Gauss has called mathematics a science of the eye, and in conformity with this view always paid the most punctilious attention to preserve his text free from typographical errors.

James Joseph Sylvester
address to British Association, 1869

TUGBOAT

THE TEX USERS GROUP NEWSLETTER
EDITOR ROBERT WELLAND

VOLUME 4, NUMBER 1 • APRIL 1983
PROVIDENCE • RHODE ISLAND • U.S.A.

ADDRESSES OF OFFICERS, AUTHORS AND OTHERS

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

BEEBE, Nelson H. F.
College of Science Computer
Department of Physics
University of Utah
Salt Lake City, UT 84112
801-581-5254

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

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

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

BIGELOW, Charles
Department of Computer Science
Stanford University
Stanford, CA 94305

BUNNER, Irene
JDJ Wordware
P O Box 354
Cupertino, CA 95015
415-965-3245

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

CHILDS, Bart
Department of Computer Science
Texas A & M University
College Station, TX 77843
409-845-5470

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

CURTIS, Pavel
ARPAnet: Pavel.Cornell@Udel-Relay

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

FUCHS, David
Department of Computer Science
Stanford University
Stanford, CA 94305
415-497-1646

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

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

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

GUTHERY, Scott B.
Schlumberger Well Services
4669 SW Freeway
P O Box 2175
Houston, TX 77027
713-928-4974

HALL, Bill
Mathematical Reviews
611 Church Street
P.O. Box 8604
Ann Arbor, MI 48107
313-764-7228

ION, Patrick
Mathematical Reviews
611 Church Street
P.O. Box 8604
Ann Arbor, MI 48107
313-764-7228

JOHNSON, John D.
JDJ Wordware
P O Box 354
Cupertino, CA 95015
415-965-3245

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

KRAPP, David
Calma
Architect, Engrg & Construc R&D
11075 Rosalie St
San Diego, CA 92121
619-453-2660

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

MacKAY, Pierre A.
Dept of Computer Science
Univ of Washington
DR-35
Seattle, WA 98195
206-543-2266

McCLURE, Robert M.
Unidot, Inc.
1026 W Maude, #309
Sunnyvale, CA 94086
408-733-6617

MOHR, August
P O Box 1757
Santa Cruz, CA 95061
408-425-7222

NAUGLE, Norman W.
Department of Mathematics
Texas A & M University
College Station, TX 77843
409-845-3104

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

PALAIS, Richard S.
Department of Mathematics
Brandeis University
Waltham, MA 02154

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

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

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

RODGERS, David
Textset, Inc
1612 Anderson
Ann Arbor, MI 48104
313-764-7228

SIEGMAN, Anthony E.
Ginzton Lab 35
Stanford University
Stanford, CA 94305
415-497-0222

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

STERKEN, Jim
2701 Lookout Circle
Ann Arbor, MI 48104

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

THEDFORD, Rilla J.
Mathematical Reviews
611 Church Street
P.O. Box 8604
Ann Arbor, MI 48107
313-764-7228

TOBIN, Georgia K. M.
Office of Research
OCLC Online Computer Library Center, Inc
6565 Frantz Rd
Dublin, OH 43017
614-764-6000

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

TRICKEY, Howard
ARPAnet: hwt@su-ai

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

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

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

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

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

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

TUGboat, the newsletter of the TEX Users Group (TUG), is published irregularly for TUG by the American Mathematical Society, P.O. Box 6248, Providence, RI 02940. Annual dues for individual members of TUG, $20.00 for 1983, include one subscription to TUGboat. Applications for membership in TUG should be addressed to the TEX Users Group, c/o American Mathematical Society, P.O. Box 1571, Annex Station, Providence, RI 02901; applications must be accompanied by payment.

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

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

OFFICIAL ANNOUNCEMENTS

TUG Meeting, July 11–15 1983, Stanford University

A meeting of the TEX Users Group is scheduled to be held at Stanford University during the week of July 11–15, 1983. Joey Tuttle, of I.P. Sharp Associates, Palo Alto, will be in charge of compiling the program and of local arrangements. The first two days will be occupied by an Introductory AMS-TEX82 Users Course for secretaries and technical typists, presented by Michael Spivak (see page 4). A preliminary program and a registration form are being mailed with this issue of TUGboat.

TUG Membership Dues and Privileges

Individual Membership

1983 dues for individual members are $20. A library subscription is also $20; see the statement on page 4 regarding the difference between individual membership and library subscriptions.

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

Institutional Membership

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

TUGboat Schedule

The deadline for submitting items for Vol. 4, No. 2 (1983), will be August 22, 1983; the mailing date will be September 30. Contributions on magnetic tape or in camera copy form are encouraged; see the statement of editorial policy, page 3, Vol. 3, No. 1. Editorial addresses are given on the inside front cover. For instructions on preparing magnetic tapes, or for transferring items directly to the AMS computer, write or call Barbara Beeton at the address given, (401) 272-9500, ext. 299.

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

TUGboat Advertising and Mailing Lists

For information about advertising rates or the purchase of TUG mailing lists, write or call TEX Users Group, Attention: Ray Goucher, c/o American Mathematical Society, P. O. Box 6248, Providence, RI 02940, (401) 272-9500, ext. 232.

* * * * * * * * * * *

General Delivery

* * * * * * * * * * *

LIBRARY SUBSCRIPTIONS— WHAT ARE THEY?

One of the subscription options for TUGboat is the "library subscription". The price is the same as for an individual membership, and a frequent question is, What is the difference?

Some organizations wish to obtain copies of TUGboat for a library, or for various reasons do not wish their "organizational" subscription to be directed to a particular individual (e.g. individuals leave, the project which required TEX information is taken over by someone else, etc.). It is to satisfy these reasons that the "library subscription" was established. Library subscribers will receive the same mailings on the same schedule as individual members. However, only members (individual members or individuals representing institutional members) may participate in TUG meetings.

* * * * * * * * * * *

USERS' COURSE IN $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX

Michael Spivak

It will be necessary for $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX to be completely re-written for TEX82. Fortunately, there are so many improvements in TEX82 that the rewrite should be easier, and the final product less kludgy, than the original. I will be spending the entire month of May at Stanford working on this, so the new $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX should be much better (or perhaps much worse) than the old one. It will probably incorporate some things from Leslie Lamport's macro package LaTEX, though how much, or how, hasn't been decided yet.

The Joy of TEX will also be completely rewritten (sometime in the distant future, on a far off galaxy), partly because of additions, differences, and a few incompatibilities with the old $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX, and partly to improve the exposition. (One main change I have in mind is to eliminate most of the side trips—telling people not to read them just because they're in small type is like the judge telling the jury to ignore that last remark—and instead collect them together into a second part. So the first part will enable people to get going using $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX, and the second part will involve more esoteric things.)

It is to be hoped that I will get a lot of ideas about how to do things from the $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX course given in July. It should be emphasized that this course is a course *only* in $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX! On the one hand (the low one) it is not a course in how to use a computer or a text editor; it will be assumed that everyone has their own favorite (or well-hated) text editor, or will be learning one. On the other hand (the high one) it won't be a course on TEX either; it will cover only those things that you can do with the $\mathcal{A}\mathcal{M}\mathcal{S}$-TEX macro package. Participants are encouraged to bring examples of horrendous typesetting problems; I will try to leave time at the end of the course to attack a few.

* * * * * * * * * * *

TUG TREASURER'S REPORT

December 31, 1982

Income:		
Membership/Publications[1]		
1981 Membership	$ 1,300	
1981 Back volume sales	2,156	
1982 Membership	10,100	
1982 Library subscriptions	535	
1982 Foreign postage option	588	
Supplements[2]	376	$ 15,055
Institutional Membership[3]		
Educational	$ 200	
Non-educational	200	400
Meetings		
Meeting, Cincinnati, 1/82	$ 4,500	
Meeting, Stanford, 7/82[4]	11,025	
Course, Stanford, 7/82[4]	27,468	
Waiver, Meeting/Course Fees[5]	(450)	
Manufacturers' Reps Fees[4]	300	42,843
Other		
Videotape sales/rental	$ 2,200	
Advertising and mailing list sales	-0-	
Royalties (TEX manual)	-0-	2,200
Total income		$ 60,498
Expenses:		
TUGboat (2 issues)		
Printing	$ 2,220	
Postage	1,143	
Editorial services[6]	3,460	
Clerical services[6]	3,300	
Computer expense	3,555	$ 13,678
Meetings		
Cincinnati, 1/82	$ 4,130	
Stanford, 7/82	2,460	6,590

Other		
Supplements[2]	$ 225	
TEX distribution support (Stanford)[7]	-0-	
ANSI meetings[8]	2,503	
Legal and tax consulting[9]	-0-	
Advertising TUG membership & TUGboat[9]	-0-	
General mailings	1,335	
Subsidies[10]	-0-	
Printing, other	1,212	
Miscellaneous[11]	1,688	6,963
Total expenses		$ 27,231

Summary:

Balance forward 1/1/82	$(8,660)
Total income	60,498
Total expenses	(27,231)
Balance 12/31/82	$ 24,607

Notes:

All expense figures include an AMS overhead charge of 17.13%.

1. 1981 membership was open through April 30, 1982; 130 1981 members were accepted in 1982; since then, individual 1981 issues have been available for $10 each ($30 per volume/3 issues). As of December 31, 1982, there were 731 1982 memberships/subscriptions, including libraries and complimentary: 258–foreign, including Canada and Mexico; 473–U.S.
2. 47 copies of reprints of Max Díaz's *Fácil TEX* were sold.
3. The issue of Institutional Membership was finally resolved at the Stanford meeting in July, but it proved to be too late to launch an effective membership drive. Only the American Mathematical Society and Calma became Institutional Members during 1982. However, as of March 31, 1983, TUG has 27 Institutional Members.
4. 136 members participated in the TUG Stanford meeting, July 25–30, 1982: 62 attended the TUG meeting; 9, the TEX82 short course; and 65, both the meeting and short course. Representatives of Hewlett-Packard, Florida Data and Symbolics participated also.
5. The Finance Committee approved the waiver of fees for one member to participate in both the TUG meeting and the TEX82 short course.
6. Editorial services include programming, reviewing and editing; clerical services include maintaining the data base and mailing list, and other administrative duties.
7. $8,500 was allocated to Stanford primarily for Professor Arthur Samuel, who acted as TEX coordinator, answering questions, distributing tapes, and fixing bugs in the TEX source code. However, another source of support was found for this function, and the amount budgeted was not used.
8. The Steering Committee authorized attendance by Lynne Price at two meetings of ANSI X3J6.
9. Some legal/tax consulting was received for which there was no charge. Advertising TUG membership, meetings and TUGboat was accomplished at no cost to TUG through news releases distributed to various trade publications with very favorable results.
10. Money available to the Finance Committee to subsidize travel and membership fees for individuals when appropriate.
11. Postage/express charges, telephone tolls and supplies, plus programmer and clerical services not associated with production of TUGboat.

