

When problem solving is reduced to hacking, there's  
usually a problem in documentation somewhere.

Bruce Ormsby Adam  
Software Corner, *EP&P*  
(December 1988)

# TUGBOAT

COMMUNICATIONS OF THE **TEX** USERS GROUP  
EDITOR BARBARA BEETON

VOLUME 10, NUMBER 1  
PROVIDENCE • RHODE ISLAND

•

APRIL 1989  
U.S.A.

## **TUGboat**

During 1989, the communications of the T<sub>E</sub>X Users Group will be published in four issues. One issue will consist primarily of the Proceedings of the Annual Meeting.

TUGboat is distributed as a benefit of membership to all members.

Submissions to TUGboat are for the most part reproduced with minimal editing, and any questions regarding content or accuracy should be directed to the authors, with an information copy to the Editor.

### **Submitting Items for Publication**

The deadline for submitting items for Vol. 10, No. 2, is May 1, 1989; the issue will be mailed in July. (Deadlines for future issues are listed in the Calendar, page 117.)

Manuscripts should be submitted to a member of the TUGboat Editorial Committee. Articles of general interest, those not covered by any of the editorial departments listed, and all items submitted on magnetic media or as camera-ready copy should be addressed to the Editor, in care of the TUG office.

Contributions in electronic form are encouraged, via electronic mail, on magnetic tape or diskette, or transferred directly to the American Mathematical Society's computer; contributions in the form of camera copy are also accepted. For instructions, write or call Karen Butler at the TUG office.

An address has been set up on the AMS computer for receipt of contributions sent via electronic mail: **TUGboat@Math.AMS.com** on the Internet.

### **TUGboat Advertising and Mailing Lists**

For information about advertising rates, publication schedules or the purchase of TUG mailing lists, write or call Karen Butler at the TUG office.

## **TUGboat Editorial Committee**

Barbara Beeton, *Editor*

Ron Whitney, *Production Assistant*

Helmut Jürgensen, *Associate Editor for Software*

Laurie Mann, *Associate Editor on Training Issues*

Georgia K.M. Tobin, *Associate Editor, Font Forum*

Don Hosek, *Associate Editor for Output Devices*

Jackie Damrau, *Associate Editor for L<sup>A</sup>T<sub>E</sub>X*

Alan Hoenig and Mitch Pfeffer, *Associate Editors*

*for Typesetting on Personal Computers*

*See page 3 for addresses.*

## **Other TUG Publications**

TUG publishes the series *T<sub>E</sub>Xniques*, in which have appeared user manuals for macro packages and T<sub>E</sub>X-related software, as well as the Proceedings of the 1987 and 1988 Annual Meetings. Other publications on T<sub>E</sub>Xnical subjects also appear from time to time.

TUG is interested in considering additional manuscripts for publication. These might include manuals, instructional materials, documentation, or works on any other topic that might be useful to the T<sub>E</sub>X community in general. Provision can be made for including macro packages or software in computer-readable form. If you have any such items or know of any that you would like considered for publication, contact Karen Butler at the TUG office.

## **Trademarks**

Many trademarked names appear in the pages of TUGboat. If there is any question about whether a name is or is not a trademark, prudence dictates that it should be treated as if it is. The following list of trademarks which appear in this issue may not be complete.

**AMS-T<sub>E</sub>X** is a trademark of the American Mathematical Society.

**APS  $\mu$ 5** is a trademark of Autologic, Inc.

**METAFONT** is a trademark of Addison-Wesley Inc.

**PC T<sub>E</sub>X** is a registered trademark of Personal T<sub>E</sub>X, Inc.

**PostScript** is a trademark of Adobe Systems, Inc.

**T<sub>E</sub>X** is a trademark of the American Mathematical Society.

**UNIX** is a trademark of AT&T Bell Laboratories.

## Addresses

**Note:** Unless otherwise specified, network addresses (shown in typewriter font) are on the Internet.

**TeX Users Group Office**

P. O. Box 9506  
Providence, RI 02940-9506  
or  
653 North Main Street  
Providence, RI 02904  
401-751-7760  
TUG@Math.AMS.com

**Peter Abbott**

Computing Service  
Aston University  
Aston Triangle  
Birmingham B4 7ET, England  
21 359 5492  
pabbott@nss.cs.ucl.ac.uk  
Janet: abbottp@uk.ac.aston

**Stephan v. Bechtolsheim**

2119 Old Oak Drive  
W Lafayette, IN 47906  
317-463-0162  
svb@cssun.tamu.edu

**Lawrence A. Beck**

Grumman Data Systems  
R & D, MS D12-237  
Woodbury, NY 11797  
516-682-8478

**Barbara Beeton**

American Mathematical Society  
P. O. Box 6248  
Providence, RI 02940  
401-272-9500  
bnb@Math.AMS.com,  
Beeton@Score.Stanford.Edu

**Don Berryman**  
(see Bart Childs)

**Marius Broeren**  
Océ Nederland B.V.  
Division Office Automation  
P. O. Box 101  
5900 MA Venlo, The Netherlands  
+31.77.76466 x135

**Martin Bryan**  
29 Oldbury Orchard  
Churchdown  
Glos. GL3 2PU, United Kingdom  
+44 452 714029

**Lance Carnes**  
c/o Personal TeX  
12 Madrona Avenue  
Mill Valley, CA 94941  
415-388-8853

**S. Bart Childs**  
Dept of Computer Science  
Texas A & M University  
College Station, TX 77843-3112  
409-845-5470  
bart@cssun.tamu.edu  
Bitnet: Bart@TAMLSR

**Adrian Clark**

Department of Electronic Systems  
Engineering  
Essex University  
Wivenhoe Park  
Colchester, Essex CO4 3SQ, U.K.  
Janet: alien@uk.ac.essex.ese  
Janet: alien@uk.ac.kcl.ph.ipg

**Malcolm Clark**

Imperial College Computer Centre  
Exhibition Road  
London SW7 2BP, England  
Janet: texline@uk.ac.ic.cc.vaxa

**John M. Crawford**

Computing Services Center  
College of Business  
Ohio State University  
Columbus, OH 43210  
614-292-1741  
Crawford-J@Ohio-State  
Bitnet: TS0135@HSTVMA

**Jackie Damrau**

Mission Research Corporation  
1720 Randolph Road SE  
Albuquerque, NM 87106-4245  
505-768-7647  
damrau@dbitch.unm.edu  
Bitnet: damrau@bootes

**Michael DeCorte**

P. O. Box 652  
Potsdam, NY 13676  
315-268-2292  
mrd@sun.soe.Clarkson.edu

**Allen R. Dyer**

13320 Tridelphia Road  
Ellicott City, MD 21043  
301-243-0008 or 243-7283

**Shawn Farrell**

Computing Centre  
McGill University  
805 Sherbrooke St W  
Montréal H3A 2K6, Québec, Canada  
514-398-3676  
Bitnet: CCSF@MCGILLA

**Jim Fox**

Academic Computing Center HG-45  
University of Washington  
3737 Brooklyn Ave NE  
Seattle, WA 98105  
206-543-4320  
fox@uwavm.acs.washington.edu  
Bitnet: fox7632@uwacdc

**David Fuchs**

1775 Newell  
Palo Alto, CA 94303  
415-323-9436

**Richard Furuta**

Department of Computer Science  
University of Maryland  
College Park, MD 20742  
301-454-1461  
furuta@mimsy.umd.edu

**Regina Girouard**

American Mathematical Society  
P. O. Box 6248  
Providence, RI 02940  
401-272-9500 x224  
RMG@Math.AMS.com

**Raymond E. Goucher**

TeX Users Group  
P. O. Box 9506  
Providence, RI 02940-9506  
401-751-7760  
REG@Math.AMS.com

**Dean Guenther**

Computing Service Center  
Washington State University  
Pullman, WA 99164-1220  
509-335-0411  
Bitnet: Guenther@WSUVM1

**Michael A. Harrison**

Computer Science Division  
University of California, Berkeley  
Berkeley, CA 94720  
415-642-1469  
Harrison@Berkeley.edu

**Doug Henderson**

Division of Library Automation  
Office of the President  
University of California  
300 Lakeside Drive, Floor 8  
Oakland, CA 94612-3550  
415-987-0561  
Bitnet: dlatex@ucbcmfa

**Alan Hoenig**

17 Bay Avenue  
Huntington, NY 11743  
516-385-0736

**Don Hosek**

3916 Elmwood  
Stickney, IL 60402  
Bitnet: U33297@UICVM

**Patrick D. Ion**

Mathematical Reviews  
416 Fourth Street  
P. O. Box 8604  
Ann Arbor, MI 48107  
313-996-5273  
ion@Math.AMS.com

**Helmut Jürgensen**

Department of Computer Science  
 University of Western Ontario  
 London N6A 5B7, Ontario, Canada  
 519-661-3560  
 Bitnet: helmut@uwovax  
 uucp: helmut@julian

**David Kellerman**

Northlake Software  
 812 SW Washington  
 Portland, OR 97205  
 503-228-3383  
 uucp: imagen!negami!davek

**Richard Kinch**

Kinch Computer Co.  
 501 South Meadow St. (Route 13)  
 Ithaca, NY 14850  
 607-273-0222

**Jan van Knippenberg**

(see Marius Broeren)

**Donald E. Knuth**

Department of Computer Science  
 Stanford University  
 Stanford, CA 94305  
 DEK@Sail.Stanford.Edu

**Kim Kubik**

Syva Company  
 P. O. Box 10058  
 Palo Alto, CA 94303  
 415-493-2200

**C. G. van der Laan**

Rekencentrum RUG  
 Landleven 1  
 9700 AV Groningen, The Netherlands  
 +31/50 633374 or +31/50 633440  
 Bitnet: cgl@hgrrug5  
 DECnet: rugr86::cgl

**Pierre A. MacKay**

Northwest Computer Support Group  
 University of Washington  
 Mail Stop DW-10  
 Seattle, WA 98195  
 206-543-6259; 545-2386  
 MacKay@June.CS.Washington.edu

**Robert W. McGaffey**

Martin Marietta Energy Systems, Inc.  
 Building 9104-2  
 P. O. Box Y  
 Oak Ridge, TN 37831  
 615-574-0618  
 McGaffey%ORN.MFEnet@nmfecc.arpa

**Frank Mittelbach**

Fachbereich Mathematik  
 Universität Mainz  
 Staudinger Weg 9  
 D-6500 Mainz  
 Federal Republic of Germany  
 Bitnet: SCHOEPF@DMZNAT51

**Michael Modest**

College of Engineering  
 208 Mechanical Engineering  
 Pennsylvania State University  
 University Park, PA 16802  
 814-863-0976

**Dezső Nagy**

Geological Survey of Canada  
 1 Observatory Crescent  
 Ottawa K1A 0Y3, Ontario, Canada  
 613-995-5449

**James Nearing**

Department of Physics  
 University of Miami  
 Box 248046  
 Coral Gables, FL 33124  
 305-284-2323  
 NEARING%PHYVAX.SPANQUMIGW.MIAMI.EDU

**David Ness**

TV Guide  
 Radnor, PA 19088  
 215-293-8860

**David Osborne**

Cripps Computing Centre  
 University of Nottingham  
 Nottingham NG7 2RD, UK  
 +44 602 484848 x2064  
 dao@uk.ac.nott.cs@nss.cs.ucl.ac.uk  
 Janet: dao@uk.ac.nott.clan

**Mitch Pfeffer**

Suite 90  
 148 Harbor View South  
 Lawrence, NY 11559  
 516-239-4110

**Arnold Pizer**

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

**Craig Platt**

Department of Math & Astronomy  
 Machray Hall  
 University of Manitoba  
 Winnipeg R3T 2N2, Manitoba, Canada  
 204-474-9832  
 CSnet: platt@uofm.cc.cdn  
 Bitnet: platt@uofmcc

**Arturo Puente**

P. O. Box 88521  
 Caracas 1080, Venezuela  
 9086523

**David F. Rogers**

817 Holly Drive E. Rt. 10  
 Annapolis, Maryland 21401  
 dfr@usna.navy.mil

**Sriram Sankar**

Department of Computer Science  
 Stanford University  
 Stanford, California 94305  
 415-723-4962  
 sankar@score.stanford.edu

**Rainer Schöpf**

Institut für Physik  
 Johannes Gutenberg Universität  
 D-6500 Mainz  
 Federal Republic of Germany  
 Bitnet: SCHOEPF@DMZNAT51

**Larry Sharlow**

10 Toltec #3  
 Flagstaff, AZ 86001  
 602-774-1630

**Alan Stolleis**

(see Bart Childs)

**Yoichi Tanaka**

Toppan Printing Co., Ltd.  
 Electronic Publishing Division  
 5-1, 1-chome Taito, Taito-ku  
 Tokyo 110 Japan  
 (03) 835-5495

**Christina Thiele**

Canadian Journal of Linguistics  
 Carleton University  
 Ottawa K1S 5B6, Ontario Canada  
 Bitnet: WSSCAT@Carleton

**Klaus Thull**

Stephanstraße 25  
 D-1 Berlin 41  
 Federal Republic of Germany  
 8383536  
 unido!fubinf!thull@uunet.UU.NET

**Georgia K.M. Tobin**

The Metafoundry  
 OCLC Inc., MC 485  
 6565 Frantz Road  
 Dublin, OH 43017  
 614-764-6087

**Samuel B. Whidden**

American Mathematical Society  
 P. O. Box 6248  
 Providence, RI 02940  
 401-272-9500  
 sbw@Math.AMS.com

**Ron Whitney**

TEX Users Group  
 P. O. Box 9506  
 Providence, RI 02940-9506  
 rfw@Math.AMS.com

**Dominik Wujastyk**

Wellcome Institute for the History  
 of Medicine  
 183 Euston Road  
 London NW1 2BP, England  
 (01) 387-4477  
 dow@wjh12.harvard.edu

**Joost Zalmstra**

Vrije Universiteit  
 Amsterdam, The Netherlands  
 mcvax!nat.vu.nl!jjaz@uunet.UUCP

**Hermann Zapf**

Seitersweg 35  
 D-6100 Darmstadt  
 Federal Republic of Germany

## General Delivery

### Donald E. Knuth Scholarship

Larry Sharlow was honored at the 1988 Annual Meeting, McGill University, Montréal, as the 1988 Scholarship Winner. He has volunteered to serve on the 1989 selection committee.

We are pleased to announce the Fourth Annual "Donald E. Knuth Scholarship" competition. This year **two** Scholarships will be awarded. The awards consist of an all-expense-paid trip to TUG's 1989 Annual Meeting and the Short Course offered immediately following the meeting. The competition is open to all 1989 TUG members holding support positions that are secretarial, clerical or editorial in nature.

To enter the competition, applicants should submit to the Scholarship Committee by May 12, 1989, the input file and final *TeX* output of a project that displays originality, knowledge of *TeX*, and good *TeXnique*. The project may make use of a macro package, either a public one such as *LATEX* or one that has been developed locally; such a macro package should be identified clearly. Such features as sophisticated use of math mode, of macros that require more than "filling in the blanks", or creation and use of new macros will be taken as illustrations of the applicant's knowledge. Along with the *TeX* files, each applicant should submit a letter stating his/her job title, with a brief description of duties and responsibilities, and affirming that he/she will be able to attend the Annual Meeting and Short Course at Stanford University, Stanford, California, August 21-25, 1989.

Selection of the scholarship recipient will be based on the *TeX* sample. Judging will take place May 13-June 12, and the winner will be notified by mail after June 12.

All applications should be submitted to the Scholarship Committee at the following address:

Larry Sharlow  
10 Toltec #3  
Flagstaff, AZ 86001

### From the President

Bart Childs

I am looking forward to the celebration of 10 years of TUG at Stanford. The call for papers has already appeared; I hope you can and will participate.

Below is an announcement for a "dingbat" competition. I have had a lot of positive response from the preliminary versions. I hope the competition will be a good step in expanding our METAFONT horizons, which I feel have been neglected.

---

### Announcing A TUG Dingbat Competition

TUG announces a METAFONT competition for the creation of the best dingbat characters. Each entry will consist of one specific character (such as a logo) or a dingbat family, the METAFONT source, annotation of the source for pedagogical use, and samples of the use of the character(s). These characters can be in the spirit of the Zapf dingbats in PostScript, symbols, icons, logos, or of some other useful or entertaining nature.

A dingbat family could be:

- a character in different orientations (such as a hand or flag),
- a character in different presentations (such as outline, solid, black on white, white on black, gray, ...),
- a set of characters for doing border designs (TUGboat, Vol. 5, no. 2),
- a set of characters for a particular use (TUG could use an anchor, a dinghy, a printing press, ...), or
- any reasonably useful, entertaining, or interesting character.

TUG is holding this competition to encourage the use of complete *TeX* systems and to complement the initial system that was created by Don Knuth and given to us all. It is hoped that this competition will contribute to excellence in fonts, graphics, and documents in general.

## Prizes

- 1<sup>st</sup> Prize: \$250 plus autographed volumes A through E of *Computers & Typesetting*.
- 2<sup>nd</sup> Prize: \$100 plus two autographed volumes of choice.
- 3<sup>rd</sup> Prize: \$50 plus one autographed volume of choice.
- Five Honorable Mentions: autographed volume of choice.

## Conditions for Submission

1. The artist/author shall be the creator of the METAFONT sources and attest to the best of his/her knowledge that public distribution of these sources will not infringe on any known property rights. TUG will assist in efforts to determine that no property rights exist in any submission.
2. Each submission should be a dingbat family or a single character. Each individual may make as many submissions as desired, but each should be on a different theme. The METAFONT source shall be submitted electronically or on a standard diskette, PC or Mac format. Other formats can be approved on an *ad hoc* basis. The METAFONT source must be compatible with plain.mf; any additional METAFONT macros that are necessary must be furnished and become a part of the submission.
3. Each submission shall be accompanied by a text, in TeX, to explain its meaning, give examples of its use, and show the character in at least three sizes. These sizes should be appropriate for use in textual documents and screen display. A range of 10pt to 72pt is considered appropriate. This text will be the primary object studied to determine the awarding of prizes.
4. Each character or family shall include a discussion of one to five pages on considerations of design and METAFONT implementation. This discussion is for the purpose of pedagogy for other METAFONT users. The discussion should be appropriate for inclusion as a part of a possible companion to Volumes C and E, *The METAFONTbook* and *Computer Modern Typefaces*. Additional pages may be added for useful METAFONT macros. This source will also be considered as part of the basis for awarding of prizes.
5. TUG shall have the right to publish and use all submissions. Submissions shall carry

explicit permission for unlimited redistribution of the sources and characters on the same basis as TeX and METAFONT. This will include acknowledgement of the creator and TUG in the sources in all redistributions.

6. The artist/author's submission carries implied agreement to abide by the terms of this announcement. It is expected that the judging will be done by ballot of the members attending TUG 89 but this is subject to change at the discretion of the Finance Committee of TUG.

## Editorial Comments

Barbara Beeton

Once again it's time to get organized and put all of the good things I've seen before anyone else into a nice neat package. The news from all over the world is that TeX continues to be both a challenge and a useful production tool, and users are organizing regional interest groups as well as conferences of wider scope. See the calendar and the reports following it for details.

## TeX news

As has already been announced, Donald Knuth will be giving the keynote address at the tenth annual TUG meeting at Stanford, 21–23 August 1989, on the subject of "The errors of TeX". His paper on this subject will soon be published in *Software—Practice and Experience*. And occasional errors continue to be found (actually, I suspect the process owes more to careful excavation than to "luck" in most cases); the current versions are

TeX 2.97  
PLAIN.TeX 2.94  
METAFONT 1.7  
PLAIN.MF 1.7

Some changes have also been made to the CM fonts, in the files

CMBASE.MF  
BIGOP.MF  
ROMANL.MF  
SYM.MF  
SYMBOL.MF

These changes definitely affect all cmsy\* and cmex\* fonts; if you have the capability, you should regenerate these. Other fonts are also affected randomly

in various magnifications; the list is too long to reproduce here, and it may not be the same for different resolutions or settings of *blacker*. You could regenerate everything if you have the capability and patience, but unless you are a distributor, a service center, or are actively shipping .dvi files to other sites, you shouldn't have any problems if you wait for someone else to do the job.

The changes to *TeX*, METAFONT and the CM fonts are posted in the errata list supplement bound into this issue.

All changes have been communicated to all the implementors and distributors on my mailing list. If you are creating a new implementation of *TeX* and distributing it to other users, you should be receiving this information. Send me your name and address (preferably an electronic address accessible via the Internet), and a short description of the implementation you're working on.

#### **TUGboat selections on-line**

The directory <*TeX.TUGBOAT*> has been in existence at Score.Stanford.edu for some time now, and slowly, very slowly, items that have appeared in TUGboat are being installed for general use. The TUGboat header files are there, of course—*TUGBOT.STY* for use with Plain *TeX* and *LTUGBOT.STY* for *LATEX* (as an *article* sub-style). Several sample articles are present, along with a “driver” file that reads in the (plain) headers and then the article files: *TUGBOT.TeX* (the driver file), *TBOHYF.TeX* (the most recent edition of the hyphenation exception list) and *TBOCYR.TeX* (the article from TUGboat 6#3 that introduced the cyrillic and AMS extra symbol fonts).

Other items now available at Score include

- the .mf sources for Knuth's Punk fonts (TUGboat 9#2); 13 files named *PUNK\*.MF*
- .mf sources and the article and macros for APL fonts (TUGboat 8#3); files *APL\*.TeX*, *CMAPL10.\**, and *TBOAPL.TeX*
- tree diagrams by David Eppstein (TUGboat 6#1); files *TREEDEF.TeX* and *TBOTREE.TeX*
- *LATEX* macros for form letters, *MERGE.STY*, by Graeme McKinstry (TUGboat 8#1, revised)
- hyphenation patterns for German (*GERMAN-HYPH.TeX*), Icelandic (*ICEHYPH.TeX*) and Portuguese (*PORTHYPH.TeX*); if anyone knows of other available patterns, please let me know.
- complete TUGboat tables of contents, files *TB-CONT.DEF*, *TBvvyy.CNT* for volume vv published in year yy and two driver files to produce the contents for volumes 1–5 (*TBCV05.TeX*) and 6–10 (*TBCV10.TeX*)

The file *-CHRONO-.DIR* (there is one in every <*TeX*> subdirectory at Score) contains a chronological list of the directory contents. *-READ-.TUG* and *TUGFIL.CHG* describe the files in the directory and identify any updates that have been made.

These files are available from Score via anonymous FTP on the Internet. Copies have also been installed in the archives at Clarkson and Aston, and we are looking into ways to make them available from the TUG office for those who have no network access.

#### **More honors for Knuth**

On 7 October 1988, as part of the celebration accompanying the dedication of the new Center for Information Technology, Brown University awarded Donald Knuth an honorary Doctor of Science degree, in accordance with the following citation.

**Donald Ervin Knuth      Doctor of Science**

Computer science pioneer, mathematical typographer, developer of the art of programming, concrete mathematician, musician, and author, you are renowned in many areas for your uncompromisingly high standards and your seminal contributions. In your web you have caught the essence of computer science. In your books you have made profound and influential contributions to your fields. In your mathematical typography you have provided the glue for the expression of ideas, great and small. Through your music you have elevated spirits. Through your writings you have expressed artistic, surreal, and transcendental connections. You have been an inspiration to generations of colleagues and students. We honor you today as a distinguished scholar and scientist who epitomizes the academic life at its best.

Propter scientiam egregiam et famam universalem et in rebus computandis et in musica litterisque, auctoritate mihi commissa te ad gradum *in Scientia Doctoris* admitto, omniaque jura atque privilegia ad hunc gradum pertinentia tibi concedo. In huius testimonium hoc diploma tibi gravitate summa do.

October seventh

Howard R Swearer

1988

Præses

**UNIVERSITAS BRUNENSIS**

Providentiae in Rhodiensis Insulæ Republica

---

## A *TEX* Encounter in Japan

Barbara Beeton

On January 27, following a meeting in Tokyo of ISO/IEC JTC1/SC18/WG8 (whew!), the international Working Group whose interest is in standardization of office and publishing systems, and from which the SGML standard was issued, I attended a presentation devoted to introducing the family of WG8 standards to members of the Japanese manufacturing and business community. Two standards were discussed in some detail: SGML (ISO 8879-1986, Standard Generalized Markup Language) and SPDL (Standard Page Description Language, still a working paper). Two others, DSSSL (Document Style, Semantics and Specification Language) and Font Information Interchange (DIS 9541, an official draft, and the reason for my presence in Tokyo), were mentioned, but not described in detail.

The rationale for SGML is similar to that for L<sub>A</sub>T<sub>E</sub>X—a document has a logical structure that is separate from its presentation, and formally indicating this structure by markup enhances the value of the document content by making it transmittable and processable. (A couple of talks on SGML were on the program at the August '88 TUG meeting in Montréal, and can be found in the Proceedings.) In Tokyo, the focus was on how these standards could support the publication needs, both paper and electronic, of large organizations (one specific example was the Department of Energy of the U.S. Government), and some demonstrations of SGML-based products—an editor and a markup aid—using an example text in Japanese describing the tools and prepared particularly for the Tokyo audience.

All very interesting, but what does it have to do with *TEX*? It turns out that the paper copy of the SGML talk, by Martin Bryan of SOBEMAP in Belgium, and Yoichi Tanaka of Toppan Printing Company of Tokyo, was prepared using *TEX*, more specifically J<sub>L</sub>A<sub>T</sub>E<sub>X</sub>, as the formatter. This is a natural pairing of the two systems—SGML for structural markup, and *TEX* for formatting. For the demonstration, *TEX* was running on a workstation, and changes to the text could be processed immediately; an impressive display, as always. The authors kindly provided me with the original typeset output of the talk; the first page appears on page 9. To prepare this copy, *TEX* was run on a FACOM (Fujitsu) computer, and the output sent to a Hell Digiset phototypesetter at 900 dpi. The beginning of the input used to prepare this copy appears on

page 10; however, this is reproduced from a photocopy of (unknown) laser printer output, so the quality has suffered. Mr. Tanaka informed me that the *TEX* software was provided by the ASCII Corp. (a long-time member of TUG).

Earlier in the day, I'd had a chance to speak with Geoff Leach, a member of ASCII's publishing division staff. He informed me that the Japanese-language edition of *The TEXbook* will probably appear early next summer; I'm looking forward eagerly to seeing a copy. There seems to be a devoted, and growing, group of *TEX* users in Japan—there have been presentations on Japanese *TEX* at the last two TUG meetings—and a tailored edition of *The TEXbook* can only make it more accessible. Stay tuned.

---

## *TEX* Would Find It Difficult...

From Donald Knuth:

When preparing *Concrete Mathematics* I ran across [this] example of an unusual displayed formula in a paper by Joseph Bertrand (*Journal de l'École Royale Polytechnique* 18 (1845)). It's [an example] that PostScript will do with relative ease, while *TEX* and METAFONT must work harder. We could pose it as a problem to get the nicest output; the French typesetting of 1845 leaves something to be desired. Curiously, Bertrand says he chose the notation *pour plus de simplicité!*

Désignons, pour plus de simplicité, par

$$(1) \quad A_p \quad A,$$

*A*  
*A*  
*A*

Editor's note: Are there any takers?

# SGML/TeX 電子出版システム

Martin Bryan SOBEMAP/田中洋一 凸版印刷(株)

1989年 1月 25日

凸版印刷(株)はSOBEMAP社より SGML<sup>1</sup>システムを導入し、本格的文書データベースを使用した電子出版システムの開発に着手した。今回試作したシステムは、TeXを印刷システムとして利用したもので、高度な数式等を含む論文を手近なパソコンを使って印刷したり、CTSシステムを通じて本格的な印刷を行うと共に、CD-ROM等のデータベース出版にも容易に対応することが出来るものである。

## 1 標準マーク付け言語(SGML)

昨年は米国において DTP<sup>2</sup>がブームとなり、今年は日本のDTP元年となることが予想されている。ワードプロセッサは既に各家庭にまで入り込み、今や文書をコンピュータで処理することはまったく当たり前のことになっている。コンピュータを用いて印刷を行なうことは、比較的古くから行なわれていて、日本でも1970年の始めより、新聞社、印刷会社を中心に CTS<sup>3</sup>システムが開発された。現在では新聞記事のほぼ100%がコンピュータ処理されている。

ところで、これらのコンピュータデータである文書情報を扱う方法や環境は、現在でも多くの問題を抱えており、日増しにその重用度を増している。ワードプロセッサのデータ非互換問題は昔からいわれていることである。世界規模の情報化がすすむ中で、さらにこの問題は深刻になりつつある。

またビデオテックス、CD-ROM、などのような、多様なメディアによる出版もニーズが高まっている。このため文書情報は印刷だけではなく、多くのアプリケーションに応用できる必要が

あるのだが、現在の多くの文書情報は汎用性に乏しく、メディアを変えるたびに変換作業に多くの負荷をかけている。

これらの問題を解決するために SGML と呼ばれる文書情報の国際標準が開発され、既に 1986 年 10 月に国際規格 (ISO8879-1096) となっている。SGML は文書情報から組版指示を分離し、情報の本質である意味関係だけを記述し、文書データベースとしようとするものである。

SGML の目的として次のものが上げられる。

- 文書情報交換の保証。情報流通の促進。
- 文書情報の有効利用。
- アプリケーション非依存。
- システム非依存。
- 編集の効率化。
- 著作の効率化。
- 文書入力の効率化。
- 入力のシステム非依存。

同様の考え方をもったものに IBM の GML がある。名前のとおりこれは SGML のモデルとなったものである。GML は DCF<sup>4</sup>という組版システムに含まれている。組版情報を文書情報より分離しようというものの例として、文書整形システムである TeX や、Scribe, Troff などのマクロを使ったものがある。

TeX の数式の例

$$\sqrt{\frac{a+b+c}{x+y+\sqrt{2}}}$$

<sup>1</sup>Standard Generalized Markup Language

<sup>2</sup>Desk Top Publishing

<sup>3</sup>Computer Typesetting System

<sup>4</sup>Document Composition Facility

## 16 The TeX File

```
%documentstyle[a4j]{jarticle}
%title{SGML/TeX 電子出版システム}
%author{Martin Bryan SOBEMAP/田中洋一 凸版印刷（株）}
%begin{document}
%西暦
%maketitle
凸版印刷（株）はSOBEMAP社よりSGML
%footnote{Standard Generalized Markup Language}システムを導入し、本格的文書データベースを使用した電子出版システムの開発に着手した。今回試作したシステムは、%TeX を印刷システムとして利用したもので、高度な数式等を含む論文を手近なパソコンを使って印刷したり、CTSシステムを通じて本格的な印刷を行うと共に、CD-ROM等のデータベース出版にも容易に対応することが出来るものである。
%section{汎用組版指示言語(SGML)}
```

昨年は米国においてDTP

%footnote{Desk Top Publishing}がブームとなり、今年は日本のDTP元年となることが予想されている。ワードプロセッサは既に各家庭にまで入り込み、今や文書をコンピュータで処理することはまったく当たり前のことになっている。コンピュータを用いて印刷を行なうことは、比較的古くから行なわれていて、日本でも1970年の始めより、新聞社、印刷会社を中心にCTS  
%footnote{Computer Typesetting System}システムが開発された。現在では新聞記事のほぼ100%がコンピュータ処理されている。

ところで、これらのコンピュータデータである文書情報を扱う方法や環境は、現在でも多くの問題を抱えており、日増しにその重用度を増している。ワードプロセッサのデータ非互換問題は昔からいわれていることである。世界規模の情報化がすすむ中で、さらにこの問題は深刻になりつつある。

またビデオテックス、CD-ROM、などのような、多様なメディアによる出版もニード高涨まっている。このため文書情報は印刷だけではなく、多くのアプリケーションに応用できる必要があるのだが、現在の多くの文書情報は汎用性に乏しく、メディアを変えるたびに変換作業に多くの負荷をかけている。

これらの問題を解決するためにSGMLと呼ばれる文書情報の国際標準が開発され、既に1986年10月に国際規格(ISO8879-1096)となっている。SGMLは文書情報から組版指示を分離し、情報の本質である意味関係だけを記述し、文書データベースとしようとするものである。

SGMLの目的として次のものが上げられる。

```
%begin{itemize}
%item 文書情報交換の保証。情報流通の促進。
%item 文書情報の有効利用
%item アプリケーション非依存。
%item システム非依存
%item 編集の効率化
%item 著作の効率化
%item
```

## Software

### News from the VORTEX Project

Michael A. Harrison\*  
 Computer Science Division  
 University of California, Berkeley

#### Introduction

This is a report on the current state of the VORTEX project as it relates to the TeX community. The project has been running for over three years. Our research funding has been extended, but we will be evolving in non TeX-compatible ways. We expect to publish traditional research papers on the novel aspects of our work. Cf. [7] for a summary of a number of issues underlying this work. Design issues are discussed in [6].

This brief summary will tell you what software we will have available for use by the research community; it can be licensed commercially as well. By the time you read this report, all of the systems discussed should be available.

The original goal of the VORTEX project was to create an integrated document preparation environment capable of producing high quality technical documents which involve mathematics, text, and graphics. In VORTEX, both source and target representations of a document are maintained and presented. The source representation refers to a TeX document in its original unformatted form; the target representation presents its formatted result. The user can edit both representations using a text editor and what is called a proof editor, respectively. Our editor is an Emacs-like editor written in our own VLisp (VORTEX Lisp). Changes made to one representation propagate to the other version automatically. It is easy to support source modifications that cause the target representation to change, but a major challenge was the transformation from a target version back to the source version. The original design ideas are described in [6]. The system reformats a document and redisplays it on the screen incrementally. Only the part of the document or the subregion of the screen that is affected by recent changes is reprocessed. The document

---

\*Sponsored by the Defense Advanced Research Projects Agency (DoD), monitored by Space and Naval Warfare Systems Command, under Contract No. N00039-88-C-0292. Additional funding came from the California MICRO program (Grant 88-083 in conjunction with IBM).

environment has support for automatic production of indexes for books [8], support for bibliographic material [3], spelling checkers, and other document-related utilities. Cf. [5].

Our current system uses workstations with bitmap displays running under the UNIX operating system and the X window system [12]. Some subsystems also support SUNVIEW. In order to provide device independent high quality graphics, we use PostScript [1,2] and have written a PostScript interpreter. We are developing methods to interpolate PostScript into our displays. Future research plans include supporting composite objects, symbolic mathematics, hypertext documents, audio, live video, etc. A key element will be access to a persistent object base which is important for many applications.

The VORTEX prototype is running and did pass the traditional Trip test [10], as well as the more demanding Trip\* test required of incremental document processors [9]. Because the TeX processor is based on Pat Monardo's C-Tex, the system is fast. Some timing data are reported in [9]. The most challenging part of VORTEX is the reverse mapping mechanism. To handle some of the semantic difficulties, functions in VLisp are used to implement reverse mapping. Only a few are implemented in the current prototype.

The VORTEX prototype will be available on the distribution sometime in January 1989. As the system currently stands, it will require a great deal of work to make into an industrial-strength system. It served our purposes admirably because we found out what we did right (and wrong) and this will influence the design of its successor.

We have produced another system called INCTEX, derived from VORTEX, which we think will be useful immediately to the TeX community. INCTEX is an incremental TeX processor.<sup>1</sup> It is very fast and is more efficient in its use of memory than VORTEX. It supports quiescence checking, convergence testing, etc. (cf. [9] for definitions). It creates a DVI file and checkpoints some state information for each page on the first run. On subsequent runs, only those parts of the document which have changed are reprocessed. When running IATEX incrementally, no unnecessary passes are used. INCTEX, unlike VORTEX, is editor independent and is designed to avoid parts of the operating system which would hamper porting.

INCTEX has not passed the Trip\* test as of this writing, but works well enough to process many research papers and a 200-page dissertation. INCTEX

---

<sup>1</sup>Any flavor of TeX you want, not just plain TeX.

should be a very valuable extension to the family of TeX processors.

### Graphics

PostScript was chosen as the language for specifying graphics [2,1]. We have a PostScript interpreter which runs under X10(R2), and under SUNVIEW. This interpreter is available for distribution. The system involves a base interpreter and separate programs for interface to various window systems. Some of the algorithms used are original and will be published. While the interpreter is still only a prototype, it has been used in some commercial implementations, and can be extended and optimized. Perhaps some Berkeley MS students can be induced to do the remaining tasks (like porting to X11(R3)), but we have neither the resources nor the mission to develop commercial-level software. Several vendors have more robust PostScript systems, but we will distribute our source code and encourage you to improve on the interpreter.

### Fonts

As part of the PostScript interpreter project, we needed outline fonts. It was possible to write a PostScript program which calculates the coordinates of the points on the outline curves for any font which is native to or may be downloaded into a PostScript printer. From this data, one can automatically construct outline fonts using well known spline techniques. Thus, we have available 35 outline fonts which are on the distribution, together with the supporting software. These fonts use our own outlines but are *congruent* to some well known fonts. Table 1 lists equivalents for the more interesting outline fonts.

Having outline fonts is important, not only for the use of the interpreter. Of course, you can do illustrations like Figure 1. The A's in Figure 1 are from our t-rom font.

We can generate a full set of pk fonts for Table 1 for use with our previewers. We have an awk script which converts our outlines into the .cf format used by the TYPO editor [13]. This editor, written by Jakob Gonczarowski and marketed by Typography, Inc., is very useful. Among other features, it converts between various descriptions and font formats. Thus we can automatically convert from our outlines into Metafont files and hence into your favorite format.

### Other Systems

There are many other subsystems available and a brief outline of some of them is given below.

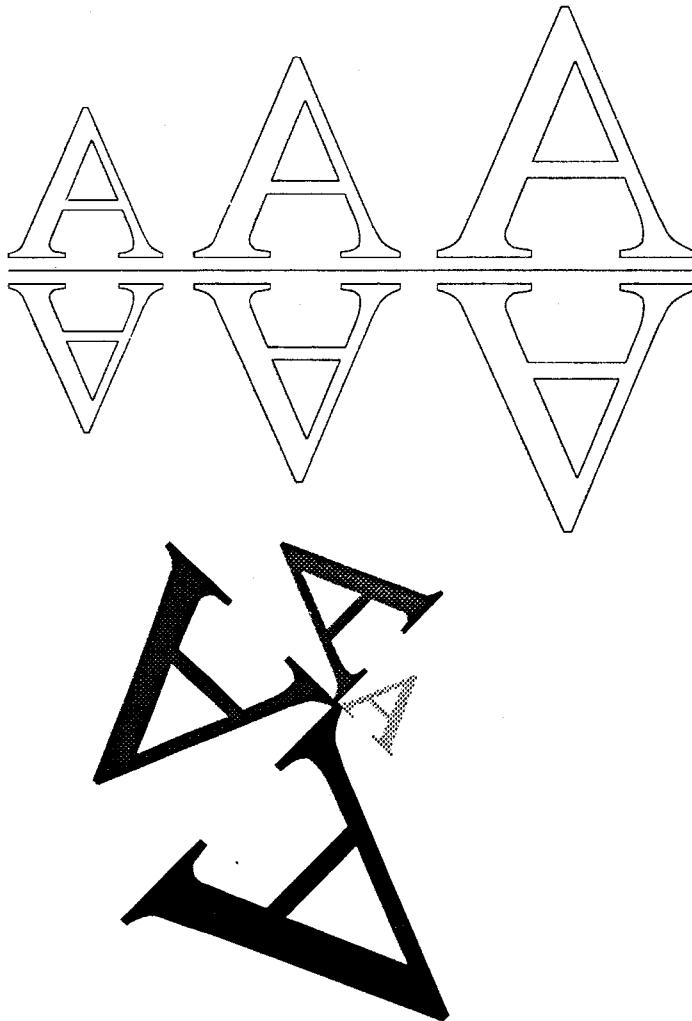
**Table 1:** Font Equivalents

Berkeley Fonts      Old Favorites

ag-book	AvantGarde-Book
ag-bookobl	AvantGarde-BookOblique
ag-demi	AvantGarde-Demi
ag-demobl	AvantGarde-DemiOblique
b-demi	Bookman-Demi
b-demiita	Bookman-DemiItalic
b-lig	Bookman-Light
b-ligita	Bookman-LightItalic
h-bol	Helvetica-Bold
h-bolobl	Helvetica-BoldOblique
h-med	Helvetica
h-obl	Helvetica-Oblique
ncs-bol	NewCenturySchlbk-Bold
ncs-bolita	NewCenturySchlbk-BoldItalic
ncs-ita	NewCenturySchlbk-Italic
ncs-rom	NewCenturySchlbk-Roman
p-bol	Palatino-Bold
p-bolita	Palatino-BoldItalic
p-bolobl	Palatino-BoldOblique
p-ita	Palatino-Italic
p-obl	Palatino-Oblique
p-rom	Palatino-Roman
t-bol	Times-Bold
t-bolita	Times-BoldItalic
t-bolobl	Times-BoldOblique
t-ita	Times-Italic
t-itaun	Times-ItalicUnslanted
t-mathita	Times-MathItalic
t-obl	Times-Oblique
t-rom	Times-Roman
zc-medita	ZapfChancery-MediumItalic
zd	ZapfDingbats

### Screen Previewers

DVItool is a previewer for DVI files which runs on the SUN workstation. This system is very robust, handles arbitrary DVI files, and provides a great many features. It is a full tool in the sense of the SUN window system and can be adjusted to any size the user finds appropriate. It is possible to keep a small window on the screen for previewing at the same time a source window is present. This is extremely valuable in debugging. Changing the view you have of a page is instantaneous. All magnifications are supported. Forward and reverse searching for strings in the DVI file is implemented as well as the ability to select a character and display its font. The previewer may be customized.



**Figure 1:** Some advantages of outline fonts.

DVI2x is a previewer for DVI files which runs under the X10 window system. It has a number of the same features as DVItool such as customizable key bindings. There are user definable functions, rulers, and side-by-side display of two pages. This program is quite fast.

DVI2X11 is a previewer for DVI files which runs under the X11(R3) window system. It is similar to DVI2x but is more than just a simple adaptation. The graphics have been improved and the scroll bars are draggable. This version is now faster than **texx**.

#### Environments

We distribute a large LISP library for use with **emacs**. We work with the **gnuemacs** version. This is intended for use with **TeX**, **IATEX** and **BIBTEX**. In addition to the most elaborate **TeX** mode known [4], the user is assisted in creating **.bib** files, and given forms-based assistance in filling in bibliographic en-

tries [3]. Functions exist for copying and duplicating fields from previous entries, etc. One particularly useful option is preparing a draft bibliography which includes numerical references, symbolic references and a formatted version of the entries. Another of the options allows previewing on the SUN or printing on any of your local printers. This system interfaces nicely with the DVI display programs mentioned above. It takes many pages of single-spaced text to explain all the options. Cf. [4,3,8]. There is also support for dealing with indices and for online-reference inspection. [11]

#### Licensing

Since we are a research project, we publish our work in scholarly publications and distribute our research software at a nominal charge. We have established site licenses of various kinds to simplify acquisition of the software.

We do ask that researchers who have found bugs or make significant extensions let us know about them.

In the past, commercial licenses have been negotiated separately, but this has proven to be too costly for us in terms of time. To simplify matters, we will now license the entire distribution for a fixed fee without royalties.

We are adding the new software to the distribution at this time and hope that it will be complete by the end of January 1989.

#### Credits

The following people have written or made major contributions to the software being distributed: Pe-hong Chen, John Coker, Michael A. Harrison, Paul N. Hilfinger, Dan Hydar, Jeffrey W. McCarrell, Ikuo Minakata, Pat Monardo, and Steven Procter.

SUNVIEW is a registered trademark of Sun Microsystems Inc. The X Window System is a registered trademark of M.I.T. Helvetica, Times, and Palatino are registered trademarks of the Allied Corporation. Avant Garde, Bookman, Dingbats, and Zapf Chancery are registered trademarks of the International Typeface Corporation.

#### References

- [1] Adobe Systems, Inc. *PostScript Language Manual*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1985. ISBN 0-201-10174-2.
- [2] Adobe Systems, Inc. *PostScript Language Manual: Tutorial and Cookbook*. Addison-Wesley Publishing Company, Reading, Massachusetts, 1985. ISBN 0-201-10179-3.

- [3] Pehong Chen. Gnu emacs BIB<sub>T</sub><sub>E</sub>X-mode. Technical Report 87/317, Computer Science Division, University of California, Berkeley, California, 1986.
- [4] Pehong Chen. Gnu emacs <sub>T</sub><sub>E</sub>X-mode. Technical Report 87/316, Computer Science Division, University of California, Berkeley, California, 1986.
- [5] Pehong Chen, J. Coker, Michael A. Harrison, J. McCarrell, and S. Procter. An Improved User Environment for <sub>T</sub><sub>E</sub>X. In J. Desarménien, editor, *<sub>T</sub><sub>E</sub>X for Scientific Documentation, Proc. 2nd European Conf. Strasbourg, France. June 19-21, 1986.* Lecture Notes in Computer Science, No. 236, pp. 32-44 (1986).
- [6] Pehong Chen, John L. Coker, Michael A. Harrison, Jeffrey W. McCarrell, and Steven J. Procter. The VORTEX document preparation environment. In J. Desarménien, editor, *<sub>T</sub><sub>E</sub>X for Scientific Documentation, Proc. 2nd European Conf. Strasbourg, France. June 19-21, 1986.* Lecture Notes in Computer Science, No. 236, pp. 45-54 (1986).
- [7] Pehong Chen and Michael A. Harrison. Multiple representation document development. *IEEE Computer*, 21(1):15-31, January 1987.
- [8] Pehong Chen and Michael A. Harrison. Index preparation and processing. *Software, Practice and Experience*, 18(9):897-915, September 1988.
- [9] Pehong Chen, Michael A. Harrison, and Ikuo Minakata. Incremental document processing. In *Proceedings of the ACM Conference on Document Processing*, New York, December 1988. Association for Computing Machinery.
- [10] Donald E. Knuth. A torture test for <sub>T</sub><sub>E</sub>X. Technical Report STAN-CS-84-1027, Computer Science Department, Stanford University, Stanford, California, November 1984.
- [11] Ethan V. Munson. Enhancements to Bibliography Management. Master's thesis, Computer Science Div., University of California at Berkeley, 1988.
- [12] Robert W. Scheifler and Jim Gettys. The X window system. *ACM Transactions on Graphics*, 5(2):79-109, April 1986.
- [13] Typographics, Ltd., Jerusalem. *TYPO Font Editing System, Reference Manual*, 1988.

---

## An Enhanced <sub>T</sub><sub>E</sub>X-Editor Interface for VMS

Adrian F. Clark  
Essex University

The author described various enhancements to the Stanford distribution of <sub>T</sub><sub>E</sub>X under VAX/VMS in a couple of previous articles. The most significant of these was to allow an editor to be invoked (by typing 'e' or 'E') when <sub>T</sub><sub>E</sub>X spotted an error in its input file. The editor which was interfaced to <sub>T</sub><sub>E</sub>X at that time was TPU. Since then, a number of requests have been made for interfaces to other editors, usually EDT. The author is pleased to announce that this has now been done, in a way which is fully compatible with the previous editor interface. This note discusses the interfaces to the editors usually encountered under VMS and shows how other editors can also be used.

VMS is supplied with a number of editors: EDT and, latterly, TPU are the most widely used, but Real Programmers and people who have used other DEC systems sometimes prefer TECO; those of us who remember the bad old days under VMS 1 will probably be familiar with SOS (which, although unsupported, can still be acquired through DECUS). There are also layered products which provide editors, such as LSE (language-sensitive editor) and a UNIX-compatible ed in DECshell. And, of course, there are third party and public-domain products—for example, Gnu Emacs and STE, the Software Tools editor.

The basic strategy for invoking an editor from <sub>T</sub><sub>E</sub>X is (in VMS jargon) to spawn a sub-process which executes a DCL command to edit the erroneous <sub>T</sub><sub>E</sub>X input file. When the corrections have been made, the editor is exited, the sub-process deleted and control returned to the <sub>T</sub><sub>E</sub>X session. However, sub-process creation and deletion is rather slow under VMS, so invoking editors in this way can give an unsatisfactory response on systems with a significant load. For this reason, both TPU and EDT are *callable*, i.e. they can be invoked as procedures from <sub>T</sub><sub>E</sub>X; this gives a much better response.

To decide which editor to use, <sub>T</sub><sub>E</sub>X looks at the logical name `TEX$EDIT` (by analogy with `MAIL` and other utilities). This should translate to the name of the editor to be used (see below). If the logical name is not defined, TPU is selected, for compatibility with the previous version of the editor interface. Following selection of the editor, <sub>T</sub><sub>E</sub>X translates the logical name `TEX$EDIT_INIT`, which should give the initialisation file for the appropriate editor. (For TPU, if `TEX$EDIT_INIT` does not exist, <sub>T</sub><sub>E</sub>X translates `TEX$TPU_INI`, again for compatibility with the

previous version.) What happens next depends on the chosen editor.

For TPU, a scratch file is created and the contents of any initialisation file copied to it. Commands are added to position the cursor at the place where *TEX* spotted the error. The scratch file is then used as the TPU initialisation file. After exiting the editor, this scratch file is deleted.

The process for EDT is similar: an initialisation file specified via *TEX\$EDIT\_INIT* is copied to a scratch file which is used to initialise EDT. However, it is not possible to position the cursor exactly at the erroneous text automatically (EDT is somewhat lacking in this respect), only to the right line. So the command sequence *GOLD M* is defined, which can be used to position the column correctly by hand.

Since both TPU and EDT are callable, but only one can be used in a particular *TEX* session, it is obviously somewhat inefficient to have both permanently linked (they are both quite large). Fortunately, both editors are implemented as *sharable images*. This allows *TEX* to determine which editor to use via *TEX\$EDIT*, then load the appropriate sharable image using the run-time library routine *LIB\$FIND\_IMAGE\_SYMBOL* before invoking the editor.

With the possible exception of LSE, which is TPU-based but not available to the author, the other DEC editors are not callable, and must be invoked, as would a non-DEC editor, by spawning a DCL command. Of the non-callable editors, only TECO can position the cursor in its initialisation file. However, input to TECO is split into pages (i.e., TECO makes a single pass through the file with a buffer of finite capacity), so it is not wise to position the cursor automatically. Instead, a macro is defined in q-register '1' to perform the positioning.

Any other editor is executed with a fixed sequence of command line arguments, separated by spaces: the file to be edited; the erroneous line; the erroneous column; and the initialisation file (if any). This allows a DCL procedure to be specified as *TEX\$EDIT*, permitting editor-specific processing. For example, the trivial procedure for use with SOS would be:

```
$ DEFINE/USER SYS$INPUT SYS$COMMAND:  
$ SOS 'P1'
```

The change file and editor-specific code for *TEX* 2.95 can be obtained by contacting the author at either *alien@uk.ac.essex.ese* or *alien@uk.ac.kcl.ph.ipg*. Both these addresses are on JANET, the U.K. academic network. The change file also features a large (>64K) memory, to enable the production of *PiCTEX* graphics and halftone images.

## The Virtual Memory Management of PubliC *TEX*

Klaus Thull

Last summer in *TEXeter*, I promised a public domain *TEX* for the PC. At that time I had solved the compiler related (arithmetic and idiosyncratic) problems and had passed the trip test. For a production version, capable of *LATEX*, *PiCTEX* and *AMS-TEX*, I still needed a Virtual Memory scheme which was promised me at *TEXeter* but never arrived. This I did then on my own, following some advice from "The Art of Computer Programming," tested it thoroughly, and completed a production version last autumn. For a while now this "PubliC *TEX*" is up and running, and has passed some few tests and productions.

This *TEX* does pass the trip test, I am proud to announce. On all accounts it is a fully developed specimen, capable of heavy work, and has proven reasonably stable. It can be configured with full memory and font space since these two are virtualized. The other table spaces must fit into real memory but even under Novell conditions which leave ca. 450-500kB this seems to be sufficient for generous sizes. The setting I use now has grown out of some experimenting with large runs in narrow conditions. Some of those large runs have been done with the new PubliC *TEX*.

As yet, this *TEX* is still slow. Its speed is ca. one fourth of that of its big commercial brother. On a 10MHz NCR AT-Compatible, it takes about 20 seconds for a plain page, and 30 for a *LATEX* page.

This *TEX* does not need the co-processor anymore. Since TURBO PASCAL's emulation knows only a 6-byte *real* datatype, some hand-coded conversion is used for *float* and *unfloat*.

*TEX* is accompanied by the complete *TEXware* and the complete GF- and PK-ware. MFT and META-FONT are still missing but I am working at that. I won't do anything about PXL-ware, yet I intend to do a PKtoCH/CHtoPK pair in order to have some font editing facility.

The entire sources are publicly available at *LISTSERV@DHDRZ1.BITNET*.

**The Compiler:** The compiler of my choice was Borland's TURBO PASCAL when version 4 was announced. This was the version introducing large memory model, multi-module compilation and 32-bit integers.

For once, I experienced a  $\mu$  compiler which deserves the name (but then, I am a spoiled

mainframe user). Where I do not have to wait many hours for a compilation. In fact, I have to wait 5 minutes of which 4 are TANGLE time. Where I didn't have to spend several weeks only to trace down compiler bugs. Where I/O, type conversion and arithmetic behavior is handled somewhat sensibly. Where code generated is decently small. *TeX*'s size is about 180k. Imagine my tears.

Of course, there are idiosyncrasies, and one (maybe-)bug. There was, of course, *line-break*'s standard feature turning *looseness* of -2 into 65534 which is displayed by every 16bit compiler. There was one new casting bug in *tangle* which spoiled *try-break*. Some more things like that.

Nevertheless, this compiler has passed its *TeX* test with all jets flaming, I think. A thousand thanks to its makers.

**Virtual Memory *TeX*:** *mem* and *font\_info* are the two tables virtualized. These two are the largest, and—alas—the ones accessed most, I suspect. I chose to devise a swapper governing one real memory page pool to serve both tables. This scheme might be extended to include other tables one day. Next on the list are INIT*TeX*'s hyphenation generator tables since for INIT*TeX* conditions do get narrow.

For testing memory access, I had an early, small, pre-VM version of *TeX*, sufficient for WEBs, output a memory access log for *mem* and *font\_info*. Then I took a couple of 10-20 page web logs as input data for statistics and simulation.

Here are some results of investigating said data as well as of experiments with the completed Virtual Memory *TeX*.

Basically, one printed page takes about 200000 memory accesses. This number of course grows for P*CTeX*, and also for huge paragraphs. The maximum record is held by my own prime number plotter with 6 million accesses, followed by a certain P*CTeX* page with 2 million accesses. About 3 to 4 consecutive accesses are on the same 256-cell memory page, in the average. This fact is yet to be exploited to construct the 'very fast' memory access.

*TeX*'s memory access behavior itself may be deemed "semi-local" which loosely means: of a row of consecutive memory accesses, some portion of them access a locally limited area. Over the long run, the area may change, but then the new area is another locality area. In the case of *TeX*, the access pattern is clear: the paragraph, the formula, the one macro under construction, each make for

hh	ighfkj	b	f	h	ah	fj
ii	ffifil		i		g	gj
ii	ilhjf		i		i	gk
ii	dkld	f	h		h	gk
ije	glie		h		g	fj
kihf	hl f		i		h	gj
ilhkc	hiji		i		i	gj
jilh	gijg		h		h	gj
iihg	ilh c		h		i	gj
ihgkjjji	jf g		i		h	gj
ih	kkffi	ig e	e	h	h	fj
eii	jlh i	g		i	i	gj
ihdilkdi	fg		h		j	gk
iifgklfifgdif		g	i		i	gk
ii	dikgicfk	h	i		g	gk
fhefhfighhhkhgfch		f			d	gei
dheeb	ejjcbcd					
dgcihfhi	ggb d					
bcbbhjidebegb		b				
dbe	ehfijhjed					
fifiijhjhghd	d					
ekichijhccigi	iegjia				h	fa
djie	ehgd	cihi	ghih		h	f
iiceikhcdcjeh		cejf			h	f
ijegkhh	dejdi	ekgf			h	g h
hh	gkif	cg h	je		g	g i
dii	jlhff	fh i	jgg		h	g
ii	gkkifg	jicifd	hjg		h	g c
chgchjjffcehee		degf				b e
chgejkjffchhee		ddfgf				bd fi ej
deekligi	ijghia	kg				e c h
cdcbjjfcf	gfbbbb	hb				d
ccbbhkf	e ffbb	gb				

**Figure U:** This is the memory access log over the last three pages of a 20 page .web (containing the index, the section list, and the TOC) using the original memory access scheme recorded in *The TeXbook*.

Each line logs, for 10000 accesses, their distribution over the 256-cell memory pages. Each letter denotes the  $\log_2$  of the number of accesses of that memory page (a:1, b:2, c:4 accesses, etc.).

This picture covers *mem*'s single node area only. The right part covers the macro area and the left part the character node area.

---

locality of accesses; non-locally there are macro, font, and glue references.

Investigating swap decision algorithms, the most important factor happened outside the swapper: locality gets lost over the run of printed pages. Free list gets scrambled, and after, say, five printed pages locality is virtually non-existent. To restore locality, I constructed a free list sorter. Indeed, on the PC the sorting decreased the number of page faults by 10% under favorable conditions (100 available memory pages) to  $1/2$  under narrow conditions (30 mem pages). Figures U and S, which

ljkkiaciajaflj	fh	h	ah	gj
kl	j	i	g	gj
m	k	i	h	gk
kl	j	h	h	fj
m	k	i	h	gk
lj	k	i	g	gk
m	j	h	i	gk
mk	j	i	i	gk
jl	k	i	h	gk
d l	j	ge e	i	h gj
jm	k	i	i	gk
m	j	i	j	gk
kl	k	i	h	gk
fe ddem	j	dfd h	i	d g ggk
ijgc	i	dh e		f
jj				
iji		d		
h c jj		dh		
gjjj				
ijji		d		
h				
jjjjjj		d		
jaahgaijklkkjj		ifhjia	h fa	f i
k	k	h jh	h f	g j
kk	k	i jg	h g	g j
hl	k	i jgf	h g h	hej
hkl	j	i kgg	h g	h j
hjldc	a	j i d hjg	h g	h j
ikii	f	h f fgf	ec	e f
ijkf	h	gf		
ijkjjjj	h	gf		
klhfgajjkjkjkkjdfka			bd	fibe
jlc	g	b bb i	e f	i
hk	f	f		
hjjjjkjjkjkjkkj		fhgf		d

**Figure S:** This image records the memory accesses during the same pages as in Fig. U to the same memory area, using the same recording conventions, but this time a free list sort is employed at the end of every *ship\_out*.

Note the locality visibly in effect now. Note also the increased number of accesses vs. the decreased number of accessed pages per line.

---

are compiled from corresponding mem access logs, demonstrate the difference between the unsorted and the sorted case.

Also, when I installed the free list sorter on PCS Cadmus which now is a paging system, it seemed to me that throughput increased by 10% while the sorter itself adds ca. 1% of CPU time. I would like to see this measured under controlled conditions (Size of the Working Set? Number of page faults?). At PCS, I cannot do that.

Memory page size of 256 cells (which is 1k) is just the right compromise. A larger size increases

the number of swaps while the average number of consecutive accesses of the same memory page never exceeds 4. A smaller size increases page translation table size which is now 3.3k (Memory page size  $\times$  page translation table size = const).

When memory page size is 256 cells and memory is 512k (which is likely under Novell) then INITEX obtains 12 pages swappable memory, and VIRTTEX about 120 pages. In that case, one average LATEX run takes about 100 memory page swaps per printed page. This is a tolerable low swap rate, I think, and so I won't spend too much time in speeding up swapping.

**1. PubliC T<sub>E</sub>X's Memory handler.** Here the two memory handling components are given in detail since they are the central part, I think, of the PC port. The solutions presented here may transcend TURBOPASCAL, and may allow for porting the WEB-to-C stuff, which is on the tape now, to small machines. Furthermore I would like to see others improve it.

A few details are left off, like most of the debugs, the procedure call cross referencing needed for TURBOPASCAL's *unit* mechanism, and the *use\_assembler* switch, since they just clog up the text without adding clarity.

```
format debug ≡ begin
format gubed ≡ end
format stat ≡ begin
format tats ≡ end
format fakebegin ≡ begin
format fakeend ≡ end
```

**2.** For a change, TURBO allows clean memory management due to an undocumented feature. This feature is not PASCAL as defined but, at least, it is cleaner than other constructs I saw used on 16bit machines. (O ye nameless compilers, get you gone into oblivion, and speedily.) If, say, *memp(x)* is a function returning a pointer of some type, then TURBO accepts *memp(x)↑ ← something*, and it does the right thing. This feature comes in handy here.

```
define mem(#) ≡ memp(#)↑
define font_info(#) ≡ fmemp(#)↑
{Types in the outer block 2} ≡
p_memory_word = ↑memory_word;
mem_pc_index = 0 .. max_mem_piece;
mem_piece = array [mem_pc_index] of
memory_word;
p_mem_index = p_mem_min .. p_mem_max;
p_fmem_index = 0 .. p_fmem_size;
```

```
mem_index = mem_min .. mem_max;
fmem_index = 0 .. font_mem_size;
```

See also section 5.

