Out flew the web and floated wide ....
Alfred, Lord Tennyson
The Lady of Shalott

TUGBOAT

THE TEX USERS GROUP NEWSLETTER
EDITOR ROBERT WELLAND

VOLUME 3, NUMBER 2 OCTOBER 1982
PROVIDENCE RHODE ISLAND U.S.A.

ADDRESSES OF OFFICERS, AUTHORS AND OTHERS

ARMADING, Tom
Medical Center Computing Facility
University of Rochester
Rochester, NY 14627
716-275-2613

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

BENNETT, J. Michael
Regional EDP Ctr, Aarhus Univ
Ny Munkegade
8000 Aarhus C, Denmark

BERTELSEN, Erik
Regional EDP Center, Univ of Aarhus (RECAU)
Ny Munkegade
Bygning 540
DK-8000 Aarhus C, Denmark
45 6 128355; Telex: 64 754 recau dk

BLANFORD, Mark
Division 5531
Sandia National Laboratories
Albuquerque, NM 87185
505-844-8999

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

CODE, Ron
Data Processing Services
1371 Sydney Dr
Sunnyvale, CA 94087

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

FUCHS, David
Department of Computer Science
Stanford University
Stanford, CA 94305
415-497-1648

GAUTHIER, Dick
TYX Corporation
11250 Roger Bacon Dr
Suite 16
Reston, VA 22090
703-471-0233

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

GROSSO, Paul
Univ of Michigan
1075 Beal St
Ann Arbor, MI 48109

HICKEY, Thomas B.
OCLC, Inc
6565 Frantz Rd
Dublin, OH 43017
614-764-6075

JACKSON, Calvin W.
Computer Science Department
California Inst of Tech
256-80
Pasadena, CA 91125
213-356-6245

JEDRZEJEK, Peter
Autologic, Inc
1050 Rancho Conejo Blvd
Newbury Park, CA 91320
213-889-7400

KELLER, Arthur M.
Department of Computer Science
Stanford University
Stanford, CA 94305
415-497-3227

KELLERMAN, David
Oregon Software
2340 SW Canyon Rd
Portland, OR 97201
503-226-7760

KELLY, Bill
Academic Computing Center
University of Wisconsin-Madison
1210 W. Dayton Street
Madison, WI 53706
608-262-8821

KNUTH, Donald E.
Department of Computer Science
Stanford University
Stanford, CA 94305

LAMPORT, Leslie
SRI International
333 Ravenswood Ave
Menlo Park, CA 94025
415-859-3652

MacKAY, Pierre A.
Dept of Classics
Univ of Washington
Seattle, WA 98195
206-543-2266

MONG, Soo Khai
Electrocon International, Inc
611 Church Street
Ann Arbor, Michigan 48105

MORRIS, Robert
Mathematics Department
UMASS at Boston
Boston, MA 02125
617-287-1900, ext. 2545

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

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

PLASS, Susan
Polya 203
Center for Information Technology
Stanford University
Stanford, CA 94305
415-497-1322

PRICE, Lynne A.
CALMA
Research and Development
527 Lakeside Drive
Sunnyvale, CA 94086
408-245-7522

SAMUEL, Arthur L.
Computer Science Department
Stanford University
Margaret Jacks Hall 436A
Stanford, CA 94305
415-497-3330

SPIVAK, Michael
2478 Woodridge Drive
Decatur, GA 30033
404-329-0372

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

THEDFORD, Rilla J.
Mathematical Reviews
611 Church Street
P.O. Box 8604
Ann Arbor, MI 48107
313-764-7229

TRABB-PARDO, Luis
Department of Computer Science
Stanford University
Stanford, CA 94305

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

WEENING, Joe
Department of Computer Science
Stanford University
Stanford, CA 94305

WELLAND, Robert
Department of Mathematics
Northwestern University
2033 Sheridan Road
Evanston, IL 60201
312-864-2898

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

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

ZABALA, Ignacio
Department of Computer Science
Stanford University
Stanford, CA 94305

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

TUGboat, the newsletter of the TEX Users Group (TUG), is published irregularly for TUG by the American Mathematical Society, P.O. Box 6248, Providence, RI 02940. Annual dues for individual members of TUG, $15.00 for 1982 and $20.00 for 1983, include one subscription to TUGboat. Applications for membership in TUG should be addressed to the TEX Users Group, c/o American Mathematical Society, P.O. Box 1571; Annex Station, 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, should be sent to Robert Welland, Editor-in-Chief, at the address shown. 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.

OFFICIAL ANNOUNCEMENTS

TUG Meeting, March 1983, Stanford University

A meeting of the TEX Users Group is tentatively scheduled to be held at Stanford University sometime during the quarter break, March 18–27, 1983. Joey Tuttle, of I.P. Sharp Associates, Palo Alto, will be in charge of compiling the program and of local arrangements. A preliminary program, housing and registration information will be mailed in mid-January.

TUG Membership Dues and Privileges

Individual Membership

1982 dues for individual members are $15 until December 31, 1982; thereafter, 1982 issues of TUGboat (Vol. 3, Nos. 1–2) will be priced at $15 each ($30 for the complete volume). For 1983, individual dues will be $20. A reminder notice will be mailed late in November to all members who have not renewed by that time.

Membership privileges include all issues of TUGboat published during the membership (calendar) year. All new members and other persons inquiring about TUG will be sent a complimentary copy of TUGboat Vol. 1, No. 1 (1980). Members residing outside North America, upon payment of a supplementary fee of $12 per subscription or volume year, may have TUGboat air mailed to them. Lengthy macro packages, such as Max Díaz's *Fácil TEX* (Appendix A, TUGboat Vol. 2, No. 2), will be published separately in the future; details will be given on the order form.

Institutional Membership

Institutional membership dues are $200 per year for 1982 and 1983. Membership privileges include: designating up to 5 persons as individual members, special rates for participation at TUG meetings, and being listed as an institutional member on the inside cover of each issue of TUGboat.

TUGboat Schedule

The deadline for submitting items for Vol. 4, No. 1 (1983), will be May 2, 1983; the mailing date will be June 10. Contributions on magnetic tape or in camera copy form are encouraged; see the statement of editorial policy, page 3, Vol. 3, No. 1. Editorial addresses are given on the inside front cover, and a form containing instructions for submitting items on tape is bound into the back of this issue.

It is TUG's policy to keep all issues of TUGboat in print. Each member is entitled to receive all issues which appear during the membership year, as well as Vol. 1, No. 1. Domestic subscriptions are mailed third class bulk, which may take up to six weeks to reach its destination; shipments outside North America are mailed surface printed matter, unless the air mail option is elected. If you have not received an issue to which you are entitled, write to TUG at the address given on the order form for general correspondence.

TUGboat Advertising and Mailing Lists

For information about advertising rates or the purchase of TUG mailing lists, write or call TEX Users Group, Attention: Ray Goucher, c/o American Mathematical Society, P. O. Box 6248, Providence, RI 02940, (401) 272-9500.

* * * * * * * * * * *

General Delivery

* * * * * * * * * * *

THE TEX LOGO: AN IMPORTANT NOTE

At the July TUG meeting, Don Knuth made the following request concerning the TEX logo. Because many devices are unable to render the lowered "E" in the original TEX logo, an alternate form has been devised, using a lower-case "e" to retain the spirit of the original: TeX. Whenever anyone refers to TEX in print in a context which may ultimately be unable to render the old-style logo properly (e.g. a news release), the alternate form, TeX, should be used, and, if appropriate, some mention should be made that this is the logo for Don Knuth's "Tau Epsilon Chi". This is necessary to distinguish TEX from an operating system called TEX that has been developed (and registered) by Honeywell.

* * * * * * * * * * *

REPORT ON BUSINESS MEETINGS TUG SUMMER MEETING STANFORD UNIVERSITY, JULY 25–27, 1982

Susan Plass

Robert Morris has resigned as TUG secretary; Susan Plass was elected to replace him. Michael Spivak has agreed to serve as chairman until the next meeting.

A continuing question has been whether or not TUG should incorporate. This issue assumed a greater urgency when the AMS decided to terminate subsidies to TUG with the next fiscal year. Sam Whidden consulted the AMS attorney about the legal ramification of incorporation and reported that as long as TUG remains a small organization with limited finances there is no need to incorporate. The lawyer did recommend that TUG draft and adopt a set of bylaws under which to operate. A bylaws committee was formed consisting of Lance Carnes, Ray Goucher, and Susan Plass. They plan to present at the next TUG meeting a draft to be voted on by the membership.

General meeting

The following actions were taken by the membership at the general meeting:

- A general policy was set that membership fees should be set to cover the direct costs of the TUGboat issues to be published in that year. Conference fees will be set to recover both direct conference costs and other TUG indirect costs.
- TUG membership for 1983 will cost $20 including a subscription to TUGboat; two issues of TUGboat are planned for 1983.
- For 1983 we will offer an institutional membership for $200. Institutional membership entitles an organization to name up to 5 individuals to receive subscriptions to TUGboat and to be listed by name in the membership list. A separate listing of institutional members will appear in each issue of TUGboat.

Steering Committee meeting

It was decided that the Winter 1983 meeting will be held at Stanford sometime during the spring break, March 18–27.

The following issues were addressed at the Steering Committee meetings:

1) It was decided that up to $1,000 will be contributed to the cost of sending a representative to the next ANSI standards meeting on typesetting.
2) Joey Tuttle of I.P. Sharp has offered to arrange the next meeting.
3) Replacements are needed on the Steering Committee for Robert Morris and Richard Zippel, who are inactive.
4) Monte Nichols has resigned from the Finance Committee; a new member at large should be sought. The Finance Committee currently consists of the Chairman, Secretary, Treasurer, and two other members of the Steering Committee.
5) The closing date for submission to TUGboat was established as six weeks after the preceding TUG general meeting.
6) Fees for advertising in TUGboat were set at $200/page.
7) On December 31 of each year, the price for each back issue of that year's TUGboat will rise to the full membership price for that year.
8) Ray Goucher was directed to handle 1982 institutional memberships retroactively based on our policy for 1983 memberships.
9) The Finance Committee was empowered to take whatever urgent actions are necessary between meetings, with the prime directive to keep TUG out of debt.

Ray Goucher reported that news releases for this meeting appeared in *TypeWorld* and *SIAM News*; several other publications, which generally print announcements of this type, were notified, but their publication deadlines were missed. A news release for the March 1983 meeting will be prepared and distributed in the Fall.

In his Treasurer's report, Sam Whidden expressed TUG's thanks to Donald Knuth for the support he has given TUG. His donated services for the TEX82 Short Course have turned an expected $12,000 deficit into a positive balance. A volunteer willing to teach a similar course at the March meeting will be sought.

The final issue addressed by the Steering Committee in its meetings is a difficult issue and was left unresolved. TUG needs people willing to work not just on TEX, but also on the bread-and-butter problems of organization, finance, and technical direction. Without the involvement of new members in the more pragmatic facets of TUG, the organization cannot thrive. It is up to the general membership to determine whether TUG as an organization will survive.

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

Program, TUG Summer Meeting and TEX82 Short Course Stanford University, July 25–30, 1982

USERS GROUP MEETING

Business meeting
Ron Whitney – introduction to TEX and TUG for new users
Don Knuth – TEX82
TEX82 Q & A
Don Knuth – the WEB system of structured documentation
Arthur Keller – tutorial for beginning TEX users (2 sessions)
Barbara Beeton – TEX problems help session
Site Coordinators – introduction and birds-of-a-feather sessions:
- DEC 10/DEC 20 (Barry Doherty)
- IBM 370 (Susan Plass)
- small architectures (Lance Carnes)
- Univac (Bill Kelly)
- VAX/UNIX (Cal Jackson)
- VAX/VMS (Monte Nichols)

Output device manufacturers' representatives – introduction and consultation
- Florida Data (Bob Booher, Frank Price)
- Hewlett-Packard (Jim Crumly, Tom Old)
- Imagen (Les Earnest, Jan Stoeckenius)
- Symbolics (Jane Durrant, Larry Hambly, Marc LeBrun)

TEX82 and WEB user experiences
Macro tutorial for intermediate TEX users
David Fuchs – output devices and drivers
Lynne Price – macro wizards roundtable
Don Knuth – demonstration of TEX82

SHORT COURSE – INTRODUCTION TO TEX82

Reading WEB programs
Representation of strings
Data structures for boxes and glue
Representation of control sequences
Syntactic routines (TEX's eyes and mouth)
Semantic routines (TEX's stomach and intestines)
Breaking paragraphs into lines
Hyphenation
Scanning file names
Input of font metric (TFM) files
Output of device-independent (DVI) files
Initializing a TEX production program

All twelve lectures of the Short Course were videotaped, and the tapes will be made available by TUG for rental or purchase; details can be obtained from Ray Goucher, c/o American Mathematical Society, P. O. Box 6248, Providence, RI 02940, (401) 272-9500.

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

Attendees, TUG Summer Meeting and TEX82 Short Course Stanford University, July 25–30, 1982

In the following list, names of persons attending only the meeting are not specially marked; attendees at only the Short Course are starred; attendees at both the meeting and Short Course are flagged by a †.

Abramson, Fred – Signetics Corp.
† Arnson, William – McGraw-Hill, Inc.
* Atkinson, Michael – Science Center, Rockwell International
† Beeman, Roger – Boeing Aerospace Company
† Beeton, Barbara – American Mathematical Society
† Berkowitz, Marc – Adapt, Inc.
† Berry, Paul – I. P. Sharp Associates, Inc.
Besset, Didier – Physics Department, Stanford University
* Blair, John – CALMA
Blanford, Mark – Sandia National Laboratories
† Bollack, Ed – Information Handling Services
† Boyce, Jim – Hewlett-Packard Co.
† Brotsky, Daniel C.
† Brown, Heather – The University Of Kent, Canterbury
Brown, Malcolm – Center For Information Technology, Stanford University
† Buckle, Normand – Université du Québec, Montréal
Carnes, Lance
† Chaffee, Roger – Linear Accelerator Center, Stanford University
† Cherry, George – Language Automation Associates
Clark, Debbie – Intergraph
† Clouthier, Pierre – Computer Output Services Information, Inc.
† Cole, Michael – Washington State University
† Crawford, Dona – Sandia National Laboratories
† Crumly, Jim – Hewlett-Packard Co.