Respectfully submitted,
Samuel B. Whidden, Treasurer

* * * * * * * * * * *

TUG BUDGET — 1983

Income:		
Membership/Publications		
1981 Back issue sales, 30@$10	$ 300	
1982 Back issue sales, 70@$15	1,050	
1983 Membership, 650@$20[1]	13,000	
1983 Library subscriptions, 30@$20[1]	600	
1983 Foreign postage option, 75@$12	900	
Supplements[2]	500	$ 16,350
Institutional Membership		
1983 Membership, 25@$200[1]	$ 5,000	5,000
Meetings		
Stanford, 7/83	$ 8,000	
Fall '83[3]	7,000	15,000
Other		
Videotape sales/rental	$ 3,000	
Advertising and mailing list sales	500	
Royalties (TEX manual)	500	4,000
Total income		$ 40,350
Expenses:		
TUGboat (2 issues)[4]		
Printing	$ 2,200	
Postage	800	
Editorial services	4,300	
Clerical services	4,000	
Computer expense	3,200	$ 14,500
Meetings		
Stanford, 7/83	$ 4,960	
Fall '83[3]	4,960	9,920
Other		
Supplements[2]	$ 350	
ANSI meetings[5]	1,180	
Legal and tax consulting	500	
Advertising TUG membership & TUGboat[6]	1,500	
General mailings	1,100	
Printing, other	2,180	

Subsidies[7]	1,180	
Miscellaneous[8]	2,300	10,290
Total expenses		$ 34,710

Budget summary: (revised 4/83)

Balance forward 1/1/83	$ 24,607
Total income	40,350
Total expenses	(34,710)
Estimated balance 12/31/83	$ 30,247

Notes:

These figures, with the exception of the budget summary, are identical to those published in TUGboat Vol. 3, No. 2. The rescheduling of the Stanford meeting from March to July has been noted.

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

1. Numbers of members and subscriptions are based on 1982 figures for individual memberships and library subscriptions. The estimate for institutional memberships is now obsolete; 27 institutional members have joined as of March 31.
2. Lengthy descriptions of macro packages will be available for purchase separately.
3. Although no formal plans have been made for a second general meeting in 1983, one is assumed for the Fall.
4. Editorial services include programming, reviewing and editing; clerical services include maintaining the data base and mailing list, and other administrative duties.
5. Support is budgeted for attendance at one meeting of ANSI X3J6.
6. Costs for advertising TUG membership in trade publications.
7. Money available to Finance Committee to subsidize travel and membership fees for individuals when appropriate.
8. Postage/express charges, telephone tolls and supplies, plus programmer and clerical services not associated with production of TUGboat.

Respectfully submitted,
Samuel B. Whidden, Treasurer

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

Software

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

TEXhax SUMMARY

David Fuchs

Here's a distillation of the information that has gone through the TeXhax mailing list over the last few months. Most of it consists of details of installing and maintaining the WEB and TEX systems, and it is not very well organized.

One change to both systems is that all of our WEB programs now discard trailing blanks on all input lines. This aids in portability, since there are a surprising number of systems in which the number of trailing spaces a program gets to see on each line is some random function of the phase of the moon. It also improves the TEX language, since it is harder to have invisible differences between two files lead to differing results. Implementers should check their change files, since we had to alter a system-dependent module in many of the programs to make this change (look for *input_ln*).

Another reason you might get the "Change file entry did not match" error is that most of the discretionary hyphens (\-) that were included in the WEB source code have been removed, since they are no longer necessary with TEX82's hyphenation algorithm. Similarly, a number of comments in the WEB programs that looked like `$x$` now look like `|x|`, since this is the correct way to use WEB. Finally, the ANSI Pascal document specifically rules out the statement `WRITE(I:0)`, so we have changed all our WEB programs to say `WRITE(I:1)` to mean "write out I with no leading or trailing blanks".

For TEX, there are a few other changes to watch for: The module ⟨Globals in the outer block⟩ has been renamed ⟨Global variables⟩, so if any of your change file entries say "⟨Globals ... ⟩" you'll have to change them. The *qi* and *qo* macros have been joined by *hi* and *ho*, which you should add to your change file if this is a module you changed. Finally, the macros *float* and *unfloat* have been added to aid TEX installers on systems on which it is impossible to use native floating point numbers for glue ratios.

There are a few new features in WEB. First, the new control sequence `@"` is similar to `@'`, except that it indicates that a hexadecimal constant follows. Note that TANGLE.WEB actually uses `@"` in one place, so if you are trying to bootstrap a new Tangle from an old one, you'll have to temporarily change `@"8000000` to `@'1000000000` in the new TANGLE.WEB when you process it with the old Tangle (otherwise, the old Tangle will ignore the `@"` and interpret 8000000 as a decimal number, and you'll end up with a TANGLE.PAS that won't run on itself). This adulterated Tangle should be able to run through a clean copy of the new Tangle and produce the same correct Pascal program. Once you reach this state, you can toss out the altered Tangle, and forget that we ever suggested changing a WEB file directly.

Two other new primitives: `@/` forces a line-break in the Pascal output of Tangle, and `@=...@>` means to put the included text into the Pascal output "verbatim". Also, WEAVE now keeps track of which modules are changed, and prints all references to these module numbers with an asterisk. WEBHDR al-

lows you to output only the changed modules, so you can make a short listing of only the modules you changed, by using your change file to put "\let\maybe=\iffalse" in limbo at the beginning of the WEB program. See the latest TOPS-20 TEX change file for an example of this.

The new TANGLE.WEB uses the new @ feature, while the old versions of Tangle will ignore it as an unknown control code. Thus if you are bootstrapping to the new Tangle by using an old Tangle, your first new TANGLE.PAS will not quite agree with the TANGLE.PAS that you get after running the first new TANGLE.PAS on the new TANGLE.WEB. But the difference is only in the debug-breakpoint code, and doesn't alter the meaning of the Pascal program, so this won't cause any special trouble.

A number of people have pointed out deficiencies in the way change files are done. For instance, it is not too convenient to change a single line in the middle of a module. There is also some sentiment for a system in which you list *all* of the lines you are changing, so that Tangle/Weave would detect the case where you get a new version of the program and one of the modules which you have in your change file has been altered (so your change might no longer be correct). I personally favor this approach, but we don't have the manpower to consider implementing this right now.

A note of caution on Tangle: More than one person has tried altering Tangle to produce all lower case output. This is dangerous business, because Tangle's code for collapsing constants looks in the output buffer for the strings "DIV" and "MOD". Thus, you can get incorrect results if you change Tangle such that "div" and "mod" might end up in the buffer.

Tangle's output is now a bit different than it used to be. First of all, line breaks occur at semi-colons or right braces whenever possible. Also, comments are inserted showing where each fragment of code came from. For instance, part of the Pascal output might look like:

```
...{123:} Foo; {456:}
        Bar; {:456} More; {:123} ...
```

This means that module number 123 looked something like this:

Foo;
(Another module)
More;

And module 456, called (Another module), consisted of the single line:

Bar;

Thus, it is now easy to tell where each piece of Pascal code came from. We intend to use this information with the data gathered in the statement counting experiment to find out how much of the total runtime of TEX82 each individual module accounts for.

Note that the new output format implies that each program produced by Tangle is of the form:

```
{xxx:} PROGRAM ZZZ; .... END. {:yyy}
```

The Pascal manual and the ANSI standard are not specific about whether a program can end with a comment, but no one has yet reported this to be a problem.

Here's a problem that a few TEX installers are facing, having to do with evaluation of expressions. Consider:

```
program expres(output);
type sixteenbit = 0..65535;
var s,t : sixteenbit; i,j : integer;
procedure p(k :integer);
begin write(k) end;
begin
    s := 65535; i := s + 10; p(i);
    8em t := 10; i := s + t; p(i);
    8em p(s + t); p(65535 + 10);
end.
```

This program should print out 65545 four times. We have heard reports of a few compilers that try to optimize some of the expression evaluations to be sixteen bit calculations, and produce the wrong result in some of these cases. The new ANSI Pascal standard document specifically says that all expressions have to be computed to full integer precision, so the compilers in question are in the wrong. This does not bode well for sixteen-bit systems with a "LONG INTEGER" type, however, since even if you use your change file to change all integer variables to LONG INTEGER, the compiler might not do the above calculations correctly.

Us folks at Stanford are interested in addressing any bugs you may find in any of our WEB programs. The bug report format we most appreciate is "Module X in program Y is wrong because Z." If you have found that a bug exists in TEX82, but you can't locate the cause, then it would help for us to look at all the data. We need input files as well as the log file. If you say \tracingall, TEX gives its most verbose output. So please turn everything on, in the vicinity of whatever bug you're diagnosing; this makes it much easier to pinpoint the problem.

In the same vein, please send a note if you come across any supposedly non-system-dependent modules in TEX that you find you had to change. They might be appropriate to be added to the index under "system dependent", or we may alter them so they aren't system dependent any more.

A hint for installers/maintainers: One trick I use is to keep a copy of Prof. Knuth's change file for the Sail system. Whenever there is a new TEX, I look to see where his new change file is different from his old one, and I check my change file for TOPS-20 to see if it needs a similar alteration. You can do the same thing by keeping a copy of the TOPS-20 change file, and seeing when it changes. Actually, now that things are pretty settled down, it is probably easier just to check TeX82.BUG to see which modules have been changed, to check whether your system-dependent stuff is impacted. I also save a copy of all the WEB programs, so that when new ones come, I can find all the changes, but this has not been necessary very often.

Advice on making your TEX efficient: It is important that all records are declared such that they will be packed efficiently into memory. Referring to Module 110 in the brown Version 0 listing of TEX, note how the type *memory_word* is made. The hope is that your compiler will use 32 bits of storage for each *memory_word*. Well, one of the variants of *memory_word*, *glue_ratio*, is a *real*. In Pascal/VS, for instance, a *real* is allotted a double-word, which blows it right there. The proper thing to do in that case is to change the definition of *glue_ratio* in module 106 to be a *short_real* (in the change file, of course).

Similarly, VS Pascal insists that if you really wanted a variable declared 0..255 to occupy a byte instead of a word, you have to say

foo: **packed** 0..255

(note that the placement of the reserved word **packed** is non-standard). So, to get TEX down to a reasonable size, you'll have to change the definitions of *quarter_word*, *half_word*, and perhaps even *two_choices* and *four_choices* in module 110. This sort of change might be appropriate in other places too, but because most of memory is taken up by *memory_word*s, it shouldn't be crucial.

After your TEX port has passed the TRIP test, you should turn off run-time debug support. For production TEX it shouldn't be necessary to do bounds checking, uninitialized variable checking, and the like. Of course, if you run into an apparent bug, you'll probably want to turn it back on to help trace the problem as far as possible before reporting it to Stanford (hint, hint).

Advice on porting TEX: First, make sure to consider the modules that are listed in the index under 'Dirty Pascal'. A few of these modules are debugging code that look through the big "mem" array, and are considered dirty because they read from variants that weren't written into. That is, TEX may have done `MEM[0].SC:=0.0`, but a dirty module might `WRITE(MEM[0].INT)`. This has worked OK on all machines we've run into so far, but one can imagine an architecture in which it would cause a problem. Of course, normal users will never run into this code—it's only for TEX installers/debuggers.