3. Here, of course, is the original reason for just that mem page size: these functions yield one-byte moves. Everything else would result in some kind of shift.

```
define mem_piece_size = 256
    { size of a memory piece, must be 256
      in order to use lo and hi }
define max_mem_piece = mem_piece_size - 1
    { min_mem_piece = 0 }
define mdiv(#) ≡ [hi](#)
define mmod(#) ≡ [lo](#)
define fdiv(#) ≡ [hi](#)
define fmod(#) ≡ [lo](#)
```

4. The swapper. T<sub>E</sub>X's *mem* and *font\_info* cannot be made smaller than, say, 70000 cells or 300kB memory if T<sub>E</sub>X is to be more than a toy. Yet, with the program proper being 180k and the other tables somewhere around 200k, this clearly exceeds a PC. So some form of memory pager must be provided.

We let *mem* and *font\_info* use same slot pool, same swapper, and same external memory. Later, when we build 64-bit T<sub>E</sub>X, INIT<sub>E</sub>X's hyphenation generator tables and/or *eqtb* may be included. One would do this by record variants on *contents*.

We let slot space and external memory grow with use.

```
define max_slots = 300
```

5. Central Intelligence Agency is the page translating table. It contains entries for the page slot allocated (or *no\_slot*) and the external memory index (or *no\_page*).

```
define no_slot ≡ nil
define no_page = -1
{ Types in the outer block 2 } +≡
time_stamp = integer;
page_p = ↑page_rec; slot_p = ↑slot_rec;
page_rec = record
    { page translation table record }
    slot_ptr: slot_p; { pointer into slot allocated,
        or no_slot }
    ext_nr: no_page .. 511;
        { pointer into external memory }
end;
slot_rec = record
    { inline memory page descriptor }
    page_ptr: page_p;
        { pointer into page allocated }
```

```
follower: slot_p; { to simplify traversing }
stamp: time_stamp; { or the RU-bit }
contents: mem_piece; { the actual page }
end;
```

6. These are the page translation tables for *mem* and *font\_info*.

```
{ Locals for virtual memory handling 6 } ≡
p_mem: array [p_mem_index] of page_rec;
p_font_info: array [p_fmem_index] of page_rec;
slot_rover: slot_p;
new_slot: slot_p;
slot_count: 0 .. max_slots;
    { number of slots allocated hitherto }
page_count: 0 .. 511; { number of external pages
    allocated so far }
clock: integer;
stat swap_no: integer; tats
```

See also sections 7 and 20.

7. Our external memory is on disk. We collect all the operations at this place so you can devise something different if you so wish.

```
define write_ext_mem_end(#) ≡
    fakebegin write(mem_file, #)
    end
format write_ext_mem_end ≡ end
define write_ext_mem(#) ≡
    begin seek(mem_file, integer(#));
    write_ext_mem_end
define read_ext_mem_end(#) ≡
    fakebegin read(mem_file, #)
    end
format read_ext_mem_end ≡ end
define read_ext_mem(#) ≡
    begin seek(mem_file, integer(#));
    read_ext_mem_end
define open_ext_mem ≡ set_ext_mem_name;
    assign(mem_file, name_of_file);
    rewrite(mem_file);
    write_ext_mem(0)(new_slot↑.contents);
define close_ext_mem ≡ close(mem_file)
```

```
{ Locals for virtual memory handling 6 } +≡
mem_file: file of mem_piece;
```

8. It is convenient to pre-allocate one slot to a convenient page, probably the *mem\_bot* page which is the first one to be accessed anyway. Actually, external memory must be opened here.

```
define for_all_mem_pages_do ≡
    for i ← p_mem_min to p_mem_max do
format for_all_mem_pages_do ≡ xclause
define for_all_fmem_pages_do ≡
    for i ← 0 to p_fmem_size do
```

```

format for_all_fmem_pages_do ≡ xclause
define first_page_no ≡ p_mem_min
⟨ Get virtual memory started 8 ⟩ ≡
  { initialize entire system }
for_all_mem_pages_do
  with p_mem[i] do
    begin ext_nr ← no_page; slot_ptr ← no_slot
    end;
for_all_fmem_pages_do
  with p_font_info[i] do
    begin ext_nr ← no_page; slot_ptr ← no_slot
    end;
  { now allocate first slot }
  new(new_slot);
  with new_slot↑ do
    begin follower ← new_slot;
    page_ptr ← addr(p_mem[first_page_no]);
    stamp ← 0;
    end;
  slot_rover ← new_slot; slot_count ← 1;
  { and connect it to first page, also acquire first
    page of external memory }
  open_ext_mem;
  with p_mem[first_page_no] do
    begin slot_ptr ← new_slot; ext_nr ← 0;
    end;
  page_count ← 1;
  clock ← 0;
  stat swap_no ← 0; tats

```

See also section 21.

**9.** We investigated five different swapping algorithms. Essentially, they are variants of the *First In First Out* (FIFO), the *Least Recently Used* (LRU) and the *Not Recently Used* (NRU) algorithms.

- The FIFO algorithm throws out the page which has been in memory longest.
- The LRU algorithm sets a time stamp per access and, in case of swapping, the slot with lowest stamp is thrown out. The subcases concern resetting of timestamp at swap time.
- The NRU algorithm sets a stamp per access and, in case of swapping, looks for a null stamp and clears a selection of stamp. The subcases concern the nature of that selection.

**10.** Only two of the algorithms studied so far turned out to be worthwhile, namely the LRU without clock reset, and the NRU following Knuth's modification. So these two we keep. The NRU showed ca. 5% more page faults than the LRU but is a trifle faster in the non-page-fault access. So in case there are few page faults and/or a fast swapper,

NRU might prove the faster, else LRU—contest is still open.

Objection to LRU may be the fear of the clock overflowing with huge or intricate jobs. The simple WEB file I logged showed ca. 200000 accesses per printed page, and, while I still wait for a chance to log a large table or a *PlCTEX* job, let's assume 1000000 accesses for a page at the worst, and you still have two thousand pages to go! Which leaves one to meditate on the magnitude of a 32-bit integer.

A variant not investigated yet is to step the clock at swap time only.

As it turned out, a *PlCTEX* page does take about two million accesses, and my own Third Root of Unity Primes Generator took six million accesses (and 12000 swaps).

```

define use_LRU ≡
define use_LRU_end ≡
define use_NRU ≡ @{
define use_NRU_end ≡ @}
format use_LRU ≡ begin
format use_LRU_end ≡ end
format use_NRU ≡ begin
format use_NRU_end ≡ end

```

**11.** These procedures describe the basic, non-swap access which must be fast. So I use **with** to stress that fact. Actually, this might be done in assembler, and *page\_ptr* and *slot\_ptr* kept in a register for further reference.

```

define not_in_memory ≡ (slot_ptr = no_slot)
define access_it(#) ≡
  begin { at this point, slot_ptr points
    to the in-memory page }
  # ← addr(contents[mmod(p)]);
  use_LRU stamp ← clock;
  use_LRU_end
  use_NRU stamp ← 1; use_NRU_end
  end

{ Include system and memory management
here 11 } ≡
{ I need fetch_mem here 13 }
function memp(p: pointer): p_memory_word;
  begin use_LRU incr(clock);
  use_LRU_end
  with p_mem[mdiv(p)] do
    begin if not_in_memory then
      fetch_mem(addr(p_mem[mdiv(p)]));
      with slot_ptr↑ do access_it(memp);
      end;
    end;
  end;

function fmemp(p: pointer): p_memory_word;

```

```

begin use_LRU incr(clock);
use_LRU_end
with p_font_info[mdiv(p)] do
  begin if not_in_memory then
    fetch_mem(addr(p_font_info[fdiv(p)]));
    with slot_ptr↑ do access_it(fmemp);
    end;
  end;

```

See also sections 17 and 18.

**12.** We describe the basic operations for swapping. Note the nesting of **with** clauses making for simpler expressions and (hopefully) faster programs.

```

define secutor(#) ≡ #↑.follower
define more_slots ≡ ((slot_count <
                      max_slots) ∧ (mem_avail > 10000))
define fakerepeat ≡
  { syntactic sugar for WEAVE }
define fakeuntil ≡
format fakerepeat ≡ repeat
format fakeuntil ≡ until
define rove_all_slots ≡
  begin s ← slot_rover;
  repeat with s↑ do
    fakeuntil
    fakeend
  define rove_slots_begin ≡
    begin fakeend
  define rove_slots_end ≡
    fakebegin fakerepeat fakebegin end;
    s ← secutor(s);
    until s = slot_rover
  end
format rove_all_slots ≡ xclause
format rove_slots_begin ≡ begin
format rove_slots_end ≡ end
define out_it ≡
  with page_ptr↑ do
    begin stat incr(swap_no); tats
      write_ext_mem(ext_nr)(contents);
      slot_ptr ← no_slot { disconnect page
                           from this slot }
    end
define in_it(#) ≡
  with #↑ do
    begin { argument is a page pointer,
            slot is on slot_rover }
      slot_ptr ← slot_rover; page_ptr ← #;
      { connect new page }
      if ext_nr ≠ no_page then
        read_ext_mem(ext_nr)(contents)
      else begin write_ext_mem(page_count)
               (contents);

```

```

ext_nr ← page_count;
incr(page_count);
end
end

```

**13.** This describes the outline of the swapping procedure. It is not required to be streamlined if swaps are minimized since slow anyway. Yet some indication is, again, given by the use of **with**.

```

{ I need fetch_mem here 13 } ≡
procedure fetch_mem(p : page_p);
var min_stamp: time_stamp; s, t: slot_p;
i: integer;
begin if more_slots then
  begin { Fetch a new slot, let slot_rover point
         to it 14 };
  with slot_rover↑ do in_it(p);
  end
else begin { decide which page to throw out,
            let slot_rover point to it }
  use_LRU { Use the LRU 15 }; use_LRU_end
  use_NRU { Use the NRU 16 }; use_NRU_end
  { up til now, nothing happened except
    slot_rover moving around }
  with slot_rover↑ do
    begin out_it;
      { the old page, that is. We assume, as
        in our TEX, that we cannot discern
        between read and write accesses }
      in_it(p); { the new one }
    end;
  end;
end;

```

This code is used in section 11.

**14.** This allocates a new slot.

```

{ Fetch a new slot, let slot_rover point to it 14 } ≡
begin new(new_slot);
with new_slot↑ do
  begin follower ← secutor(slot_rover);
  slot_rover↑.follower ← new_slot;
  end;
incr(slot_count);
{ now the new slot is officially present }
slot_rover ← secutor(slot_rover);
end

```

This code is used in section 13.

**15.** Least recently used.

```

{ Use the LRU 15 } ≡
begin min_stamp ← clock; t ← slot_rover;
rove_all_slots
rove_slots_begin if stamp < min_stamp then
  begin min_stamp ← stamp; t ← s;
  end;

```

```

rove_slots_end;
slot_rover ← t;
end

```

This code is used in section 13.

**16.** Not recently used. We realize Knuth's suggestion to switch off used-bits for those pages only that are touched during the search process. Pages whose bits stay on then may be termed "recently recently used."

```

define recently_used(#) ≡ (#↑.stamp ≠ 0)
define un_use_it(#) ≡ #↑.stamp ← 0

{ Use the NRU 16 } ≡
begin slot_rover ← secutor(slot_rover);
while recently_used(slot_rover) do
  begin un_use_it(slot_rover);
  slot_rover ← secutor(slot_rover);
  end;
end

```

This code is used in section 13.

**17.** At this place, external memory should be closed, deleted, freed or whatever. We output statistics.

```

{ Include system and memory management
here 11 } +≡
procedure close_mem;
begin close_ext_mem;
stat wlog_cr;
wlog_ln(`took', swap_no : 1, `swappings for',
clock : 1, `accesses on'); wlog(`',,
page_count : 1, `memory pages and',
slot_count : 1, `slots.');
tats
end;

```

**18. Reorganizing the free lists.** When we consider the various nodes strung out sequentially as allocated from the free lists then TeX's access is kind of local most of the time. It is clear: One paragraph of text is under consideration in one period of time, one formula, one batch of finished lines. In a paging environment (and most of the machines are today), such locality is an advantage: Consider the "Working Set", the collection of memory pages accessed during a certain period of time. With good locality, the Working Set needs be small only, and page faults few.

For TeX and other programs with similar memory management, the free list tends to be scrambled and scattered during the first few pages already so that any locality will be non-existent at all. Thus the Working Set may grow about a third again as large. The solution is to reorganize the free

list(s) at certain times such as to reflect physical neighbourhood again.

This amounts to a Sort. A full sort, however, is out of question, it may take up to 16 sweeps through the list. It is not necessary even, since there is no harm in a scramble inside a memory page. So we do one sweep with as many buckets as there are memory pages, then recombine. What follows, then, is straightforward. (Really? I did crash. Where, Dear Reader, I won't tell you. You find out as an exercise.)

The proper place for this to be inserted is right after the grand *free\_node\_list* at the end of *ship\_out*.

```

define mem_page(#) ≡ mdiv(#)

{ Include system and memory management
here 11 } +≡
procedure reorganize_free_lists;
var p, q, r, s, t: pointer; this_tail: pointer;
i, a_p: p_mem_index;
{ indices of memory pages }
v_p_min, v_p_max, s_p_min, s_p_max:
p_mem_index; { the single and variable
free list maximum page indices found so
far }

begin debug check_mem(false);
{ we suppose memory to be OK at this
point, I simply want the was_free bits set for
checking later }

gubed { Initialize free list reorganization 22 };
{ Distribute variable size free list to the separate
slots 23 };
{ Recombine variable size free list 24 };
{ Distribute single word free list 25 };
{ Recombine single word free list 26 };

debug check_mem(true);
{ Any non-trivial output here would mean
trouble, but, as it turned out, the program
crashed before reaching this point }

gubed
end;

19. define mem_page_avail ≡ m_p_avail
{ avoiding identifier conflict }
define mem_page_tail ≡ m_p_tail

20. { Locals for virtual memory handling 6 } +≡
mem_page_avail, mem_page_tail: array
[p_mem_index] of pointer;

21. { Get virtual memory started 8 } +≡
for i ← p_mem_min to p_mem_max do
begin { prepare the mem page buckets }
mem_page_avail[i] ← null;
mem_page_tail[i] ← null;
end;

```

**22.** { Initialize free list reorganization 22 } ≡  
 $p \leftarrow \text{get\_node}('10000000000');$   
{ re-merge them first thing right away }  
 $v\_p\_min \leftarrow \text{mem\_page}(\text{mem\_end});$   
 $s\_p\_min \leftarrow \text{mem\_page}(\text{mem\_end});$   
 $v\_p\_max \leftarrow \text{mem\_page}(\text{mem\_min});$   
 $s\_p\_max \leftarrow \text{mem\_page}(\text{mem\_min});$

This code is used in section 18.

**23.** It appears that *rover* is not supposed to be empty ever.

```
define insert_first_var_per_page ≡
begin mem_page_avail[a_p] ← p;
rlink(p) ← p; llink(p) ← p;
end
define insert_var_per_page ≡
begin r ← mem_page_avail[a_p];
s ← llink(r); rlink(s) ← p;
llink(p) ← s; rlink(p) ← r;
llink(r) ← p;
end
```

{ Distribute variable size free list to the separate slots 23 } ≡  
 $p \leftarrow \text{rover};$   
repeat  $q \leftarrow \text{rlink}(p)$ ;  $a_p \leftarrow \text{mem\_page}(p)$ ;  
if  $v\_p\_min > a_p$  then  $v\_p\_min \leftarrow a_p$ ;  
if  $v\_p\_max < a_p$  then  $v\_p\_max \leftarrow a_p$ ;  
if  $\text{mem\_page\_avail}[a_p] = \text{null}$  then  
insert\_first\_var\_per\_page  
else insert\_var\_per\_page;  
 $p \leftarrow q$ ;  
until  $p = \text{rover}$ ;

This code is used in section 18.

**24.** We clean up carefully behind us. One of those buckets may be reused very soon.

```
define append_this_var_list ≡
begin r ← mem_page_avail[i];
s ← llink(r); mem_page_avail[i] ← null;
t ← llink(rover); rlink(s) ← rover;
llink(rover) ← s; rlink(t) ← r;
llink(r) ← t;
end
```

{ Recombine variable size free list 24 } ≡  
 $\text{rover} \leftarrow \text{mem\_page\_avail}[v\_p\_min];$   
 $\text{mem\_page\_avail}[v\_p\_min] \leftarrow \text{null};$   
if  $v\_p\_max > v\_p\_min$  then  
for  $i \leftarrow v\_p\_min + 1$  to  $v\_p\_max$  do  
if  $\text{mem\_page\_avail}[i] \neq \text{null}$  then  
append\_this\_var\_list;

This code is used in section 18.

**25.** This must be considered part of the inner loop since every single character freed after printing gets through here.

```
define insert_first_avail_per_page ≡
begin mem_page_avail[a_p] ← avail;
mem_page_tail[a_p] ← avail;
link(avail) ← null;
end
define insert_avail_per_page ≡
begin r ← mem_page_avail[a_p];
mem_page_avail[a_p] ← avail;
link(avail) ← r;
end
```

{ Distribute single word free list 25 } ≡  
while  $avail \neq \text{null}$  do  
begin  $q \leftarrow \text{link}(avail)$ ;  
 $a_p \leftarrow \text{mem\_page}(avail)$ ;  
if  $s\_p\_min > a_p$  then  $s\_p\_min \leftarrow a_p$ ;  
if  $s\_p\_max < a_p$  then  $s\_p\_max \leftarrow a_p$ ;  
if  $\text{mem\_page\_avail}[a_p] = \text{null}$  then  
insert\_first\_avail\_per\_page  
else insert\_avail\_per\_page;  
 $avail \leftarrow q$ ;  
end

This code is used in section 18.

**26.** This code works even if *avail* has been empty in the first place.

```
define append_this_avail_list ≡
begin r ← mem_page_avail[i];
link(this_tail) ← r;
this_tail ← mem_page_tail[i];
mem_page_avail[i] ← null;
mem_page_tail[i] ← null;
end
```

{ Recombine single word free list 26 } ≡  
 $avail \leftarrow \text{mem\_page\_avail}[s\_p\_min]$ ;  
 $this\_tail \leftarrow \text{mem\_page\_tail}[s\_p\_min]$ ;  
 $\text{mem\_page\_avail}[s\_p\_min] \leftarrow \text{null}$ ;  
 $\text{mem\_page\_tail}[s\_p\_min] \leftarrow \text{null}$ ;  
if  $s\_p\_max > s\_p\_min$  then  
for  $i \leftarrow s\_p\_min + 1$  to  $s\_p\_max$  do  
if  $\text{mem\_page\_avail}[i] \neq \text{null}$  then  
append\_this\_avail\_list;

This code is used in section 18.

---

## TurboMETAFONT: A New Port in C for UNIX and MS-DOS

Richard J. Kinch

We have recently released TurboMETAFONT, a new, TRAP-certified implementation of METAFONT in the C language. Like our TurboTEX software, Turbo-METAFONT is portable to computers having C language compilers meeting ANSI or Kernighan & Ritchie standards and sufficient memory. We certified the software for MS-DOS PC's and AT&T UNIX System V in January. Certification for other targets is in progress.

Our earlier TUGboat article, "TurboTEX: A New Port in C for UNIX and MS-DOS" (TUGboat, vol. 9, no. 1, April 1988, pp. 48-52), described how we created TurboTEX from the Knuth autograph *tex.web*. The techniques and tools included Paschal, our Pascal-to-C translator program; a large TurboTEX change file; a few hand-written C functions to connect to the operating system environment; and utilities for splitting the large amount of C code generated into smaller, separately-compiled pieces. The earlier article describes these items in detail, so we will not repeat the particulars now.

To create TurboMETAFONT, we took the same approach as for TurboTEX, since the WEB programs for TEX and METAFONT are similar. A few items in METAFONT required some extra effort, but once the TurboMETAFONT change file was complete the same tools we perfected for the TurboTEX translation worked well for TurboMETAFONT.

### METAFONT Yields One Surprise.

One substantial translation issue did arise which was absent from TurboTEX but critical for Turbo-METAFONT. METAFONT uses integer arithmetic for all its calculations, while TEX uses some integer and some floating point. While the METAFONT syntax thus appears to be a subset of TEX's, in fact METAFONT makes a semantically broader use of integer arithmetic in ways that TEX never does.

For example, METAFONT takes differences here and there between *integer* variables (32 bits) and *halfword* variables and constants. Pascal prodigiously promotes all subrange integral types to the largest integral type when computing expressions. Thus, mixing of integral types in Pascal expressions is not prone to error, but is often inefficient. C economizes with an elaborate set of promotional protocols which attempt to minimize the word

length of the arithmetic instructions in the generated code. Thus, mixing of integral types in C can be optimally efficient for the machine's instruction set.

However, in C certain combinations of integral types are dangerous, and the programmer is expected to be wary of them. For example, an *unsigned int* in C is a variable which can only take a non-negative integer value. Subtracting one C *unsigned int* from another always yields an *unsigned int*-expression result. If the result should have been negative, the effect is tacit nonsense. The similar case in Pascal, namely using the *0..maxint* subrange type, works because Pascal first promotes the subrange components of the difference to signed integers before using them to compute expressions.

We did not want to translate the METAFONT Pascal code to C and completely impose the Pascal integer promotion on the C code; we prefer to retain the C efficiency. But there is no way for a translator to know from syntax or semantics whether a given expression in Pascal will have problems in C of the kind illustrated above. So we had to make a painstaking manual examination of each suspect expression (fortunately, these are not too prevalent in METAFONT), and analyze whether any possible run-time behavior required C type casting for correct results. The change file implements C type casts where needed.

### MS-DOS PC's: The Final Frontier.

Getting a 23,000-line program like METAFONT or TEX to run on a personal computer with limited memory is a big job. The portability traps of the IBM PC/Intel 8086 architecture and the MS-DOS operating system make the task still more imposing. We feel that Microsoft has done penance for inflicting MS-DOS upon the computer industry (and producing a 32-year-old billionaire as a side effect) by producing their C compiler, which makes MS-DOS look almost like UNIX to the programmer. This compiler significantly simplifies the task of porting to the PC.

The earlier TurboTEX software contained two features peculiar to the PC version: overlays and virtual memory simulation. At the time of the earlier article we had not completed these features, and so we will describe them now since they are complete, and also relate to TurboMETAFONT.

Both TurboTEX and TurboMETAFONT require overlays for the INITEX and INIMF versions of the executable programs. These versions are slightly larger than the production versions, since they contain initialization code otherwise absent. We

implemented optimally overlayed code by a method of run-time tracing. By adding a run-time trace feature to the Paschal translator, we obtain function-call histograms for the anticipated uses, namely, running INITEX to obtain a Plain TeX or LATEX format file, or running INIMF to obtain a Plain METAFONT or Computer Modern METAFONT base file. By overlaying the largest modules which are called only once or never during these runs, we obtain executables with near the performance of a non-overlayed version.

We developed a virtual memory simulator, released May 1987, for TurboTeX. The virtual memory feature allows a mainframe-sized TeX to run on PC's, which have an architectural and operating system limit of less than 600K bytes of usable program memory. By putting TurboTeX's largest array, *mem*, into simulated virtual memory we free enough space for a large version to run in the available PC memory. This technique succeeds with large macro packages and complex documents that cause other PC implementations of TeX to fail.

We do not now see the need to include the virtual memory simulator in the TurboMETAFONT programs. Since the TurboMETAFONT programs are smaller than TurboTeX, TurboMETAFONT will run on any PC with 640K memory installed. Furthermore, the enterprise of generating fonts with METAFONT does not seem to encourage the use of enormous macros or tables that accumulate from character to character, the way some applications of TeX quickly gobble memory for index entries or bitmap manipulation. We may still offer a virtualized version of TurboMETAFONT in the future for PC users, if commercial demand proves a need.

#### Portable Preloading.

Another TurboTeX feature not needed by Turbo-METAFONT is the portable preloading. This method in TurboTeX uses our utility program, FMT2INIT, to convert TurboTeX format files into C variable initializers, thus creating executable programs that start quicker with formats preloaded. Having such a preloaded executable is desirable for TurboTeX, because the TeX user typically makes many short runs of the program, so that the program load time dominates. The typical use of METAFONT, however, is for a lengthy run to generate an entire, existing, debugged font in a new magnification. Thus the value of preloading a TurboMETAFONT executable is small, and we have not carried over this feature.

#### Graphics Support.

A desirable feature of METAFONT is the online graphical output. This allows the user to view each character in a font, or even each stroke in a character, as METAFONT generates it. For the font developer, this interactive mode is critical to effective debugging of character programs. For the rest of us, it is just fun to watch the artistry in action.

On the PC, TurboMETAFONT supports the most common graphics adapters: the Hercules monochrome graphics adapter (HGA), the IBM color graphics adapter (CGA), and the CGA-compatible modes of the IBM extended graphics adapter (EGA) and virtual graphics adapter (VGA). We wrote the HGA driver code ourselves, but used the CGA/EGA/VGA drivers in the Microsoft C compiler's graphics library. We provide a separate executable program for each graphics adapter type to minimize memory use and increase execution speed. If there is sufficient demand we may incorporate Tektronix 4010 graphics output as an option for applications on larger (non-PC) computers.

#### Distributing the Product and Enlarging the TeX Community.

We have drawn much encouragement from the market acceptance of TurboTeX, having shipped over 500 copies in the first six months of publication with only TUGboat advertising. We will continue our approach of one low price for a complete package, now to include both TurboTeX and Turbo-METAFONT, with complete source code available.

The TurboTeX emphasis on the pure C language and portability has served to significantly enlarge the number of TeX users. There is a large and growing body of potential TeX users who share a set of characteristics: They are programmers who work with C, they own or use a PC, they know and admire Knuth's *The Art of Computer Programming* series, they need a technical typesetting system, they have heard about TeX, and they know the shortcomings of other "desktop publishing" programs. TurboTeX, we have found, appeals to such people as a good fit in all these ways to get them started as serious TeX users. We plan to broaden our approach to appeal to these unTeX'ed masses, and perhaps they will soon be the new generation of TeX users.

---

## The **TEX** PostScript Software Package

Stephan v. Bechtolsheim

### Introduction

This report describes the **TEXpS** Software Package. **TEXpS** stands for **TEX**-PostScript. This package contains, among other things, a new DVI to PostScript driver and the facility to use PostScript fonts in **TEX**.

The software's copyright is in the GNU spirit (Free Software Foundation, Cambridge, MA): it is free except for a nominal distribution charge for those of you who order the software because of the lack of access to ftp. The software must be passed on according to the same terms, also if you port the software to a new operating system, for instance. See the end of this article for more information on how the software is distributed.

In this short report I will describe the following programs:

1. *dvitps*, the driver, which converts DVI files into PostScript files<sup>1</sup>.
2. *pdf2tfm*, a program to generate TFM files for PostScript fonts which makes PostScript fonts accessible in **TEX**.
3. *printpdr*, an auxiliary program which will be discussed later.

### Features of the **TEXpS** Software Package

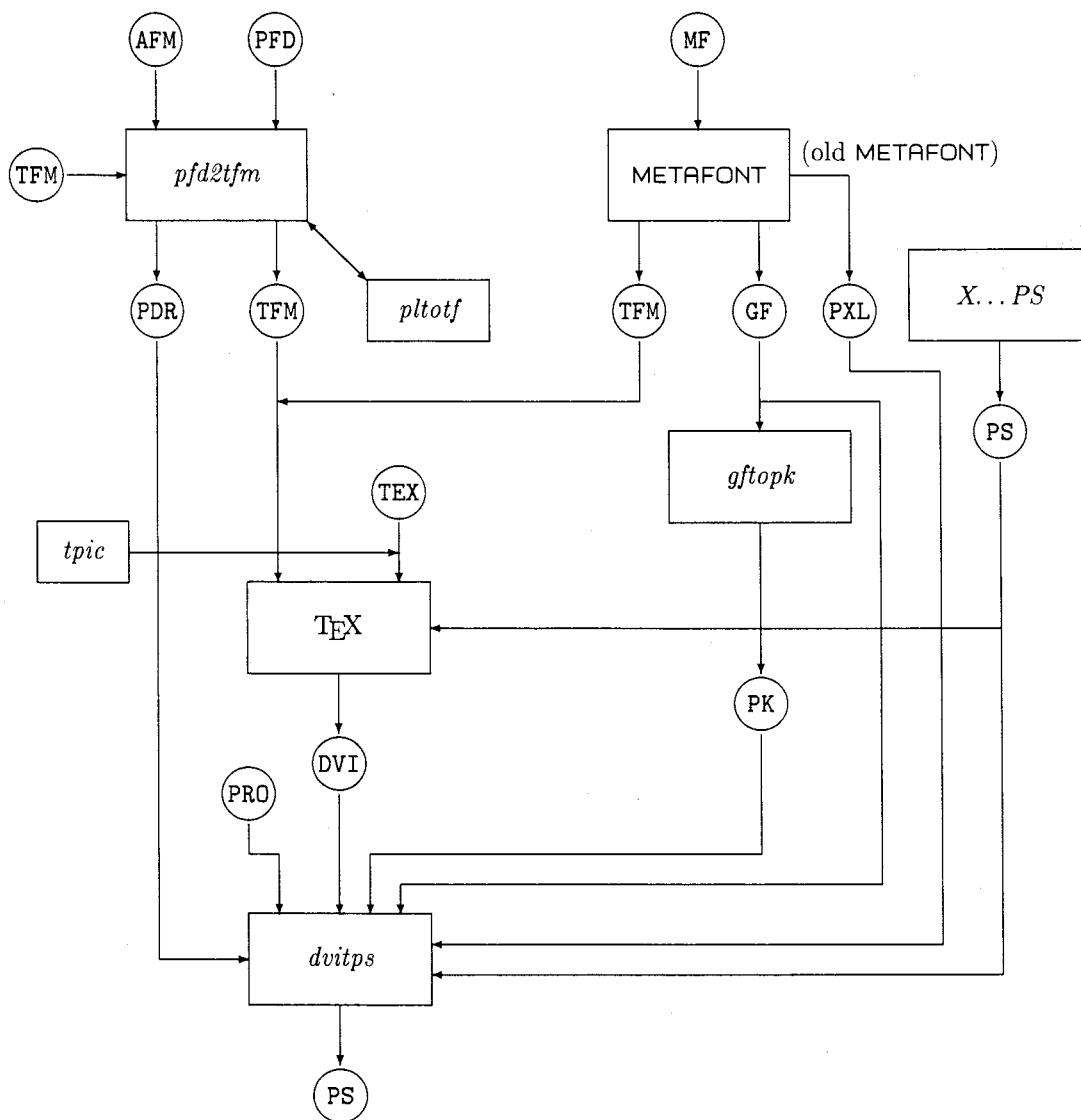
Here is a list of features of the **TEXpS** Software Package. The main emphasis below will be on *dvitps* which is the central part of this software package.

1. The software runs under UNIX (currently tested on VAXes and SUNs (BSD 4.x)); ports to other flavors of UNIX should be straightforward.
2. The software is very easy to install. A central file contains all local definitions and a small C program is used to update all makefiles.
3. The software comes with a *150 page manual* which explains it thoroughly. The manual contains numerous examples intended to be a testbed for the software.
4. *Manual pages* (done in *troff*...) are included.
5. The software was *tested carefully* at various locations before its release and should perform according to its specification. Continuous support by the author is offered.

---

<sup>1</sup> The driver is not called *dvi2ps* because there are already so many different PostScript drivers called *dvi2ps* floating around.

6. *dvitps* allows the use of the usual pixel based fonts of **TEX**. All three pixel file types PXL, GF and PK are supported. Fonts may have up to 256 characters.
7. *PostScript fonts* can be used in **TEX**! The program *pdf2tfm* converts AFM files into TFM files which then can be accessed by **TEX**. Ligature and kerning information from AFM files is used by *pdf2tfm*. This program also generates PDR files which are accessed by *dvitps*.
8. PostScript fonts can be *tilted*, *stretched*, and *modified* in various other ways.
9. The *encoding vector* of the PostScript fonts used is under complete user control. In other words, every character of a PostScript font can be mapped to any character code offering full flexibility. Defaults are provided, making the encoding vector of PostScript fonts as close as possible to the encoding vectors of the usual **TEX** fonts. The encoding vector is among the information stored in the previously mentioned PDR files.
10. Encoding vectors can be printed out using the program *printpdr*.
11. It is *dvitps* and not the PostScript interpreter inside the printer which is "in charge of positioning." *dvitps* generates only integer position coordinates and movements. The term 'integer' here is used with respect to the resolution of the PostScript printer. As a consequence it was possible to make *dvitps* follow the rules of *dvitype*, the model device driver, precisely. *dvitps* includes, for instance, a maxdrift control.
12. *dvitps* has an option to *reset the virtual memory of the PostScript printer* used every *n* pages.
13. *dvitps* has an option to *show all attempts to locate font files*.
14. *dvitps* has an option to *generate "conforming documents"*, i.e. PostScript files conforming to a standard defined by Adobe, Inc. This allows PostScript files generated by *dvitps* to be included into other PostScript files.
15. *Font emulation* is provided. Font emulation means that the spacing and positioning of characters within **TEX** refers to one TFM file but printing by *dvitps* is done by using a font with a different TFM file. This can be used to print documents which were processed with TFM files for which no pixel files are available. This situation occurs when native fonts of phototypesetters are used and the producer of the phototypesetter does not make any pixel files available.



A data flow diagram explaining the connection between *TeXPS*, *TeX* and *METAFONT*.

16. PostScript fonts can be extended so that *arbitrary PostScript procedures can be executed* in place of certain character codes.
17. The *inclusion of PostScript figures in T<sub>E</sub>X documents* is possible with macros of the psfig macro package.
18. *Figures generated with tpic* can be included into a T<sub>E</sub>X document.
19. *X Window screen dumps* (X11.3) can be included into a T<sub>E</sub>X document.

### An Overview

The data flow diagram contains an overview of the T<sub>E</sub>XPS Software Package. Observe that this figure contains as subdiagram the flow of information when generating fonts with METAFONT and processing a document by T<sub>E</sub>X using METAFONT based fonts.

The left side of the figure shows the programs added by T<sub>E</sub>XPS. The *pfd2tfm* program generates TFM files from AFM files and PFD files. PFD files contain, for instance, encoding vector specifications. The resulting TFM files are read in by T<sub>E</sub>X to produce a document with PostScript fonts. The *pfd2tfm* program may read in TFM files itself for the purpose of font emulation. The *pfd2tfm* program also produces PDR files which are read in by *dvitps*. These PDR files contain width information and encoding vector information which is needed by *dvitps*.

Observe also that *tpic* can be used with our *dvitps*. *tpic* generates a T<sub>E</sub>X source code file which in turn generates \specials which are included into the DVI file.

Finally with the help of the psfig macros, PostScript based figures generated by some program X...PS can be included into a T<sub>E</sub>X document. Such a program should but does not need to generate a bounding box comment in its PostScript output which is read in by the psfig macros to reserve the proper space in the T<sub>E</sub>X document. The rest of the PostScript file is discarded by T<sub>E</sub>X, and only read in by *dvitps*.

PRO files mentioned in the figure are prologue files which contain certain standard PostScript definitions read in by *dvitps*.

### Availability

The software of the T<sub>E</sub>XPS package consists of the following parts:

1. Source code for the three basic programs *dvitps*, *pfd2tfm* and *printpdr*.
2. The L<sup>A</sup>T<sub>E</sub>X source of a 150 page document describing the software.
3. Manual pages.

4. The basic set of AFM files needed to produce the corresponding TFM and PDR files.
5. The sources of *pltotf* (needed by *pfd2tfm*) and of *makedd*, a program to compute dependencies of C program sources.

The software is made available in the following two ways:

1. Use ftp to the machine `csseq.tamu.edu` and retrieve the file `dvitps.tar.Z` from there.
2. Send \$80.00 (\$110.00 outside the continental US, shipped air mail) to Stephan v. Bechtolsheim, 2119 Old Oak Drive, West Lafayette, IN 47906 and you will receive a SUN cartridge plus a complete set of the documentation.

### Acknowledgements

The software was developed by the author as part of a consulting contract with the Menil Foundation in Houston, Texas. It was a combined decision by the author of this software and the Menil Foundation to make this software available to the T<sub>E</sub>X community at large.

My email address is `svb@cssun.tamu.edu`. Feel free to contact me for further information.



# A Handy Little Font

This short communication presents the code for a couple of dingbats that the author has found useful in memos and other correspondence. The code is quite straightforward, and can easily be put to use by the reader on any METAFONT implementation. Despite the simplicity of the code, there are a couple of interesting things done which I will enlarge upon a bit when we get to them.

The first part of the code looks like this:

```
% hands.mf
mode_setup;
size=48pt#;
font_size size;
em#:=size; cap#:=7/10em#; desc#:=3/10em#;
thinline#:=1/100em#;
define_pixels(em,cap,desc);
define_blackter_pixels(thinline);
```

Here we assign values to the height, depth and width of the character box and define the single pen size to be used. Since this is a very simple font, there is no call for overshoots or multiple pens; and the height and depth of the character box is just expressed as a fraction of the width.

Now, we define the whole character in a macro:

```
%Hand pointing right
def handpointing=
% define points for thumb and cuff
x1=x3=1/2[0,1/15w];
x2=x5=x4=x23=4/16w;
y1=y2=10/15[-desc,cap];
y3=y4=2/15[-desc,cap];
y5=6/7[y4,y2]; y23=1/7[y4,y2];
x6=9.75/16w;
y6=y2;
x7=11.25/16w;
y7=4/5[y23,y5];

x8=8.75/16w;
y8=1/4[y7,y6];
x17=14.5/16w;
y17=9.25/15[-desc,cap];
% find a point at a certain height on
% the curve from z6 to z7
path dummyCurve; path dummyLine;
x.dummy=1/2[x8,x7];
y.dummy=y17;
dummyCurve:=z6z5-z2..z7..tension1.4..z8;
dummyLine:=z.dummy--z17;
z18=dummyCurve intersectionpoint dummyLine;
x16=x17;
y16=y7;
x9=7/16w;
y9=y8;
x10=6/16w;
y10=2/5[y23,y5];
% find another point on the
% curve from z6 to z7
x.dummy2=x5;
y.dummy2=y16;
x.dummy3:=1/2[x8,x7];
y.dummy3=y.dummy2;
dummyLine:=z.dummy3--z.dummy2;
z12=dummyCurve intersectionpoint dummyLine;
% define points for curled fingers
x15=x14=x19=x22=1/3[x18,x17];
x13=x20=x21=x12;
y15=y16;
y13=y14=y15-(y17-y16);
y20=y19=y13-(y17-y16);
y21=y22=y20-(y17-y16);
% pick up pen and draw whole image
pickup pencircle scaled thinline;
draw z1--z2--z4--z3--cycle;
draw z5(1,1)..tension 1.5..z6
&z6z5-z2..z7..tension 1.4..z8
```

```

zz8down..tension3..z9
zz9..tension 1.8..leftz10;
draw z18--z17right..z16--z7;
draw z7--z15right..z14--z13left..z12;
draw z14right..z19--z20left..z13;
draw z19right..z22--z21left..z20;
draw z21(-1,-1)..tension1.5..z23;
enddef;

```

The interesting bits I referred to at the start of this article are the lines in which  $z_{18}$  and  $z_{12}$  are defined.  $z_{18}$  is the point at which the top of the pointing finger touches the top of the thumb;  $z_{12}$ , the point at which the top of the second finger touches the underside of the thumb. Notice how METAFONT calculates the precise point of intersection easily, and obviates the need for fudging by the user.

Using this macro, we can write the code for a character depicting a hand pointing right in a few short lines; these lines, in fact:

```

beginchar("A",16/15em#,cap#,desc#);
handpointing;
endchar;

```

When we run this (and recall, *size* is set equal to 48 points), we get:



It would be nice to have a dingbat of a hand identical to the above except pointing left. We could describe such a character fairly easily by rewriting the code with all horizontal coordinates shifted; for example, the line:

```
x1=x3=1/2[0,1/15w];
```

would be rewritten:

```
x1=x3=1/2[w,14/15w];
```

But, there is a much simpler solution. We can write a file that redefines *endchar* in the same way that I discussed in "The ABC's of Special Effects" (*TUGboat*, Vol. 9, No. 1, April 1988, pp. 15-18):

```

% pattern.mirror
def pattern=
def endchar=
tracingequations:=1;
cullit;
picture normalchar;
normalchar:=currentpicture;

```

```

picture mirrorimage;
mirrorimage:=normalchar
reflectedabout ((0,0),(0,h))
shifted (w,0);
currentpicture:=mirrorimage;
scantokens extra_endchar;
chardx:=w;
shipit;
if displaying>0: showit; fi
endgroup;
enddef;
enddef;

```

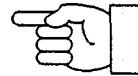
This redefinition will, in essence, capture the image shown above, reflect it about the y-axis, and shift it to the right by the character width. We can input that file within another character definition, call *pattern*, and then call our *handpointing* routine:

```

beginchar("B",16/15em#,cap#,desc#);
input pattern_mirror;
pattern;
handpointing;
endchar;

```

Running the above (and recall, size is set equal to 48 point) we get:



We can make use of another file providing another definition of *pattern*, namely:

```

% pattern_rev
def pattern=
def endchar=
tracingequations:=1;
cullit;
picture phaseone; phaseone=currentpicture;
currentpicture=nullpicture;
fill (0,-desc)--(w,-desc)--(w,cap)--
(0,cap)--cycle;
cullit;
picture phasetwo;
phasetwo=currentpicture-phaseone;
currentpicture:=phasetwo;
scantokens extra_endchar;
chardx:=w;
shipit;
if displaying>0: showit; fi
endgroup;
enddef;

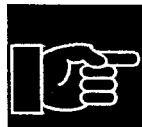
```

```
enddef;
```

This redefinition captures the image drawn above and removes each pixel in it from an entirely blackened character box. As before, we input that file within another character definition, call *pattern* and then call the *handpointing* routine; thus:

```
beginchar("C",16/15em#,cap#,desc#);
input pattern_rev;
pattern;
handpointing;
endchar;
```

and get:



Finally, we can use another file that combines the features of the previous two patterns:

```
% pattern_revmirror.mf
def pattern=
def endchar=
tracingequations:=1;
cullit;
picture phaseone; phaseone=currentpicture;
currentpicture=nullpicture;
fill (0,-desc)--(w,-desc)--(w,cap)--
(0,cap)--cycle;
cullit;
picture phasetwo;
phasetwo=currentpicture-phaseone;
picture phasethree;
phasethree:=phasetwo
```

```
reflectedabout ((0,0),(0,h))
shifted (w,0);
currentpicture:=phasethree;
scantokens extra_endchar;
chardx:=w;
shipit;
if displaying>0: showit; fi
endgroup;
enddef;
enddef;
```

and which, accessed by this code:

```
beginchar("D",16/15em#,cap#,desc#);
input pattern_revmirror;
pattern;
handpointing;
endchar;
```

gives us the reverse video mirror image of the original character:



The user can, of course, change the second line of the code shown above to create the font at any size desired; you might wish to actually put that line outside the rest of the code, and simply specify it at run time.

To create the font, all you need is the code shown in this article and METAFONT. Start up METAFONT, specify the device for which you are creating the font by typing `mode=lowres;` (or `mode=` whatever you wish) and then inputting *hands.mf*.



## Typesetting Concrete Mathematics

Donald E. Knuth  
Stanford University

During 1987 and 1988 I prepared a textbook entitled *Concrete Mathematics* [1], written with co-authors Ron Graham and Oren Patashnik. I tried my best to make the book mathematically interesting, but I also knew that it would be typographically interesting—because it would be the first major use of a new typeface by Hermann Zapf, commissioned by the American Mathematical Society. This typeface, called AMS Euler, had been carefully digitized and put into METAFONT form by Stanford's digital typography students [8]; but it had not yet been “tuned up” for real applications. My new book served as an ideal test case, because (1) it involved a great variety of mathematical formulas; (2) I was highly motivated to make the book readable and attractive; (3) my experiences with tuning up Computer Modern gave me insights into how to set the mysterious font parameters used by TeX in math mode; and (4) the book was in fact being dedicated to Leonhard Euler, the great mathematician after whom the typeface was named.

The underlying philosophy of Zapf's Euler design was to capture the flavor of mathematics as it might be written by a mathematician with excellent handwriting. For example, one of the earmarks of AMS Euler is its zero, ‘0’, which is slightly pointed at the top because a handwritten zero rarely closes together smoothly when the curve returns to its starting point. A handwritten rather than mechanical style is appropriate for mathematics because people generally create math with pen, pencil, or chalk. The Euler letters are upright, not italic, so that there is a general consistency with mathematical symbols like plus signs and parentheses, and so that built-up formulas fit together comfortably. AMS Euler includes seven alphabets: Uppercase Roman (ABC through XYZ), lowercase Roman (abc through xyz), uppercase Greek (ΑΒΓ through ΧΨΩ), lowercase Greek (αβγ through χψω), uppercase Fraktur (ΑΒϹ through ΩΩΩ), lowercase Fraktur (abc through ρηζ), and uppercase Script (ΑΒϹ through ΧΨΖ). It also includes two sets of digits (0123456789 and 0123456789), as well as special characters like ϕ, ℶ, and some punctuation marks. Details about its design are discussed in another article [7].

To refine the digitized characters for mathematical use, I began by correcting the way they appeared in their “boxes,” from TeX's viewpoint. For this purpose I used the \math tests of the standard

testfont routine [5, Appendix H]; these tests put the characters through their basic paces by typesetting formulas such as  $|A| + |B| + |C| + \dots + |Z|$ ,  $a^2 + b^2 + c^2 + \dots + z^2$ ,  $a_2 + b_2 + c_2 + \dots + z_2$ , and  $\hat{A} + \hat{B} + \hat{C} + \dots + \hat{Z}$ . I noticed among other things that the Fraktur characters had all been placed too high above the baseline, and that more blank space was needed at the left and right of the characters in subscript/superscript sizes. After fixing such problems I also needed to set the italic corrections so that subscripts and superscripts would have proper offsets; and I needed to define suitable kerns with a \skewchar so that accents would appear visually centered. AMS Euler contains more than 400 characters, and Hermann had made them beautiful; my job was to find the right adjustments to the spacing so that they would fit properly into mathematics.

The next step was to design a set of TeX macros so that AMS Euler could be used conveniently in math mode. This meant adding new “families” to the conventions I had defined in previous formats. Plain TeX typesets mathematics with four basic font families (namely, \fam0 for text, \fam1 for math italic, \fam2 for symbols, and \fam3 for large delimiters), plus a few others that are used less frequently (\fam4 for text italic, \fam5 for slanted text, \fam6 for boldface roman, and \fam7 for typewriter style). I added \fam8 for AMS Euler Script and \fam9 for AMS Euler Fraktur; the AMS Euler Greek and Roman went into the old position of math italic, \fam1. Hermann had designed new parentheses and brackets, which were bundled together with the Fraktur fonts; therefore my macro file changed plain TeX's conventions by defining, e.g., \mathcode`('="4928 and \delcode`("=928300. Similarly, Euler has the symbol ‘≤’ as an alternative to ‘≤’, packaged with the Script alphabet; to make TeX recognize this substitution I said \mathchardef\leq="3814 \let\le=\leq.

With such additions to the plain TeX macros I could type formulas like \$tan(x+y)\$ as usual and get not ‘tan(x+y)’ but ‘tan(x+y)’. There was, however, one significant difference between typing the manuscripts for *Concrete Mathematics* and for *The Art of Computer Programming*, caused by the fact that the Euler numerals 0123456789 are distinctly different from the numerals 0123456789 in ordinary text. In previous work, I used to “optimize” my typing by saying, e.g.,

\$x\$ is either 1 or -\$1\$,

thereby omitting \$'s around a mathematical constant unless I needed them to get a minus sign instead of a hyphen. After all, I reasoned, those extra \$'s just make TeX work harder and the result looks the same; so why should I be logical? But in *Concrete Mathematics* I needed to type

\$x\\$ is either \\$1\\$ or \\$-1\\$,

to keep x from being 'either 1 or -1'. The early drafts of my manuscript had been prepared in the old way; therefore I needed to spend several hours laboriously hunting down and correcting all instances where the new convention was necessary. This experience proved to be worthwhile, because it taught me that there is a useful and meaningful distinction between text numerals and mathematical numerals. Text numerals are used in contexts like '1776' and 'Chapter 5' and '41 ways', where the numbers are essentially part of the English language; mathematical numerals, by contrast, are used in contexts like 'the greatest common divisor of 12 and 18 is 6', where the numbers are part of the mathematics. (Authors of technical texts in languages like Japanese, where Arabic numerals are used in formulas but not in ordinary text, have always been well aware of this distinction; now I had a chance to learn it too.)

As I was tooling up to begin using AMS Euler, my publishers were simultaneously showing the preliminary manuscript of *Concrete Mathematics* to a book designer, Roy Howard Brown. I had sent Roy a copy of the first Euler report [8] so that he could see examples of the typeface we planned to use for mathematics. Our original intention, based on Zapf's original plans when he began the design in 1980, was to use Computer Modern Roman for the text and AMS Euler for the mathematics. But Roy noticed that AMS Euler was somewhat darker in color than a traditional mathematical italic, so he decided that the text face should be correspondingly heavier. He sent me several samples of typefaces with more suitable weights, so that I could prepare a special font compatible with AMS Euler. (One of my basic premises when I had developed the Computer Modern meta-font was that it should be readily adaptable to new situations like this.) When I saw Roy's samples, I decided to pursue something that I'd wanted an excuse to do for several years, namely to find settings for the parameters of Computer Modern that would produce an "Egyptian" (square-serif) style.

The cover designs for *Computers & Typesetting*, Volumes A-E, show a gradual transition of

the respective letter pairs Aa, Bb, Cc, Dd, and Ee from the style of standard Computer Modern Roman to an Egyptian style. I had made these cover designs just for fun, at the suggestion of Marshall Hendricks, but I had never had time to experiment with a complete text face in that style. Now I had a good reason to indulge that whim, and after a pleasant afternoon of experiments I found a combination of parameters that looked reasonably attractive, at least when I examined samples produced by our laser printer. (I magnified the fonts and viewed them from a distance, to overcome the effects of 300 dpi resolution.) Then I made more elaborate samples of text and printed them on Stanford's APS phototypesetter, to see if the new fonts would really pass muster. Some characters needed to be adjusted (e.g., the 'w' was too dark), but I was happy with the result and so was Roy.

I decided to call the resulting font Concrete Roman, because of its general solid appearance and because it was first used in the book *Concrete Mathematics*. (In case you haven't guessed, the text you are now reading is set in Concrete Roman.) *There also is Concrete Italic, a companion face that is used for emphasis in the book.* EVEN STRONGER EMPHASIS IS OCCASIONALLY ACHIEVED BY USING A CONCRETE ROMAN CAPS AND SMALL CAPS FONT. Anybody who has the METAFONT sources for Computer Modern can make the Concrete fonts rather easily by preparing parameter files such as *ccr10.mf*, analogous to *cmr10.mf*; you just need to change certain parameter values as shown in the accompanying table.

Here are samples of Concrete Roman in the 9 pt, 8 pt, 7 pt, 6 pt, and 5 pt sizes:

Mathematics books and journals do not look as beautiful as they used to. It is not that their mathematical content is unsatisfactory, rather that the old and well-developed traditions of typesetting have become too expensive. Fortunately, it now appears that mathematics itself can be used to solve this problem.

A first step in the solution is to devise a method for unambiguously specifying mathematical manuscripts in such a way that they can easily be manipulated by machines. Such languages, when properly designed, can be learned quickly by authors and their typists, yet manuscripts in this form will lead directly to high quality plates for the printer with little or no human intervention.

A second step in the solution makes use of classical mathematics to design the shapes of the letters and symbols themselves. It is possible to give a rigorous definition of the exact shape of the letter "a", for example, in such a way that infinitely many styles (bold, extended, sans-serif, italic, etc.) are obtained from a single definition by changing only a few parameters. When the same is done for the

other letters and symbols, we obtain a mathematical definition of type fonts, a definition that can be used on all machines both now and in the future. The main significance of this approach is that new symbols can readily be added in such a way that they are automatically consistent with the old ones.

Of course it is necessary that the mathematically-defined letters be beautiful according to traditional notions of aesthetics. Given a sequence of points in the plane, what is the most pleasing curve that connects them? This question leads to interesting mathematics, and one solution based on a novel family of spline curves has produced excellent [sic] fonts of type in the author's preliminary experiments.

We may conclude that a mathematical approach to the design of alphabets does not eliminate the artists who have been doing the job for so many years; on the contrary, it gives them an exciting new medium to work with. [2]

Heavier weight makes the type more resilient to xeroxing and easier to read in a poorly lighted li-

brary, so these new typefaces may help solve some of the legibility problems we all know too well. But a typeface that is too bold can also make a book tiresome to read. To avoid this problem, Roy decided to use a \baselineskip of 13 pt with 10 pt type. This gives an additional advantage for mathematical work, because it prevents formulas like  $\sum_{0 \leq k < n} a_k^d$ , in the body of the text from interfering with each other; the normal 12 pt baselineskip used in most mathematics books can get uncomfortably tight. Of course, the increased space between lines also increases the number of pages by about 8%; this seems a reasonable price to pay for increased readability.

Table of parameter values for Concrete fonts, using the conventions found on pages 12–31 of [6]:

name	cmr10	ccr10	ccr9	ccr8	ccr7	ccr6	ccr5	ccslc9	ccti10	ccmi10	ccsc10	lower
font_ident	CMR	CCR	CCR	CCR	CCR	CCR	CCR	CCSLC	CCTI	CCMI	CCCSC	
serif_fit	0	1	1	1	1	1	1	0	1	1	1	
cap_serif_fit	5	3	2.8	2.6	2.4	2.2	2	2	3	3	3	2
x_height	155	165	148.5	132	115.5	99	82.5	155	165	165	155	116
bar_height	87	92	78.3	69.6	60.9	52.2	43.5	85	92	92	87	65
tiny	8	11	10	9	8	7	6	9	11	11	11	
fine	7	6	6	6	6	6	5	6	6	6	6	
thin_join	7	17	17	15	13	12	11	13	17	17	17	
hair	9	21	20	19	17	15	14	16	21	21	21	
stem	25	25	24	22	20	18	16	22	24	25	25	23
curve	30	27	26	24	21.5	19	17	23	26	27	27	
ess	27	25	24	22	20	17	12	25	24	25	25	
flare	33	29	26	24	23	20	18	28	28	29	29	22
cap_hair	11	21	20	19	17	15	14	16	21	21	21	21
cap_stem	32	27	26	24	21.5	19	17	23	26	27	27	24
cap_curve	37	28	27	25	22.5	20	18	24	27	28	28	26
cap_ess	35	27	24	22	21.5	19	14	23	26	27	27	24
bracket	20	5	5	4	4	3	3	5	5	5	5	
jut	28	30	27	24	21	19	17	15	30	30	30	
cap_jut	37	32	29	26	23	20	18	16	32	32	32	24
vair	8	21	20	19	17	15	14	15	21	21	21	
notch_cut	10	5/6	3/4	2/3	7/12	1/2	5/12	3/4	5/6	5/6	5/6	
bar, etc.*	11	21	20	19	17	15	14	15	21	21	21	21
cap_notch_cut	10	1	.9	.8	.7	.6	.5	.9	1	1	1	3/4
serif_drop	4	5	3.6	3.2	2.8	2.4	2	3.6	5	5	5	
o	8	4	4	3	3	3	4	4	4	4	4	3
apex_o	8	3	3	3	3	2	3	3	3	3	3	3
other values from	cmr10	cmr9	cmr8	cmr7	cmr6	cmr5	cmsl9	cmti10	cmmi10	cmcsc10		

\*The measurements for *bar* apply also to *slab*, *cap\_bar*, and *cap\_band*. All of the Concrete fonts have *dish* = 0, *fudge* = .95, *superness* = 8/11, *superpull* = 1/15, and *beak\_darkness* = 11/30. Parameters not mentioned here are inherited from the corresponding cm fonts, except that *ccsc10* has *lower.fudge* = .93; *ccti10* has the *u* value 20 not 18.4, and the *crisp* value 11 not 8; *ccmi10* has the *crisp* value 0 not 8.

Is the extra weight of Concrete Roman really necessary for compatibility with AMS Euler? Here is a small sample that uses ordinary cmr10 as the text font, so that readers can judge this question for themselves:

The set  $S$  is, by definition, all points that can be written as  $\sum_{k \geq 1} a_k(i-1)^k$ , for an infinite sequence  $a_1, a_2, a_3, \dots$  of zeros and ones. Figure 1 shows that  $S$  can be decomposed into 256 pieces congruent to  $\frac{1}{16}S$ ; notice that if the diagram is rotated counterclockwise by  $135^\circ$ , we obtain two adjacent sets congruent to  $(1/\sqrt{2})S$ , since  $(i-1)S = S \cup (S+1)$ . [3]

And now let's replay the same text again, using Concrete Roman and  $\backslash baselineskip=13pt$ :

The set  $S$  is, by definition, all points that can be written as  $\sum_{k \geq 1} a_k(i-1)^k$ , for an infinite sequence  $a_1, a_2, a_3, \dots$  of zeros and ones. Figure 1 shows that  $S$  can be decomposed into 256 pieces congruent to  $\frac{1}{16}S$ ; notice that if the diagram is rotated counterclockwise by  $135^\circ$ , we obtain two adjacent sets congruent to  $(1/\sqrt{2})S$ , since  $(i-1)S = S \cup (S+1)$ . [3]

Equation numbers presented us with one of the most perplexing design questions. Should those numbers be typeset in Euler or cast in Concrete? After several experiments we hit on a solution that must be right, because it seems so obvious in retrospect: We decided to set equation numbers in an "oldstyle" variant of Concrete Roman, using the digits '0123456789'. The result—e.g., '(3.14)'—was surprisingly effective.

After I had been using AMS Euler for several months and was totally conditioned to "upright mathematics," I began to work on a chapter of the book where integral signs appear frequently. It suddenly struck me that the traditional integral sign is visually incompatible with AMS Euler, because it slopes like an italic letter. Such a slope was now quite out of character with the rest of the formulas. So I designed a new, upright integral sign to match the spirit of the new fonts. Then I could typeset

$$\int_a^b f(x) dx - \frac{1}{2\pi i} \oint_{z=r} \frac{g(z) dz}{z^n}$$

and  $\int_{-\infty}^{\infty} \cos x^2 dx$ , instead of

$$\int_a^b f(x) dx - \frac{1}{2\pi i} \oint_{z=r} \frac{g(z) dz}{z^n}$$

and  $\int_{-\infty}^{\infty} \cos x^2 dx$ . The new integral signs went into a new font called euex10, which became \fam10 in math mode; I told TeX to get integral signs from the new font by simply saying

```
\mathchardef\inttop="1A52
\mathchardef\ointtop="1A48
```

in my macro file. Later I noticed that the infinity sign ' $\infty$ ' of Computer Modern was too light to be a good match for the Euler alphabets, so I created a darker version ' $\infty$ ' and put it into euex10 with the new integral signs.

Hermann Zapf was helping to advise me during all this time. For example, he approved a draft of Chapter 1 that had been phototypeset in Concrete Roman and AMS Euler, while I was tuning things up. Later, when he received a copy of the first printing of the actual book, he saw Chapter 2 and the other chapters for the first time; and this led him to suggest several improvements that he could not have anticipated from Chapter 1.

Chapter 2 is about summation, and I had used the sign  $\sum$  from Computer Modern's cmex10 font, together with its displaystyle counterpart

$$\sum_{k=0}^n f(k),$$

to typeset hundreds of formulas that involve summation. Hermann pointed out that the capital \Sigma of Euler looks quite different—it is ' $\Sigma$ ', without breaks—so he suggested changing my summation signs to look more like the  $\Sigma$  of Euler. I did this in the second printing of the book, using  $\sum$  in text formulas and

$$\sum_{k=0}^n f(k)$$

in displays. Hermann also asked me to make the product symbols less narrow, more like Euler's ' $\Pi$ '; so I changed them

from  $\prod$  and  $\prod$  to  $\prod$  and  $\prod$ .

Moreover, he wanted the arrows to have longer and darker arrowheads: ' $\rightarrow$ ', not ' $\rightarrow$ '. And he wanted curly braces to be lighter, so that

$\left\{ \begin{matrix} a \\ b \\ c \\ d \end{matrix} \right\}$  would become  $\left\{ \begin{matrix} a \\ b \\ c \\ d \end{matrix} \right\}$ .

All of these new characters were easy to design, using the conventions of Computer Modern [6], so I added them to *euex10* and used them in the second printing.

Readers of *Concrete Mathematics* will immediately notice one novel feature: There are “graffiti” printed in the margins. My co-authors and I asked students who were testing the preliminary book drafts to write informal comments that might be printed with the text, thereby giving the book a friendly-contemporary-lifelike flavor. We weren’t sure how such “remarks from the peanut gallery” should be typeset, but we knew that we did want to include them; in fact, we collected almost 500 marginal notes. Roy hit on the idea of putting them in the *inner* (gutter) margin, where they would not have too much prominence. He also sent a sample of a suitably informal typeface, on which I modeled “Concrete Roman Slanted Condensed” type.