†Cuzzo, Clint – Hewlett-Packard Co.
†Doherty, Barry – American Mathematical Society
†Doyle, Peter G. – Dartmouth College
†Dupree, Charles – Digital Equipment Corp.
Eastman, Pat – Cadtec
Ellis, Wade – The Computer Tutors
†Elsell, Anders – Royal Institute Of Technology, Stockholm
Felippa, Carlos – Lockheed Palo Alto Research Lab
Ferguson, Michael – Université du Québec, Montréal
Ferris, Barry – Science Applications, Inc.
Fina, Pat – Massachusetts Institute of Technology
*Fisher, Glen – The Code Works
Foata, Dominique – Université de Strasbourg
Fuchs, Abe
†Fuchs, David – Department of Computer Science, Stanford University
†Gabelnick, Stephen – Argonne National Laboratory
†Gaskins, Robert – Bell-Northern Research, Inc.
Graham, Ron – American Mathematical Society
Grim, Daniel – University of Delaware
†Grosso, Paul – University of Michigan
†Guenther, Dean – Washington State University
Hage, Carl – Signetics Corp.
†Hallin, Mary Elizabeth – Bell-Northern Research, Inc.
*Harris, Kent – Consultant
†Hauck, Roger – Smithsonian Institution
Hickey, Thomas – Online Computer Library Center
†Holsinger, Gregory – McGraw-Hill, Inc.
Hsia, Doris – Computer Systems Laboratory, Stanford University
Huang, Edith – California Insitute of Technology
†Hurley, Peter – Hewlett-Packard Co., Berkshire, England
†Hutchinson, George – National Institutes of Health
†Ion, Patrick – Mathematical Reviews, American Mathematical Society
†Jackson, Calvin – California Institute of Technology
Janson, Barbara – American Mathematical Society
Jeffries, Ron – The Code Works
Jespersen, Dennis – Informatics General Corp.
†Jones, Roger – Science Typographers, Inc.
†Kado, Don – Information Handling Services
Kanerva, Dianne – Department of Computer Science, Stanford University
†Keller, Arthur – Department of Computer Science, Stanford University
†Kellerman, Dave – Oregon Software
†Kelly, William – University of Wisconsin at Madison
†Knuth, Don – Department of Computer Science, Stanford University
Kockinos, Constantin
†Kröger, Heinz – Zentralblatt für Mathematik, Berlin
Lamport, Leslie – SRI International
Larson, Richard – University of Illinois at Chicago Circle
Leino, Arthur – Linear Accelerator Center, Stanford University
Lodi, Ed – West Valley College & Computer Tutors
MacKay, Pierre – University of Washington
†Mansell, Alana – Information Handling Services
McGilton, Henry
Mohr, August
Mooney, Jim – West Virginia University
Moffat, Shannon – Center for Information Technology, Stanford University
†Moortgat, Hugo – University of Santa Clara
†Naugle, Norman – Texas A&M University
†Nodera, Takashi – Keio University, Yokohama
†O'Connor, George – Hewlett-Packard Co.
Old, Tom – Hewlett-Packard Co.
†Osmond, Carolyn – Information Handling Services
†Penny, Keith – Nuclear Division, Union Carbide
*Pettinicchio, Frank – McGill University
†Peuto, Bernard – View Tech, Inc.
†Plass, Susan – Center for Information Technology, Stanford University
*Ramshaw, Lyle – Xerox Palo Alto Research Center
Reisor, Ronald – University of Delaware
Robbins, Jeffrey – Springer-Verlag New York, Inc.
Rodgers, David – University of Michigan
Roth, David – Bell-Northern Research, Inc.
Roudabush, Glenn – CTB/McGraw-Hill, Inc.
†Roy, Richard – Stanford University/ESL, Inc.
†Ruggles, Lynn – University of Massachusetts, Amherst
†Samuel, Arthur – Department of Computer Science, Stanford University
Sandelin, Jon – Center for Information Technology, Stanford University
Savitzky, Steve – Zilog
†Schildt, Barbara – Cadtec
Schneble, Jack – McGraw-Hill, Inc.
†Senn, Mark – Purdue University
Seropian, Hasmig – Zilog
Shepherd, Gary – Sandia National Laboratories
†Spivak, Michael
Spragens, Alan – Linear Accelerator Center, Stanford University
Spurlock, Arlene – Lawrence Berkeley Laboratory, University of California
*Stoeckenius, Jan – Imagen Corp.
Sterken, Jim – University of Michigan
†Tenney, Glenn – Fantasia Systems, Inc.
†Thedford, Rilla – Mathematical Reviews, American Mathematical Society
Thomas, Arthur – ADAC Laboratories
*Thomas, Joe – Datalogics, Inc.
Thomas, Margaret – Science Applications, Inc.
†Tobin, Richard – Online Computer Library Center
Tomandl, Daniel – University of Washington
†Trabb-Pardo, Luis – Department of Computer Science, Stanford University
Tuttle, Joey – I. P. Sharp Associates, Inc.
Van Camp, Warren – Informatics General Corp.
VanVoorhis, Sandra – Data Composition
Villere, Gary – Informatics General Corp.
†Vitanye, David – Handar Associates
Wakabayashi, Nobuo – Otaru University of Commerce, Hokkaido, Japan
†Walker, Gordon – American Mathematical Society
Wendel, Whit – Addison-Wesley
Whartman, Yoni – Ben-Zvi Printing Enterprises
Whidden, Sam – American Mathematical Society
†Whitney, Ron – American Mathematical Society
Wong, Henry – Signetics Corp.
†Woodsmall, Roger L. – Intergraph Corp.

† Woolf, Bill – Mathematical Reviews, American Mathematical Society
* Wu, Sheau-Huei – Datalogics, Inc.
Wulff, Robert – QUBIX
* Yugin, John – Scan Laser Printing Ltd.
† Zabala, Ignacio – Department of Computer Science, Stanford University

* * * * * * * * * * *

AN INFORMATION INTERCHANGE FORMAT FOR TEX FILES

Pierre A. MacKay
University of Washington

In an earlier issue of this journal, there was reported a proposal made by Patrick Milligan for an information interchange standard for TEX files. I remember that there were several very strong points in that proposal, but I do not have it before me now and I will probably be repeating some parts of it unconsciously. I do remember noting at the time, however, that it was in part a proposal for a tape file information interchange standard and yet made no reference to *American National Standard X3.27-1978*, "magnetic tape labels for information interchange." Since it was clear at the July 1982 TUG meeting that the problem of mutually compatible tape file formats is still very much with us, I would like to refine Patrick Milligan's suggestions by proposing that, rather than attempting to define our own unique information interchange standard, we adopt and promote the use of the ANSI standard, which has already been adopted as a Federal Information Processing Standard (see FIPS publication No. 79. 1980, October 17. U. S. Department of Commerce. National Bureau of Standards).

The name of this standard is perhaps a bit misleading in that it might suggest a concern with only the text of ANSI standard tape labels—fixed-length 80-character records containing ASCII decimal numeric and upper-case alphabetic characters. At higher levels of implementation, however, the use of these labels imposes a certain discipline in file and record format, with the result that a reference to the upper levels of implementation is, in effect, a sufficient description of a standardized file format for character files and requires little more than the agreement to convert all binary files into BigEndian hexadecimal character notation to serve as an all-purpose information interchange convention for users of TEX.

The various levels of implementation of the magnetic tape label standard are described in *ANSI X3.27-1978*, Appendix A: "Levels of Systems," and most particularly in section A3: "Distinguishing characteristics of levels of labelling." This section is meant to serve as a guide for the thorough integration of tape label processing into the basic operating system, but it can also serve as the outline for a user-level utility program if nothing better is possible. I confess to a certain missionary zeal to convert a wider range of installations toward the provision of level 3 capacities as part of the standard operating system, in the manner, for instance, of the VAX/VMS operating system, where Systems level 3 label processing is the default for all character files. I am sadly aware, however, that this conversion will take time, and that many users will have to bargain for utility programs to supplement the present inadequacies of tape label processing as they are found on most systems. It would be no small service to computing if the TEX Users Group were to contribute to the wider acceptance and use of the most effective ANSI standards. (Incidentally, the ISO standard for magnetic tape labels is virtually identical with the ANSI standard.)

For a full understanding of the content of all labels used in a Systems level 3 tape label processing utility there is no substitute for a reference to the ANSI standard itself, which can be purchased (prepaid only, and not exactly cheap) from the American National Standards Institute, 1430 Broadway, New York, N.Y. 10018. The details of special interest for TEX users are that Systems level 3 should provide

File formats:	Single file single volume, Single file multivolume, Multifile single volume, Multifile multivolume.
Labels:	VOL1 HDR1 HDR2 EOV1 EOV2 EOF1 EOF2 (Full analysis and decoding of all required fields.)
Record formats:	Fixed-length or variable-length records (A prefixed fixed-length character count field is assumed for all variable-length records. Special terminator codes are never used.)

Any Systems level 3 operation ought also to allow for the inclusion in the prescribed order of user volume labels (UVL1 through UVL9) and user header labels (HDR3 through HDR9) together with the answering EOV and EOF labels. The standard does not require that anything be done with such labels, but it is highly desirable that a tape label processing system be able to read and bypass them. A truly courteous operating system will provide a buffer from which the user can retrieve information contained in these extra-standard labels.

The content of VOL1 and HDR1 labels is pretty generally known. They provide fields for owner I.D., File I.D., File Creation date, Version number, System I.D. and a few other more esoteric matters. Two fields are best avoided or left to a well-chosen system default. A bad setting of the expiration date field can be rather a nuisance on a finicky system, and setting any of the protection fields can prevent you from reading your own tapes, let alone anyone else's. (As an example of an unfortunate system default, the TOPS-20 tape processing utility at the University of Washington sets a Volume Protection code, which inhibits transfer of files from the DEC-20 to the VAX. I have not yet discovered whether that setting can be avoided.)

The most interesting label for our purposes is the HDR2 label, since this provides a properly designed tape label processing system with all the necessary information for deblocking a tape. Among these fields is the record type field, which contains (level 3) either an ASCII upper-case 'F' for fixed-length records, or an ASCII upper-case 'D' for variable-length records. A D-type record is preceded by a four-character count field indicating the full length of that record in eight-bit bytes or tape-frames. The count is expressed in ASCII decimal digits, right-justified, and includes the length of the count field itself. Thus, an 80-character card-image may appear on tape as an 84-byte D-type record, with the first 4 bytes containing the ASCII characters '0084'. (Since the VAX/VMS operating system makes the convenient assumption that all character files are to be recorded as D-type records, a file of card-images often appears in this form. You have to specify fixed-length records to avoid the slight overhead cost of the decimal count field.)

In addition to the record type field, there is the maximum record length field and the maximum block length field, which together allow the operating system to set up its deblocking buffers efficiently. When all these fields are properly filled, there is no need to send along the usual sheaf of papers describing the tape's deblocking factors with each tape file. All the required information is contained in the labels, and a simple reference to ANSI X3.27-1978, Systems level 3, will indicate that this is so. We will, of course, need to set some reasonable limit on both block length and record length. I should think that a 2048-byte tape blocking buffer ought to be within the capacity of all operating systems that can read 9-track tape at all; it results in a quite efficient use of the tape, and conforms with ISO practice. The VAX/VMS operating system limits character records to a maximum length of 255, which, in a D-type record is 259, including the character count. This ought probably to be adopted as an absolute maximum, and perhaps a smaller maximum record length would prove more convenient on some systems. At any rate, we can easily come to some agreement about this and then get to work convincing systems programmers, with the authoritative sounding initials FIPS behind us.

The special problem that Patrick Milligan addressed was the treatment of binary files, and for these I would strongly support the use of hexadecimal coding. Communications protocols tend to take an arbitrary view of the "parity bit" in eight bit transmissions, and they can do weird things with such special characters as ASCII NUL, with or without parity. But all protocols that we are likely to be dealing with will allow the transmission of the 48-character ASCII subset which includes the ten decimal digits and the uppercase alphabetics.

I have been using BigEndian hexadecimal coding for six months now. It may be twice as slow as uncoded binaries, but it works, and it works more consistently than any alternative. For my own purpose I have found it useful to establish a very strict format for binaries. I use an 80-character fixed-length record which may be of particular interest to TEX-on-IBM users. Each 32-bit BigEndian quantity occupies a 10-column card-image field, left justified, with two trailing columns of fill. The fill columns might even be used for a simple check-sum, but I have never yet had a missed-bit error that would make this seem necessary. This is a convention which works as well over a DECNet as on magnetic tape. By choosing the time of day rather carefully—usually the hours between 2:00 A.M. and 6:00 A.M.—I have been able to send the entire DVI file resulting from the application of TEX to WEAVE.TEX across a DECNet from a 36-bit DEC 2060 to a 32-bit VAX. There are undoubtedly faster ways to achieve the same end, but I suspect that the TEX Users Group will be best served by adopting the slow but sure protocols of hexadecimal coding as the basic information interchange format for DVI, TFM and Font Raster files.

INTRODUCTION TO TEX AND TUG FOR NEW USERS

Ron Whitney

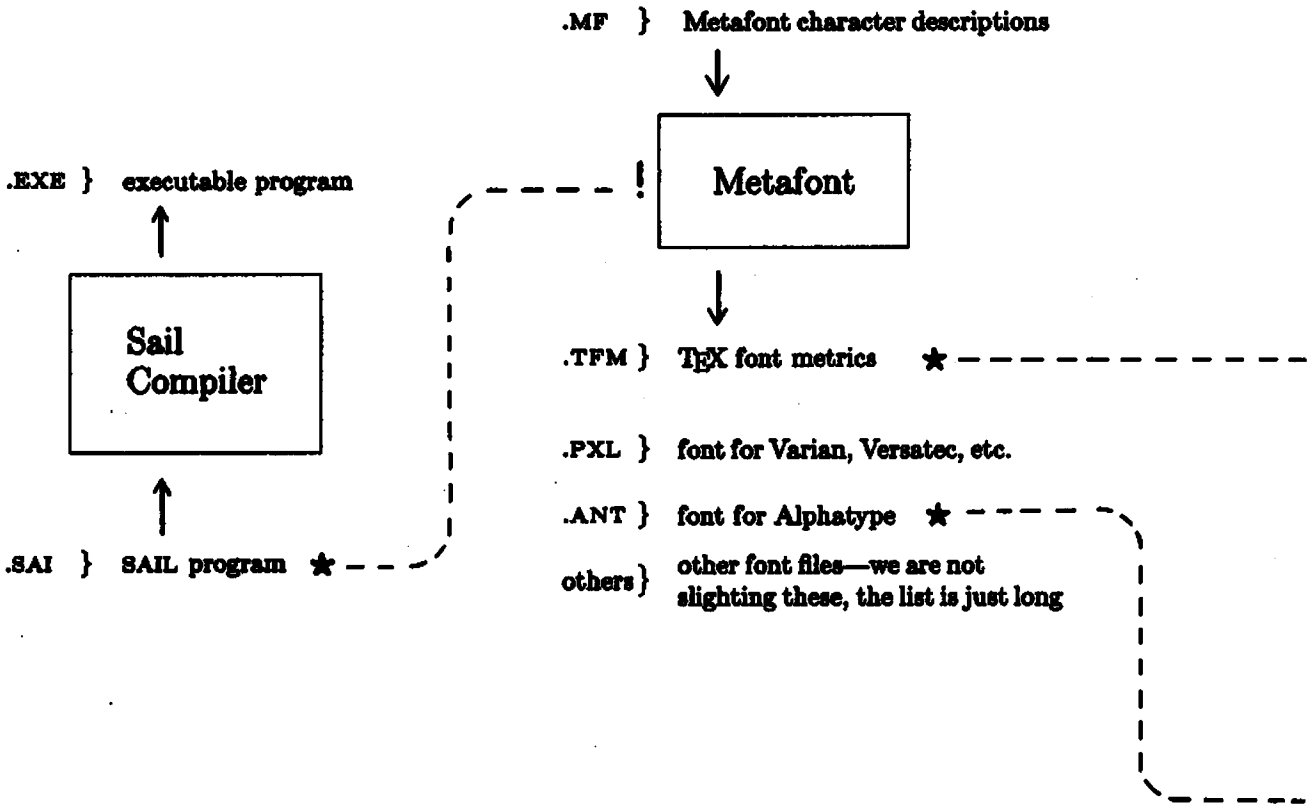
The schematic diagram on the following pages gives a picture of the files and programs involved in the TEX typesetting system. The names used are the standard extensions seen on DEC10's and 20's, but they are sufficiently mnemonic to be representative of all systems. Movement along arrows shows the general flow of files through programs. Particular files and programs within one flow may be the result of particular input from another, and these connections are indicated by paths (- - - -) from input (*) to output (!). A prospective user should be aware that this scheme is somewhat different than the pre-TEX82 setup and that the METAFONT side will change in the next year or so when that program is written in WEB. Otherwise, the following generalities should be of some help in discerning what one needs to start a system and what jargon will help in asking questions.