The dirty module ⟨Display the value of *glue_set*(*p*)⟩ requires a little special attention. On our system, if *glue_set*(*p*) was erroneously set to a pattern of bits that did not represent a legal floating point value, due to a bug in TEX, then our run-time system would blow up while trying to print out its value. In order to make the code robust in the face of such bugs, so that the person trying to find the origin of the bug would be able to continue the job and use TEX82's internal debugging support to look around for further clues, the module in question was changed so that it first looked at *glue_set*(*p*) as an integer and figured out whether it was a legal floating point number. If not, it simply prints "?.?" instead of *write*(*glue_set*(*p*)). Of course, this is very system-dependent. On other computers, it may be appropriate to remove this test, but it will certainly be true that you'll at least have to change it.

Other than the debugging modules mentioned above, TEX should never read from a different variant than it writes into in any record. Also, TEX should never refer to an uninitialized variable, except for the variable *ready_already*. The details about *ready_already* are pretty well covered in the section of the TEX program titled "**The Main Program.**"

Different systems have different conventions about I/O to the user's terminal. On some systems, INPUT is hardwired to the keyboard, OUTPUT is the screen, and that's it. On others, there might be another built-in file that is hardwired to the screen, and INPUT and OUTPUT might always refer to disk files. Another possibility is that the program can dynamically tell the system which files should be associated with the terminal, and which with the disk. The TEXware programs and TEX itself try to be flexible enough to deal with all these possibilities. Consider TFtoPL, which mentions three files in its program statement in module 2: *tfm_file*, *pl_file* and *output*. Module 2 also mentions that all of the writing to the *output* file goes through the *print* and *print_ln* macros; so if you have a system, say, where output to the terminal must go to file *tty*, then you can change the definitions to:

@d *print*(#)==*write*(*tty*,#)
@d *print_ln*(#)==*write_ln*(*tty*,#)

You'd probably also want to change the program statement not to include the file *output*, and you might have to do a rewrite on *tty*.

The same comments go for PLtoTF. DVItype is a bit different, though. It uses the file *output* for its main output, so you probably don't want this file associated with your terminal. Hence, if *output* is hardwired to the terminal on your system, you will want to change the macros in module 3 to:

```
@d print(#)==write(type_file,#)
@d print_ln(#)==write_ln(type_file,#)
```

You'll also have to include a declaration of *type_file*, and do a *rewrite* in some other modules.

DVItype holds a dialog with the user to get the values of certain parameters. The files *term_in*, *term_out*: *text_file* are declared in the section **Optional Modes of Output** to be used for this purpose. If, say, your system reserves the pre-declared files *input* and *output* for this function, then you can change the declarations to macros:

```
@d term_in==input
@d term_out==output
```

Pretty sneaky! You can do the same thing if the file *tty* is hardwired to your terminal.

There are more headaches due to differing approaches to I/O on different systems. On many systems, reading a single character from a file is a relatively expensive operation. That is, the time spent doing

```
program slow;
var c: char;
begin
while not eof do begin
    while not eoln do read(c);
    readln;
    end;
end.
```

is a major portion of how long TEX itself takes to run. There's not too much we can do about this if your system does *read*(*c*) via a slow procedure call. However, many systems provide some sort of extension so that you can read a whole line of input at once, more efficiently. For instance, on our system, you can say:

```
var line: packed array [1..80] of char;
    howmany: integer;
read(line:howmany);
```

and the variable *howmany* will get the number of characters actually read in. In any case, all of our programs always read a line at a time into a buffer array (usually in a procedure called *input_ln*), so if a facility similar to the one just mentioned exists in your system, you should be able to use it with TEX by changing just a few modules. (Some people may be able to do this sort of thing by calling a procedure in another language.)

Things are even worse for I/O of binary byte data (TFM and DVI files) and word data (FMT files). Not only might it be inefficient, but I/O of binary data is even less standard than character. Even if your compiler accepts things like:

```
var w: file of integer;
    b: file of 0..255;
write(w,456); write(b,123);
```

you are well advised to check out that these things will work as expected. It is best to experiment with a small program to read and write such files before jumping into the TEX system, if there is any doubt as to how these files will work on your system. Once again, for efficiency's sake, you may have to block things up yourself using an array as a buffer.

Two installation points: There have been some questions on how to run the TRIP test file. To get results that are identical to ours, you'll have to compile a special version of TEX that has some compile-time constants set to values that probably don't match the values you'd want to use in a production version of TEX. In particular, you should turn on the **stat** and **debug** switches, and make the following definitions in your change file:

```
@!mem_max=3000; {greatest index in TeX's internal
    mem array, must be strictly less than max_halfword;
    this is the value appropriate to the TRIP test file}
@!error_line=64; {width of context lines
    on terminal error messages}
@!half_error_line=32; {width of first lines
    of contexts in terminal error messages,
    should be between 30 and error_line-15}
@!max_print_line=72; {width of longest
    text lines output, should be at least 60}
@!dvi_buf_size=800; {size of the output buffer,
    must be a multiple of 8}

@d hi_mem_base=2200 {smallest index in the
    single-word area of mem, must be
    substantially larger than mem_base
    and smaller than mem_max}
```

Finally, TEX's *try_break* procedure is still to big for some people's compilers when the **stat** switch is turned on. We suggest using your change file to put the **stat** code into a small procedure statically embedded within *try_break*, so that you won't have to worry about local/global variables.

OUTPUT DEVICES AND COMPUTERS

	Alpha CRS	APS-5	Comp. 8600	Epson MX-80	Facit 4542	Fla. Data OSP	HP2680	Imagen Imprint 10	Laser-grafix	Linotron 202	Perq/ Canon	Symbolics LGP-1	Varian	Versatec	Xerox Dover	Xerox XGP	Xerox 9700
Amdahl(MTS)																	*Michigan
Amdahl (MVS)			Wash. St. U														
Apollo								*OCLC						*OCLC			
Ethernet							Stanford								Stanford		
DEC10								Vanderbilt						Vanderbilt			
DEC10 *														*G A Technology			*Univ. Del.
DEC20	AMS					Math Reviews		*SRI		Adapt, Inc.			AMS		*CMU		
DG MV8000									*Texas A&M								
HP1000				*JDJ Wordware													
IBM(MVS)																	*CIT
IBM(VM)														*SLAC			
IBM370		Info. Handling															
Onyx C8002								TYX Corp.									
Prime														*Livermore			
Sail																Stanford	
Siemens BS2000											*GMD Bonn						
Sun *		*Textset				*Textset				*Textset							
Univac 1100			Univ. Wis.														
VAX (Unix)												UC Santa Cruz		Cal. Tech.			
VAX (Unix) *								*UC Irvine						*Univ. Wash.	*Stanford		
VAX (VMS)					*INFN CNAF			Argonne	Texas A&M			Calma	Sci. Appl.	Sandia			

*(running TEX82)

Index to Sample Output from Various Devices

As with previous issues of TUGboat, several articles have been submitted for publication in the form of camera copy. The following items were prepared on the devices indicated, and can be taken as representative of the output produced by those devices. With each item is given a percentage, which is the size of the copy as received; items received as copy larger than 100% were reduced photographically using the PMT process. The bulk of this issue, as usual, has been prepared on the DEC 2060 and Alphatype CRS at AMS.

– Epson MX-80; 100%: I. J. Bunner and J. D. Johnson, *TEX on the HP-1000*, p. 16; HP-1000.

– Florida Data OSP 130; 130%: p. 13; DEC 2060.

– HP 2680A; 145%: L. Carnes, *"Small" TEX*, p. 24; HP-3000.

– Imagen Imprint-10; 100%: G.M.K. Tobin, *Computer Calligraphy*, p. 26; Apollo.

– Versatec; 130%: R. Furuta and P. MacKay, *Unix TEX Site Report* and the two articles which follow, p. 17; VAX/UNIX.

LOW-COST DOWNLOADABLE FONT DEVICES

Nelson H.F. Beebe
College of Science Computer
Department of Physics
University of Utah
Salt Lake City, UT 84112
Tel: (801) 581-5254

A fundamental problem with typesetting is that output devices capable of displaying material as it would appear in typeset form are expensive and slow. Electrostatic printer plotters have resolutions of 100 to 400 dots per inch and can produce up to 20 pages/minute, but cost from $10K to $100K and require special paper. Laser printers offering resolutions of 240 to 300 dots per inch can produce up to 12 pages/minute on normal paper and cost around $20K to $25K. High-quality phototypesetters start around $20K, but are generally quite slow, requiring as much as 10 minutes/page, often on expensive photographic paper.

It seems worthwhile therefore to investigate what low-cost output devices capable of accepting downloaded fonts are available on the market today. The goal is to find output devices which can display a typeset page about as rapidly as normal text can be displayed. This survey sets an upper limit of about $5K ($10K for color terminals) and includes both dot-matrix printers and display terminals. The cost limit excludes workstations like the Xerox Star and the Apple Lisa which are designed from the beginning to support multiple fonts. Multi-font devices whose fonts are pre-recorded in ROM storage by the manufacturer are excluded because of their general lack of usefulness for scientific typesetting.

Since the number of manufacturers of printers and terminals has become very large, I will no doubt have missed several. If reader response is sufficient, I would be willing to update this column in later issues of TUGboat. At present, all but two of these devices have severely limited maximum character matrix sizes, and most cannot properly handle proportional spaced fonts. The maximum matrix size is noted as "$n \times n$", and the number of downloaded characters or amount of memory for font storage, where available, is also given. For terminals, the screen resolution ($H \times V$) is also given, since this is a limiting factor on the display quality. Entries are tabulated alphabetically by manufacturer name.

. . . . . .

Printers

Model(s): CI-300, CI-600
Character Grid: 17×17
Manufacturer: C-Itoh Electronics, Inc., 5301 Beethoven Street, Los Angeles, CA 90066, USA
Telephone: (213) 306-6700.

Model(s): Florida Data OSP 120 and 130
Character Grid: matrix variable depending on character configuration; maximum dot density - obtainable 360H x 192V (graphics option available)
Manufacturer: Florida Data, 600-D John Rodes Blvd., Melbourne, FL 32935, USA
Telephone: (305) 259-4700.
Remarks: Up to 18K RAM font storage. Sample output of the earlier Florida Data BNY model may be found in TUGboat Vol. 2, No. 1 (February 1981), pp. 130-131; sample output of the current OSP 130 appears on p. 12 of this issue.

Model(s): Infoscribe 1000
Character Grid: 7×9
Manufacturer: Infoscribe, 2720 S. Croddy Way, Santa Ana, CA 92704, USA
Telephone: (714) 741-8595.
Remarks: 96 characters font storage

Model(s): Okidata Microline 92 and 93
Character Grid: 9×9
Manufacturer: Okidata Corp., 111 Gaither Drive, Mount Laurel, NJ 08054, USA
Telephone: (800) 654-3282.
Remarks: 96 characters font storage

Model(s): Printek 900 series
Character Grid: 9×9 and 18×18
Manufacturer: Printek, 1517 Townline Road, Benton Harbor, MI 49022, USA
Telephone: (616) 925-3200.

