioned embedded graphics, like so:



and/or overlaid graphics (see our logo above) for letterheads, borders, forms, grids and "rubber stamp" emulation. QDRIVE is available on VAX/VMS, DEC-10 and DEC-20.

### LASER JOY

Talaris offers three fast, affordable, high quality laser printers: the Talaris 2400 (24 pages per minute), the Talaris 1200 (12 pages per minute) and the Talaris 800 (8 pages per

minute, desk-top). All are full page bit-map, 300-dots-per-inch text and graphics laser printers. Each functions as a line printer, a plotter (both vector and raster graphics), a Diablo-style printer and a typesetter.

### T<sub>E</sub>X SUPPORT

T<sub>E</sub>Xsupport is our software kit for T<sub>E</sub>X82 on DEC-10, DEC-20 and VAX/VMS based Talaris laser printers, and for T<sub>E</sub>X82 on VAX/Berkeley 4.1 or 4.2 UNIX based Talaris laser printers. Each T<sub>E</sub>Xsupport kit comes with Q<sub>T</sub><sub>E</sub>X, our T<sub>E</sub>X82 DVI file post-processor.

T<sub>E</sub>Xsupport comes with a complete library of 239 T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X fonts (237 downloadable, 2 in ROM), spooler or symbiont enhancements, and T<sub>E</sub>X version 1.1, L<sup>A</sup>T<sub>E</sub>X and S<sub>L</sub><sub>T</sub><sub>E</sub>X at no extra charge. We also offer 4 optional T<sub>E</sub>X ROM font sets. (Our T<sub>E</sub>X fonts are completely compatible with Stanford's.)

### FONT FLASH!

Now available on the VAX/VMS: The VMS Print Server, also known as the Font Daemon. The Font Daemon provides automatic down-loadable font memory management. While your document is being printed, the Print Server dynamically loads for you any font it encounters in your text. It first checks to see if the referenced font has already been loaded, and if not, automatically loads it for you. If font memory is too full, it deletes the oldest unused font before loading the new one. (Bonus: the Font Daemon works with any text file, not just T<sub>E</sub>X text files.)

For more information on our T<sub>E</sub>X-related products, call or write us today.

## WE TALK T<sub>E</sub>X!

P.O. Box 261580, San Diego, CA 92126 (619) 587-0787

TEMPLATE is a registered trademark of Megatek Corporation.

**PC T<sub>E</sub>X™**  
**A Complete Implementation of T<sub>E</sub>X**  
**For the IBM PC/XT and AT**

Now available for your microcomputer: a *real* typesetting system, capable of producing journal quality output! Your IBM PC/XT or AT can now run T<sub>E</sub>X, the state-of-the-art typesetting program developed by PROF. DONALD E. KNUTH at Stanford University. T<sub>E</sub>X reads standard ASCII files and produces files that can be used to obtain printed output on a wide variety of devices—from dot matrix printers to laser printers to phototypesetters.

T<sub>E</sub>X is being supported as a standard language for mathematical typesetting by the *American Mathematical Society*, which has produced the *A<sub>M</sub>S-T<sub>E</sub>X* macro package. *A<sub>M</sub>S-T<sub>E</sub>X* greatly simplifies the setting of complex mathematical formulas and also allows the same computer file to be printed in the style of different journals. You can produce preprints of a paper on your own printer, while a journal can use the same file to print the version that will be published.

PC T<sub>E</sub>X is a complete T<sub>E</sub>X—with input and output files compatible with all implementations of T<sub>E</sub>X. PC T<sub>E</sub>X was developed by Lance Carnes, the first to implement T<sub>E</sub>X on a small computer (Hewlett-Packard 3000) and the editor for the “small” T<sub>E</sub>X department of the T<sub>E</sub>X Users Group Newsletter.

Minimum system requirements: IBM PC/XT or AT or work-alike with a 10M hard disk and 360K floppy. 512K memory to run T<sub>E</sub>X, 640K to run L<sup>A</sup>T<sub>E</sub>X.