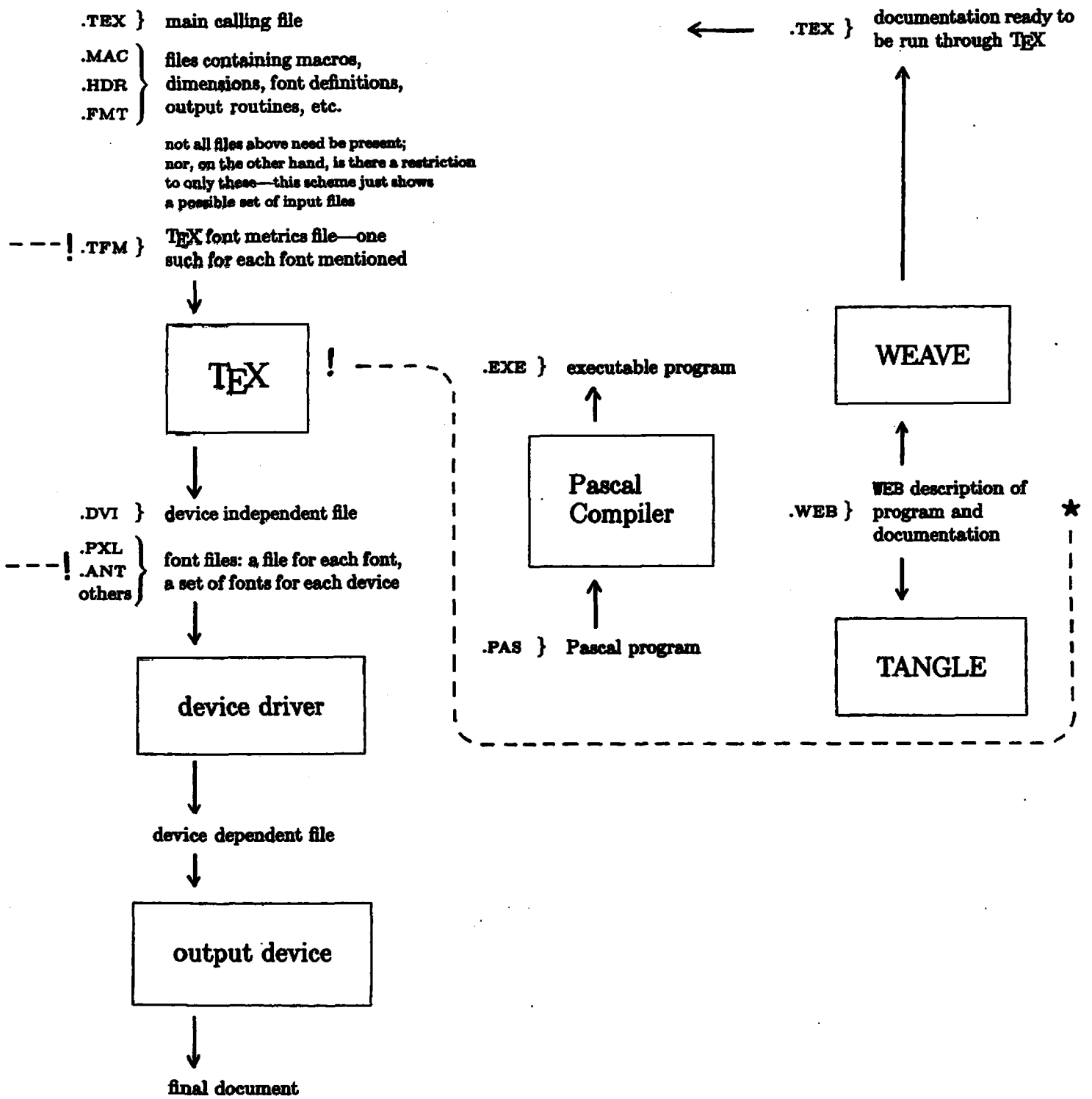
The column containing the TEX program comprises a minimal set of programs and files needed to produce a typeset document. The files used as input to TEX contain the text to be set as well as the instructions which determine where characters are placed on a page. The most comprehensive reference for the TEX language is, of course, the TEX manual (*TEX and METAFONT*, Digital Press, 1979; the TEX82 manual is in preparation and will be published by Addison-Wesley sometime soon). Raw TEX provides complete control over the final page at the possible expense of excruciating detail in specification. If the user wishes to follow the style of another author (either to cut down on one's own work or to produce a paper with a certain format), it is possible to encode that style in "macros" whose definitions can be input from .MAC, .HDR, and .FMT files. Each macro may call many TEX primitives, so a well constructed set of ancillary macro, header and format files will create an environment where (typographically) complicated papers can be put out fairly easily. We will not go into the differences traditionally associated with the "kinds" of input files (as with other specifics, one should consult issues of TUGboat and the TEX manual for help), but simply note that this is an area which deserves great attention once TEX is up and running. It is also worth noting in this context that AMS-TEX and other macro packages are *not* alternative forms of TEX, but are auxiliary files input with other matter to be run under TEX.

When a proper collection of statements is fed to TEX, the latter produces a .DVI (device independent) file. One of the advantages of TEX is its ability to transcend hardware peculiarities. A consequence of this is that if authors set their own papers with TEX on an available proofing device and send all their input files to a publishing house which runs TEX, they can be assured that the publisher's typesetters will break lines and pages in exactly the same way. This could significantly reduce both editing time and exasperating communication. TUGboat has already taken several .TEX files from authors and reproduced them with a minimum of editorial fiddling.

In order to achieve "device independence", TEX describes the set page in an absolute way: characters are only boxes of a certain height, depth, and width, and distances between characters are given in terms of printer's points (72.27 to the inch). Given this invisible ink description of a page, it is left to each typesetter to fill in the details of the raster pattern or whatever else specifies a character. Thus, each site must construct, or otherwise obtain, a program which combines the instructions in the .DVI file with information about the local fonts to produce a "painted" page for the output device. Typically, this point in the sequence has involved the most work for implementers simply because a programmer must digest something about both TEX's output and the internal workings of the printing device in order to write the driver. It may also be desirable to drive the output device with a microprocessor separate from the machine which runs TEX. Needless to say, an effort at bringing TEX up will be greatly aided by finding a site where TEX is running with the same operating system and output device. The TUG membership list contains information on architectures and output devices currently supporting TEX.

Fonts are also an important consideration. Even if an output device comes with reams of beautiful characters, users will need to make sure that these fonts can be connected to TEX. In order to construct the .DVI file, TEX needs to know certain information about the characters it is setting. Specifically, it needs to know the width, height above the baseline, depth below the baseline and ligature and kerning information for each symbol. This is exactly the information in the .TFM (for TEX Font Metrics) files created by METAFONT. Currently the easiest way to get started is to use fonts generated by METAFONT since .TFM's are available and the fonts can be made, in some sense, device independent. Fonts for two devices of different resolution


.MF } Metafont character descriptions
Metafont
.EXE } executable program
Sail Compiler
.TFM } TEX font metrics
.PXL } font for Varian, Versatec, etc.
.ANT } font for Alphatype
.SAI } SAIL program
others } other font files—we are not slighting these, the list is just long


.TEX } main calling file
.MAC
.HDR
.FMT
files containing macros, dimensions, font definitions, output routines, etc.
not all files above need be present; nor, on the other hand, is there a restriction to only these—this scheme just shows a possible set of input files
.TFM } TEX font metrics file—one such for each font mentioned
TEX
.DVI } device independent file
.PXL
.ANT
others
font files: a file for each font, a set of fonts for each device
device driver
device dependent file
output device
final document
.EXE } executable program
Pascal Compiler
.PAS } Pascal program
.TEX } documentation ready to be run through TEX
WEAVE
.WEB } WEB description of program and documentation
TANGLE

require two separate runs of METAFONT, but, if the source files have been properly constructed, identical .TFM's are produced. TEX will view such a pair of fonts as identical because it looks only at the .TFM file, not the font files. The drawback so far with METAFONT fonts has been the lack of a wide range of styles and sizes. If non-METAFONT fonts are to be used, .TFM's will have to be constructed by hand. Access to the innards of the fonts will also have to be obtained to write the device driver and not all typesetting companies are excited about making such information available. As TEX becomes more widely available and typesetters see a growing market for their machines, it is anticipated that more of them will open their font libraries to TEX users.

The discussion so far has encompassed the TEX-METAFONT system before 1982. Some out-of-date points have not been discussed here (such as the difference between .TFX and .TFM files), but, for those curious about or in need of such things, we suggest reading back issues of TUGboat.

TEX82 resembles TEX in its basic box and glue approach to page make-up, but improves upon its sibling by adding new features to the language and more space to store information. Documentation on the changes and a full description of the new language can be obtained in TUGboat and the new TEX manual when it appears. In a certain way, however, the younger TEX is hardly first kin to the older: TEX82 is written in the new WEB language whereas the original TEX was an exercise in SAIL code. WEB is a system of structured documentation which combines the features of both a programming language (in this case, Pascal) and a document formatter (in this case, TEX). Writing a program in WEB, the programmer is able to display and expose the structures of his or her algorithms in ways which are not possible with commented Pascal code. A .WEB file generates both a running Pascal program (passing a .WEB file through the program TANGLE produces a .PAS file of properly written Pascal code) and a nicely formatted description of whatever structures the programmer thinks are essential to the understanding of that program (passing a .WEB file through the program WEAVE produces a properly written .TEX file which can be run through TEX). The "essential structures" need not be Pascal subroutines and the overall approach may be top-down, bottom-up, or some combination. Since the same source produces both program and documentation, there are fewer occasions when the two are mismatched, and reliable, clear programs are easier to write. Of course, the first major project written in WEB was TEX82, but the uses are much more general.

Finally, a note on reference where to seek help. *TEX and METAFONT* co Knuth's 1978 Gibbs Lecture as well as the TE METAFONT manuals. It can be obtained from

Digital Press
Dept. AMS
Educational Services
Digital Equipment Corporation
12A Esquire Road
North Billerica, MA 01862

The new TEX manual will contain a c plete description of the TEX language as well a the WOVEN and TEX'd output of the WEB sources f the program. This manual will be published by Addison-Wesley and should be available early in 1983. Knuth's next project is to be the WEB version of METAFONT and this will also be accompanied by a new manual. The third volume of Knuth's series on typesetting and type design will contain a description of the work he has done on the Computer Modern family of fonts. Additional papers and research articles on the algorithms involved in TEX can be obtained from:

Computer Science Department
Stanford University
Palo Alto, CA 94305
attention: Dawn Yolton

To obtain information about WEB and a tape containing TANGLE, WEAVE and other related programs, write to Professor Arthur Samuel at Stanford University. In fact, Arthur is the man to contact for many of your Stanford needs, be they fonts or programs. Contact one of us here at the AMS for information about *AMS*-TEX and AMS fonts, and consult your TUGboats for more names and resources. Finally, please share your experiences and ideas about anything connected with TEX or METAFONT by writing to TUGboat. The community welcomes your comments and suggestions.

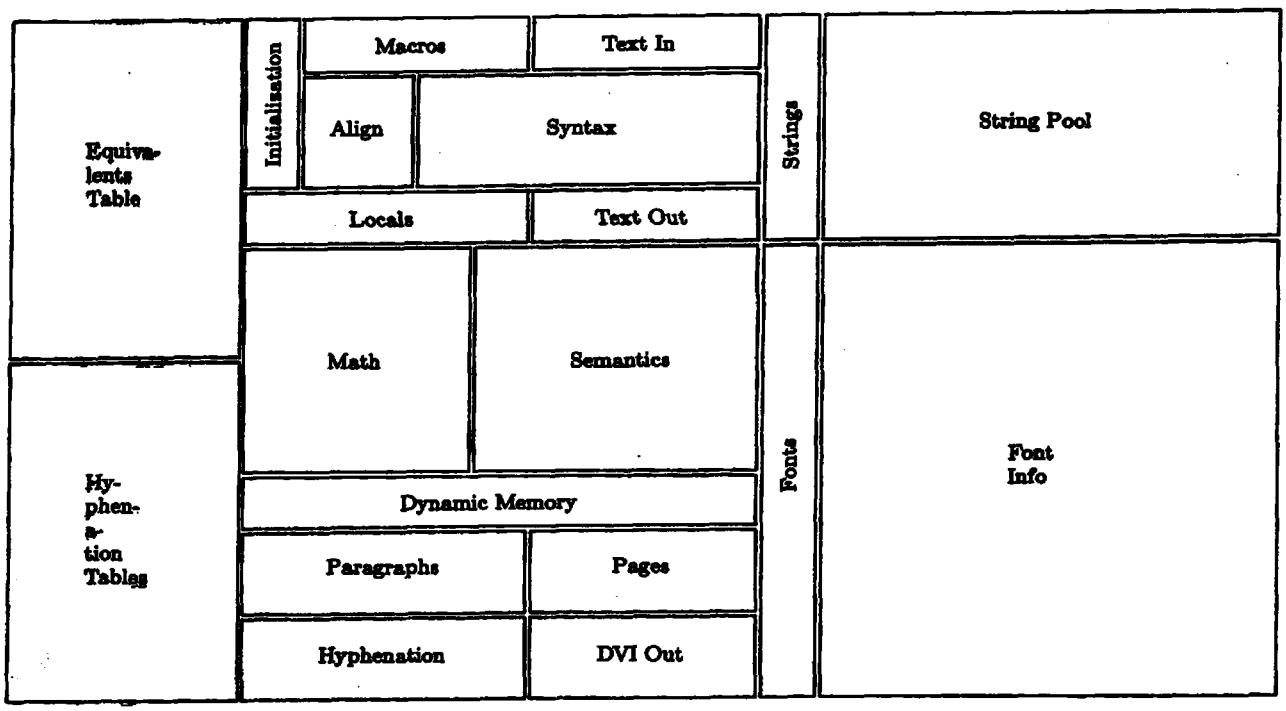
* * * * * * * * * * *

Software

* * * * * * * * * * *

TEX82 MEMORY STRUCTURE

Editor's note: The following diagram shows the relative amount of memory required by TEX82 for various tables and work areas. It has been redrawn (not using TEX, although that would be a good problem) from a slide that Don Knuth prepared for the lecture on data structures at the TEX82 Short Course held in conjunction with the July TUG meeting.



≈ 1K bytes

Memory represented in this diagram is approximately 260K bytes, plus (not shown) 100K of dynamic memory. On a DEC 20, TEX82 occupies total memory of between 350K and 600K bytes, excluding Pascal run-time requirements.

* * * * * * * * * * *

Editor's note: David Fuchs has provided the following copy describing the format of .DVI files, including some changes made since the July TUG meeting. A summary of the changes appears in David's "News from Stanford" article, which begins on page 20.

1. Device-independent file format. The most important output produced by a run of TEX is the "device independent" (DVI) file that specifies where characters and rules are to appear on printed pages. The form of these files was designed by David R. Fuchs in 1979. Almost any reasonable device can be driven by a program that takes DVI files as input, and dozens of such DVI-to-whatever programs have been written. Thus, it is possible to print the output of TEX on many different kinds of equipment, using TEX as a device-independent "front end."

A DVI file is a stream of 8-bit bytes, which 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 '*set_rule*' command has two parameters, each of which is four bytes long. Parameters are usually regarded as nonnegative integers; but four-byte-long parameters, and shorter parameters that denote distances, can be either positive or negative. Such parameters are given in two's complement notation. For example, a two-byte-long distance parameter has a value between -2^{15} and $2^{15}-1$.

A DVI file consists of a "preamble," followed by a sequence of one or more "pages," followed by a "postamble." The preamble is simply a *pre* command, with its parameters that define the dimensions used in the file; this must come first. Each "page" consists of a *bop* command, followed by any number of other commands that tell where characters are to be placed on a physical page, followed by an *eop* command. The pages appear in the order that TEX generated them. If we ignore *nop* commands and *fnt_def* commands (which are allowed between any two commands in the file), each *eop* command is immediately followed by a *bop* command, or by a *post* command; in the latter case, there are no more pages in the file, and the remaining bytes form the postamble. Further details about the postamble will be explained later.