Terminals

Model(s): BBN BitGraph
Screen: 768×1024 monochrome
Character Grid: unlimited; proportional fonts supported.
Manufacturer: Bolt Beranek and Newman, Inc., 10 Moulton Street, Cambridge, MA 02174, USA
Telephone: (617) 497-3178.
Remarks: With the default 12×16 font, the BitGraph displays 64 lines of 85 characters. With a 5×8 terminal font, the smallest that has lowercase letters with descenders, it shows 113 lines of 128 characters, and is still quite readable.

Model(s): CIT-427
Screen: 640×480 color
Character Grid: 8×14
Manufacturer: C-Itoh Electronics, Inc., 5301 Beethoven Street, Los Angeles, CA 90066, USA
Telephone: (213) 306-6700.
Remarks: font storage 7 96-character sets

Model(s): Datacopy
Screen: 1728 × 2200 monochrome
Character Grid:
Manufacturer: Datacopy, 1070 East Meadow Circle, Palo Alto, CA 94303, USA
Telephone: (415) 493-3420.
Remarks: Datacopy does not yet make this display available as a terminal, but I am including it here as something to be watched closely. The screen is sharp enough to display readable 4-point type on a full page of text, or the engraving lines in a page image of a dollar bill. The January 1983 issue of *Computer Graphics World* (p. 65) contains a photograph of a sample image.

Model(s): Direct 828, 831, 1000, 1025
Screen: 640 × 480 monochrome
Character Grid: 10 × 12
Manufacturer: Direct Inc., 4201 Burton Drive, Santa Clara, CA 95054 USA
Telephone: (800) 538-8404.
Remarks: 2 128-character user-definable fonts; downloading must be in blocks of 16 characters.

Model(s): Quadram Omega Data X7
Screen: 960 × 528 monochrome
Character Grid: 5 × 7 ASCII or 6 × 8 symbol
Manufacturer: Quadram Corp., 4357 Park Drive, Norcross, GA 30093, USA
Telephone: (404) 923-6664.
Remarks: 66 lines of 160 characters, or two split screens with 66 lines of 80 characters. Up to 2048 downloaded characters.

Model(s): Ramtek 6211
Screen: 640 × 480 color
Character Grid: 8 × 12, proportional spacing
Manufacturer: Ramtek Corp., 2211 Lawson Lane, Santa Clara, CA 95050, USA
Telephone: (408) 988-2211.

Model(s): Tektronix 4027, 4112, 4113 (raster), 4114, 4116 (storage tube)
Screen: 640 × 480 color (4027, 4113), 640 × 480 monochrome (4112), 4096 × 4096 monochrome (4114, 4116)
Character Grid: 8 × 14 on raster, vector fonts on storage tubes
Manufacturer: Tektronix, Inc., Instruments Division, P.O. Box 500, Beaverton, OR 97077, USA
Telephone: (503) 627-2256.
Remarks: font storage 31 96-character sets (tube), and 4116 support user-defined vector fonts.

Model(s): Terak 8510a, 8600
Screen: 640 × 480 monochrome (8510a) and color (8600)
Character Grid: 8 × 10 and 16 × 10 in monochrome, unlimited in color
Manufacturer: Terak Corp., 14151 N. 76 St., Scottsdale, AZ 85260, USA
Telephone: (602) 998-4800.
Remarks: LSI-11 based workstation with DEC RT11 or UCSD Pascal operating systems.

Model(s): Vectrix VX128 and VX384
Screen: 672 × 480 color
Character Grid: 8 × 8
Manufacturer: Vectrix Corp., 700 Battleground Ave., Greensboro, NC 27401, USA
Telephone: (800) 334-8181.
Remarks: Low-cost frame buffer; no cursor commands; font magnification factors 1..16 plus slants in 45 degree increments; 96 downloadable characters.

* * * * * * * * * * *

TEX ON THE OSP130

Patrick Ion
Bill Hall
Rilla J. Thedford
Mathematical Reviews
611 Church Street, Ann Arbor, Michigan 48104

Mathematical Reviews is now using the Florida Data OSP130 as a TEX output device. The OSP130 prints on continuous stock (cards and paper) as well as on cut-sheet paper. Average time per dense page of TEX output is 1 min:40 sec. Current dot resolution is H120 x V128, although the machine does have the potential for producing much higher resolution. This improvement requires a modified spooler capable of handling a second set of font masks. At present the fonts used are those with a resolution of 128 x 128 pixels/inch prepared by METAFONT for the Florida Data BNY. Although the dot resolution is similar, the quality of the output is greatly improved. We are all very pleased.

Our OSP130 is driven by a Monolithic Systems MSC8004 Z80 processor which is fed and controlled through a multiplexor by the Dec2060 machine in Providence, RI. We used a parallel Centronics interface and a hardware handshake, checking Busy only [this involves using the 8255 in Mode 1], to interface the OSP with the Monolithic.

The authors wish to thank the following people for all their help in making our first attempt at such a task a SUCCESS: Frank Price and Jim Neil of Florida Data Corporation, Marty Haase and Joel Berkman of Monolithic Systems, and David Fuchs of Stanford.

Anyone desiring more information about TEX on the Florida Data OSP should contact the authors at the above address or call 1-313-763-6828.

and obtain the factorizations

$$(7)\quad 5^{5k}-1=(5^k-1)L_{5k}M_{5k},\ L_{5k},M_{5k}=5^{2k}+3.5^k+1\mp 5^k(5^k+1)$$

$$(8)\quad 6^{6k}+1=(6^{2k}+1)L_{6k}M_{6k},\ L_{6k},M_{6k}=6^{2k}+3.6^k+1\mp 6^k(6^k+1)$$

$$(9)\quad 7^{7k}+1=(7^k+1)L_{7k}M_{7k},\ L_{7k},M_{7k}=(7^k+1)^3\mp 7^k(7^{2k}+7^k+1)$$

$$(10)\quad 10^{10k}+1=(10^{2k}+1)L_{10k}M_{10k},\quad \text{where } L_{10k},M_{10k}$$
$$=10^{4k}+5.10^{3k}+7.10^{2k}+5.10^k+1\mp 10^k(10^{3k}+2.10^{2k}+2.10^k+1)$$

$$(11)\quad 11^{11k}+1=(11^k+1)L_{11k}M_{11k},\quad \text{where } L_{11k},M_{11k}$$
$$=11^{5k}+5.11^{4k}-11^{3k}-11^{2k}+5.11^k+1\mp 11^k(11^{4k}+11^{3k}-11^{2k}+11^k+1)$$

The appropriate formulas for L and M are printed at the end of each relevant main table.

The numbers with an Aurifeuillian factorization can be completely factored more readily than other numbers b^n-1, because they break into two roughly equal pieces. For this reason, Table 2LM has been extended to 2400, twice as far as the other base 2 tables. The Aurifeuillian factorizations for the other bases (in Tables 3+, 5−, 6+, 7+, 10+, 11+ and 12+) are not given in a separate table as in base 2, but are incorporated in a special format in the tables themselves and are carried somewhat farther than the consecutively indexed entries, the extensions being listed below a line of dashes in the respective tables.

Since the factorizations produced in (4) to (11) cut across those produced in (2) and (3), it is important to analyze how the two factorizations relate to each other.

Example. Since $156=2^2.39$, we have from (3) that

$$2^{78}+1=\prod_{d|39}\Phi_{4d}(2)=\Phi_4(2)\Phi_{12}(2)\Phi_{52}(2)\Phi_{156}(2)$$
$$=(5.13.53.157.1613)\ \underline{13^*}.\underline{313}.\underline{1249}.\underline{3121}.\underline{21841}$$

and from (4) that

$$2^{78}+1=L_{78}M_{78}=(13.53.157.\underline{13^*}.\underline{313}.\underline{1249})(5.1613.\underline{3121}.\underline{21841})$$

The fact that the second factorization splits both the algebraic and primitive parts of $2^{78}+1$ suggests that in order to describe this multiplicative structure, the primitive parts of L_n and M_n should be defined and then L_n and M_n can be expressed as a product of primitive parts as in (2). To do this we denote the respective primitive parts by L_n^* and M_n^*. For base b, let $\epsilon_d=\epsilon_d(b)=[1+(b|d)]/2$, where d is odd, $(b,d)=1$, and $(b|d)$ is the Jacobi symbol. (Recall that $(b|1)=1$.) Also, let $n=2^sm$, m odd, $s\geq 0$. Then we have the formulas (which we state without proof)

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

Site Reports

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

NEWS FROM ALL OVER

David Fuchs

Here's lots of news in no particular order:

TEX82. The code's in pretty good shape. We've been running with version 0.95 for six weeks, during which time we found three small bugs. At the same time, installing TEX found as many bugs in the VAX/VMS Pascal 2.0 compiler, and one bug in the DECSystem 10/20 microcode! This must mean something.

As I am writing this, Professor Knuth is testing version 0.96, which corrects the bugs, and adds a dozen or so new features, and about as many small improvements. Barring any further problems with microcode, by the time you read this, version 0.96 should be available through normal distribution channels. We expect to need only one more round of changes by the time the manual is complete.

Chapter 21 of the new manual ('Boxes') is currently being written. When Chapter 23 is done, we'll do a pre-print of the whole manual, and declare the current version of the code to be 1.0. Our tape distribution people [Ron and Maria Code, Data Processing Services, 1371 Sydney Drive, Sunnyvale CA 94087 (408-735-8006)] are willing to take orders for version 1.0, and they will hold them until everything is ready to go. Of course, you can order version 0.96 if you like, since it will be almost the same.

A new feature of the tape distribution is that you can order all the fonts at any of a number of different resolutions. Currently, we have 200, 220, 240, 260, 280 and 300 dot/inch versions available, but more will be forthcoming if the demand is great enough.

Right now, the distribution tape includes change files for TOPS-20 only. I hope to be able to add change files for other systems in the near future. On the other hand, it is fine for users of the same kind of systems to band together and handle it themselves. For instance, the VAX/UNIX port is up and running at a number of sites, and you should contact Rick Furuta for details on its distribution.

There have been several independent IBM ports, by Susan Plass at Stanford CIT, Roger Chaffee at SLAC, and Bruce Nemnich at MIT. The latter two have both discovered a method of saving a preloaded TEX under VM/CMS! John Johnson reports he has TEX82 running on his HP-1000, and is happy to talk to interested parties (he's getting output on his Epson printer). Bart Childs and Norm Naugle at Texas A&M are in the final stages of getting TEX82 running on their Data General MV8000. I'm sorry if I've left anyone out, my only excuse is that things are very hectic.

Conspicuously absent from the list is any 68000-based system. Our SUNs are just coming into operation, but it looks like the modifications done to the VAX/UNIX compiler to enable it to deal with TEX can be transferred directly to the SUN version of UNIX 4.1C BSD, and SUN Microsystems has promised to do so. Another encouraging sign: Apple is giving us a Lisa to try to put TEX on; it remains to be seen whether their compiler is good enough.

I've been working on a VMS port. Tangle, Weave, PLtoTF, TFtoPL and DVItype went into VMS 3.2 running Pascal 2.1 with little problem, since the new compiler has much improved I/O support. TEX82 on VMS currently is able to correctly run through the TRIP test file. The biggest installation problems were the half-dozen or so compiler bugs I ran into on the way and had to kludge around. These have been reported to Digital, so when the next release of the compiler rolls around everything should be beautiful.