To typeset such graffiti, I introduced a `\g` macro into my *T<sub>E</sub>X* format file, so that it was possible to type simply ‘`\g` Text of a graffito.`\g`’ on whatever line of text I wanted the marginal comment to begin. I did a bit of positioning by hand to ensure that no two comments would overlap; but my `\g` macro did most of the work. For example, it automatically decided whether to put graffiti into the left margin or the right margin, based on an auxiliary ‘`grf`’ file that recorded the choice that would have been appropriate on the previous *T<sub>E</sub>X* run.

My macros for *Concrete Mathematics* cause *T<sub>E</sub>X* to produce not only the usual *dvi* file and *log* file corresponding to the input, but also the *grf* file just mentioned and four other auxiliary files:

- The *ans* file contains the text of any answers to exercises that appeared in the material just typeset; such answers will be `\input` at an appropriate later time. (Page 422 of *The T<sub>E</sub>Xbook* discusses a similar idea. The only difference between *Concrete Mathematics* and *The T<sub>E</sub>Xbook* in this regard was that I used one file per chapter in *Concrete Mathematics*, while *The T<sub>E</sub>Xbook* was typeset from one long file.)
- The *inx* file contains raw material for preparing the index. After everything but the index was ready, I put all the *inx* files together, sorted them, and edited the results by hand. (See pages 423–425 of *The T<sub>E</sub>Xbook*, where I describe similar index macros and explain why I don’t believe in fully automatic index preparation.)

*This 9 pt typeface proved to work very nicely for marginal graffiti, where it is typeset ragged right, 6 picas wide, with 10 pt between baselines.*

- There’s also a *ref* file, which contains the symbolic names of equations, tables, and exercises that may be needed for cross references. (A *ref* file is analogous to an *aux* file in *L<sup>A</sup>T<sub>E</sub>X*.)

- Finally, a *bnx* file records the page numbers of each bibliographic reference, so that I can include such information as a sort of index to the bibliography. That index was done automatically.

I wouldn’t want to deal with so many auxiliary files if I were producing a simpler book or a system for more general consumption. But for the one-shot purposes of *Concrete Mathematics*, this multiple-file approach was most convenient.

We decided to use a nonstandard numbering system for tables, based on the way superhighway exits are numbered in some states: Table 244, for example, refers to the table on page 244. (The idea wasn’t original with us, but I don’t remember who suggested it.) Macros to accommodate this convention, and to update the cross-references when the page numbers change, were not difficult to devise.

All of the macros I wrote for *Concrete Mathematics* appear in an 814-line file called *gkpmac.tex*, which (I’m sorry to admit) includes very little documentation because it was intended only to do a single job. Macro writers may like to look at this file as a source of ideas, so I’ve made it publicly accessible; for example, it is <*tex.doc>gkpmac.tex* at *score.stanford.edu* on the Arpanet. But people who attempt to use these macros should be aware that I have not pretended to make them complete or extremely general. For example, I implemented a subset of *L<sup>A</sup>T<sub>E</sub>X*’s picture environment, and used it to prepare all but one of the illustrations in the book; but I didn’t include everything that *L<sup>A</sup>T<sub>E</sub>X* makes available. Moreover, I didn’t need boldface type in the mathematical formulas of *Concrete Mathematics* (except for the Q on page 143 of the second printing); so I didn’t include macros for accessing any of the bold fonts of AMS Euler in math mode. In this respect, the book was not a perfect test, because almost half of the Euler characters are boldface and therefore still untried. Macros for bold mathematics would be easy to add, following the pattern already established in *gkpmac*; but I must leave such tasks to others, as I return to my long-delayed project of writing the remaining volumes of *The Art of Computer Programming*.

## References

- [1] Ronald L. Graham, Donald E. Knuth, and Oren Patashnik, *Concrete Mathematics*. Addison-Wesley, 1989.
- [2] Donald E. Knuth, "Mathematical typography," *Bulletin of the American Mathematical Society* (new series) 1 (1979), 337-372.
- [3] Donald E. Knuth, *Seminumerical Algorithms*, second edition. Addison-Wesley, 1981.
- [4] Donald E. Knuth, *The T<sub>E</sub>Xbook*, Volume A of *Computers & Typesetting*. Addison-Wesley, 1984 and 1986.
- [5] Donald E. Knuth, *The METAFONTbook*, Volume C of *Computers & Typesetting*. Addison-Wesley, 1986.
- [6] Donald E. Knuth, *Computer Modern Typefaces*, Volume E of *Computers & Typesetting*. Addison-Wesley, 1986.
- [7] Donald E. Knuth and Hermann Zapf, "AMS Euler—A new typeface for mathematics," *Scholarly Publishing*, to appear.
- [8] David R Siegel, *The Euler Project at Stanford*. Computer Science Department, Stanford University, 1985.

## Outline fonts with METAFONT

Doug Henderson

Lately I've been feeling like no one else out there is having any fun with METAFONT. I sure am. Recently, I came across a small gem in the rough in the METAFONTbook and decided to polish it up some. In chapter thirteen there is an interesting exercise (13.23, page 121) which calls for the user to replace a character by its "outline". Since it was an idea that appealed to me (double dangerous bend kind of fun) I set about applying the solution in various places in the METAFONT source code to see the affects. What I finally settled on was hooking it in as a definition extension of the `endchar` macro. I reasoned that for each Computer Modern character METAFONT produces in a font, there exists both a `beginchar` statement, with which you relate things like the height, width and depth, and an `endchar` statement, which tells the program, among other things, to create a grid box for proof characters and

to ship the character off to the Generic Format file (GF file). So, the `endchar` definition seemed like a good spot to tell METAFONT to take whatever character image had been created and convert it to an outline, since it is called once per character, right before it's shipped out.

Here is the macro definition I used to create outline fonts with:

```
message"Loading the font outline macros.";
boolean outlining;
% only outline when told to
outlining:=false;

def outline =
if outlining:
  cull currentpicture keeping (1,infinity);
  picture v; v:=currentpicture;
  cull currentpicture keeping (1,1)
  withweight 3;
  addto currentpicture also v - v
  shifted right -v shifted left - v
  shifted up - v shifted down;
  cull currentpicture keeping (1,4);
  if (pixels_per_inch >= 600) :
    addto currentpicture also currentpicture
    shifted left;
    addto currentpicture also currentpicture
    shifted up;
  fi
  showit;
fi
enddef;

extra_endchar:=extra_endchar & "outline";
```

The first statement declares a boolean variable named `outlining`. The next line initializes `outlining` to be `false`, so we don't create outline fonts by default. The definition of `outline` includes the line `if outlining:` which tests to see whether the outlining feature is desired. If so, the macro proceeds to punch out the pixels on the inside of our character (leaving more than one pixel for the outline if using a high resolution printer or phototypesetter) and show the results, and, if not, ends the `if` statement with the `fi` statement. The last statement is interesting since it shows a nifty way to tack on new features when creating your characters. Instead of redefining the definition of the `endchar` macro with your special effects (in this case a character outline), just add to the definition of `endchar` with the `extra_endchar` statement. Some similar "hooks" exist for the `beginchar` and

`mode_setup` macros; they are `extra_beginchar` and `extra_setup`.

This macro was intended to be used (by me anyway) as an extension of the plain and `cm` (Computer Modern) base files. This way I did not need to input it each time I wanted to make an outline font. Since I place all of my local extensions to either base file into a file named `local.mf`, that is where my outline macro went.

To load it into the plain base file using the initialization version of METAFONT, `INIMF`, use the following line:

```
inimf plain input local dump
```

Similarly, for the Computer Modern base file use:

```
inimf plain input cmbase input local dump
```

Now that it is a part of your base files, you simply need to set the boolean switch `outlining` to true to make an outline font. Here is one way to make a 17 point sans serif font:

```
mf &cm \mode=varityper;
outlining:=true; input cmss17
```

After making my first few outline fonts I was feeling quite pleased with them so I happily sent them off to Professor Knuth thinking I had some interesting results to share. Here is a small sample of how the characters had turned out.

### Some were funny looking! # =

Notice the last two characters in the line above have fairly thin strokes, so the resulting outline characters appear mostly filled in. Professor Knuth suggested that the values for `rule_thickness` be increased to match the other characters, so I experimented a bit and came up with `rule_thickness=1pt` for the font sample I sent, `cmss17` (a 17pt sans serif font). Another parameter which he suggested I change was `notch_cut`. Looking at the `W` reveals why. The `notch_cut` parameter helps "black" fonts look correct at the meeting place between diagonal strokes by removing pixels there. While I was at it, I also found that the `cap_notch_cut` parameter, when set to a very high value, made the outline fonts look less dark. Knuth's suggestion of setting the `notch_cut` value to `17pt#` (relative infinity) helped out considerably. Here is our reworked `W`.

### Some were better looking! # =

Another issue to be addressed is that of font naming conventions. The method supplied above has the side-effect of creating METAFONT runs with the names "cmss17.600" for a GF file, "cmss17.tfm" for a TFM file, and "cmss17.log" for the log file

METAFONT creates. Since what we are doing is modifying Computer Modern fonts, we should also change the name of our outlined fonts. Besides, if you installed them in your font directories, you would overwrite the real `cmss17 CM` fonts!

I believe the best way around this name collision is to create unique fonts by copying the original name, say `cmss17.mf`, and copying it to another test file you can then work with. I put forth the suggestion that outline fonts have the letter "o" prepended such that an outline font created with `cmss17` would thus become `ocmss17`. This way the names and values of fonts which Knuth has perfected will remain untouched and we can happily play with the font "`ocmss17.mf`" without fear of reprimand. Below is a sample of some of the type of information I would change at the beginning of the file `ocmss17.mf`.

```
% This is OCMSS17.MF in text format,
% as of September 10, 1988.
% Computer Modern Sans Serif Outline 17.28 pt
if unknown cmbase: input cmbase fi

outlining:=true;

font_identifier:="OCMSS"; font_size 17.28pt#;

% rule_thickness#:=.6pt#; % old value
rule_thickness#:=1pt#; % new value
% thickness of lines in math symbols
% notch_cut#:=32/36pt#; % old value
notch_cut#:=17pt#; % new value
% maximum breadth above or below notches
% cap_notch_cut#:=46/36pt#; % old value
cap_notch_cut#:=17pt#; % new value
% max breadth above/below uppercase notches

% and of course the remaining fifty-eight
% parameters in a parameter file
```

The title is changed to reflect our new name and date of creation. The line `outlining:=true;` is used here to show an alternative to placing it on the command line, and the `font_identifier` is also changed to have the outline present (OCMSS). The other changes to the file are changing the values for `notch_cut`, `cap_notch_cut`, and `rule_thickness`, as they are.

This small amount of tuning I have found to be adequate for most of the outline fonts I have created. Here are some samples to show how well (or not) particular types of Computer Modern fonts look in outline form.

## Computer Modern Roman Outline

*5pt the lazy brown fox jumped*  
**THE LAZY BROWN FOX JUMPED**

*10pt the lazy brown fox jumped*  
**THE LAZY BROWN FOX JUMPED**

*12pt the lazy brown fox jumped*  
**THE LAZY BROWN FOX JUMPED**

*17pt the lazy brown fox*  
**THE LAZY BROWN FOX**

## Computer Modern Sans Serif Outline

*8pt the lazy brown fox jumped*  
**THE LAZY BROWN FOX JUMPED**

*10pt the lazy brown fox jumped*  
**THE LAZY BROWN FOX JUMPED**

*12pt the lazy brown fox jumped*  
**THE LAZY BROWN FOX JUMPED**

*17pt the lazy brown fox*  
**THE LAZY BROWN FOX**

## Computer Modern Typewriter Outline

*9pt the lazy brown fox jumped*  
**THE LAZY BROWN FOX JUMPED**

*10pt the lazy brown fox jumped*  
**THE LAZY BROWN FOX JUMPED**

*12pt the lazy brown fox jumped*  
**THE LAZY BROWN FOX JUMPED**

We can see from these samples that fonts which have a large degree of variance in the **stem**, **hair**, **curve**, and **vair** parameters, such as Computer Modern Roman fonts, do not make particularly good-looking outline fonts; often the horizontal bars come out looking black (or nearly so). This is also true for most variations of the Roman family, including CM bold (CMBX), slant (CMSL), and italic (CMTI).

Here is a sample of OCMBX12, OCMSL12, and OCMIT12:

*12pt the lazy brown fox jumped*  
**THE LAZY BROWN FOX JUMPED**

*12pt the lazy brown fox jumped*  
**THE LAZY BROWN FOX JUMPED**

*12pt the lazy brown fox jumped*  
**THE LAZY BROWN FOX JUMPED**

The sans serif fonts, on the other hand, with nearly uniform thickness in **hair**, **stem**, and **curve** are better subjects for outlines than the serifed fonts and, I thought, came out rather nicely. Personally I thought that the CMSSDC (Computer Modern Sans Serif Demi-Bold Condensed) font came out looking the best of those that I tested. It looks like this:

*10pt the lazy brown fox jumped over the smelly green dog*  
**THE LAZY BROWN FOX JUMPED OVER AND OVER**

*12pt the lazy brown fox jumped over the smelly*  
**THE LAZY BROWN FOX JUMPED**

*17pt the lazy brown fox jumped*  
**THE LAZY BROWN FOX**

I hope others will do some experimenting with outline fonts and share their results with the rest of us.

A few general observations that may be obvious but need to be said anyway are that the larger the font size, the better the "outline" due to the overall thickness of the digital pen strokes. This means that the five through ten point sizes are not really ideal for outline fonts for a 300dpi laser printer. I will be working on changing some parameters to make better low-resolution fonts and report on the results in a future issue. For now, though, we can see for ourselves how they look with the resolution of the APS phototypesetter (722.909 pixels per inch) since this article was typeset with them.

**Font News**

Dominik Wujastyk

**Concrete Roman and Italic**

The new book *Concrete Mathematics* by Ronald L. Graham, Donald E. Knuth and Oren Patashnik\* is naturally typeset using  $\text{\TeX}$ , and also uses new typefaces. The maths is set in AMS Euler, a typeface designed by Hermann Zapf for the AMS. The text is set in special versions of Knuth's CM family roman and italic, with weights designed to blend with AMS Euler. This has been named Concrete Roman and Italic.

Zapf's design for AMS Euler is intended to suggest the look of mathematics as written on blackboards. This is how maths has chiefly been written by generations of maths teachers and researchers and is the medium in which most mathematics has always been seen by most mathematicians. The face is distinctly calligraphic, as opposed to italic, and in my view achieves the effect it seeks. But it faces the same difficulty as any striking and original new type design: it initially distracts the reader from the underlying text. It would be interesting to hear from anyone who reads *Concrete Mathematics* right through how the typefaces fare after protracted reading.

The Concrete roman face appears to have features in common with the CM typewriter font, although at the time of writing I have not seen the parameter files. It is a face somewhat in the genre of Bigelow's Lucida or Carter's Bitstream Charter, though different from these, of course.

For an example of the Concrete and Euler fonts, see Knuth's article "Typesetting Concrete", page 31 in this issue.

**Lucida**

In December 1988 Chuck Bigelow informed me that:

Atari is soon (January 1989) bundling Lucida text fonts with its PostScript clone upgrade for its laser printer, the SLM 804. The Lucida fonts include the  $\text{\TeX}$  text character set. The Lucida math fonts will also be available for Atari systems, but from the Imagen Corp., later in 1989. Also, QMS-Imagen are bundling Lucida fonts in the same character set with a software PostScript clone "UltraScript PC" for IBM PCs and various printers. The Lucida  $\text{\TeX}$  math fonts will also be available from Imagen for that system.

**Graphics****Computer Graphics and  $\text{\TeX}$  — A Challenge**

David F. Rogers

Aerospace Engineering Department  
United States Naval Academy

Of late there has been considerable interest in the inclusion of graphics output within a  $\text{\TeX}$  document. Programs such as  $\text{PjCTEX}$ ,  $\text{gnu\TeX}$ , Fig, and TransFig seek to provide a mechanism for the inclusion of graphics within a  $\text{\TeX}$  document. Each of these programs attempts to provide a nearly complete environment for the design and generation of line art or halftones for inclusion in  $\text{\TeX}$ . All are worthwhile efforts. However, each suffers from a serious problem — device dependence. For example, the  $\text{PjCTEX}$  macro package is too large to run on a microcomputer or in fact many workstations — it really requires a special large implementation of  $\text{\TeX}$ ; Fig and TransFig are graphics device dependent (Sun workstations);  $\text{gnu\TeX}$  only generates  $\text{IATEX}$  compatible output; etc. Yet one of the strong attractions of  $\text{\TeX}$  is its device independence.  $\text{\TeX}$  itself runs on literally dozens of different machines from Crays, through the latest Silicon Graphics, Ardent, and Stellar supermini-computer engineering/scientific workstations, to a lowly PC XT running MS-DOS.

Systems such as  $\text{PjCTEX}$ , Fig, Transfig and  $\text{gnu\TeX}$  basically require the user to *recreate graphical output* or to generate it *ab initio*. This is rather inefficient. There are literally dozens of graphics programs that produce better graphical output, more efficiently than any of these systems.

An alternate technique for importing graphics into  $\text{\TeX}$  is to use the  $\backslash\text{special}$  command. Unfortunately, this requires giving up device independence. Further, not all dvi drivers support all  $\backslash\text{special}$  commands.

**A Suggested Minimal Set of  $\text{\TeX}$  Graphics Macros**

Graphics can be incorporated into  $\text{\TeX}$  documents most efficiently by importing the output of graphics programs directly into the  $\text{\TeX}$  document in the form of Plain  $\text{\TeX}$  commands. The important question is how to do this easily and efficiently. Fortunately, only passive graphics is contained in a publishable document. Consequently, the functional requirements for graphics is quite limited (see Refs. 1 and 2). Specifically, these are the ability to move the

\* Reading, Mass.: Addison-Wesley, 1989.

writing instrument both invisibly and visibly about the writing surface to create thin lines and small dots (points), to 'plot' finite sized areas called pixels in various colors or monochrome and to generate text at specified size, orientation and location. It is convenient to be able to specify movements and locations in either absolute or relative coordinates. Additionally, it is convenient to be able to initialize the graphics system, to explicitly exit the graphics system and to specify the size and location of the writing surface. Finally, it is convenient to be able to specify the thickness of thin lines.

A small suite of less than a dozen and a half Plain *TEX* macros can provide this functionality. Only two of these functional requirements create difficulties. Specifically, character generation at arbitrary orientations is bothersome. Initially, character generation orientations should be limited to horizontal and a 90 degree rotation counter-clockwise from the horizontal. The latter can be accomplished by using *METAFONT* to generate a rotated font. The rotated font is positioned and displayed by stacking the character boxes vertically. Additionally, the generation of halftone images is a bit difficult (see below).

Currently color is not normally incorporated into *TEX* documents. However, with technological advances, this is not far off. Hence, it must be considered.

### The Macros

The macros and their suggested calling arguments required to implement this functionality are briefly described below. Unless otherwise specified units are any acceptable *TEX* values.

**\beginpicture** (Begin picture)

Sets any required initial parameters. Saves all current parameters.

**\endpicture** (End picture)

Resets all parameters to the saved values.

**\viewport#1#2** (Viewport)

#1 The horizontal size of the area for the graphics.

#2 The vertical size of the area for the graphics. Default size is `\hsize` by `\vsize`

The origin is in the lower left corner, positive upward and to the right.

Viewport is the common computer graphics name for this function.

**\linethickness#1** (Line thickness)

#1 The thickness of a thin line. Default is 1pt.

**\setrs1#1#2** (Set resolution)

#1 The number of pixels in the horizontal direction within the viewport.

#2 The number of pixels in the vertical direction w

Set resolution applies only to the `\setpxl` command given below. It should be called only once within a picture.

**\ma#1#2** (Move absolute)

#1 The distance to be moved invisibly in a horizontal direction (x-direction) in absolute coordinates, i.e., from the origin set by the viewport command.

#2 The distance to be moved invisibly in a vertical direction (y-direction) in absolute coordinates, i.e., from the origin set by the viewport command.

Moves the cursor from the current cursor position to that specified by #1, #2 in absolute coordinates.

**\mr#1#2** (Move relative)

#1 The distance to be moved invisibly in a horizontal direction (x-direction) in relative coordinates, i.e., from the current location of the cursor.

#2 The distance to be moved invisibly in a vertical direction (y-direction) in relative coordinates, i.e., from the current location of the cursor.

Moves the cursor from the current cursor position to that specified by #1, #2 in relative coordinates.

**\da#1#2** (Draw absolute)

#1 The distance to be moved visibly in a horizontal direction (x-direction) in absolute coordinates, i.e., from the origin set by the viewport command.

#2 The distance to be moved visibly in a vertical direction (y-direction) in absolute coordinates, i.e., from the origin set by the viewport command.

Draws a line from the current cursor position to that specified by #1, #2 in absolute coordinates.

**\dr#1#2** (Draw relative)

#1 The distance to be moved visibly in a horizontal direction (x-direction) in relative

coordinates, i.e., from the current location of the cursor.

- #2 The distance to be moved visibly in a vertical direction (y-direction) in relative coordinates, i.e., from the current location of the cursor.

Draws a line from the current cursor position to that specified by #1, #2 in relative coordinates.

**\pa#1#2** (Point absolute)

- #1 The distance to be moved visibly in a horizontal direction (x-direction) in absolute coordinates, i.e., from the origin set by the viewport command.
- #2 The distance to be moved visibly in a vertical direction (y-direction) in absolute coordinates, i.e., from the origin set by the viewport command.

Moves from the current cursor position to that specified by #1, #2 in absolute coordinates and creates a dot at that point.

**\pr#1#2** (Point relative)

- #1 The distance to be moved visibly in a horizontal direction (x-direction) in relative coordinates, i.e., from the current location of the cursor.
- #2 The distance to be moved visibly in a vertical direction (y-direction) in relative coordinates, i.e., from the current location of the cursor.

Moves from the current cursor position to that specified by #1, #2 in relative coordinates and creates a dot at that point.

**\text#1** (Text)

- #1 The text string to be plotted.

The current font is used. Default is cmr10.

**\tangle#1** (Text angle)

- #1 The angle ccw in degrees from the horizontal at which text is to be plotted. Initially only 0 and 90 degrees are allowed.

**\setrgb#1#2#3** (Set rgb)

- #1 The red component of the current drawing color.
- #2 The green component of the current drawing color.
- #3 The blue component of the current drawing color.

The red, green and blue components of the color are given as decimal numbers in the range 0 to 1 with 0 representing no intensity of the component and 1 full intensity.

Black is indicated by r = 0, g = 0, b = 0.

White is indicated by r = 1, g = 1, b = 1.

The default is black.

**\setpxl#1#2#3#4#5** (Set pixel)

- #1 The x-coordinate of the lower left corner of the pixel.
- #2 The y-coordinate of the lower left corner of the pixel.
- #3 The red component of the color of the pixel.
- #4 The green component of the color of the pixel.
- #5 The blue component of the color of the pixel.

A pixel is a finite area of the writing surface extending to the right and upward from the location specified by the coordinate given in #1 and #2.

The red, green and blue components of the color are given as decimal numbers in the range 0 to 1 with 0 representing no intensity of the component and 1 full intensity.

For monochrome (gray scale) images the monochrome value is obtained by averaging the red, green and blue components.

**\setgray#1#2** (Set gray levels)

- #1 The number of gray levels available.
  - #2 Parameter that determines the gray level representation scheme.
- p – patterning (see Ref. 3 and below)  
d – dither (see Ref. 3)

Using patterning the maximum number of available gray levels depends on the resolution of the output device.

With dither the specified maximum number of gray levels sets the size of the dither matrix. It must be a power of 2.

**\setlut#1#2#3** (Set look-up table)

- #1 Number of red bits
- #2 Number of green bits
- #3 Number of blue bits

The number of shades or intensities of red, green, and blue are  $2^{\#1}$ ,  $2^{\#2}$ , and  $2^{\#3}$  respectively. The number of colors is  $2^{\#1+\#2+\#3}$ .

### Implementation Considerations

Since normal *TEX* output is ultimately onto raster scan devices, e.g., laser printers and digital phototypesetters, the line drawing macros must implement Bresenham's line drawing algorithm (or a DDA) (see Ref. 3). *PICTE*X, in fact, does this using the period as the plotting character. Although using the period as the plotting character achieves device independence, if variable line thickness is required, using only the period as the plotting character is not sufficient.

Several alternate techniques suggest themselves. Two are mentioned here. The first is to use METAFONT to define graphic plotting characters, e.g., various sized dots, squares or diamonds from which various thickness lines can be constructed. The second is to directly define various size squares using *\hrule* or *\vrule* (Ref. 4 page 64) to use in constructing the various thickness lines. In either case the 'symbols' should be overlapped to decrease aliasing effects.

Inclusion of photographs or continuous tone images in *TEX* requires dealing with pixels to digitally represent these pictures. Hence the *\setpxl* macro above. Pixels have varying intensity. Printers use a photographic screen process called halftoning to print continuous tone images. There are two principal digital analogs of this process: patterning and dither (see Refs. 3, 5 or 6). Patterning gives up spatial resolution to achieve intensity resolution. Dither introduces randomness into the digitized image to give the impression of multiple intensity (gray) levels without losing spatial resolution. Both, actually print only black 'dots' on 'white' paper. Both are easily extended to color.

In patterning, elements of a small grid are either black or white. For example a  $2 \times 2$  grid using a single dot size yields 5 intensity levels as shown here



The fifth intensity is, of course, no dots. Multiple dot sizes can be used. With multiple dot sizes, the number of intensity levels is  $(\text{width} \times \text{height of the grid})^{(\text{the number of dot sizes} + 1)}$ . From this it is easy to see that a  $2 \times 2$  grid with 3 different dot sizes yields 256 different intensity (gray) levels. However, not all these patterns necessarily yield unique intensity levels. Monochrome images are quite adequately represented by 256 intensity (gray) levels. Note also that 256 is precisely the number in a new *TEX* font. Thus, one method

of generating these intensity levels is to create a special graphics font using METAFONT. One word of caution is in order: the number of  $2 \times 2$  patterns depends on the available spatial resolution. For example, for a  $2 \times 2$  pattern grid, an image digitized with 512 pixels across its width requires a minimum width on the page of a 300 dpi output device of 3.41 inches. This assumes that one physical output device dot is used for each horizontal grid location. Thus, only 5 intensity levels are available. Higher resolution output devices, e.g., phototypesetters, yield more intensity levels. Minimum acceptable output resolutions are 1000–1200 dpi. Consequently, 300 dpi laser printers would be useful for proofing only. One additional subtlety should be mentioned. Unless the patterns are carefully selected, moiré as well as other undesirable patterns appear in the output. These can be minimized by randomly rotating the patterns 90, 180 and 270 degrees. Unfortunately, unless a font rotation macro becomes available, this feature requires four different graphics fonts.

Dither introduces controlled randomness into the digitized image to produce the impression of multiple gray levels. Intensity resolution is increased without loss of spatial resolution. The algorithm is conceptually quite simple:

```

for each scanline in the image
  for each pixel along the scanline
    determine the position in a dither matrix
    i = (x Mod n) + 1
    j = (y Mod n) + 1
    determine the pixel display value
    if image pixel intensity(x,y) <
        dither matrix(i,j) then
          write black pixel
        else
          write white pixel
        end if
    next pixel
next scanline

```

Details are given in Refs. 3 and 6. The number of apparent intensity levels depends on the dither matrix. Dither matrices are square. Their sizes typically increase by factors of 2, e.g.,  $2 \times 2$ ,  $4 \times 4$  and  $8 \times 8$ . The number of apparent intensity levels is the dither matrix size squared, e.g., an  $8 \times 8$  dither matrix yields 64 apparent intensity levels. The optimal  $2 \times 2$  dither matrix is

$$\begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$$

The  $4 \times 4$  dither matrix, which can be derived from the  $2 \times 2$  matrix, is

$$\begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix}$$

Additional intensity levels can be obtained using multiple dot sizes. TeX has quite adequate facilities for implementing a dither algorithm.

### Using the Macros

The set of TeX graphics macros described above can certainly be used directly to create both line art and halftones. However, except for touch-up work, that is not their intended use. Their intended use is as the end product of a filter pipe<sup>†</sup> that can be processed by TeX in a device independent way. Conceptually, the intended pipeline is

The output of your favorite graphics program is saved into a file.

A program converts this format to a neutral file format here called the standard display file format (.sdf).

A program converts the .sdf format to a TeX file containing only the macro calls described above.

TeX processes this file in the normal fashion producing a .dvi file.

An output driver converts the .dvi file to some device dependent form, e.g., Postscript, HP LaserJet+, phototypesetter, etc.

Assuming that the output of your favorite graphics program is already saved in a file, then the UNIX pipe command line is

```
graf2sdf < graphicsfile | sdf2tex > file.tex
```

### Standard Display File Format

The concept of storing a picture in a data file is part of current international standards efforts. The applicable graphics standard is that for the Computer Graphics Metafile (CGM) (see Refs. 7 and 8). However, the CGM is somewhat more detailed and complex than required for the current application. A simple alternative, here called the standard display file, is given in Table 1.<sup>‡</sup>

The first line of the standard display file contains an identification string. Each subsequent line of the file begins with an operation code

**Table 1: Standard Display File (.sdf) Codes**

Code	Operation	Parameters <sup>†</sup>
0	Move Absolute	x,y
1	Point Absolute	x,y
2	Draw Absolute	x,y
3	Move Relative	$\Delta x, \Delta y$
4	Point Relative	$\Delta x, \Delta y$
5	Draw Relative	$\Delta x, \Delta y$
6	Text	Text string
7	Set color	red, green, blue
8	Set look-up table	red bits, green bits, blue bits
9	Set pixel	x, y, red, green, blue
10	New frame	
11	Pause	
12	Set resolution	xresolution, yresolution
99	Print message	Message string

<sup>†</sup>x,y are floating point numbers in NDC; i.e., in the range  $0 \leq x, y \leq 1.0$ . red, green, blue are floating point numbers in the range  $0 \leq \text{red}, \text{green}, \text{blue} \leq 1$  with  $0 \equiv \text{no intensity}$  and  $1 \equiv \text{full intensity}$ . Red bits, green bits, blue bits are powers of 2 (see Ref. 1 for a discussion of look-up tables). This table is taken from Ref. 1.

number, followed by the appropriate information required by that operation. All coordinate values are specified as floating point numbers in the range  $0 \leq x \leq 1.0$ ,  $0 \leq y \leq 1.0$ . All items are separated by commas. A simple standard display file for a square is:

```
0, 0.05, 0.05
2, 0.95, 0.05
2, 0.95, 0.95
2, 0.05, 0.95
2, 0.05, 0.05
```

Using the macros discussed above the corresponding TeX code is:

```
\begin{picture}
\viewport{2in}{2in}
\linethickness{2pt}
\ma{0.05}{0.05}
\da{0.95}{0.05}
\da{0.95}{0.95}
\da{0.05}{0.95}
\da{0.05}{0.05}
\end{picture}
```

The format is simple enough that a picture can be easily modified or, in fact, generated using a text

<sup>†</sup>Pipe is used here in the context of a UNIX or MS-DOS pipe.

<sup>‡</sup>A similar concept has been in use at the author's home institute for more than a decade.

editor. The format also allows easy extension to include local or additional functionality.

### Conclusions

The *TEX* graphics macros presented above provide all the capability currently, and for the near future, required to include publication quality line art within *TEX* source files. They also include all the required capability to seriously begin experimenting with the direct inclusion of halftone images within *TEX* source files. Including publication quality halftone images within *TEX* source files requires both further development of digital ‘halftoning’ techniques and the general availability of higher resolution output devices.

### The Challenge

The Challenge is for the macro gurus to implement the above *TEX* graphics macros and for the *METAFONT* gurus to generate the required fonts. I'll be happy to consult on the graphics aspects of the development, to test the results and to implement the filter program from .sdf format files to *TEX*.

Aerospace Engineering Department  
United States Naval Academy  
Annapolis, MD 21402, USA  
E-mail: dfr@usna.navy.mil

### References

1. Rogers, David F. and Adams, J. Allan, *Mathematical Elements for Computer Graphics*, 2nd Edition McGraw-Hill Book Co., New York, 1990, Appendix A. (Available in August 1989.)
2. Rogers, David F. and Rogers, Stephen D., A Raster Display Graphics Package for Education, *IEEE Computer Graphics & Applications*, Vol. 6, No. 4, April 1986, pp 51-58.
3. Rogers, David F., *Procedural Elements for Computer Graphics*, McGraw-Hill Book Co., New York, 1985.
4. Knuth, Donald E., *The TEXbook*, Addison-Wesley Publishing Co., Reading, MA, 1984.
5. Knuth, Donald E., Digital Halftones by Dot Diffusion, *ACM TOG*, Vol. 6, pp 245-273, 1987.
6. Jarvis, J. F., Judice, C. N., and Ninke, W. H., A survey of techniques for the display of continuous tone pictures on bilevel displays, *Comput. Graph. Image Process.*, Vol. 5, pp 13-40, 1976.
7. Computer Graphics Metafile for the Storage and Transfer of Picture Description Information (CGM), ISO 8632-Parts 1 through 4: 1987; also, ANSI/X3.122-1986.
8. Arnold, D.B., and Bono, P.R., *CGM and CGI*, Springer-Verlag, Heidelberg, 1988.

---

### A Portable Graphics Inclusion

Bart Childs  
Alan Stolleis  
Don Berryman

One of the serious limitations of current *TEX* usage is the lack of a portable inclusion of graphics. We propose a means of making this possible. It will require a little discipline on the creation of drivers; the procedures outlined herein should make it straightforward to add this to existing drivers.

Graphics inclusion has been a part of the drivers from the Computer Science Department at Texas A&M since soon after the initial release of the earliest QMS-QUIC drivers. In these cases it required significant positioning by the user and therefore became dependent upon the particular printer the document was destined for. This violated the intent of the *dvi* file, namely being device independent. Our previous graphics inclusion was much like the “Ph.D. with a screwdriver” concept.

It is my (Bart Childs) opinion that in spite of the beauty, power, and widespread use of PostScript, it is not a suitable answer for *TEX* output. There are two reasons for this opinion:

- A convenient manner for incorporating fonts from *METAFONT* is not yet in the public domain. *PostScript downloading of bitmaps is inconvenient at best!*
- The size of PostScript files is inordinately large, and use of the system in networks both clogs the network and makes the printing of documents happen at a fraction of the rated speed of the printer, especially if you use Computer Modern fonts.

These comments should not be taken to imply that the immense contribution of PostScript is not appreciated. I think that PostScript is in severe need of a binary mode. Another question of relevance for *de facto* standards is, will it be good for three-dimensional graphics? Many other questions need to be answered before we should treat it as a standard.

We need only a few elements to enable portable graphics inclusion. These elements are:

1. A standard template for the allocation of the size of the graphical area in the *TEX* document.
2. A standard means of putting the size of the graphical area in the *dvi* file.
3. A standard means of communicating the “name” of the file containing the graphics.

A typical line of *TEX* to cause the inclusion of a graphical element would be

```
\fignput{name}{horiz}{float}{caption}
```

The four arguments are:

**name** - The name is reasonably straightforward.

Notice that it does not have an extension.

File names could be the biggest problem in portability between dissimilar systems.

**horiz** - This positioning command would have one of three options: *\left*, *\center*, or *\right*. Text could flow around the graphics as well but that should be handled outside this consideration. We have borrowed this idea from Tom Reid's comments about positioning around figures.

**float** - This argument should be *\tied* or *\float*, which tells *TEX* to put the inclusion in a *\vbox* or an *\insert*.

**caption** - The caption should be optional. If present, it should include all required font changes.

There is a graphics standard which has been approved by ANSI and ISO. ANSI X3.124-85 and ANSI X3.124.1-85 is the Graphics Kernel System (GKS). The GKS standard defines how graphics interfaces function and how GKS data files should be stored. The computer graphics interface, CGI, is a program that provides user support for the creation of graphics. The Graphics Kernel System Metafile, GKSM, is simply an audit trail of the commands used by the CGI to create a graphic image. Within the GKS standard, graphic images have associated *cgm* files. The acronym comes from "computer graphics metafile" and is defined by ANSI X3.122-86 or ISO standard 8632. The *cgm* is a device independent file that describes a graphic image. The *cgm* is similar in nature to the *dvi* files of *TEX*. The GKSM is different from the *cgm* in that the *cgm* is a definition and translation of the final graphic image. The *cgm* will not have all the steps used to produce the picture.

Most vendors deliver reasonable support for a GKS environment or it is available from third party vendors for common systems. Data General (DG) provides support to its end user community in a variety of ways, one being with the Comprehensive Electronic Office-Drawing Board (CEO-DB). Programming support is also provided for ANSI F77, PL/1, and C languages. DG also provides extensive GKS support to output devices. These include DG's graphics peripherals, PostScript devices, the HP family of printers, and others.

Our third author created a utility which inputs *name.cgm* and creates a file *name.siz*. Our second

author has rewritten the original utility in CWEB and made extensive changes. This output file contains text like:

```
% This file created by cgmsize Version 0.1
\hsize %(width)
3.00000in %(height)
% Graphics file: name
% Date of creation: 01/06/89
% Graphics designer: Anonymous
\fignputtrue
% vsize is estimated.
```

Lines 1, 4, 5, 6, and 8 are commentary. We intended that the *\fignput* macro would or would not place the word "DRAFT" in the margin based on the contents of the seventh line. If the *cgm* file does not exist, the utility will prompt for lines two through eight. The second line shows the default value if an answer is not given to the width prompt.

In addition to creating a properly sized and placed *\vbox* for the inclusion of the graphics, the *\fignput* macro should furnish a *\special* to communicate the name of the file containing the graphics. We use a line like:

```
\special{copy{name}}
```

Notice that the filename does not include an extension; remember the goal is device independence! The actual graphics file name should have the appropriate extension furnished by the driver depending upon the intended printer. We suggest the following three character (or shorter) extensions:

*ps* PostScript.

*qic* QMS QUIC engines.

*1bp* Vanilla Canon engines or nearly compatibles.

*jep* Hewlett Packard Laser Jet and compatibles.

One additional burden must be placed on the *\fignput* macro. Some printers will require placement of the "cursor" at the top left corner of the box for the graphics, some will require it to be at the bottom right; some will require the width of the graphics; etc. We propose that the *\vbox* have a height as indicated by the *siz* file. To enable device independence, the contents of the *\vbox* should contain:

1. An *\hrule* with the width of the box, height of 1sp (scaled point), and zero depth.
2. An *\hrule* with the height of the box, zero depth, and width of 1sp.
3. A *\vskip* back to the top left corner of the box.
4. The above *\special*.

Drivers should be modified to notice rules that have 1sp vertical and horizontal spans and to remember the nonzero dimensions in separate registers. This gives all drivers the necessary information to position themselves correctly and to copy the appropriate data file. We recommend passing the current position (the `\vskip` in item 3) at the top left because it agrees with the coordinate system of `dvi` files in some sense. We also understand that many would prefer adding two arguments to the `\special`'s `copy`, the height and depth. If this is done, we strongly recommend it be done in points only. This should be a decision by the standards committee.

The following is an outline of how we would create graphics and incorporate them in `TEX` documents:

1. Create a GKS file by use of utilities or applications that use GKS. We use CEO-DB to create a graphics image.
2. Have GKS output a `cgm` file. In our case with CEO-DB we would create a `cgm` file from the CEO environment.
3. Run `cgmsize` to create the appropriate file needed by `TEX` using `\figinput`.
4. Have a utility that creates the appropriate printer file.
5. We also have to run a utility that properly prepares the graphics output file for the desired printer. This utility will strip off characters at the beginning and end of the file, and reset the printer for `TEX`.
6. Run `TEX`.

After outlining this prototype system, we realized that the format would be appropriate for `dvi` inclusion as well. We might want to create a figure or table using `LATEX`, `PICTEX`, `T2D4`, or other macro package that uses lots of memory. A `dvisize` utility could produce an appropriate size file. A `dvimerge` file could then be run when needed to produce a "complete" `dvi` file.

We are forwarding this to Robert McGaffey and his group working on printer standards. We will be pleased to furnish this and the appropriate sources to all interested parties. Please respond with comments to the first author and copy Mr. McGaffey.

## Output Devices

### `TEX` Output Devices

Don Hosek

The device tables on the following pages list all the `TEX` device drivers currently known to TUG. Some of the drivers indicated in the tables are considered proprietary. Most are not on the standard distribution tapes; those drivers which are on the distribution tapes are indicated in the listing of sources below. To obtain information regarding an interface, if it is supposed to be included in a standard distribution, first try the appropriate site coordinator or distributor; otherwise request information directly from the sites listed.

The codes used in the charts are interpreted below, with a person's name given for a site when that information could be obtained and verified. If a contact's name appears in the current TUG membership list, only a phone number or network address is given. If the contact is not a current TUG member, the full address and its source are shown. When information on the drivers is available, it is included below.

Screen previewers for multi-user computers are listed in the section entitled "Screen Previewers". If a source has been listed previously under "Sources", then a reference is made to that section for names of contacts.

Corrections, updates, and new information for the list are welcome; send them to Don Hosek:  
 Bitnet U33297@Uicvm,  
 Internet U33297@Uicvm.Uic.Edu  
 (postal address, page 3).

### Sources

**ACC** Advanced Computer Communications,  
 Diane Cast, 720 Santa Barbara Street, Santa Barbara,  
 CA 93101, 805-963-9431 (DECUS, May '85)

**Adelaide** Adelaide University, Australia

The programs listed under Adelaide have been submitted to the standard distributions for the appropriate computers. The PostScript driver permits inclusion of PostScript files in a `TEX` file. The driver is described in *TUGboat, Vol. 8, No. 1*.

**AMS** American Mathematical Society, Barbara Beeton, 401-272-9500 Internet: `Bnb@Math.AMS.com`

**Arbor** ArborText, Inc., Bruce Baker, 313-996-3566,  
 Arpanet: `Bwb@Arbortext.Com`

ArborText's software is proprietary and ranges in price from \$150 to \$3000. The drivers for PostScript

printers, the HP LaserJet Plus, the QMS Lasergrafix, and Imagen printers are part of their DVILASER series. The drivers all support graphics and include other special features such as use of resident fonts or landscape printing when supported by the individual printers.

Printing on the Autologic APS-5 and  $\mu$ -5 phototypesetters with DVIAAPS includes support of Autologic standard library fonts and logo processing.

**Aurion** Aurion Tecnología SA de CV,  
Armando Jinich, Arquímedes #3, 501, Polanco 11570,  
México, D.F., 905-545-7315

**Bochum** Ruhr Universität Bochum,  
Norbert Schwarz, 49 234 700-4014

**Caltech** California Institute of Technology,  
Chuck Lane, Bitnet: CEL@CITHEX

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

**Carleton** Carleton University, Neil Holtz,  
613-231-7145

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

**Columb.** Columbia University, Frank da Cruz,  
212-280-5126

**COS** COS Information, Gilbert Gingras,  
514-738-2191

**DEC** Digital Equipment Corporation, John Sauter,  
603-881-2301

The LN03 driver is on the VAX/VMS distribution tape.

**DECUS** DECUS Program Library, Library  
Order Processing, 219 Boston Post Road, BPO2,  
Marlboro, MA 01752, 508-480-3418, 508-480-3659,  
508-480-3446,

The previewer and PostScript driver are combined in a single program, DVIOUT. The program uses GF, PK, and PXL files. It allows landscape printing, inclusion of MacDraw bitmaps, inclusion of Tektronix plot files, drawing of line, arc, point, and filled polygons through \special commands, and TeX-XeT support. Written in C and Macro-32. The program comes with a well-featured PostScript symbiont. There is a charge of \$35 for DECUS members, \$40 for non-members to obtain this program. It is distributed on a 600' 6250 bpi magnetic tape.

**ENS** Ecole Normale Supérieure, Chantal Durand,  
Centre de Calcul, Ecole Normale Supérieure,  
45 rue d'Ulm, 75005 Paris, France

**GA Tech** GA Technologies

**GMD1** Gesellschaft für Mathematik und  
Datenverarbeitung, Federal Republic of Germany,  
Ferdinand Hommes, Bitnet: Grztex@Dbngmd21,  
+49 228 8199621

**GMD2** Gesellschaft für Mathematik  
und Datenverarbeitung, Federal Republic

of Germany, Dr. Wolfgang Appelt,  
uucp: unido!gmdzi!zi.gmd.dbp.de!appelt

**HP** Hewlett-Packard, Stuart Beatty, 303-226-3800

**INFN** INFN/CNAF, Bologna, Italy, Maria Luisa  
Luvisetto, 51-498286, Bitnet: Miltex@Icineca2

The CNAF device drivers are on the VAX/VMS distribution tape.

**Interg'ph** Intergraph, Mike Cunningham,  
205-772-2000

**JDJW** JDJ Wordware, John D. Johnson,  
415-965-3245, Arpanet: M.John@Sierra.Stanford.Edu

**LaserPrint** LaserPrint, P.O. Box 35, D-6101  
Fränkisch Crumbach, Federal Republic Germany,  
+49 6164 4044

The driver supports graphics inclusion in device dependent format. PK font files are used. This program is proprietary. Contact LaserPrint for further information.

**LLL** Lawrence Livermore Laboratory

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

**Milan1** Università Degli Studi Milan, Italy,  
Dario Lucarella, 02/23.62.441

**Milan2** Università Degli Studi Milan, Italy,  
Giovanni Canzii, 02/23.52.93

**MIT** Massachusetts Institute of Technology,  
Chris Lindblad, MIT AI Laboratory, 617-253-8828

The drivers for Symbolics Lisp machines use the Symbolics Generic Hardcopy interface as a back end, so it should work on any printer that has a driver written for it. The printers listed in the table indicate drivers the program has been tested on.

The UNIX drivers for PostScript and QMS printers both support landscape printing and graphics inclusion via specials.

**MPAE** Max-Planck-Institut für Aeronomie,  
H. Kopka, (49) 556-41451, Bitnet: Mio40L@D606wd01

**MPS** Micro Publishing Systems, Incorporated,  
#300-1120 Hamilton Street, Vancouver, B.C.,  
Canada, V6B 2S2, 604-687-0354

The TeXprint laser printer drivers allow landscape printing, collating, odd or even page selection, graphics inclusion, and direct output to the printer. A translation utility for HP soft fonts is included with the HP driver and TFM files for PostScript fonts and a utility for creating TFM's from AFM files are included with the PostScript drivers. The drivers use GF, PK, and PXL files.

The drivers cost \$189 each, \$150 for educational and governmental institutions.

**MR** Math Reviews, Dan Litterer, 313-996-5266

**NLS** Northlake Software, David Kellerman,  
503-228-3383

The VAX/VMS Imagen driver supports graphics.

**Océ** Océ Nederland B.V., Marius Broeren,  
Division Office Automation, P. O. Box 101,  
5900 MA Venlo, The Netherlands, +31.77.76466 x135

**OCLC** OCLC, Thom Hickey, 6565 Frantz Road,  
Dublin, OH 43017, 616-764-6075

**OSU1** Ohio State University, John M. Crawford,  
614-292-1741, Bitnet: Ts0135@hstvma,  
Internet: Crawford-j@Ohio-state.Edu

**OSU2** Ohio State University, Ms. Marty Marlatt,  
Department of Computer and Information Science,  
2036 Neil Avenue, Columbus, OH 43210

The drivers are distributed on either ANSI or  
TOPS-20 DUMPER tapes, with hardcopy documentation.  
There is a \$125 service charge (payable to Ohio  
State University) to cover postage, handling, photo-  
copying, etc.

**Philips** Philips Kommunikations Industrie AG,  
TEKADE Fernmeldeanlagen, Attn. Dr. J. Lenzer,  
Thurn-und-Taxis-Str., D-8500 Nürnberg,  
Federal Republic Germany, +49 911 5262019

**PPC** Princeton Plasma Physics Lab, Charles  
Karney, Arpanet: Karney%PPC.MFENET@NMFECC.ARPA

Versatec output from TeXspool is produced via the  
NETPLOT program. TeXspool also produces output  
for the FR80 camera. Color and graphics primitives are  
supported through specials.

**Procyon** Procyon Informatics, Dublin, Ireland,  
John Roden, 353-1-791323

**PTI** Personal TeX, Inc., Lance Carnes,  
415-388-8853

Graphics output is supported on Imagen, Post-  
Script, and QMS printers.

**Rad Eye** Radical Eye Software, Tom Rokicki,  
Box 2081, Stanford, CA 94309, 415-326-5312

**RTI** Research Triangle Institute, Randy Buckland,  
Arpanet: rcb@rti.rti.org

The program is available in the `comp.sources.misc`  
archives on Arpanet and Usenet.

**Saar** Universität des Saarlandes, Saarbrücken,  
Federal Republic of Germany, Prof. Dr. Reinhard  
Wilhelm, uucp: wilhelm@sbsvax.UUCP

**SARA** Stichting Acad Rechenzentrum Amsterdam,  
Han Noot, Stichting Math Centrum,  
Tweede Boerhaavestraat 49, 1091 AL Amsterdam  
(see *TUGboat*, Vol. 5, No. 1)

**Scan** Scan Laser, England, John Escott,  
+1 638 0536

**Sci Ap** Science Applications, San Diego, CA,  
619-458-2616

**SEP** Systemhaus für Elektronisches Publizieren,  
Robert Schöninger, Arndtstrasse 12, 5000 Köln,  
Federal Republic of Germany

DVIP400 uses PXL files. Landscape printing is  
supported in all versions and graphics inclusion in all

but the IBM PC version. Source is available on request.  
Cost varies from 300-1848DM.

**Stanford** Stanford University

The Imagen driver from Stanford is present on  
most distributions as the file `DVIIMP.WEB`. It provides  
limited graphics ability.

**Sun** Sun, Inc.

**Sydney** University of Sydney, Alec Dunn,  
(02) 692 2014, ACSnet: alecd@facet.ee.su.oz

**Talaris** Talaris, Sam Hassabo, 619-587-0787

All of the Talaris drivers support Textronix graph-  
ics. Device-dependent special fonts are used for older  
printers and all previewers; newer printers use PK fonts.

**T A&M1** Texas A&M, Bart Childs, 409-845-5470,  
CSnet: `Childs@TAMU`

Graphics is supported on the Data General drivers  
for the Printronix, Toshiba, and Versatec on the Data  
General MV. On the TI PC, graphics is supported  
on the Printronix and Texas Instruments 855 printers.  
There are also previewers available for both the Data  
General and the TI.

**T A&M2** Texas A&M, Ken Marsh, 409-845-4940,  
Bitnet: `KMarsh@TAMN1L`

**T A&M3** Texas A&M, Norman Naugle,  
409-845-3104

The QMS driver supports inclusion of QUIC graph-  
ics commands via specials as well as landscape printing.

**T A&M4** Texas A&M, Thomas Reid, 409-845-8459,  
Bitnet: `X066TR@TAMVM1`

The TeXrox package includes a GF/PK/PXL to  
Xerox font converter (PXLrox2), and utility to build  
TFM files from licensed Xerox fonts (Xetrix). The  
programs are all written in C. Fonts not present on the  
Xerox printers can be printed as bitmaps on printers  
with the graphics handling option (GHO).

At present the TeXrox package is being distributed  
on a twelve-month trial basis; the trial is free for  
U.S. educational and government institutions, \$100 for  
foreign or commercial institutions. Licensing agreements  
will be available when the trial offer expires.

**TeXsys** TeXsys, Joachim Schrod, Kranichweg 1,  
D-6074 Rödermark, Federal Republic Germany,  
+49 6074 1617

The LaserJet driver supports graphics inclusion in  
device dependent format. PK font files are used. This  
program is proprietary. Contact TeXsys for further  
information.

**Tools** Tools GmbH Bonn, Edgar Fuß,  
Kessenicher Straße 108, D-5300 Bonn 1,  
Federal Republic of Germany

The Tools implementation of TeX and the drivers  
listed are described in *TUGboat*, Vol. 8, No. 1.

**TRC Finl'd** Technical Research Centre of Finland,  
Tor Lillqvist, +358 0 4566132, Bitnet: `tml@fingate`

**UBC** University of British Columbia, Afton Cayford, 604-228-3045

**UCB** University of California, Berkeley, Michael Harrison, Arpanet: `vortex@berkeley.arpa`

**UCIrv1** University of California, Irvine, David Benjamin

**UCIrv2** University of California, Irvine, Tim Morgan, Arpanet: `Morgan@UCI.ARPA`

**U Del** University of Delaware, Daniel Grim, 302-451-1990, Arpanet: `grim@huey.udel.edu`

The distribution includes a program to convert font files generated by METAFONT to Xerox font format.

**UIC** University of Illinois at Chicago, Don Hosek, Bitnet: `U33297@Uicvm`, Internet: `U33297@Uicvm.Uic.Edu`

**U Ill** University of Illinois, Dirk Grunwald, Arpanet: `Grunwald@M.Cs.Uiuc.Edu`

The previewers are available via anonymous FTP in the directory `pub/iptex.tar.Z` on `a.cs.uiuc.edu`.

**U Köln** Univ of Köln, Federal Republic of Germany, Jochen Roderburg, 0221-/478-5372, Bitnet: `A0045@Dk0rrzk0`

**U Mass** University of Massachusetts, Amherst, Gary Wallace, 413-545-4296

**U MD** University of Maryland, Chris Torek, 301-454-7690, Arpanet: `chris@mimsy.umd.edu`

The UNIX Imagen driver is on the UNIX distribution tape. The drivers may be obtained via anonymous FTP from `a.cs.uiuc.edu` in the directory `pub/iptex.tar.Z` or from `mimsy.umd.edu` in the directory `tex`.

**U Mich** University of Michigan, Kari Gluski, 313-763-6069

**UNI.C** Aarhus University, Regional Computer Center, Denmark

**URZ** University of Heidelberg, Federal Republic of Germany, Joachim Lammarsch, Bitnet: `Rz92@Dhdurdz1`

**U Shef** University of Sheffield, England, Ewart North, (0742)-78555, ext. 4307

**Utah** University of Utah, Nelson H. F. Beebe, 801-581-5254, Arpanet: `Beebe@Science.Utah.edu`

All of the Beebe drivers are distributed together. They are available on IBM PC-DOS floppy disks (about 6), or 1600bpi 9-track tape in TOPS-10/20 BACKUP/DUMPER format, VAX/VMS BACKUP format, UNIX tar format, and ANSI D-format. Send tape or disks for a copy. The programs are available for anonymous FTP from `SCIENCE.UTAH.EDU` on the Internet; information is in the file `PS:<ANONYMOUS>OOREADME.TXT`. A VAX/VMS binary distribution is available for anonymous FTP (password guest) from `CTRSCI.UTAH.EDU`. `OOREADME.TXT` in the login directory gives details. On JANET, the programs may be obtained from the directory `aston.kirk::[public.texdvi210]`. The drivers are available from Listserv on EARN to European Bitnet users. Sending the command GET DRIVER FILELIST

(in an interactive message, or as the first line of a mail message) to `LISTSERV@DHDURZ1`. Files are obtained with the command `GET filename filetype`. Graphics is supported only in the DVIALW (PostScript) driver.

**U Wash1** University of Washington, Pierre MacKay, 206-543-6259, Arpanet: `MacKay@June.CS.Washington.edu`

The programs listed under U Wash1 are all on the standard UNIX distribution tape.

**U Wash2** University of Washington, Jim Fox, 206-543-4320, Bitnet: `fox7632@uwacdc`

The QMS driver for the CDC Cyber was written under NOS 2.2 and supports graphics.

**Vander** Vanderbilt University, H. Denson Burnum, 615-322-2357

**Wash St** Washington State University, Dean Guenther, 509-335-0411, Bitnet: `Guenther@Wsuvml`

**Wash U** Washington University, Stanley Sawyer, 314-889-6703

The IBM PC LN03 driver is a modified version of Flavio Rose's `DVI2LN3`. Graphics support is provided through inclusion of LN03 plotfiles and line drawing specials. All three PXL formats on the PC are supported. The program is available free of charge with the receipt of a blank disk and return mailer.

**W'mann** Weizmann Institute, Rehovot, Israel, Malka Cymbalista, 08-482443, Bitnet: `Vumalki@Weizmann`

**Xerox** Xerox, Margaret Nelligan, Xerox Printing Systems Division, 880 Apollo Street, El Segundo, CA 90245, 213-333-6058

**XOrbit** XOrbit, P. O. Box 1345, D-8172 Lenggries, Federal Republic Germany, +49 8042 8081

This driver supports graphics inclusion in device dependent format. PK font files are used. This program is proprietary. Contact XOrbit for further information.

**Yale** Yale University, Jerry Leichter, Arpanet: `Leichter-jerry@Cs.Yale.Edu`, Bitnet: `Leichter@Yalevms`

`DVIDIS` is available for anonymous FTP from `Venus.Ycc.Yale.Edu`. Log in as anonymous and do a CD [`DVIDIS`]. That directory contains the three required files needed to run the previewer. The image must be transferred using BINARY mode.

## Screen Previewers — Multi User Systems

### ■ Data General MV

### T A&M1

### ■ DEC-20

### OSU2 ASCII Output

### Utah BBN Bitgraph terminal

### ■ HP9000/500

### Utah BBN Bitgraph terminal

### ■ IBM MVS

**GMD** GDDM supported devices: IBM 3179, 3192, 3193, and 3279

**Milan1** Tektronix 4014

#### ■ IBM VM/CMS

**UIC** Terminals connected through 7171 Protocol converters: Tektronix compatible, VT-640 compatible, GDDM driven IBM 3179 and 3279 terminals, GDDM driven Tektronix 816

DVIview may be obtained by sending \$30 (to defray duplication costs), a blank tape, and a return mailer to Don Hosek. The program is still in the developmental stages, and enhancements will be made in the future. The program uses PK files.

**Wash St** GDDM driven IBM 3179 and 3279 terminals

Uses PXL files at 120dpi. Allows viewing of the page in eight parts normal size or three parts compressed.

**W'mann** IBM 3279, 3179-G

Previewing is provided by DVI82, the Weizmann driver for the Versatec plotter. The program uses PXL files.

#### ■ UNIX

**Utah** BBN Bitgraph

**U Wash1** DMD5620

Uses GF, PK, or PXL files at 118dpi. **tpic** output is supported. The program consists of two parts: a program running on the host computer and another that is downloaded to the terminal.

#### ■ VAX VMS

**Adelaide** AED 512, ANSI-compatible, DEC ReGIS, DEC VT100, DEC VT220, Visual 500, 550

Uses PK or PXL files.

**DECUS** Tektronix 4014

Uses PK, GF, or PXL files.

**INFN** DEC ReGIS

Uses PXL files.

**Talaris** Talaris 7600, 7800

**Utah** BBN Bitgraph

### Screen Previewers — Microcomputers and Workstations

#### ■ Amiga

**Rad Eye**

Uses PK files. Included with AmigaTEX.

#### ■ Apollo

**Arbor**

Uses GF, PK, and PXL files. Preview is available for \$500.

**U Ill** X-11 Windows System

#### ■ Atari ST

**TeXsys**

**Tools**

#### ■ Cadmus 9200

**U Köln**

#### ■ IBM PC

**Arbor, PTI** EGA, MCGA, UGA, Hercules, Olivetti, Tecmar, Genius full page, ETAP Neftis, Toshiba 3100, AT&T 6300

Uses GF, PK, and PXL files as well as tuned PostScript fonts (the base set available with PostScript printers). Preview of integrated bit map graphics, font substitution, magnification on the fly, two-up display of pages, and searching for character strings are supported. Preview is available for \$175.

**Aurion, PTI** EGA, CGA, VGA, Hercules Graphics Card, Wyse WY/700, Genius VHR Full Page Display, AT&T 6300

Uses fonts from the laser printer driver in PK or PXL format to display text. Magnification may be set on entry. Maxview is available for \$125.

#### PTI

Uses fonts in GF, PK, or PXL format. On the fly magnification, on the fly inclusion of DVI files, font substitution, and 256 character fonts are supported. PTIVIEW is available for \$149.

**T A&M3** EGA, CGA, Hercules

The **cdvi** program is available for \$175.

#### ■ IBM PC/RT

**U Ill** X-11 Windows

#### ■ Integrated Solutions

**UCIrV1**

**Utah** BBN Bitgraph

#### ■ SUN

**Arbor**

Uses GF, PK, and PXL files. Preview is available for \$500.

**UCB**

**UCIrV2**

**U Ill** X-11 Windows, Sunview Window System

Uses GF, PK, and PXL files.

#### ■ Vaxstation/Unix

**U Ill** X-11 Windows

Uses GF, PK, and PXL files.

#### ■ Vaxstation/VMS

**Arbor** GPX(UIS)

Uses GF, PK, and PXL files.

Preview is available for \$500.

**INFN** GPX(UIS)

Uses PXL files.

**Philips** GPX(UIS)

**RTI** GPX(UIS)

Uses PK files at 78, 94 and 112dpi. Written in ADA. Source is included.

**Yale** GPX(UIS)

Uses PK files at 300dpi.