Some parameters in DVI commands are "pointers." These are four-byte quantities that give the location number of some other byte in the file; the first byte is number 0, then comes number 1, and so on. For example, one of the parameters of a *bop* command points to the previous *bop*; this makes it feasible to read the pages in backwards order, in case the results are being directed to a device that stacks its output face up. Suppose the preamble of a DVI file occupies bytes 0 to 99. Now if the first page occupies bytes 100 to 999, say, and if the second page occupies bytes 1000 to 1999, then the *bop* that starts in byte 1000 points to 100 and the *bop* that starts in byte 2000 points to 1000. (The very first *bop*, i.e., the one that starts in byte 100, has a pointer of -1.)

2. The DVI format is intended to be both compact and easily interpreted by a machine. Compactness is achieved by making most of the information implicit instead of explicit; when a DVI-reading program reads the commands for a page, it keeps track of several quantities: (a) The current font f is an integer; this value is changed only by *fnt* and *fnt_num* commands. (b) The current position on the page is given by two numbers called the horizontal and vertical coordinates, h and v. Both coordinates are zero at the upper left corner of the page; moving to the right corresponds to increasing the horizontal coordinate, and moving down corresponds to increasing the vertical coordinate. Thus, the coordinates are essentially Cartesian, except that vertical directions are flipped; the Cartesian version of (h, v) would be $(h, -v)$. (c) The current spacing amounts are given by four numbers w, x, y, and z, where w and x are used for horizontal spacing and where y and z are used for vertical spacing. (d) There is a stack containing (h, v, w, x, y, z) values; the DVI commands *push* and *pop* are used to change the current level of operation. Note that the current font f is not pushed and popped; the stack contains only information about positioning.

The values of h, v, w, x, y, and z are signed integers having up to 32 bits, including the sign. Since they represent physical distances, there is a small unit of measurement such that increasing h by 1 means moving a certain tiny distance to the right. The actual unit of measurement is variable, as explained below; TEX sets things up so that its DVI output is in sp units, i.e., scaled points, in agreement with all the *scaled* dimensions in TEX's data structures.

8. Here is list of all the commands that may appear in a DVI file. With each command we give its symbolic name (e.g., *bop*), its opcode byte (e.g., 129), and its parameters (if any). The parameters are followed by a bracketed number telling how many bytes they occupy; for example, '*p*[4]' means that parameter *p* is four bytes long.

set_char_0 0. Typeset character number 0 from font *f* such that the reference point of the character is at (h, v). Then increase h by the width of that character. Note that a character may have zero or negative width, so one cannot be sure that h will advance after this command; but h usually does increase.

set_char_1 through *set_char_127* (opcodes 1 to 127). Do the operations of *set_char_0*, but use the appropriate character number instead of character 0.

set1 128 *c*[1]. Same as *set_char_0*, except that character number *c* is typeset. TEX82 uses this command for characters in the range $128 \leq c < 256$.

set2 129 *c*[2]. Same as *set1*, except that *c* is two bytes long, so it is in the range $0 \leq c < 65536$. TEX82 never uses this command, but it should come in handy for extensions of TEX that deal with oriental languages.

set3 130 *c*[3]. Same as *set1*, except that *c* is three bytes long, so it can be as large as $2^{24} - 1$. Not even the Chinese language has this many characters, but this command might prove useful in some yet unforeseen extension.

set4 131 *c*[4]. Same as *set1*, except that *c* is four bytes long. Imagine that.

set_rule 132 *a*[4] *b*[4]. Typeset a solid black rectangle of height *a* and width *b*, with its bottom left corner at (h, v). Then set $h \leftarrow h + b$. If either $a \leq 0$ or $b \leq 0$, nothing should be typeset. Note that if $b < 0$, the value of h will decrease even though nothing else happens. See below for details about how to typeset rules so that consistency with METAFONT is guaranteed.

put1 133 *c*[1]. Typeset character number *c* from font *f* such that the reference point of the character is at (h, v). (The 'put' commands are exactly like the 'set' commands, except that they simply put out a character or a rule without moving the reference point afterwards.)

put2 134 *c*[2]. Same as *set2*, except that h is not changed.

put3 135 *c*[3]. Same as *set3*, except that h is not changed.

put4 136 *c*[4]. Same as *set4*, except that h is not changed.

put_rule 137 *a*[4] *b*[4]. Same as *set_rule*, except that h is not changed.

nop 138. No operation, do nothing. Any number of *nop*'s may occur between DVI commands, but a *nop* cannot be inserted between a command and its parameters or between two parameters.

bop 139 c_0[4] c_1[4] ... c_9[4] *p*[4]. Beginning of a page: Set $(h, v, w, x, y, z) \leftarrow (0, 0, 0, 0, 0, 0)$ and set the stack empty. Set the current font *f* to an undefined value. The ten c_i parameters hold the values of \count0 ... \count9 in TEX at the time \shipout was invoked for this page; they can be used to identify pages, if a user wants to print only part of a DVI file. The parameter *p* points to the previous *bop* command in the file, where the first *bop* has $p = -1$.

eop 140. End of page: Print what you have read since the previous *bop*. At this point the stack should be empty. (The DVI-reading programs that drive most output devices will have kept a buffer of the material that appears on the page that has just ended. This material is largely, but not entirely, in order by v coordinate and (for fixed v) by h coordinate; so it usually needs to be sorted into some order that is appropriate for the device in question.)

push 141. Push the current values of (h, v, w, x, y, z) onto the top of the stack; do not change any of these values. Note that *f* is not pushed.

pop 142. Pop the top six values off of the stack and assign them respectively to (h, v, w, x, y, z). The number of pops should never exceed the number of pushes, since it would be highly embarrassing if the stack were empty at the time of a *pop* command.

right1 143 *b*[1]. Set $h \leftarrow h + b$, i.e., move right *b* units. The parameter is a signed number in two's complement notation, $-128 \leq b < 128$; if $b < 0$, the reference point actually moves left.

right2 144 *b*[2]. Same as *right1*, except that b is a two-byte quantity in the range $-32768 \le b < 32768$.

right3 145 *b*[3]. Same as *right1*, except that b is a three-byte quantity in the range $-2^{23} \le b < 2^{23}$.

right4 146 *b*[4]. Same as *right1*, except that b is a four-byte quantity in the range $-2^{31} \le b < 2^{31}$.

w0 147. Set $h \leftarrow h + w$; i.e., move right w units. With luck, this parameterless command will usually suffice, because the same kind of motion will occur several times in succession; the following commands explain how w gets particular values.

w1 148 *b*[1]. Set $w \leftarrow b$ and $h \leftarrow h + b$. The value of b is a signed quantity in two's complement notation, $-128 \le b < 128$. This command changes the current w spacing and moves right by b.

w2 149 *b*[2]. Same as *w1*, but b is two bytes long, $-32768 \le b < 32768$.

w3 150 *b*[3]. Same as *w1*, but b is three bytes long, $-2^{23} \le b < 2^{23}$.

w4 151 *b*[4]. Same as *w1*, but b is four bytes long, $-2^{31} \le b < 2^{31}$.

x0 152. Set $h \leftarrow h + x$; i.e., move right x units. The '*x*' commands are like the '*w*' commands except that they involve x instead of w.

x1 153 *b*[1]. Set $x \leftarrow b$ and $h \leftarrow h + b$. The value of b is a signed quantity in two's complement notation, $-128 \le b < 128$. This command changes the current x spacing and moves right by b.

x2 154 *b*[2]. Same as *x1*, but b is two bytes long, $-32768 \le b < 32768$.

x3 155 *b*[3]. Same as *x1*, but b is three bytes long, $-2^{23} \le b < 2^{23}$.

x4 156 *b*[4]. Same as *x1*, but b is four bytes long, $-2^{31} \le b < 2^{31}$.

down1 157 *a*[1]. Set $v \leftarrow v + a$, i.e., move down a units. The parameter is a signed number in two's complement notation, $-128 \le a < 128$; if $a < 0$, the reference point actually moves up.

down2 158 *a*[2]. Same as *down1*, except that a is a two-byte quantity in the range $-32768 \le a < 32768$.

down3 159 *a*[3]. Same as *down1*, except that a is a three-byte quantity in the range $-2^{23} \le a < 2^{23}$.

down4 160 *a*[4]. Same as *down1*, except that a is a four-byte quantity in the range $-2^{31} \le a < 2^{31}$.

y0 161. Set $v \leftarrow v + y$; i.e., move down y units. With luck, this parameterless command will usually suffice, because the same kind of motion will occur several times in succession; the following commands explain how y gets particular values.

y1 162 *a*[1]. Set $y \leftarrow a$ and $v \leftarrow v + a$. The value of a is a signed quantity in two's complement notation, $-128 \le a < 128$. This command changes the current y spacing and moves down by a.

y2 163 *a*[2]. Same as *y1*, but a is two bytes long, $-32768 \le a < 32768$.

y3 164 *a*[3]. Same as *y1*, but a is three bytes long, $-2^{23} \le a < 2^{23}$.

y4 165 *a*[4]. Same as *y1*, but a is four bytes long, $-2^{31} \le a < 2^{31}$.

z0 166. Set $v \leftarrow v + z$; i.e., move down z units. The '*z*' commands are like the '*y*' commands except that they involve z instead of y.

z1 167 *a*[1]. Set $z \leftarrow a$ and $v \leftarrow v + a$. The value of a is a signed quantity in two's complement notation, $-128 \le a < 128$. This command changes the current z spacing and moves down by a.

z2 168 *a*[2]. Same as *z1*, but a is two bytes long, $-32768 \le a < 32768$.

z3 169 *a*[3]. Same as *z1*, but a is three bytes long, $-2^{23} \le a < 2^{23}$.

z4 170 *a*[4]. Same as *z1*, but a is four bytes long, $-2^{31} \le a < 2^{31}$.

fnt_num_0 171. Set $f \leftarrow 0$. Font 0 must previously have been defined by a *fnt_def* instruction, as explained below.

fnt_num_1 through *fnt_num_63* (opcodes 172 to 234). Set $f \leftarrow 1, \ldots, f \leftarrow 63$, respectively.

fnt1 235 *k*[1]. Set $f \leftarrow k$. TEX82 uses this command for font numbers in the range $64 \le k < 256$.

fnt2 236 *k*[2]. Same as *fnt1*, except that k is two bytes long, so it is in the range $0 \le k < 65536$. TEX82 never generates this command, but large font numbers may prove useful for specifications of color or texture, or they may be used for special fonts that have fixed numbers in some external coding scheme.

fnt3 237 $k[3]$. Same as *fnt1*, except that k is three bytes long, so it can be as large as $2^{24}-1$.

fnt4 238 $k[4]$. Same as *fnt1*, except that k is four bytes long; this is for the really big font numbers (and for the negative ones).

xxx1 239 $k[1]$ $x[k]$. This command is undefined in general; it functions as a $(k+2)$-byte *nop* unless special DVI-reading programs are being used. TEX82 generates *xxx1* when a normal \xsend appears, setting k to the number of bytes being sent. 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 $0 \le k < 65536$.

xxx3 241 $k[3]$ $x[k]$. Like *xxx1*, but $0 \le k < 2^{24}$.

xxx4 242 $k[4]$ $x[k]$. Like *xxx1*, but k can be ridiculously large. TEX82 uses *xxx4* when sending a string of length 256 or more.

fnt_def1 243 $k[1]$ $c[4]$ $s[4]$ $d[4]$ $a[1]$ $l[1]$ $n[a+l]$. Define font k, where $0 \le k < 63$; font definitions will be explained shortly.

fnt_def2 244 $k[2]$ $c[4]$ $s[4]$ $d[4]$ $a[1]$ $l[1]$ $n[a+l]$. Define font k, where $0 \le k < 65536$.

fnt_def3 245 $k[3]$ $c[4]$ $s[4]$ $d[4]$ $a[1]$ $l[1]$ $n[a+l]$. Define font k, where $0 \le k < 2^{24}$.

fnt_def4 246 $k[4]$ $c[4]$ $s[4]$ $d[4]$ $a[1]$ $l[1]$ $n[a+l]$. Define font k, where $-2^{31} \le k < 2^{30}$.

pre 247 $i[1]$ $num[4]$ $den[4]$ $mag[4]$ $k[1]$ $x[k]$. Beginning of the preamble; this must come at the very beginning of the file. Parameters i, *num*, *den*, *mag*, k, and x are explained below.

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.

4. The preamble contains basic information about the file as a whole. As stated above, there are six parameters:

$$i[1]\ num[4]\ den[4]\ mag[4]\ k[1]\ x[k].$$

The i byte identifies DVI format; currently this byte is always set to 2. (Some day we will set $i = 3$, when DVI format makes another incompatible change—perhaps in 1992.)

The next two parameters, *num* and *den*, are positive integers that define the units of measurement; they are the numerator and denominator of a fraction by which all dimensions in the DVI file could be multiplied in order to get lengths in units of 10^{-7} meters. Since there are 72.27 points per inch and 2.54 centimeters per inch, and since TEX82 works with scaled points where there are 2^{16} sp in a point, TEX82 sets $num = 25400000$ and $den = 7227 \cdot 2^{16} = 473628672$.

The *mag* parameter is what TEX calls \mag, i.e., 1000 times the desired magnification. The actual fraction by which dimensions are multiplied is therefore $mag \cdot num/1000den$. Note that if a TEX source document does not call for any 'true' dimensions, and if you change it only by specifying a different \mag setting, the DVI file that TEX creates will be completely unchanged except for the value of *mag* in the preamble and postamble. (Fancy DVI-reading programs allow users to override the *mag* setting when a DVI file is being printed.)

Finally, k and x allow the DVI writer to include a comment, which is not interpreted further. The length of comment x is k, where $0 \le k < 256$.

define *id_byte* = 2 {identifies the kind of DVI files described here}

5. Font definitions for a given font number k contain further parameters

$$c[4]\ s[4]\ d[4]\ a[1]\ l[1]\ n[a+l].$$

The four-byte value c is the check sum that TEX found in the TFM file for this font; c should match the check sum of the font found by programs that read this DVI file.

Parameter s contains a fixed-point scale factor that is applied to the character widths in font k; font dimensions in TFM files and other font files are relative to this quantity, which is called the "at size" elsewhere in this documentation. The value of s is always positive and less than 2^{27}. It is given in the same units as the other DVI dimensions, i.e., in sp when TEX82 has made the file. Parameter d is similar to s; it is the "design size," and it is given in DVI units that have not been corrected for the magnification *mag* found in the preamble. Thus, font k is to be used at $mag \cdot s/1000d$ times its normal size.

The remaining part of a font definition gives the external name of the font, which is an ascii string of length $a + l$. The number a is the length of the "area" or directory, and l is the length of the font name itself; the standard local system font area is supposed to be used when $a = 0$. The n field contains the area in its first a bytes.

Font definitions must appear before the first use of a particular font number. Once font k is defined, it must not be defined again; however, we shall see below that font definitions appear in the postamble as well as in the pages, so in this sense each font number is defined exactly twice, if at all. Like *nop* commands, font definitions can appear before the first *bop*, or between an *eop* and a *bop*.

6. Sometimes it is desirable to make horizontal or vertical rules line up precisely with certain features in characters of a font. It is possible to guarantee the correct matching between DVI output and the characters generated by METAFONT by adhering to the following principles: (1) The METAFONT characters should be positioned so that a bottom edge or left edge that is supposed to line up with the bottom or left edge of a rule appears at the reference point, i.e., in row 0 and column 0 of the METAFONT raster. This ensures that the position of the rule will not be rounded differently when the pixel size is not a perfect multiple of the units of measurement in the DVI file. (2) A typeset rule of height $a > 0$ and width $b > 0$ should be equivalent to a METAFONT-generated character having black pixels in precisely those raster positions whose METAFONT coordinates satisfy $0 \le x < \alpha b$ and $0 \le y < \alpha a$, where α is the number of pixels per DVI unit.