- Price \$279.00. PC-DOT, the program to produce printed output on the IBM graphics printer, or on the Epson RX, MX, FX series, is priced separately at \$100.00. Prof. Knuth's T<sub>E</sub>X manual, *The T<sub>E</sub>Xbook*, is also available, for \$15.00.
- California residents, please add sales tax.
- For each PC T<sub>E</sub>X or PC T<sub>E</sub>X, PC-DOT purchase, add \$3.00 for shipping (UPS). Or add \$8.00 for UPS 2<sup>nd</sup> Day Air.
- Quantity discounts available. Also available: memory boards, previewer for use with high-resolution screens, drivers for Imagen and QMS laser printers.

Personal T<sub>E</sub>X, Inc. 20 Sunnyside, Suite H, Mill Valley, CA 94941 (415) 388-8853

T<sub>E</sub>X is a trademark of the American Mathematical Society, PC T<sub>E</sub>X is a trademark of Personal T<sub>E</sub>X, Inc., IBM is a registered trademark of International Business Machines, Inc., Epson is a registered trademark of Epson, Inc., Imagen is a trademark of Imagen, Inc., and QMS is a trademark of QMS, Inc.

**Request for Information**

The TeX Users Group maintains a database and publishes a membership list containing information about the equipment on which members' organizations plan to or have installed TeX, and about the applications for which TeX would be used.

Please answer the questions below, in particular those regarding the status of TeX and the computer(s)/operating system(s) on which it runs or is being installed. (Especially for IBM and VAX, the operating system is more relevant than the model.)

If it has not yet been done for your site, please also answer the questions about output devices on the other side of this form, obtaining information from the most knowledgeable person at your installation if necessary. If this information has already been provided by another TUG member, please indicate that member's name, and the information will be repeated. If you need more space than is provided here, feel free to use additional paper.

If your current listing is correct, you need not answer these questions again. Your cooperation is appreciated.

- Send completed form with remittance (checks, money orders, UNESCO coupons) to:

TeX Users Group  
P. O. Box 594  
Providence, Rhode Island 02901, U.S.A.

- For foreign bank transfers direct payment to the TeX Users Group, account #002-610871, at:

Rhode Island Hospital Trust National Bank  
One Hospital Trust Plaza  
Providence, Rhode Island 02903-2449, U.S.A.

- General correspondence about TUG should be addressed to:

TeX Users Group  
P. O. Box 9506  
Providence, Rhode Island 02940-9506, U.S.A.

Name: _____
Home <input type="checkbox"/> _____
Bus. <input type="checkbox"/> Address: _____
_____
_____

QTY	ITEM	AMOUNT
	1985 TUGboat Subscription/TUG Membership (Jan.-Dec.) - <b>North America</b> New (first-time): <input type="checkbox"/> \$20.00 each Renewal: <input type="checkbox"/> \$30.00; <input type="checkbox"/> \$20.00 - reduced rate if renewed before January 31, 1985	
	1985 TUGboat Subscription/TUG Membership (Jan.-Dec.) - <b>Outside North America *</b> New (first-time): <input type="checkbox"/> \$25.00 each Renewal: <input type="checkbox"/> \$35.00; <input type="checkbox"/> \$25.00 - reduced rate if renewed before January 31, 1985	
	TUGboat back issues, \$15.00** 1980 (v. 1) 1981 (v. 2) 1982 (v. 3) 1983 (v. 4) 1984 (v. 5) per issue, circle issue(s) desired: #1 #1, #2, #3 #1, #2 #1, #2 #1, #2	
	First Grade TeX: A Beginner's TeX Manual by Arthur L. Samuel - \$6.00 each	
	User's Guide to the HP TeX Macros by Susan Daniels - \$6.00 each	
	TeX and Metafont: Errata and Changes (final edition, September 1983) - \$4.00 each	
	The TeXbook: Errata and Changes (included with TUGboat) - additional copies \$3.00 each	
	TeX Lectures on Tape (Prices reduced - see cover 3, Vol. 5, No. 2)	

\* Air mail postage is included in the rates for all subscriptions and memberships outside North America.  
\*\* Discount: 5-7 copies, 10%; 8 or more, 15%

TOTAL ENCLOSED: \_\_\_\_\_  
(Prepayment in U.S. dollars required)

\* \* \* \*

**Membership List Information**

Institution (if not part of address):  
  
Title:  
Phone:  
Specific applications or reason for interest in TeX:  
  
My installation can offer the following software or technical support to TUG:

Date:  
Status of TeX:  Under consideration  
 Being installed  
 Up and running since  
Approximate number of users:  
Version of TeX:  SAIL  
Pascal:  TeX82  TeX80  
 Other (describe)

Please list high-level TeX users at your site who would not mind being contacted for information; give name, address, and telephone.