### Low-Resolution Printers on Multi-User Systems — Laser Xerographic, Electro-Erosion Printers

	Amdahl (MTS)	CDC Cyber	Data General MV	DEC-10	DEC-20	HP9000 500	IBM MVS	IBM VM/CMS	IBM VM/UTS	Prime	Siemens BS2000	Sym- bolics Lisp	UNIX	VAX VMS
Agfa P400							SEP	SEP			Saar		Saar SEP	SEP
Canon					Utah	Utah							Canon Utah	Utah
DEC LN03					Utah	Utah							Utah	DEC NLS Procyon Utah
Golden Laser 100					Utah	Utah							Utah	Utah
HP LaserJet Plus					Utah	T A&M2 Utah				OSU1			Arbor Utah	Arbor LaserPrint Utah
IBM 38xx, 4250, Sherpa							GMD1 URZ	GMD1 Wash St						
Imagen	Arbor UBC		T A&M1	Stanford Vander	Columb. Utah	Utah	Arbor	Arbor W'mann				MIT	Arbor U Md Utah	Arbor NLS Utah
Kyocera													MPAE	LaserPrint MPAE
Océ 6750														Océ
PostScript printers					Utah	Arbor Utah		Arbor		OSU1		MIT	Arbor Carleton MIT Utah	Arbor DECUS Sydney Utah
QMS Lasergrafix	Arbor	U Wash2	T A&M1			T A&M2	Arbor GMD1	Arbor GMD1		OSU1 T A&M3	GMD1	MIT	Arbor MIT U Wash1	Arbor GA Tech T A&M3
Talaris														Talaris
Xerox Dover					CMU								Stanford	
Xerox 2700II		Bochum			OSU2 Xerox			ENS					Xerox	
Xerox 9700	Arbor U Mich						Arbor T A&M4	Arbor T A&M4	T A&M4			U Del	ACC Arbor T A&M4	

### Low-Resolution Printers on Multi-User Systems — Impact and Electrostatic Printers

	CDC Cyber	Cray	Data General MV	DEC-10	DEC-20	HP9000 500	IBM MVS	IBM VM	Prime	UNIX	VAX VMS
Apple ImageWriter					Utah	Utah				Utah	LSU Utah
DEC LA75, LP100					OSU2 Utah	Utah				Utah	Utah
Epson FX/RX					Utah	Utah				Utah	Utah
Facit 4542											INFN
Florida Data					MR						
MPI Sprinter					Utah	Utah				Utah	Utah
Okidata					Utah	Utah				Utah	Utah
Printronix			T A&M1		Utah	Utah				Utah	Utah
Toshiba			T A&M1		Utah	Utah				Utah	Procyon Utah
Varian											Sci Ap
Versatec	U Köln	PPC	T A&M1	GA Tech Vander	U Wash1		GMD1 U Milan2	W'mann	LLL	U Wash1	Caltech NLS

**Low-Resolution Printers on Microcomputers and Workstations — Laser Xerographic, Electro-Erosion Printers**

	Amiga	Apollo	Atari ST	HP1000	HP3000	HP9000 200	IBM PC	Integrated Solutions	SUN
Agfa P400							SEP		
Canon			Utah				Utah	Utah	Utah
Cordata LP300							PTI		
DEC LN03			Utah				Utah Wash U	Utah	Utah
Golden Laser 100			Utah				Utah	Utah	Utah
HP 2680				JDJW	PTI				
HP 2688A				JDJW		HP			
HP LaserJet Plus	Rad Eye	Arbor	TeXsys Tools	TRC Finl'd		MPAE	Arbor LaserPrint MPS, PTI XOrbit Utah	Utah	Utah
Imagen		Arbor OCLC	Utah				Arbor PTI Utah	Utah	Arbor Sun U Md Utah
Kyocera			LaserPrint			LaserPrint			
PostScript printers	Rad Eye	Arbor				Arbor	Arbor MPS PTI Utah	Utah	Arbor MIT Utah
QMS Kiss, Smartwriter	Rad Eye								
QMS Lasergrafix		Arbor Scan					Arbor PTI		Arbor MIT U Del
Xerox 9700		COS Scan							T A&M4

## Low-Resolution Printers on Microcomputers and Workstations — Impact and Electrostatic Printers

**Typesetters**

	Apollo	CDC Cyber	HP3000	IBM MVS	IBM PC	IBM VM/CMS	Siemens BS2000	Sperry 1100	Sun	UNIX	VAX VMS
Allied Linotype CRTronic											Procyon
Allied Linotype L100, L300P					PTI						
Allied Linotype L202					PTI						Procyon
Autologic APS-5, Micro-5	COS Scan				Arbor PTI				Arbor	Arbor	Arbor Interg'ph
Compugraphic 8400			U Shef		Arbor PTI						NLS
Compugraphic 8600		UNI.C			Arbor PTI	Wash St		U Wisc			NLS
Compugraphic 8800					Arbor						
Harris 7500										SARA	
Hell Digiset				GMD2			GMD2				

---

## Report from the DVI Driver Standards Committee

Don Hosek

The newly-constituted **T<sub>E</sub>X** Users Group DVI driver standards committee has been working on the development of standards for device drivers since the fall of 1988. This article is a first report on our status to the membership of **T<sub>E</sub>X** Users Group.

At the time of this writing, we are in the midst of discussion of \special standards for device drivers. By the **T<sub>E</sub>X** Users Group meeting this August, we should have a preliminary report on this topic available for distribution to all interested parties. We welcome all input from members of the **T<sub>E</sub>X** community; if you have any suggestions, comments, etc. regarding the issue of \special handling, we would appreciate it if you could send these to Robert McGaffey (internet: McGaffey%Orn.Mfnet@Nmfcc.Arpa) for distribution to the members of the committee.

The members of the committee are: Robert McGaffey, chair, Oak Ridge National Laboratory; David P. Babcock, Hewlett-Packard; Elizabeth Barnhart, TV Guide; Stephan v. Bechtolsheim, Integrated Computer Software Inc.; Nelson Beebe, University of Utah; Jackie Damrau, University of New Mexico; Donald Goldhammer, University of Chicago; Don Hosek, University of Illinois at Chicago; David Ness, TV Guide; Thomas J. Reid, Texas A&M University; David Rodgers, Arbortext, Inc.; Brian Skidmore, Addison-Wesley Publishing Co.; Glenn Vanderburg, Texas A&M University; and Ralph Youngen, American Mathematical Society.

**Editor's note:** Earlier TUGboat articles dealing with the subject of DVI driver standards include the following:

Robert McGaffey, The ideal **T<sub>E</sub>X** driver, 8#2, 161-163.

Thomas J. Reid, DVI driver considerations for high-volume printing systems, 8#3, 287-291.

Glenn L. Vanderburg and Thomas J. Reid, \special issues, 8#3, 291-300.

Shane Dunne, Why **T<sub>E</sub>X** should NOT output PostScript—yet, 9#1, 37-39; addendum, 9#2, 178.

---

## High Quality Printing of **T<sub>E</sub>X** - DVI Output Files in the VAX/VMS Environment

Marius Broeren & Jan van Knippenberg  
Océ-Nederland B.V.

Océ-van der Grinten N.V. is the parent of an international group of companies, the Océ Group, which distributes, produces and develops a large range of copiers and copying supplies as well as office automation products, including word processors and laser printers, for both commercial and design engineering offices.

Océ-Nederland B.V. has developed the Océ 6750 laser printer. This printer is based on the well-known engine of the Océ 1900 copier family. The laser printer has a resolution of 508 dpi (20 dots/mm). The printspeed is 23 pages per minute. The heavy duty engine prints a target load of up to 200,000 pages per month. Paper input and paper output are as advanced as usual for the Océ copiers. The level of quality printing of the Océ 6750 laser printer is perfectly suited for printing the output of the high quality typesetting program **T<sub>E</sub>X**. For this reason Océ has developed software for connecting the Océ 6750 laser printer to a wide range of VAX/VMS computers. On these VAX/VMS computers **T<sub>E</sub>X** runs as an application and the **T<sub>E</sub>X**-DVI files are converted to the ECMA/ODA protocol of the Océ 6750 laser printer. The combination of the high quality typesetting program **T<sub>E</sub>X**, the VAX/VMS computer and the Océ 6750 laser printer is responsible for a high level of quality printing.

### The Océ 6750 laser printer

In this part we will describe the printing process. The information of the VAX/VMS computer is received via an IEEE 488 interface. This information has the ECMA/ODA format and is processed in the Raster Image Electronics. Here the information is converted into pixels, a processable form for the Laser Scan Module. The electrophotographic process consists of six steps:

1. charging
2. exposure
3. developing
4. transferring
5. fusing
6. cleaning

**1. Charging.** The 370-inch continuous photoconductor is a polyester belt coated with a thin layer of zinc-oxide. Zinc-oxide has the following characteristics: in the dark it is not electrically conductive. It is an insulator. But when you expose

zinc-oxide to the light it becomes a conductor. The photoconductor is charged from the corona unit.

**2. Exposure.** The charged parts of the photoconductor move in the direction of the exposure area where the Laser Scan Module will expose the photoconductor. The laser discharges the non-black part of the image (write white engine). The black part of the image is still charged and is called a latent image.

**3. Developing.** This image has to be made visible. The process is called developing. The latent image is passed along a rotating metal tube to which toner has been applied. We use a dry monocomponent toner. The toner particles on the tube are attracted by the electrical charge of the latent image. Now we have a visible black image on the photoconductor.

**4./5. Transferring and fusing.** The next task is to bring the image to the paper. This is achieved in two transfer steps. In the first transfer step a soft belt is pressed up against the photoconductor and picks up the image. The rubber belt is heated and at a temperature of 105 degrees Celsius, the melted toner is fused on the warmed paper (second transfer step). The paper has been fed from one of the two paper trays.

**6. Cleaning.** The final step of the process is to discharge the photoconductor and then brush off the toner residue. The photoconductor is now ready for recharging.

The size of the toner particles, the spot size of the laser and the described process are of great importance for the final print quality. The benefits of this process are a high image resolution, uniform density, consistent quality from first to last print, no developer to be mixed, no direct contact between photoconductor and paper, no paper jams.

As mentioned in the introduction the paper handling is as advanced as usual for Océ copiers. This includes the following:

- *Input.* The printer has two input trays (1600 + 600 A4 sheets) available. The large capacity and the fact that the paper compartment is outside of the printer give you the possibility of non-stop printing.
- *Output.* The sorter with 20 selectable bins of 100 sheets each takes care of the collation and storage of a large capacity of 2,000 sheets. You can use the ergonomic designed sorter in a personal, set or sortwise printing mode.

## ODA

As stated the Océ 6750 laser printer uses the ECMA / ODA protocol.

What is ODA?

ODA, Office Document Architecture, is an international standard which offers a solution for integrating office systems. ODA is an interchange standard for multi-media documents which has been produced in order to allow documents (text and/or graphics) to be interchanged between computer systems anywhere in the world. ODA documents can contain information represented in the form of character text, raster graphics and geometric graphics. ODA enables communicating systems to interchange documents across networks with the integrity of the content, format and layout. It is possible to reproduce, reprocess, store or print the document in the form intended by the originator. To assist in the integration of computer systems, ODA employs the following established standards:

- the character content of ODA is compatible with ISO 6937, and thus with Telex and Teletex.
- the geometric graphics content is compatible with ISO 8632, and thus with the GKS/CGM family of graphics standards.
- the raster graphics content is compatible with CCITT Recommendations T4 and T6 and thus with group 3 and group 4 facsimile.

At Cebit 1988 at Hannover ODA was used to demonstrate mixed media document interchange. Complex pages of text, image and graphics were originated by one company, displayed and edited by a second company and printed by the Océ 6750 laser printer with communications via X.400 mail and X.25 lines.

## Interfacing to VAX/VMS computers

With the standard IEEE 488 interface of the Océ 6750 laser printer it is very easy to connect the printer to a wide range of VAX/VMS computers. In the VAX/VMS computer range we find three types of busses:

1. Q-bus
2. Unibus
3. BI-bus

Connecting the Océ 6750 to a VAX with respectively one of these busses can be done with standard from Digital Equipment Corporation available IEEE boards and drivers. Besides the hardware and the driver you need host resident software for the communication with the printer and the conversion

of **TEX**-DVI files to ECMA/ODA. The software developed by Océ contains:

1. the printer protocol
2. the symbiont
3. the conversion programs
4. the font management tools

**1. The printer protocol** The Low Level and the High Level Printer Protocol (LLPP and HLPP) take care of the cooperation between symbiont and the IEEE 488 device driver. The printer protocol is used for informing the host about the printer status and setting the printer in a certain status. The font downloading procedures are also implemented in the printer protocol. Other typical tasks of the printer protocol are:

- sending/receiving packets to/from the Océ 6750
- initialize the printer at start time
- converting messages to packets (two way)
- font management

**2. Symbiont** The standard symbiont of the VAX was not applicable to control the conversion programs and the communication between the user and the printer (two input trays, 20 output bins, setwise, sortwise or personal printing, handle messages coming from the printer etc.). Océ developed a symbiont specially for the 6750. The symbiont controls all the conversion programs and interacts with the job controller. Typical aspects of the symbiont are:

- errorhandling
- starting and controlling the HLPP and conversion programs
- communicates with the job controller
- interprets the DCL extensions and options
- including burst/flag/trailer pages

**3. Conversion programs** The conversion programs re-sort under the "umbrella" of the symbiont and take care of several conversions to ECMA/ODA, the input standard of the 6750 laser printer. There are converters for:

LN03 Plus to ECMA/ODA  
Lineprinter to ECMA/ODA  
**TEX**-DVI to ECMA/ODA

**4. Font management tools** With an easy to use font-tool it is possible to use Metafont to generate PXL-fonts. These are transformed to the Océ format and can be downloaded to the printer.

The combination of the high quality typesetting program **TEX**, the VAX/VMS computers and the

Océ 6750 laser printer is responsible for producing high quality documents.

If you wish to receive an original set of printouts or additional information, please contact:

Océ-Nederland B.V.  
Marius Broeren  
Office Automation  
P.O. Box 101  
5900 MA VENLO  
The Netherlands  
tel. (0)77 - 592222

## Query

Editor's note: When answering a query, please send a copy of your answer to the TUGboat editor as well as to the author of the query. Answers will be published in the next issue following their receipt.

### Output driver for Xerox 4045 on IBM 3090

At Intevep, the Venezuela government research facility for the petroleum industry, we work with several kinds of computers. We have an IBM 3090-200, two VAX 11/780, one Data General MV4000, and 71 Sun Workstations.

We are running **TEX** on almost all these machines, but especially on the IBM, for almost three years now.

The output from the IBM 3090 is done on an IBM 3820, an IBM 4250, and several QMS 800 laser printers.

Last year we bought 50 Xerox 4045 laser printers for our needs of distributed printing. All our people are asking for a **TEX** driver for the 4045, for proofing their work. We obtained a driver for a Xerox 4045 connected to the IBM by means of a serial interface, but all our printers have a coax interface to the IBM.

We hope to find someone who has or wants to develop this driver for our environment. The use of **TEX** in our installation will increase if we can find this driver.

Arturo Puente  
Intevep  
P. O. Box 88521  
Caracas 1080, Venezuela

## Site Reports

### **UKTEX and the Aston Archive**

Peter Abbott  
Aston University, UK

Towards the end of 1988 it became obvious that the demands on UKTEX and the Aston archive were still increasing, threatening a major flood. As a result of interest from several parties at Nottingham (the first UKTUG meeting), the idea of a group effort to maintain the archive germinated.

The number of contributions continues to grow and unfortunately there is an element of duplication and 'duff' material in the archive. Some contributions have passed through many gateways and gateway table translations, or even encoding and unencoding routines. As yet there is no consistent format, and items are stored in a variety of forms including SHAR, ARC etc.

Magnetic tapes arrive regularly at Aston, and numerous requests have been received for material on floppy discs. Currently I support three major tape distributions:

- a copy of the Washington tape,
- a VMS backup of the archive at 6250bpi (2 reels),
- a VMS backup of our working set of TEX/LATEX with PSSPRINT at 3.0 (either 1600bpi or 6250bpi).

I have also recently been asked if I can supply material on cartridge tapes (for SUNs). In addition, Adrian Clark of the University of Essex has for some time been supplying magnetic tapes for VMS sites.

At the Nottingham meeting we identified one major 'hole' in these facilities. Commercial users (in the main) are unable to take UKTEX or access the archive unless a local JANET site is willing to host them. I am currently looking at ways to provide a Bulletin Board Service which will list (amongst other information) details of providers and 'wants'. At the same time, mailboxes will probably be made available.

The idea of group maintenance of the archive resulted in a meeting being held in London in December 1988 at which the following — agreed — were persuaded — succumbed:

Adrian Clark	University of Essex
Malcolm Clark	Imperial College
Charles Curran	Oxford University
David Osborne	University of Nottingham

Sebastian Rahtz	Southampton University
Philip Taylor	Royal Holloway and Bedford New College (RHBC)

plus myself. Each brings different talents to the group, and it is hoped that the first phase will have been completed as you are reading this (*it is impossible writing on 4th January 1989 to assume that no changes will have taken place in our plans*).

Each member of the group has been allocated one or more of the subdirectories of PUBLIC. A tree structure will be created using pointers so that there is only one 'real' copy of a file (for example *latex.tex*) but when looking for the files required for any particular implementation then all files needed will be included.

Directory and filenames will all be standardised on 8 characters (max) with 3 suffix characters. This will satisfy most systems including MS DOS. There will be a standard file in each directory called *00readme.txt* which explains the purpose and contents of that directory, and, where appropriate, *00files.txt* and/or *00map.txt* files will also be provided. The *00files.txt* file provides details on each individual file, while *00map.txt* can be used to generate a filename mapping 'script' on systems which do not enforce VMS's file-naming rules (e.g., mixed case filenames under UNIX).

We shall provide 'kit' files which detail all files needed for a particular installation e.g. *vms.kit* or *cms.kit* or *msdos.kit*. It may be necessary to provide subkits especially for *unix.kit*. These kits will also be used to create the magnetic media distributed from Aston on demand.

We agreed at the meeting (please note, however, that it is still subject to change) to start with the following top-level structure for the archive:

*tex, metafont, latex (subdir for slitex),  
amstex, digests (texhax uktex texmag),  
bibtex, utils, fonts etc, drivers, docs,  
langs, tools (for use by the 'group' not  
the general public)*

We shall announce these revised arrangements in UKTEX, but will ensure that mail and NIFTP access to the present structure is unaffected whilst work progresses, and we may even provide a short overlap period.

Longer term plans include the distribution of kits on relevant media and the BBS described above.

Aston University has a policy of 'customer care' and I am working on providing the help file for the mail server in as many languages as necessary. Most Europeans speak excellent English, but I consider a user-friendly interface as a major asset. I

already have translators for Danish, Italian, Spanish, Swedish and other languages.

My thanks are due to the many providers of material, in particular Michael DeCorte from Clarkson, Jon Radel (DECUS tapes) and many others too numerous to name. I must however mention the other members of the group who give so freely of their time and expertise. Without them the archive could fossilise and maybe even die.

### Data General site report

Bart Childs

The distribution continues to be stable. We have improved the robustness of the previewer for alphanumeric terminals and are in the process of adding Stephan von Bechtolsheim's PostScript driver.

I hope the note on "Portable Graphics Inclusion" that appears elsewhere is seriously considered by all. I feel that it does offer a reasonable plan for portable inclusion of graphics and merging of dvi files.

### UnixTEX Site Report

Pierre A. MacKay

TEX and METAFONT came into the New Year in updated versions. TEX is now supplied as version 2.95 and METAFONT as version 1.7. There is also a new version (2.9) of tangle. None of these changes need worry you very much, since the basic functionality of the programs is more or less unaltered. The change to tangle allows a reference to numeric macros before they are defined, which is a bit more permissive than the old version, and the other changes have to do with the behavior of the programs when they terminate abnormally. The changes to TEX and METAFONT had already been made in versions 2.94 and 1.6 respectively, but certain features of those versions interacted badly with some operating systems. One useful Unix feature is gone from the new version (unless we

sneak it back in through a change file); you can no longer exit with a ^D key-in.

Chris Torek looked over my version of a Bourne shell script for running TEX, LATEX, and SLiTEX, and sent a vastly improved version which runs as follows:

```
case "$0" in
  */tex|tex) me=tex; fmt=plain;;
  */latex|latex) me=latex; fmt=lplain;;
  */slitex|slitex) me=slitex; fmt=splain;;
 *) echo "don't know how to be $0" \
    1>&2; exit 1;;
esac
# verify 1 or 2 arguments
case $# in
  1|2) ;;
 *) echo "usage: $me foo[.tex [my[.fmt]]]" \
    1>&2; exit 1;;
esac
virtex "&"${2-$fmt} ${1+"$1"}
```

There are no major changes in web2c which will probably be at about version 2.26 by the time this report is printed. As each new variety of Unix operating system works up to a successful compilation, small improvements emerge which make the system yet more general. The list of successful machines is by now too large to maintain accurately. In December, I had the opportunity of trying out a compilation on a NeXT system. As might be expected from what is essentially a BSD4.3 kernel, compilation was a complete success, using a recent version of the GNU gcc compiler. The loader had trouble with the -s option but that is a known bug and will soon be fixed. We intend to provide copies of the distribution on 256Kbyte laser disks in the very near future. Any help with previewers and METAFONT display routines will be very much appreciated.

There is even more interesting news about fonts. Several years ago, through the courtesy of the American Mathematical Society, it became possible to offer compiled versions of the Euler fonts (fraktur, script, etc.) which were developed by Hermann Zapf, with the support of the AMS. These compilations were of mixed quality, because some defects still remained in the METAFONT programs, and some styles would not compile at all in the smaller design-sizes. Donald Knuth has reviewed the entire set, and remapped them into a slightly

different arrangement. There is now a full range in 10pt, 7pt, and 5pt sizes available as a regular part of the distribution, made up at true size, and magsteps 0, 0.5, and 1. These compiled fonts are made available by the AMS for non-profit scholarly use. A site which wishes to use them in other ways, or which needs, for some reason, to produce different compilations, should get in touch with the AMS and arrange to acquire the `eu*.mf` files under license.

A completely new font family is the "concrete fonts" which are discussed elsewhere in this issue. The `cc*.mf` files for these are part of the distribution, but are not intended to serve merely as another canonical font family. They are offered as a lesson in how to use METAFONT to its full effectiveness, by creating one-off fonts that may be especially appropriate only for a specific publication. It is this capacity that distinguishes a font-design tool such as METAFONT from a system for font expression, such as PostScript. We welcome other similar experiments, which we will make part of a special directory known simply as `metafonts`.

In addition to the "concrete" fonts, the `metafonts` directory will include the *Pandora* family, designed by Neenie Billawalla. We have eagerly awaited the release of this new set of fonts, which was developed independently from, and on somewhat different principles from Computer Modern. This is an original creation, and shows what a professional designer can do with METAFONT. As with any genuinely new font design it has taken several years of work to bring it to its present shape. I would urge those who make use of this font to include in their publications some acknowledgement of Neenie Billawalla's generosity in making it available for free non-profit and educational use.

An anomaly in the `chardx` values of some characters in Computer Modern was discovered during 1988, and the relevant Computer Modern METAFONT files have been corrected. A complete recompilation of all the `plain.tex`, `lfonts.tex`, `sfonts.tex`, and some others was done in December, 1988, and reflects all the improvements described in `cm85.bug`. These fonts are still offered in 118dpi, 200–240dpi, and two 300dpi resolutions. I should like to hear from readers whether the 200–240dpi versions are still really useful. I have not seen an advertisement for a 240dpi laser printer in some time, and I do not have any sense of how many sites still use the 200dpi Versatec for output. The 300dpi fonts come in `CanonCX` mode for generic write-black

print engines, and in `RicohFourZeroEightZero` mode for generic write-black print-engines. Notice the change in the naming convention for the Ricoh engine. We are going to need quite a few new `mode_defs` as new varieties of print-engine become available, and the suggestion that was made a while back (I forget where) that the names be reasonably consistent, with all numeric digits spelled out, seems a very good one. `LN03` fonts, for example (if that really is a print-engine), might be addressed with a `mode_def` named `LNZeroThree`.

There are no major additions to the `babel` foreign language directory, but some important new sets of hyphenation patterns are in the works for Dutch and Russian. Work is also well advanced on the creation of a Cyrillic font in new METAFONT coding, which will be made available both in the old AMS mapping, and in a new mapping more compatible with the Russian hyphenation system.

The various systems of support software collected under `TeXcontrib` continue to grow. There is a new version of `TiB`, a bibliographic preprocessor which is ultimately based on the Unix `refer` bibliographic system. Wolfgang Appelt's `knit` and `twist` (see *TUGboat* 7:1, pp 20–21) is included, under his collective name for it, "patchwork". Joachim Schrod's wide-ranging adaptation of `WEB` appears as `literate_macros`. There is always room for more.

All the DVI interpreter software has now been moved to a directory named `DVIware`. The old, familiar `ctex` (Chris Torek's collection of DVI interpreter programs), is now to be found as `umd-dvi`. This renaming was undertaken to avoid confusion with the `ctex` directory used by `web2c`. Several interpreters besides the ones under `umd-dvi` now use the system of distributed directories for font access that first appeared with those programs. The file `SUBDIRmakefile` will copy all the fonts supplied with the distribution into such an arrangement.

Recent correspondence with R. M. Damerell leads me to hope also that by the time this report is printed there will be a working version of `crudetype` which will be particularly useful as a previewer for proof correction on old-fashioned alphanumeric terminals.

## VAX/VMS Site Report

David Kellerman  
Northlake Software

We have been shipping a new distribution of **T<sub>E</sub>X** for VAX/VMS since September. It contains all the changes and bug fixes that had accumulated in the Stanford distribution at that time, and corrections and improvements to our VAX/VMS-specific modifications. The L<sup>A</sup>T<sub>E</sub>X macros gained many bug fixes since our previous release, and we cleaned up loose ends in the arrangement of S<sub>L</sub>T<sub>E</sub>X. The font sets are considerably enlarged, and there is also a new conversion program called XXtoXX. It converts between any combination of GF, PK, and PXL formats, can process all RMS record formats as input or output, and makes quick work of converting large numbers of font files.

Much work went into making the new distribution easier to install and use. Martin Havlicek did most of the work of dividing it into pieces, then organizing each as a VMSINSTAL kit. Beginning users can install two or three kits for a basic system, and reliably end up with working software; later, if need arises, they can install additional kits. We organized and rewrote the VMS-specific documentation, too, and the result is both an improvement for the naive user and a source of more useful reference material (it looks better, too).

Of course we ran out of space on the tape again. And it is probably just as well, because the space limitation keeps us focused on providing a reliable core **T<sub>E</sub>X** system. For the broad range of publicly available **T<sub>E</sub>X**-related packages, we still find it better to forward inquiries to their actual developers. They are better at providing up-to-date versions, and they do a better job of answering questions about their own software.

Which leaves me to look forward to updating to **T<sub>E</sub>X** 2.96, and wondering what version comes after 2.99.

## Typesetting on Personal Computers

### The Land of the Free and the Near Free

Alan Hoenig

I've received a surprising number of requests—from as far away as Cameroon—for information about low cost implementations of **T<sub>E</sub>X**. It's now possible to put together several such systems. For this article, a "T<sub>E</sub>X system" includes in addition to T<sub>E</sub>X, a text editor (to create the input into T<sub>E</sub>X), a previewer (to preview on the screen the output of T<sub>E</sub>X before you send it to your printer), and a driver (the program which you need to translate from the language T<sub>E</sub>X uses to the language your printer understands), and (for the first time!) METAFONT. Because this column has talked too much about the IBM-compatible family of computers, we will begin with a *non*-IBM system. But IBMers should read on—among other things, we describe below an impressive integrated T<sub>E</sub>X environment for PC compatibles at a bargain price.

Before we begin, please note that you cannot make indiscriminate copies of the software *unless it is very clearly marked as being in the public domain*. Low cost is not synonymous with public domain! What follows is a summary of low-cost software components; please assume they are *not* public domain unless specifically so noted.

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

The Amiga microcomputers, models 500 and 2000, are powerful home computers, with built-in high-resolution graphics, a large memory capacity, and the ability to multitask. If all things were equal, it would probably be the computer of choice for most microcomputer users. Unfortunately, things are not at all equal—vastly more software is available for IBM and compatibles and for Macintoshes.

Nevertheless, a fine implementation of T<sub>E</sub>X for the Amiga is available from Radical Eye Software (Box 2081, Stanford, CA 94309; (415) 32-AMIGA). The T<sub>E</sub>X part of this system consists of T<sub>E</sub>X and a previewer and costs \$200. If you provide a blank Amiga floppy and a SASE, you can get the mg editor free, which is their local version of a micrognuemacs-type editor. Printer drivers are \$100 apiece, and support the HP LaserJet series, PostScript, QMS KISS and SmartWriter, HP DeskJet, Epson LQ series, NEC Pinwriter series, Epson MX and FX

series, and ImageWriter II printers. (Some of the dot matrix drivers are capable of 360 dpi resolution!) A version of METAFONT with interactive screen support is only \$75. The software includes a very well-written manual; you'll have no trouble getting started with it. A tantalizing description of AmigaTeX appeared last year in *TUGboat* (9, 1, April 1988, 40-41), and a user report is in this issue starting on page 65.

The best news about AmigaTeX is the current street prices of Amiga computers. Rumors abound that Commodore (the Amiga's parent company) will be releasing new models, the 2500 and 3000, and are perhaps lowering the prices of the 500 and 2000 to clear the shelves. Whatever the reason, prices here in the New York City area are low. The 2000 sells for \$1495 complete, including 1 megabyte of memory, 1 floppy drive and the color monitor. The Amiga 500—with 512K memory, 1 floppy drive, and a color monitor—can be had for \$800 or less. To run AmigaTeX, you only need 512K of RAM and two floppy disk drives or one floppy drive and 1.5M of RAM. (Of course, the software really hums with a hard drive, but it is not necessary. AmigaMETAFONT definitely requires 1 megabyte of RAM, as will LATEX.) The extra memory or floppy adds only a few hundred dollars to these prices. These computers are more expensive than the plainest of 8088 IBM compatibles, but you get substantially more computer power, including multitasking and stunning graphics.

### **PCompatible-T<sub>E</sub>X**

The components of a PCompatible T<sub>E</sub>X system have to be assembled in the same way as are most of these compatibles—component by component, each component of which comes from someplace else.

My choice for text editor remains PC-Write, now up to version 3.0. PC-Write is a shareware program, so you should look for one from a friend or bulletin board. Otherwise, contact QuickSoft for their current distribution policy (QuickSoft, 219 First North, #224, Seattle, WA 98109; (206) 282-0452). One irritation with PC-Write is its inclination to break lines right after an explicit hyphen. T<sub>E</sub>X turns this line break to an extra space, so if you're not careful, you get extra spaces after these hyphens in the typeset document. A public domain version of the Emacs editor JOVE (*Joe's Own Version of Emacs*) is also widely available.

TurboTeX is a low-cost T<sub>E</sub>X available from the Kinch Computer Company (501 South Meadow

Street, Ithaca, NY 14850; (607) 273-0222). TurboTeX is described extensively in *TUGboat*, 9, 1, 48-52, April, 1988. TurboTeX weighs in at \$100, and Kinch Computer also offers the C source listing for the T<sub>E</sub>X program for committed compile-it-yourselfers. (Contact them for details.) The big news though is that Kinch Computer now has a low-cost version of METAFONT available as well (see page 23)! The combination of T<sub>E</sub>X plus METAFONT executables will be \$150; the C source for the pair will be \$300. Monitor support for their METAFONT includes Hercules, CGA, and EGA.

A superb low-cost previewer called CDVI (because, I guess, you use it to *see* a DVI file) has crossed our desk recently. It's easy to install, and at \$35 it'd be a bargain even if it *didn't* work well. (That's \$35 for 360K disks; add \$10 for 3.5in disks. These prices include postage and handling, but please add sales tax for Texas orders.) When ordering, be sure to specify your monitor type—either EGA, Hercules mono, CGA, ATT 6300, MCGA/VGA, or Toshiba 3100. The program is offered by SullivanSFT, Box 292431, Lewisville, TX 75029.

Most previewers I've tried seem to take their time loading font information, so you have to wait a bit to do any previewing. CDVI is blazingly fast. On my AT compatible, it is essentially instantaneous in operation. I do not perceive any pause between the time I execute its command line and the appearance of the screen image. Viewing subsequent pages in your document is as rapid. Wayne Sullivan, a Dallas-born, Georgia- and Oxford-educated mathematician now at University College in Dublin and the author of CDVI, has designed a fine user interface as well.

CDVI will not be all things to all people, however. The secret to its speed lies in the fact that it has built-in information about all the fonts in PLAIN.TEX, but only at their unmagnified sizes. You will only see unmagnified images on your screen. Magnification is simulated by showing these images while using the letterspacing appropriate for the magnified font. Thus, you can check layout and page breaks, which is all most of us use previewers for anyway. If you use non-CM Roman fonts, you must use an included utility which will let CDVI substitute the plain T<sub>E</sub>X font of your choice (although it will use your .TFM file to position the letters properly). (SullivanSFT cautions that the use of too many nonstandard fonts will slow down CDVI's operation.) If you work with non-Roman fonts, or are doing extensive METAFONTing, CDVI may not be for you.

Nelson Beebe has previously reported on the public domain printer drivers he has prepared. (See *TUGboat*, 8, 1, April 1987, 41-42.) The demand for this software has greatly exceeded their author's expectations; Nelson estimates that they are now in use at over 1000 sites in 28 countries. Because of this extraordinary demand, Nelson has had to modify his distribution policy. IBM PC floppy distributions are available from him for \$100, which includes documentation, media, and shipping. (Nelson H.F. Beebe, PhD, Center for Scientific Computing, Department of Physics, University of Utah, Salt Lake City, UT 84112.) However IBM PC floppies of this software are also available from the following sources: Personal *TeX*, Inc. (12 Madrona Ave., Mill Valley, CA 94941); Kinch Computer Co. (501 S Meadow St., Ithaca, NY 14850); and Jon Radel (Rt 2 110 Sydnor Dr., Leesburg, VA 22075). Prices from these suppliers may be cheaper than Nelson's; the current Personal *TeX* catalog, for example, lists this material for \$35. One caveat: these sources may not be distributing the latest versions of these drivers. (I do not have pricing from the other suppliers.)

Nelson's software is actually a series of C modules intended for those wishing to prepare a variety of drivers for a variety of devices on a variety of printers. The drivers support the HP LaserJet and PostScript (Apple LaserWriter) laser printers, as well as a generous selection of dot matrix machines (Epsons, Okidatas, Toshibas, and Apple ImageWriters among others).

The original *TUGboat* announcement was for version 2.07 of this software; Nelson is now up to version 2.10. Major changes of 2.10 include substantial improvements to the HP LaserJet Plus and Apple LaserWriter drivers. (Printer memory management is much better than before.) Version 3.0 of the software is due out in January. This new version will add support for 2 or 3 new operating systems, over a dozen new drivers, and will have support for font paths with multiple directories, run-time specifications of the format for font file names, and better control over the magnification search when substitutions are required. There will also be support for a startup file for common options, and much more powerful control over paper sizes.

Nelson asks us to conclude with a plea. In the past, several people had promised that they would merge in support for PostScript resident fonts from the *dvi2ps* program of Stephan von Bechtolsheim; this program runs only on UNIX. So far, no one has been forthcoming! Nelson himself won't have

time for this task for several more months at least. Volunteers wishing to get cracking on this should get in touch with him first (to make sure no one else has already begun).

A savvy user—that is, one who reads this column regularly—can put together an AT-class compatible for about \$1600 or so. Such a machine won't be fancy, but it will run *TeX* (actually, it will walk *TeX*. Hmm—this makes the Amiga look better and better.)

### Late-Breaking News: The Deal of the Century?

Always wanted to try *TeX* but afraid to take the plunge? *1-800-USA-BOOKS*, a mail order outfit and that's their toll-free number as well, makes an incredible offer. They have assembled true public domain versions of all the software you might need to run *TeX* on a PC, and will offer their *TeX-Kit* for \$38.50 (no misprint!). Here's what you get:

- A version of PC-Write with a utility to patch this program so it *won't* break at hyphens.
- A true, public domain version of *TeX*.
- A stripped-down, public-domain version of the CDVI screen previewer.
- Nelson Beebe's public domain laser printer drivers. The fonts that come with this *TeX* are only those 300dpi fonts that work with the previewer.
- A shell program to take command of these separate components.

It's this last item that makes the *TeX-Kit* such an extraordinary bargain, because for the first time, users of PC-compatible *TeX* can work within an integrated *TeX* environment, a workspace long available to users of Macintosh or Amiga versions. A flick of your finger on a function key brings up your document for editing. Close the file, and *TeX* it with a tap on another key. Oops—a *TeX* error on line 134? Invoke the "e" option from within *TeX*, and the shell automatically opens your file for editing on line 134. Previewing and printing are similarly controlled with single function keystrokes.

This same company reports plans for similar *TeX* kits at different levels. Watch this space for details!

P.S. *Writer's Tools II* will appear Real Soon Now. Gathering the remainder of the information has been more time-consuming than we initially imagined!

---

## AmigaTeX... or How Envy Was Resisted and Knowledge Found on the Road to Öög

Kim Kubik

**T**HEY SAY THAT ENVY is one of the seven deadly sins. In early 1986 I finally decided I had envied others long enough, it was time to spend my own money on a personal computer to run TeX. I made a list of the pros and cons of the 8086 and 68000 based computers for which TeX was available and would vacillate from week to week on which one I thought best. Then the June TUGboat arrived and a short notice announced that TeX now ran on the Commodore Amiga, further stating that due to the multitasking operating system one could build "an impressive TeX environment." I knew little about the Amiga but worked near Stanford so called the author, Tom Rokicki, and spent part of my lunch hour getting a quick overview of AmigaTeX. When I returned to work I got out that computer list, looked it over a last time, and tossed it in the garbage. And now, a year and a half after I bought an Amiga, I'm still finding out just what a gross understatement that phrase "an impressive TeX environment" really was.

◊      ◊      ◊

**D**EVELOPMENT of what would eventually become the Amiga began in the early 1980's, a time when computer games were still the rage and many "real" computer companies in Silicon Valley were lamenting that they couldn't get good engineers because the game companies were offering such lucrative salaries. A group formed out of this already select pool of talent to design a chip set for the ultimate game machine. Essentially they would tightly couple a Motorola 68000 cpu to custom RISC-style coprocessors to offload many of the cpu intensive chores necessary for sound and color graphics and animation. These components were designed to be modular with a multitasking executive at the core to coordinate their interactions.

Thus, when the bottom dropped out of the game market and the startup was refinanced by Commodore under the stipulation that the system be configured as more "business-like" the basis for a full-function workstation had already been implemented. A real-time message-passing operating system known as Tripos was ported to this hardware by a group of computer scientists in England. Tripos had been developed as a systems software research project at Cambridge (UK) with vision towards a

day when single users would have networked desktop workstations capable of running multiple programs simultaneously with sophisticated interprocess communication. Both a mouse-driven icon-based shell and a windowed Unix-style command line interpreter were added to Tripos and AmigaDOS was born. Suddenly the little game machine vaulted from the realm of single-tasking PCs and WYSIWYG Wonders toward league with the Suns and Apollos.

**A**MIGATEX makes extensive use of the multiwindow multitasking capabilities of the computer. Any text editor, the system command interpreter, the TeX compiler, and the AmigaTeX screen previewer may all be active in separate windows during a session. As soon as e.g. '`tex myFoo.tex`' is typed at the prompt in the TeX window the user can slide back to the editor window and work on something else; as soon as page one of `myFoo.dvi` is output the user can preview it on the hi-res screen while the rest of `myFoo.tex` continues to be processed.

Or while the compiler continues to run the user can preview or print any other `.dvi` file, or score and listen to computer generated music, build objects for a color 3-D ray trace animation, or simultaneously download and unarc files from a bbs. The user is not forced into a linear one-dimensional mode of interacting with the computer. If there is sufficient memory to load a program the Amiga can generally run it no matter what else is active.

While the Amiga operating system does not support virtual memory like the big workstations it also does not burden the buyer with the expensive necessity of hardware memory management and a hard disk just for operating system overhead. As all executable programs are RAM bound a completely functional system can be run from floppies. This leads to one great advantage of the Amiga over any other true multitasking workstation: price. I purchased a single-floppy A1000 with 2.5 Mbytes of RAM in late 1986 for \$1600. I ran AmigaTeX on this with no hitches for a year before I even bought a second drive. The A1000 has been discontinued (but *not* obsoleted!) but the new models are low-end and high-end repackages of the same system.

For the TeX user on a severe budget a one Mbyte two-floppy A500 can be purchased discount for about \$1200; for someone needing maximum performance at minimal cost an add-in 68020 cpu increases the price only another \$1000. The Amiga model B2000 adds greater expandability and IBM-PC compatibility at a greater cost, but it is still priced far below any workstation with similar capabilities.

---

## AmigaTeX... or How Envy Was Resisted and Knowledge Found on the Road to Ööç

Kim Kubik

**T**HEY SAY THAT ENVY is one of the seven deadly sins. In early 1986 I finally decided I had envied others long enough, it was time to spend my own money on a personal computer to run TeX. I made a list of the pros and cons of the 8086 and 68000 based computers for which TeX was available and would vacillate from week to week on which one I thought best. Then the June TUGboat arrived and a short notice announced that TeX now ran on the Commodore Amiga, further stating that due to the multitasking operating system one could build "an impressive TeX environment." I knew little about the Amiga but worked near Stanford so called the author, Tom Rokicki, and spent part of my lunch hour getting a quick overview of AmigaTeX. When I returned to work I got out that computer list, looked it over a last time, and tossed it in the garbage. And now, a year and a half after I bought an Amiga, I'm still finding out just what a gross understatement that phrase "an impressive TeX environment" really was.

◊        ◊        ◊

**D**EVELOPMENT of what would eventually become the Amiga began in the early 1980's, a time when computer games were still the rage and many "real" computer companies in Silicon Valley were lamenting that they couldn't get good engineers because the game companies were offering such lucrative salaries. A group formed out of this already select pool of talent to design a chip set for the ultimate game machine. Essentially they would tightly couple a Motorola 68000 cpu to custom RISC-style coprocessors to offload many of the cpu intensive chores necessary for sound and color graphics and animation. These components were designed to be modular with a multitasking executive at the core to coordinate their interactions.

Thus, when the bottom dropped out of the game market and the startup was refinanced by Commodore under the stipulation that the system be configured as more "business-like" the basis for a full-function workstation had already been implemented. A real-time message-passing operating system known as Tripos was ported to this hardware by a group of computer scientists in England. Tripos had been developed as a systems software research project at Cambridge (UK) with vision towards a

day when single users would have networked desktop workstations capable of running multiple programs simultaneously with sophisticated interprocess communication. Both a mouse-driven icon-based shell and a windowed Unix-style command line interpreter were added to Tripos and AmigaDOS was born. Suddenly the little game machine vaulted from the realm of single-tasking PCs and WYSIWYG Wonders toward league with the Suns and Apollos.

**A**MIGATeX makes extensive use of the multiwindow multitasking capabilities of the computer. Any text editor, the system command interpreter, the TeX compiler, and the AmigaTeX screen previewer may all be active in separate windows during a session. As soon as e.g. '`tex myFoo.tex`' is typed at the prompt in the TeX window the user can slide back to the editor window and work on something else; as soon as page one of `myFoo.dvi` is output the user can preview it on the hi-res screen while the rest of `myFoo.tex` continues to be processed.

Or while the compiler continues to run the user can preview or print any other `.dvi` file, or score and listen to computer generated music, build objects for a color 3-D ray trace animation, or simultaneously download and unarc files from a bbs. The user is not forced into a linear one-dimensional mode of interacting with the computer. If there is sufficient memory to load a program the Amiga can generally run it no matter what else is active.

While the Amiga operating system does not support virtual memory like the big workstations it also does not burden the buyer with the expensive necessity of hardware memory management and a hard disk just for operating system overhead. As all executable programs are RAM bound a completely functional system can be run from floppies. This leads to one great advantage of the Amiga over any other true multitasking workstation: price. I purchased a single-floppy A1000 with 2.5 Mbytes of RAM in late 1986 for \$1600. I ran AmigaTeX on this with no hitches for a year before I even bought a second drive. The A1000 has been discontinued (but *not* obsoleted!) but the new models are low-end and high-end repackages of the same system.

For the TeX user on a severe budget a one Mbyte two-floppy A500 can be purchased discount for about \$1200; for someone needing maximum performance at minimal cost an add-in 68020 cpu increases the price only another \$1000. The Amiga model B2000 adds greater expandability and IBM-PC compatibility at a greater cost, but it is still priced far below any workstation with similar capabilities.

One also gets something else for the modest investment: access to the end products from a community of some of the most avid computer enthusiast/programmers in the world. There are literally hundreds of public domain software tools for the Amiga. The open operating system allows these utilities to work together so that a user can tailor a truly flexible personal application environment.

This flexibility can yield a productivity increase a few orders of magnitude greater than any benchmark of straight cpu performance could possibly indicate. Perhaps a better "benchmark" of both the developer community's sophistication and the Amiga's capabilities is the fact that already next to my Amiga are five software product manuals typeset with AmigaTeX.

These capabilities are not without price: users have to learn AmigaDOS on their own as documentation of the Amiga as a workstation is all but nonexistent. Most dealers are completely ignorant of its potential, and it can take many (enjoyable) hours of testing new utilities to optimize an Amiga for specific work habits. Without a hardware deinterlacer (available only for the model B2000) there is an intolerable screen flicker in the hi-res mode that requires a very stable balanced room light and careful choice of colors to reduce.

Like TeX, AmigaDOS can initially seem frustrating because it offers so many choices: modular extensible software exacts a different kind of toll in the freedom it affords. When the Rexx language became available for the Amiga it was as if the last piece of a puzzle fell into place. A common vocabulary can now be used both to write macros within programs as well as to mediate interaction between programs.

As every capability of the system is available at all times the overlapping window paradigm becomes a true asset rather than mere dressing. The user who understands why TeX is not Desktop Publishing, and understands that ease of use is not enviable when the tradeoff is quality, might check out the Amiga. Like TeX, it can reward effort.

◊                   ◊                   ◊

**E**NVY is a transgression in that it focuses attention to surface appearance, diverting one from the true substance, *the kernel*, of a thing. The heretic might suggest that all one needs for an optimum TeX environment is a fast 32-bit cpu and a large screen. But when the system beneath such glitz is basically an 8-bit style program loader the suggestion is ludicrous. Absolution, at least in this

one case, came easily: before I got an Amiga, I *really* envied people who had those big Sun workstations to run TeX.

I don't any more.

## Using TeX and L<sup>A</sup>T<sub>E</sub>X with WordPerfect 5.0

Michael F. Modest  
Pennsylvania State University

### 0.1 Introduction

While TeX is an extremely powerful typesetting language that can make your publications look as close to professional as you could possibly hope for, this beauty comes at a price: TeX by itself is somewhat tedious to use, and it is not WYSIWYG (what you see is what you get). There appears to be no truly WYSIWYG word processor available for the IBM PC or compatibles. The good news is that — when combined with WordPerfect, which allows customization of printer drivers — the tediousness of TeX can be overcome. I have written a TeX-L<sup>A</sup>T<sub>E</sub>X driver for WordPerfect which allows you to use WordPerfect normally while typing ordinary text, and uses simple macros when typing equations. I have written the preprocessor in such a way that it may be used for L<sup>A</sup>T<sub>E</sub>X documents or for straight TeX documents as long as the keys/macros specifically geared towards L<sup>A</sup>T<sub>E</sub>X are not employed. The user prepares his or her WordPerfect file and sees centering, indented paragraphs, Greek letters, mathematical symbols, subscripts, etc., actually displayed (and entered with single keystrokes). Then, rather than sending this file directly to a printer, "prints" it to a diskfile using the preprocessor which converts the WordPerfect format into a standard TeX or L<sup>A</sup>T<sub>E</sub>X file. If only the macros activated by the WordPerfect keyboards are used, the preprocessor even assures matching of braces, dollar signs and \begin{ and \end of L<sup>A</sup>T<sub>E</sub>X environments.

### 0.2 WordPerfect Fonts

I have programmed the driver so that — using the ordinary WordPerfect font change commands — a number of different types of fonts are available, as well as a predefined math displaymode, as indicated below. Depending on the graphics card and monitor present in your PC, different fonts and/or attributes may or may not be displayed:

- EGA card and monitor: choosing the 1 font/512 character option the full Greek alphabet

plus most other important symbols will be displayed. Bold, underline, italics, etc. are color-coded.

- Hercules graphics card Plus, or Incolor card: choosing the 6 font/512 character option together with a *Ramfont* set prepared by myself the full Greek alphabet, all other important symbols, bold, underline, italic, courier (i.e., verbatim), sub- and superscripts, small caps, calligraphic letters, "Large", etc. will be displayed.
- Neither: sorry, you are out of luck: only the standard 256 ASCII characters, bold and underline will be displayed.

### 0.3 Compatibility Considerations

You can use most of the features of the WordPerfect wordprocessor that you may usually use, such as super<sup>a</sup> or sub<sub>b</sub>-scripts, the speller, the thesaurus, the search and replace routines, block mark/copy/move/retrieve routines, the "merge" and "macro" features, as well as **bolding** and underlining. A number of special features of WordPerfect, such as *tabs*, *indent*, *center*, *flush right*, *footnote*, and *line-spacing* are converted into blank spaces by WordPerfect before sending it to the printer and, thus, cause a problem. Most of these features have been made compatible as indicated in the section on *WordPerfect Keyboards*, a few (e.g., linespacing) must be treated by using *T<sub>E</sub>X* macros directly for these functions.

If a little care is taken a text file can be printed like a normal WordPerfect file or as a *T<sub>E</sub>X* file, as long as the file does not contain any fancy typesetting, such as equations using *T<sub>E</sub>X*-commands or any "backslash"-commands. Even then the file can be printed out under WordPerfect, but it will now not enforce these commands. Since a number of "special *T<sub>E</sub>X* characters" are frequently used in normal text I have replaced three of them in my version of WordPerfect: \$, # and % are no longer reserved, i.e., they print out as seen here. No replacement has been programmed for # and %, since the user of this WordPerfect/L<sup>A</sup>T<sub>E</sub>X customization has no longer any need for them, while the reserved character \$ has been replaced by the  $\Delta$  symbol<sup>1</sup>. An exception is the *Verbatim* font: in this font there are **no** special characters, i.e., \ will print out as \,

<sup>1</sup>The reason why the  $\Delta$  symbol is used at all is due to a WordPerfect weakness: WordPerfect is unable — even when using *Ramfonts* — to display two "attributes" at once, such as italics and subscripts. Therefore, I have chosen to display math-italic like ordinary text, with  $\Delta$  as delimiters.

etc. (usually used in conjunction with \ttt, i.e., in typewriter mode).

### 0.4 WordPerfect Keyboards

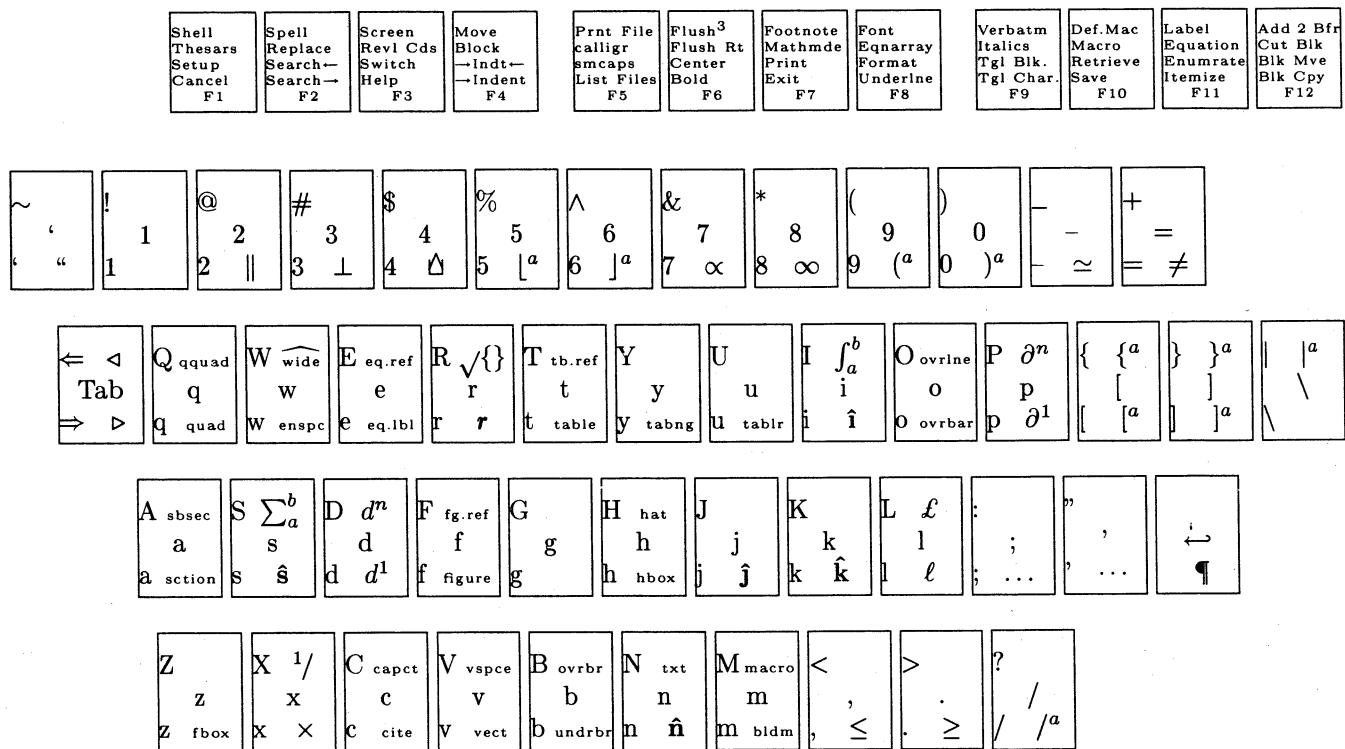
WordPerfect 5.0 allows you to define custom keyboards and to access zillions of special symbols which cannot normally be displayed on the screen such as  $\alpha\beta\gamma$ , or  $\rightarrow$  or  $\partial$  or  $\infty$  etc. This is done by depressing the corresponding key in the appropriate keyboard, sometimes accompanied by simultaneously pressing either the *Alt*-key (abbreviated here as @) or the *Ctrl*-key (here ^). Thus most keys may be used in four ways, e.g., a, shft-a (A), @-a and ^-a, although some keys are limited to three by not allowing either the *Alt* or *Ctrl* key combination. I have defined six new keyboards to be used with *T<sub>E</sub>X*: NORMAL, FOREIGN, GREEK, SINGLE, SMCAPS and CALLIGRAPHIC.

#### NORMAL keyboard

As the name implies this keyboard is the one used most of the time. Its basic layout is shown in Fig. 1, where the central character in each box is the name of the key, the bottom left character shows the character displayed if the key itself is pressed, top left shows the *shift* character, bottom right the *Alt* character and the top right shows the *Ctrl* character (for the function keys the WordPerfect bottom-to-top sequence is used instead). The function of a few of the individual keys is explained below in some detail (note that in most of the examples below unnecessary blanks have been included to distinguish a key like @-a from the three-key sequence @ - a).

Many function keys in this keyboard have been made compatible with *T<sub>E</sub>X* (such as *indent*, *underline*, etc.) while others have been redefined. For example:

- ^-F5 This key, newly labeled *Print File* translates a WordPerfect file (prepared with the present set-up) into a variety of documentstyles.
- F9 This key, labeled *Toggle Character*, switches into keyboard SINGLE for a single character, then switches back into NORMAL.
- F11 This key, labeled *Itemize*, produces a L<sup>A</sup>T<sub>E</sub>X itemized list: the user is prompted for each item ending with a <ret>; items are by default labeled with a •, other labels can be used by enclosing them in square brackets (e.g., the present label was achieved by typing [F11] after the prompt); itemization is exited by hitting <ret> (to end each item), followed by ^-<ret> (to exit list); lists may be nested by hitting <ret> twice (once to end item, second time to start nesting), followed by another



<sup>a</sup> delimiters of self-optimizing size

Figure 1: The NORMAL keyboard

- F11 (itemize) or shift-F11 (enumerate); nesting can be done to arbitrary levels, but only the first two levels of indentation are shown on the screen; after the nested list is exited (with  $\text{--ret}$ ) another  $\text{ret}$  must be hit to get back into the original list. (L<sup>A</sup>T<sub>E</sub>X only.)
- @-F11** This key, labeled *Equation*, switches into the Math Italic font to *display* an equation centered on a separate line (without equation number), ended by hitting the  $\text{ret}$  key; for visualization, the expression is surrounded by the  $\Delta\Delta$  symbol (on separate lines above and below the equation).

Since WordPerfect allows for the definition of other keys on the regular keyboard (by pressing *Alt* or *Ctrl* at the same time), many additional keys have been defined as indicated in Fig. 1. For example:

- @-a** Produces an invisible  $\backslash\text{section}\{\dots\}$  or  $\backslash\text{section}*\{\dots\}$  command, first asking whether the section should be numbered (displayed by a preceding  $\#$ -symbol) or unnumbered (preceding  $*$ ), then prompting for a title, ended by  $\text{ret}$ ; the title is displayed in "Very large Type". (L<sup>A</sup>T<sub>E</sub>X only.)

**@-p** Produces a visible  $\backslash p\{\dots\}\{\dots\}$  (partial derivative) command, prompting for a numerator and a denominator, each ended by  $\text{ret}$ ; for example,  $\text{@-p T ret x ret}$  results in  $\frac{\partial T}{\partial x}$ . (*Mathmode* only.)

**@-i** Displays and prints a visible integral sign, and prompts for a lower limit (displayed as subscript) and an upper limit (displayed as superscript), each ended by  $\text{ret}$ ; for example,  $\text{@-i x=0 ret @-8 ret}$  results in  $\int_{x=0}^{\infty}$ . (*Mathmode* only.)

In addition, I have programmed a few keys of the cursor pad, and of the number key pad (not displayed in Fig. 1), e.g.:

**@-↑** Displays and prints a single-character superscript.

**Num-/** Produces a fraction  $\{\dots\}\over\{\dots\}$ , prompting for numerator and denominator, each ended by  $\text{ret}$ , for example  $\text{Num-/ x+2 ret x+1 ret}$  produces  $\frac{x+2}{x+1}$ .

#### GREEK and SINGLE keyboards

These keyboards contain the full Greek alphabet plus a number of math symbols as shown in Fig. 2. None of the function keys have been redefined, i.e., they contain the original definitions supplied by

WordPerfect. The SINGLE keyboard is identical to GREEK except that control is returned to NORMAL after each character, and is toggled by hitting the F9 key. Thus typing a F9 bc results in "a $\beta$ c".

#### FOREIGN, SMCAPS, CALLIGRAPHIC keyboards

For people who type in foreign languages I have defined the FOREIGN keyboard which allows you to type umlauts, etc. The keyboard is identical to NORMAL except that some characters have been replaced. At this point only some umlauts etc. have been redefined. For example, typing @-a gives ä, ^-a gives Ä, etc. The SMCAPS keyboard contains a displayable set<sup>2</sup> of *small caps* characters A, B, THROUGH Z. Control is returned to the normal keyboard by hitting <ret>. The CALLIGRAPHIC keyboard acts like SINGLE, i.e., after typing a single calligraphic character (upper case A through Z), control is returned to NORMAL automatically.

#### Equation example

To demonstrate how the system works for typing equations, consider the equation

$$f(x) = \frac{1}{4} \int_{\xi=0}^{\infty} \left[ \frac{g(x, \xi)}{1 - \xi^2} + 1 \right] \frac{\partial f}{\partial \xi} d\xi$$

which is typed as:

```
@-F11 F6 f F6 (x)=^-x 4^-i F9 x=0 ret
@-8 ret @- [ num-/ g(x, F9 x) ret 1- F9
x num-↑ 2 ret +1 @-] ^-p f ret F9 x ret
@-n d F9 x ret
```

and appears on the screen approximately as

```

$$f(x) = \frac{1}{4} \int_{\xi=0}^{\infty} \left[ \frac{g(x, \xi)}{1 - \xi^2} + 1 \right] p(f)(\xi) d\xi$$

```

(The big brackets actually appear "broken" on the screen to indicate their variable size).

#### 0.5 TeX Printer Drivers

As indicated in the description for the ^-F5 key above I have defined a number of "printers", which will take your WordPerfect files and translate them into TeX files (`tex.prs`), or into LATEX documentstyle files (`article`, `book`, `conference`, `letter`, `memo` or `sideways`). Besides translating TeX and LATEX commands (such as  $\alpha$  to `\alpha`, etc.), these drivers also add the necessary commands at the beginning and end of the TeX file.

<sup>2</sup>While the normal WordPerfect command for *small caps* may be used, this would not *display* them (because of the 6 font limitation)

#### 0.6 Custom Made TeX Macros

I have defined the following local macros which, therefore, are only available in my version of TeX/LATEX:

\flush#1#2#3 Identical to the prompted ^-F6, i.e., this macro will place the text #1 flush left, text #2 centered, and text #3 flush right,  
e.g. \flush{left}c{right}:  
left c right

\ls#1 This macro sets the linespacing, where #1 must be an integer number and sets the spacing in multiples of "jots" ( $1jot = 3pt = \frac{1}{4}$  of regular line spacing); the default is \ls4, i.e., one line every 4 jots or 12 points, translating to six lines per inch, while this paragraph is surrounded by {\ls6...} (notice the "1½-spacing"?). The unit of "jot" was chosen to make simple input (\ls followed by a single digit) possible for fractional values of line spacing. (Enclosing the paragraph in {...} reestablishes former linespacing beyond the }).