7. The last page in a DVI file is followed by '*post*'; this command introduces the postamble, which summarizes important facts that TEX has accumulated about the file, making it possible to print subsets of the data with reasonable efficiency. The postamble has the form

post $p[4]$ *num*$[4]$ *den*$[4]$ *mag*$[4]$ $l[4]$ $u[4]$ $s[2]$ $t[2]$
⟨font definitions⟩
post_post $q[4]$ $i[1]$ 223's$[\ge 4]$

Here p is a pointer to the final *bop* in the file. The next three parameters, *num*, *den*, and *mag*, are duplicates of the quantities that appeared in the preamble.

Parameters l and u give respectively the height-plus-depth of the tallest page and the width of the widest page, in the same units as other dimensions of the file. These numbers might be used by a DVI-reading program to position individual "pages" on large sheets of film or paper.

Parameter s is the maximum stack depth (i.e., the largest excess of *push* commands over *pop* commands) needed to process this file. Then comes t, the total number of pages (*bop* commands) present.

The postamble continues with font definitions, which are any number of *fnt_def* commands as described above, possibly interspersed with *nop* commands. Each font number that is used in the DVI file must be defined exactly twice: Once before it is first selected by a *fnt* command, and once in the postamble.

8. The last part of the postamble, following the *post_post* byte that signifies the end of the font definitions, contains q, a pointer to the *post* command that started the postamble. An identification byte, i, comes next; this currently equals 2, 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). TEX 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 DVI file makes it feasible for DVI-reading programs to find the postamble first, on most computers, even though TEX wants to write the postamble last. Most operating systems permit random access to individual words or bytes of a file, so the DVI 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 DVI reader discovers all the information needed for typesetting the pages. Note that it is also possible to skip through the DVI file at reasonably high speed to locate a particular page, if that proves desirable. This saves a lot of time, since DVI files used in production jobs tend to be large.

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 DVI format has been designed to work most efficiently with modern operating systems. But if DVI files have to be processed under the restrictions of standard PASCAL, one can simply read them from front to back, since the necessary header information is present in the preamble and in the font definitions. (The l and u and s and t parameters, which appear only in the postamble, are "frills" that are handy but not absolutely necessary.)

OUTPUT DEVICES AND COMPUTERS

	Alpha CRS	APS-5	Canon LBP10	Comp. 8600	Fla. Data BNY	HP2680	Linotron 202	Varian	Versatec	Xerox 9700	Xerox Dover	Xerox XGP
Ethernet						Stanford					Stanford	
DEC10			Vanderbilt						Vanderbilt	Univ. Del.		
DEC20	AMS				Math Reviews		Adapt, Inc.	AMS				
IBM (VM)									SLAC			
IBM 370		Info. Handling										
Onyx C8002			TYX Corp.									
Sail												Stanford
Univac 1100				Univ. Wis.								
VAX (Unix)									Cal. Tech.			
VAX (VMS)			Argonne						Sandia			

* * * * * * * * * * *

Output Devices

* * * * * * * * * * *

OUTPUT DEVICES

Rilla J. Thedford
Mathematical Reviews

Thanks to all who responded to the questionnaire on output devices; it was genuinely appreciated. The most common question concerning output devices seems to be "Who has what device on what machine?" In response, we have compiled the above chart of *active* TEX configurations. It is far from complete. Please, if you know of or hear of a new or an existing configuration that is not on the chart, notify me or David Fuchs. We recognize that more than one site can have the same configuration, so we listed only one for each. If a list of known sites for a device or for a configuration is desired please contact me at (313) 764-7228. We will do our best to get you the information.

Future articles will describe the characteristics of various output devices, with user comments.

Editor's note: In the membership list, computer hardware and output devices are shown for individual members in the form communicated by the member. For the listing by device type, however, an effort has been made to consolidate similar devices into groups which, when possible, coincide with Site Coordinator coverage. In the case of VAX, for example, there are two significant groups, depending on operating system (VMS or UNIX). When communicating information about hardware for publication in the membership list, please take a look at the device groupings, and provide enough specifics to yield both a satisfactory individual listing and a suitable group assignment. (If your hardware does not belong to any of the existing groups, that is also useful information.)

* * * * * * * * * * *

Site Reports

* * * * * * * * * * *

NEWS FROM STANFORD

David Fuchs

TEX82 runs. We're about to put it up for general use. The test input file, called TRIP.TEX, causes 99% of the statements that make up the TEX82 program to be executed at least once. Additionally, some of us have been using TEX82, and it really works. It seems to be about the same speed as the old Sail version of TEX80, even though our compiler does not optimize at all, so with a good compiler and runtimes, it could well be faster. We're getting output on our Dover printer (thanks to Ignacio Zabala), and there's a nice new program by Joe Weening that displays pages of a DVI file on the Sail (DEC10) computer's Data-Disc displays. The TOPS-20 change file seems to be in order, and we hope that installation on VAX and IBM systems will proceed smoothly.

Professor Charles Bigelow just arrived to start the joint Art Department and Computer Science Department Program in Digital Typography. He and his students will be telling us everything that is wrong with TEX and METAFONT.

We have been in contact with Max Díaz, and he has accepted our invitation to stop by and update his Fácil TEX macro package.to work under TEX82. It may even be given new features based on the added flexibility of TEX82.

Joe Weening worked out a first draft of PLAIN.TEX (the new name for BASIC), which Professor Knuth is now polishing up. In the next few days, WEAVE will be changed slightly to work with TEX82. When this is done, we'll be making up a new distribution tape with the whole system, so now might be a good time to send Ron Code an order form for 'complete, debugged TEX82'. The new tape will include new versions of some of the Computer Modern fonts, but not all. The following release, which will come out along with the first version of the TEX82 manual, will have a whole new version of Computer Modern; you may wish to wait for the later distribution because it will also include a more widely user-tested TEX, as well as whatever change files we receive from those of you who have brought it up.

Version 2 of DVI format is now frozen. There are some changes from the documentation handed out at the recent TUG meeting. These changes allow DVI files to be read from front to back, for systems that can't do a random access to find the postamble. This also helps systems that utilize Unix-like pipes—you can pipe the output of TEX right into a DVI-to-device translator program. I extracted the relevant modules from TEX.WEB and ran WEAVE on them–the results appear elsewhere in this issue of TUGboat (our Dover printer was used for this, so please excuse the reproduction quality). The revised sections are the ones dealing with the '*preamble*', '*postamble*' and '*font def*' commands. The *preamble* contains the static job information; *font def*s appear all through the DVI file as new fonts are defined, as well as in the *postamble* as before.

As a study in the feasibility of automatic Pascal-to-C translation, I recently translated TANGLE into C by hand. The results work both on our VAXes (running Berkeley Unix) and on our SUN stations (with M68000 micro-processors running stand-alone, reading and writing files over the Ethernet). Interestingly, both versions run at about the same speed, but TANGLE is an I/O bound program, so this may not be too significant. On the other hand, TANGLE running on VAX/VMS Pascal runs a fair bit slower, probably because the character-by-character runtimes are very slow. In any case, this is a reasonably hopeful sign that TEX82 on small computers will be feasible. We're hoping to find a good Pascal compiler for the M68000 soon, but none seems to be available right now. We intend to go ahead with the C translation effort, which is another path to getting TEX82 onto micro-processors, but there are problems here as well. In particular, TANGLE.C found one bug in the M68000 C compiler which no one has yet managed to fix. Our SUNs with stand-alone Unix will be arriving soon, so we'll have more information next time.

Autologic continues to be the photo-typesetter manufacturer most sensitive to the TEX community. They have designated an official TEX person, Peter Jedrzejek, who can be reached at 213-889-7400. I suggest you give him a call and tell him you're out there. Autologic has also officially requested raster information on METAFONT fonts, e.g. the Computer Modern family, so that they can provide them to their users. We're delaying them, since an overhauled version of Computer Modern is next on Professor Knuth's to-do list.

* * * * * * * * * * *

FIXES TO KNOWN BUGS IN TEX/370

Susan Plass

About twenty-five sites have obtained TEX80 in the form distributed by the Stanford Center for Information Technology (CIT) as TEX/370. Many thanks to those patient people who have installed this version and found, fixed, and reported bugs in it. The following list represents an amalgamation of the problems those users have found and how they have been corrected.

In ASCIITBL:

Change line 34 from ` | (X'4F') to ! (X'5A')
and line 125 from ↑ (X'6A') to | (X'4F')

In SYSDEP:

Change the assignment statement
chrX['7E'*x*] :='D0'*xc*; (*¬r*c**)
in line 333 to *chrX*['7E']*x* := '5F'*xc*; (*¬¬*)

After line 1143 add the following:

```
IF eofchan(fyl)
  THEN
    BEGIN
    eoff := true;
    bufptr := 1;
    buffer[bufptr] := 13;
    goto 0;
    END;
```

In addition there are several problems with \send files. The text to be sent must have length less than the LRECL for that file. And in line 1540 change

rwritefile(fil,'DDNAME ='||trim(fname));

to *rwritefile(fil,fname);*

Change line 1584 from **ELSE IF** *fj* <= 8 to **ELSE IF** *fj*<=8

The development effort for TEX at CIT is now directed at installing TEX82. I expect to report on our progress at the next meeting, and I encourage others who are also installing TEX82 to report back on your efforts either to me before the meeting or at the IBM Birds of a Feather session at the March meeting.

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

TEX INSTALLATION AT THE UNIVERSITY OF MICHIGAN

Paul Grosso

ABSTRACT. Work has been done at the University of Michigan to convert parts of the TEX system to run under MTS. Currently we have the capability to run TEX and produce text output on a Linotron 202 typesetter in standard, bold, and italic fonts in a variety of point sizes. Further work needs to be done on input macro packages and the user interface as well as the support of other output devices.

Over the past year, the joint IBM/University of Michigan Word Processing Project has been experimenting with parts of the TEX system. Having gained much experience with it as well as some related software, the University of Michigan (UM) ordered a Pascal/VS copy of TEX from the Stanford Center for Information Technology (CIT) designed to run under IBM's MVS operating system.

The Word Processing Project (WPP) has an IBM 370/148 running VM/CMS, MVS, and MTS (Michigan Terminal System) with a network connection to UM's Amdahl 470V/8 running MTS (MTS-UM). When the tape came from Stanford (in record time—a week after we mailed out our order we had a tape in our hands), we read most of the files directly into files under VM/CMS; one file, that containing the font file partitioned data set, was written in "unloaded format", so we read it under MVS and transfered the individual font files to VM.

TEX compiled with no problems; with a few VM file definitions (to reassign files to DDNAMEs), TEXPRE ran generating TEXINTBL. A few more file definitions and TEX generated a DVI file for a sample input file. We then transfered the files to the MTS operating system on the 148 (MTS-WPP) and made several changes to the SYSDEP module to convert TEX to run on MTS.

File naming conventions were the major changes including the removal of most file extensions, allowing file names of up to 20 characters including periods, altering the option specifications of the RESET and REWRITE calls for MTS usage, and changing the PDSIN call used to open font files into a standard RESET call.

Some values in the ASCII↔EBCDIC tables had to be altered to allow for differences between the EBCDIC codes that IBM and MTS uses for some of the characters; we had some hard to trace problems before these differences were noticed—would you believe the characters in question were \ (backslash), ` (grave), { (left curly brace), and } (right curly brace). Then there were the differences in character sets caused by the discrepancies between standard ASCII and Stanford SUAI code. Presumably because the right curly brace is ´175 in ASCII and ´176 for SUAI, the CHRX array in SYSDEP had both the '7D'x and '7E'x mapped into 'D0'x (right curly). Normally in ASCII (and to be consistent with its inverse in the ORDX array), '7E'x should get mapped into '5F'x (not-sign or tilde)

We also had some problems with ASCIITBL. Line 34 (character ´042) was a vertical bar (| or '4F'x) and should have been an exclamation point (! or '5A'x). Line 95 (character ´136) was a not-sign (¬ or '5F'x) which we changed to a cent-sign (¢ or '4A'x) to agree with SYSDEP. Line 125 (character ´174) was a broken vertical bar (¦ or '6A'x) rather than the vertical bar expected by SYSDEP (| or '4F'x). Finally to maintain consistency with the right curly brace change to SYSDEP mentioned above, lines 126 and 127 (characters ´175 and ´176) were changed from ALT and right curly to right curly (} or 'D0'x) and not-sign (¬ or '5F'x) respectively. (Of course, the chcode declaration in the BASIC macro file making character ´176 a close delimiter had to be changed accordingly.

We encountered a Pascal/VS problem running TEXPRE on MTS. Due to its large memory requirements (and some glitches in the MTS Pascal/VS), we had to increase the STACK and HEAP Pascal/VS run-time parameters to get TEXPRE to run properly. TEX itself bombed reading font files that had lines that had been padded with a blank (introduced because files are shipped from VM/CMS to MTS-WPP through a "virtual card reader," and MTS trims all but one blank when trimming file lines). When this was corrected, we ran across several problems with the code in INLN in SYSDEP.

A logical test for end-of-line failed at the end of a file and an end-of-line test had to be added to the logical expression. (Technically, in Pascal EOLN is not defined for a text input file when EOF is true. On page 115 of IBM's Pascal/VS Language Reference Manual, EOLN is to be true if the file is positioned to the EOLN character and false otherwise. However, under MVS and VM/CMS, EOLN is true at the end of the file, and TEX depended on this. In the Pascal/VS at UM, EOLN was false when EOF was true.) Also, we got a Pascal/VS run time error when reading input lines that were too long and needed to be broken. The code in INLN failed to assign a value to the last character of the buffer; we assigned "buffer[bufptr]:=c" after the REPEAT...UNTIL. All in all, the initial installation went smoothly and took less than a week of part time work.

Noting that TEX reads and writes four byte file lines (most notably TEXINTBL which is over 44,000 lines and the DVI file itself) and that Pascal/VS does no blocking of such files, we added some blocking to TEX's I/O routines. We noticed about a threefold improvement in TEX's execution speed on MTS by blocking TEXINTBL and the DVI files into 4000 byte records. Furthermore, since MTS does its own dynamic memory allocation (and charges for memory for just the time it is allocated), we made several changes to TEX's memory management process.

Initially we allocate a small part of the MEM array (by using the REF feature of Pascal/VS and allocating the space dynamically with our operating system) and then get and free blocks of extra memory as required. We still would like to make some additions and changes to SYSDEP so that the interaction between TEX and the user at the terminal works a bit better on MTS.

With the help of David Fuchs, a program was developed to convert DVI files to a format that the Linotron 202 can handle. Of course, much of the display mathematics and more interesting capabilities of TEX are not available since we are restricted to the fonts available on the Linotron, but the output looks good. Next we hope to investigate cheaper proof quality output devices so the expensive Linotron copy need only be produced for the final copy. In the future we may consider purchasing a phototypesetter such as an Autologic APS-Micro 5.

VAX/VMS

Monte C. Nichols
Sandia Laboratories
Livermore, CA

David Kellerman
Oregon Software
Portland, Oregon

The most recent Stanford version of TEX-Pascal along with the improved spooler for the Versatec is now available from Oregon Software. This is of course the version having magnification capability, etc., and is the last version prior to TEX82. The new version should get rid of most of the remaining bugs and keep the VAX/VMS community running in a superior fashion until a version of TEX82 becomes available for VMS. For those of you new to TUG, Oregon Software has volunteered to distribute TEX for the VAX/VMS community. For $50 they will send you a tape with all the VAX/TEX related files on it. See TUGboat Vol. 2, No. 2 and Vol. 3, No. 1 for further information.

A rather large number of VMSer's attended the most recent TUG meeting at Stanford as well as the classes for TEX82 that followed. Several individuals have gotten WEB, TANGLE and WEAVE running on the VAX/VMS system, so we should be able to see a preliminary version of TEX82 at some reasonable date. Our best guess for the availability of a production version of TEX82 would be Nov–Dec 1982, but it could happen sooner.

The newest version of TEX as well as TEX82 use .tfm and .pxl font files. Unfortunately the 200 dot/in font files made available from Stanford in this new format did not include some of the old font files we had come to know and love; especially noted by their absence were the larger fonts that had proven useful for a number of purposes. One of us (MCN) has obtained METAFONT files for some of these missing files and plans to find a DEC 10 or 20 soon where the missing 200 dpi fonts can be generated.