Since the last TUGboat, we've added a new means of communication for the TEX community: an inter-network mailing list called TeXhax. Recipients of messages to TeXhax are on the ARPAnet, CSnet or UUCPnet. The mailing list has 70 members scattered around the country, and even in England and Germany. A number of the addresses are computer bulletin boards, so the readership is actually larger than 70. Most messages to date have dealt with TEX installation and portability questions, but I'm hoping that when the manual is completed and both TEX82 and CSnet are in wider use, TeXhax will become a forum for information about macros, output devices, fonts, etc. For those of you with no access to any network, we'll be summarizing newsworthy items from TeXhax in TUGboat (see my other article in this issue).

If you want to be added to TeXhax, send mail to TEXHAX-REQUEST@SU-SCORE on the ARPAnet, or DRF@Stanford on the CSnet, or Shasta!DRF on UUCP. Please include your network address, your real name and location, and CPU and output devices you're interested in. To send a message to everyone on TeXhax, send mail to TEXHAX@Score (or ..Shasta!Score!TEXHAX, or to me, and I'll relay it).

One of the most exciting things going on is Leslie Lamport's work on his LaTeX system. This macro package makes TEX82 look a bit like Brian Reid's SCRIBE system, including such features as:

- A Scribe-like cross-reference system, where sections, theorems, etc. are given symbolic names by a `\label{foo}` command, and then referenced as "Section `\ref{foo}`".
- Scribe-like environment commands—e.g., for an enumerated list, one writes `\begin{enumerate} ... \end{enumerate}`. These will nest properly, and different levels will be numbered differently.
- A PICTURE environment, with picture-drawing commands.
- The ability to define different document styles.
- Automatic table of contents and list of figures.
- Flexible figure and formula numbering.
- Partial document compilation.

The system is now in "Beta Test" at a few installations, and the first preprint of the manual is done. Leslie has offered to speak about his system at this summer's TUG get-together, which is reason enough for everyone to attend! We hope to be able to put the entire thing on the distribution tape, and distribute the manual, before the TUG meeting.

We heard from Dr. Wolfgang Appelt of the Gesellschaft für Mathematik und Datenverarbeitung, a research organization in Germany. He reports that a meeting for TEX users in Germany was attended by people from more than a dozen installations, many of which were already running TEX82. We hope that this group will turn into the German chapter of TUG. Interested parties should contact Dr. Appelt at GMD, Postfach 12 40, Schloß Birlinghoven, D-5205 St. Augustin 1, Bonn. They are producing TEX82 output on a Perq/Canon printer, and are looking into the possibility of interfacing to the Hell-Digiset typesetter. Also, they hope to make use of Frank Liang's hyphenation-pattern-generator (which should be available on the distribution tape soon) to enable TEX82 to do German hyphenation.

I've heard from quite a few people with Imagen and Symbolics printers. It looks like prices are coming down a little, and I hope the new entry in the marketplace, the Lasergrafix machine based on the Xerox 2700 engine, will help continue the trend. I don't know any more about Lasergrafix, except that the folks at Texas A&M report that they are promising support for TEX output. Meanwhile, our HP 2680 has been connected to our Ethernet, and we hope to start seeing real TEX output from it soon.

On the phototypesetter front, things are heating up. The TEX project is looking for a new machine more appropriate for our needs than our Alphatype CRS. Autologic is promising that we'll be able to download our fonts to a Micro-5 if we buy one and write the code ourselves, but recently Compugraphic has become re-interested in TEX (perhaps taking the cue from HP and Apple). If they decide they're willing to let the 8600 be made Metafont-compatible, we'll be facing a hard decision. This has to get straightened out pretty quickly—we want to print the TEX manual on the new machine. Stay tuned.

A number of corporations around the country have inquired about the availability of TEX experts to hire, on a full-time or consulting basis. So far, I haven't been able to help them very much, but if you fit the bill, you might want to get in contact with me for more information.

Some people have encountered trouble with square roots on TEX82. Recall that TEX82 makes a different assumption about the position of the square root symbol within its box than TEX80 did. To get acceptable output with TEX82, you'll need up-to-date *symbol* and *math extension* fonts. (The reason for this change was to improve the alignment of the surd with the top rule, even in the face of rounding on different raster resolutions.)

Another item of note is that we're trying to improve the efficiency of TEX82. I should point out that it's none too shoddy as it is—the compiler being used in its development is very stupid when it comes to code generation, but we find that TEX82 is about the same speed as TEX80 in Sail, and it's quite a bit better than the old Pascal version. A good optimizing compiler, coupled with a reasonable disk I/O system should produce a fairly zippy TEX. In order to squeeze out lurking inefficiencies, however, we're conducting an experiment: A number of compilers support some form of statement execution profiling, where the user is told, after his program runs, how many times each statement was executed. With the aid of a little system wizardry, we're compiling such counts for TEX82, with the results cumulative over all users of TEX on the SU-AI computer. See if you can guess which Pascal statement was executed over 80 million times during the 2900 or so times TEX was run over the last 6 weeks (answer below). We're not just playing games, of course; we're trying to find where we should spend our time trying to speed up the code, by un-rolling loops, using macros instead of procedure calls, etc. Finally, there will be index entries in the published TEX code indicating the modules that are part of the inner loop, so that they can be hand-optimized on various systems if anyone cares to do so. The answer to the question above is `CSPTR:=0` at the beginning of the `GETNEXT` procedure. By the way, of the almost 20000 lines of code in TEX82, 752 of them were not executed in over a month. Many of these statements are fatal error messages.

TEX on the HP-1000

Irene J Bunner and John D Johnson
JDJ Wordware, P.O. Box 354, Cupertino, CA 95015; (415) 965-3245

TEX82 for Hewlett-Packard HP-1000 Computer Systems has been implemented by JDJ Wordware. The code was obtained from Donald E. Knuth and the WEB programming system was used to perform the modifications required to make TEX82 run on this mini-computer. It is interfaced to an inexpensive ($500.00) Epson MX-80 dot matrix impact printer. A combination of automatic translation and hand editing have been used to create a set of fonts usable by this device.

HP-1000 mini-computers are 16 bit architecture machines with a real time, multiuser operating system. Programming is supported by a screen mode editor, a Pascal compiler, and a symbolic debugger. The standard addressing accommodates programs up to 64K Bytes, much too small for TEX. Extended addressing, supported by a virtual memory system, permits larger data sizes. Program segmentation permits larger code sizes. Heavy utilization of these two features enable us to fit TEX82 on the HP-1000.

The Epson MX-80 dot matrix impact printer is used in its high resolution graphics mode. In this mode it has a horizontal resolution of 120 dots per inch and a vertical resolution of 144 dots per inch. The dot size is larger than this resolution resulting in adjacent dots overlapping. A program was written that reads in the DVI file produced by TEX and special pixel files. It constructs the raster image for a complete page in memory before outputting to the printer. Interweaving rows of dots is required during this outputting. Eight horizontal rows of dots are printed, the paper is advanced by $\frac{1}{2}$ dot width then the eight interweaved horizontal rows of dots are printed. About six minutes are required to print a full $8\frac{1}{2}$ by 11 inch page.

Special raster descriptions for the fonts are required so the definition of pixel files was changed for this implementation. One of the changes is to a vertical raster instead of a horizontal raster. Even though the print head moves horizontally, the MX-80 is basically a vertical raster device. The eight dots it prints in parallel are arranged in a vertical row. Raster descriptions are changed from a 32 bit organization into a 16 bit organization to accommodate the word width of the HP-1000. The biggest change is to low resolution. Metafont was unavailable so the first approximation to the required pixel files was created by writing a program to automatically translate standard pixel files into the new format. This progam reads in a 240 dot per inch PXL file and produces a 120 dot per inch vertical raster pixel file. The small difference in the printer's horizontal and vertical resolution is ignored. Horizontal dimensions are accurate while vertical dimensions are compressed.

Translation to low resolution pixel files is achieved by mapping four pixels from the input file to a single pixel in the output file. If two or more of the input pixels are set, then the corresponding output pixel is set. Characters appear too black in these automatically created pixel files. The large dot size compared to the resolution causes the strokes to be too wide. Better looking characters are obtained by hand editing the fonts. A program was written which translates a pixel file into an ASCII file that contains a picture of each character built up using at signs and spaces. The system editor is then used to modify these pictures. Finally, another program translates from the ASCII file back into a pixel file. Hand editing a font requires several hours and several iterations.

Porting TEX to a small machine turned out to be quite a challenge. The first step was to bring up the Tangle program so that the WEB sources could be translated to Pascal. This was not too difficult. Porting TEX itself was much harder. Its data is too large, its code is too big and it has too many files for the standard Pascal runtime I/O system. The large data problem is handled by utilizing extended addressing which requires moving large arrays to the heap. WEB macros in the change file are used to do this. For each large array, the change file creates a pointer to the array as well as an entry in the initialization code which does a "new" on this pointer. Other macros change all the old accesses to the array into pointer dereferenced accesses.

The large code problem is solved by segmenting. This was very difficult because TEX is too large to be handled by the automatic multilevel segmenter. Special segmenting is used which allows the passing of value parameters between segments. Luckily, TEX uses only value parameters so no changes to parameter lists are required to accommodate the segmentation. A program was written to aid in segmenting TEX. First it reads and parses the Pascal source and builds a procedure call graph. Then this program is used interactively to place procedures in various segments. The goal is to minimize cross segment calls subject to an overall segment size restriction. Once the user is satisfied with the segmentation, this program builds the required source files to pass to the compiler as well as the segmentation files to pass to the linking programs. Several iterations were required before acceptable segmentation was achieved.

TEX declares more files than the standard Pascal runtime I/O system can handle. The main problem with this I/O package is its inability to use file buffers in the extended addressing area. To overcome this problem, a special runtime I/O system was written which can put file buffers in the heap. WEB macros are used to map standard I/O statements into calls to these special I/O routines.

TEX82 is now available on the HP-1000 family of computers. Currently, typesetting performance is roughly 1500 words per minute. This page was formatted using an A700 Series HP-1000 and output on an Epson MX-80 printer.

Unix TEX Site Report

Richard Furuta and Pierre MacKay†
Department of Computer Science
University of Washington

We'd like to take this opportunity to introduce ourselves to the *TUGboat* readership and report on new developments. We took over the Unix TEX site coordinator duties from Robert Morris in November, 1982. Since then, much of our TEX time has been devoted to TEX82, both on Unix and on Tops20. We have submitted a separate article in which we describe TEX at our particular site in more detail, but here, let us turn our attention to the state of TEX on Unix machines.

We are pleased to report that TEX82 is currently running on VAX/Unix, 4.1BSD, thanks to the efforts of Pavel Curtis of Cornell University and Howard Trickey of Stanford University. New versions of TEX82 have been coming up on Unix within a couple of weeks of their announcement at Stanford, so we are pretty much up to date. Howard and Pavel have written an article summarizing their experiences in porting TEX82 to Unix which follows this report. Details on how to obtain TEX82 for your own VAX are below.