Note that this macro takes effect only as soon as TeX goes into "vertical mode" after it is called (i.e., between paragraphs); thus putting {\lsn ... } within a paragraph will have no effect at all.

\d#1#2 Total derivative of #1 with respect to #2 (mathmode only), e.g. \d Tx results in  $\frac{dT}{dx}$  (accessed via @-d)

\p#1#2 Partial derivative of #1 with respect to #2 (mathmode only), e.g. \p Tx results in  $\frac{\partial T}{\partial x}$  (accessed via @-p)

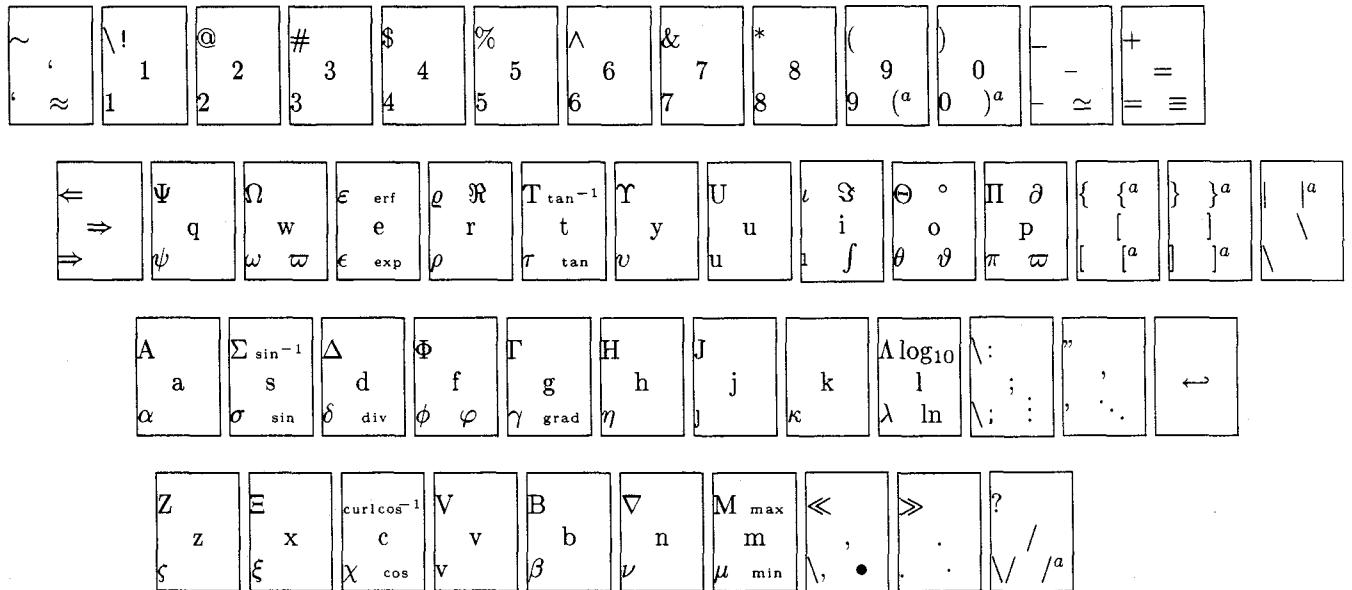
\dn#1#2#3 #1-th order total derivative of #2 with respect to #3 (mathmode only), e.g. \dn 4Tx results in  $\frac{d^4 T}{dx^4}$  (accessed via ^-d)

\pn#1#2#3 #1-th order partial derivative of #2 with respect to #3 (mathmode only), e.g. \pn 4Tx results in  $\frac{\partial^4 T}{\partial x^4}$  (accessed via ^-p)

\fr#1 #1-th fraction, (mathmode and normal mode), e.g. \fr4 →  $\frac{1}{4}$  (accessed via ^-x)

\Cases{#1} is identical to the normal TeX \cases {#1} command, but both columns are typeset in mathmode.

\txt#1 Small text between equations like "and" can be inserted by typing \txt{text} (accessed via ^-n)



<sup>a</sup> delimiters of size “Big”

Figure 2: The GREEK and SINGLE keyboards

### 0.7 Typing Memos With L<sup>A</sup>T<sub>E</sub>X

The `memo` document style is designed to make a number of memos at once and is really just a slightly modified L<sup>A</sup>T<sub>E</sub>X `letter` style file. The first page of each memo will have a L<sup>A</sup>T<sub>E</sub>X-generated version of the PENNSTATE logo at the top (which, obviously, other users may want to modify to their needs).

### 0.8 Custom Journal Styles

Every journal has slightly different ways to arrange the list of references, to refer to references within text, to refer to equations, etc. I have defined customized `.sty` files for four types of journals in which I usually publish, viz. the Transactions of the American Society of Mechanical Engineers (`asme.sty`), journals of the American Institute of Aeronautics and Astronautics (`aiaa.sty`), the International J. of Heat and Mass Transfer (`ijhmt.sty`), and the J. of Quantitative Spectroscopy and Radiative Transfer (as well as many other *Pergamon Press* journals) (`jqsrt.sty`). Of these four styles the first one (`asme`) orders references alphabetically, while the other three order them sequentially.

For these bibliography styles I have redefined or newly defined a number of L<sup>A</sup>T<sub>E</sub>X macros, which act differently for different journals. These macros are discussed under *Custom Made L<sup>A</sup>T<sub>E</sub>X Macros* below.

### 0.9 Custom Made L<sup>A</sup>T<sub>E</sub>X Macros

I have written a number of T<sub>E</sub>X macros which act together with L<sup>A</sup>T<sub>E</sub>X macros, mostly for labeling and

referencing. The following macros will behave differently depending on the journal style option chosen, i.e., `asme`, `aiaa`, `ijhmt` or `jqsrt`:

`\cite{key(s)}` will place the reference number (or year) into the text (i.e., the year within parentheses for `asme`, superscripted number for `aiaa` and `jqsrt`, and bracketed number for `ijhmt`). In addition, a BIBT<sub>E</sub>X command will be placed for those that use a bibliography database (accessed via `\@c`).

**Note:** for the `asme` style the `\bibitem` command must have the year of publication as a label, i.e., `\bibitem[year]{key}`. (Of course, for BIBT<sub>E</sub>X users this is taken care of automatically).

`\Cite{key(s)}` similar to `\cite` but the numbers are large and accompanied by “Ref.” or “Refs.”. Obviously, this macro is not defined for the `asme` style (since references are not numbered).

`\capcite{key(s)}` identical to `\cite` except that it only *reads* the `.aux` file (to display the correct reference number or whatever), but does not *write* an entry into the `.aux` file; therefore, `\capcite` can only be used if `\cite` is used elsewhere for the same reference; `\capcite` may be used in places where `\cite` cannot, primarily within figure and table captions.

`\e{key}` same as the `\ref` command but surrounds the number with parentheses.

\eq{key} will write something like “equation (*n*)” (depending on the journal), where *n* is the referenced equation (accessed via  $\text{\^e}$ ).

\eq+{key1, key2} will write something like “equations (*n*1) and (*n*2)” (depending on the journal), where *n*1 and *n*2 are the referenced equations (accessed via  $\text{\^e}$ ).

\eq-{key1, key2} will write something like “equations (*n*1) through (*n*2)” (depending on the journal), where *n*1 and *n*2 are the referenced equations (accessed via  $\text{\^e}$ ).

\eqs{key} will write something like “equations (*n*)” (depending on the journal), where *n* is the first referenced equation (to be used with \e if some other inter-equation text than “and” or “through” is desired) (accessed via  $\text{\^e}$ ).

\Eq{key}, . . . All of the above \eq... commands exist also as \Eq... commands for equation references at the beginning of a sentence (accessed via  $\text{\^e}$ ).

\fg{key}, \fg+{key}, \fg-{key}, \fgs{key}, . . . All the \eq commands are repeated for figures. In this case all four journals have the same format, i.e., the \fg commands translate to “Fig. *n*” or “Figs. *n*”, while the \Fg command translates to “Figure *n*” and “Figures *n*” (accessed via  $\text{\^f}$ ).

\tb{key}, \tb+{key}, \tb-{key}, \tbs{key}. All the lower case \fg commands are repeated for tables. In this case all four journals have the same format, (and the upper case version does not exist since there is no difference between beginning-of-sentence and mid-sentence reference), i.e., the \tb commands translate to “Table *n*” and “Tables *n*” (accessed via  $\text{\^t}$ ).

Another group of macros that I have written deals with cross-referencing between different documents. For example, if you write a long report or a book, you may want to break up the documents into parts or chapters, but without losing the ability to reference equations/figures/pages etc. from other parts, keeping up continuous page/chapter etc. numbering, preparing an overall table-of-contents, etc.:

\crossref{file(s)} will read the files file(s).crf (if the entry is non-empty), and will open the new file present-file-name.crf;

\closecrf will close the file present-file-name.crf (if it is open, i.e., if \crossref{...} has been called within the document. This command should come after the \end{document} (and is automatically included in my WordPerfect book documentstyle);

\TOC will open and prepare a present-file-name.toc file similar to the \tableofcontents command, but will not actually print the Table of

Contents (eventually one can splice all .toc files together to make a single Table of Contents);

\LOF is similar to \TOC, and is designed to be used instead of the \listoffigures command;

\LOT is similar to \TOC, and is designed to be used instead of the \listoftables command.

\Section is equivalent to the \section\* command, but places an unnumbered entry into the Table of Contents;

\Label is equivalent to the \label command, but places a cross-referencing line not only into the .aux file, but also into the .crf file;

\plcounter{\#1} places the value of a counter into the .crf file, (which can later be read by the \crossref command from a different document part). #1 may be page (page number), chapter (chapter number), section (section number), subsection (subsection number), figure (figure number), table (table number), or equation (equation number). When placed at the very end of a file (last entry before the \end{document} command) this transmits to the \crossref command the latest page number etc., making continuous numbering possible.

\widen{\#1} increases the document \textwidth by #1, and decreases both margins by  $\frac{1}{2} \times \#1$ . This macro comes in handy when L<sup>A</sup>T<sub>E</sub>X breaks pages in such a fashion that only one or two lines end up on the last page (particularly in letters and memos).

## 0.10 BIBTeXing

Since the standard L<sup>A</sup>T<sub>E</sub>X formats are geared towards mathematical and computer journals (styles plain and unsrt are available), I have written a few more styles, viz. asme, aiaa, jqsrt, ijhmt, i.e., drivers for the aforementioned journals, which are primarily of interest to engineers and physicists. Others could be customized easily.

## 0.11 Availability

I am prepared to supply anybody who would like it with the software that I have developed (i.e., you supply your own WordPerfect and T<sub>E</sub>X). To receive the software you need to send \$10 (to cover diskettes and postage), and specify whether you want 360kB or 1.2MB 5.25in floppies, or a 1.44MB 3.5in diskette. In addition I would appreciate any suggestions on how to further improve the present WordPerfect-to-T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X preprocessor.

## Macros

### A Page Make-up Macro

Joost Zalmstra\*

David F. Rogers\*\*

In a previous article in TUGboat (Ref. 1) it was pointed out that using *TeX* to produce heavily illustrated mathematical, engineering and scientific books containing large numbers of complex display equations and tables poses problems. The reason is that publishers have stringent requirements for the placement of figures with respect to the text while neither Plain *TeX* nor *L<sup>A</sup>T<sub>E</sub>X* provides good control of this placement.

Currently a set of macros is being developed to help in solving this problem. The macros are based on a FIFO queue. Basically, the algorithm is quite simple. When making up a page the FIFO queue is checked. If the queue is not empty and if the first figure on the queue fits on the current page, it is removed from the queue and placed on the current page at the top. If the queue is still not empty, and the second figure on the queue also fits on the page it is removed from the queue and placed at the bottom of the current page. If the figures do not fit on the current page, the page is made-up normally and the process repeated for the next page. If a new figure is encountered, and if the queue is not empty, the figure is placed on the queue. If the queue is empty, and if the figure fits on the current page, it is placed on the current page, either at the top or at the bottom. If the figure does not fit on the current page, it is placed on the queue and the page made-up normally.

In implementing the algorithm in Plain *TeX*, a form of floating keep with the following properties is introduced:

The figure and any caption must be placed in a *\vbox*.

The order in which these *\vbox*-es appear in the text is maintained.

No more than two *\vbox*-es appear on one page, one at the top and one at the bottom of the page.

---

\*Vrije Universiteit, Amsterdam, The Netherlands, *mcvax!nat.vu.nl!jjaz@uunet.uucp*

\*\*United States Naval Academy, Annapolis, MD, USA, *dfr@usna.mil*

These characteristics satisfy the criteria presented by Rogers in Reference 1. The floating keep is implemented as a macro *\addfigure* that is called with one parameter: the number of a box-register that contains the figure *\vbox*. If the *\vbox* can not appear on the current page (either there are already two figures on this page or there are figures left over from a previous call to *\addfigure*) then *\addfigure* puts the *\vbox* on a list (FIFO queue) implemented as the box-register *\figlist*. Otherwise, it tries to fit the box on the current page, taking into account *\pagetotal*, *\pagegoal* and the size of the figure. The figure size includes the space separating the *\vbox* from the text. If the figure won't fit, it is added to *\figlist*. If the figure fits on the page the *\vbox*, together with the space separating it from the text, is inserted on the current page using Plain *TeX*'s *\topinsert*. The queue (*\figlist*) is implemented using *TeX*'s box operations. An element (figure) is added to the queue by placing it at the top of the existing vertical list of boxes. The 'head' of the queue is accessed via *TeX*'s *\unvbox* and *\lastbox* commands.

The output routine splits the insertion, if necessary, and places the first figure at the top of the page and the second, if present, at the bottom. After the *\shipout* for the current page occurs, the output routine checks *\figlist* to see if there are any figures left. If there are, it attempts to insert a maximum of two figures on the next page, again using *\topinsert*. Before calling *\topinsert* the total size of the figures on the page is checked to insure that it does not exceed *\dimen\topins*. If this is not done, *\topinsert* would shift the insertion to the next page, thus producing a figure on that page without *\addfigure* knowing about it.

The space separating the figure from the text is specified by two skip registers: *\topfigureskip* and *\bottomfigureskip*. The first specifies the *\vskip* below a figure placed at the top of the page, the second the *\vskip* above a bottom figure. Note that this *\vskip* must not be included in the *\vbox* offered to *\addfigure*.

There are several refinements to the basic algorithm that have been implemented and are worth mentioning:

If the amount of text remaining on a page after insertion of the figure(s) is less than a certain amount (specified by the dimen register *\minpagesize*) then no text is placed on the page. If two figures appear on that page they are placed at the top and bottom of the page.

If there is only one figure on that page, it appears at the bottom of the page.

The switch `\ifbottomforce` is used to force a bottom figure. If `\bottomforcetrue` then if `\addfigure` finds that it can fit a figure on the current page as the first figure (which would normally be placed at the top of the page) then it forces the figure to the bottom of the page (by inserting an empty `\vbox` first, which appears at the top of the page). This feature can be used to insure that the figure appears after the first reference to it.

If the `\vbox` offered to `\addfigure` has a height of exactly `\vsize` then it is considered to be a page sized figure. No extra `\vskip` is added. `\pageinsert` is used to insert it into the text.

For completeness, a set of macros to create the space for a figure and to set the figure caption is also presented. These macros are called from within the macro `\figspace`. The `\figspace` macro has four calling parameters:

The vertical dimension of the white space to be left for the figure.

The horizontal dimension of the white space to be left for the figure.

The figure number.

The figure caption.

If the horizontal size of the figure (parameter 2) is smaller than a certain value specified by `\sidearttest` then `\sidefig` is called and the caption is placed to the right of the figure (separated from it by `\sideartspace`). Otherwise `\normalfig` is called and the caption appears below the figure separated by `\artsurround`.

The `\addfigure` and `\figspace` macros are combined by calling `\figplace` with the same parameters as `\figspace`. This creates space for the figure, sets the caption, places the result in a `\vbox` and calls `\addfigure`. Note that `\addfigure` and `\figspace` are completely independent. Changing the `\figspace`-macros has no influence on the working of `\addfigure`.

The macros have been used both with test files and with production text (Ref. 2). The production text consisted of three separate files. The complexity of the page make-up task is indicated by the details given in Table 1. Page size was 46 pc by 29 pc set 10/12 using the Computer Modern Fonts. The figure space quoted in Table 1 does not include space for the figure captions nor that

for `\topfigureskip`, `\bottomfigureskip` or `\artsurround`. The data in Table 1, plus allowance for these spaces, indicates that approximately 25% of the available space is occupied by illustrations. This fact, plus the large numbers of display equations, many of which are 4 to 8 row matrices within an `\eqalign`, illustrates the complexity of page make-up for these files.

**Table 1. Page Make-up Complexity**

	File 1	File 2	File 3
No. Pages	40	106	40
No. Figs.	16	50	16
Total Figure Space	357 pc	1255 pc	280.5 pc
Average Figure Space	22.3 pc	25.1 pc	17.5 pc
Smallest Figure	11 pc	6.5 pc	5 pc
No. Display Equations	295	380	250
No. Tables	1	0	10
No. Direct Calls to <code>\normalfig</code>	0	7	2

The `\figplace` macro performed very well. Calling `\figplace` immediately after the first reference to the figure often resulted in acceptable placement of the figure. Placement of the figure can be changed by changing the location of the call to `\figplace` in the source file. Occasionally when `TEX` was particularly recalcitrant, it was necessary to call `\normalfig` or `\sidefig` directly.

There are still some small bugs known to us and some improvements that can be made.

Since the vertical size of a figure caption is not known *a priori*, there is no way to initially exactly specify a full page figure. At least one pass through `TEX` must be made. The vertical size of the figure caption is then measured and the value of the first parameter in `\figplace` adjusted accordingly. To solve this problem `\figplace` is called with a vertical size (parameter 1) of 0. The `\figspace` macros then change the vertical size to produce a full page figure. Directly incorporating this capability into `\figplace` to maintain complete independence of the two macros is desirable.

If `\raggedbottom` is used, the vertical size of pages may vary. However, `\figplace` always makes full page figures exactly the specified `\vsize`.

Verso and recto pages may then be unbalanced (of different vertical lengths). The workaround is to call `\figplace` within a `\vbox` of the required size. However, if the required size is greater than the specified `\vsize`, an overfull `\vbox` results and `TEX` prints the appropriate message. The message can be ignored since the page is set and output correctly.

`\figplace` is limited to a maximum of two figures on a page. However, difficult make-up situations or reader convenience may dictate that more than two figures be placed on the page. For example, it may be desirable to group several small figures together at either the top or bottom of the page. Here, the workaround is to use `\figspace` directly, place the small figures together in a `\vbox` and call `\addfigure` directly, e.g.,

```
\setbox0=\vbox{
    \figspace{#1a}{#2a}{#3a}{#4a}
    \figspace{#1b}{#2b}{#3b}{#4b}
}
```

`\addfigure0`

The `\vbox` containing the figures is added to `\figlist` and processed in sequence.

The `\sidefig` macro assumes that the illustration is left justified on the page with the caption set to the right. A companion macro with the illustration right justified on the page with the caption set to the left is needed.

A macro to set two or more small figures side-by-side with the captions below is also needed. Here, the calling sequence might be

```
\sidebysidefig#1#2#3#4#5#6#7#8
```

where items 1–4 correspond to those of `\figplace` for the lefthand figure and 5–8 to those of `\figplace` for the righthand figure.

A major problem is the fact that `\addfigure` must be called between paragraphs. The reason is that `\addfigure` calculates the space available on the current page from `\pagetotal` and `\pagegoal`. However, at the moment that these are evaluated they have the current value at the start of the paragraph.

Despite the problems remaining, the macros seem to do the job. Hopefully they will be useful to other people and will stimulate further discussion of the solution to the page-makeup challenge.

## References

1. Rogers, David F., A Page Make-up Challenge, *TUGboat*, Vol. 9, No. 3, pp 292–293, December 1988.
2. Rogers, David F. and Adams, J. Alan, *Mathematical Elements for Computer Graphics* 2nd Edition, McGraw-Hill, New York, 1989.

## The macros `figplace.tex`

```
% name: figplace.tex
% programmer: Joost Zalmstra
% version/date: Version(1.8)  of 89/16/1 08:30:00
% language: Plain \TeX
% purpose: Implementation of a floating keep that allows
%           better control of the placement of figures
% history: Based on David Rogers tests and comments the original macro
%           was modified as follows:
%           Added separate \topfigureskip and \bottomfigureskip to partially
%           solve the problem of unequal spaces between a topfigure and the
%           text and a bottomfigure and the text.
%           Corrected problem of loss of all subsequent figures if a
%           figure + caption is larger than \vsize.
%           Added code to approximately center the text between a topfigure and
%           a bottomfigure on the same page.
%           Added code to push a figure larger than \vsize - \minpagesize to the
%           bottom of the page.
%           Added code to force a single figure on the page to be at the
%           bottom of the page.

% declaration of some registers
```

```

\newif\ifbottomforce      \bottomforcefalse
\newbox\figlist
\newcount\figuresonpage   \figuresonpage=0
\newcount\figuresonlist   \figuresonlist=0
\newskip\topfigureskip    \topfigureskip=1.25 pc
\newskip\bottomfigureskip \bottomfigureskip=1.5 pc
\newdimen\pagesize
\newdimen\minpagesize     \minpagesize=5 pc
\newdimen\figuresizeonpage
\input figspace.tex%       % \figspace is the macro that actually sets the white space
%                           % for the art and the figure caption.
%
%                           % It is independent of \figplace.

% user interface for inclusion of figures
\catcode`@=11%             % borrowing macros from plain TeX
% Note: the redefinition of \raggedbottom has nothing to do with
%        the implementation of \addfigure. It is only placed here
%        because \raggedbottom uses @ as a letter
\def\raggedbottom{\topskip10pt plus 24pt \r@ggedbottomtrue}
\def\figplace#1#2#3#4{%
  \setbox0=\vbox{%
    \figspace{#1}{#2}{#3}{#4}
  }
  \addfigure0
}
% add figure in box #1 to the document
\def\addfigure#1{%
  \dimen0=\ht#1
  \ifdim\dimen0>\vsize%           % be sure it will fit
    \message{figure too big}
    \setbox#1=\vbox to \vsize{\box#1}
  \fi
  \ifnum\figuresonpage>1%          % already two figures on this page
    \addtolist{#1}%                % so save for later
  \else\ifnum\figuresonlist>0%      % figlist is not empty
    \addtolist{#1}%                % so save for later
  \else%
    \dimen0=\ht#1
    \ifdim\dimen0<\vsize%          % not a pagesize figure
      \advance\dimen0 by \dp#1%     % size of the box
      \ifnum\figuresonpage=0
        \advance\dimen0 by \topfigureskip%   % extra space below
      \else
        \advance\dimen0 by \bottomfigureskip% % extra space above
      \fi
    \fi
    \dimen1=\dimen0
    \advance\dimen0 by \pagetotal
    \ifdim\dimen0>\pagegoal%        % no space on this page
      \addtolist{#1}%                % so save for later
    \else
      \advance\dimen1 by \figuresizeonpage
      \ifdim\dimen1>\dimen\topins%    % exceeding maximum insertion size
        \addtolist{#1}%                % so save for later
      \fi
    \fi
  \fi
}

```

```

\else%                                % insert figure
  \ifnum\figuresonpage=0%              % force a bottom insert
    \ifbottomforce%                  % by adding an empty box first
      {\setbox1=\vbox{%
        \vskip -\topfigureskip}% compensate for figureskip
        \figureinsert1}
    \fi
  \fi
  \figureinsert{\#1}%                % insert the figure
\fi
\fi\fi
}

% \addtolist adds \box#1 to the FIFO queue \figlist
\def\addtolist#1{%
  \global\setbox\figlist=\vbox{%
    \nointerlineskip
    \box#1
    \unvbox\figlist}
  \global\advance\figuresonlist by 1
}

% \figureinsert inserts \box#1 using Plain TeX's \topinsert
% and updates the registers \figuresizeonpage and \figuresonpage
\def\figureinsert#1{%
  \dimen0=\ht#1
  \advance\dimen0 by \dp#1
  \ifdim\ht#1<\vsize%                 % not a full page figure
    \ifnum\figuresonpage=0
      \advance\dimen0 by \topfigureskip
      \insert\topins{\vbox{%
        \unvbox#1\vskip\topfigureskip}}%   % skip below the figure
    \else
      \advance\dimen0 by \bottomfigureskip
      \insert\topins{\vbox{%
        \vskip\bottomfigureskip\unvbox#1}}% % skip above the figure
    \fi
  \else%                                % full page figure
    \pageinsert
    \unvbox#1
  \endinsert
  \fi
\global\advance\figuresonpage by 1
\global\advance\figuresizeonpage by \ht#1
\global\advance\figuresizeonpage by \dp#1
\ifnum\figuresonpage=0
  \global\advance\figuresizeonpage by \topfigureskip
\else
  \global\advance\figuresizeonpage by \bottomfigureskip
\fi
}

% see if there is a figure in the list that will fit on the next page
% return \box0 containing the figure or void if no figure will fit
\def\checkfiglist{%
  \ifnum\figuresonlist>0%               % there are figures on the queue

```

```

\global\setbox\figlist=\vbox{%
  \unvbox\figlist
  \setbox0=\lastbox%                                % \box0 contains first figure
  \global\advance\dimen0\ht0
  \ifdim\ht0<\vsize
    \global\advance\dimen0 by \dp0
    \ifnum\figuresonpage=0
      %
      % topfigure may loose \figureskip
    \else
      \global\advance\dimen0 by \bottomfigureskip
    \fi
  \fi
  \ifdim\dimen0>\vsize%                            % test if it will fit
    \vbox{\box0}%                                    % no room ; replace the figure
  \else
    \ifdim\dimen0>\dimen\topins
      \vbox{\box0}%                                % exceeding maximum insertion size
    \else%
      \global\advance\figuresonlist by -1
    \fi
  \fi
  \global\setbox0=\box0
}
\else%
  {\global\setbox0=\box0\voidb@x}                  % no figures left on the queue
\fi
}

% output routines
% note: these are just modifications of the Plain \TeX output routines
\output{\myoutput}
\def\myoutput{\shipout\vbox{\makeheadline\pagebody\makefootline}%
  \advancepageno
% reset \addfigure registers
\global\figuresonpage=0
\global\figuresizeonpage = Opt
\dimen0=Opt%                                     % \dimen0 contains \figuresizeonpage
\checkfiglist%                                    % check if there is figure waiting that will fit
\ifvoid0%                                         % nothing left
\else
  \figureinsert0%                                 % insert this figure
  \global\advance\dimen0 by \topfigureskip
  \checkfiglist%                                  % check for a second figure
  \ifvoid0\else
    \figureinsert0
  \fi
\fi
\ifvoid255\else\unvbox255\penalty\outputpenalty\fi%  % put unused page back on list
\ifnum\outputpenalty>-1\else\dosupereject\fi}

% actual contents of this page
\def\pagecontents{%
  \pagesize=\ht255
  \advance\pagesize by \dp255%                      % determine size of box 255
  \ifnum\figuresonpage>0%
    \unvbox\topins%                                % check for insertion of figures
    % place the figure(s) on top
}

```

```

\ifdim\pagesize>\minpagesize
  \ifnum\figuresonpage=2
    \setbox0=\lastbox%           % place the second figure in \box0
  \fi
  \setbox1=\lastbox
  \unvbox1%                   % remove surrounding box
\else%
  \ifnum\figuresonpage=1
    \setbox0=\lastbox
    \vfill%                     % figure becomes bottomfigure
    \unvbox0%                   % remove surrounding box
    \unskip%                   % remove \topfigureskip
  \fi
\fi
\fi

% topfigure is placed on this page
% put the text on this page if it is large enough
\ifdim\pagesize>\minpagesize
  \dimen0=\dp255 \unvbox255%   % the real page contents
  \ifr@ggdbottom \kern-\dimen0\vfil\fi
\else
  \ifdim\ht255=0pt%           % box is empty
  \else
    \vfill%                   % too small so leave for next page
  \fi
\fi

% Produce the second figure, if present
\ifnum\figuresonpage=2
  \box0%                      % second figure
\fi

% produce possible footnotes
\ifvoid\footins\else%
  \vskip\skip\footins
  \footnoterule
  \unvbox\footins\fi
}

\catcode`@=12%                 % at signs no longer letters

```

## The macros figspace.tex

```

% name: figspace.tex
% programmer: David F. Rogers
% version/date: Version(1.6) of 89-16-01 08:30:00
% language: Plain \TeX
% purpose: To provide white space for stripped in art and set the caption for the art
% history: Modified by Joost Zalmstra to work more efficiently with the \figplace macro
%          Added code to check for nearly page size figure, expand it to fill the page
%          and print a message to that effect.
%          Added code to check for figure + caption larger than page size, to
%          reduce the figure size (parameter #1) so that the art plus caption just
%          fills the page and to print a message to that effect.

```

```
% define top and bottom cornerrules to indicate the size of the art
\newdimen\ruleht      \ruleht=0.5pt%      % take the rules out for final
\newdimen\ruledpth    \ruledpth=0pt%      % production by setting
\newdimen\rulelength  \rulelength=1pc%     % \ruleht=0pt and \rulelength=0pt
\def\settopcornerrules#1{\hbox to #1{\vrule width\rulelength height\ruleht
    \hfil \vrule width \rulelength}
\hbox to #1{\vrule height\rulelength width\ruleht depth\ruledpth
    \hfil \vrule width\ruleht depth\ruledpth}}
\def\setbotcornerrules#1{%
    \vskip -2\rulelength%
    \hbox to #1{\vrule height\rulelength width\ruleht depth\ruledpth%
        \hfil \vrule width\ruleht depth\ruledpth}%
    \vskip -\rulelength%
    \hbox to #1{\vrule width \rulelength height\ruleht%
        \hfil \vrule width \rulelength}
}

% define a figure caption macro
% #1 is the figure number #2 is the caption
% the caption is to be set in a 'box' left and right justified 1em to
% the right of the figure number
% the size to the box containing the word Figure, its number and the 1em
% skip is found in box0
% box1 is \hsize less the width of box0.
% a \vtop is used along with an \halign to obtain the flush left and right
% effect.
\def\figcap#1#2{%
    \setbox0=\hbox{{\bf Figure } #1}\hskip 1em}%
    \setbox1=\vtop{\advance \hsize by -\wd0 \noindent
        \spaceskip=.3em plus .2em minus .2em
        #2}
    \halign{## & ## \cr
        \box0 & \box1 \cr}
    \leavevmode
}

% define a figspace macro
% A 1.5pc space is required above the art.
% If a figure is more than 13pc wide then the caption is to be set
% below the art with a 1.5pc space between the art and the figcap.
\newdimen\artsurround \artsurround=1.5pc%   % space between art and the caption
\newdimen\sidearttest \sidearttest=13pc%     % if more than set side caption
\newdimen\sideartspace \sideartspace=2em%     % horizontal space to caption
\def\figspace#1#2#3#4{%
    \ifdim #2 > \sidearttest%
        \normalfig{#1}{#2}{#3}{#4}
    \else
        \sidefig{#1}{#2}{#3}{#4}
    \fi
}
% this is the case when the figure is wider than \sidearttest (13 pc's)
\def\normalfig#1#2#3#4{%
    % #1 is vertical size in pc's
    % #2 is horizontal size in pc's
    % #3 is the figure number -- used in figcap
    % #4 is the figure caption -- used in figcap
```

```

\setbox0=\vbox{\figcap{#3}{#4}}
\dimen0=\ht0
\advance\dimen0 by \dp0
\advance\dimen0 by \artsurround
\dimen1=#1
\ifdim\dimen1 = 0 pt%           % figure should fill the rest of the page (jz)
    \dimen1 = \vsize
    \advance\dimen1 by -\dimen0
    \message{expanding figure to \the\dimen1}
\else
    \advance\dimen0 by \dimen1
    \ifdim\dimen0>\vsize%        % figure + caption larger than page size (jz)
        \advance\dimen0 by -\vsize
        \advance\dimen1 by -\dimen0
        \message{warning: reducing vertical figure size to \the\dimen1}
    \fi
\fi
\leftline{%
    \vbox to \dimen1{%
        \settopcornerrules{#2}%   % set top corner rules
        \vfill%                  % push the figcap down to bottom
        \setbotcornerrules{#2}%  % set bottom corner rules
    }%
}%
\nointerlineskip
\vskip \artsurround%           % put space between figure & caption
\box0%                         % set the caption
}

% this is the case when the caption is to be set to the right
\def\sidefig#1#2#3#4{%
%
%                                #1 is vertical size in pc's
%                                #2 is horizontal size in pc's
%                                #3 is the figure number -- used in figcap
%                                #4 is the figure caption -- used in figcap
\ifdim#1>\vsize%             % figure + caption larger than page size (jz)
    \dimen0=\vsize
    \message{warning: reducing vertical figure size to \the\dimen0}
\else
    \dimen0=#1
\fi
\setbox0=\vbox to \dimen0{%
    \advance \hsize by -#2%      % reduce \hsize by the horiz.
    \advance \hsize by -\sideartspace%  % size and blank space between
    \vfill%                      % push the figcap down to bottom
    {\figcap{#3}{#4}}%          % of the box
}
\setbox1=\hbox to #2{\vbox to \ht0{%
    \settopcornerrules{#2}%    % set top corner rules
    \vfill%                  % push the figcap down to bottom
    \setbotcornerrules{#2}%  % set bottom corner rules
}}%
\hbox{\box1 \hskip \sideartspace \box0}%  actually do the setting
}

```

## Extended Equation Numbering in Plain TeX

James Nearing  
Physics Department  
University of Miami

This is a macro package designed to provide automatic equation numbering.\* It has various options: allow chapter (or section) numbers; allow forward references to equations or figures; allow equations to be labeled by chapter number as well as equation number; allow automatic lettering of appendices.

The five commands used for equation numbers and for reference to the equations are

```
\eqnum
\eqalignnum
\eqlabel{...}
\eqalignlabel{...}
\eqref{...}
```

The first (`\eqnum`) is the simplest. Simply place it at the end of a displayed equation, and a new equation number will be put at the end of the equation. The third (`\eqlabel{...}`) not only inserts the correct equation number, it assigns a label of your choice to the equation so that you can refer to it in the text. You will use another command (`\eqref`) to refer to the equation by label instead of by number; future editing will not then affect the reference.

Example:

```
$$
\int_0^\infty dx e^{-x^2} = \sqrt{\pi}/2
\eqnum
$$
$$
p_x = -i\hbar \frac{\partial}{\partial x}
\eqlabel{momentum}
$$
```

will produce

$$\int_0^\infty dx e^{-x^2} = \sqrt{\pi}/2 \quad (7)$$

$$p_x = -i\hbar \frac{\partial}{\partial x} \quad (8)$$

The second and fourth commands are like the first and third, except that they are to be used in another context. When you want numbers on an *aligned* equation, the syntax is different; you must use these instead to produce the same results. They are used when you use the `\eqalignno` command to get a numbered aligned equation. For example,

```
 $$\eqalignno{%
\sqrt{-1}&=2+1\cdot 5+2\cdot 5^2+\cdots \\
\eqalignnum\cr
(p\llap{/}\!-\!m)\psi&=0 \\
\eqalignlabel{zap}\cr
} $$
```

will produce

$$\sqrt{-1} = 2 + 1 \cdot 5 + 2 \cdot 5^2 + \dots \quad (9)$$

$$(p - m)\psi = 0 \quad (10)$$

References to equations produced as above are done by `\eqref{...}` where the dots are the label that was introduced when either of the commands `\eqlabel` or `\eqalignlabel` are used. The sequence `\eqlabel{momentum}` will produce (8) and `\eqlabel{zap}` will be (10).

If you use the same label for two different equations, the first one is discarded as soon as the second one appears. Sometimes you will want to refer to a particular equation immediately after its appearance, but you know that you will never refer to it again. In this case it is convenient to pick some standard, simple label for continual re-use. For example you can use `\eqlabel{x}` or `\eqalignlabel{x}` any time this circumstance arises.

Another simple but useful command in this series is `\eq`. It is nothing more than an abbreviation for "Eq. ~". Equation references in the text will then appear as `\eq\eqref{...}`, thereby generating the appearance of "Eq. (10)" in the final document.

When you first use the macros as stated above, you should set the system to operate in "proof-mode." This shows the equation labels you have defined in the margin; it puts the date, filename, and pagenumber in the headline; it pushes the text to the left on the page; if you have forward references, it stops the error messages that would result from an undefined equation label. To do this, use the set of three commands

```
\input eqmacros
\proofmodetrue
\initialeqmacro
```

If you want the option of referring to equations that have not yet appeared in the text, there is another step to take. The setup is

```
\input eqmacros
\proofmodetrue
\forwardreferencetrue
\initialeqmacro
```

In this case, you *must* also create in your directory an empty file with the name `LABELFILE.TEX`. This

---

\* An earlier article by J. E. Pittman, TUGboat Vol. 9, No. 3, showed a more limited method for automatic equation numbering.

is needed because the program must write out the equation labels to a file in order to have access to them the next time that *TEX* is run.

When you are ready for a final copy, simply modify `\proofmodetrue` to `\proofmodefalse`.

You may prefer a style in which the text is divided into chapters or sections. You may even want all of the equations to be marked in this way, as (1.5) and (4.2). The internal variable `\chapno` is used for this purpose. It is started at zero; when you begin a new chapter it is incremented and numbered with one of the commands

`\chapnum`    or    `\chaplabel{...}`

They are used in a way exactly parallel to `\eqnum` and `\eqlabel`. There is a corresponding command `\chapref` used to refer to a labeled chapter. Instead of using `\chapnum` or `\chaplabel` directly, it is typically better to use a macro for the section or chapter titles. This makes the presentation uniform. Examples of such commands are

```
\centerhead{label}{Title}
\lefthead{label}{Title}
\bighead{label}{Title}
```

They provide labels for reference to the chapter, and give standard formats for the appearance of the title. These are macros in the `eqmacros.tex` package, so you can use them as models for the particular format that you want.

If you want the appendices to be labeled by *letters* instead of *numbers*, place the command

`\chapno=-1`

at the beginning of the chapter, just before a command such as `\chapnum` or `\centerhead`.

If you want to use the chapter number (or letter) in a headline, use the command `\chapfolio`. This will give a number or a letter as appropriate.

If you do nothing else, then the equation numbers will only contain a single number, as (7). If you prefer that all the equations have both the chapter number and equation number (3.7) or (B.7), place the command `\chapternumberstrue` in your file. In either case, when you use `\eqref` to refer to an equation in another chapter it will always use the full form (2.9).

For figures, the corresponding commands are

```
\fignum  \figlabel{...}  \figref{...}
\figurechapternumberstrue
```

## Example

```
\input eqmacros
\proofmodetrue
\forwardreferencetrue
\chapternumberstrue
\initialeqmacro
\centerhead{intro}{Introduction}
Some introductory text. We shall see
in section \chapref{venus} that $2+2$ is
indeed 4, but in section \chapref{mars} that it is not.
```

```
\centerhead{venus}{The Ineffable All}
$$ a+b=c\eqnum $$
but obviously we see that
$$
c+d=e\eqlabel{cde}
$$
implying that \eq\eqref{cde} tells it all. \eq\eqref{deltatron} will imply even more.
```

```
\lefthead{zap}{The Contradiction}
This header for the section is
equivalent to a combination of commands
“noindent,” “boldface,”
“chaplabel,” and “hfil break.”
$$\eqalignno{%
q+w&=e\eqalignlabel{x}\cr
1+1&=0\eqalignlabel{xx}\cr
}$$
```

```
\chapno=-1
\bighead{apple}{Appendix: Still More}
and this is still another macro to
place a heading. Here, the chapters are
lettered because the chapternumber,
\chapno, is negative.
etc.
```

## Summary

\eqnum	Number this equation, but make no reference to it.
\eqlabel{alabel}	" " " , and set up a reference to it.
\eqalignnum	like \eqnum but in the context of \eqalignno
\eqalignlabel{alabel}	" \eqlabel " " " " " "
\eqref{alabel}	refer to a labeled equation.
\chapnum	Number this chapter, but make no reference to it.
\chaplabel{alabel}	" " " , and set up a reference to it.
\chapref{alabel}	refer to a labeled chapter.
\fignum	Number this figure, but make no reference to it.
\figlabel{alabel}	" " " , and set up a reference to it.
\figref{alabel}	refer to a labeled figure.
\eq	Eq.~ convenient if you change your mind
\eqs	Eqs.~
\fig	Figure~
\figs	Figures~
\centerhead{alabel}{TITLE}	Typical macros for labeled titles
\lefthead{alabel}{TITLE}	
\bighead{alabel}{TITLE}	
\input eqmacros	To use these macros:
\proofmodetrue	Initially set to true, later false
\forwardreferencetrue	If you plan on any, set true
\chapternumberstrue	If you want equation numbers like (3.2)
\figurechapternumberstrue	" " " figure " " " "
\initialeqmacro	To initialize the setup:
\bumpchapno	Like \chapnum, but doesn't display number
\continuousnumberingtrue	If you DON'T want eq numbers to restart fresh
\continuousfigurenumberingtrue	" " " " fig " " " "
\chapfolio	The current chapter number or letter
\today	Today's date (\proofmode puts it in headline)
\chapno=-1	Starts LETTERING instead of NUMBERING

## Macros

```

\newif\ifproofmode                                % true => wide right margin, eqn.
\proofmodefalse                                     % labels shown; date in headline

\newif\iffforwardreference                         % true => allow reference to an
\fforwardreferencefalse                           % equation that appears later

\newif\ifchapternumbers                            % true => equations labeled as e.g.
\chapternumbersfalse                             % (3.7) instead of (7)

\newif\ifcontinuousnumbering                      % true => don't reset numbering of
\continuousnumberingfalse                        % equations in each chapter

\newif\iffigurechapternumbers                     % true => figures labeled as e.g.
\figurechapternumbersfalse                      % (3.7) instead of (7)

\newif\ifcontinuousfigurenumbering                % true => don't reset numbering of
\continuousfigurenumberingfalse                 % figures in each chapter

\font\eqsixrm=CMR6                                 % given these odd names to avoid
\font\eqtwelverm=CMR12                            % possible conflict with a user.
\def\marginstyle{\eqsixrm}                         % for proofmode labels

\newtoks\chapletter                               % for appendices
\newcount\chapno                                  % chapter number
\newcount\eqlabelno                               % equation #'s
\newcount\figureno                                % figure numbers

\chapno=0
\eqlabelno=0
\figureno=0

% This returns either a number or a letter depending on the sign
% of \chapno. It uses the CURRENT value of \chapno. Typically used
% only by other macros. The user may however, want to place the chapter
% number (or letter) into the headline, then \chapfolio should be used.
%
\def\chapfolio{\ifnum \chapno>0 \the\chapno \else \the\chapletter \fi}

% This increments \chapno in the correct direction (more positive OR
% more negative). If as is normal, there is NO continuous numbering
% of equations and figures, those variables are reset. It is typically
% used only by the other macros, not directly by the user.
%
\def\bumppchapno{\ifnum \chapno>-1 \global \advance \chapno by 1
                  \else \global \advance \chapno by -1 \setletter\chapno \fi
                  \ifcontinuousnumbering \else \global\eqlabelno=0 \fi
                  \ifcontinuousfigurenumbering \else \global\figureno=0 \fi}

```

```
% This is a very awkward way to turn a number into a letter, but some
% difficulty with simpler methods occurs in the \write routines.
%
\def\setletter#1{\ifcase-#1 {} \or{} \or\global\chapletter={A}
\or\global\chapletter={B} \or\global\chapletter={C} \or\global\chapletter={D}
\or\global\chapletter={E} \or\global\chapletter={F} \or\global\chapletter={G}
\or\global\chapletter={H} \or\global\chapletter={I} \or\global\chapletter={J}
\or\global\chapletter={K} \or\global\chapletter={L} \or\global\chapletter={M}
\or\global\chapletter={N} \or\global\chapletter={O} \or\global\chapletter={P}
\or\global\chapletter={Q} \or\global\chapletter={R} \or\global\chapletter={S}
\or\global\chapletter={T} \or\global\chapletter={U} \or\global\chapletter={V}
\or\global\chapletter={W} \or\global\chapletter={X} \or\global\chapletter={Y}
\or\global\chapletter={Z}\fi}

% And a non-global version of the above:
%
\def\tempsetletter#1{\ifcase-#1 {} \or{} \or\chapletter={A} \or\chapletter={B}
\or\chapletter={C} \or\chapletter={D} \or\chapletter={E}
\or\chapletter={F} \or\chapletter={G} \or\chapletter={H}
\or\chapletter={I} \or\chapletter={J} \or\chapletter={K}
\or\chapletter={L} \or\chapletter={M} \or\chapletter={N}
\or\chapletter={O} \or\chapletter={P} \or\chapletter={Q}
\or\chapletter={R} \or\chapletter={S} \or\chapletter={T}
\or\chapletter={U} \or\chapletter={V} \or\chapletter={W}
\or\chapletter={X} \or\chapletter={Y} \or\chapletter={Z}\fi}

% A utility: it produces a number or a letter, depending on
% the sign of the argument. Used by other macros for appendices.
% (It is like \chapfolio, but need not refer to the current chapter.)
%
\def\chapshow#1{\ifnum #1>0 \relax #1%
\else {\tempsetletter{\number#1}\chapno=#1 \chapfolio}\fi}

% In proofmode, it is useful to put today's date on each output page.
%
\def\today{\number\day\space \ifcase\month\or Jan\or Feb\or
Mar\or Apr\or May\or Jun\or Jul\or Aug\or Sep\or
Oct\or Nov\or Dec\fi, \number\year}

% The initialization procedure. Output registers 1 and 2 are used.
%
\def\initialeqmacro{\ifproofmode
\headline{\tenrm \today\hfill \jobname\ --- draft\hfill\folio}
\hoffset=-1cm \immediate\openout2=alldcrossreferfile \fi
\iffowardreference \input labelfile
\ifproofmode \immediate\openout1=labelfile \fi \fi}

\def\initialeqmacros{\initialeqmacro} % either spelling
```

```
% Various ways to place a number on a chapter (or section):
%
\def\chapnum{\bumpchapno \chapfolio}

\def\chaplabel#1{\bumpchapno \ifproofmode \iffowardreference
  \immediate\write1{\noexpand\expandafter\noexpand\def
    \noexpand\csname CHAPLABEL#1\endcsname{\the\chapno}\fi\fi
  \global\expandafter\edef\csname CHAPLABEL#1\endcsname
  {\the\chapno}\ifproofmode\llap{\hbox{\marginstyle #1 }}\fi\chapfolio}

\def\chapref#1{\ifundefined{CHAPLABEL#1}??\ifproofmode\iffowardreference
  \else \write16{ ***Undefined Chapter Reference #1*** }\fi
  \else \write16{ ***Undefined Chapter Reference #1*** }\fi
  \else \edef\LABxx{\getlabel{CHAPLABEL#1}}\chapshow\LABxx\fi
  \ifproofmode\write2{Chapter #1}\fi}

% Various ways to place a number on an equation.
%
\def\eqnum{\global\advance\eqlabelno by 1
  \eqno(\ifchapternumbers\chapfolio.\fi\the\eqlabelno)}

\def\eqlabel#1{\global\advance\eqlabelno by 1 \ifproofmode\iffowardreference
  \immediate\write1{\noexpand\expandafter\noexpand\def
    \noexpand\csname EQLABEL#1\endcsname{\the\chapno.\the\eqlabelno?}\fi\fi
  \global\expandafter\edef\csname EQLABEL#1\endcsname
  {\the\chapno.\the\eqlabelno?} \eqno(\ifchapternumbers\chapfolio.\fi
  \the\eqlabelno)\ifproofmode\rlap{\hbox{\marginstyle #1}}\fi}

\def\eqalignnum{\global\advance\eqlabelno by 1
  &(\ifchapternumbers\chapfolio.\fi\the\eqlabelno)}

\def\eqalignlabel#1{\global\advance\eqlabelno by1 \ifproofmode
  \iffowardreference\immediate\write1{\noexpand\expandafter\noexpand\def
    \noexpand\csname EQLABEL#1\endcsname
    {\the\chapno.\the\eqlabelno?}\fi\fi
  \global\expandafter\edef\csname EQLABEL#1\endcsname
  {\the\chapno.\the\eqlabelno?}&(\ifchapternumbers\chapfolio.\fi
  \the\eqlabelno)\ifproofmode\rlap{\hbox{\marginstyle #1}}\fi}

\def\eqref#1{(\ifundefined{EQLABEL#1}***\ifproofmode\iffowardreference)%
  \else \write16{ ***Undefined Equation Reference #1*** }\fi
  \else \write16{ ***Undefined Equation Reference #1*** }\fi
  \else \edef\LABxx{\getlabel{EQLABEL#1}}%
  \def\LAByy{\expandafter\stripchap\LABxx}\ifchapternumbers
  \chapshow{\LAByy}.\expandafter\stripeq\LABxx
  \else\ifnum \number\LAByy=\chapno \relax\expandafter\stripeq\LABxx
  \else\chapshow{\LAByy}.\expandafter\stripeq\LABxx\fi\fi\fi
  \ifproofmode\write2{Equation #1}\fi}
```

```
% These are analogous to \eqnum etc. They will automatically
% generate figure numbers and in the second case accept your
% label for later reference.
%
\def\fignum{\global\advance\figureno by 1 \relax
    \iffigurchapternumbers\chapfolio.\fi\the\figureno}

\def\figlabel#1{\global\advance\figureno by 1\relax
    \ifproofmode\iffowardreference
    \immediate\write1{\noexpand\expandafter\noexpand\def
        \noexpand\csnameFIGLABEL#1\endcsname{\the\chapno.\the\figureno?}}\fi\fi
    \global\expandafter\edef\csname FIGLABEL#1\endcsname
    {\the\chapno.\the\figureno?}\iffigurchapternumbers\chapfolio.\fi
    \ifproofmode\llap{\hbox{\marginstyle #1\ }}\relax\fi\the\figureno}

\def\figref#1{\ifundefined{FIGLABEL#1}!!!!\ifproofmode\iffowardreference
    \else \write16{ ***Undefined Figure Reference #1*** }\fi
    \else \write16{ ***Undefined Figure Reference #1*** }\fi
    \else \edef\LABxx{\getlabel{FIGLABEL#1}}%
    \def\LAByy{\expandafter\stripchap\LABxx}%
    \iffigurchapternumbers\chapshow{\LAByy}.\expandafter\stripeq\LABxx
    \else\ifnum\number\LAByy=\chapno \relax\expandafter\stripeq\LABxx
    \else\chapshow{\LAByy}.\expandafter\stripeq\LABxx\fi\fi
    \ifproofmode\write2{Figure #1}\fi\fi}

\def\eq{Eq.~}
\def\eqs{Eqs.~}
\def\fig{Figure~}
\def\figs{Figures~}

% TYPICAL macros to place headers on chapters or sections:
% Form to use is: ...head{label}{Title}
%
\def\centerhead#1#2{\vskip10pt\centerline{\chaplabel{#1}. #2}\vskip10pt}
\def\lefthead#1#2{\bf \noindent \chaplabel{#1}. #2\hfil\break}
\def\bighead#1#2{\vskip10pt%
    \centerline{\eqtwelverm \chaplabel{#1}. #2}\vskip10pt}

% Utilities for use by other macros
%
\def\getlabel#1{\csname#1\endcsname}
\def\ifundefined#1{\expandafter\ifx\csname#1\endcsname\relax}
\def\stripchap#1.#2?{#1}
\def\stripeq#1.#2?{#2}
```

## APE—A Set of TeX Macros To Format Ada Programs

Sriram Sankar\*

### 1 Introduction

This report describes a set of macros designed for the purpose of formatting Ada [Ada83] programs in TeX<sup>1</sup>. These macros were implemented by the author in early 1987, and they have been refined a few times since. They have also been extended for the language extensions of Ada developed by the *Program Analysis and Verification Group* at Stanford University. Some of these language extensions are Anna [LvHKO84] and TSL [HL85]. These macros exist as one collection in a file which is included into the document using the \input command. This file is available on request from the author.

The design of the macros was motivated by the `programexample` environment in Scribe. It is for this reason that they have been named *APE*, which is an acronym for “Ada Program-Example”. The macros are of two kinds—global macros and local macros. The global macros are visible throughout the scope in which the \input command has been inserted. The most important global macros are \apebegin, \apeend and \ap. \apebegin ... \apeend defines an environment in which a complete Ada program can be formatted, while \ap is a macro with one argument which is used for insertion of program text within ordinary text. The local macros are available for use only within environments defined by \apebegin ... \apeend and within the program text of arguments to \ap.

### 2 Formatting Ada Programs using APE

The example below shows a means of formatting a `procedure` EXCH. The following text:

```
\apebegin
\Procedure EXCH(X,Y : \In \Out INTEGER) \Is
  T : INTEGER;
  \cm{2.5in}{T is a temporary variable.}
\Begin
  T := X; X := Y; Y := T;
\End EXCH;
\apeend
```

\*Department of Computer Science, Stanford University, Stanford, California 94305.

EMail: sankar@score.stanford.edu.  
Phone: (415)723-4962.

<sup>1</sup>These macros can easily be rewritten for other languages.

produces the following output:

```
procedure EXCH(X,Y : in out INTEGER) is
  T : INTEGER;
  -- T is a temporary variable.
begin
  T := X; X := Y; Y := T;
end EXCH;
```

The text that produced the very first line in this section is shown below:

The example below shows a means of formatting a \ap{\Procedure EXCH}.

The following text:

As is obvious from the above example, keywords are entered as macros whose names are the same as the keywords (with the first letter converted to upper-case). Apart from \apebegin, \apeend and \ap, the keyword macros are the only global macros. Also, comments are inserted using the macro \cm which takes two arguments—the first is the maximum width allowed for the comment, and the second is the actual comment itself. If the comment does not fit in one line, it is split into lines each of the specified width. Each of the lines is prefixed by the Ada comment symbol — in the output<sup>2</sup>. If only the comment symbol is desired (with no comment following it), the macro \- can be used. Again, as is obvious from the above example, all spaces and carriage-returns are significant everywhere except within the arguments of the macro \cm.

The very first thing that one would like to do is to choose their own favorite fonts for the various Ada constructs. To do this, the following macros should be redefined after the \input command:

\apekeywordfont	\apkeywordfont
\apecommentfont	\apcommentfont
\apebodyfont	\apbodyfont

Each of the above macros defines a font. Those that are of the form \ape... define fonts for the \apebegin ... \apeend environment, while the others define the fonts for the \ap macro. In addition, the following macros can be redefined to specify the horizontal space that each space character inserts; the vertical space that each carriage-return inserts; and the margin at the beginning of each line:

\apehspace	\apevspace
\aphspace	\apelmargin

<sup>2</sup>For a similar use of this feature see [Des84].

The range delimiter (..) in Ada is obtained using the macro \.. The meanings of \_ (underscore) and \\_ have been interchanged. Hence underscores in Ada identifiers can be entered without a preceding backslash. However a backslash is required when the underscore is used for subscripting. Most other characters and macros have their usual plain TeX meaning. For example, math mode has to be entered for subscripting and superscripting, and to obtain characters like & and # one has to enter \& and \# respectively. There is one exception — math mode need not be entered to get a small space; one can enter \\_, directly.

Page-breaks are by default disabled within the \apebegin ... \apeend environment. However, there are macros that can enable or disable page-breaks within this environment. The macro \bhinge enables page-breaks, while the macro \ehinge disables page-breaks. Both \bhinge and \ehinge have global effect — even if they occur inside a group, they continue to have effect outside the group. \bhinge and \ehinge can be used within comment text (the second parameter to \cm) also. When used within comments, these macros also insert a space at their location. It is sometimes useful to specify the particular lines at which page-breaks should be enabled. For this there is a macro — \\\\. \\ has to be entered at the end of a line just before the carriage-return (<CR>). Actually, \\<CR> is equivalent to \bhinge<CR>\ehinge. Note that enabling page-breaks does not mean that TeX will actually move over to a new page at these locations; it only means that TeX can move over to a new page if its page-breaking algorithms decide that this is necessary. Commands that force page-breaks (like \eject) will not work within the \apebegin ... \apeend environment.

There is also a tabbing facility that can be used within the environment \apebegin ... \apeend. This is quite similar to the tabbing facility of TeX. For this, there are three more commands, the most important of which is the character &. This command shifts to the next tab position. If the current text has overshot this position, then it will back up to the tab position. However, if there is no tab position to skip to, then a new tab position is created at the current distance from the left margin. A carriage-return moves back to before the first tab. The command \kill at the end of a line inhibits output of the current line, but any new tab positions created in this line remain set. The command \actabs clears all the current tab settings. An implicit \actabs is performed by \apebegin. Tabbing commands cannot be used within comments

created using \cm. Any scope that begins within the \apebegin ... \apeend environment must be ended before the end of the line or the next &, whichever is earlier.

Just as in the case of TeX, these tabbing commands format text into a box of appropriate width and inserts \hss at the end of the text. Hence it is possible to achieve interesting results like centering text between two tab positions (by inserting glue more infinite than \hss on both sides of the text).

To conclude, three more examples are shown below. The first example demonstrates multi-line comments; the second example shows the use of tabbing and different fonts; and the third example shows how centering about a column can be achieved.

### Example 1:

The following input:

```
\apebegin
\Procedure CLOSE(FILE: \In \Out FILE_TYPE);
\cm{2.5in}{Severs the association
between the given file and its
associated external file. The
given file is left closed. The
exception {\apebodyfont STATUS_ERROR}
is raised if the given file is
not open.}
\apeend
```

produces:

```
procedure CLOSE(FILE: in out FILE_TYPE);
-- Severs the association between the given
-- file and its associated external file. The
-- given file is left closed. The exception
-- STATUS_ERROR is raised if the given file
-- is not open.
```

### Example 2:

The following input:

```
{\def\apekeywordfont{\normalsize\bf}
\def\apebodyfont{\normalsize\it}
\def\apehspace{0.5em}
\apebegin
\If n < r &\Then n := n + 1;
&\Else &\Begin print_totals; n := 0;
&&\End;
\End \If;
\apeend}
```

produces:

```

if n < r then n := n + 1;
else begin print_totals; n := 0;
end;
end if;

```

**Example 3:**

The following input:

```

\apebegin
\Type DAY \Is (&WEDNESDAY&,&\kill
\Type DAY \Is (&\hfill{SUNDAY}\hfill&,
&\hfill{MONDAY}\hfill&,
&\hfill{TUESDAY}\hfill&,
&\hfill{WEDNESDAY}\hfill&,
&\hfill{THURSDAY}\hfill&,
&\hfill{FRIDAY}\hfill&,
&\hfill{SATURDAY}\hfill&&);
\apeend

```

produces:

```

type DAY is ( SUNDAY ,
MONDAY ,
TUESDAY ,
WEDNESDAY,
THURSDAY ,
FRIDAY ,
SATURDAY );

```

**3 Conclusions**

The APE macros have been used extensively by the author and other members of the *Program Analysis and Verification Group* at Stanford. It has been used to format examples in two books and in many papers. No problems with its use have been encountered so far. The author invites comments and suggestions for the improvement of this set of macros. The complete listing of the macros (for Ada without any extensions) is provided in the Appendix with detailed comments explaining the various aspects of the macro.

**References**

- [Ada83] The Ada Programming Language Reference Manual. US Department of Defense, US Government Printing Office, February 1983. ANSI/MIL-STD-1815A-1983.
- [Des84] Jacques Désarménien. How to run TeX in a French environment: hyphenation, fonts, typography. *TUGboat*, 5(2):91–102, 1984.

[HL85] David P. Helmbold and David C. Luckham. TSL: task sequencing language. In *Ada in Use: Proceedings of the Ada International Conference*, pages 255–274, Cambridge University Press, May 1985.

[LvHKO84] David C. Luckham, Friedrich W. von Henke, Bernd Krieg-Brückner, and Olaf Owe. *Anna—A Language for Annotating Ada Programs*. Technical Report 84-261, Computer Systems Laboratory, Stanford University, July 1984. (Program Analysis and Verification Group Report 24.)

**Appendix A The APE Macros**

Editor's note: These macros have been reformatted for presentation in two columns. Problems arising in macros re-keyed from the text below should first be referred to the TUG office. The macros (in their original form) are available for anonymous FTP at `Score.Stanford.edu` in the directory `<TEX.TUGBOAT>`.

```

% Copyright 1988 by Sriram Sankar.
%
% GLOBAL DEFINITIONS:
%
% The fonts for the keywords.
\def\apekeywordfont{\large\bf}
\def\apkeywordfont{\normalsize\bf}

% The fonts for comments.
\def\apecommentfont{\normalsize\sffamily}
\def\apcommentfont{\normalsize\sffamily}

% The default fonts for everything else.
\def\apebodyfont{\small\rm}
\def\apbodyfont{\small\rm}

% The space generated by the <space> character.
\def\apehspace{0.65em}
\def\aphspace{0.5em}

% The space inserted between each line.
\def\apevspace{0pt}

% The space left at the beginning of each line.
\def\apelmargin{0pt}

% The above ten macro definitions can be
% redefined within the document as many
% times as desired. The default settings
% above are in terms of LaTeX macros
% (e.g. \large, \small, etc.). Note: These
% are the only LaTeX dependencies in this
% file.