Anyone who has developed spoolers for devices other than Versatec is encouraged to send them to Oregon Software for inclusion on the next distribution tape.

* * * * * * * * * * *

"small" TEX

* * * * * * * * * * *

Send submissions to:
Lance Carnes
163 Linden Lane
Mill Valley, CA 94941
(415) 388-8853

In the 16-bit and supermicro arena there are now three implementations available, and a fourth on the horizon with the advent of TEX82. There is a new entry in the small TEX family, an implementation on the Apollo from OCLC. Bob McClure and Kent Harris' ONYX/C version is now marketed by TYX in Reston, Virginia and has a few new wrinkles. The HP3000 implementation gained a new output device, the HP2680A — HP's entry in the Laser printer marketplace. (This column was printed on the HP2680A.) And, good news for would-be UNIX TEX users, a possible C version of TEX82!

Tom Hickey and his associates at OCLC have TEX-in-Pascal running on their Apollo supermicro. (Tom has METAFONT running on this machine, also.) At the recent TUG conference we saw a document on Chel, a METAFONT-designed alphabet, printed using the Apollo TEX with Versatec output. If you want more information on this implementation, contact Tom Hickey at OCLC (617) 486-3661.

TYX in Reston, VA is marketing McClure and Harris' TEX in C on the ONYX (16-bit micro which runs UNIX). In addition to their rewrite of the SAIL TEX they offer a forms design package, and a user-friendly front end. Their implementation is also offered on the PLEXUS or PDP-11/44 (or any of the 11's with separate IND space), and TYX has an interface to the Canon LBP. For more information contact: Dick Gauthier, TYX, 11250 Roger Bacon Drive, Suite 16, Reston, VA 22090, (703) 471-0233.

My famous Hewlett-Packard HP3000 implementation now includes a driver for the HP2680A, HP's entry in the page printer market. Barbara Beeton kindly consented to having the small TEX column printed on the HP2680A, to give readers an idea of its quality. The fonts used are those available on the Stanford tape (nnnnnn.1300PXL). The 2680A allows 32 fonts to be downloaded into its memory, and the driver software need only address the font and character desired thereafter. Since it has a resolution of 180 dots/inch, the already magnified Versatec fonts are further magnified, giving characters 1.4444 times larger than real life. This page was TEX-ed with magnification 1444 (i.e. 1.4444 times larger) and photo–reduced to the publication size. At the end of the column are some samples of mathematics output and various font sizes. For more information contact me at the above address.

And now for all those who have been waiting for TEX on a personal computer, or at least on PDP-11's, here is good news. With the advent of TEX82, written in WEB, there is a good chance a C version of TEX82 may become available, given a C–TANGLE. (For details on WEB, see TUGboat Vol. 3 No. 1. For details on TEX82, refer to articles in this issue.) Please note that TEX82 is not yet ready for general distribution and will probably not hit the streets till later this year. The version of TANGLE which will produce C source is only a gleam in its creator's eye at this point. (Note: the C version of TEX from TYX is proprietary.)

The basic idea is that if WEB can be TANGLEd into Pascal, why not into C? or the language of your choice? Presumably, a diligent hacker could adapt TANGLE to emit almost anything, and then bring up TEX on his or her favorite system. There are many powerful microcomputers, and even some personal computers (e.g. the DEC350), which could run such an implementation. Let us know what you would like to see — use this column as a forum.

The following are samples of HP2680A output.

Bigg Big

Boldface *Italic* *Slant*

Teletype $SYMBOL\ \infty\in\emptyset\surd\nabla\int\forall\exists\neg\aleph\Re\int$

$$\int\oint\odot\bigodot\oplus\bigoplus\otimes\bigotimes$$

$$\Sigma\amalg\int\hat{Y}$$

$$1+\cfrac{1}{a_1+\cfrac{1}{a_2+\cfrac{1}{a_3+\cfrac{1}{a_4}}}}$$

$$\frac{1}{2\pi}\int_{-\infty}^{\sqrt{3}}\Big(\sum_{k=1}^{n}\sin^2 x_k(t)\Big)\big(f(t)+g(t)\big)\,dt.$$

$$\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+\sqrt{1+x}}}}}}}$$

* * * * * * * * * * *

Fonts

* * * * * * * * * * *

A FORTRAN VERSION OF METAFONT

Sao Khai Mong
Electrocon International, Inc.
611 Church Street, Ann Arbor, Michigan 48105

I have been using a Fortran 77 version of **METAFONT** which was developed for Harris Computers. This program was translated from the Stanford SAIL version of the program. The intent was to use **METAFONT** to design characters for the Burmese script and eventually to use it in something similar to TEX. The program is, of course, useable for scripts other than Burmese.

The conversion from Sail to Fortran 77 was, for the most part, straightforward. A direct statement-for-statement substitution was possible for the major part of the code. However, there were a few problem areas that did not lend themselves to this method. Among the major things that Fortran 77 sorely lacked were:

- Recursive procedure calls
- Strings as nice as those of SAIL
- Case statements
- Macro calls
- Bit manipulations

Each of the problem points was solved in an ad-hoc fashion. Because of this, the translation is not as neat and tidy as I would have liked. No attempt was made to adhere too closely to the ANSI standards for Fortran 77. The data structures and tables used by the program are an exact copy of the SAIL version. The translated program turns out to be 2-1/2 to 3 times more lengthy than the SAIL version in terms of line count. The translation effort took about six weeks of night and weekend work including debugging and testing.

Harris Fortran has several nonstandard features which I used extensively to speed up the translation. It supports a 48-bit integer word as well as bitwise logical operations on them which was quite essential for the direct translation. Memory limitations were not a concern as the machine can address up to 3 megabytes. A 1024×640 raster map was accommodated with no problems. Harris Fortran also supported additional control structures, as well as type checking of variables, which were very useful.

The only output device and display mode currently implemented is a DEC VT-100 fitted with the VT-640 Retrographics board. The characters drawn by the program can only be viewed in the "drawdisplay" mode. Unfortunately, no suitable hardcopy device is available for the results to be shown here. (Using a Tektronix terminal in the point-plot mode was found to be unbearably slow.) None of the extensive output routines that I received have been translated yet. This includes output for TEX. I neither have a suitable Pascal compiler nor a working TEX program to motivate me to take on the task. However, except for these points, the program does work almost exactly as described in the book "TEX and **METAFONT**." I do believe that I have captured most of the heart and sole described in the book. **METAFONT** programs from Stanford run with no problems after suitable conversion of the SUAI character set to the standard ASCII set.

I have found **METAFONT** to be quite adequate for defining the Burmese script. The script has evolved from the Pali and Sanskrit languages, and to simplify things, may be said to consist of arcs and straight lines. I felt that there was a lot of room for improving the quality of the fonts as well as introducing new ones. The script does not have a very long tradition of commercial printing as in the West, and hopefully, there is not yet too much inertia about accepting new ideas.

People interested in obtaining a copy of the program may contact me.

* * * * * * * * * * *

Warnings & Limitations

* * * * * * * * * * *

Charting the Generation Gulf

Until TEX82 is universal, a major source of incompatibility among TEX sites will be the difference in available features owing to the difference in creation dates of the program. Even at a single site, many failures to run a TEX job successfully are the consequence of unknowingly using a less-than-current version of TEX. If some macro that you think should work just won't, it's worth trying to determine the age of your TEX, and here's a quick way to do it.

In the errata list distributed with each issue of TUGboat, additions are dated. Select a suitable item from each generation, and enter these items in chronological order to an on-line TEX session; the first failure will date your program with reasonable accuracy. The following items were suggested during a problems session at the July meeting.

- TEX80: the last ... : `\let` or `\specskip`(digit)

- since October 1, 1980: \topsep (note – true and magnification may not be implemented even if other items in this group have been)
- since February 4, 1981: \hbox par SIZE{...}
- since June 30, 1981: \ifx (cs1)(cs2){(true text)}\else{(false text)}

(At the meeting, \chcode }=2 was suggested for the last group; when tested at AMS, however, it gave no error message, although it was not acted on.)

The version of TEX in production at the Math. Society is current as of March 1981. The following log resulted from performing the suggested tests. Note that at least one font code must be assigned to avoid the error "Whoa---you have to assign a font first."

```
@tex
AMS TeX version of 11500 created
  Wednesday, August 18, 1982 16:07:51
this run of TeX begun:
  Thursday, September 2, 1982 13:08:52
*\font a=cmr10  \:a
*\let \a=\alpha
*\topsep{\vskip 23pt}
*\hbox par SIZE{a}
*\chcode ~=0
*~foo
*\ifx\a\alpha{aa}\else{bb}
! Undefined control sequence.
(*) \ifx
        \a\alpha{aa}\else{bb}

^x
Type x again to exit:xNo output file.
```

Once you have determined the appropriate date, it is advisable to apply all relevant errata to your reference copy of the TEX manual. This will save both time and embarrassment (from asking silly questions), until TEX82 redefines the problems.

Barbara Beeton

FONT CODES IN POPULAR USE

Calvin Jackson

The following tables list the 64 possible *fontcodes* as used in several popular macro packages.

font code	CIT-CS	BASIC	Fácil	AMS TEX	ACP
`					
a	cmr10	cmr10	cmr10	cmr10	cmr10
b	cmr9		cmr9	cmr9	cmr9
c	cmr8		cmr6	cmr8	cmr8
d	cmr7	cmr7	cmr7	cmr7	cmr7
e	cmr6		cmb8	cmr6	cmr6
f	cmr5	cmr5	cmr5	cmr5	cmr5
g	cmi10	cmi10	cmi10	cmi10	cmi10
h	cmi9		cmi9	cmi9	cmi9
i	cmi8		cmsy9	cmi8	cmi8
j	cmi7	cmi7	cmi7	cmi7	cmi7
k	cmi6		cmb9	cmi6	cmi6
l	cmi5	cmi5	cmi5	cmi5	cmi5
m	cmcsc		cmr8	cmcsc	cmsc10
n	cms10	cms10	cms10	cms10	cms10
o	cms9		cmsy8	cms9	cms9
p	cms8		cms8	cms8	cms8
q	cmb10	cmb10	cmb10	cmb10	cmb10
r	cmb9		cmi8	cmb9	cmb9
s	cmb8		cms9	cmb8	cmb8
t	cmtt		cmtt	cmtt	cmtt
u	cmsy10	cmsy10	cmsy10	cmsy10	cmsy10
v	cmsy9		cmi6	cmsy9	cmsy9
w	cmsy8		cmsy6	cmsy8	cmsy8
x	cmsy7	cmsy7	cmsy7	cmsy7	cmsy7
y	cmsy6		cmtt9	cmsy6	cmsy6
z	cmsy5	cmsy5	cmsy5	cmsy5	cmsy5
;	cmtitl			cms6	cmtitl
<	cmssb			cmssb	cmssb
=				cmcsc8	cmss12
>	cmss8			cmb6	cmss8
?	cmti10	cmti10	cmti10	cmti10	cmss8

font code	CIT-CS	BASIC	Fácil	AMS TEX	ACP
@	cmathx	cmathx	cmathx	cmathx	cmathx
A	cmr18		cmr18	cmcy10	
B	cmr12		cmr12		
C	cmdunh		cmdunh	cmcy8	
D	cmss12		cmss12		
E	cmu10		cmu10	cmcy6	
F	cmcsc9		cmcsc9		
G	cmcsc		cmcsc		
H	cmtt8		cmtt8		
I	cmti9		cmti9		
J	cmti8		cmti8		
K	cmcsc8		cmcsc8		
L	cmr30		cmr30		
M					
N					
O					
P					
Q					
R					
S					
T					
U					
V					
W					
X					
Y					
Z					
[				cmti9	
\					
]				cmti8	
↑					
_					

* * * * * * * * * * *

MACRO COLUMN

Send Submissions to:
Lynne A. Price
TUG Macro Coordinator
Calma R&D
527 Lakeside Drive
Sunnyvale, CA 94086

Many existing macros will be replaced when TEX82 is distributed. The current versions tend to exist in large packages; future macros will be most useful if each feature is self-contained so that TEX users can pick and choose pieces from several packages. In order to promote modularity, Art Keller and Dan Brotsky have volunteered to work on standard mechanisms such as allocation of font codes and of box and counter numbers. In addition, they have suggested that this column include a "phone book" of TEX82 macro names. Macro writers should submit macro names, along with a very brief description, to the editor. When providing an alternate implementation of a similar function, other writers can use a name that appears on the published list; for new capabilities, existing names should be avoided. Of course, writers should contribute their macros as well as the macro names to TUG. Names can be reserved before macros are written. However, names listed in one issue will be deleted, unless the corresponding macro is received before the following issue.

* * * * * * * * * * *

TUGBOAT MACRO INDEX

The following list catalogues macros that have appeared in TUGboat. Entries are listed by volume, number, and page as well as author's name. Items that could not be categorized by an obvious headword have been listed under "miscellaneous". Many items refer to parts of large macro packages; users of other packages may find them valuable models for macros of their own.

Readers' comments on the format as well as the contents of this index are welcome.

* * * * * * * * * * *

MULTI-COLUMN OUTPUT FORMAT

Barbara N. Beeton
American Mathematical Society

At the AMS, we are still using the old SAIL version of TEX, which is severely limited in memory capacity. Several of our publications are formatted with very small type in multiple columns; one such publication, the *Combined Membership List* of the Society and two other mathematical organizations, can require over 15,000 6-point characters on a single printed page.

To avoid overloading memory (both `memsize` and `varsize` are susceptible), we take advantage of the fact that, to TEX, each column is a "\page". Instead of saving all columns on a page until the final column is complete, each column is shipped out to the .DVI file as soon as it is ready. The several columns which comprise a true page are then "pasted up" by the output driver software, using instructions stored in an "option" file or interactively by responding to a "format spec" request.

There is another advantage to this technique has: Our publication-quality output device, an Alphatype CRS, sets type one baseline at a time, across the full page width for each baseline. Mechanically, a lens (which transmits the type image from a CRT screen to photographic paper) rides along a worm gear for the required distance, then returns. For most applications, type is set in both forward and reverse directions, but in some cases (because of alignment problems), type can be produced in only one direction. If a page contains 4 columns, say, but the baselines from column to column are not evenly aligned (as in the TUG membership list), driving the lens across the full page width could cause the distance traveled to be over 300% greated than necessary, with a corresponding increase in the length of time required to complete a page. Since the Alphatype is a slow machine (wall-clock time can be over 5 minutes for particularly dense pages), the saving is significant.

Initialization and defaults

The output routine requires two "page counters": \count0 keeps track of columns (or "\pages" in the TEX sense), and \count9 is used to record the printing page number. Depending on how the output driver keeps track of where it is in a .dvi file, one or the other of these can be used to restart a job in the middle, or to print only selected pages.

Registration marks may be output to delimit the trim area; "T" marks, centered top and bottom, or drawn corners may be chosen. Vertical rules may be drawn between columns. Running heads and folios are accommodated, as are top and bottom matter on the first page; these "full-width" elements are output only on the last segment of a page, while registration marks are generated on each segment, assuring that all true pages have the same number of columns for the sake of the output device driver. Multiple sections, each with its own "first page", may be strung together in the same TeX run.

Page width is calculated dynamically. Column and intercolumn widths are specified (in the input) as an integral number of points. If \leaders are to be used, the column and intercolumn widths must be multiples of the leader width, in order to assure correct alignment.

All parameters are initialized; if none are reset, output will be two-column pages of TUGboat dimensions, that is, suitable for printing on 8.5×11 inch paper, with 1-inch side margins and .75-inch margins top and bottom. All running head and folio strings are initially empty, and only those required for a particular job need be reset.