From whom obtained:  
  
Computer(s) and operating system(s):

Please answer the following questions regarding output devices used with TeX  
 unless this form has already been filled out by someone else at your site.  
 Use a separate form for each output device.

Name \_\_\_\_\_ Institution \_\_\_\_\_

A. Output device information

- Device name
- Model
- 1. Knowledgeable contact at your site
  - Name
  - Telephone
- 2. Device resolution (dots/inch)
- 3. Print speed (average feet/minute in graphics mode)
- 4. Physical size of device (height, width, depth)
- 5. Purchase price
- 6. Device type
  - photographic  electrostatic
  - impact  other (describe)
- 7. Paper feed  tractor feed
  - friction, continuous form
  - friction, sheet feed  other (describe)
- 8. Paper characteristics
  - a. Paper type required by device
    - plain  electrostatic
    - photographic  other (describe)
  - b. Special forms that can be used  none
    - preprinted one-part  multi-part
    - card stock  other (describe)
  - c. Paper dimensions (width, length)
    - maximum
    - usable
- 9. Print mode
  - Character: ( ) Ascii ( ) Other
  - Graphics  Both char/graphics
- 10. Reliability of device
  - Good  Fair  Poor
- 11. Maintenance required
  - Heavy  Medium  Light
- 12. Recommended usage level
  - Heavy  Medium  Light
- 13. Manufacturer information
  - a. Manufacturer name
    - Contact person
    - Address
    - Telephone
  - b. Delivery time
  - c. Service  Reliable  Unreliable

B. Computer to which this device is interfaced

- 1. Computer name
- 2. Model
- 3. Type of architecture \*
- 4. Operating system

C. Output device driver software

- Obtained from Stanford
- Written in-house
- Other (explain)

D. Separate interface hardware (if any) between host computer and output device (e.g. Z80)

- 1. Separate interface hardware not needed because:
  - Output device is run off-line
  - O/D contains user-programmable micro
  - Decided to drive O/D direct from host
- 2. Name of interface device (if more than one, specify for each)
- 3. Manufacturer information
  - a. Manufacturer name
    - Contact person
    - Address
    - Telephone
  - b. Delivery time
  - c. Purchase price
- 4. Modifications
  - Specified by Stanford
  - Designed/built in-house
  - Other (explain)
- 5. Software for interface device
  - Obtained from Stanford
  - Written in-house
  - Other (explain)

E. Fonts being used

- Computer Modern
- Fonts supplied by manufacturer
- Other (explain)

1. From whom were fonts obtained?

2. Are you using Metafont?  Yes  No

F. What are the strong points of your output device?

G. What are its drawbacks and how have you dealt with them?

H. Comments - overview of output device

\* If your computer is "software compatible" with another type (e.g. Amdahl with IBM 370), indicate the type here.



# AMERICAN MATHEMATICAL SOCIETY

P. O. Box 6248

Providence, RI 02940

Ordered by: \_\_\_\_\_

Mail to (if different): \_\_\_\_\_

\_\_\_\_\_  
 \_\_\_\_\_  
 \_\_\_\_\_

QTY	CODE	AUTHOR and TITLE	PRICE
	TEXBKT	Donald E. Knuth: The T <sub>E</sub> Xbook @ \$15 each	\$
	JOYTT	Michael Spivak: The Joy of T <sub>E</sub> X @ \$10 each (revised preliminary edition, 1982, with AMS-T <sub>E</sub> X82 supplement)	
		Shipping and Handling <input type="checkbox"/> Surface <input type="checkbox"/> Air	
		<b>Total due (All orders must be prepaid)</b>	<b>\$</b>

To order using VISA or MasterCard (for **book orders only**)

VISA  MasterCard, Account Number \_\_\_\_\_

Expiration Date \_\_\_\_\_ Signature \_\_\_\_\_

### Ordering, Shipping and Handling

**Books** are sent via surface mail (UPS to U.S. addresses and printed matter elsewhere) unless air delivery is requested. The shipping and handling charges for book orders are shown below.