```

```

%
\catcode`\!=11
\catcode`\^=9
% Tabs will now be ignored, so they can be
% used to format the macros below.
%
% The following macros are for formatting
% the Ada keywords. Note that they accept
% the keyword as an argument in LOWER-CASE.
% To format keywords in other ways
% (e.g. upper-case), one could either change
% the case of these keyword arguments below,
% or define \apekeywordfont in such a way
% that it changes the case of its argument
% appropriately.

\def\!keyword{\apekeywordfont}

\def\Abort{\{!\!keyword{abort}}}
\def\Abs{\{!\!keyword{abs}}}
\def\Accept{\{!\!keyword{accept}}}
\def\Access{\{!\!keyword{access}}}
\def\All{\{!\!keyword{all}}}
\def\And{\{!\!keyword{and}}}
\def\Array{\{!\!keyword{array}}}
\def\At{\{!\!keyword{at}}}
\def\Begin{\{!\!keyword{begin}}}
\def\Body{\{!\!keyword{body}}}
\def\Case{\{!\!keyword{case}}}
\def\Constant{\{!\!keyword{constant}}}
\def\Declare{\{!\!keyword{declare}}}
\def\Delay{\{!\!keyword{delay}}}
\def\Delta{\{!\!keyword{delta}}}
\def\Digits{\{!\!keyword{digits}}}
\def\Do{\{!\!keyword{do}}}
\def\Else{\{!\!keyword{else}}}
\def\Elsif{\{!\!keyword{elsif}}}
\def\End{\{!\!keyword{end}}}
\def\Entry{\{!\!keyword{entry}}}
\def\Exception{\{!\!keyword{exception}}}
\def\Exit{\{!\!keyword{exit}}}
\def\For{\{!\!keyword{for}}}
\def\Function{\{!\!keyword{function}}}
\def\Generic{\{!\!keyword{generic}}}
\def\Goto{\{!\!keyword{goto}}}
\def\If{\{!\!keyword{if}}}
\def\In{\{!\!keyword{in}}}
\def\Is{\{!\!keyword{is}}}
\def\Limited{\{!\!keyword{limited}}}
\def\Loop{\{!\!keyword{loop}}}
\def\Mod{\{!\!keyword{mod}}}
\def\New{\{!\!keyword{new}}}
\def\Not{\{!\!keyword{not}}}
\def\Null{\{!\!keyword{null}}}
\def\Of{\{!\!keyword{of}}}
\def\Or{\{!\!keyword{or}}}
\def\Others{\{!\!keyword{others}}}
\def\Out{\{!\!keyword{out}}}

```

```

\def\Package{\{!\!keyword{package}}}
\def\Pragma{\{!\!keyword{pragma}}}
\def\Private{\{!\!keyword{private}}}
\def\Procedure{\{!\!keyword{procedure}}}
\def\Raise{\{!\!keyword{raise}}}
\def\Range{\{!\!keyword{range}}}
\def\Record{\{!\!keyword{record}}}
\def\Rem{\{!\!keyword{rem}}}
\def\Renames{\{!\!keyword{renames}}}
\def\Return{\{!\!keyword{return}}}
\def\Reverse{\{!\!keyword{reverse}}}
\def>Select{\{!\!keyword{select}}}
\def\Separate{\{!\!keyword{separate}}}
\def\Subtype{\{!\!keyword{subtype}}}
\def\Task{\{!\!keyword{task}}}
\def\Terminate{\{!\!keyword{terminate}}}
\def\Then{\{!\!keyword{then}}}
\def\Type{\{!\!keyword{type}}}
\def\Use{\{!\!keyword{use}}}
\def\When{\{!\!keyword{when}}}
\def\While{\{!\!keyword{while}}}
\def\With{\{!\!keyword{with}}}
\def\Xor{\{!\!keyword{xor}}}
%
% Some more global definitions follow:
%
\def\!lbr{\{$}
\def\!rbr{$}
\def\!str{\$*\$}
\def\!pls{\$+\$}
\def\!min{\$-\$}
\def\!col{\$:\$}
\def\!scl{\$;\$}
\def\!les{\$<\$}
\def\!gref{\$>\$}
\def\!bar{\$|\$}
\def\!equ{\$=\$}
%
\def\!sla{\{/sl/}}
\def\!dqt{\$/raise.2ex\hbox{\tt{"}}}
\def\!dots{\$.{\hskip1mm}.}
\def\!comment{%
  {\apecommentfont\rm--\thinspace--}}
\def\!hinge{\vfil\penalty5000\vfilneg}
\def\!space{\hskip\apehspace}
\def\!underscore{\underbar{\!space}}
%
% Now follow the global definitions for the
% tabbing commands. First a set of dimen,
% count and box registers are allocated.
% The number of dimen registers limit the
% total number of tabs permitted.
%
\newdimen\!apetabi
\newdimen\!apetabii
\newdimen\!apetabiii
\newdimen\!apetabiv
\newdimen\!apetabv
\newdimen\!apetabvi

```

```

\newdimen\!apetabvii
\newdimen\!apetabviii
\newdimen\!apetabix
\newdimen\!apetabx
\newdimen\!apetabxi
\newdimen\!apetabxii
\newdimen\!apetabxiii
\newdimen\!apetabxiv
\newdimen\!apetabxv
\newdimen\!apetabxvi
\newdimen\!apetabxvii
\newdimen\!apetabxviii
\newdimen\!apetabxix
\newdimen\!apetabxx
\newcount\!apetotaltabs
\newcount\!apetab
\newbox\!apetabbox
\newbox\!apelinebox
%
% Dimen register i stores the distance from
% the tab stop (i-1) to the tab stop i.
% \!apetotaltabs maintains the total number
% of tabs currently set. \!apetab maintains
% the tab stop that will be reached if
% another '&' is encountered. The box
% registers are explained below.
%
% Between the beginning of lines, tab stops
% and the end of lines, text is formatted
% into the box \!apetabbox and padded on
% the right with the glue \hss. The macro
% \!apebbox is what needs to be done at the
% beginning of each of these boxes.
%
\def\!apebbox{%
  \ifnum\!apetab>\!apetotaltabs
    \setbox\!apetabbox=\hbox
  \else
    \setbox\!apetabbox=\hbox to
      \csname !apetab\romannumeral\!apetab
    \endcsname
  \fi
  \bgroup
}
%
% The boxes mentioned above (that are stored
% into \!apetabbox) are put together into
% another \hbox called \!apelinebox. This
% box is output once a full line has been
% read from input. The macro \!apepline
% does what is needed at the beginning of
% each input line, while the macro \!apeeline
% does what is needed at the end of each
% input line.
%
\def\!apebline{%
  \noindent
  \global\!apetab=1
  \setbox\!apelinebox=
}
%
\hbox\bgroup
  \hskip\apelmargin\!apebbox
}
%
\def\!apeeline{%
  \hss\egroup
  \box\!apetabbox
}
\egroup
\box\!apelinebox
}
%
% The macro \!apekill is similar to
% \!apeeline, except that it does not output
% the contents of \!apelinebox. In addition,
% this macro also performs a \!apepline, thus
% getting ready for the next line of input.
%
\def\!apekill{%
  \hss\egroup
  \box\!apetabbox
}
\egroup
\!apepline
}
%
% The macro \!apetabskip is expanded when an
% '&' is encountered in the input file. It
% ends the current \!apetabbox, and if a new
% tab stop needs to be set, it increments the
% counters appropriately and sets the
% appropriate dimen register to the width of
% \!apetabbox. Finally, \!apebbox is invoked
% to start off a new \!apetabbox.
%
\def\!apetabskip{%
  \hss\egroup
  \ifnum\!apetab>\!apetotaltabs
    \global\advance\!apetotaltabs by 1
    \global\csname
      !apetab\romannumeral\!apetotaltabs
    \endcsname=\wd\!apetabbox
  \fi
  \box\!apetabbox
  \global\advance\!apetab by 1
  \!apebbox
}
%
\def\!apecr{%
  \strut\par
  \!break
  \vskip\apevspace
}
%
\def\!par{\!apeeline\!apecr\!apebline}
%
% The following global definitions are for
% formatting Ada comments. The macros \!cma
% and \!cmb are used in the definition of the
% \cm macro later. The comment text is split
% at all \bhinge's and \ehinge's. Each

```

```
% portion of the comment is then put into a
% \vbox separately and output one after the
% other. Actually all but the last line of
% each \vbox is output. The last line is
% included at the beginning of the next \vbox.
% At the end of it all, there will be one more
% line to output. \!cmb explicitly outputs
% this line, and moves back to the beginning
% of this line. The box registers below are
% used to store the boxes created during this
% process. Dimen register \!cmtpos contains
% the distance of the comment from the left
% margin while \!cmtwd contains the width of
% the comment.
%
\newif\ifnonvoid
\newbox\!cmti
\newbox\!cmtii
\newbox\!cmtiii
\newbox\!cmtiv
\newbox\!cmtv
\newbox\!cmtvi
\newdimen\!cmtpos
\newdimen\!cmtwd
%
% \!cmtout is the output routine. It assumes
% that the \hboxes to be output are enclosed
% within a \vbox in \!cmti. It consists
% mainly of two loops. The first loop
% reverses the order of the \hboxes in \!cmti
% and puts this into a \vbox in \!cmtv. The
% second loop pulls out the \hboxes from
% \!cmtv and outputs them.
%
\def\!cmtout{%
\setbox\!cmtii=
\vbox{
\unvbox\!cmti
\global\setbox\!cmtiii=\lastbox
\unskip
\unpenalty
}%
\nonvoidtrue
\ifvoid\!cmtiii\nonvoidfalse
\fi
\ifnonvoid
\setbox\!cmti=\box\!cmtii
\setbox\!cmtv=\vbox{\box\!cmtiii}%
\nonvoidtrue
\loop
\setbox\!cmtii=
\vbox{
\unvbox\!cmti
\global\setbox\!cmtiii=\lastbox
\unskip
\unpenalty
}%
\ifvoid\!cmtiii\nonvoidfalse
\fi
\ifnonvoid
\setbox\!cmti=\box\!cmtii
\!cmtout
\global\def\!break{\!hinge}%
\setbox\!cmti=
\vbox\bgroup
\hsize=\!cmtwd
\noindent\apecommentfont
{} \unhbox\!cmtvi{} %
}
```

```
\ifnonvoid
\setbox\!cmti=\box\!cmtii
\setbox\!cmtiv=
\vbox{\unvbox\!cmtv\box\!cmtiii}%
\setbox\!cmtv=\box\!cmtiv
\repeat
\nonvoidtrue
\loop
\setbox\!cmtii=
\vbox{
\unvbox\!cmtv
\global\setbox\!cmtiii=\lastbox
}%
\ifvoid\!cmtiii\nonvoidfalse
\fi
\ifnonvoid
\setbox\!cmti=\box\!cmtii
\noindent\hskip\!cmtpos
\!comment{} \space
\box\!cmtiii{} \!apecr
\repeat
\fi
}
%
% The following two macros \!cmtbhinge and
% \!cmtehinge end the current comment box
% being created. They then pull out the
% last line from this box and ship the
% rest of the box to \!cmtout. The value
% of \!break is then modified to ensure that
% the correct penalty value is inserted at
% the end of the next line. They then start
% off a new comment box after inserting the
% last line of the previous box and a space
% character at the beginning.
%
\def\!cmtbhinge{%
\egroup
\setbox\!cmtii=
\vbox{
\unvbox\!cmti
\global\setbox\!cmtiii=\lastbox
\unskip
\unpenalty
}%
\setbox\!cmtvi=
\hbox{%
\unhbox\!cmtiii
\unskip\unskip
\unpenalty
}%
\setbox\!cmti=\box\!cmtii
\!cmtout
\global\def\!break{\!hinge}%
\setbox\!cmti=
\vbox\bgroup
\hsize=\!cmtwd
\noindent\apecommentfont
{} \unhbox\!cmtvi{} %
```

```

}

%
\def\!cmtehinge{%
  \egroup
  \setbox\!cmtii=
  \vbox{
    \unvbox\!cmti
    \global\setbox\!cmtiii=\lastbox
    \unskip\unpenalty
  }%
}
\setbox\!cmtvi=
\hbox{%
  \unhbox\!cmtiii
  \unskip\unskip
  \unpenalty
}%
\setbox\!cmti=\box\!cmtii
\!cmtout
\global\def\!break{\nobreak}%
\setbox\!cmti=
\vbox\bgroup
  \hsize=\!cmtwd
  \noindent\apecommentfont
  {} \unhbox\!cmtvi{} %
}

%
% \!cma begins the comment by finishing off
% the boxes being created and then outputting
% them. It also measures the width of the
% box output to determine the value of
% \!cmtpos.
%
\def\!cma{%
  \egroup
  \setbox\!cmti=\hbox{\unhbox\!apetabbox}%
  \box\!cmti
  \egroup
  \!cmtpos=\wd\!apelinebox
  \hbox to Opt{\box\!apelinebox\hss}%
}
%
% \!cmb takes over from \cm. It starts off
% the first comment box and finishes
% off the last comment box. If there is
% only one comment box (no \bhinge or
% \ehinge in between), then it starts and
% finishes this one box. It then pulls
% out the last line of the last box and
% ships the rest of the box to \!cmtout.
% It then outputs the last comment line
% and moves to the beginning of the line
% in which the last comment line was output.
%
\def\!cmb#1#2{%
  \!cmtwd=#1%
  \setbox\!cmti=
  \vbox\bgroup
    \hsize=\!cmtwd
    \noindent
    \apecommentfont
    {} \#2\egroup
    \setbox\!cmtii=
    \vbox{
      \unvbox\!cmti
      \global\setbox\!cmtvi=\lastbox
      \unskip
      \unpenalty
    }%
    \setbox\!cmti=\box\!cmtii
    \!cmtout
    \noindent
    \hbox to Opt{%
      \hskip\!cmtpos
      \!comment{}\!space
      \box\!cmtvi
      \hss
    }%
  }
  \endgroup
  \!apepline
}

%
% The following global definition of \ap is
% used in the definition of the \ap macro
% later. As in the case of \!cmb and \cm,
% there is an \endgroup in \ap whose
% corresponding \begin{group} is in \ap. In
% this case, there is also a corresponding
% \apebegin in \ap.
%
\def\!ap#1{\apeend\endgroup{}}

%
% We cannot have the sequence "\let=\!equ"
% within the body of the ape macro since
% the "=" will not be scanned with the
% correct category code. Hence, the macro
% \!defequ is defined in an environment
% where "=" has the correct category code,
% and then this macro is used within the body
% of the ape macro. \!defequ is now defined
% as a global macro.
%
{
  \catcode`\==\active
  \gdef\!defequ{\let=\!equ}%
}
%
\catcode`\!=12
\catcode`\^`I=10
%
% END OF GLOBAL DEFINITIONS.
%
% NOW THE MACRO DEFINITIONS:
%
% The macro has to be defined in an
% environment with the category codes set
% correctly. This environment is first set
% up below, and then within this environment,
% the macros are defined.

```

```

%
% The macro definition environment
%
% Firstly, "outer" macros cannot occur
% within any macro expansions. The only
% such macro that occurs in the macros
% below is \+. Hence this is redefined to
% its usual value. The effect of this
% redefinition just changes the status of
% \+ to now be a non-"outer" macro.
%
\def\+{\tabalign}%
%
% Necessary changes to category codes are
% made right now since category codes
% are used while scanning and the macro
% is scanned at definition time, and not
% at macro expansion time.
%
\catcode`(`=\active%
\catcode`)=\active%
\catcode`*= \active%
\catcode`+= \active%
\catcode`-= \active%
\catcode`:= \active%
\catcode`;= \active%
\catcode`<= \active%
\catcode`>= \active%
\catcode`|= \active%
\catcode`/= \active%
\catcode`"= \active%
\catcode`'^M= \active%
\catcode`\\ = \active%
\catcode`\_ = \active%
\catcode`!=11%
\catcode`\&= \active%
%
\gdef\apebegin{%
\begingroup%
\global\def\!break{\nobreak}%
%
% First the category codes of the characters
% to be redefined are made active.
%
\catcode`(`=\active%
\catcode`)=\active%
\catcode`*= \active%
\catcode`+= \active%
\catcode`-= \active%
\catcode`:= \active%
\catcode`;= \active%
\catcode`<= \active%
\catcode`>= \active%
\catcode`|= \active%
\catcode`/=\active%
\catcode`"= \active%
\catcode`'^M= \active%
\catcode`\\ = \active%
\catcode`\_ = \active%
%
\catcode`\!=11%
\catcode`\&= \active%
%
% Now each of these characters is given
% its definition.
%
\let(=\!lbr%
\let)=\!rbr%
\let*= \!str%
\let+= \!pls%
\let-= \!min%
\let:= \!col%
\let;= \!sc1%
\let<= \!les%
\let>= \!gre%
\let!= \!bar%
\let/= \!sla%
\let"= \!dqt%
\let^^M= \!par%
\let = \!space%
\let_= \!underscore%
\catcode`^^A=8\def\_{``A}%
%
% The above two lines interchange the
% meanings of _ and \_. To do this, a new
% subscript character ^A is defined, and
% then \_ is defined as a macro to
% expand to ^A.
%
\let&= \!apetabskip%
%
% Now follows the remaining macro
% declarations needed to complete the \ape
% environment.
%
\def\.{\!\dots}%
\def\,{\!\thinspace}%
\def\-\{\!\comment}%
%
\def\bhinge{\global\def\!break{\!hinge}}%
\def\ehinge{\global\def\!break{\nobreak}}%
\def\^\^M{\bhinge\^\^M\ehinge}%
\def\actabs{%
\global\!apetotaltabs=0%
\global\!apetab=1{}% end \def of actabs
\actabs%
\def\kill\^\^M{\!\apekill}%
\def\!keyword{\apekeywordfont}%
%
% Following is the \cm macro definition.
% This macro eventually lets the macro
% \!cmb take over so that the text
% following is read in with the correct
% category codes. The redefinition of
% \+ below is for the same reason as
% before: to convert it from an outer
% to a non-outer macro.
%
\def\+{\tabalign}%

```

```

%
\def\cm{\!cma\begingroup%
%
% The category codes are temporarily
% changed back to their original values.
%
\catcode`\-=12%
\catcode`\\=12%
\catcode`*=12%
\catcode`+=12%
\catcode`\-=12%
\catcode`\:=12%
\catcode`\;=12%
\catcode`\<=12%
\catcode`\>=12%
\catcode`\|=12%
\catcode`\/=12%
\catcode`\"=12%
\catcode`\^M=5%
\catcode`\ =10%
\catcode`\!=12%
%
\def\bhinge{\!cmtbhinge}%
\def\ehinge{\!cmtehinge}%
%
\!cmb}%
%
% Now = is redefined. Had this been done
% earlier, then it would have affected
% the previous definitions, since many
% of them contain =.
%
\catcode`\==\active\!defequ%
%
\apebodyfont\!apepline%
%
}% end of definition of \apebegin
%
\gdef\apeend{\apeeline\endgroup}%
%
\gdef\ap{\begingroup%
\def\apekeywordfont{\apkeywordfont}%
\def\apecommentfont{\apcommentfont}%
\def\apebodyfont{\apbodyfont}%
\def\apehspace{\aphspace}%
\def\apelmargin{Opt}%
\apebegin\!ap%
}%
%
%
% The reason for letting \!ap take over
% from \ap at this point is to make TeX
% scan the argument with the correct
% category codes.
%
}%
%
% This ends the environment in which the
% macros are defined.
%

```



## Contents of Archive Server as of 16 January 1989

Michael DeCorte  
Clarkson University

Several changes to the LATEX Archive have been made since the last TUGboat. It has been split into several different subarchives: LATEX style files; TEX style files; AMS-TEX style files; BIBTEX style files; BIBTEX 0.98 style files; TEX programs. Also there are several new archives: AMS-TEX source; BIBTEX source; CM fonts source; LATEX source; TEX documentation; TEX inputs; TEX source; TEX tests; TUGboat files; TEXhax digests; TEXMaG digests; UKTEX digests.

As always, submissions are encouraged. If you do submit a file please include at the top of the file: your name; your email address; your real address; the date. Also please make certain that there are no lines in the file longer than 80 characters as some mailers will truncate them. Mail should be sent to

mrd@sun.soe.clarkson.edu  
archive-management@sun.soe.clarkson.edu

### For Internet users: How to ftp

An example session is shown at the bottom of the following page. Users should realize that ftp syntax varies from host to host. Your syntax may be different. The syntax presented here is that of Unix ftp. Comments are in parentheses. The exact example is for retrieving files from the LATEX Archive; the syntax is similar for the other archives, only the directories differ. The directory for each archive is given in its description.

### Non-Internet users: How to retrieve by mail

To retrieve files or help documentation, send mail to archive-server@sun.soe.clarkson.edu with the body of the mail message containing the command help or index or send. The send command must be followed by the name of the archive and then the files you want. Users who are not in the uucp maps database are strongly encourage to include a path command followed by a path from Clarkson to you in domain style format. If you don't include a path command, your mail may not get to you and will definitely be delayed as Michael will have to mail it by hand. You should realize that Clarkson does not have a uucp connection; therefore you must send it to an Internet or Bitnet

host that does have a uucp connection. For example, host!user@uunet.uu.net should send:

```
To: archive-server@sun.soe.clarkson.edu
Subject:
path host!user@uunet.uu.net
send latex-style README Index
send latex-style resume.sty
```

Traffic on the network servers and gateways has been very high recently, and in order to provide improved service, there have been some volunteers to maintain local "slave" repositories of the L<sup>A</sup>T<sub>E</sub>X style collection. There is usually a geographic or network restriction requested, since the idea is to cut down traffic, not add to it. The following areas will be covered by the volunteers listed.

- Bitnet users: Texas A&M maintains a list-server and file-server which is already handling (with TEX-L) much of the Bitnet distribution of T<sub>E</sub>Xhax. An inquiry via listserv will retrieve a list of all T<sub>E</sub>X-related files:  
tell listserv at tamvm1 get tex filelist
- UK users: Aston University maintains a T<sub>E</sub>X archive covering all aspects of T<sub>E</sub>X/L<sup>A</sup>T<sub>E</sub>X/METAFONT and ancillary software. UKT<sub>E</sub>X (like T<sub>E</sub>Xhax) digests are distributed from Aston. For users with Colour book software, FTP access is available; for all users, mail access is available. Send enquiries in the first instance to info-tex@uk.ac aston (via Internet use pabbott@nss.cs.ucl.ac.uk).

Additional volunteers should contact Michael.

#### Distribution for IBM PC and clone users

There are two sources.

- David W. Hopper  
446 Main Street  
Toronto, Ontario  
Canada M4C 4Y2

has L<sup>A</sup>T<sub>E</sub>X style files only.

1. Either one 1.2 MB diskette or three 360 KB diskettes, blank and formatted.
2. Indication of the format required,
3. A self-addressed mailer, and
4. A \$5.00 donation per set of files, to cover postage and equipment wear & tear. (If you live outside North America, airmail delivery will probably require more postage. You should probably contact David for details.)
5. No phone calls or personal visits please.

■ Jon Radel  
P.O. Box 2276  
Reston, VA 22090

has L<sup>A</sup>T<sub>E</sub>X style files and utilities. For a list of other info send a SASE.

1. 360 KB diskettes, blank and formatted.
2. A stamped, self-addressed mailer, and
3. \$1.50 per disk. If you live outside North America, skip the stamps and send additional money or International Reply Coupons.

As a convenience for people who have more money than floppies, Jon will supply everything for \$6.00 per disk to U.S./Canada/Mexico addresses.

#### Contents of the *AMS-TEX* Source

This contains the T<sub>E</sub>X source needed to build *AMS-TEX*. It is a duplicate directory of *tex.amstex* on Score. Files are located in *pub/amstex* for ftp users. Mail users should request files from the *amstex* archive.

#### Contents of the *BIBTEX* Source

This contains the BIBTEX style files and the WEB files need to build BIBTEX. It is a duplicate directory of *tex.bibtex* on Score. Files are located in *pub/bibtex* for ftp users. Mail users should request files from the *bibtex* archive.

#### Sample FTP session for Internet users

```
% ftp sun.soe.clarkson.edu
...
user: anonymous
password: <any non-null string>
ftp> cd pub/latex-style
ftp> ls
...
ftp> get Index
...
ftp> quit
```

(a.k.a. 128.153.12.3)  
(general blurb)

(where the files are)  
(to see what is there)  
(lots of output)

(more blurb)

### Contents of the CM Fonts

This contains the METAFONT files needed to build the CM fonts. It is a duplicate directory of `tex.cm` on Score. Files are located in `pub/cm-fonts` for ftp users. Mail users should request files from the `cm-fonts` archive.

### Contents of the L<sup>A</sup>T<sub>E</sub>X Source

This contains the T<sub>E</sub>X files needed to build L<sup>A</sup>T<sub>E</sub>X. It is a duplicate directory of `tex.latex` on Score. Files are located in `pub/lamport` for ftp users. Mail users should request files from the `lamport` archive.

### Contents of the T<sub>E</sub>X Documentation

This contains documentation on T<sub>E</sub>X. It is a duplicate directory of `tex.doc` on Score. Files are located in `pub/tex-doc` for ftp users. Mail users should request files from the `tex-doc` archive.

### Contents of the T<sub>E</sub>X Inputs

This contains the T<sub>E</sub>X files needed to build plain T<sub>E</sub>X. It is a duplicate directory of `tex.inputs` on Score. Files are located in `pub/tex-inputs` for ftp users. Mail users should request files from the `tex-inputs` archive.

### Contents of the T<sub>E</sub>X Source

This contains the WEB files needed to build T<sub>E</sub>X. It is a duplicate directory of `tex.web` on Score. Files are located in `pub/tex-source` for ftp users. Mail users should request files from the `tex-source` archive.

### Contents of the T<sub>E</sub>X Tests

This contains the files needed to test T<sub>E</sub>X using the `triptest`. It is a duplicate directory of `tex.tests` on Score. Files are located in `pub/tex-tests` for ftp users. Mail users should request files from the `tex-tests` archive.

### Contents of the TUGboat Files

This contains files related to TUGboat. It is a duplicate directory of `tex.tugboat` on Score. Files are located in `pub/tugboat` for ftp users. Mail users should request files from the `tugboat` archive.

### Contents of the T<sub>E</sub>Xhax Digests

This contains all of the back issues of T<sub>E</sub>Xhax. Files are located in `pub/texhax` for ftp users. Mail users should request files from the `texhax` archive.

### Contents of the T<sub>E</sub>XM<sub>A</sub>G Digests

This contains all of the back issues of T<sub>E</sub>XM<sub>A</sub>G. Files are located in `pub/texmag` for ftp users. Mail users should request files from the `texmag` archive.

### Contents of the UKT<sub>E</sub>X Digests

This contains all of the back issues of UKT<sub>E</sub>X. Files are located in `pub/uktex` for ftp users. Mail users should request files from the `uktex` archive.

### Contents of the L<sup>A</sup>T<sub>E</sub>X Style Collection

This contains files that are specific to L<sup>A</sup>T<sub>E</sub>X. Most of these are style files but some of them are programs. Some of the files support BIBT<sub>E</sub>X style files that are in the BIBT<sub>E</sub>X Collection or the BIBT<sub>E</sub>X 0.98 Collection. Files are located in `pub/latex-style` for ftp users. Mail users should request files from the `latex-style` archive.

<code>a4.sty</code>	Set page size to A4
<code>a4wide.sty</code>	Set page size to A4 with narrow margins (needs <code>a4.sty</code> )
<code>a5.sty</code>	Sets A5 page size (use only with 10pt)
<code>a5comb.sty</code>	Sets A4 page style but for spirally-bound documents (bigger inner margins)(needs <code>a5.sty</code> )
<code>aaai-doc.tex</code>	Style file for <i>AAAI conference 1988</i>
<code>aaai.sty</code>	(BIBT <sub>E</sub> X style in <code>bibtex-style</code> and <code>bibtex-style-0.98</code> )
<code>aip.sty</code>	For American Institute of Physics journals
<code>agugrl-sample.tex</code>	
<code>agugrl.sty</code>	<i>AGU Geophysical Research Letters</i>
<code>agujgr-sample.tex</code>	
<code>agujgr.sty</code>	<i>AGU Journal of Geophysical Research</i>
<code>album.shar</code>	For printing cassette labels
<code>alleqno.sty</code>	Makes all displayed equations numbered by default
<code>alltt.sty</code>	Like <code>verbatim</code> , but permits other commands inside
<code>amssymbols.sty</code>	Load the AMS fonts
<code>apalike.sty</code>	<i>American Psychological Association</i> (BIBT <sub>E</sub> X style in <code>bibtex-style</code> )
<code>biihead.sty</code>	Underlined heading
<code>boxedminipage.sty</code>	Puts a box round a minipage
<code>bsf.sty</code>	Provide access to bold sans serif fonts in L <sup>A</sup> T <sub>E</sub> X
<code>captcont.sty</code>	Needed by <code>lcustom.tex</code> . Creates a caption without updating any counters

<b>changebar.sty</b>	Changebars for LATEX	<b>ist21.sty</b>	IST21 document style option for cover page
<b>chapterbib.sty</b>	Allow LATEX and BIBTEX to produce separate bibliographies for each chapter	<b>jbs.sty</b>	For <i>The Journal of Business Strategies</i>
<b>cyrillic.sty</b>	Load Cyrillic font	<b>layout.readme</b>	
<b>deprocldc.tex</b>		<b>layout.tex</b>	Prints nice diagram showing page parameters of LATEX
<b>deproc.sty</b>	<i>DEC Users Proceedings</i>	<b>lcustom.tex</b>	Useful macros and definitions for LATEX
<b>draft.sty</b>	Draft option for documents for debugging	<b>local-suppl.tex</b>	
<b>drafthead.sty</b>	Prints DRAFT in heading		Supplement to local guide that describes lcustom.tex, sfwmac.sty, tgrind.sty, trademarks.sty xxxcustom.tex, xxxslides.sty
<b>drop.sty</b>	For making large dropped initials for starting paragraphs	<b>lfonts_ams.readme</b>	
<b>doublespace.sty</b>	Double spacing in text	<b>lfonts_ams.tex</b>	Use AMS symbols in LATEX
<b>eepic10.shar</b>	A picture environment that uses tpic specials	<b>manual.readme</b>	
<b>env.sty</b>	For printing on envelopes	<b>manual.sty</b>	Like "book" but for manuals. You should read the documentation for "book" to learn how to use this style file
<b>epic1.shar</b>	Shar archive of an extended picture	<b>man10.sty</b>	
<b>epic2.shar</b>	environment	<b>man11.sty</b>	
<b>equations.sty</b>	Macros to aid in constructing displayed equations in LATEX	<b>man12.sty</b>	
<b>espo.sty</b>	For Esperanto	<b>memo.sty</b>	Memo and "memo for the record" style option
<b>figplace.tex</b>	TEX macros to handle floating insertion	<b>mfr.sty</b>	
<b>fixup.sty</b>	Fix up plain's \bigl, etc. to track LATEX size changes	<b>merge.sty</b>	Form letter option to LATEX letter style
<b>flowchart.sty</b>	For writing flow charts	<b>mitthesis-sample.tex</b>	
<b>fnpara.sty</b>	Sets footnotes as paragraphs	<b>mitthesis.sty</b>	Massachusetts Institute of Technology thesis format
<b>format.sty</b>	Print FP numbers in fixed format	<b>multibox.sty</b>	Provides multiple boxes in pictures
<b>frontiers.sty</b>	For <i>Frontiers '88 Symposium</i> and other IEEE conferences	<b>natsci.sty</b>	Natural sciences (BIBTEX file in bibtex-style-0.98)
<b>fullpage.sty</b>	Gives one inch margins	<b>nl.sty</b>	For Dutch
<b>german.sty</b>	For German	<b>nofm.sty</b>	For "n of m" style pagination
<b>greek1.shar</b>	Shar file to help Greek users of	<b>nopagenumbers.sty</b>	Remove page numbers
<b>greek2.shar</b>	LATEX	<b>pslatex.shar</b>	Use printer resident PostScript fonts. Requires dvi2ps that understands PostScript fonts.
<b>headerfooter.sty</b>	Adds the capability of underlining the heading, ala the LATEX manual	<b>remark.sty</b>	Like newtheorem but no \it
<b>icassp.sty</b>	For the <i>ICASSP '89 Conference</i> and possibly other IEEE conferences	<b>resume-sample.tex</b>	
<b>insertplot.readme</b>		<b>resume.sty</b>	A format for doing resumes
<b>insertplot.sty</b>	Macros for inserting PostScript in files printed with Arbortext's DVIPS.	<b>romanneg.sty</b>	Roman-numbered pages get negative page numbers (useful when selecting only part of a document to be printed)
<b>iso-doc.tex</b>	For use by those writing ISO standards	<b>rotate.sty</b>	Rotation of TEX output with Textures (and maybe other PostScript drivers)
<b>iso.sty</b>		<b>sc21-wg1.sty</b>	ISO/TC97/SC21 document style
<b>iso9.sty</b>		<b>sc21.sty</b>	option for cover page
		<b>schedule.sty</b>	Style for generating schedule sheets
		<b>screen.sty</b>	Helps create a document suitable for screen previewing

**semitic.sty** Used to set Semitic languages  
**sfrmac.sty** Useful macros for Unix documentation  
**showlabels.sty** Shows labels and references to them  
**slem.sty** Change \sl to \em  
**siam10.sty** Document style for SIAM  
**siam11.sty** (Society for Industrial and Applied  
**siam12.sty** Mathematics). (BIBTEX style in  
**siam.sty** *bibtex-style*)  
**siam.tex**  
**siam.bib**  
**spacecites.sty** Put spacing between citations  
**subeqn.sty** Allows related equations to be numbered with the same number but further qualified by a,b,c etc.  
**subeqnarray.sty** Allows related eqnarrays to be numbered with the same number but further qualified by a,b,c etc.  
**subfigure.sty** Allows related figures to be numbered with the same number but further qualified by a,b,c etc.  
**supertab.sty** Allows multipage tabulars  
**suthesis.sty** Stanford University thesis style  
**svma-doc.tex** Style for Springer-Verlag reports.  
**svma.sty** svma is for multiple authors; svsa  
**svsa.sty** is for single author  
**tables-doc.tex**  
**tables.sty** Ruled and unruled tables made easy  
**texnames.sty** Define a couple more LATEX names  
**tgrind.sty** Tgrind macros for LATEX instead of TEX  
**threepart.sty** Three part page headers  
**trademarks.sty** Definitions of common trademarks  
**twoup.sty** Change the page sizes so that two pages can fit onto one page with the help of dvdiv  
**ucthesis.readme**  
**ucthesis.sty** A document style for the  
**uct10.sty** University of California thesis with  
**uct11.sty** documentation and support files.  
**uct12.sty**  
**ukdate.sty** Changes the \today command to UK format  
**vdm-doc.tex** Vienna Development Method  
**vdm.sty**

**xxxcustom.tex**  
Supplementary macros for  
**xxx-tex**, for some **xxx**  
**xxsslides.sty**  
Supplementary macros for SLIDE, includes **slides.sty**

### Contents of the **TEX** Collection

This contains style files for plain **TEX**. Files are located in pub/**tex-style** for ftp users. Mail users should request files from the **tex-style** archive.  
**celluar.tex** Macros to allow for both vertical and horizontal spans within a ruled  
**cell1.tex** table and take steps to prevent  
**cell2.tex** “nubs” and “gaps” when rules are used  
**cell3.tex**  
**cell4.tex**  
**dayofweek.tex**

Macros to compute day of week and phases of the moon. Examples of how to use **TEX** arithmetic capabilities

**deprocdoc.tex**  
**deproc.tex** DEC Users Proceedings  
**epigram.tex** Print text either centered or in a displayed paragraph  
**fnpara.tex** Sets footnotes as paragraphs  
**hyphen-nederland.tex**  
A dutch hyphen.tex  
**texinfo.tex** **TEX** macros to handle Gnu texinfo files  
**select.tex** Selectively print pages in a **TEX** document

### Contents of the **AMS-TEX** Collection

This contains style files specific to **AMS-TEX** users. Files are located in pub/**amstex-style** for ftp users. Mail users should request files from the **amstex-style** archive.

**amstexsiam-sample.tex**  
**amstexsiam-doc.tex**  
**amstexsiam.sty** For SIAM (Society for Industrial and Applied Mathematics)  
**imappt-sample.tex**  
**imappt.sty** For reports and preprints on 8.5  
**imappt-doc.tex** by 11 inch paper

### Contents of the **BIBTEX** Collection

This contains files that are specific to version 0.99 of BIBTEX. Many of these files are to be used with files in the **LATEX** Collection. Files are located in pub/**bibtex-style** for ftp users. Mail users should request files from the **bibtex-style** archive.

<b>acm.bst</b>	Association for Computing Machinery	<b>rscsencode.shar</b>	Programs to encode and decode files with rscs
<b>aaai-named.bst</b>	AAAI conference	<b>uuencode.shar</b>	Programs to encode and decode files
<b>apalike.bst</b>	American Psychological Association (LATEX style file in latex-style)	<b>texindex.shar</b>	Style file and processor for index entries for VMS
<b>cpp.el</b>	A C preprocessor written for GnuEmacs for use in generating BIBTEX style files from a master file	<b>wsltex.shar</b>	Wordstar to LATEX filter, C and Pascal versions
<b>ieeetr.bst</b>	IEEE Transactions		
<b>makebst.sh</b>	A shell script to make bibtex style files from a master bst file		
<b>physics.btx</b>	Various physics journals; must be run through <b>cpp.el</b> or <b>makebst.sh</b>		
<b>siam.bst</b>	SIAM (LATEX style in latex- style)		

### Contents of the BIBTEX 0.98 Collection

This contains files that are specific to version 0.98 of BIBTEX. Many of these files are to be used with files in the LATEX Collection. Files are located in pub/bibtex-style-0.98 for ftp users. Mail users should request files from the bibtex-style-0.98 archive.

<b>aaai-named.bst</b>	AAAI conference 1988 (LATEX style file in latex-style)
<b>btxbst.readme</b>	
<b>btxbst.doc</b>	A master file for BIBTEX styles with standard styles and some new ones
<b>natsci.bst</b>	Generic natural sciences (LATEX style file in latex-style)
<b>newalpha.bst</b>	Modified alphabetic

### Contents of the TEX Programs

This contains programs that are of general interest to TeX users in general. Files are located in pub/tx-programs for ftp users. Mail users should request files from the tx-programs archive.

<b>docsty.shar</b>	Program to convert .doc to .sty by stripping comments
<b>dvidoc1.shar</b>	DVI to character device filter for
<b>dvidoc2.shar</b>	Unix BSD systems
<b>dvidvi.shar</b>	Program to select pages from dvi files or print multiple pages on a single page
<b>fig2epic1c.shar</b>	Converts fig code to epic or eepic files
<b>lgraph.shar</b>	Data to graph command filter in Pascal
<b>pcwritex.uue</b>	PC-Write to TeX interface that has been uuencoded

---

**“A new implementation of the array- and  
tabular-environments of L<sup>A</sup>T<sub>E</sub>X”  
(TUGboat 9#3) — addenda**

Frank Mittelbach  
Universität Mainz

### 1 Corrections to the macros

After submitting the article describing the new implementation of the array- and tabular-environments of L<sup>A</sup>T<sub>E</sub>X two errors were found and corrected. The changes below correspond to the items in the figure at the bottom of this page.

- a. If a flushright column entry is left empty, a hidden \unskip in the macro \insert@column cancels the stretch (\hfil). The fix is easy. The lines to be altered in the definition of the macro \classz are below.
- b. The second bug was a typo. In the macro \otfor we should test for \empty which represents “empty”. The typo \empty, however,

resolves to “undefined”. Actually this doesn’t make any difference. Only the case with the empty argument is no longer optimized. The proper first line in the definition of \otfor is below.

- c. These corrections should also be reflected in the version number.
- d. Before the two lines of c, we also add a new line of code to avoid reading this file twice.

### 2 Future versions

After many discussions with L<sup>A</sup>T<sub>E</sub>X users I think that it might be better to rename the preamble options t to p (old L<sup>A</sup>T<sub>E</sub>X meaning) and p to m (for middle). This will probably change in version 2 coming “sooner or later”<sup>1</sup>. If you are interested in changing the current source at your site you should do so provided you also add a \typeout line which reflects

---

<sup>1</sup>Quoted from the song “History will teach us nothing” (STING).

- 
- a. % The templates for {\tt l} and {\tt r} (i.e. \verb+\chnum+ \$1\$ or \$2\$)  
% are generated the same way. Since one \verb+\hfil+ is missing  
% the text is moved to the relevant side.  
% \begin{macrocode}  
% \dollar \insert@column \dollar \hfil \or  
% \hfil \dollar \insert@column \dollar \or

should be changed to

```
% The templates for {\tt l} and {\tt r} (i.e. \verb+\chnum+ $1$ or $2$)  
% are generated the same way. Since one \verb+\hfil+ is missing  
% the text is moved to the relevant side.  
% The \verb+\kern\z@+ is needed in case of an empty column  
% entry. Otherwise the \verb+\unskip+ in \verb+\insert@column+  
% removes the \verb+\hfil+.  
% \begin{macrocode}  
% \dollar \insert@column \dollar \hfil \or  
% \hfil\kern\z@ \dollar \insert@column \dollar \or
```

- 
- b. \def\otfor{\def\otfor{\do{\def\fortmp{\ifx\fortmp\empty

- 
- c. \typeout{Style-Option: ‘array’ v1.9k \space\space <30.11.88> (F.M.)}  
\typeout{English documentation dated \space\space <30.11.88> (F.M.)}

- 
- d. \ifundefined{\dollar}{\endinput}

the change.<sup>2</sup> You only have to change the two corresponding characters in the `\ifx` tests in the macro `\@testpach`.

The old source (v1.9g) should be available by now at the Heidelberg listserver and in the L<sup>A</sup>T<sub>E</sub>X style collection in US listservers.

### 3 Further developments in Mainz

The following is a short overview concerning the recent activities in Mainz. If there is similar work ongoing at other sites I would be glad to learn about it.

- One year ago I wrote a set of macros extending the theorem-environments of L<sup>A</sup>T<sub>E</sub>X which allows the user to use and define `\theoremstyle`'s similar to the `\pagestyle` macro. This code is documented and tested and will be submitted to TUGboat (The bottleneck is the translation to English, sigh!).
- Our main project is to remove the `\protect` macro in L<sup>A</sup>T<sub>E</sub>X, because we think this is the most critical feature, especially for novice users. This work is almost finished (only the letter-environment remains to be changed), but the documentation is still rather poor. I hope the code can be released this year ...
- As a reaction to my last article I was asked if I am willing to put the style file used to produce the documentation into the public domain. This is my intention but I feel that contrary to the user interface the inner macros need polishing. Therefore I shall describe the user interface in the next section, to enable others to use this documentation tool in the near future.<sup>3</sup>

### 4 The user interface of the 'doc' style option

#### 4.1 General conventions

The main characteristic of a T<sub>E</sub>X file prepared for the 'doc' style option is the use of the % character. Every line of documentation starts with this character in column one. Therefore such a (style) file might be used directly by T<sub>E</sub>X since all comments are bypassed. The documentation style file

---

<sup>2</sup>Other changes are only allowed if the style file is renamed but this is seldom sensible because it increases the number of (normally) unsupported files. See for example the L<sup>A</sup>T<sub>E</sub>X style collection. I therefore hope that all improvements, suggestions and/or bug reports are sent back to me.

<sup>3</sup>If someone is interested in using this style file right now, he or she should send me a short message.

(`doc.sty`) redefines this character to 'ignore'. In this way documentation comments are made visible to T<sub>E</sub>X. In this mode we want to format the code 'verbatim' rather than evaluating it. So lines of code are surrounded by

```
% \begin{macrocode}
...
% \end{macrocode}
```

Note that there must be exactly four spaces between the % and the `\end{macrocode}`—T<sub>E</sub>X is actually looking for this string and not for a macro.

This environment also has a star form which formats the spaces as `\` just in case somebody needs this.

#### 4.2 Describing a new macro

A macro definition is surrounded by the environment `macro` which has one argument—the macro name *without* the backslash. This argument is used to generate a main index entry. It is also printed in the left margin prefixed by a backslash. Nesting of this environment is provided (up to three levels) to allow macro definitions within others.

#### 4.3 Index facilities

As mentioned above the argument from every macro environment is used to generate a main index entry. Furthermore all code sections surrounded by the `macrocode` environment are scanned and every control sequence starting with a backslash will produce a normal index entry for cross referencing purposes<sup>4</sup>. The actual index entries are produced with the macros `\SpecialMainIndex` and `\SpecialIndex`. Both insert a backslash in front of their argument which is also used as the sort key. There also exists a general `\SortIndex` command with two arguments—the sort key and the index entry.

The resulting `idx` file might be sorted by the `makeindex` program, but other programs can be used if the commands mentioned above are redefined.

#### 4.4 Additional bells and whistles

The `verbatim` environment is changed to ignore % characters. This makes it possible to place examples in the documentation.

There also exist several style parameters which determine the documentation layout.

---

<sup>4</sup>This was not implemented in the version used for the last article.

---

## Drawing histogram bars inside the L<sup>A</sup>T<sub>E</sub>X picture-environment

Rainer Schöpf

Institut für Physik

Johannes Gutenberg Universität

### Abstract

This article describes an enhancement of the L<sup>A</sup>T<sub>E</sub>X picture-environment to draw histogram bars. It is written in the self documenting T<sub>E</sub>X format developed by Frank Mittelbach.

### 1 User interface

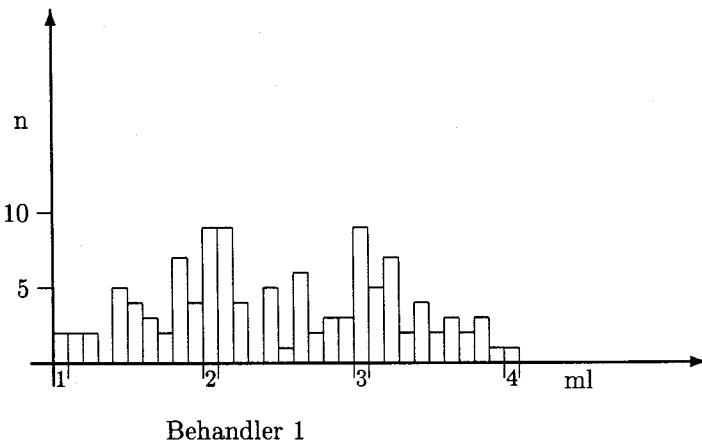
```
\typeout{^^JDocument style option 'histogr',
version 1.0 by RmS, released Nov 15, 1987}
```

This is a macro collection to draw histogram bars inside a picture-environment.  
Use is as follows:

```
\histogram(x0,y0)(x1,y1)...(xn,yn)
```

**\noverticallines**      The coordinate pairs specify the upper left corner of the histogram bars, i.e. this  
**\verticallines**      will draw a horizontal line from  $(x_i, y_i)$  to  $(x_{i+1}, y_i)$ , then a vertical line from  
 $(x_{i+1}, y_i)$  to  $(x_{i+1}, y_{i+1})$  if **\noverticallines** was specified, else from  $(x_{i+1}, y_0)$   
 $\text{to } (x_{i+1}, \max(y_i, y_{i+1}))$ . Default is **\verticallines**.  $y_0$  should be less or equal  
the minimum of all the  $y_i$  (i.e. other cases have not been tested).

Let's start with an example: to get the following picture:



Behandler 1

I used these L<sup>A</sup>T<sub>E</sub>X commands:

```
\setlength{\unitlength}{1mm}
\begin{picture}(100,65)(-10,-15)

\thicklines
\put(0,-3){\vector(0,1){50}}
\put(-3,0){\vector(1,0){90}}
\thinlines

\put(0,0){\line(0,-1){2}}
\put(2,0){\line(0,-1){2}}
\put(20,0){\line(0,-1){2}}
\put(22,0){\line(0,-1){2}}
\put(40,0){\line(0,-1){2}}
\put(42,0){\line(0,-1){2}}
```

```

\put(60,0){\line(0,-1){2}}
\put(62,0){\line(0,-1){2}}

\put(0,-1){\makebox(2,0)[t]{\small 1}}
\put(20,-1){\makebox(2,0)[t]{\small 2}}
\put(40,-1){\makebox(2,0)[t]{\small 3}}
\put(60,-1){\makebox(2,0)[t]{\small 4}}
\put(70,-1){\makebox(0,0)[t]{m1}}


\put(0,10){\line(-1,0){2}}
\put(0,20){\line(-1,0){2}}


\put(-3,8){\makebox(0,4)[r]{5}}
\put(-3,18){\makebox(0,4)[r]{10}}
\put(-3,30){\makebox(0,4)[r]{n}}


\put(15,-10){Behandler 1}

\histogram(0,0)(0,4)(2,4)(4,4)(6,0)(8,10)(10,8)(12,6)(14,4)
(16,14)(18,8)(20,18)(22,18)(24,8)(26,0)(28,10)(30,2)
(32,12)(34,4)(36,6)(38,6)(40,18)(42,10)(44,14)(46,4)
(48,8)(50,4)(52,6)(54,4)(56,6)(58,2)(60,2)(62,0)
\end{picture}

```

## 2 Implementation

- \hist@x Here's how it is implemented: first we allocate three counters that are needed later on. \hist@x and \hist@y are the *x* and *y* coordinate of the *current point*, i.e. the point that serves as a start for the next box of the histogram. \hist@ystart holds the *y* coordinate of the first point, i.e. *y<sub>0</sub>*.
- ```

\newcount\hist@x
\newcount\hist@y
\newcount\hist@ystart

```
- \noverticallines We need a switch to decide if the vertical lines of the histogram boxes are to be drawn from *y<sub>i</sub>* to *y<sub>i+1</sub>* or from *y<sub>0</sub>* to  $\max(y_i, y_{i+1})$ . Default is the latter.
- ```

\newif\ifhist@vert
\let\verticalines\hist@verttrue
\let\noverticalines\hist@vertfalse

```
- \hist@verttrue
- \histogram The \histogram command takes the starting point as argument and initializes the counters. \hist@x, \hist@y and \hist@ystart are set to *x<sub>0</sub>*, *y<sub>0</sub>* and *y<sub>0</sub>*, respectively.
- ```

\def\histogram(#1,#2){\hist@x #1 \hist@y #2 \hist@ystart\hist@y
Then the macro \hist@next is used.
\hist@next}

```
- \hist@next \hist@next looks at the next token to see if there is another open parenthesis. If this is the case it calls \hist@box, otherwise \hist@end.
- ```

\def\hist@next{\@ifnextchar ({\hist@box}{\hist@end}}

```
- \hist@box The macro \hist@box does nearly all the work. The first thing to do is to set the temporary counter \tempcnta to  $x_{i+1} - x_i$ . Remember that \hist@x is the *x* coordinate of the last point (i.e. *x<sub>i</sub>*) whereas the macro's first argument is *x<sub>i+1</sub>*. So we write
- ```

\def\hist@box(#1,#2){\tempcnta -\hist@x
\advance\tempcnta #1

```

The next step is easy: draw the horizontal part of the histogram box. The line starts at  $(x_i, y_i)$  and has length  $\@tempcnta\unitlength$ .

```
\ifnum \@tempcnta > \z@  
    \put(\hist@x,\hist@y){\line(1,0){\@tempcnta}}\else  
    \put(\hist@x,\hist@y){\line(-1,0){-\@tempcnta}}\fi
```

Now set  $\hist@x$  to  $x_{i+1}$ :

```
\hist@x #1
```

If  $\verticallines$  was set we first set  $\@tempcnta$  to  $\max(y_i, y_{i+1})$ :

```
\ifhist@vert  
    \ifnum \hist@y > #2 \@tempcnta \hist@y  
    \else \@tempcnta #2 \fi
```

then we set  $\@tempcntb$  to the same value and  $\@tempcnta$  to the length of the line to draw.

```
\@tempcntb\@tempcnta  
\advance\@tempcnta -\hist@ystart
```

We draw the line

```
\put(\hist@x,\@tempcntb){\line(0,-1){\@tempcnta}}%
```

which finishes this case.

```
\else
```

In the other case (i.e. if  $\noverticalines$  was set) we have to draw a line from  $y_i$  to  $y_{i+1}$ . We set  $\@tempcnta$  to  $y_{i+1} - y_i$

```
\@tempcnta -\hist@y  
\advance\@tempcnta #2
```

and draw the line.

```
\ifnum \@tempcnta > \z@  
    \put(\hist@x,\hist@y){\line(0,1){\@tempcnta}}\else  
    \put(\hist@x,\hist@y){\line(0,-1){-\@tempcnta}}\fi
```

Thus endeth the drawing.

```
\fi
```

Finally we set  $\hist@y$  to  $y_{i+1}$  and call  $\hist@next$  to look for the next coordinate pair.

```
\hist@y #2\hist@next}
```

**\hist@end** There is only one thing we left out: what if there is not another open parenthesis? That's the easy part: do nothing.

```
\def\hist@end{}
```

Frank Mittelbach has suggested that the  $x$ -coordinate should specify the midpoint of the histogram bar, not the upper left corner. However, I don't see how this will work if the bars have different widths. What do you think about it?

Well, that's all. Use it and enjoy.

---

## Vertical Centering for Transparencies

Dezső Nagy  
Geological Survey of Canada

The problem of vertical adjustment, when preparing transparencies is well known. It is very frustrating, that after some small changes, the number of lines changes and the vertical centering must be done again. In addition to this, first time users face the problem of selection of the size of letters to be used, the maximum page dimensions, which fit into a standard transparency holder, etc. Also, the question of the location of an identification on the page must be addressed.

In the following, a very simple procedure for L<sup>A</sup>T<sub>E</sub>X users is presented, which solves most of the problems related to producing transparencies. There are enough comments included to use it; here only a few pointers are given, which may be of some help.

If one wants to use the macro as is, then everything needed is done for the user. A good size bold-face character set is selected, identification with the date is produced in the footnote, and an automatic `\newpage` is provided. There is no check to see if there is enough room for the text on the page. Possibly the only parameters which require new values are the

`\voffset \hoffset`

which are device-specific.

As indicated on the comment lines, material for the transparency can be read in from a file with the standard `.tex` extension : in this case at least one space must be left between the name and the closing bracket (see `transp1`).

As can be seen from the examples, there is some flexibility in the usage of the macro. However if the default setting is not suitable, then the user should modify the macro to suit the requirements.

**Editor's note:** Following is the code used to obtain the adjoining transparencies. The driver file defines a few macros and uses `\vc` to create the centered text.

### 1 Driver File

```
%%%%%
% Driver file defining macros and creating
% 3 transparencies.
%
\documentstyle{article}
%
```

```
\textheight=9.0truein
\textwidth=6.5truein
\topskip=Opt

\voffset=-1.0truein
\hoffset=-1.0truein

\def\bc{\begin{center}}
\def\ec{\end{center}}


\def\vc#1#2{%
  \rule
  \vfill
  \footnotetext{%
    \large\hspace{-4mm}{#1}\quad\today\%
  }%
  {\Huge\bf#2}%
  \vfill\newpage
}

\def\thefootnote{} % to avoid footnote numbers
\pagestyle{empty} % to avoid page numbers

\begin{document}

% call the macro with two parameters as :
%
%           \vc{#1}{#2}
%
% #1 : id, i.e. the name, title, etc., which
% goes to the bottom of the page.
% The date is supplied automatically.
%
% #2 : text for the transparency,
% including \indent, \noindent,
% \itemize, or whatever. This text
% can be put in directly, or read in
% by using : \input text where
% text.tex contains the material for
% the transparency.

% first example with defaults

\vc{Nagy : Geoid 1}{\input transp1 }

% second example: the fonts are changed,
% both for the id and the text, but not
% for the title. The text is supplied
% directly.

\vc{\sc Nagy : Geoid 2}{%
  \bc T R I A L \quad T E X T\ec \Large
  This is a trial text to find out how
  the linebreaking algorithm and other
}
```

special instructions may work in this arrangement. It is hoped that this text may clarify the points hidden by the fog earlier.)

```
% third example: \sl is used for id, and
% default font for text. The text is
% read from the file : transp3.tex, then
% augmented by directly putting in the
% rightflush text : FINI to the end
% of the last line.

\vc{\sl Nagy : Geoid 3}%
  {\input transp3 \unskip~\hfill FINI}

\end{document}
% End of driver file
```

## 2 TRANSP1.TEX

Following is the code contained in the file transp1.tex.

```
%%%%%%%
% input file to test the \vc#1#2 macro
% transp1.tex
%
\bc GEOID COMPUTATION \ec

\vspace{10mm}

\noindent The problems related to the computation of a gravimetric geoid for Canada can be summarized as follows :
\vspace{5mm}
\begin{itemize}
\item Selection of reference system
\item Preparation of input data
\item Computation of geoid
\item Presentation of results
\end{itemize}
\vspace{5mm}
In the following, these topics will be discussed in some detail.
```

## 3 TRANSP3.TEX

Following is the code contained in the file transp3.tex.

```
%%%%%%
% input file to test the \vc#1#2 macro
% transp3.tex
%
\bc PRESENTATION OF RESULTS \ec

\vspace{10mm}
```

\noindent The Oblique Lambert Zenithal Equal Area projection has been selected as a base map. The followings will be shown :

```
\vspace{5mm}
\begin{itemize}
\item Data distribution for
\begin{itemize}
\item North America, Worldwide
\end{itemize}
\item Satellite model
\begin{itemize}
\item Gravity and geoid
\end{itemize}
\item Gravity and residual anomaly for
\begin{itemize}
\item North America and Worldwide
\end{itemize}
\item Gravimetric geoid map
\end{itemize}
\vspace{5mm}
```

The results will be presented in various forms, such as APPLICON-map, contour-map, etc.

## GEOID COMPUTATION

The problems related to the computation of a gravimetric geoid for Canada can be summarized as follows :

- Selection of reference system
- Preparation of input data
- Computation of geoid
- Presentation of results

In the following, these topics will be discussed in some detail.

## **T R I A L      T E X T**

This is a trial text to find out how the linebreaking algorithm and other special instructions may work in this arrangement. It is hoped that this text may clarify the points hidden by the fog earlier.

## PRESENTATION OF RESULTS

The Oblique Lambert Zenithal Equal Area projection has been selected as a base map. The following will be shown :

- Data distribution for
  - North America, Worldwide
- Satellite model
  - Gravity and geoid
- Gravity and residual anomaly for
  - North America and Worldwide
- Gravimetric geoid map

The results will be presented in various forms, such as APPLICON-map, contour-map, etc.

FINI

## Typesetting Bridge via L<sup>A</sup>T<sub>E</sub>X

C.G. van der Laan  
Rekencentrum RUG  
The Netherlands

### Abstract

L<sup>A</sup>T<sub>E</sub>X macros and a bidding environment for typesetting bridge card distributions and bidding sequences are given. Examples borrowed from bridge literature are supplied.

### 1 Card deals

In bridge literature diagrams of distribution of cards over the hands are often given in order to demonstrate bidding sequences or to explain play technique. In order to do this systematically and to abstract from layout details I wrote a macro—\crdima—with six parameters:

**first parameter:** text, especially who is the dealer and what is the vulnerability. For example: N/None, for North dealer and vulnerability none.

**second parameter:** text. For example, indication of deal as in Deal 1 or in  
`\begin{minipage}[t]{\br}`  
`Deal:\demo`  
`\end{minipage}`

**next four parameters:** the four hands N, E, S, W, clockwise. Each hand is a call of the \hand macro with four parameters: the ♠, ♥, ♦, ♣ cards.

As example,

```
\crdima{N/None}{%
  \begin{minipage}[t]{\br}
    Deal:\demo
  \end{minipage}}%
{\hand{J74}{AJ}{QJT2}{Q874}}%N
{\hand{K86}{T9542}{874}{T3}}%E
{\hand{QT952}{Q83}{AK5}{A6}}%S
{\hand{A3}{K76}{963}{KJ952}}%W
```

yields

|                                   |                                   |               |
|-----------------------------------|-----------------------------------|---------------|
| N/None                            | ♠ J74<br>♥ AJ<br>♦ QJT2<br>♣ Q874 | Deal:<br>demo |
| ♠ A3<br>♥ K76<br>♦ 963<br>♣ KJ952 | ♠ K86<br>♥ T9542<br>♦ 874<br>♣ T3 |               |
| ♠ QT952<br>♥ Q83<br>♦ AK5<br>♣ A6 |                                   |               |

**Remarks.** By this levelling I circumvented the limit of the number of parameters. Because parameter substitution is done by ‘text’ replacement there is no ‘(strong) type checking’ as in modern high-level programming languages.

There is no check on the correctness of the cards (correct number, distribution, multiple occurrence or omission), nor on the correct sequence of the parameters. In SGML, compliance with the input syntax can be imposed with enhanced user convenience and alleviated proofreading, but at the expense of elaborate coding, [4]. No test on the correctness of the sequence of the hands is possible, except for the mechanism of ‘named’ parameters. The chosen order of the parameters is in agreement with how the play goes.

The \crdima macro can be used to display all phases of the play. Hands can be suppressed at discretion of the user by empty actual parameters. A void can be supplied via --. In the listings of the commands used for the examples the quote environment command is omitted.