Unfortunately, we have not heard of any versions of TEX82 for other flavors of Unix. Some interest has been expressed in porting TEX82 to the Sun workstation (running 4.2BSD) and some preliminary work is in progress. We have also heard from a person who wants to port TEX82 to a PDP-11 running 2BSD Unix, but as far as we know, he has not yet started working actively on it. If you're doing a port to one of these other machines or versions of Unix, please keep us posted.

We are trying to handle as much of our correspondence as possible through electronic mail. We can be reached from the Arpanet, from CSNet, and via uucp. Our Arpanet and CSNet addresses are Furuta@Washington and MacKay@Washington. On uucp, it's

...decvax!microsoft!uw-beaver!uw-june!furuta

and ...uw-june!mackay. An alternate uucp route is ...ucbvax!lbl-csam!uw-beaver....

†Our work is funded, in part, by a grant from Northern Telecom.

We also are maintaining an electronic mailing list, Unix-TEX@Washington, for discussions between Unix TEX sites. If you want to be included on this list, send Arpanet, CSNet, or uucp mail to Furuta.

TEX82 for Berkeley Unix, 4.1BSD

TEX82 is now available for sites running Berkeley Unix, version 4.1. We are now making available a "beta test" distribution which includes sources for TEX and WEB, libraries, fonts, and whatever device drivers we can obtain. At the time of writing, we have drivers for the Versatec, thanks to Carl Binding of our Department, and for the Imagen laser printer, thanks, again, to Pavel Curtis. We also hope to have a Symbolics laser printer driver in place before the distribution begins. If you have DVI 2 drivers for other devices, please send them to us and we'll happily include them in future distributions. Our fonts are now in the PXL format, 128 characters per font (the older arrangement), and with magnifications of 1.0, 1.2, and 1.3 included for most entries.

The present distribution will be the latest version of TEX82 available to us (presently 0.95). Note that versions before 1.0 are considered to be pre-releases so the language they define is subject to change. If you want to wait for 1.0, please let us know when you request your tape.

As the implementation uses a modified version of Berkeley's *pc* Pascal compiler, we will only be able to provide a complete distribution to those sites with source licenses for 4.1BSD. We will, however, try to accomodate those with binary licenses for 4.1BSD. Please make sure that you indicate clearly whether you have a source or a binary license for 4.1BSD.

The complete distribution presently occupies something on the order of 11 to 12 megabytes of disk storage on our machine. Of this, about 3 megabytes is used to store the fonts, 1 megabyte for the modified *pc* sources, and the remainder for TEX82, WEB, device drivers, and other parts of the system. Of course, not all of this needs to be retained on-line. The frugal site may be able to get by using only perhaps a half of this amount of disk space.

Tapes will be in tar format, blocked 20, and written at 1600 bpi, unless otherwise specified (we can also write 800 bpi).

To order, send a check for $50 (U.S. Funds) made to the University of Washington, documentation of the type of Unix license you hold, and your address to:

Richard Furuta
Computer Science, FR-35
University of Washington
Seattle, WA 98195

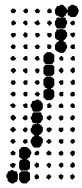
We would appreciate it if foreign sites could increase the amount of their check as appropriate to pay the added postal costs necessary for mailing the tape. We'd prefer not to receive purchase orders. Please contact us if that causes you problems and we'll try to set up alternate arrangements. If you have a CSNet, Arpanet, or uucp mail address, please include it—we'll add you to our mailing list.

The lower limit of low-res fonts

As noted above, we run TEX on a DECSystem 2060 as well as on the VAX. Since we do have use of the 2060, we occasionally run off PXL fonts for other Unix sites with METAFONT. (METAFONT is written in the SAIL programming language and so does not run on Unix machines. Indeed, use of METAFONT is pretty much limited to DECSystem-10's and DECSystem-20's.) We have recently received several requests for very coarse-grained PXL fonts at resolutions of 100 dots to the inch or less, which prompts a few observations.

100 dots to the inch is just acceptable for upright fonts, but rather chancy for slanted or italic fonts, and anything significantly lower is hopeless. There is not much point in asking for a PXL font at 0.36 magnification (72 dots to the inch) unless you simply want it as a rough guide for a pixel-editing program. At least half the characters are unidentifiable at that resolution. In the long run, the best results at any resolution under about 300 dots to the inch will require a stage of pixel editing. METAFONT is astonishingly good at making the critical decisions about difficult regions of a font character, but it is not perfect. At 100 dots to the inch, it is often overstrained. This point can best be appreciated by looking at an example from CMI10, produced at .5 magnification. The slant brings out all the worst problems of aliasing, and in this case produces a comically misleading result.

```
Char '135   Right bracket
Pixel Width 6 Pixel Height 15
X-offset 1 Y-offset 10
Raster Pointer 844
Width 0.27777 (2.7777 pt.)
```



(modified output from David Fuchs's PXLTYP)

The upper horizontal of the right square bracket is caught at a point where METAFONT's *rounding* operation shifts the entire lot of pixels one position to the right and as a result the bar ends up on the wrong side of the character. A pixel editor can fix this better than METAFONT can. There is no point in complicating the basic METAFONT character descriptions with statements that will handle such an extreme case. Even after editing, the italic and slant fonts (CMSnn, CMInn and CMTInn) will still look pretty crude. But most of the characters in roman, MATHEX and even symbol fonts are not too bad. Anyway, until the price and availability of 200 pixel/inch bit-mapped displays comes to be very different from what it is now, we must live with what we can get.

* * * *

Porting TEX to VAX UNIX

Pavel Curtis and Howard Trickey

Over the past few months, the authors have independently ported TEX82 to run under Berkeley's VAX/Unix† (version 4.1BSD). One of us (Pavel) subsequently merged our ideas to produce the VAX/Unix distribution of TEX announced in this issue's Unix TEX Site Report. Both implementations required small changes to Berkeley's *pc* compiler. This note describes the problems encountered in making these changes and in bringing up TEX itself.

TEX porters are aware that the capabilities of the available Pascal compiler will almost totally determine how difficult the port will be. Unhappily, the *pc* compiler has more deficiencies than one might wish. This is somewhat understandable, since it was designed for instructional use and not for the preparation of production software. On the plus side, the documentation was good enough to allow us to anticipate most of the problems.

†Unix and VAX are registered trademarks of Bell Laboratories and Digital Equipment Corporation, respectively.

Also, we seem to be the exception to "Fuchs' Law" that TEX uncovers at least one compiler bug for every compiler tried—our problems were caused by deficiencies, not bugs. This meant that there were few unexpected problems, and we were able to do the initial port in surprisingly little time (one or two weeks).

The most significant shortcoming of *pc* was also the one major place where Professor Knuth departed from Jensen and Wirth's Pascal and from the proposed standard: a default clause in the **case** statement. It has been suggested that the best way to handle this problem is to write a program to translate the **case** statements used in TEX to **case** statements without default clauses. This is actually fairly difficult to do properly, since it entails discovering the type of the case expression (character, integer, or integer subrange) and then forming a succinct test for the gaps between the expressions that actually occur as case labels. Since the *pc* compiler already generates code to check for an unmentioned value, the best possible translation would result in doing this work twice. The old PTEX implementation made a half-hearted attempt, by using an editor script to collect most of the information needed, but one still had to do the translation manually. We considered it imperative that bringing up a new version of TEX be fast and easy, enabling all installations to track bug fixes and added features.

The last thing one normally resorts to in order to circumvent a language deficiency is that horror of horrors: compiler modification. No one except the author or maintainer of a compiler really understands the implications of a given change to the code, so it is easy to introduce obscure bugs. Also, the compiler modifier has to be wary of new versions of the compiler from its maintainer. Having said this, we must now say that we both came to the conclusion, in this case, that compiler modification was the answer. Examination of the then current code showed that *pc* was already trapping the default case in order to have an error message printed. It was immediately obvious how to give control to another statement instead, and it was hard to see how any bugs could be introduced. Since so few changes were needed, the new-compiler-version problem was not too daunting, so the deed was done.

Compiler modification was definitely *not* the answer for the next problem, memory packing. To avoid wasting a large amount of memory, we needed to define the types *quarterword* and *halfword* to contain 8 and 16 bits, respectively. It is not possible to convince *pc* to pack 0..65535 into 16 bits or 0..255 into 8 bits. While it might not be too difficult to get the memory allocator to do this, the code generator could easily be a different question. Fortunately, the compiler *will* pack $-128 .. 127$ and $-32768 .. 32767$ into the proper sizes and TEX has been written so that these ranges can be easily used in the all important *memory_word* record. A similar problem occurs with the non-character *byte_files*, but here one must be careful that the bits actually written to the files look like the range $0 .. 255$ has been used (otherwise, device independent output files and TEX font metric files can't be transported between sites). So, for example, instead of writing a value of 200 to a file, we write $200 - 256 = -56$.

In spite of the above, we would still waste a tremendous amount of memory if we could not find a way to fit the type called *glue_ratio* into 32 bits. Usually a *glue_ratio* is made the same as **real**, but *pc* puts **reals** into 64 bits, and has no provision for any kind of "short" **reals**. Fortunately, the VAX hardware has a 32-bit floating point data type, so the following trick could be used: Pascal is told that a *glue_ratio* is a (32-bit) integer, but we really store floating point numbers there; external procedures (written in C) are then used to convert back and forth between those pseudo-**reals** and *pc*'s 64-bit variety.

The rest of the changes necessary were more conventional. There were a few places where TEX uses variables names that *pc* uses for special things (e.g., *input*, *text*, *time*, and *date*), but the WEB system provided an easy way to fix them. The *pc* runtime system doesn't provide for recovery from file opening errors, or for closing a file immediately. It was easy to modify a few of the run-time library procedures to make files behave the way TEX assumes they will. Specifically, the following changes were made:

- Opening an input or output file that isn't there causes the "end-of-file" flag to be set true or false, respectively, rather than being a fatal error.
- Reading past end-of-file is silently ignored, rather than being a fatal error.
- The first **get** from the terminal is now ignored.
- Provision is made to explicitly close a file. The *pc* run-time system as supplied simply closes files on scope exit.

The alternative to changing the run-time library would be to do all input/output with external C procedures, but this would have meant changing every place in the code that used **get**, **put**, **read**, **write**, tests for end-of-file, etc. Macros would have

helped to some extent, but it is hard to see how to convert a file variable argument into information that conveys the type of the file. Also, one has to be on guard for a varying number of arguments. Adding onto this the need to simulate Pascal's conventions about text files, it just seemed easier to make the run-time library changes.

One particularly irritating thing about *pc* is that it doesn't accept keywords which are written using uppercase characters, even though that is required by the standard! (More precisely, there is a flag allowing for upper case keywords to be used but this switch also turns off all of the other extensions Berkeley *did* make to the Jensen and Wirth standard.) There were a number of places in TANGLE that had to be changed in order to prevent the conversion to uppercase. A subtle bug can show up if one is not careful in making this change: when TANGLE does constant folding, it looks for keywords like MOD and DIV in the output buffer. The easiest place to convert to lowercase is before these comparisons, and if the comparisons are not changed an incorrect program may result.