	First Book	Each Additional	Maximum
Surface	\$2	\$1	\$ 25
Air	\$5	\$3	\$100

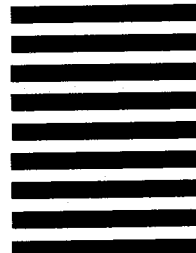
**Ordering Information:** Individuals may use **VISA** or **MasterCard** to order these books by mail or phone. For callers in the continental U.S., the Society's toll-free number, 800-556-7774, is attended from 8:00 a.m. to 4:15 p.m., Eastern Time, Monday through Friday except on holidays. Send **charge orders** to the AMS, P.O. Box 6248, Providence, RI 02940. Send orders being paid by **foreign transfer** to Rhode Island Hospital Trust National Bank, Account #000-753-111, One Hospital Trust Plaza, Providence, RI 02903, U.S.A.

Staple here

Fold here



NO POSTAGE  
NECESSARY  
IF MAILED  
IN THE  
UNITED STATES



**BUSINESS REPLY MAIL**

FIRST CLASS PERMIT NO. 554B PROVIDENCE, RI

POSTAGE WILL BE PAID BY ADDRESSEE

**AMERICAN MATHEMATICAL SOCIETY**  
Membership and Sales Department  
P. O. Box 6248  
Providence, RI 02940

Fold here

Fasten PAYMENT securely

### TeX82 Order Form

The latest official versions of TeX software and documents are available from Maria Code by special arrangement with the Computer Science Department of Stanford University.

Eight different tapes are available. The generic distribution tape contains the source of TeX82, WEB, and the latest (prototype) version of WEB METAFONT, standard test programs for TeX and METAFONT, a few "change" files, the collection of fonts in TFM format, and other miscellaneous materials; a PASCAL compiler will be required to install programs from a generic tape. The TeX distribution tapes include the AMS-TeX, LaTeX and HP TeX macro packages; other macro packages will be added as they become available. The special distribution tapes are for the indicated systems only, and should be ordered for these systems instead of a generic tape. Two tapes are PXL font collections covering various magnifications at 200/240 dots/inch and 300 dots/inch respectively.

Each tape will be a separate 1200 foot reel which you may send in advance or purchase (for the tape media) at \$10.00 each. Should you send a tape, you will receive back a different tape. Tapes may be ordered in ASCII or EBCDIC characters. You may request densities of 6250, 1600 or 800 (800 is discouraged since it is more trouble to make).

The tape price of \$82.00 for the first tape and \$62.00 for each additional tape (ordered at the same time) covers the cost of duplication, order processing, domestic postage and some of the costs at Stanford University. Extra postage is required for first class or export.

Manuals are available at the approximate cost of duplication and mailing. Prices for manuals are subject to change as revisions and additions are made. It is assumed that one set of manuals will suffice you. If you require more than two sets, please write for prices since we must ask for more money for postage and handling.

Please send a check or money order (payable on a US bank) along with your order if possible. Your purchase order will be accepted, as long as you are able to make payment within 30 days of shipment. Please check this out before sending a purchase order since many large firms seem to be unable to make prompt payment (or don't worry about it).

The order form contains a place to record the name and address of the person who will actually use the TeX tapes. This should *not* be someone in the purchasing department.

Your order will be filled with the most recent versions of software and manuals available from Stanford at the time your order is received. If you are waiting for some future release, please indicate this. Orders are normally filled within a few days. There may be periods (like short vacations) when it will take longer. You will be notified of any serious delays. If you want to inquire about your order you may call Maria Code at (408) 735-8006 between 9:30 a.m. and 2:30 p.m. West Coast time.

If you have questions regarding the implementation of TeX or the like, you must take these to Stanford University or some other friendly TeX user.

Now, please complete the order form on the reverse side.