For (bridge) tournaments, plays are often dealt by computer. At the end of tournaments players appreciate prints of the deals. For that purpose my (Pascal) deal program generates ASCII output—for simple display on the PC—as well as L<sup>A</sup>T<sub>E</sub>X input, optionally. This input is printed with the aid of \crdima. Parameter testing is superfluous for L<sup>A</sup>T<sub>E</sub>X input generated this way.

### 2 Bidding

In the context of bidding theory I use a bidding environment. The given card deal takes the following ACOL bidding

|      |       |      |       |
|------|-------|------|-------|
| West | North | East | South |
| —    | 1♠    | no   | 1♦    |
| no   | 2♠    | no   | 4♦    |
| a.p. |       |      |       |

obtained via

```
\begin{bidding}
-- \> 1\c\> no \> 1\s\\
no \> 2\s\> no \> 4\s\\
a.p.
\end{bidding}
```

**Remark.** The bidding environment is independent of the number of bid rounds.

### 3 Macro texts

```
\newcommand{\hand}[4]{
\begin{minipage}[t]{\br}\%I chose \br=8em
\begin{tabbing}
%width of parbox equals:
%min{\br, max{string #1, ..., string #4}}
\(\spadesuit\)\ \=\#1 \\
\(\heartsuit\)\ \>\#2 \\
\(\diamondsuit\)\>\#3 \\
\(\clubsuit\)\ \>\#4
\end{tabbing}
\end{minipage} }%end \hand
%
\newsavebox{\NESW}
\savebox{\NESW}[4em]{%
\raisebox{-1.5\baselineskip}{%
\fbox{\small W
\raisebox{2.6ex}{N}
\hspace*{-1em}
\raisebox{-2.6ex}{S}
{E}
}
}%
} }%end \NESW
%
\newcommand{\crdima}[6]{%
\begin{tabular}[t]{lll}
#1 & #3 & #2\\
#6 & \usebox{\NESW} & #4\\
& #5 &
\end{tabular} }%end \crdima
%
\newenvironment{bidding}%
{\begin{tabbing}
xxxxxx=xxxxxx=xxxxxx=xxxxxx \kill
West \>North \>East \> South\\
\end{tabbing}}%end bidding
```

To eliminate data integrity errors the listings of the above macros and the listings of the commands used in the examples are ‘included’ via a transparent verbatim like environment, [7]; so the same files were used for execution and listing.

### 4 Some more examples

a. In order to illustrate general bidding theory from the viewpoint of one hand only, the \hand macro can be used. The following layout, heavily used in [3],

|         |      |       |      |       |
|---------|------|-------|------|-------|
| ♠ AKJ42 | West | North | East | South |
| ♥ AK9   | -    | 1♦    | no   | 1NT   |
| ♦ T832  | 2♣   | ?     |      |       |
| ♣ T     |      |       |      |       |

is obtained via

```
\hand{AKJ42}{AK9}{T832}{T}\hfill
\begin{minipage}[t]{\br}
\begin{bidding}
-- \> 1\s\> no \> 1NT \\
2\c\> ?
\end{bidding}
\end{minipage}
```

b. For issues related to defense play one often displays only the dummy hand and your own hand. The following example—layout and text—is from [2].

|         |     |
|---------|-----|
| ♠ AJ632 | N   |
| ♥ 43    | W E |
| ♦ KQ7   | S   |
| ♣ A85   |     |

|        |  |
|--------|--|
| ♠ 985  |  |
| ♥ 852  |  |
| ♦ AJ5  |  |
| ♣ KQT3 |  |

|      |       |      |       |
|------|-------|------|-------|
| West | North | East | South |
| 1♠   | no    | 2♥   | no    |
| 2NT  | no    | 4♦   | a.p.  |

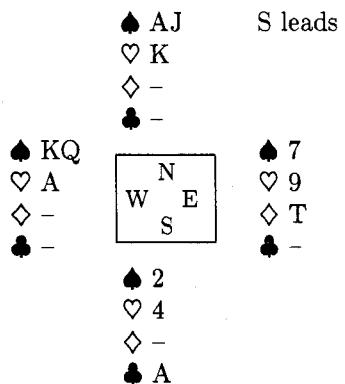
Against 4♥ South starts ♠K, taken with ♣A. Leader continues ♥AKQ. On the third round of ♥’s, partner discards ♦9 (indicates interest in ♠). Leader continues with ♦2, how do you continue?

The example is obtained via

```
\crdima{}{}%
{}{}{\hand{985}{852}{AJ5}{KQT3}}%S
{\hand{AJ632}{43}{KQ7}{A85}}%W
\begin{bidding}
1\s\> no \> 2\h\> no\\
2NT\> no \> 4\h\> a.p.
\end{bidding}
```

**Remark.** In a similar way W-N, N-E, E-S hands, or W-E, N-S hands, or one hand only, with NESW diagram, can be displayed simply by a suitable call of \crdima.

c. Finally, an endplay — positional squeeze — from [5] is given.



The example is obtained via

```
\crdimaf{\S leads \c A}%
{\hand{AJ}{K}{--}{--}}%N
{\hand{7}{9}{T}{--}}%E
{\hand{2}{4}{--}{A}}%S
{\hand{KQ}{A}{--}{--}}%W
```

## 5 Variation

An elementary, and in a sense more general, crdimma macro is

```
\newcommand{\crdimaele}[9]{%
\begin{tabular}[t]{lll}
#1 & #2 & #3 \\
#4 & #5 & #6 \\
#7 & #8 & #9
\end{tabular}}%end crdimaele
```

All the given examples can be handled with `crdimaele`. `crdimaele` applied to the original deal without NESW diagram reads

```
\crdimaele{N/None}%
Left text
{\hand{J74}{AJ}{QJT2}{Q874}}%N
{\begin{minipage}[t]{\br}
Deal: demo; no\n NESW diagram
\end{minipage}}% Right text
{\hand{A3}{K76}{963}{KJ952}}%E
{}% Empty 'diagram'
{\hand{K86}{T9542}{874}{T3}}%S
{}%
```

with result

|        |                                   |                             |
|--------|-----------------------------------|-----------------------------|
| N/None | ♠ J74<br>♥ AJ<br>♦ QJT2<br>♣ Q874 | Deal: demo; no NESW diagram |
|        | ♠ A3<br>♥ K76<br>♦ 963<br>♣ KJ952 |                             |
|        | ♠ K86<br>♥ T9542<br>♦ 874<br>♣ T3 |                             |
|        | ♠ QT952<br>♥ Q83<br>♦ AK5<br>♣ A6 |                             |

**Remarks.** A NESW diagram is obtained with `\usebox{\NESW}` — or something you have designed yourself — as fifth parameter.

An elegant solution to the problem of having a default NESW figure which could be overruled by another figure is the optional parameter mechanism, which — helas — is lacking in the macro facility of L<sup>A</sup>T<sub>E</sub>X. The same applies to the bidding environment with the default bid sequence West North East South. Again via the mechanism of optional parameters one could provide another bid sequence order, use abbreviations or names suited for other languages. For the hand parameters one could think of the mechanism of ‘named parameters’ with ultimately complete freedom of the sequence order of the parameters. A step in this direction is to use variables, e.g., `\ns` for spades of the North hand and so on. The use of `\hand` for North is then `\hand{\ns}{\nh}{\nd}{\nc}`. Of course one could modify `crdima` into a version with parameter calls replaced by ‘global’ variables. My case rests.

## Conclusions

The author claims that bridge publications with respect to card distributions and bidding sequences can be typeset easily with high quality via L<sup>A</sup>T<sub>E</sub>X and the given macros. Proofreading of deals not generated and typed by computer is error prone and remains tiresome.

The lack of the facility of optional parameters in the `\newcommand` command and the `newenvironment` environment is felt as an understandable inelegancy.

## Acknowledgements

The paper is inspired upon Appelt’s Typesetting Chess, [1]. I didn’t follow his approach of updating the hands along with each ‘move’, because in Bridge there is no algebraic notation for what is played ev-

ery round; generally a description is given, see section 4 example b.

The author is grateful to Victor Eijkhout of the University of Nijmegen and Nico Poppelier of the University of Utrecht for their suggested improvements. With my colleague Jan Luyten I had the pleasure of fruitful discussions.

### References

- [1] Appelt, W. (1988): Typesetting Chess. TUGboat, 9, 3, 284-287.
- [2] BRIDGE. Monthly of the NBB (Dutch Bridge Union).
- [3] Crowhurst, E. (1986): ACOL in competition. Pelham. London.
- [4] Grootenhuis, J. (priv. comm.): Kaartverdelingen en biedverloop bij bridgen—Een SGML tutorial. (Dutch).
- [5] Kelder, J., B. van der Velde (1986): Dwangposities tegen één tegenstander. Becht. A'dam. (Dutch). Translated from:  
Kelsey, H.W. (1985, paperback):  
Simple squeezes. Gollancz. London.
- [6] Lamport, L. (1986): LATEX, a document preparation system. Addison-Wesley.
- [7] Mulders, H.P.A. (priv. comm.): \verb+input+.

## News & Announcements

### WOODMAN '89: Workshop on Object Oriented Document Manipulation

29-31 May 1989  
Rennes, France

The advent of electronic publishing has led to increasing interest in object-based description and processing of document structure. Standards such as ODA and SGML are a first step in that direction although this is not made explicit. Hypertext and Hypermedia systems follow another related road. Object-Oriented Languages, object-oriented databases and more generally innovative techniques based on the object model may help to support the handling of document logical structure.

This workshop will deal with ideas such as these. Presentations have been invited on research results, prototype experiences and surveys on

#### Object Oriented Languages AND:

- Document Structuring
- ODA
- SGML
- Imaging Models
- Mixed Mode Documents
- Hypertext and Hypermedia
- Document Databases
- Character Design and Recognition
- Parallel Processing of Documents

WOODMAN'89 is organized with the help of the BIGRE bulletin. Co-chairmen are Jacques André and Jean Bézivin. For more information, contact

BIGRE/WOODMAN'89  
IRISA – Campus de Beaulieu  
F-35042 Rennes Cedex, France  
Telefax: (33) 99 38 38 32  
Telex: 950 473F  
e-mail: jandre@irisa.irisa.fr

## Calendar

**1989**

- May 1      **TUGboat Volume 10, No. 2:**  
Deadline for receipt of manuscripts.
- May 16–17    Congres GUTenberg, Paris, France.  
Theme: TeX and graphics. For  
information, contact Bernard Gaulle  
(Bitnet: UCIR001@FR0RS31. See also  
page 118.)
- May 29–31    WOODMAN '89, Workshop on  
Object Oriented Document  
Manipulation, Rennes, France. For  
information, contact Jacques André  
(jandre@irisa.irisa.fr or  
fax: (33) 99 38 38 32. See also  
page 116.)
- Jun 29–30    NTG — Nederlandse TeX  
Gebruikers, "TeX happening".  
Utrecht, The Netherlands.  
For information, contact  
C.G. van der Laan (Bitnet:  
cgl@hgrrug5, or +31/50 633374.  
See also TUGboat 9#3, page 316.)
- Jul 30–  
Aug 4        ACM SIGGRAPH '89, Boston,  
Massachusetts. Contact: Chris Herot  
or Branko Gerovac, (312) 644-6610.
- Sep 11–13    TeX89: 4th Annual Meeting  
of European TeX Users,  
University of Karlsruhe, FRG.  
For information, contact  
Rainer Rupprecht (Bitnet:  
RZ32@DKAUNI48. See also page 118.)
- Oct 3–6        Protext V Conference:  
5th International Conference on  
Computer-Aided Text Processing  
and its Applications. Boston,  
Massachusetts. For information,  
contact Protext Conference, INCA,  
P. O. Box 2, Dun Laoghaire, Ireland;  
+353-1-613749.
- Oct 12–13    RIDT'89 — Raster Imaging  
and Digital Typography.  
Ecole Polytechnique Fédérale,  
Lausanne, Switzerland.  
For information, contact  
Prof. R.D. Hersch,  
Lausanne, Switzerland;  
(4121) 47 43 57/693 43 57  
or hersch@elde.epfl.ch;  
or Debra Adams, (415) 494-4022  
or adams.pa@Xerox.com.  
(See announcement, TUGboat 9#3,  
page 316.)

**TeX Users Group 1989 Conference****— Tenth Anniversary —****Stanford University, Stanford, California**

Aug 14–18 Short Courses: to be announced

Aug 21–23 **TUG Annual Meeting**

Aug 24–25 Short Courses: to be announced

- Sep 11      **TUGboat Volume 10, No. 3:**  
Deadline for receipt of manuscripts  
(tentative).

**1990**

- Jan 15        **TUGboat Volume 11, No. 1:**  
Deadline for receipt of manuscripts  
(tentative).

For additional information on the events listed  
above, contact the TUG office (401-751-7760) unless  
otherwise noted.

---

**GUTenberg Congrès:  
TeX et les graphiques**

Paris, 16–17 May 1989

The second annual Congrès of the Groupe francophone des Utilisateurs de TeX will be held in Paris on 16–17 May 1989. The theme will be TeX and graphics. Various papers will present a survey of TeX graphics under MS/DOS, UNIX, VMS and Mac. New products will be unveiled. Major topics are presently oriented around METAFONT and PostScript. A “grand gourou” speaker will be invited. Will this person speak about new color products?

An introductory LATEX course is being organized by Olivier Nicole and Jacques Beigbeder; a tutorial on METAFONT is planned by Victor Ostromoukhov.

A general assembly of GUTenberg members will be held on the evening of the first day in order to approve the composition of GUTenberg's executive board and to approve its decisions.

The two-day cost for the Congrès will be nearly 400FF for GUTenberg members, plus an additional 200FF for non-members. The location will be at a Paris north gate (Porte de la Chapelle) that's not so far from Roissy airport.

The *Cahiers GUTenberg* will begin publication officially during 1989. This journal will be formatted for A4 paper, using French typographic conventions. A LATEX style file will be provided to prospective authors. It is expected that many of the papers to be presented at the Congrès will be published in the *Cahiers*. Subscription prices have been set at 150FF per year for GUTenberg members, 250FF for non-members. Four issues are planned for this year.

For information on the Congrès or the *Cahiers*, contact

Bernard Gaulle  
CIRCE/CNRS  
BP 167  
91403 Orsay Cedex, France  
16.(1) 69.82.41.07  
Bitnet: UCIR001@FROR31

---

**TeX89: 4th Annual Meeting of  
European TeX Users**

September 11–13, 1989  
Karlsruhe, FRG

TeX89, the 4th European TeX Conference, will take place at Karlsruhe University, FRG, from Monday, September 11, to Wednesday, September 13, 1989. The conference will be organized by Anne Brüggemann-Klein, Department of Computer Science, University of Freiburg, and Rainer Rupprecht, Computing Center, University of Karlsruhe.

Following the tradition of last years' conferences, contributions are welcome from all areas of TeX, METAFONT, and related subjects. Likely themes might include:

- Document structures  
(LATEX, SGML, ODA, ...)
- Non-technical TeX  
(humanities, music, exotic languages, ...)
- Other technical areas  
(chemistry, physics, biology, ...)
- Difficult jobs with TeX, LATEX, ...
- Graphics and TeX
- TeX training
- TeX as part of a larger system  
(user interfaces, tools, environments, ...)
- TeX as a production tool
- Fonts to use with TeX  
(METAFONT and other systems)
- TeX and PostScript
- Macro packages
- Public domain TeX vs. commercial TeX

Besides traditional paper sessions, discussion groups on special subjects and exhibitions will be organized. In a special session at the end of the conference, highlights of the discussion groups will be presented to the general audience. Conference proceedings will be published after the conference.

Various workshops and participatory seminars will be offered before and after the conference. Proposals for topics and voluntary tutors are welcome.

The conference fee will be approximately DM 280,-. The fee includes registration materials, lunches, social events and a copy of the conference proceedings.

For information, contact

Rainer Rupprecht  
Rechenzentrum  
Universität Karlsruhe  
Postfach 6980  
7500 Karlsruhe 1, FRG  
Bitnet: RZ32@DKAUNI48

---

## A UK TeX Users' Group— Report of a preliminary meeting

David Osborne  
University of Nottingham, UK

National user groups in the TeX-speaking world are quite the fashion these days, it seems. With the formation of user groups in West Germany, France and the Netherlands, there is an active and growing involvement of European TeXers in meeting to discuss TeX matters. To date, although there has been a widespread and growing use of TeX within the United Kingdom academic community and also, fortunately, outside it, contact between users of TeX has been largely informal. It has been common for TeX users who encountered each other at various non-TeX seminars and conferences to swap experience and news during coffee-breaks.

The undoubtedly success of the Third European TeX conference in Exeter in July ("TeXeter88") brought together a significant number of British users and produced a momentum for the formation of a user group devoted to TeX. Credit for masterminding both the Exeter conference and the first organised meeting of UK TeX users goes to Malcolm Clark of Imperial College, London. Thus it was during one of those coffee-break TeX get-togethers in late September that I found myself the subject of Malcolm's well-known persuasive skills and agreed to organise an inaugural meeting at which a UK group might be formed.

Due to the short notice at which the meeting was arranged, its announcement was by electronic mail and details were also given in the UKTeX Digest. This led to an unfortunate emphasis in favour of academic users; however, a number of TeX users from commercial companies were among the 36 who came to the meeting held at the University of Nottingham on November 4th. I tried to plan the programme with a balance between talks and an opportunity to discuss the aims of the group but, as might be expected, there was just not enough time for everything.

The first speaker was Charles Curran of Oxford University. Charles has been actively interested in METAFONT for some years and discussed how fonts can be matched to specific output devices by tuning the `mode_def` parameters. It was obvious that this is a lengthy process of trial and error. During questions which followed, it was suggested that optimum results could be obtained if a definitive standard were available in the form of a laser-printed

Computer Modern "Sample Book", in the manner of Volume E of "Computers and Typesetting".

Since I have been involved in setting up TeX on systems in our computing centre, I tried to give an overview of some of the pitfalls which await the novice implementor, despite the amount of helpful information in README files and other documentation. In fact, this is part of the problem: the sheer volume of information which needs to be assimilated and, ideally, before starting. I wondered if there is scope for a succinct yet detailed "How to implement TeX and METAFONT on system X..." document for each major system.

From the somewhat arcane topics of METAFONT and TeX implementation, our attention turned to an topic of which concerns many users in a stimulating look at "LATEX in ConTEXt" from Sue Brooks of BUSS Ltd. As someone using LATEX to document a significant software package, Sue's point of view was that of the author and she gave a clear overview of the benefits which LATEX offers a writer for logically structuring their document.

After lunch, we were treated to two very interesting reports from recent conferences. Cathy Booth from Exeter University had attended this year's TUG meeting in Montreal and described those threads running through the meeting which had most caught her attention. Chris Rowley of the Open University gave a companion talk on events at the Freiburg meeting in West Germany. It was clear from his description of the proposed further development of TeX and LATEX that we will be hearing much of interest from West German TeX users in future.

As has already been mentioned, time was short and it was with regret that, with his agreement, we postponed the talk on "Picture Languages" from Sebastian Rahtz of Southampton University; hopefully, Sebastian will give this talk at a later meeting. Malcolm Clark then took the floor to give a wide-ranging assessment of the world TeX situation. He had recently been appointed European TeX co-ordinator (*in absentia!*) and was not yet sure what this entailed.

Malcolm first outlined the activities of TUG itself, then broadened the picture to include the various national groups which have been formed, not forgetting those groups with peripheral interests in TeX as one aspect of electronic typesetting. Focusing on the main reason for the meeting, Malcolm then looked at what he felt was needed to further the TeX cause and how a UK group could help to this end. There is clearly a need for English hyphenation patterns and some tentative plans were made

to develop these. He suggested there is scope for more in the way of information for those currently outside the *TeX* community. This might include a descriptive "flyer"; details of typesetting bureaux who accept DVI files; descriptions of style files and more help for the implementor. Workshops on topics such as METAFONT, style files and document design could help current *TeX* users to make best use of the software.

In the end, no firm plans were made at this meeting to form a UK *TeX* users' group but those present confirmed an interest in doing so. It was agreed that a further meeting would be held in London in February or March, since London is, for many, more accessible than Nottingham and the date would allow Malcolm to report on a TUG committee meeting which he would attend in January.

Finally, my thanks go to all the speakers who agreed, at short notice, to talk at the Nottingham meeting and to my colleagues in the Cripps Computing Centre who helped with the domestic arrangements. Thanks, too, go to the trustees of *TeXeter88* for financial support which assisted in providing lunch and refreshments. I hope those who attended found the meeting useful and that it will be the first of many.

## Late-Breaking News

### Production Notes

Barbara Beeton

### Input and input processing

Electronic input for articles in this issue was received by mail and on floppy disk. Four articles were accepted in the form of camera copy (see the section on output), as were several illustrative samples that required special fonts or that could not be prepared on the American Mathematical Society's typesetter for other reasons.

Authors who had written articles previously for *TUGboat* typically submitted files that were fully tagged and ready for processing with the *TUGboat* macros—*tugbot.sty* for *PLAIN*-based files and *ltugbot.sty* for those using *LATEX*. (When

possible, a copy of the file actually used for production is returned to the author, along with the current version of the macros, if the author has requested them or if there have been changes. This seems to provide authors with incentive to write again for *TUGboat*.)

Articles in which no, or limited, *TeX* coding was present were tagged according to the *tugbot.sty* conventions. Articles tagged according to the author's own schemes were modified sufficiently to permit them to be merged with the rest of the stream. Especial care was taken to identify macro definitions that conflicted with ones already defined for *TUGboat*. In the case of *LATEX*-based articles, it was not necessary to consider interactions with other articles. (*\documentstyle{article}* is the basis for *ltugbot.sty*; no method has yet been devised for processing multiple articles in a stream, so each is processed separately, and physical pasteup is used where required to merge partial pages.) For *PLAIN*-based articles, the side-effects of an author's own definitions can usually be kept to a minimum by posting *\begingroup... \endgroup* around the article.

Most submissions for this issue were *PLAIN*; for convenience in processing, two items submitted in *LATEX* but using no significant *LATEX* features were converted to *PLAIN*. For these items, test runs of *TeX* separately and in groups were used to determine the arrangement and page numbers (to satisfy any possible cross references). The final processing of these articles was in three *TeX* runs, with ranges of page numbers skipped where *LATEX*-based items would be inserted; starting page numbers of *LATEX* items were indicated to support cross-referencing. *LATEX* items, as mentioned above, were processed individually, and arranged in the proper order after camera copy was produced.

The following articles were prepared using *LATEX*; all others (except for items received as camera copy, for which see below) used the regular *tugbot.sty*.

- Michael Harrison, *News from the VORTEX project*, page 11.
- Adrian Clark, *An enhanced *TeX*-editor interface for VMS*, page 14.
- Peter Abbott, *UKTeX and the Aston archive*, page 59.
- Michael Modest, *Using *TeX* and *LATEX* with WordPerfect 5.0*, page 68.
- Sriram Sankar, *APE—A set of *TeX* macros to format Ada programs*, page 89.

- Frank Mittelbach, Addendum to *A new implementation of the array- and tabular-environments*, page 103.
- Schöpf, Rainer, *Drawing histogram bars inside the LATEX picture-environment*, page 105.
- Dezső Nagy, *Vertical centering for transparencies*, page 108.
- C. G. van der Laan, *Typesetting Bridge via LATEX*, page 113

### Output

Camera copy for this issue of TUGboat was prepared on the devices indicated, and can be taken as representative of the output produced by those devices. The bulk of this issue was prepared at the American Mathematical Society on a VAX 8700 (VMS) and output on an APS- $\mu$ 5 using resident CM fonts and additional downloadable fonts for special purposes. The items listed below were received as camera copy; they were prepared on the devices indicated. The output devices used to prepare the advertisements were not usually identified; anyone interested in determining the device used for a particular ad should inquire of the advertiser.

- Unidentified:
- all advertisements.
- sample page of Japanese *TEX* input, page 10.
- Apple LaserWriter (300 dpi):  
Michael Harrison, *News from the VORTEX Project*, figure page 13.
- Autologic APS- $\mu$ 5 (1440 dpi):  
Donald E. Knuth, *Typesetting concrete*, page 31; DEC 10.
- Canon CX (300 dpi): Georgia Tobin, *A handy little font*, page 28.
- Hell Digiset (900 dpi): sample page accompanying *A TEX encounter in Japan*, page 9; FACOM (Fujitsu).
- Océ 6750 (20 dots/mm): Marius Broeren and Jan van Knippenberg, *High quality printing of TEX-DVI files in the VAX/VMS environment*, page 56; VAX/VMS.
- QMS PS 810 (300 dpi): Kim Kubik, *AmigaTEX...*, page 65; Amiga.

## Coming Next Issue

### Printing Vietnamese Characters by Adding Diacritical Marks via *TEX*

Brother Eric Vogel FSC  
Saint Mary's College, Moraga, CA

The technique described uses *TEX* to produce the diacritical marks for the various vowels needed for Vietnamese. The marks used are: accent grave, accent acute, tilde, question mark and dot (below). New vowels will be introduced using the above by placing a hat (above a or e) or breve accent (above a) or by attaching a "beard" (using the breve accent) to o and u. The macros for these diacritics will be given.

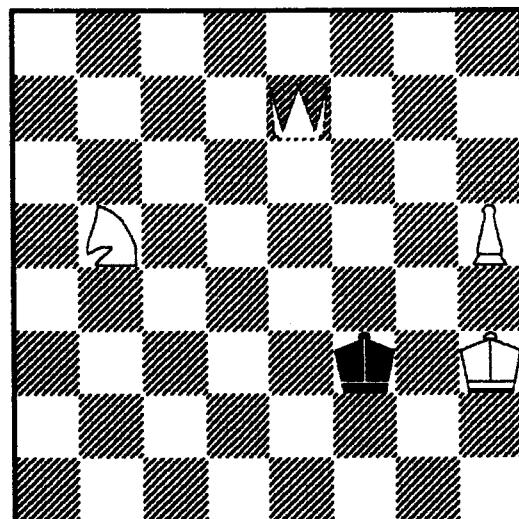
---

### Chess Printing via METAFONT and *TEX*

Zalman Rubenstein

Every chess fan knows the pleasant difference between seeing an interesting chess position or a chess problem printed, and looking at the standard description of the pieces by means of an  $8 \times 8$  coordinate system a1 to h8.

To help bridge this gap we have written a METAFONT-*TEX* program which enables one to print chess positions with ease, and to incorporate these positions with an arbitrary *TEX* output.



**Mate in three. Illustration 2.**  
**My 64K chess computer solved it in twenty seconds.**

## Institutional Members

Addison-Wesley Publishing Company, *Reading, Massachusetts*  
 The Aerospace Corporation, *El Segundo, California*  
 Air Force Institute of Technology, *Wright-Patterson AFB, Ohio*  
 American Mathematical Society, *Providence, Rhode Island*  
 ArborText, Inc., *Ann Arbor, Michigan*  
 ASCII Corporation, *Tokyo, Japan*  
 Aston University, *Birmingham, England*  
 Brookhaven National Laboratory, *Upton, New York*  
 Brown University, *Providence, Rhode Island*  
 California Institute of Technology, *Pasadena, California*  
 Calvin College, *Grand Rapids, Michigan*  
 Carnegie Mellon University, *Pittsburgh, Pennsylvania*  
 Centre Inter-Régional de Calcul Électronique, CNRS, *Orsay, France*  
 City University of New York, *New York, New York*  
 College of St. Thomas, Computing Center, *St. Paul, Minnesota*  
 College of William & Mary, Department of Computer Science, *Williamsburg, Virginia*  
 COS Information, *Montreal, P. Q., Canada*  
 Data General Corporation, *Westboro, Massachusetts*  
 DECUS, L&T Special Interest Group, *Marlboro, Massachusetts*  
 Department of National Defence, *Ottawa, Ontario, Canada*  
 Digital Equipment Corporation, *Nashua, New Hampshire*

dit Company, Ltd., *Tokyo, Japan*  
 Edinboro University of Pennsylvania, *Edinboro, Pennsylvania*  
 Electricité de France, *Clamart, France*  
 Environmental Research Institute of Michigan, *Ann Arbor, Michigan*  
 European Southern Observatory, *Garching bei München, Federal Republic of Germany*  
 Fermi National Accelerator Laboratory, *Batavia, Illinois*  
 Försvarets Materielverk, *Stockholm, Sweden*  
 General Motors Research Laboratories, *Warren, Michigan*  
 Geophysical Company of Norway A/S, *Stavanger, Norway*  
 Grinnell College, Computer Services, *Grinnell, Iowa*  
 GTE Laboratories, *Waltham, Massachusetts*  
 Harvard University, Computer Services, *Cambridge, Massachusetts*  
 Hewlett-Packard Co., *Boise, Idaho*  
 Hobart & William Smith Colleges, *Geneva, New York*  
 Humboldt State University, *Arcata, California*  
 Hutchinson Community College, *Hutchinson, Kansas*  
 IBM Corporation, Scientific Center, *Palo Alto, California*  
 Illinois Institute of Technology, *Chicago, Illinois*  
 Imagen, *Santa Clara, California*  
 Informatika, *Hamburg, Federal Republic of Germany*  
 Institute for Advanced Study, *Princeton, New Jersey*  
 Institute for Defense Analyses, Communications Research Division, *Princeton, New Jersey*  
 Intevep S. A., *Caracas, Venezuela*  
 Iowa State University, *Ames, Iowa*  
 Istituto di Cibernetica, Università degli Studi, *Milan, Italy*  
 Kuwait Institute for Scientific Research, *Safat, Kuwait*  
 The Library of Congress, *Washington, DC*  
 Los Alamos National Laboratory, University of California, *Los Alamos, New Mexico*  
 Louisiana State University, *Baton Rouge, Louisiana*  
 Marquette University, Department of Mathematics, Statistics, and Computer Science, *Milwaukee, Wisconsin*  
 Massachusetts Institute of Technology, Artificial Intelligence Laboratory, *Cambridge, Massachusetts*  
 Massachusetts Institute of Technology, Information Services, *Cambridge, Massachusetts*  
 Mathematical Reviews, American Mathematical Society, *Ann Arbor, Michigan*  
 Max Planck Institut für Mathematik, *Bonn, Federal Republic of Germany*  
 Max Planck Institute Stuttgart, *Stuttgart, Federal Republic of Germany*  
 McGill University, *Montréal, Québec, Canada*  
 Michigan State University, Mathematics Department, *East Lansing, Michigan*  
 National Cancer Institute, *Frederick, Maryland*  
 National Center for Atmospheric Research, *Boulder, Colorado*  
 National Institutes of Health, *Bethesda, Maryland*  
 National Research Council Canada, Computation Centre, *Ottawa, Ontario, Canada*  
 National Semiconductor Corporation, *Santa Clara, California*  
 New Jersey Institute of Technology, *Newark, New Jersey*

- New York University, Academic Computing Facility, *New York, New York*
- Nippon Telegraph & Telephone Corporation, Software Laboratories, *Tokyo, Japan*
- Northeastern University, Academic Computing Services, *Boston, Massachusetts*
- Norwegian Pulp & Paper Research Institute, *Oslo, Norway*
- Online Computer Library Center, Inc. (OCLC), *Dublin, Ohio*
- Pennsylvania State University, Computation Center, *University Park, Pennsylvania*
- Personal TeX, Incorporated, *Mill Valley, California*
- Promis Systems Corporation, *Toronto, Ontario, Canada*
- Peter Isaacson Publications, *Victoria, Australia*
- Purdue University, *West Lafayette, Indiana*
- QMS, Inc, *Mobile, Alabama*
- Queens College, *Flushing, New York*
- Research Triangle Institute, *Research Triangle Park, North Carolina*
- RE/SPEC, Inc., *Rapid City, South Dakota*
- Rice University, Department of Computer Science, *Houston, Texas*
- Royal Marsden Hospital, *Surrey, England*
- Ruhr Universität Bochum, *Bochum, Federal Republic of Germany*
- Rutgers University, Hill Center, *Piscataway, New Jersey*
- St. Albans School, *Mount St. Alban, Washington, D.C.*
- Sandia National Laboratories, *Albuquerque, New Mexico*
- SAS Institute, *Cary, North Carolina*
- I. P. Sharp Associates, *Palo Alto, California*
- Smithsonian Astrophysical Observatory, Computation Facility, *Cambridge, Massachusetts*
- Software Research Associates, *Tokyo, Japan*
- Sony Corporation, *Atsugi, Japan*
- Space Telescope Science Institute, *Baltimore, Maryland*
- Springer-Verlag, *Heidelberg, Federal Republic of Germany*
- Stanford Linear Accelerator Center (SLAC), *Stanford, California*
- Stanford University, Computer Science Department, *Stanford, California*
- Stanford University, ITS Graphics & Computer Systems, *Stanford, California*
- State University of New York, Department of Computer Science, *Stony Brook, New York*
- Stratus Computer, Inc., *Marlboro, Massachusetts*
- Syracuse University, *Syracuse, New York*
- Talaris Systems, Inc., *San Diego, California*
- Texas A & M University, Computing Services Center, *College Station, Texas*
- Texas A & M University, Department of Computer Science, *College Station, Texas*
- Tribune TV Log, *Glens Falls, New York*
- TRW, Inc., *Redondo Beach, California*
- Tufts University, *Medford, Massachusetts*
- TV Guide, *Radnor, Pennsylvania*
- TYX Corporation, *Reston, Virginia*
- UNI-C, *Aarhus, Denmark*
- University College, *Cork, Ireland*
- University of Alabama, *Tuscaloosa, Alabama*
- University of British Columbia, Computing Centre, *Vancouver, British Columbia, Canada*
- University of British Columbia, Mathematics Department, *Vancouver, British Columbia, Canada*
- University of Calgary, *Calgary, Alberta, Canada*
- University of California, Division of Library Automation, *Oakland, California*
- University of California, Berkeley, Academic Computing Services, *Berkeley, California*
- University of California, Berkeley, Computer Science Division, *Berkeley, California*
- University of California, Irvine, Department of Mathematics, *Irvine, California*
- University of California, Irvine, Information & Computer Science, *Irvine, California*
- University of California, San Diego, *La Jolla, California*
- University of California, San Francisco, *San Francisco, California*
- University of Canterbury, *Christchurch, New Zealand*
- University of Chicago, Computation Center, *Chicago, Illinois*
- University of Chicago, Computer Science Department, *Chicago, Illinois*
- University of Crete, Institute of Computer Science, *Heraklio, Crete, Greece*
- University of Delaware, *Newark, Delaware*
- University of Exeter, Computer Unit, *Exeter, Devon, England*
- University of Glasgow, Department of Computing Science, *Glasgow, Scotland*
- University of Groningen, *Groningen, The Netherlands*

University of Illinois at Chicago,  
Computer Center, *Chicago, Illinois*  
University of Illinois at Urbana-Champaign, Computer Science  
Department, *Urbana, Illinois*  
University of Kansas, Academic  
Computing Services, *Lawrence, Kansas*  
University of Maryland, *College Park, Maryland*  
University of Massachusetts,  
*Amherst, Massachusetts*  
Université de Montréal, *Montréal, Québec, Canada*  
University of North Carolina,  
School of Public Health,  
*Chapel Hill, North Carolina*  
University of Oslo, Institute  
of Informatics, *Blindern, Oslo, Norway*  
University of Ottawa, *Ottawa, Ontario, Canada*

University of Salford, *Salford, England*  
University of Southern California,  
Information Sciences Institute,  
*Marina del Rey, California*  
University of Stockholm,  
Department of Mathematics,  
*Stockholm, Sweden*  
University of Texas at Austin,  
*Austin, Texas*  
University of Vermont, *Burlington, Vermont*  
University of Washington,  
Department of Computer Science,  
*Seattle, Washington*  
University of Western Australia,  
Regional Computing Centre,  
*Nedlands, Australia*  
University of Wisconsin, Academic  
Computing Center, *Madison, Wisconsin*  
Uppsala University, *Uppsala, Sweden*

Vanderbilt University, *Nashville, Tennessee*  
Vereinigte Aluminium-Werke AG,  
*Bonn, Federal Republic of Germany*  
Villanova University, *Villanova, Pennsylvania*  
Vrije Universiteit, *Amsterdam, The Netherlands*  
Washington State University,  
*Pullman, Washington*  
Widener University, Computing  
Services, *Chester, Pennsylvania*  
John Wiley & Sons, Incorporated,  
*New York, New York*  
Worcester Polytechnic Institute,  
*Worcester, Massachusetts*  
Yale University, Computer Center,  
*New Haven, Connecticut*  
Yale University, Department of  
Computer Science, *New Haven, Connecticut*

## TeX Typesetting Services

The American Mathematical Society can produce typeset pages from your DVI or source files. Features of our services include:

- **QUALITY** - We use an Autologic APS Micro-5 typesetter.
- **FONTS** - We offer AM, CM and Times Roman. Several more Autologic typefaces will be added in the near future.
- **LOW-COST** - We charge only \$5 per page for the first 100 pages; \$2.50 per page for additional pages.
- **SPEED** - Turnaround time is no more than one week for up to a 500 page job.
- **EXPERIENCE** - If you have a problem with a DVI or source file, we can usually solve it with our staff who are trained in TeX.
- **FULL-SERVICE** - We also offer keyboarding, camera work, printing and binding services.



For more information, or to schedule a job, please contact Regina Girouard

American Mathematical Society      (401) 272-9500  
PO Box 6248      800-556-7774  
Providence, RI 02940

### Request for Information

The TeX Users Group maintains a database and publishes a membership list containing information about the equipment on which TeX is (or will be) installed and about the applications for which TeX is used. This list is updated periodically and distributed to members with TUGboat, to permit them to identify others with similar interests. Thus, it is important that the information be complete and up-to-date.

Please answer the questions below, in particular those regarding the status of TeX and the hardware on which it runs. (Operating system information is particularly important in the case of IBM mainframes and VAX.) This hardware information is used to group members in the listings by computer and output device.

If accurate information has already been provided by another TUG member at your site, indicate that member's name and the same information will be repeated automatically under your name. If your current listing is correct, you need not answer these questions again. Your cooperation is appreciated.

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

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

- For foreign bank transfers

direct payment to the TeX Users Group,  
account #002-031375, at:

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

- General correspondence

about TUG should be addressed to:

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

|                                                                |
|----------------------------------------------------------------|
| Name: _____                                                    |
| Home [ ] Address: _____<br>[ ] _____<br>[ ] _____<br>[ ] _____ |

| QTY | ITEM                                                                                                                                                                                                                                                                                                                                         | AMOUNT |
|-----|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
|     | 1989 TUGboat Subscription/TUG Membership (Jan.-Dec.) - <b>North America</b><br>New (first-time): [ ] \$35.00 each<br>Renewal: [ ] \$45.00; [ ] \$35.00 - reduced rate if renewed before February 1, 1989                                                                                                                                     |        |
|     | 1989 TUGboat Subscription/TUG Membership (Jan.-Dec.) - <b>Outside North America</b><br>New (first-time): [ ] \$45.00 each<br>Renewal: [ ] \$50.00; [ ] \$45.00 - reduced rate if renewed before February 1, 1989                                                                                                                             |        |
|     | TUGboat back volumes      1980    1981    1982    1983    1984    1985    1986    1987    1988<br>Circle volume(s) desired:    vol. 1    vol. 2    vol. 3    vol. 4    vol. 5    vol. 6    vol. 7    vol. 8    vol. 9<br>Indiv. issues \$18.00 ea.      \$18      \$50      \$35      \$35      \$35      \$50      \$50      \$50      \$50 |        |

Air mail postage is included in the rates for all subscriptions and memberships outside North America.

Quantity discounts available on request.

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

### Membership List Information

Institution (if not part of address): \_\_\_\_\_

Date: \_\_\_\_\_

Title: \_\_\_\_\_

Status of TeX: [ ] Under consideration

Phone: \_\_\_\_\_

[ ] Being installed

Network address: \_\_\_\_\_

[ ] Up and running since: \_\_\_\_\_

[ ] Arpanet [ ] BITnet  
[ ] CSnet [ ] uucp  
[ ] JANET [ ] other \_\_\_\_\_

Approximate number of users: \_\_\_\_\_

Specific applications or reason for interest in TeX:

Version of TeX:

[ ] Pascal

[ ] C

[ ] other (describe) \_\_\_\_\_

My installation can offer the following software or technical support to TUG:

From whom obtained: \_\_\_\_\_

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

Hardware on which TeX is used:

| Computer(s) | Operating system(s) | Output device(s) |
|-------------|---------------------|------------------|
| _____       | _____               | _____            |
| _____       | _____               | _____            |
| _____       | _____               | _____            |

## Integrated Computer Software, Inc

### Stephan v. Bechtolsheim

**TEX Related Services:**

Macro package design  
 Specialized output routines  
 Training  
 Consulting

**X Window System Programming****POSTSCRIPT Related Services:**

Programming  
 Training

**Data Base Publishing****LATEX Related Services:**

Style file modifications  
 Training  
 Consulting

**NeWS Programming****TEXPS Software Package**  
 (see article in this issue)**Unix Related Programming**

*2119 Old Oak Drive  
 West Lafayette, IN 47906*

(317) 463 0162

**THE COMPLETE PC TYPESETTING SYSTEM**

**TEX \* METAFONT \* Previewer \* HP Laser Driver \* Dot Matrix Driver**

To earn your patronage, ScripTek, Inc., makes this limited introductory offer of its Complete System I:

- TEX with virtual memory (The TEXbook extra);
- METAFONT with virtual memory (The METAFONTbook extra);
- Screen previewer with three zoom levels;
- Laser printer driver for HP Laser Jet II or compatible;
- Dot matrix driver for IBM graphics printer or compatible;
- All 75 Computer Modern Fonts with source code;

Total package price \$125. Available for the IBM PC XT/AT or compatible. Requires 512K RAM and hard disk. For purchase orders only, CALL: (800) 383-5022. Mastercard and Visa accepted.

For more information, or to order by mail, WRITE: ScripTek, Inc.,  
 Sales Department, P.O. Box 5310, Kansas City, Missouri 64131.

## Public Domain T<sub>E</sub>X

The authorized and current versions T<sub>E</sub>X software are available from *Maria Code - Data Processing Services* by special arrangement with Stanford University and other contributing universities. The standard distribution tape contains the source of T<sub>E</sub>X and METAFONT, the macro libraries for A<sub>M</sub>S-T<sub>E</sub>X, L<sub>A</sub>T<sub>E</sub>X, SliT<sub>E</sub>X and HP T<sub>E</sub>X, sample device drivers for a Versetec and LN03 printers, documentation files, and many useful tools.

Since these are in the public domain, they may be used and copied without royalty concerns. They represent the official versions of T<sub>E</sub>X. A portion of your tape cost is used to support development at Stanford University.

If you have a DEC VAX/VMS, IBM CMS, IBM MVS or DEC TOPS operating system, you will want to order a special distribution tape which contains "ready-to-run" T<sub>E</sub>X and METAFONT. If you do not have one of these systems, you must perform a more involved installation which includes compiling the source with your Pascal compiler. Ready-to-run versions of T<sub>E</sub>X are available for other systems from various sources at various prices. You may want to examine these before ordering a standard distribution tape.

The font tapes contain GF files for the Computer Modern fonts. While it is possible to generate these files yourself, it will save you a lot of CPU time to get them on tape.

All systems are distributed on 9 track, 1600 bpi magnetic tapes. If both a distribution tape and a font tape are ordered, they may be combined on a single 2400' reel, space permitting.

Your order will be filled with the current versions of software and manuals at the time it is received. If you want a specific version, please indicate that on your order.

Please use the form on the next page for your order. Note that postage, except domestic book rate is based on the item weights in pounds. If you want to place your order by telephone, please call (408) 735-8006 between 9:00 am and 2:00 pm West Coast time. Do not call for technical assistance since no one there can help you.

We normally have a good stock of books and tapes, so your order can be filled promptly — usually within 48 hours.

Make checks payable to *Maria Code - Data Processing Services*. Export orders must have a check drawn on a US bank or use an International Money Order. Purchase orders are accepted. If you want to pay with Visa or MasterCard, include your number, card type, expiration date and signature.

## T<sub>E</sub>X Order Form

**T<sub>E</sub>X Distribution tapes:**

- Standard ASCII format
- Standard EBCDIC format
- Special VAX/VMS format Backup
- Special DEC 20/TOPS 20 Dumper format
- Special IBM VM/CMS format
- Special IBM MVS format

Tape prices: \$92.00 for first tape,  
\$72.00 for each additional tape.

**Font Library Tapes (GF files)**

- 300 dpi VAX/VMS format
- 300 dpi generic format
- IBM 3820/3812 MVS format
- IBM 3800 CMS format
- IBM 4250 CMS format
- IBM 3820/3812 CMS format

Total number of tapes \_\_\_\_\_

Postage: allow 2 lbs. for each tape

**Documents:**

|                                                                   | Price \$ | Weight | Quantity |
|-------------------------------------------------------------------|----------|--------|----------|
| T <sub>E</sub> Xbook (vol. A) softcover .....                     | 27.00    | 2      | _____    |
| T <sub>E</sub> X: The Program (vol. B) hardcover .....            | 37.00    | 4      | _____    |
| METAFONT book (vol. C) softcover .....                            | 22.00    | 2      | _____    |
| METAFONT the Program (vol. D) hardcover ...                       | 37.00    | 4      | _____    |
| Computer Modern Typefaces (vol. E) hardcover                      | 37.00    | 4      | _____    |
| I <sub>A</sub> T <sub>E</sub> X document preparation system ..... | 27.00    | 2      | _____    |
| WEB language * .....                                              | 12.00    | 1      | _____    |
| T <sub>E</sub> Xware * .....                                      | 10.00    | 1      | _____    |
| BibT <sub>E</sub> X * .....                                       | 10.00    | 1      | _____    |
| Torture Test for T <sub>E</sub> X * .....                         | 8.00     | 1      | _____    |
| Torture Test for METAFONT * .....                                 | 8.00     | 1      | _____    |

\* published by Stanford University

**Payment calculation:**

|                             |       |                           |       |
|-----------------------------|-------|---------------------------|-------|
| Number of tapes ordered     | _____ | Total price for tapes     | _____ |
| Number of documents ordered | _____ | Total price for documents | _____ |
| Add the 2 lines above       |       |                           |       |

Orders from within California: Add sales tax for your location. \_\_\_\_\_

**Shipping charges: (for domestic book rate, skip this section)**

Total weight of tapes and books \_\_\_\_\_ lbs.

|              |                          |                                       |                 |
|--------------|--------------------------|---------------------------------------|-----------------|
| Check<br>One | <input type="checkbox"/> | domestic priority mail                | rate \$1.50/lb. |
|              | <input type="checkbox"/> | air mail to Canada and Mexico:        | rate \$2.00/lb. |
|              | <input type="checkbox"/> | export surface mail (all countries):  | rate \$1.50/lb. |
|              | <input type="checkbox"/> | air mail to Europe, South America:    | rate \$5.00/lb. |
|              | <input type="checkbox"/> | air mail to Far East, Africa, Israel: | rate \$7.00/lb. |

Multiply total weight by shipping rate. Enter shipping charges: \_\_\_\_\_

**Total charges:** (add charges for materials, tax and shipping) \_\_\_\_\_

**Send to:** Maria Code, DP Services, 1371 Sydney Drive, Sunnyvale, CA 94087.  
Include your name, organization, address, and telephone number.

Are you or your organization a member of TUG? \_\_\_\_\_

K-Talk Introduces

# *MathEdit*

## Math Editing Made Easy- For the PC!

*MathEdit* makes it easy to create and edit math formulas for your TeX files. With its simple menu, **anyone** can immediately begin constructing complex formulas. And as you create your formulas, *MathEdit* guides you completely through them, so you can't go wrong.

- Sizes of math symbols are adjusted automatically to fit equations of varying size and complexity.
- A display window lets you see your formula as you create it.
- *MathEdit* takes full advantage of EGA and Hercules Graphics Plus RamFonts to give you excellent quality on-screen viewing.
- And your equation is output in WordPerfect format AND TeX, so it can be used in word processing as well as TeX typesetting.

*MathEdit* is \$149 and has a 30-day money back guarantee. For more information, please contact:



50 McMillen Avenue  
Columbus, Ohio 43201  
(614) 294-3535

# IS YOUR PC STILL TEXLESS?

*It doesn't have to be.*

*End its Texlessness with the Personal TeX, Inc. products below...  
including New PC TeX 2.93 and New PTI View.*

**PC TeX Version 2.93.** New! The latest version of TeX, includes support for EMS memory. **\$249**

**PTI View.** New product! Document previewer for EGA, VGA, CGA and Hercules graphics adapters. Scales on-the-fly. **\$149**

**PC TeX + PTI LASER + PTI VIEW.** TeX82, Version 2.93: professional formatting and typesetting results—for amateur prices. Includes INITTeX, LaTeX, AMS-Tex, VANILLA Macros, PCTeX and LaTeX manuals. Plus a PTI Laser device driver, to take full advantage of your laser printer. PLUS the PTI View screen previewer for on-screen previewing of your TeX documents and immediate editing. Top performance and low cost make this our most popular package. **\$499**

**PC TeX + PTI LASER.** As above, but without the PTI View screen previewer. **\$399**

**PC TeX + PCDOT + PTI View.** This package gives you all the TeX and PTI View benefits, together with our dot-matrix device driver for reliable, low cost printing. **\$399**

**PC TeX + PCDOT.** As above, without the PTI View screen previewer. **\$325**

**pcMF—METAFONT for the PC.** Lets you design fonts and create graphics. (Not for the novice.) Version 1.7a. **\$195**

**PTI LASER HP+ SERIES.** This device driver for the HP LaserJet Plus and Series II laser printers takes full advantage of the 512K resident memory. **\$195**

**PTI LASER POSTSCRIPT.** Device driver for PostScript printer; allows the resident fonts and graphic images to be used in TeX documents. **\$195**

**TABLES TO DIE FOR.** Lets you easily produce even the most complex tables—with professional typesetter refinements. **\$125**

**BITSTREAM Font Family.** An extensive library of over 50 type families, in any size you specify, with true typographic quality. Each family **\$195**

**NELSON BEEBE Device Drivers.** C source code for the do-it-yourself TeX hacker. **\$35**

**TeX Documentation Source.** 6 diskettes. **\$25**

**TeX Program Source.** 10 diskettes. **\$25**

Don't spend another day Texless. To order, just dial

**(415) 388-8853**



12 Madrona Avenue Mill Valley, CA 94941  
FAX: (415) 388-8865 VISA, MC accepted.

Requires: DOS 2.0 or later, 512K RAM, 10M hard disk. TeX is an American Mathematical Society TM. PC TeX is a Personal TeX, Inc. TM. Manufacturers' names are their TMs. Outside the USA, order through your local PC TeX distributor. Inquire about available distributorships and site licenses. This ad was produced using PC TeX and Bitstream fonts.

# TYPESETTING: JUST \$2.50 PER PAGE!

Send us your *TEX DVI* files and we will typeset your material at 2000 dpi on quality photographic paper — \$2.50 per page!

Choose from these available fonts: Computer Modern, Bitstream Fontware™, and any METAFONT fonts. (For each METAFONT font used other than Computer Modern, \$15 setup is charged. This ad was composed with PCTEX® and Bitstream Dutch (Times Roman) fonts, and printed on RC paper at 2000 dpi with the Chelgraph IBX typesetter.)

And the good news is: just \$2.50 per page, \$2.25 each for 100+ pages, \$2.00 each for 500+ pages! Laser proofs \$.50 per page. (\$25 minimum on all jobs.)

Call or write today for complete information, sample prints, and our order form. **TYPE 2000, 16 Madrona Avenue, Mill Valley, CA 94941. Phone 415/388-8873.**

---

TYPE  
2000

---

If you find yourself needing a TEXnician each time you want to create a format for a new document, or modify an existing one, then you need *TEXT1*.

*TEXT1* makes it easy for the TEX novice and expert alike to quickly modify the style of your chapters, contents, running head, margin notes, indices, etc. You can get *TEXT1* for a microcomputer for \$150.

*TEXT1* also has an International Phonetic font optionally available for \$100.

For more information, write to: or call:

*TEXT1* Distribution  
WSUCSC  
Pullman, WA 99164-1220

*TEXT1* Distribution  
509-335-0411



## THE SOLUTION...

$$\text{CTEX}^{\text{TM}} = \sum_{\text{QUALITY}} \left\{ \sqrt{\text{CWEB}} + \left( \frac{\text{TEX}^{2.95}}{\text{DOS}} \right) \right\}$$

**Available now, from Micro Publishing Systems Inc., CTEX, a fully TRIP-certified implementation of T<sub>E</sub>X ver. 2.95 for IBM PC and compatibles running MS DOS.**

### T<sub>E</sub>X Version 2.95 TODAY...

CTEX means never having to wait for your update! For instance, if T<sub>E</sub>X version 2.96 was announced tomorrow, we would provide you with an update virtually same day. Why wait for up to two years, as in some cases, for the most recent T<sub>E</sub>X release? With CTEX you will always be up to date!

### THE COMPLETE PICTURE

With the CTEX Plus package you will receive all you need to T<sub>E</sub>X your most demanding project. CTEX really *is* fast, (not just a fast sounding name)! T<sub>E</sub>XWRITE allows you to T<sub>E</sub>X, preview, or print, at the stroke of a key without leaving your document! T<sub>E</sub>XPRINT/PS and T<sub>E</sub>XPRINT/HP laser printer drivers are included for a savings of more than \$240.

### THE CTEX PLUS PACKAGE

|                               |                                       |                                    |                          |
|-------------------------------|---------------------------------------|------------------------------------|--------------------------|
| T <sub>E</sub> X Version 2.95 | T <sub>E</sub> XWRITE                 | T <sub>E</sub> XPRINT/HP           | T <sub>E</sub> XPRINT/PS |
| INIT <sub>E</sub> X           | LaT <sub>E</sub> X                    | AMST <sub>E</sub> X                | PLAIN T <sub>E</sub> X   |
| CTEX User's Guide             | The T <sub>E</sub> Xbook              | or LaT <sub>E</sub> X User's Guide | Binder/Slip Case         |
| 518 CM Fonts                  | Picture and SliT <sub>E</sub> X Fonts |                                    |                          |

**All for only .....\$495 US**

CTEX, T<sub>E</sub>XWRITE AND T<sub>E</sub>XPRINT drivers may be purchased separately. Add Maxview screen previewer to above for \$100.

For more information or to place an order,  
please contact:

**Micro Publishing Systems, Inc.,**  
**Suite 300-1120 Hamilton St.,**  
**Vancouver, B.C., V6B 2S2,**  
**Canada (604) 687-0354**

*Dealer and distributorship inquiries are welcome.*

T<sub>E</sub>X is a trademark of the American Mathematical Society. Product names are trademarks of their respective manufacturers.



# Math and Technical Book Publishers . . .

AMS-TEX  
Computer Modern fonts  
are now available!

If you are creating your files using T<sub>E</sub>X,  
Computer Composition Corporation can now  
offer the following services:

- Converting T<sub>E</sub>X DVI files to fully paginated typeset pages in either "AM" or "CM" fonts.
- Providing 300 dpi laser-printed page proofs which simulate the typeset page in "AM" fonts only.
- Keyboarding services from traditionally prepared manuscripts via the T<sub>E</sub>X processing system in either "AM" or "CM" fonts.
- Full camera work services, including halftones, line art, screens and full-page negatives or positives for your printer.

---

*Call or write for sample pages  
in either "AM" or "CM" fonts*

---

COMPUTER  
COMPOSITION  
CORPORATION

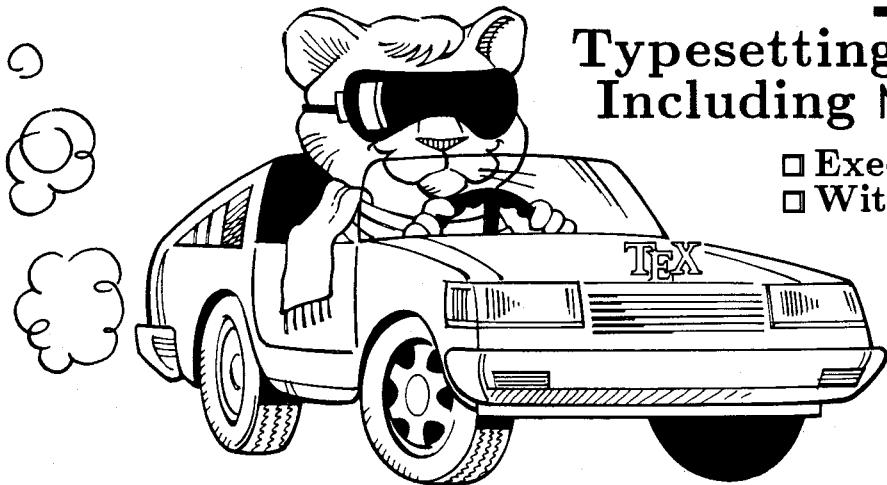
1401 WEST GIRARD AVE., MADISON HEIGHTS, MI 48071

(313) 545-4330

# TurboTEX

**Typesetting Software  
Including METAFONT**

- Executables \$150
- With source \$300



TurboTEX Release 2.0 software offers you a complete typesetting package based on the TeX 2.95 and METAFONT 1.7 standards: preloaded plain TeX, LATEX, INITEX, VIRTEX, and plain METAFONT interfaced to CGA/EGA/VGA/Hercules graphics; TRIP and TRAP certification; Computer Modern and LATEX fonts, and printer drivers for HP LaserJet Plus/Series II, PostScript, and dot-matrix printers. New features in the HP LaserJet driver put PCX or TIFF graphics files directly into your TeX documents. This wealth of software fills over 10 megabytes of diskettes, and runs on your IBM PC, UNIX, OS/2, or VAX/VMS system.

**■ Power Features:** TurboTEX brings big-machine performance to your small computer. TurboTEX breaks the 640K memory barrier under MS-DOS on the IBM PC with our virtual memory sub-system. You'll have the same sized TeX that runs on multi-megabyte mainframes, with plenty of memory for large documents, complicated formats, and demanding macro packages that break other TeX implementations. On

larger computers, TurboTEX runs up to 3 times faster in less memory than the Stanford Pascal distribution.

- Source code:** Order the TurboTEX source in portable C, and you will receive more disks with over 85,000 lines of generously commented TeX, TurboTEX, METAFONT, and printer driver source code, including: our WEB system in C; PASCHAL, our proprietary Pascal-to-C translator; and preloading, virtual memory, and graphics code. TurboTEX meets C portability standards like ANSI and K&R, and is robustly portable to a growing family of operating systems.
- TeX-FAX:** Connects TeX to the FAX revolution. Send perfect TeX output instantly, anywhere, without scanning. With TeX-FAX, any FAX machine in the world becomes your output device! Complete with PC board and software for \$395 (4800 bps) or \$795 (9600 bps).
- Desktop Publishing Interface Option:** Converts TeX output (such as equations or tables) for direct use in programs like Ventura Publisher and Pagemaker. (\$50 for PC).

**■ Availability & Requirements:** TurboTEX executables for IBM PC's include the User's Guide and require 640K and hard disk. Order source code (includes Programmer's Guide) for other machines. Source compiles with Microsoft C 5.0 or later on the PC; other systems need 1 MB memory and a C compiler supporting UNIX standard I/O. Media is 360K, 5-1/4" PC floppy disks; other formats at extra cost.

**■ No-risk trial offer:** Examine the documentation and run the PC TurboTEX for 10 days. If you are not satisfied, return the software for a 100% refund or credit. (Offer applies to PC executables only.)

#### Ordering TurboTEX

Order by phone, FAX, or mail.  
Terms: Check with order (free media and ground shipping in US), VISA, Mastercard (free media, shipping extra); Net 30 to well-rated firms and public agencies (shipping and media extra). Discounts available for quantities or resale.

Ask for the free, 50-page Buyer's Guide.

The Kinch Computer Company

PUBLISHERS OF TURBOTEX

501 South Meadow Street  
Ithaca, New York 14850  
Telephone (607) 273-0222  
FAX (607) 273-0484

# Kinch

**Publishing Companion translates  
WordPerfect**

to

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

It doesn't take a T<sub>E</sub>Xpert to use T<sub>E</sub>X.

With **Publishing Companion**, you can publish documents using T<sub>E</sub>X with **little or no T<sub>E</sub>X knowledge**. Your WordPerfect files are translated into T<sub>E</sub>X files, so anyone using this simple word processor can immediately begin typesetting their own documents!

And now, K-Talk introduces **Publishing Companion** version 2.0, which translates **WordPerfect 5.0** files into T<sub>E</sub>X.

Other word processors are supported using Mastersoft's WordForWord file conversion utility, \$70.

***Special Introductory Offer***

|                                 |              |
|---------------------------------|--------------|
| Retail Price .....              | \$249        |
| Academic Discount Price .....   | \$199        |
| <b>Introductory Price .....</b> | <b>\$179</b> |

This offer good until July 31, 1989. Upgrade from Publishing Companion v.1.XX is \$49.

For the power of T<sub>E</sub>X with the ease of a word processor, **Publishing Companion** is your "best friend" for desktop publishing.

For more information or to place an order, call or write:



50 McMillen Ave  
Columbus, Ohio 43201  
(614) 294-3535

**DESKTOP PUBLISHING HAS NEVER BEEN SIMPLER  
AND WILL NEVER BE THE SAME**

Introducing...