Parameters are of 4 types:

n = integer
d = dimension (e.g. 12in)
x = single letter or text string
c = control sequence

This header file (called `multcol.hdr`) may be used with AMS-TEX. If it is, the user must specify

```
\useamstex
```

and input the header files in the following order:

```
\input 0-0at.mac
\input multcol.hdr
\input 0-0at.fnl
```

Page dimensions (in points) are set as follows:

Page width:

\setcolmax{n}	number of columns
\setcolwd{n}	width of one column
\setintercol{n}	width of gap between columns
\resetpagewd	

Page length:

\settoplgt{n}	height of first-page header
\setbotlgt{n}	height of first-page footer
\setrheadlgt{n}	height of running head box
\setrfootlgt{n}	height of folio box
\setcollgt{n}	height of full-page column
\resetfpagelgt	or \resetpagelgt

Page and column numbers (initialized to 1) are set by:

\setpageno{n}	printing page number
\setspoolno{n}	column number for spooler

To establish type and placement of trim marks:

\settrimtype{x}	C, T, or U
\settrimlgt{d}	default = 11in
\settrimwd{d}	default = \pagewd
\setheadmargin{d}	

Type C gives top and bottom corners at the trim boundaries, T (default) gives "T" marks, and U gives upper corners only. The page contents are centered horizontally within the trim width (the present version does not permit different treatment of left- and right-hand pages, e.g.), and vertically within the trim length unless a different head margin has been specified.

A vertical rule will be drawn between columns if

```
\userule{T}
```

is specified, or suppressed (default) if F.

To define running heads (all text strings default to null):

\setrunners{	
\firstrun x\\	running head on first page: T = yes or F = no (default)
\rheadfont c\\	font; default is 'current' font
\outside x\\	
\inside x\\	
\midhead x\\	
\leftmid x\\	
\rightmid x\\	
\runskip d\\	
}	end of running head items

\outside and \inside specify the running head segments which appear at the outer margin (left on even pages, right on odd) and toward the spine, respectively. If centered header text is to be the same on left (even) and right (odd) pages, \midhead is used; otherwise \leftmid and \rightmid give the different segments. \runskip gives the distance between the baseline of the running head and the top of the page body; default = 10pt. \setrunners{...} is cumulative: different portions may be initiated at different times, as convenient; no portion returns to the default value automatically, but must be reset.

A folio, or page footer, may be defined by giving the full description:

\setfolio{x}

This is not implemented as elaborately as the running head, mainly because folios are not as common as running heads in AMS publications.

Top and bottom matter for the first page are specified by:

setfirsthead{x}
setfirstfoot{x}

These items are set within \vboxes of heights specified by \settoplgt and \setbotlgt.

The following marks are made available for each completed page:

\topterm	\firstmark at top of first column
\lastterm	\botmark from last completed column

At the end of a section (bottom of last data column, just before \eject) a message or special routine may be inserted:

\def \endjobmsg{x}

A common use of this feature is a \send to establish the starting page number for a subsequent section. In any event, the following message is sent to the terminal (and to the .err file) at the end of the job:

```
\send0{data ends on page \curpage, column \xcol}
```

(this requires that file 0 not be \opened).

Macro definitions

The following "utility" macros are required:

```
% avoid vertical glue when making up pages:
\def \basezero{\baselineskip 0pt\lineskip 0pt}
% pseudo-counters:
% structure: \xcount{name}{value}
\def \setxcount#1#2{\setcount7#2
        \xdef#1{\count7}}
\def \advxcount#1{\setcount7#1
        \advcount7\xdef#1{\count7}}
\def \chgxcount#1#2{\setcount7#1
        \advcount7 by #2\xdef#1{\count7}}
% registration marks:
% "T" marks centered on top and bottom trim edges
\def \topregister{\vbox to 0pt{\vss
     \hbox to \trimwd{\hfil
       \vrule height 24 pt width 0.2pt\hfil}
     \hbox to \trimwd{\hfil
       \vrule height 0.2pt width 0.5in\hfil}}}
\def \tbotregister{\vbox to 0pt{
     \hbox to \trimwd{\hfil
       \vrule height 0.2pt width 0.5in\hfil}
     \hbox to \trimwd{\hfil
       \vrule height 24 pt width 0.2pt\hfil}
     \vss}}
% corners at limits of trim area
\def \ctopregister{\vbox to 0pt{
     \hbox to \pagewd{\hss\hbox to \trimwd
        {\vrule depth .5in width 0.2pt
         \vrule depth 0.2pt width .5in
         \hfil
         \vrule depth 0.2pt width .5in
         \vrule depth .5in width 0.2pt}\hss}
     \vss}}
\def \cbotregister{\vbox to 0pt{\vss
     \hbox to \pagewd{\hss\hbox to \trimwd
        {\vrule height .5in width 0.2pt
         \vrule height 0.2pt width .5in
         \hfil
         \vrule height 0.2pt width .5in
         \vrule height .5in width 0.2pt}\hss}}}
```

AMS-TEX "protects" certain control sequences, e.g. \page as \page|, and disables the "basic" sequence. The following permits an orderly transition to the AMS-TEX conventions:

```
\def \isamstex{B}
\def \useamstex{\gdef\isamstex{A}
     \gdef\normaloutput|{\outa}}
```

Initialization comprises a large number of control sequence pairs, of the following structure:

```
\def \colmax{2}
\def \setcolmax #1{\gdef\colmax{#1}}
```

The following conform to this structure, with defaults as shown:

\setcolmax	\colmax	2
\setcolwd	\xcolwd	225
\setintercol	\intercol	18
\settoplgt	\xtoplgt	0
\setbotlgt	\xbotlgt	0
\setrheadlgt	\xrheadlgt	24
\setrfootlgt	\xrfootlgt	\xrheadlgt
\setcollgt	\xcollgt	648
\settrimlgt	\xtrimlgt	11in
\settrimwd	\xtrimwd	\pagewd
\userule	\xrule	F
\setfolio	\folio	{}

Some of the initialization macros are more elaborate:

```
\def \headmarginsw{F}
\def \headmarginlgt{}
\def \setheadmargin #1{\gdef\headmarginsw{T}
     \gdef\headmarginlgt{#1}}

\def \topregister{\ttopregister}
\def \botregister{\tbotregister}
\def \settrimtype #1{
     \if T#1{\gdef\topregister{\ttopregister}
             \gdef\botregister{\tbotregister}}
     \else{\if C#1{\gdef\topregister{\ctopregister}
                   \gdef\botregister{\cbotregister}}
     \else{\if U#1{\gdef\topregister{\ctopregister}
                   \gdef\botregister{}}
     \else{\send0{invalid trim type; T marks will be used}}}}}

\def \firstrunner{F}
\def \firstfolio{F}
\def \rhfont{}
\def \outrunner{}
\def \inrunner{}
\def \leftmidrunner{}
\def \rightmidrunner{}
\def \runskiplgt{10pt}
\def \setrunners #1{
     \def\firstrun##1\\{\gdef\firstrunner{##1}}
     \def\rheadfont##1\\{\gdef\rhfont{##1}}
     \def\outside##1\\{\gdef\outrunner{{##1}}}
     \def\inside##1\\{\gdef\inrunner{{##1}}}
     \def\midhead##1\\{\gdef\leftmidrunner{{##1}}
                       \gdef\rightmidrunner{{##1}}}
     \def\leftmid##1\\{\gdef\leftmidrunner{{##1}}}
     \def\rightmid##1\\{\gdef\rightmidrunner{{##1}}}
     \def\runskip##1\\{\gdef\runskiplgt{##1}}
     #1}

\def \firsthead{}
\def \setfirsthead #1{\gdef\firsthead{
     \vbox to \xtoplgt pt{
              \if \xcol\colmax{{#1}}
              \else{}}}}

\def \firstfoot{}
\def \setfirstfoot #1{\gdef\firstfoot{
     \vbox to \xbotlgt pt{
              \if \xcol\colmax{{#1}}
              \else{}}}}

\def \setcurpage{\ifpos9{\xdef\curpage{\count9}}
                 \else{\setcount7 -\count9          adjust for roman numerals
                       \xdef\curpage{-\count7}}}
\def \pageno{1}
\def \resetcurpage{\setcount9\pageno \setcurpage}
\def \setpageno #1{\gdef\pageno{#1}\resetcurpage}
\resetcurpage

\setcount0 1
\def \setspoolno #1{\setcount0 #1 }

\topbaseline 0pt                    align tops of multiple columns rather than baselines
                                    to accommodate type of different sizes
```

Page dimensions are calculated using counter arithmetic:

```
\def\resetpagelgt{                              ordinary page
    \setcount2 \collgt                          length of full-page column
    \advcount2 by \xrheadlgt                    add length of running head
    \advcount2 by 2                             include \maxdepth
    \advcount2 by \xrfootlgt                    add length of folio
    \xdef \pagelgt{\count2 pt}                  full-page length
    \xdef \rheadlgt{\xrheadlgt pt}
    \xdef \rfootlgt{\xrfootlgt pt}}

\def \resetfpagelgt{\resetpagelgt               first page of a section
     \xdef \toplgt{\xtoplgt pt}                 length of first page top matter
     \xdef \botlgt{\xbotlgt pt}                 length of first page bottom matter
     \setcount1 \collgt
     \advcount1 by -\xtoplgt
     \advcount1 by -\xbotlgt
     \vsize \count1 pt                          length of column on first page
     \gdef \fpage{T}}
\resetfpagelgt

\def \howwide{\setcount8\xcol \setcount3 0 \sowide}
\def \sowide{\advcount8 by -1
     \advcount3 by \xcolwd                      for each column, add column width
     \ifpos8{\advcount3 by \intercol            add intercol for all but last column
             \sowide}                           keep going, up to number of columns
     \else{\xdef\thiswide{\count3 pt}}}         in current "page"

\def \resetpagewd{
     \xdef \colwd{\xcolwd pt}                   column measure
     \hsize \colwd
     \xdef \xcol{\colmax}
     \howwide
     \xdef \pagewd{\thiswide}
     \xdef \xcol{1}}
\resetpagewd
```

Running heads and folios are pieced together from input segments for use in the output routine:

```
\def \runner{\hbox to \pagewd{\rhfont
     \spose{\hbox to \pagewd
           {\ifeven9{\hfil\leftmidrunner\hfil}
            \else{\hfil\rightmidrunner\hfil}}}\!
     \ifeven9{\unskip\outrunner\hfill\null\inrunner\unskip}
     \else{\unskip\inrunner\hfill\null\outrunner\unskip}}}

\def \runhead{\vbox to \rheadlgt{\vss
     \if \xcol\colmax{
         \if T\fpage{\if F\firstrunner{}
                     \else{\runner}}
         \else{\runner}}
     \else{}
     \vskip\runskiplgt}}

\def \runfoot{\vbox to \rfootlgt{\vss
     \if \xcol\colmax{\folio}
     \else{}}}
```

A couple more utility definitions for special circumstances:

```
\def \ruler{\if T\xrule{\hbox to \intercol pt{\hfil\vrule\null
                       \vbox to size{\hbox to 0pt{}\vfil}\hfil}}
            \else{}}

\def \markit{}                  to permit tricky code of the sort used to
                                insert continuation entries at top of next column
```

At last! The actual output routine!

```
\def \midpage{\hbox to \trimwd{\hfil
     \vbox to \trimlgt{\basezero
          \topregister
          \if T\headmarginsw{\vskip\headmarginlgt}
          \else{\vfill}
          \markit\par
          \vbox to \pagelgt{\basezero
               \runhead
               \if T\fpage{\firsthead}
               \else{}
               \gdef\lastterm{\botmark}
               \howwide                    calculate width of current "page"
               \vbox to size{\hbox to \thiswide{\hfil
                    \if 1\xcol{}
                    \else{\ruler}\!
                    \if A\isamstex{\page|}
                    \else{\page}}}
               \if T\fpage{\firstfoot}
               \else{}
               \runfoot }
          \vfill
          \botregister}\hfil}
     \advcount0 }                          number each output segment uniquely

\def \xcolstart{\if \xcol\colmax{\output{\outa}}
                \else{\output{\outb}} }

\def \xcolend{\if \xcol\colmax{\advcount9 \xdef\curpage{\count9}
                \vsize\xcollgt pt
                \gdef\xcol{1}
                \gdef\fpage{F}}
           \else{\advxcount\xcol} }

\output{\outa}
\def \outa{\xcolstart
     \xdef\topterm{\firstmark}
     \midpage
     \xcolend}
\def \outb{\xcolstart
     \midpage
     \xcolend}
```

And finally, various macros to fill out incomplete columns, terminate sections neatly, and finish up a job, reporting to the user where it ended. In \newcol, intended to permit manual balancing of the last page of a section, the penalty is necessary to overcome possible large negative penalties at other points in the column, especially when the column is broken very close to maximum length.

```
\def \newcol{\par\penalty -900\vfil\eject}

\def \nullcol{\hbox to \colwd{\null}\eject}        empty column to fill page

\def \blankit{
     \if T\fpass{\xdef\fpass{F}\vfill\reportlastcol\eject\blankit}
     \else{\if 1\xcol{}
          \else{\nullcol\blankit}}}

\def \endsection{\gdef\fpass{T}\blankit}

\def \endjobmsg{}          allow special messages, sending starting page for
                           next section, etc.

\def \reportlastcol{\send0{data ends on page \curpage, column \xcol}\endjobmsg}
```

SOME TEX PROGRAMMING HACKS

Leslie Lamport
SRI International

When writing a complicated TEX macro, one is essentially writing a program. Since TEX is a macro substitution language, writing programs in it can be a bit tricky. This note describes how to implement some ordinary programming language features in TEX. This allows you to write a macro by first writing it as an ordinary program, then translating it into TEX. I find this to be quite helpful.

For program control structures – in particular, while and if statements – I refer you to Brendan D. McKay's macros that appeared in TUGBOAT. I will concentrate on assignment statements and data structures. First, let me define some terms.

text: Any TEX input containing balanced braces.

variable: A TEX macro name, beginning with \ – *e.g.*, \foo. I will use \var as a generic variable.

val(\var): A piece of text that is the value of the variable \var.

eval(*text*): The result of evaluating *text* by recursively replacing all nonprimitive macro names and all \counter expressions by their values. For example, if \count5 = 13, *val*(\foo) = \bar yz and *val*(\bar) = xx, then

$$eval(\text{\hbox to \count5 pt\{\foo\}}) = \text{\hbox to 13pt\{xxyz\}} \quad .$$

Note that *eval* is idempotent, so

$$eval(eval(text)) = eval(text) \quad .$$

The first thing to note is that TEX provides a block-structured language, so each variable can nested definitions as in Algol. In TEX, a block is begun by a { and ended by a }. This means that there are two kinds of assignments:

\var := ...: Changes the the value of \var in the local block.

\var :== ...: Changes the the value of \var in the local block and all enclosing blocks.

Here are some kinds of assignment statements that you can construct.

\var := *text*. It is implemented in TEX as:

\def\var{*text*}

Note that if *text* consists of a single variable name, this makes \var synonymous with that variable.

\var :≡ *text*. It is implemented as:

$$\gdef\var\{text\}$$

\var :≡ *eval*(*text*). Implemented as:

$$\xdef\var\{text\} \quad .$$

\var :≡ *val*(\varx), where \varx is another variable. It is implemented as:

$$\let\var=\varx \quad .$$

There are two data structures that I have figured out how to implement: a push-down stack and an array. The macros are the following.

\vpush\stk{*text*} – pushes *eval*(*text*) onto stack \stk.

\vpop\var\stk – performs the assignment

$$\var :\equiv head(\stk)$$

and pops stack \stk.

\rdarray\var\array{*i*} – performs the assignment

$$\var := \array(eval(i)) \quad ,$$

where *i* should evaluate to an integer.