Another problem we had with TANGLE was getting it to put the statement

```
#include "ext.h"
```

in the Pascal output. This statement is needed to tell *pc* about externally compiled procedures. The double quote marks in the statement could not be emitted by TANGLE, and there was no way to force the '#' to be just after a line break. We introduced primitives into WEB that eventually evolved into the "verbatim Pascal string" and the "force Pascal line break" primitives now in the standard WEB language.

The amount of time it takes to compile TEX is a major annoyance. Using an otherwise unloaded VAX 11/780, it takes approximately an hour of real time to create a runnable TEX from the WEB source code. Back in the real world, one of us has experienced upwards of a five hour wait when other people were using the machine. Now that the port is complete, this is no longer such a bother. On those infrequent occasions when a new version of TEX is being installed, one can simply have the compilation done in the middle of the night.

For regular use, it appears necessary to have a version of the program which has the PLAIN macros preloaded. Unfortunately, UNIX is not as amenable to this idea as some other systems, notably TOPS-20. We were able, however, to retrieve a program from the net.sources group on USENET which will do the trick. The program, called "undump", takes a core-image and the executable file which produced it and constructs a new executable file incorporating all of the data areas of the core-image. After some involved machinations and close perusal of the order of file opens or closes within TEX, a procedure was derived for preloading any macro set. This saves an average of 30 seconds or so of start-up time over the non-preloaded version.

Some statistics regarding the port are in order. The production version compiles into about 235,000 bytes of code. This doesn't include the large data arrays, which need not be allocated until runtime. At least 250,000 bytes of data area are needed in addition to the code to get a usable TEX. In a memory-rich system it is not unreasonable to allocate 500,000 bytes for this purpose. It is possible, as mentioned above, to save a version of TEX that has its data areas preloaded with fonts and macros, but one wouldn't want to have too many such versions; they are on the order of 750,000 bytes long each. As for running speed, a sample six-page paper (single-spaced, 10-point type, using a page of macros and a couple of fonts on top of PLAIN) took 44 cpu seconds to format using the preloaded-with-PLAIN version of the program. For comparison, the same paper took 37 cpu seconds on the SAIL DEC-10 computer used to develop TEX.

With regard to output drivers for UNIX, there are not many yet available. One of us (Howard) has a driver written in C for the Xerox Dover printer "press" format, adapted from a program that handled the old (version 1) DVI files. The new program handles either version. Pavel has written one for the Imagen/Canon ImPrint Laser Printer, also in C.

Looking back on the port, it must be said that the combination of the WEB system and Professor Knuth's careful coding has resulted in quite a portable program. It now takes less than a day to bring up a new version of TEX and check that it passes the nefarious TRIP test. It is rarely necessary to touch the changes we've made, since the new features and bug fixes usually don't occur in the "system-dependent" parts. As a result, TEX is a program that can easily stay up to date and incorporate the latest bug fixes at many sites on many hosts. That, combined with the care taken to ensure that the output will be the exactly the same from host to host, means that we have a solid basis for portable documents.

TEX at the University of Washington: Tops-20, Unix, Versatec, and the Monolithic

Pierre MacKay and Richard Furuta†
Department of Computer Science
University of Washington

We would like to take this opportunity to describe the TEX environment in the Department of Computer Science at the University of Washington.

We presently run TEX82 on a DECSystem-2060 and on a VAX11/780, located in our Computer Science Laboratory. Our primary output device has been an elderly Versatec printer/plotter which prints on 11" wide paper. The Versatec is shared by TEX, **troff**, and Scribe users. We expect to receive a laser printer shortly.

Since we have the use of the DEC-2060 and of the Arpanet, we have had the opportunity to copy working versions of TEX82 directly from the SCORE machine at Stanford and to prepare our documentation using TEX82 even before the problems involved in creating the Unix version of TEX were entirely solved. We therefore gave a high priority to putting together drivers for our Versatec. This effort was complicated by the fact that while the version 2 **DVI** files were created on the DEC-20, the Versatec was connected to the 780's Unibus, and we had no controller to drive the Versatec directly from the DEC-20 in any case.

We decided to begin by modifying a magnetic tape file transfer system which we had developed for TEX80 output. Although slow, this system had the distinct advantage that it supplied bit-mapped rasters to the Versatec at an even rate and avoided the extreme variations in toner density which can appear if the paper races past the print head to skip over white space. Our first, very provisional output driver, **DVItoVRT**, did all the necessary translation for the Versatec, and wrote full-width Versatec rasters, ten to a block, on magnetic tape on the DEC-20. We then read the tape on the VAX11-780 and printed the rasters on the Versatec. Generally, we could fit twenty or thirty pages worth of information onto a 2400' tape reel. **DVItoVRT** still exists, but it has never been entirely finished. It boasts the version number 0.8, but that may be something of an exaggeration. In any case it works. It is dreadfully slow, but it works, though only for vertically oriented output on roll-paper.

The **DVItoVRT** driver is based on the **DVItype** processor which was made available concurrently with TEX82, and it preserves all possible elements of **DVItype**. The addition to **DVItype** which may be of most general interest is the part that reads the **PXL** fonts. **DVItype** gets all the font information it needs from the **TFM** (TEX Font Metric) files associated with each font, but a driver program must take account of the structure of the font itself. Our programs read **PXL** files as described by David Fuchs in *TUGboat*, Volume 2, No. 3, pages 8-12; they allow for scaling either in the TEX input itself, or at the time when the **DVI** file is interpreted for the output device. The most practical use of scaling is the latter. As a general habit, we format all TEX material at true size and expand it to a 1.3 magnification in the driver program. A reduction of 77%, which is available on a several photocopying machines, restores true dimensions and very much improves the apparent sharpness of each typeface.

Since **DVItoVRT** does the necessary bit-pushing to create a Versatec raster in as nearly standard Pascal as we can manage, it is infuriatingly slow. Processing and writing a page of **WEB** output to tape takes about 30 seconds of elapsed time on a moderately loaded TOPS-20 system, which is enough to discourage all but the most devoted user. Fortunately, in December, BNR Inc. lent us a Monolithic Systems Corp. processor, which is intended to be used as an intelligent controller for the Versatec. (For a description of earlier uses of the Monolithic and of the support programs for earlier versions of TEX, see Phil Sherrod and Alan Wright. "TEX Support Programs." *TUGboat*, Volume 2, No. 1, pages 17-19.) Except on the rare occasions when its font memory fills up, and a new font has to be written over one of the old fonts, the Monolithic allows us to drive the Versatec just about as fast as it can physically be driven. Sherrod and Wright reported some improvements that they made at Vanderbilt University, but we have not felt the need to consider those yet. At present we are content with a two-step process based on software developed at Stanford some years ago involving (1) translation from **DVI** format to an intermediate work file (**VER**) format, and (2) shipment of the **VER** command and data file to the Monolithic processor. We did modify the format of the **VER** file to allow us to use the same second pass with both DVI 1 and DVI 2 and we updated the software to use **PXL** files. These intermediate work files (**VER** files) are queued and deleted automatically after use. This can be an inconvenience, since they must be totally regenerated again if needed, but they are so large that we cannot afford to leave them on the disk.

†Our work is funded, in part, by a grant from Northern Telecom.

We wrote a new program, **DVI2VER**, to handle the first pass for DVI 2 files. This program, based (like **DVItoVRT**) on **DVItype**, is now fully debugged for vertical output on continuous roll paper. We have not yet decided what to do about rotated output on fan-fold paper. The one thing we do not want to do is go back to bit-pushing in standard Pascal. The regular use of 1.3 magnification and **1300PXL** fonts makes fan-fold paper rather impractical in any case. We have left the necessary hooks in the program so that rotated output could be added, but there has been no decision when or whether we will hang anything on those hooks.

At present, the intermediate work file makes no attempt to conform with BigEndian conventions. It is very much an artifact of a 36-bit environment. We have worked out a coding system which would fit very nicely into a 32-bit environment, but at present we do not know of any users who could take advantage of it. The 32-bit coding would actually result in a slightly larger work file if the TEXed material included a large number of tall characters (e. g., most of the characters in **MATHEX** fonts), but it would result in a slightly smaller work file on a page with short rule segments. In any case, we will soon need to modify our present **VER** file format again to allow for the new 256 character **PXL** file format and to take advantage of the larger number of fonts which may be defined in TEX82.

Once we had our output from version 2 **DVI** working nicely, we were left with one more anomaly: our METAFONT still produced proof-mode output using version 1 **DVI**. We rewrote our copy of **MFOUT.SAI** for the new conventions, and now our entire TEX environment is consistent with TEX82. (When you try to produce postamble values to simulate TEX output, you have to remember the discrepancy between the TEX82 *scaled point* which is $1/(2^{16})$th of a printer's point, and the **PXL** font FIX, which is $1/(2^{20})$th of the design size. Otherwise you get some very strange dimensions.) We have also modified our METAFONT to use DEC's GIGI terminal as the "drawdisplay" output device rather than Stanford's Datadisc terminal. At present, our principal concern with METAFONT is the production of a set of *Naskhi* characters for an Arabic Script enhancement to TEX82.

* * * * * * * * * * *

VAX/VMS

Monte C. Nichols
Sandia National Laboratory
Livermore, California

Last issue I reported that we expected to see TEX82 running on VAX/VMS systems by late 1982. Boy was I wrong. One problem has been the wait for the new VMS version of Pascal, although that is no longer a problem for most of us. Additionally, the group at Oregon Software has been unable to find spare time to devote to bringing up TEX82 (note that their work on TEX has been on an uncompensated basis — Yes, you can get something for nothing, Virginia, it just takes longer). Several others have been working on the implementation but have been thwarted by other problems. David Fuchs (Stanford) has just reported success using VMS 3.2 and Pascal 2.1; see page 14. Hopefully the new version of Pascal will soon reside on many VMS machines (it was delivered in Italy long before it was available here on the West coast) and we will all be in business.

There should be several other articles in this issue written by VMS folks. In addition, I have received several letters very recently that contain leads for further articles of interest to the VMS community. Hopefully they will find their way into the next issue of TUGboat.

David Fuchs has very kindly made available many more fonts in .tfm and .pxl format. They are for 200dpi devices and the .pxl files exist in magnifications of 1000, 1100, 1200, 1300, 1400, and 1500. These files are presently being unpacked from tape and will be sent soon to Oregon Software where they can be distributed to you. These fonts will work for the most recent version of TEX now being distributed by OS as well as the version of TEX82 that we all long to see. In addition, I still have the files sent me courtesy of AMS which supplement those from D. F. but which exist only as METAFONT files and thus need to be "treated" by the METAFONT program on a DEC 10 or 20 machine.

I want to again encourage anyone who develops a spooler for devices other than those presently available on VMS to send them to me or to Oregon Software so they can be made available to the rest of the VMS community.

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

TEX AT CALMA R&D

David Krapp

At Calma R&D in San Diego, we produce approximately five thousand pages of technical documentation every six months. The majority of this work is in the form of user manuals for our computer-aided design systems. Until now, the typesetting of these books was done by a commercial typesetter, at considerable expense to the company. The typeset copy is then mailed to our corporate publications office for printing.