# T<sub>E</sub>X Made Easy

Using T<sub>E</sub>X With The Plain Macro Package

## Instruction

Since *T<sub>E</sub>X Made Easy* was created, author/instructor Daniel Zirin has taught hundreds of new T<sub>E</sub>X users with this 10 hour seminar format class. Many of the participants were unsophisticated computer users like administrative aids and business people. Mr. Zirin has consistently received high marks in presenting beginning T<sub>E</sub>X by *T<sub>E</sub>X Made Easy* students.

Now you can bring *T<sub>E</sub>X Made Easy* to your place of business at affordable rates. For only \$750\* per day (plus expenses), Mr. Zirin will present *T<sub>E</sub>X Made Easy* over a period of two to five days at your office (unlimited class size). To schedule a two day seminar in most U.S. cities would cost an average \$2500.

Class Format: 10 hours (seminar)

\$750 per day (2 day min.)

\*Air, hotel, trans., meals,  
*T<sub>E</sub>X Made Easy* manuals,  
and *The T<sub>E</sub>Xbook* all extra.

Offered by DECUS in Atlanta on May 7th.  
Call DECUS at (508) 480-3307 for details.

## Book

*T<sub>E</sub>X Made Easy*, the book, is written for people interested in a quick simple approach to learning how to start using PlainT<sub>E</sub>X. Written for the business professional, *T<sub>E</sub>X Made Easy* has been given high praise by those considered "computer shy".

### Topics Covered

- Creating T<sub>E</sub>X files.
- Defining page layout and margins.
- Accenting and overstriking characters.
- Using different fonts.
- How to use the T<sub>E</sub>X program.
- Specializing the headline and footline for each page.
- Generating footnotes.
- Setting a tabular environment.
- Using alignment to make tables.
- Dealing with figures and insertions.
- Automatic enumeration of paragraphs, equations, references, etc...
- Using math and equation modes.
- Creating simple T<sub>E</sub>X macros.
- A sample macro package to generate an automatic table of contents.

Price: \$15 for each 90 page book.

\$3 shipping U.S., \$6 all others.  
(Call for special discounts)

Zar Limited (818) 794-1224  
P. O. Box 372, Pasadena, CA 91102, USA

# The JOY of TEX



## A Gourmet Guide to Typesetting with the *AMS-TEX* macro package

M. D. SPIVAK, Ph.D.

*The Joy of TEX* is the user-friendly user's guide for *AMS-TEX*, an extension of *TeX*, Donald Knuth's revolutionary program for typesetting technical material. *AMS-TEX* was designed to simplify the input of mathematical material in particular, and to format the output according to any of various preset style specifications.

There are two primary features of the *TeX* system: it is a computer system for typesetting technical text, especially text containing a great deal of mathematics; and it is a system for producing beautiful text, comparable to the work of the finest printers.

Most importantly, *TeX*'s capabilities are not available only to *TeX*erts. While mathematicians and experienced technical typists will find that *TeX* allows them to specify mathematical formulas with great

accuracy and still have control over the finished product, even novice technical typists will find the manual easy to use in helping them produce beautiful technical *TeX*t.

This book is designed as a user's guide to the *AMS-TEX* macro package and details many features of this extremely useful text processing package. Parts 1 and 2, entitled "Starters" and "Main Courses," teach the reader how to typeset most normally encountered text and mathematics. "Sauces and Pickles," the third section, treats more exotic problems and includes a 60-page dictionary of special *TeX*niques.

Exercises sprinkled generously through each chapter encourage the reader to sit down at a terminal and learn through experimentation. Appendixes list summaries of frequently used and more esoteric symbols as well as answers to the exercises.



ISBN 0-8218-2999-8, LC 85-7506  
290 pages (softcover), 1986  
AMS Indiv. Memb. \$26, AMS Inst.  
Memb. \$30, List price \$33  
To order specify JOYT/T

Shipping/Handling: 1st book \$2, each  
add'l \$1, max. \$25; by air, 1st book  
\$5, each add'l \$3, max. \$100

**PREPAYMENT REQUIRED.** Order from  
American Mathematical Society  
P. O. Box 1571  
Annex Station  
Providence, RI 01901-1571  
or call 800-556-7774 to use VISA or MasterCard.  
Prices subject to change.



*We're VMS TeX specialists. Our products fit. They get you started quickly, and are comfortable and efficient to use, so you can concentrate on your design and writing, instead of struggling with your tools.*

### *Fit for VMS*

No surprises here. Our VMS TeX and companion device interfaces all use standard DCL command processing, file handling, and error reporting. Device interfaces mimic the familiar VMS Print command, but let you control page arrangement, placement, and scaling. VMSINSTAL installation and configuration is fast and familiar.

### *Fit for Professional Use*

You won't have to babysit our products. The software can be tailored to match your conventions and pattern of use. It reports intelligibly when errors occur. Printed documentation and additions to the VMS Help facility are as carefully designed as the software itself. The software operates efficiently in normal use, and accommodates extreme situations gracefully.

### *Fit to Print*

Take full advantage of your output device. All the device interfaces provide access to device-specific fonts and native graphics capabilities. Careful font handling produces output dependably, even when device memory is limited. A font management utility lets you merge commercial fonts from Bitstream, Compugraphic, DEC, HP, and other vendors into your standard TeX font set.

### *Supported*

You can turn to us for help. Our customer support offers prompt assistance, and can provide direct contact with our software engineers.

### **HP LaserJet**

Support for the current suite of LaserJet printers: the LaserJet II, the most economical printer that can effectively support TeX; the LaserJet IID, which adds duplexing at a small increment in price; up to the LaserJet 2000, for high-volume use.

### **DEC LN03**

Support for the LN03 and LN03 Plus, DEC's low-volume laser printers. A specially tuned font set matches their imaging characteristics.

### **Compugraphic**

Support for the 8400 and 8600 phototypesetters. You get true typeset quality, production volumes, and access to the full Compugraphic font library.

### **IMAGEN**

Support for the full line of IMAGEN printers: The IMAGEN-2308 and -3308s, flexible low-volume printers with the ability to emulate a wide variety of other output devices; the IMAGEN-4324, capable of sustained 24 page/minute output; the IMAGEN-XX20 series, with versatile paper handling, including duplexing and signature printing on paper up to ledger size.

### **Kodak Ektaprint**

The Kodak Ektaprint 1392 laser printer produces duplexed and stapled output at 90 pages per minute. It's there if you need it.

*For more information, please call.*

## **Northlake Software**

812 SW Washington, Suite 1100  
Portland, Oregon 97205-3215  
USA

503-228-3383 Fax 503-228-5662

TeX is a trademark of the American Mathematical Society. DEC and VMS are trademarks of Digital Equipment Corporation. IMAGEN is a registered trademark of IMAGEN Corporation. Ektaprint 1392 is a registered trademark of Eastman Kodak Company. Tektronix is a registered trademark of Tektronix, Incorporated.

# We have it all.

ArborText offers more software for your money, on more computers, than any other company. We have it all:  $\text{\TeX}$ ,  $\text{\TeX}$  Preview, DVILASER, phototypesetter drivers on PCs, Apollos, Suns, VAXstations, and IBM and DEC mainframes.

$\mu\text{\TeX}$ , for example, ArborText's  $\text{\TeX}$  for IBM PC, PS/2 and compatibles, has more *included* features:

- MAPPIT*— converts Standard Generalized Markup Language to  $\text{\TeX}$ .
- ArborText's knowledgeable customer support.
- Integrated previewer which you can use while  $\mu\text{\TeX}$  is still formatting.

- Ini $\text{\TeX}$  for creating preloaded macro packages.
- $\text{\LaTeX}$  document preparation macro package.
- AM and CM font .tfm files.
- Fast and easy installation script which saves disk space by checking for redundant files.
- Latest version of  $\text{\TeX}$ , 2.9.3.
- Consistent directory naming.
- Corrections to all known bugs in Addison-Wesley's Micro $\text{\TeX}$ , 1.5A.

## We have it all. We're ArborText.

The following are trademarks of their respective companies:  $\text{\TeX}$  of American Mathematical Society;  $\mu\text{\TeX}$  of ArborText, Inc.; IBM PC and PS/2 of International Business Machines Corp; Micro $\text{\TeX}$  of Addison-Wesley, Corp. Technology developed by David Fuchs.



ARBORTEXT INC.

535 West William Street Suite 300 Ann Arbor, MI 48103 (313) 996-3566 FAX (313) 996-3573

## “Literate Programming” for the Macintosh!

*MacWEB makes your Mac programming easier and more beautiful*

### The same powerful programming tool Knuth used to write **TEX**

- 100% Knuth compatible – based on the official WEB source code.
- Lets you develop programs top-down, using stepwise refinement.
- Your design decisions are preserved *with the code*, where you can keep them up to date as you make changes.
- Actually makes it fun to document your programs!
- Generates both **TEX** documentation and compilable Pascal source code from the same easy-to-write WEB input.
- Automatically pretty-prints your Pascal code with typeset quality, using the full range of **TEX**'s capabilities.
- Automatically generates a table of contents and index, including a full cross-reference of all your program's identifiers and all your top-down code refinements.

### Much more than just the public-domain system

- Offers a choice of pretty-printing styles, including Knuth's original style, “Apple” style, “LightSpeed” style, and more. Or define your own custom style.
- Generates more readable Pascal source files – optionally preserve identifiers, constants, and comments as you wrote them, etc. Use with MPW's *PasMat* tool to make nicely formatted Pascal files for debugging.
- Extensive documentation includes tutorial on how to use WEB effectively, with a special section of hints and trouble-shooting tips.
- 12 example WEB programs included. Each one comes with makefile and everything else you need to compile it or print it with **TEX**.

### Customized for the Macintosh and MPW

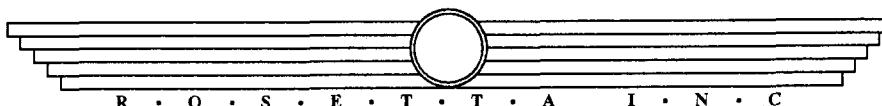
- Fully compatible with MPW – *tangle* and *weave* run as tools under the MPW Shell, and take standard command-line arguments. *Commando* interface, too.
- Handles the complete syntax of MPW Pascal, including Object Pascal.
- Put *tangle* and *weave* commands in your makefiles, and rebuild both programs and documentation automatically.
- Fully compatible with Textures.
- Customized “webmac” for best results on Apple LaserWriter. (Standard “webmac” also included, of course.)
- Fast! Tangles “tangle.web” in less than a minute on a Mac SE, weaves it in less than three (producing 45 kb of Pascal source code, or the **TEX** for a 75 page document).

### Order today!

Complete WEB system includes manual and 2 disks containing MPW tools, installer, examples, and the full public-domain WEB source code. Requires a Mac Plus or better, MPW 3.0, MPW Pascal, and your favorite implementation of **TEX** (on your Mac or any other convenient computer).

**ONLY \$79.95** with free shipping by UPS Second Day Air.  
C.O.D., check, or money order.  
Educational discounts and site licenses are available.

Rosetta, Inc.  
2502 Robinhood  
Houston, TX 77005  
(713) 528-8350



R • O • S • E • T • T • A • I • N • C

## TEX USERS GROUP

**HAVE YOU EVER WISHED YOU KNEW HOW TO USE THE  
FULL POTENTIAL OF *TeX* OR *LATEX* PROPERLY?**

**Can't get away to take a course? Let us help you!**

**Here's your opportunity to learn more about *TeX*.**

Since 1983 our instructors have taught nearly 50 weeks of classes at all levels of *TeX*, to include the *LATEX* and *AMS-TeX* macro packages, plus *METAFONT*, at a various university computing facilities throughout the U.S., Canada and Europe. In addition, more than 60 weeks of courses were conducted at several of the national scientific research labs and other organizations using *TeX* to publish research. Some of the courses we offer are:

- *TeX*: all levels, macro writing, output routines, wizard
- *LATEX*: all levels, style files
- *METAFONT*
- PostScript

Instruction can be tailored to the level of knowledge and experience of participants as well as the uses to which *TeX* is being put and is offered on all mainframe computers, PCs, Macs, etc.

Our fee of \$5,000 for a 5-day course covers everything: instructor's fee, travel and living expenses, course texts and instructor handouts—everything! (Shorter courses are priced accordingly.) Proceeds from these courses go to support other activites and services offered by TUG.

If you would like detailed course descriptions or need additional information, contact Charlotte Laurendeau or Ray Goucher at the TUG office. Phone: (401) 751-7760.

### Index of Advertisers

|          |                                    |
|----------|------------------------------------|
| 124, 138 | American Mathematical Society      |
| 140      | ArborText                          |
| cover 3  | Blue Sky Research                  |
| 134      | Computer Composition               |
| 127-128  | DP Services                        |
| 126      | Integrated Computer Software, Inc. |
| 129, 136 | K-Talk Communications              |
| 135      | Kinch Computer Company             |
| 142      | Micro Programs, Inc.               |
| 133      | Micro Publishing Systems, Inc.     |
| 139      | Northlake Software                 |
| 130      | Personal <i>TeX</i> Inc.           |
| 141      | Rosetta, Inc.                      |
| 126      | ScripTek, Inc.                     |
| 142      | <i>TeX</i> Users Group             |
| 131      | Type 2000                          |
| 132      | Washington State University        |
| 137      | Zar Limited                        |

## THE WRITE STUFF, TEXNICALLY SPEAKING

### *TeX* is the write stuff

*TeX* is the powerful publishing system that is guaranteed to make any document you write easier to read... and guaranteed to make that document say the right stuff about you!

*Micro Programs, Inc.* is your source for *TeX* and related ArborText products for IBM PC and Sun workstations.

Call Bob Harris on (516) 921-1351 and get the name of the dealer nearest you.

**MICRO PROGRAMS, INC.  
251 JACKSON AVENUE  
SYOSSET  
NY 11791**

# **COMPUTERS & TYPESETTING:**

## **Errata and Changes**

As of 20 February 1989

This document contains all known errata and changes to *Computers & Typesetting*, Volumes A-E, as compiled by the *TeX* Project at Stanford, from 16 June 1987 through the date shown above. An up-to-date log of changes is available online to users with Internet access. Changes are in the files <*tex.doc*>*ERRATA.ONE* (through 22 August 1984), <*tex.doc*>*ERRATA.TWO* (through 6 January 1986), <*tex.doc*>*ERRATA.THR* (through 15 June 1987), and <*tex.doc*>*ERRATA.TeX* (the most recent changes). All errata files are on the Score system at Stanford (*@SCORE.Stanford.Edu*).

The full errata list is published only once a year, and distributed with the first issue of TUGboat. Supplements appear as necessary in subsequent TUGboat issues. The date of the last entry in each section of this document is listed in the contents, below, so that it is not necessary to check the detail to see whether anything has been added.

### CONTENTS

|                                                                 |    |
|-----------------------------------------------------------------|----|
| Colophon for <i>Computers &amp; Typesetting</i>                 | 2  |
| Bugs in <i>Computers &amp; Typesetting</i>                      |    |
| Volume A: <i>The TeXbook</i> (4 January 1989)                   | 3  |
| Volume B: <i>TeX: The Program</i> (20 February 1989)            | 4  |
| Volume C: <i>The METAFONTbook</i> (23 December 1988)            | 10 |
| Volume D: <i>METAFONT: The Program</i><br>(14 December 1988)    | 11 |
| Volume E: <i>Computer Modern Typefaces</i><br>(23 January 1989) | 12 |
| Changes to the programs and fonts                               |    |
| <i>TeX</i> (ver. 2.97, 23 January 89)                           | 15 |
| <i>METAFONT</i> (ver. 1.7, 14 December 1988)                    | 22 |
| Computer Modern fonts (23 January 1989)                         | 25 |

Distributed with TUGboat Volume 10 (1989), No. 1. Published by

**TeX Users Group**

P. O. Box 9506

Providence, R.I. 02940-9506, U.S.A.

### **Colophon for Computers & Typesetting**

The five volumes of *Computers & Typesetting* were composed by *TeX* (*TeX82*) using the Computer Modern (CM85) fonts as produced by *METAFONT* (*METAFONT84*). Thus, the books themselves describe exactly how they were prepared for printing. Camera-ready copy was set on an Autologic APS Micro-5 typesetter at Stanford University from font images resident on the host computer and shipped to the typesetter a character at a time, as needed.

The proof-style illustrations in Volume E were also set on the Micro-5, each figure comprising two images that were combined photographically with the text material after the images of the character shapes had been screened. The "color separation" to produce those proofs was done by a program written for that purpose.

All the copy on the cover, spine, and book jackets was also typeset by the APS, using Computer Modern fonts, except for the ISBN number.

The books were printed on Finch Opaque, basis 50 lb. acid-free paper, which has a life expectancy of several hundred years. The hardcover edition was printed and bound by Halliday Lithograph Corp., Hanover, Massachusetts, as were the spiral-bound editions of *The TeXbook* and *The METAFONTbook*.

### **Corrections to earlier editions**

Corrections and changes to the AMS/Digital Press edition of the *TeX* and *METAFONT* manual (December 1979) and to the *TeX78* and *METAFONT79* programs are described in the booklet *TeX and METAFONT: Errata and Changes* dated September 1983 (originally distributed with *TUGboat* Volume 4, No. 2). This document also contains a comparison of *TeX78* (formerly known as *TeX80*) with *TeX82*.

Errata to editions of *The TeXbook* published prior to 1986 are described in *The TeXbook: Errata and Changes* dated February 1986 (originally distributed with *TUGboat* Volume 7, No. 1).

The booklet *Computers & Typesetting: Errata and Changes* dated 15 June 1987 contains errata through that date as well as listings of changes to *TeX.WEB*, *MF.WEB* and to the *METAFONT* sources of the Computer Modern fonts from the documentation files *TeX82.BUG*, *MF84.BUG* and *CM85.BUG* for the corresponding period.

These documents are available from TUG; for information on how to obtain copies, write to TUG at the address on the front cover.

**Bugs in Computers & Typesetting**

20 February 1989

This is a list of all corrections made to *Computers & Typesetting*, Volumes A–E since 15 June 1987. Corrections to the softcover version of *The TeXbook* are the same as corrections to Volume A. Corrections to the softcover version of *The METAFONTbook* are the same as corrections to Volume C. Some of these corrections have already been made in reprintings of the books. Some of these corrections affect the indexes and mini-indexes of Volumes B and D in ways not shown here. Corrections prior to 15 June 1987 were published by the *TeX* Users Group and are available from the TUG office.

|                                                                                                                                                                                                                                                                                   |            |
|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|------------|
| Page A159, line 22                                                                                                                                                                                                                                                                | (2/15/88)  |
| 'nolimits' if the normal \displaylimits convention has been overridden; a Rad                                                                                                                                                                                                     |            |
| Page A213, lines 34–35                                                                                                                                                                                                                                                            | (12/23/87) |
| text will be a single control sequence token, defined to be like \relax if its meaning is currently undefined.                                                                                                                                                                    |            |
| Page A299, line 30                                                                                                                                                                                                                                                                | (7/6/88)   |
| Fatal format file error; I'm stymied.                                                                                                                                                                                                                                             |            |
| Page A326, line 12                                                                                                                                                                                                                                                                | (12/12/87) |
| its natural width. The \hbox version also invokes \everymath.                                                                                                                                                                                                                     |            |
| Page A359, line 2                                                                                                                                                                                                                                                                 | (11/6/88)  |
| \mathchardef\ldotp="613A\mathchardef\cdotp="6201\mathchardef\colon="603A                                                                                                                                                                                                          |            |
| Page A359, lines 35–38                                                                                                                                                                                                                                                            | (5/24/88)  |
| \def\updownarrow{\delimter"326C33F } \def\arrowvert{\delimter"033C000 } \def\Updownarrow{\delimter"326D377 } \def\Arrowvert{\delimter"033D000 } \def\vert{\delimter"026A30C } \def\Vert{\delimter"026B30D } \def\backslash{\delimter"026E30F } \def\bracevert{\delimter"033E000 } |            |
| Page A364, line 35                                                                                                                                                                                                                                                                | (11/6/88)  |
| \def\fmtname{plain}\def\fmtversion{2.94} % identifies the current format                                                                                                                                                                                                          |            |
| Page A379, line 15                                                                                                                                                                                                                                                                | (10/12/87) |
| \def\deleterightmost#1{\edef#1{\expandafter\xyzzz#1\xyzzz}}                                                                                                                                                                                                                       |            |
| Page A383, lines 7–15 from the bottom                                                                                                                                                                                                                                             | (1/4/89)   |

```
209 strings out of 1685
1659 string characters out of 17636
27618 words of memory out of 52821
1172 multiletter control sequences out of 2500
```

Consequently there was plenty of room for more macros:  $52821 - 27618 = 25203$  unused cells of main memory,  $2500 - 1172 = 1328$  of name memory,  $1685 - 209 = 1476$  of string memory, and  $17636 - 1659 = 15977$  of character memory. But a fairly large *TeX* was being used, and only the macros of Appendices B and E were loaded; in other circumstances it might have been necessary to conserve space.

---

Page A454, lines 23–29

(8/13/87)

 If a suitable starting letter is found, let it be in font  $f$ . Hyphenation is abandoned unless the `\hyphenchar` of  $f$  is between 0 and 255, and unless a character of that number exists in the font. If this test is passed, TeX continues to scan forward until coming to something that's not one of the following three "admissible items": (1) a character in font  $f$  whose `\lccode` is nonzero; (2) a ligature formed entirely from characters of type (1); (3) an implicit kern. The first inadmissible item terminates this part of the process; the trial word consists of all the letters found in admissible items. Notice that all of these letters are in font  $f$ .

---

Page A458, left column, line 19

(2/15/88)

\I ( \| ), 146–147, 171, 361, 435, 438.

---

Page A462, left column, line 7

(10/9/87)

152, 178, 360.

---

Page A463, left column

(4/17/88)

\*\day, 273, 349, 406.

---

Page A464, left column, under Displays

(12/8/88)

non-centered, 186, 326, 375–376, 420–421.

---

Page A465, entry for \everymath

(12/12/87)

[Include also a reference to page 326.]

---

Page A465, right column

(7/6/88)

Fatal format file error, 299.

---

Page A473, entry for 'page builder'

(8/13/87)

when exercised, 122, 280–283, 286–287.

---

Page A474, left column

(12/27/88)

\*\parshape, 101–102, 214, 271, 277, 283,

---

Page A480, right column

(2/15/88)

\vdots ( : ), 177, 359.

---

Page A481, right column

(7/3/87)

\z@, 347, 348.\z@skip, 347, 348.

---

Page B2, line 32

(2/20/89)

define banner ≡ 'This is TeX, Version 2.97' { printed when TeX starts }

---

Page B38, lines 7–9 from the bottom

(11/6/88)

[Delete this paragraph; it is being moved to page B214.]

---

Page B38, line 5 from the bottom (12/14/88)

**begin if** *log-opened* **then** *selector*  $\leftarrow$  *term-and-log*

---

Page B39, line 5 (12/14/88)

**if** *log-opened* **then** *error*;

---

Page B52, line 5 (8/13/87)

cannot be done, i.e., if *hi\\_mem\\_min* = *lo\\_mem\\_max* + 1, we have to quit.

---

Page B54, lines 34–35 (7/9/88)

**begin if** *hi\\_mem\\_min* – *lo\\_mem\\_max*  $\geq$  1998 **then** *t*  $\leftarrow$  *lo\\_mem\\_max* + 1000  
**else** *t*  $\leftarrow$  *lo\\_mem\\_max* + 1 + (*hi\\_mem\\_min* – *lo\\_mem\\_max*) **div** 2; { *lo\\_mem\\_max* + 2  $\leq$  *t*  $<$  *hi\\_mem\\_min* }

---

Page B108, new line after line 8 (5/24/88)

*d*: *integer*; { number of characters in incomplete current string }

---

Page B108, lines 31–33 (5/24/88)

```

str_room(l); d  $\leftarrow$  cur_length;  

while pool_ptr  $>$  str_start[str_ptr] do  

  begin decr(pool_ptr); str_pool[pool_ptr + l]  $\leftarrow$  str_pool[pool_ptr];  

    end; { move current string up to make room for another }  

for k  $\leftarrow$  j to j + l - 1 do append_char(buffer[k]);  

  text(p)  $\leftarrow$  make_string; pool_ptr  $\leftarrow$  pool_ptr + d;
```

---

Page B115, line 12 (4/28/88)

*group\_code* = 0 .. *max\_group\_code*; { *save\_level* for a level boundary }

---

Page B141, line 19 (4/28/88)

*par\_token*: *halfword*; { token representing ‘\par’ }

---

Page B150, line 24 (4/28/88)

**358.** The present point in the program is reached only when the *expand* routine has inserted

---

Page B151, mini-index (4/28/88)

Delete the entry for ‘no-expand’; replace it by:

*expand*: **procedure**, §366.

---

Page B154, lines 25, 29, 34 respectively (9/20/87)

```

cvl_backup, radix_backup, co_backup: small_number; { to save cur_val_level, etc. }  

co_backup  $\leftarrow$  cur_order; backup_backup  $\leftarrow$  link(backup_head);  

cur_order  $\leftarrow$  co_backup; link(backup_head)  $\leftarrow$  backup_backup;
```

---

Page B155, new entry for mini-index (9/20/87)

*cur\_order*: *glue\_ord*, §447.

---

Page B156, line 28 (12/23/87)

**begin** *eq\_define(cur\_cs, relax, 256)*;

---

Page B157, mini-index (12/23/87)

Delete the entries for 'eqtb' and 'frozen\_relax'; replace them by the following:  
*eq\_define: procedure*, §227.  
*relax = 0*, §207.

---

Page B162, lines 12–14 (4/30/88)

```
repeat link(temp_head) ← null;
if (info(r) > match_token + 127) ∨ (info(r) < match_token) then s ← null
else begin match_chr ← info(r) – match_token; s ← link(r); r ← s; p ← temp_head; m ← 0;
```

---

Page B177, bottom line before mini-index (7/13/88)

```
cur_val ← 0; cur_val_level ← int_val; radix ← 0; cur_order ← 0;
```

---

Page B181, line 31 (4/28/88)

[Change 'x units per sp' to 'x sp per unit'! This change also should be made on line 1 of page B183 and line –8 of page B590.]

---

Page B188, line 8 (5/25/88)

```
function str_toks(b : pool_pointer): pointer; { changes the string str_pool[b .. pool_ptr] to a token list }
```

---

Page B188, line 13 (5/25/88)

```
begin str_room(1); p ← temp_head; link(p) ← null; k ← b;
```

---

Page B188, line 20 (5/25/88)

```
pool_ptr ← b; str_toks ← p;
```

---

Page B188, new line after line 28 (5/25/88)

```
b: pool_pointer; { base of temporary string }
```

---

Page B188, line 31 (5/25/88)

```
else begin old_setting ← selector; selector ← new_string; b ← pool_ptr;
```

---

Page B188, line 41 (5/25/88)

```
selector ← old_setting; the_toks ← str_toks(b);
```

---

Page B190, lines 16–18 (5/25/88)

```
b: pool_pointer; { base of temporary string }
begin c ← cur_chr; { Scan the argument for command c 471 };
old_setting ← selector; selector ← new_string; b ← pool_ptr; { Print the result of command c 472 };
selector ← old_setting; link(garbage) ← str_toks(b); ins_list(link(temp_head));
```

---

Page B210, line 36 (5/25/88)

```
begin if (pool_ptr + name_length > pool_size) ∨ (str_ptr = max_strings) ∨ (cur_length > 0) then
```

---

Page B211, new line of code before the mini-index (12/14/88)

```
log_opened: boolean; { has the transcript file been opened? }
```

---

Page B212, line 5 (12/14/88)

```
job_name ← 0; name_in_progress ← false; log_opened ← false;
```

---

Page B213, line 24

(12/14/88)

```
log_name ← a_make_name_string(log_file); selector ← log_only; log_opened ← true;
```

---

Page B214, lines 2 and 3

(12/14/88)

messages or even to *show\_context*. The *prompt\_file\_name* routine can result in a *fatal\_error*, but the *error* routine will not be invoked because *log\_opened* will be false.

The normal idea of *batch\_mode* is that nothing at all should be written on the terminal. However, in the unusual case that no log file could be opened, we make an exception and allow an explanatory message to be seen.

---

Page B214, lines 7–11 reduce to a single line

(12/14/88)

```
begin selector ← term_only;
```

---

Page B224, second-last line

(4/28/87)

```
done: if file_opened then b_close(tfm_file);  
read_font_info ← g;
```

---

Page B229, lines 6–8

(11/17/87)

than  $2^{27}$ . If  $z < 2^{23}$ , the individual multiplications  $b \cdot z$ ,  $c \cdot z$ ,  $d \cdot z$  cannot overflow; otherwise we will divide  $z$  by 2, 4, 8, or 16, to obtain a multiplier less than  $2^{23}$ , and we can compensate for this later. If  $z$  has thereby been replaced by  $z' = z/2^e$ , let  $\beta = 2^{4-e}$ ; we shall compute

---

Page B229, lines 11–12

(11/17/87)

if  $a = 0$ , or the same quantity minus  $\alpha = 2^{4+e}z'$  if  $a = 255$ . This calculation must be done exactly, in order to guarantee portability of T<sub>E</sub>X between computers.

---

Page B230, lines 2–5

(11/17/87)

```
begin alpha ← 16;  
while z ≥ '40000000 do  
  begin z ← z div 2; alpha ← alpha + alpha; end;  
  beta ← 256 div alpha; alpha ← alpha * z;
```

---

Page B245, new entry for mini-index

(8/7/87)

```
cur_s: integer, §616.
```

---

Page B254, line 29

(8/7/87)

```
cur_s: integer; { current depth of output box nesting, initially -1 }
```

---

Page B254, line 31

(8/7/87)

[Remove the statement '*cur\_s* ← -1;' and put it on page B244 at the end of line 31.]

---

Page B259, line 13

(11/9/87)

```
begin rule_wd ← rule_wd + 10; { compensate for floating-point rounding }  
edge ← cur_h + rule_wd; lx ← 0; { Let cur_h be the position of the first box, and set }
```

---

Page B259, line 17

(11/9/87)

```
cur_h ← edge - 10; goto next_p;
```

Page B263, line 21

(11/9/87)

```
begin rule_ht  $\leftarrow$  rule_ht + 10; { compensate for floating-point rounding }
edge  $\leftarrow$  cur_v + rule_ht; lx  $\leftarrow$  0; { Let cur_v be the position of the first box, and set }
```

Page B263, line 25

(11/9/87)

```
cur_v  $\leftarrow$  edge - 10; goto next_p;
```

Page B266, line 8

(8/7/87)

```
dvi_out(eop); incr(total_pages); cur_s  $\leftarrow$  -1;
```

Page B266, new code between lines 31 and 32

(8/7/87)

```
while cur_s  $>$  -1 do
  begin if cur_s  $>$  0 then dvi_out(pop)
  else begin dvi_out(eop); incr(total_pages)
    end;
  decr(cur_s);
end;
```

Page B285, line 21

(4/28/88)

is subsidiary to the *nucleus* field of some node; the dot is replaced by ‘\_’ or ‘^’ or ‘/’ or ‘\’ if *p* is

Page B338, second-last line

(8/19/87)

```
q  $\leftarrow$  link(head); s  $\leftarrow$  head;
```

Page B339, line 4

(8/19/87)

```
s  $\leftarrow$  q; q  $\leftarrow$  link(q);
```

Page B339, new code to insert after line 10

(8/19/87)

```
if o  $\neq$  0 then
  begin r  $\leftarrow$  link(q); link(q)  $\leftarrow$  null; q  $\leftarrow$  hpack(q, natural);
  shift_amount(q)  $\leftarrow$  o; link(q)  $\leftarrow$  r; link(s)  $\leftarrow$  q;
end;
```

[These new lines also imply changes to the index that aren't shown in this errata list.]

Page B387, line 2

(5/24/88)

is quite short. In the following code we set *hc*[*hn* + 2] to the impossible value 128, in order to

Page B387, line 8

(5/24/88)

```
hc[0]  $\leftarrow$  127; hc[hn + 1]  $\leftarrow$  127; hc[hn + 2]  $\leftarrow$  128; { insert delimiters }
```

Page B390, lines 17–18

(5/24/88)

```
{ Enter as many hyphenation exceptions as are listed, until coming to a right brace; then return 961};
```

```
[The same change applies to lines 20–21, and to page 582.]
```

Page B396, new line after line 34

(5/24/88)

```
trie_link(trie_size)  $\leftarrow$  0; trie_back(0)  $\leftarrow$  trie_size; { wrap around }
```

Page B396, bottom line

(12/12/87)

```
trie_link(0)  $\leftarrow$  0; trie_char(0)  $\leftarrow$  0; trie_op(0)  $\leftarrow$  min_quarterword;
```

Page B397, lines 15-17

(5/24/88)

```

begin c ← trie_c[p];
if c < trie_min then trie_min ← c;
if trie_min = 0 then z ← trie_link(trie_size)
else z ← trie_link(trie_min - 1); { get the first conceivably good hole }

```

Page B400, lines 3-4

(5/24/88)

⟨ Enter all of the patterns into a linked trie, until coming to a right brace 961 ⟩ ≡  
 [The same change applies to page B399, lines 29-30, and to page 582.]

Page B402, line 10

(5/24/88)

```
r ← trie_size; { finally, we will zero out the holes }
```

Page B406, line 9 from the bottom

(1/23/89)

```
shrink_order(r) ← normal; delete_glue_ref(q); glue_ptr(p) ← r; q ← r;
```

Page B417, line 10

(1/23/89)

```
q ← new_skip_param(top_skip_code); { now temp_ptr = glue_ptr(q) }
```

Page B418, line 14

(1/23/89)

```
shrink_order(r) ← normal; delete_glue_ref(q); glue_ptr(p) ← r; q ← r;
```

Page B507, line 13

(12/14/88)

```
if log_opened then selector ← selector + 2;
```

Page B527, line 21

(12/14/88)

```
if log_opened then
```

Page B528, line 5

(12/14/88)

```
if log_opened then
```

Page B547, right column

(9/20/87)

```
co_backup: 366.
```

Page B548, right column

(9/20/87)

```
cur_order: 366, 447, 448, 454, 462.
```

Page B548, right column

(8/7/87)

```
cur_s: 593, 616, 619, 629, 640, 642.
```

Page B551, both columns

(12/23/87)

[Remove '372' from *eqtb* and put it into *eq\_define*.]

Page B552, left column

(4/28/88)

[Insert '358' into *expand*.]

Page B554, left column

(12/23/87)

[Remove '372' from *frozen\_relax*.]

---

Page B559, new entry (12/14/88)

*log\_opened*, 92–93, 527, 528, 534–535, 1265, 1333–1334.

---

Page B559, right column (8/13/87)

[Delete the entry for *low\_mem\_max*.]

---

Page B562, left column (4/28/88)

[Remove ‘358’ from *no\_expand*.]

---

Page B565, left column (8/7/87)

*pop*: 584–585, 586, 590, 601, 608, 642.

---

Page B567, left column (12/23/87)

[Insert ‘372’ into *relax*.]

---

Page B568, left column (4/28/88)

[Move ‘269’ from *save\_index* to *save\_level*.]

---

Page C26, bottom line (7/18/87)

What angle corresponds to the direction North-Northwest?

---

Page C107, line 13 (10/7/87)

**pickup penrazor xscaled heavyline rotated (angle( $z_{32} - z_{31}$ ) + 90);**

---

Page C164, line 10 (4/27/88)

$y_{\$c} = top\ y_{\$l};\ y_{\$d} = y_{\$r};\ x_{\$c} = x_{\$l} - left\_jut;\ x_{\$d} = x_{\$r} + right\_jut;$

---

Page C175, line 23 (1/11/88)

expand into a sequence of tokens. (The language SIMULA67 demonstrated that it is

---

Page C241, line 11 (5/25/88)

**numeric ht#, dp#;**  $ht\# = body\_height\#;$   $.5[ht\#, -dp\#] = axis\#;$

---

Page C248, line 21 becomes two lines (1/24/89)

which might not be numerically stable in the presence of rounding errors.) Another case, not really desirable, is *left\_jut* = *right\_jut* = 0.

---

Page C262, line 15 (12/23/88)

**string base\_name, base\_version;**  $base\_name = "plain";\ base\_version = "1.7";$

---

Page C271, line 12 (1/4/89)

the user and METAFONT’s primitive picture commands. First, some important program

---

Page C271, line 4 from the bottom (12/23/88)

**def cutdraw expr p = % caution: you may need autorounding=0**

---

Page C272, lines 5 and 6 (12/23/88)

```
(cut_ scaled (1+max(pen_lft,pen_rt,pen_top,pen_bot))
rotated theta shifted z)t_;
```

---

Page C273, lines 20 and 22 (9/26/88)

```
(z_+(0,pen_top))t_=round((z+(0,pen_top))t_); z_ enddef;
(z_+(0,pen_bot))t_=round((z+(0,pen_bot))t_); z_ enddef;
```

---

Page C290, line 6 from the bottom (12/23/88)

(2) A throwaway variable, 'whatever', nullifies an unwanted equation at the beginning

---

Page C331, just below the illustration (7/18/87)

Such a pattern is, of course, rather unlikely to occur in a **gf** file, but **GFToDVI** would

---

Page C337, line 11 (4/28/88)

An online "menu" of the available test routines will be typed at your terminal

---

Page C346, entry for **autorounding** (12/23/88)

212, 262, 264, 271–272.

---

Page C350, left column (7/6/88)

**Fatal base file error**, 226.

---

Page C356, left column (1/11/88)

SIMULA67 language, 175.

---

Page C358, right column (2/15/88)

\***yoffset**, 212, 220, 315, 324.

---

Page D2, line 27 (12/14/88)

```
define banner ≡ 'This is METAFONT, Version 1.7' { printed when METAFONT starts }
```

---

Page D36, lines 3–5 (11/6/88)

[Delete this paragraph; it is being moved to page D349.]

---

Page D36, line 7 (12/14/88)

```
begin if log_opened then selector ← term_and_log
```

---

Page D36, line 16 (12/14/88)

```
if log_opened then error;
```

---

Page D66, lines 34–35 (7/9/88)

```
begin if hi_mem_min – lo_mem_max ≥ 1998 then t ← lo_mem_max + 1000
else t ← lo_mem_max + 1 + (hi_mem_min – lo_mem_max) div 2; { lo_mem_max + 2 ≤ t < hi_mem_min }
```

---

Page D347, new line of code after line 5 (12/14/88)

*log-opened: boolean;* { has the transcript file been opened? }

---

Page D347, line 11 (12/14/88)

*job-name*  $\leftarrow$  0; *log-opened*  $\leftarrow$  false;

---

Page D348, line 4 from the bottom (12/14/88)

*log-name*  $\leftarrow$  *a-make-name-string(log-file); selector*  $\leftarrow$  *log-only*; *log-opened*  $\leftarrow$  true;

---

Page D349, lines 6 and 7 (12/14/88)

print error messages or even to *show-context*. The *prompt-file-name* routine can result in a *fatal\_error*, but the *error* routine will not be invoked because *log-opened* will be false.

The normal idea of *batch-mode* is that nothing at all should be written on the terminal. However, in the unusual case that no log file could be opened, we make an exception and allow an explanatory message to be seen.

---

Page D349, lines 11–15 reduce to a single line (12/14/88)

**begin** *selector*  $\leftarrow$  *term-only*;

---

Page D420, bottom line (5/25/88)

**if** *txx mod unity* = 0 **then**

---

Page D441, delete line 2 and change line 12 as follows (5/25/88)

*done: if eq-type(x) ≠ tag-token then clear-symbol(x, false);*  
*if equiv(x) = null then new-root(x);*  
*scan-declared-variable*  $\leftarrow$  *h*;

---

Page D444, line 8 from the bottom (12/14/88)

**if** *log-opened* **then** *selector*  $\leftarrow$  *selector* + 2;

---

Page D510, line 14 (12/14/88)

**if** *log-opened* **then**

---

Page D511, line 11 (12/14/88)

**if** *log-opened* **then**

---

Page D530, new entry (12/14/88)

*log-opened*, 87–88, 782, 783, 788–789, 1023, 1205, 1208.

---

Page D545, left column (10/31/87)

**zscaled** primitive: 893.  
Zabala Saleilles, Ignacio Andres: 812.

---

Page E32, second-last line (9/20/87)

after which comes ‘*math-axis#*; **generate mathsy**’ (which we won’t bother to

---

Page E111, line 29 (10/16/88)

*lft*  $x_{1l}$  = hround  $u$ ;  $x_{1l} - x_{11} = x_{2l} - x_{12} = x_{22} - x_{2r}$  = hround 1.6*cap\_jut*;

---

Page E285, bottom line (12/1/87)

Due to Technical Developments (1968)

---

Page E333, lines 9–11 (1/9/89)

*lft*  $x_{1l}$  = hround( $2.5u - .5mjudged.stem$ );  $x_{1l} = x_{1'l} = x_{2l} = x_{2'l}$ ;  
*lft*  $x_{3l}$  = hround( $.5w - .5mjudged.stem$ );  $x_5 - x_3 = x_3 - x_1$ ;  
**if** not monospace:  $r :=$  hround( $x_5 + x_1$ ) +  $r - w$ ; **fi** % change width for better fit

---

Page E353, lines 38–39 (8/12/87)

**else:** **fill** *diag\_end*( $6r, 5r, 1, 1, 5l, 6l$ ) -- .9[ $z_{5l}, z_{6l}$ ]  
 $\dots \{z_5 - z_6\} .1[z_{5r}, z_{6r}]$  -- cycle; % middle stem

---

Page E387, line 13 (8/12/87)

**pickup** *tiny.nib*; **bulb**(3, 4, 5); % bulb

---

Page E413, lines 37–38 (8/12/87)

**else:** **fill** *diag\_end*( $6r, 5r, 1, 1, 5l, 6l$ ) -- .9[ $z_{5l}, z_{6l}$ ]  
 $\dots \{z_5 - z_6\} .1[z_{5r}, z_{6r}]$  -- cycle; % middle stem

---

Page E459, line 24 (8/7/87)

[Delete the '=' sign between '*lft*' and ' $x_5$ '.]

---

Page E471, line 5 (12/11/88)

$x_2 = good.x .5w$ ; **center\_on**( $x_2$ );

---

Page E471, insert two lines below the rule at bottom of page (12/11/88)

**def** *center\_on(expr x)* = **if** not monospace: % change width for symmetric fit  
 $r := r + 2x - w$ ;  $w := 2x$ ; **fi enddef**;

---

Page E477, line 20 (12/11/87)

$x_4 = good.x .5w$ ; **center\_on**( $x_4$ );  $x_2 = w - x_6 = good.x(x_4 + a)$ ;

---

Page E483, third line of elementary division operator (12/11/88)

$x_3 - .5dot_size =$  hround( $.5w - .5dot_size$ ); **center\_on**( $x_3$ );

---

Page E485, line 4 (8/7/87)

[Delete the '=' sign between '*lft*' and ' $x_5$ '.]

---

Page E487, line 17 (8/4/88)

**fill** *fullcircle scaled (bold + 3.8dw + eps) shifted (.5[z<sub>4</sub>, z<sub>8</sub>])*; % dot  
[Also remove page 487 from the index entry for *dot\_size*, and add it to the entries for *bold* and *dw*.]

---

Page E515, lines 5 and 12 (12/11/88)

$.5[x_1, x_2] = x_3 = good.x .5w$ ; **center\_on**( $x_3$ ); *lft*  $x_1$  = hround( $.5w - u * sqrt48$ );

---

Page E515, line 21 (1/23/89)

**labels(5, 6); zero\_width; endchar;**

[Also put labels '5' and '6' on the upper right figure, page E514.]

---

Page E521, lines 4 and 14 (12/12/88)

$x_1 = x_2 = \text{good.x.}5w$ ; **center\_on**( $x_1$ ); **lft**  $x_3 = \text{hround } u$ ;  $x_4 = w - x_3$ ;

---

Page E537, line 6 (12/11/88)

$x_1 = x_2 = x_3 = x_4$ ;  $x_1 - .5stem = \text{hround}(.5w - .5stem)$ ; **center\_on**( $x_1$ );

---

Page E537, line 19 (12/11/88)

$x_1 = x_2 = x_3$ ;  $x_1 - .5stem = \text{hround}(.5w - .5stem)$ ; **center\_on**( $x_1$ );

---

Page E539, line 4 (12/11/88)

$x_1 = x_4 = x_{30} = x_{33} = \text{good.x.}5w$ ; **center\_on**( $x_1$ );

---

Page E539, line 21 (12/11/88)

$x_1 = x_4 = \text{good.x.}5w$ ; **center\_on**( $x_1$ );

---

Page E541, line 4 (12/11/88)

$x_1 = x_5 = \text{good.x.}5w$ ; **center\_on**( $x_1$ );

---

Page E541, line 17 (12/11/88)

$x_1 = x_{10} = \text{good.x.}5w$ ; **center\_on**( $x_1$ );

---

Page E550, new line after line 23 (8/15/87)

**forsuffixes \$ = notch\_cut, cap\_notch\_cut: if \$ < 3: \$ := 3; fi endfor**

[To make room for this, combine lines 38 and 39 into a single line.]

---

Page E550, line 29 (7/9/88)

**define\_whole\_vertical\_blackr\_pixels(vair, bar, slab, cap\_bar, cap\_band);**

---

Page E572, new entry at bottom (12/11/88)

**center\_on**, 473, 477, 483, 515, 521, 537-541.

**Changes to the Programs and Fonts**

20 February 1989

**TEX**

Changes since 15 June 1987.

331. jump\_out must fix unfinished output (Found by Klaus Gunterman, 3 Aug 87)

```
@x module 593
doing_leaders:=false; dead_cycles:=0;
@y
doing_leaders:=false; dead_cycles:=0; cur_s:=-1;
@z
@x module 617
cur_s:=-1; ensure_dvi_open;
@y
ensure_dvi_open;
@z
@x module 640
dvi_out(eop); incr(total_pages);
@y
dvi_out(eop); incr(total_pages); cur_s:=-1;
@z
@x module 642
if total_pages=0 then print_nl("No pages of output.")
@y
while cur_s>-1 do
  begin if cur_s>0 then dvi_out(pop)
  else begin dvi_out(eop); incr(total_pages);
  end;
  decr(cur_s);
  end;
if total_pages=0 then print_nl("No pages of output.")
@z
```

332. \hangindent=1pt\$\$\halign{... \cr\noalign{\hrule}}\$\$ problem (19 Aug 87)

```
@x module 805
q:=link(head);
@y
q:=link(head); s:=head;
@z
@x module 805, continued
  q:=link(q);
@y
  s:=q; q:=link(q);
@z
```

```

@x module 806
if is_running(depth(q)) then depth(q):=depth(p);
@y
if is_running(depth(q)) then depth(q):=depth(p);
if o<>0 then
begin r:=link(q); link(q):=null; q:=hpack(q,natural);
shift_amount(q):=o; link(q):=r; link(s):=q;
end;
@z

333. \hskip Opt plus ifil\ifdim problem (found by Alan Guth, 20 Aug 87)
@x module 366
@!cvl_backup,@!radix_backup:small_number; {to save |cur_val_level| and |radix|}
@y
@!cvl_backup,@!radix_backup,@!co_backup:small_number;
{to save |cur_val_level|, etc.}
@z
@x
backup_backup:=link(backup_head);
@y
co_backup:=cur_order; backup_backup:=link(backup_head);
@z
@x
link(backup_head):=backup_backup;
@y
cur_order:=co_backup; link(backup_head):=backup_backup;
@z

334. leaders too sensitive near exact multiples (M. F. Bridgland, 9 Nov 87)
@x module 626
begin edge:=cur_h+rule_wd; lx:=0;
@y
begin rule_wd:=rule_wd+10; {compensate for floating-point rounding}
edge:=cur_h+rule_wd; lx:=0;
@z
@x ibid
cur_h:=edge; goto next_p;
@y
cur_h:=edge-10; goto next_p;
@z
@x module 635
begin edge:=cur_v+rule_ht; lx:=0;
@y
begin rule_ht:=rule_ht+10; {compensate for floating-point rounding}
edge:=cur_v+rule_ht; lx:=0;
@z
@x ibid
cur_v:=edge; goto next_p;
@y
cur_v:=edge-10; goto next_p;
@z

```

335. Glitch in fixed-point multiplication of negatives (W.G. Sullivan, 17Nov87)

```
@x module 572
begin alpha:=16*z; beta:=16;
while z>=@'40000000 do
  begin z:=z div 2; beta:=beta div 2;
    end;
@y
begin alpha:=16;
while z>=@'40000000 do
  begin z:=z div 2; alpha:=alpha+alpha;
    end;
beta:=256 div alpha; alpha:=alpha*z;
@z
```

336. If there are no \patterns and some \lccode is 1 (Breitenlohner, 12Dec87)

```
@x module 952
trie_link(0):=0; trie_char(0):=0; trie_op(0):=0;
@y
trie_link(0):=0; trie_char(0):=0; trie_op(0):=min_quarterword;
@z
```

337. \csname might encounter undefined\_cs in a group (Chris Thompson, 23Dec87)

```
@x module 372
begin eqtb[cur_cs]:=eqtb[frozen_relax];
@y
begin eq_define(cur_cs,relax,256);
@z
```

338. \outer\def\{ }\a\ shows temp\_head list garbage (Silvio Levy, 20Apr88)

```
@x module 391
repeat if (info(r)>match_token+127)or(info(r)<match_token) then s:=null
else begin match_chr:=info(r)-match_token; s:=link(r); r:=s;
  p:=temp_head; link(p):=null; m:=0;
@y
repeat link(temp_head):=null;
if (info(r)>match_token+127)or(info(r)<match_token) then s:=null
else begin match_chr:=info(r)-match_token; s:=link(r); r:=s;
  p:=temp_head; m:=0;
@z
```

339. \def\#1{}\input a\b failed (Robert Messer, 24Apr88)  
Qx module 259  
var h:integer; {hash code}  
Qy  
var h:integer; {hash code}  
Q!d:integer; {number of characters in incomplete current string}  
Qz  
Qx module 260  
str\_room(l);  
for k:=j to j+l-1 do append\_char(buffer[k]);  
text(p):=make\_string;  
Qy  
str\_room(l); d:=cur\_length;  
while pool\_ptr>str\_start[str\_ptr] do  
begin decr(pool\_ptr); str\_pool[pool\_ptr+1]:=str\_pool[pool\_ptr];  
end; {move current string up to make room for another}  
for k:=j to j+l-1 do append\_char(buffer[k]);  
text(p):=make\_string; pool\_ptr:=pool\_ptr+d;  
Qz

340. Make patterns work when trie\_min=0 (Peter Breitenlohner, 10May88)  
Qx module 951  
trie\_max:=128; trie\_min:=128; trie\_link(0):=1; trie\_taken[0]:=false;  
Qy  
trie\_max:=128; trie\_min:=128; trie\_link(0):=1; trie\_taken[0]:=false;  
trie\_link(trie\_size):=0; trie\_back(0):=trie\_size; {wrap around}  
Qz  
Qx module 953  
begin c:=trie\_c[p]; {we have |c>0|}  
if c<trie\_min then trie\_min:=c;  
z:=trie\_link(trie\_min-1); {get the first conceivably good hole}  
Qy  
begin c:=trie\_c[p];  
if c<trie\_min then trie\_min:=c;  
if trie\_min=0 then z:=trie\_link(trie\_size)  
else z:=trie\_link(trie\_min-1); {get the first conceivably good hole}  
Qz  
Qx module 966  
r:=0; {finally, we will zero out the holes}  
Qy  
r:=trie\_size; {finally, we will zero out the holes}  
Qz

341. Avoid possible trie\_pointer out of range (24May88)  
Qx module 923  
hc[0]:=127; hc[hn+1]:=127; hc[hn+2]:=256; {insert delimiters}  
Qy  
hc[0]:=127; hc[hn+1]:=127; hc[hn+2]:=128; {insert delimiters}  
Qz

```

342. \input a\romannumeral1 etc.; similar to bug 339. (25May88)
@x module 464
@p function str_toks:pointer; {changes the current string to a token list}
@y
@p function str_toks(@!b:pool_pointer):pointer;
{changes the string |str_pool[b..pool_ptr]| to a token list}
@z
@x module 464, continued
p:=temp_head; link(p):=null; k:=str_start[str_ptr];
@y
p:=temp_head; link(p):=null; k:=b;
@z
@x module 464, concluded
pool_ptr:=str_start[str_ptr]; str_toks:=p;
@y
pool_ptr:=b; str_toks:=p;
@z
@x module 465
begin get_x_token; scan_something_internal(tok_val,false);
if cur_val_level>=ident_val then @<Copy the token list@>
else begin old_setting:=selector; selector:=new_string;
@y
@!b:pool_pointer; {base of temporary string}
begin get_x_token; scan_something_internal(tok_val,false);
if cur_val_level>=ident_val then @<Copy the token list@>
else begin old_setting:=selector; selector:=new_string; b:=pool_ptr;
@z
@x module 465, continued
selector:=old_setting; the_toks:=str_toks;
@y
selector:=old_setting; the_toks:=str_toks(b);
@z
@x module 470
begin c:=cur_chr; @<Scan the argument for command |c|@>;
old_setting:=selector; selector:=new_string;
@<Print the result of command |c|@>;
selector:=old_setting; link(garbage):=str_toks; ins_list(link(temp_head));
@y
@!b:pool_pointer; {base of temporary string}
begin c:=cur_chr; @<Scan the argument for command |c|@>;
old_setting:=selector; selector:=new_string; b:=pool_ptr;
@<Print the result of command |c|@>;
selector:=old_setting; link(garbage):=str_toks(b); ins_list(link(temp_head));
@z

343. **\input\romannumeral6 confusion bypassed (25May88)
@x module 525
begin if (pool_ptr+name_length>pool_size)or(str_ptr=max_strings) then
@y
begin if (pool_ptr+name_length>pool_size)or(str_ptr=max_strings)or
(cur_length>0) then
@z

```

## METAFONT

Changes since 15 June 1987.

540. Typo suppresses an error detection (Chris Thompson, 2May88)  
 @x module 963  
   if txy mod unity=0 then if tyy mod unity=0 then  
     @y  
     if txx mod unity=0 then if tyy mod unity=0 then  
       @z
541. get\_x\_token can lose a scanned declared variable (Chris Thompson, 4May88)  
 @x module 1011  
   if equiv(x)=null then new\_root(x);  
   @y  
   @z  
 @x module 1011  
   done:scan\_declared\_variable:=h;  
   @y  
   done: if eq\_type(x)<>tag\_token then clear\_symbol(x,false);  
   if equiv(x)=null then new\_root(x);  
   scan\_declared\_variable:=h;  
   @z
542. Avoid negative divisor rounding upward (Chris Thompson, fixed 19Jun88)  
 @x module 168  
   else t:=(lo\_mem\_max+hi\_mem\_min+2) div 2; { |lo\_mem\_max+2<=t<hi\_mem\_min| }  
   @y  
   else t:=lo\_mem\_max+1+(hi\_mem\_min-lo\_mem\_max)div 2; { |lo\_mem\_max+2<=t<hi\_mem\_min| }  
   @z
543. Better strategy when near memory overflow (Chris Thompson)  
 @x module 168  
   begin if lo\_mem\_max+1000<hi\_mem\_min then t:=lo\_mem\_max+1000  
   @y  
   begin if hi\_mem\_min-lo\_mem\_max>=1998 then t:=lo\_mem\_max+1000  
   @z
544. Avoid fatal\_error after terminal eof (Tim Morgan, reported 25Oct88)  
 @x module 66 [serious problem occurred if this was called in open\_log\_file]  
   if not input\_ln(term\_in,true) then fatal\_error("End of file on the terminal!");  
   @y  
   if not input\_ln(term\_in,true) then t\_open\_in;  
   @z
545. Force terminal output when open\_log\_file aborts (6Nov88)  
 @x module 789  
   begin print\_err("I can't write on file "");  
   @y  
   begin selector:=term\_only; print\_err("I can't write on file "");  
   @z

546. By popular request, undo #544 and fix the bug a more complex way.

```
Qx module 66 [this undoes change #544]
if not input_ln(term_in,true) then t_open_in;
Qy
if not input_ln(term_in,true) then fatal_error("End of file on the terminal!");
Qz
Qx module 87 [now we get to the new stuff]
begin if job_name>0 then selector:=term_and_log
Qy
begin if log_opened then selector:=term_and_log
Qz
Qx module 88
    error;
Qy
    if log_opened then error;
Qz
Qx module 782
Q!job_name:str_number; {principal file name}
Qy
Q!job_name:str_number; {principal file name}
Q!log_opened:boolean; {has the transcript file been opened?}
Qz
Qx module 783
Q<Initialize the output...Q>=job_name:=0;
Qy
Q<Initialize the output...Q>=job_name:=0; log_opened:=false;
Qz
Qx module 788
selector:=log_only;
Qy
selector:=log_only; log_opened:=true;
Qz
Qx module 789
begin if interaction<scroll_mode then {bypass |fatal_error|}
    begin selector:=term_only; print_err("I can't write on file ''");
    Q.I can't write on file x0>
        print_file_name(cur_name,cur_area,cur_ext); print("'.');Q/
        job_name:=0; history:=fatal_error_stop; jump_out;
        end; {abort the program without a log file}
Qy
begin selector:=term_only;
Qz
Qx module 1023 [this change is optional, but it's a slight improvement]
    if job_name<>0 then selector:=selector+2;
Qy
    if log_opened then selector:=selector+2;
Qz
```

```
@x module 1205
if job_name>0 then
@y
if log_opened then
@z
@x module 1208
if job_name>0 then {the log file is open}
@y
if log_opened then
@z
```

547. (I sincerely hope that there won't be any more)

**Computer Modern fonts**

Changes since 15 June 1987.

```

@x in SYM, the plus-or-minus character
x1=x2=.5w; lft x3=lft=x5=hround u-eps; x4=x6=w-x3;
@y
x1=x2=.5w; lft x3=lft x5=hround u-eps; x4=x6=w-x3;
@z actually the code worked but it was "infelicitous"

@x in SYMBOL, the minus-or-plus character
x1=x2=.5w; lft x3=lft=x5=hround u-eps; x4=x6=w-x3;
@y
x1=x2=.5w; lft x3=lft x5=hround u-eps; x4=x6=w-x3;
@z actually the code worked but it was "infelicitous"

@x in ROMANU, letter J [fixes a bug if dish=0 and crisp<tiny and serifs]
bulb(3,4,5); % bulb
@y
pickup tiny.nib; bulb(3,4,5); % bulb
@z

@x in ROMANL, letter w [makes notch_cut more useful]
else: fill diag_end(6r,5r,1,1,5l,6l)--.5[z5l,z6l]
--.5[z5r,z6r]--cycle;% middle stem
@y
else: fill diag_end(6r,5r,1,1,5l,6l)--.9[z5l,z6l]
..{z5-z6}.1[z5r,z6r]--cycle; % middle stem
@z the same change applies also to letter W in ROMANU

@x in CMBASE, makes lowres types (especially TT) look better
define_blacker_pixels(notch_cut,cap_notch_cut);
@y
define_blacker_pixels(notch_cut,cap_notch_cut);
forsuffixes $=notch_cut,cap_notch_cut: if $<3: $:=3; fi endfor
@z

@x in BIGOP, the \displaystyle coproduct sign
lft x11=hround u; x11-x11=x21-x12=x22-x2r=hround cap_jut;
@y
lft x11=hround u; x11-x11=x21-x12=x22-x2r=hround 1.6cap_jut;
@z

@x in ROMANL, the letter m
lft x11=hround(2.5u-.5stem); x11=x1'l=x2l=x2'l;
lft x3l=hround(.5w-.5stem); x5-x3=x3-x1;
if not monospace: r:=hround(x5+x1)-1; fi % change width for better fit
@y
lft x11=hround(2.5u-.5stem); x11=x1'l=x2l=x2'l; % stem, sic
lft x3l=hround(.5w-.5mfudged.stem); x5-x3=x3-x1;
if not monospace: r:=hround(x5+x1)+r-w; fi % change width for better fit
@z

```

```
@x a new routine for CMBASE, following change_width
@y
def center_on(expr x) = if not monospace: % change width for symmetric fit
  r:=r+2x-w; w:=2x; fi enddef;
@z
@x in SYMBOL, the elementary division operator
x3-.5dot_size=hround(.5w-.5dot_size); w:=r:=2x3;
@y
x3-.5dot_size=hround(.5w-.5dot_size); center_on(x3);
@z
Similarly, whenever the construction "w:=r:=2x*" appears, change it to
"center_on(x*)". This happens in the programs for elementary division operator
(as noted above), large triangle, large inverted triangle, lattice top,
lattice bottom, dagger mark, double dagger mark, club/diamond/heart/spade suit
(all in SYMBOL), plus the diamond operator and universal quantifier in SYM.
The following additional change needs to be made in the programs
for lattice top and lattice bottom:
@x
x1=x2=good.x .5w; center_on(x1); lft x3=hround u; x4=r-x3;
@y
x1=x2=good.x .5w; center_on(x1); lft x3=hround u; x4=w-x3;
@z

@x in SYMBOL, at end of zero-width slash
labels(1,2); zero_width; endchar;
@y
labels(5,6); zero_width; endchar;
@z
```

(I sincerely hope there won't be any more!)