\vrarray\array{*i*}{*text*} performs the assignment

$$\array(eval(i)) \equiv text$$

For these array operations, \array must be defined to have the form

$$\downdownarrows\var_1\downdownarrows\var_2\ldots\downdownarrows\var_n$$

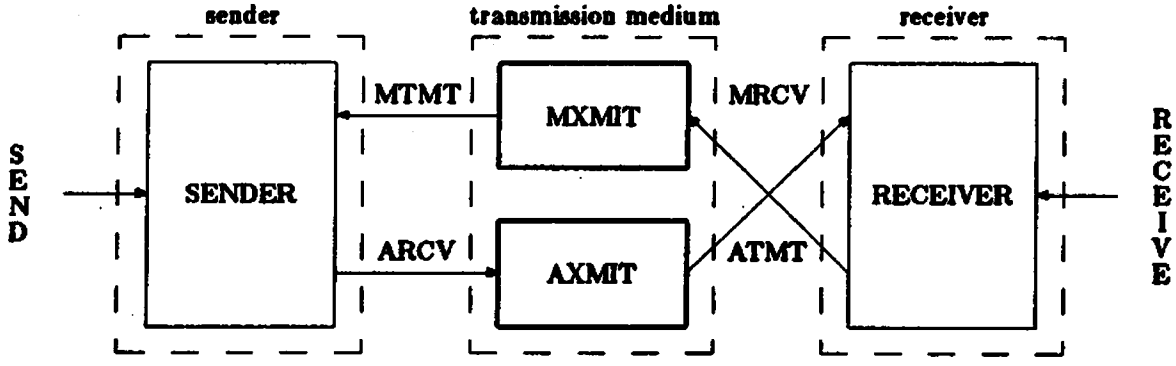
where the $\var_i$ are variables not used elsewhere.

These macros are defined as follows.

```
\def\vpush#1#2{\xdef #1{\xdef \ii{#2}\xdef #1{#1}}}
\def\vpop#1#2{#2\xdef #1{\ii}}
\def\rdarray#1#2#3{\def\ii##1{\ifpos9{\gdef\tmp{\let#1=##1}\advcount9by-1}\else
                    {}}\setcount9#3#2\tmp}
\def\wrarray#1#2#3{\def\ii##1{\advcount9by-1\ifzero9{\gdef##1{#3}}\else
                    {}}\setcount9#2#1}
```

. . . . . .

Editor's note: The preceding item is reproduced from Canon copy supplied by the author. He provided the editor with several other documents which were not able to be processed satisfactorily in time for the deadline (the editor's fault, not Les'). Les is using a DEC 10-compatible system, and has developed a macro package (FaCSL TEX,based on Max Díaz' Fácil TEX) with an EMACS preprocessor (PRETEX) capable of performing syntax checks and expanding bibliographic citations from references stored separately from the root file. One particularly interesting feature permits a user to create "just about any kind of picture you want that doesn't contain curved lines", such as:



This feature requires fonts not yet available at AMS. We will try to install them and present the details in the next issue.

* * * * * * * * * * *

UNBLOCKING AN $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX TAPE

Barbara Beeton
American Math Society

Several recipients of $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX tapes written for computers other than DEC 10/20s have complained that the tape format does not conform to the description supplied with the tape. The description specifies fixed records, fixed blocks, with the implication that no carriage return or line-feed codes are present. In fact, these tapes contain variable-length records, blocked, with each record terminated by a CR/LF.

Donald C. Wells, of the National Radio Astronomy Observatory, was kind enough to send a listing of a VAX/VMS Fortran program which "reads entire blocks, of arbitrary length, and scans them character by character to build up the proper lines of text. [The] program produces an auxiliary file which tabulates the lengths of all the blocks in the 20 files on the tape. ... It might be of assistance to someone else who is using a VAX under VMS."

```
      PROGRAM AMSTAP
C-----------------------------------------------------------------------
C   Program to unblock Fred's AMS macro tape.
C
C      ALL MMAO: TAPE
C      MOU/FOR/BLO=2000 TAPE: DUMMY
C
C   DCW 17May82.
C-----------------------------------------------------------------------
C
      PARAMETER       (MAXBLK = 2000)
      PARAMETER       (MAXLIN = 100)
      INTEGER*2       NF, NR, NBYTE, L, K
      CHARACTER       BLOCK*(MAXBLK), RTYPE*8, DFILE*6, SCLOSE*6,
     *                LINE*(MAXLIN)
C
      LT = 1
      RTYPE = 'VARIABLE'
      OPEN (UNIT=LT, FILE='TAPE:', STATUS='OLD', READONLY,
     *          ACCESS='SEQUENTIAL', FORM='FORMATTED',
     *          RECORDTYPE=RTYPE, RECL=MAXBLK, BUFFERCOUNT=2)
      REWIND LT
C
      LD = 2
      LS = 3
      NF = 0
C
 10   CONTINUE
      NF = NF + 1
      WRITE (DFILE, '(''AMS'', I2.2, ''.'')') NF
      OPEN (UNIT=LD, NAME=DFILE, STATUS='NEW', CARRIAGECONTROL='LIST')
      NR = 0
      LINE = ' '
      L = 0
C
 20   CONTINUE
        READ (LT, '(Q, A)', END=40) NBYTE, BLOCK(1:NBYTE)
        NR = NR + 1
        TYPE       *, 'NF=', NF, ', NR=', NR, ', NBYTE=', NBYTE
        WRITE (LS, *) 'NF=', NF, ', NR=', NR, ', NBYTE=', NBYTE
C
        DO 35 I = 1, NBYTE
           K = ICHAR (BLOCK(I:I))
           IF (K.NE.13) GO TO 24
C                                       carriage return:
              GO TO 35
C
 24        CONTINUE
           IF (K.NE.10) GO TO 27
C                                       line feed:
              WRITE (LD, '(A)') LINE (1:MAX(1,L))
              LINE = ' '
              L = 0
              GO TO 35
C
 27        CONTINUE
           IF (K.NE.0) GO TO 30
C                                       zero bytes:
              GO TO 35
 30        CONTINUE
C
              L = L + 1
              LINE(L:L) = CHAR (K)
C
 35        CONTINUE
        GO TO 20
C
 40   CONTINUE
      IF (L.GT.0) WRITE (LD, '(A)') LINE(1:L)
      SCLOSE = 'KEEP'
      IF (NR.LE.0) SCLOSE = 'DELETE'
      CLOSE (UNIT=LD, STATUS=SCLOSE)
      TYPE *, DFILE, SCLOSE, 'NR=', NR
      WRITE (LS, *) 'TAPEMARK.'
      IF (NR.GT.0) GO TO 10
C
      END
```

* * * * * * * * * * *

Problems

* * * * * * * * * * *

Send Submissions to:
Lynne A. Price
TUG Macro Coordinator
Calma R&D
527 Lakeside Drive
Sunnyvale, CA 94086

Paragraphs in tables

Problem #1: It is occasionally useful to produce tables in which the contents of some fields consist of entire paragraphs. At the last TUG meeting, Mark Blanford suggested the following technique

```
\input basic
\DefineFont{cmr18}{\bigfont}
\halign{#\hfil
       &\quad #\hfil
       &\save0\hbox par 1.5in
              {#}#\vtop{\unbox0}#
        \cr
this is a test
&of vtop     $\int ↓{-\infty}↑{+\infty}$
&Now is the time for
all good men to come to the aid of their
party.  The quick brown
fox jumps over the lazy dog.  Ha ha ha.
\cr
A second
&paragraph
&uses {\bigfont a larger} font in order to test
what is really going on and does this work?
\cr
}
\vfill\eject\end
```

which can be used to produce the following:

this is a test	of vtop $\int_{-\infty}^{+\infty}$	Now is the time for all good men to come to the aid of their party. The quick brown fox jumps over the lazy dog. Ha ha ha.
A second	paragraph	uses a larger font in order to test what is really going on and does this work?

Hanging punctuation

Problem #2: The term "hanging punctuation" describes punctuation symbols that extend into the margin so that line-initial and line-final letters will be aligned even when followed by punctuation. The following paragraph illustrates the technique:

> 'Now is the time for all good men to come to the aid of their party.' 'I hope this works.' "Do you also?" "Hope it works, I mean." 'In quotes.' 'In single quotes.' "In double quotes." How now brown cow? "The rain in spain stays mainly on the plain." This is a sentence. I should take the time to look for a nicely typeset book that has paragraphs, paragraphs, and paragraphs with hanging punctuation, so that I will have examples of characters, such as periods, commas, and quotation marks, that "protrude" into the margin. If I am lucky, this rather short, and very disjointed, paragraph will illustrate what I need to use to debug these macros.

The problem for this column is to prepare a file PUNCTUATION.TEX, so that the input shown below will produce the above paragraph. A solution will be given in the next issue.

```
\input punctuation

'Now is the time for all good men
to come to the aid of their party.'
'I hope this works.' ``Do you also?''
``Hope it works, I mean.''
'In quotes.' 'In single quotes.'
``In double quotes.''
How now brown cow? ``The rain in spain
stays mainly on the plain.''
This is a sentence.
  I should take the time to look for a
nicely typeset book that has paragraphs,
 paragraphs, and paragraphs with
hanging punctuation, so that I will
have examples of characters,
such as periods, commas, and quotation
marks, that ``protrude'' into
the margin.  If I am lucky, this
rather short, and very disjointed,
paragraph will illustrate what I need
to use to debug these macros.
\par\vfill\eject\end
```

Hint for Problem #2: Users of versions of TEX that do not provide the \ifx control sequence may want to modify the input slightly, by inserting backslash characters before punctuation symbols.

TUG TREASURER'S REPORT

August 31, 1982

	Budget 1982	Actuals as of 8/31	Estimated thru 12/31
Income:			
Membership/Publications			
1981 Back volume sales	$ 1,050	$ 1,125	$ 1,245
1981 and 1982 Membership[1]	9,500	10,625	10,775
1982 Library subscriptions[2]	450	300	360
1982 Foreign postage option	600	492	516
Supplements*,[3]	500	204	244
Meetings			
Cincinnati, 1/82	4,500	4,500	4,500
Meeting, Stanford, 7/82[4]	3,000	10,150	10,150
Short Course, Stanford, 7/82[4]	-0-	25,043	25,043
Manufacturers' Reps Fees, Stanford, 7/82	-0-	300	300
Institutional Membership[5]			
Educational*	3,750	200	1,000
Non-educational*	5,000	200	1,000
Other			
Videotape sales/rental	3,000	1,400	2,000
Advertising & mailing list sales*	1,000	-0-	-0-
Royalties (TEX manual)*	500	-0-	-0-
Total income	$ 32,850	$ 54,539	$ 57,133
Expenses:			
TUGboat (2 issues)[6]			
Printing	$ 1,800	$ 1,040	$ 2,100
Postage	700	392	800
Editorial services	4,920	1,749	4,200
Clerical services	2,950	2,340	4,000
Computer expense	3,310	1,528	3,100
Meetings			
Cincinnati, 1/82[7]	2,950	3,532	3,532
Stanford, 7/82	1,770	420	4,500
Other			
Supplements[3]	350	192	300
TEX distribution support[8]	8,500	-0-	-0-
ANSI meetings[9]	1,180	1,416	2,500
Legal and tax consulting	1,180	-0-	-0-
Advertising membership/TUGboat[10]	1,770	-0-	-0-
General mailings	710	696	1,000
Subsidies[11]	1,180	-0-	70
Printing, other	1,030	570	1,000
Miscellaneous[12]	2,360	826	2,000
Total expenses	$ 36,660	$ 14,701	$ 29,102
Summary:			
Balance forward, 1/1/82	$ (8,660)	$ (8,660)	$ (8,660)
Total income	32,850	54,539	57,133
Total expenses	36,660	14,701	29,102
Balance	$ (12,470)	$ 31,178	$ 19,371

Treasurer's Report

(Continued from preceding page)

Notes:

* These income categories were not budgeted in years prior to 1982, and therefore represent our best estimate.

All expense figures include an AMS overhead charge of 18%.

1. 1981 memberships were accepted through April 30, 1982; thereafter, individual issues are priced at $10 each (or $30 per volume/3 issues). As of September 10, 1982, there were 672 memberships/subscriptions, including libraries and complimentary.
2. Libraries may subscribe to TUGboat without applying for individual membership.
3. Lengthy descriptions of macro packages will be available for purchase separately.
4. Only the 2-day TUG meeting had been planned when the original budget was prepared. 62 members registered for the TUG meeting, 9 for the TEX82 Short Course, and 65 for both the meeting and course. (A combined fee was established for participation in both; $75 has been credited to TUG meeting income for each joint participant.)
5. Institutional membership was originally budgeted at $250 for educational and $500 for non-educational institutions. At the July 1982 meeting, a single rate of $200 was established.
6. Editorial services include programming, reviewing and editing; clerical services include maintaining the data base and mailing list, and other administrative duties.
7. This allocation partially covers expenses for participation by Don Knuth, Luis Trabb-Pardo, David Fuchs, Ignacio Zabala and Arthur Samuel in the TUG Cincinnati meeting, January 11–12, 1982.
8. Allocation to Stanford primarily for Professor Arthur Samuel, who is acting as TEX coordinator, answering questions, distributing tapes, and fixing bugs in the TEX source code. Another source of support has been found for this function, and the amount budgeted will not be required during 1982.
9. The Steering Committee authorized attendance by Lynne Price at one meeting of ANSI X3J6. Attendance at another meeting has been authorized.
10. Costs for advertising TUG membership in trade publications.
11. Money available to Finance Committee to subsidize travel and membership fees for individuals when appropriate.
12. Postage/express charges, telephone tolls and supplies, plus programmer and clerical services not associated with production of TUGboat.

Respectfully submitted,
Samuel B. Whidden, Treasurer

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

TUG BUDGET – 1983

Income:		
Membership/Publications		
1981 Back issue sales, 30@$10	$ 300	
1982 Back issue sales, 70@$15	1,050	
1983 Membership, 650@$20[1]	13,000	
1983 Library subscriptions, 30@$20[1]	600	
1983 Foreign postage option, 75@$12	900	
Supplements[2]	500	$16,350
Meetings		
Stanford, 3/83	8,000	
Fall '83[3]	7,000	15,000
Institutional Membership		
1983 Membership, 25@$200[1]	5,000	5,000
Other		
Videotape sales/rental	$ 3,000	
Advertising and mailing list sales	500	
Royalties (TEX manual)	500	4,000
Total income		$40,350
Expenses:		
TUGboat (2 issues)[4]		
Printing	$ 2,200	
Postage	800	
Editorial services	4,300	
Clerical services	4,000	
Computer expense	3,200	$14,500
Meetings		
Stanford, 3/83	4,960	
Fall '83[3]	4,960	9,920

Other		
Supplements[2]	$ 350	
ANSI meetings[5]	1,180	
Legal and tax consulting	500	
Advertising *TUG* membership & TUGboat[6]	1,500	
General mailings	1,100	
Printing, other	2,180	
Subsidies[7]	1,180	
Miscellaneous[8]	2,300	10,290
Total expenses		$ 34,710
Budget summary:		
Balance forward 1/1/83		$ 19,371
Total income		40,350
Total expenses		(34,710)
Estimated balance 12/31/83		$ 25,011

Notes:

All expense figures include an AMS overhead charge of 18%.

1. Numbers of members and subscriptions are based on present figures for individual memberships and library subscriptions, and on our best guess for institutional memberships.
2. Lengthy descriptions of macro packages will be available for purchase separately.
3. Although no formal plans have been made for a second general meeting in 1983, one is assumed for the Fall.
4. Editorial services include programming, reviewing and editing; clerical services include maintaining the data base and mailing list, and other administrative duties.
5. Support is budgeted for attendance at one meeting of ANSI X3J6.
6. Costs for advertising TUG membership in trade publications.
7. Money available to Finance Committee to subsidize travel and membership fees for individuals when appropriate.
8. Postage/express charges, telephone tolls and supplies, plus programmer and clerical services not associated with production of TUGboat.

Respectfully submitted,
Samuel B. Whidden, Treasurer

Contents

October 1982