Late last year, after a brief familiarization and testing period, a decision was made to use TEX to produce camera-ready copy of all of our documentation. Lower costs, faster typesetting time, and keeping control of the entire production process in-house were major factors in this decision. The conversion to TEX was to occur within our normal delivery schedule, necessitating very fast and concentrated work to produce this large volume of material.

Work began immediately on the creation of a suitable macro package that allows us to create many of the complex layouts used in our books with relative ease. Lynne Price, TEX wizard from our Sunnyvale office, was (and is) instrumental in creating this package, as well as advising me on debugging, creation of new macros and generally soothing frazzled nerves. We now have an excellent table of contents generator, with an index generator just around the corner. Most of our problems resulted from trying to duplicate the current typeset layout of the books, instead of simply redesigning them in a way that's easy for TEX. Happily, we have not encountered anything that absolutely *can't* be done (somehow).

Our macro package was designed so that our typists could easily insert control sequences in the text of the manuals. I then review the manuals to insert any complicated layouts, such as table alignments and the syntax descriptions of computer commands, for example:

$$FOO\left\{\begin{array}{l} \text{P P P} \\ \text{P var} \\ \text{VPA P ws} \\ \text{MAX Xs} \qquad [[\text{vec}]\ [\text{ADD}]\ \text{OTR lst}] + \\ \text{MAX Ys} \\ \text{MAX Zs} \\ \text{PIR IS}_{arc,con} \\ \left[\text{MAG}\left\{\begin{array}{l}\text{P P}\\ \text{ALL}\\ \text{UPP}\\ \text{DWN}\end{array}\right\}[\text{IS}_{pinsrf}]^*\right] \end{array}\right\} C/R$$

(One book had nearly 400 such syntaxes.) Finally, I debug and run the input file to generate the manual.

In February of this year (two months before our production deadline) we switched from the original TEX to the newer version (using TFM files). We now run TEX on both our VAX-11/780 and 730 under VMS, with output available on a Versatec V-80 (used for draft copy) and a Symbolics LGP-1 Laser Graphics printer (for camera copy). As this is written, we are pushing hard to meet the deadline, with some 3500 pages completed in draft form and about 1700 in final, camera-ready copy.

As a tool for high-volume production, TEX has proven itself to us at Calma R&D. It *does* require quite a bit of familiarization and practice, but the results so far have been worth it.

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

TEX AT TEXAS A&M UNIVERSITY

Norman Naugle and Bart Childs

The best news about TEX at TAMU is the arrival of a QMS Lasergrafix 1200 printer and the ability to produce TEX82 *dvi* files successfully on a Data General MV8000. Although we have been running TEX using VAX/VMS systems, the lack of a reasonable output device has hampered our progress. Until now, our output devices have been Printronix and Trilog printers—suitable for overhead projectors. These printers use a driver (modified) and 200 bpi fonts obtained from Oregon Software.

Initially, the QMS printer will be used in the graphics mode to paint pages, but development of a driver is under way. The printer will have TEX82 fonts at 300bpi in ROM plus down-loadable fonts. It is anticipated that the printer will digest *dvi* files a page at a time, i.e., from *bop* to *eop*. This strategy should allow the printer to approach its 12 page per minute capacity. Hooks for embedded graphics are possible and are being considered.

Our plan is to offer TEX82 on all major computers on the TAMU campus. This includes VAX, Prime, Amdahl, DG, and Cyber. Available output devices will include Lasergrafix, Versatec, Trilog, Printronix, Xerox 9700, and Linotron 202. We are hoping for something like an Autologic APS-5 in 1984.

Although we can't say TEX correctly since we're from TeXAS, we can say **Thank you!** to everyone in the TEX community for their generous support to an isolated outpost.

* * * * * * * * * * *

"small" TEX

* * * * * * * * * * *

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

The number of small TEX implementations is growing by leaps and bounds. In addition to TYX's TEX-in-C, Tom Hickey's Apollo implementation, and my HP3000 version, there is an implementation on the HP1000 and a possible Apple LISA version

The people at TYX inform me that they are busy bringing up TEX82 in a C version which will run on all of their machines (ONYX, PDP11-44, PLEXUS) as well as on 68000–based systems. My understanding is that this is not a WEB transport, but a rewrite in C (as was their TEX80).

Tom Hickey has already transported TEX82 to the Apollo, and reports that it was an easy task. Tom notes that the new version is faster, though this may be partially due to the new Apollo processor. He is using an Imagen printer, and a report on the activities at OCLC appears elsewhere in this issue.

TEX82 is in the works for the HP3000. See the site report below. (Again this issue, this page was printed on the HP2680A Laser Printer.)

There is a new implementation of TEX82 for the HP1000 from JDJ Wordware, Cupertino, CA. They have a driver for the Epson MX-80, as well as for the Imagen. See their report elsewhere in this issue.

Alas, there is still nothing for the 8-bit micros, but there is good chance the Apple LISA will soon support a version of TEX. See David Fuch's article in this issue.

HP3000 SITE REPORT

Lance Carnes

TEX82 is struggling to life on the HP3000! Transporting the WEB system was unbelievably painless, and bringing up TEX82 is still before me. What follows is a brief glimpse of what has been done so far.

After unloading the WEB sources from tape, it took me exactly three tries to successfully compile and run TANGLE, and have it output itself in Pascal! Stunned, I decided to press my luck and try to bring up WEAVE. This came up the first time, mostly because the change file is almost identical to that for TANGLE.

Most of the problems encountered so far had to do with the change files. After trying to create them *ad hoc* based on the listings, it occurred to me combine all of the system dependent modules (identifiable by their index reference @ ↑**system dependencies**) as a first approximation change file. This works out well, since you need only read and modify the lines in this file.

TEX82 should be up in the next few weeks, and I will be able to report on its performance at the July meeting.

TWO BUGS IN TEX80-in-Pascal (or FLOGGING A DEAD HORSE)

Lance Carnes

For those of you still hacking away at TEX80 the following bugs have been identified. Both are caused by variables which contain uninitialized values.

The first bug occurs in the procedure **hyphenate** where exception lookup is done. In the August 1981 listing of TEX, section 470, the following appears:

```
for i := j + 1 to n do
        truncword[i]:= shortAsciiNull;
```

And then in section 472 there appears:

```
for i := 1 to hashlength do
        hash := hash*16 + truncword[i]
```

The problem I experienced was that *hash*, which is computed from the first *hashlength* characters, became a negative number. It turned out this occurred because for $n < hashlength$ the last $hashlength - n$ places in the array *truncword* were not assigned a value. Since *truncword* is a local array, its initial value is just whatever garbage was left on the stack, and if the garbage happened to be a negative 32-bit integer, *hash* became negative also.

The fix for the bug is to replace *n* with *maximumDistinctionLength* in the code from section 470 shown above.

The second bug turned up when a source file had the following: `\xdef\junkie{}`. The symptom is an array index violation in Pascal runtime. The bug occurs in the procedure **scantoks**.

Refer to Section 194 in the August 1981 listing of TEX. The array index violation occurred in the third to last line:

```
link(q) := 0; { delimit token list }
```

It turns out *q* is not set in the case of an empty definition, i.e. {}, and since it is a local variable, it just contains whatever garbage is on the stack.

The fix for this bug is to replace *q* with *p*. A workaround is to use `\xdef\junkie{{}}`.

The moral of this story is: if you are going to use a Pascal variable be sure you have previously assigned it a value.

* * * * * * * * * * *

Fonts

* * * * * * * * * * *

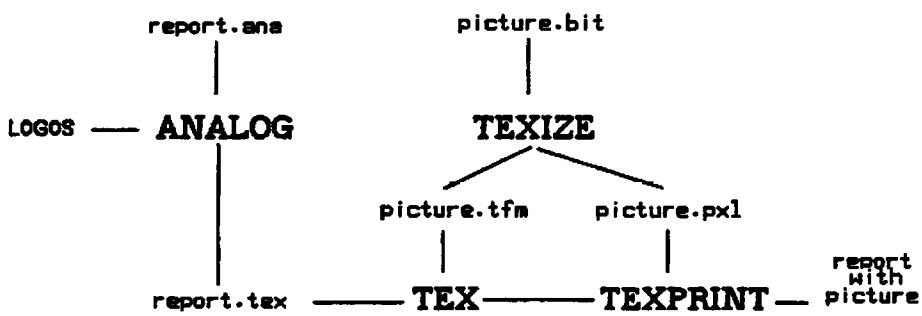
Pictures Are Just Big Letters

Scott B. Guthery

Schlumberger

We have found that a handy way to include pictures coming from raster graphics software in TEX reports is to transform the pictures into a characters in a TEX font. Using two excellent articles by David Fuchs we wrote a program called TEXIZE that accepts rasterized pictures coming from graphics packages on our time-shared system or from personal workstations and builds TFM and PXL files which contain the pictures. In the TEX document one then simply switches to this font, calls out the character corresponding to the picture one wants to display and places it where one wishes on the page.

The logo at the beginning of this article was prepared on one of our workstations as was the following diagram which shows how we generate reports containing pictures from our LOGOS database system:



David Fuchs, *TEX Font Metric Files*, TUGBOAT, Volume 2, Number 1 (February, 1981).

David Fuchs, *The Format of PXL Files*, TUGBOAT, Volume 2, Number 3 (November, 1981).

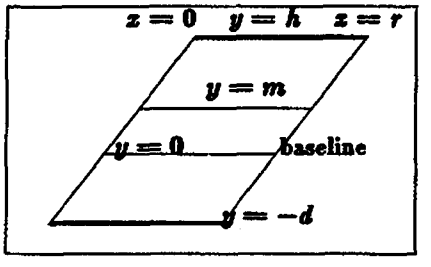
Computer Calligraphy

Georgia K. M. Tobin
Office of Research, OCLC Online Computer Library Center, Inc.

CuPl is a font I designed using Thom Hickey's Pascal version of METAFONT on an Apollo microcomputer. It is intended to capture the flavor of a type of calligraphy called Copperplate, a form of script originally engraved directly onto smooth copper plates using a stylus. This brief report will provide a general description of the design and implementation in METAFONT of *CuPl*, focusing its attention on the set-up of the pens and subroutines used in drawing the characters. In designing *CuPl*, I followed the specifications for the letter forms set down in, "Calligraphy in the Copperplate Style," by Herb Kaufman and Geri Homelsky. According to them, Copperplate letters are characterized by thick downward strokes and hairline-thin upward strokes, and the characters' axes have a uniform slant of 54 degrees from the baseline.

Cubase is the file that fills the same function for *CuPl* that *cmbase* fills for *cmr*, i.e., it contains all the assorted definitions and subroutines that take care of the nasty details that METAFONT needs to know. *Cubase* began life as a clone of the *base* that Thom Hickey used in designing Chel (Computer Helvetica), and only gradually developed its own character. The first and most obvious change was to adjust the grid upon which characters are designed to account for the slant and proportions of Copperplate-style characters. The first requirement is easily satisfied by setting *trxy* (one of METAFONT's transformation parameters) to 0.75; this gives the designated slant of 54 degrees from the baseline. (Of course, other *trxy*s may be used. While these do not give the canonical Copperplate look, they result in some interesting and useful fonts. For instance, I used a *trxy* of 0 to produce a non-slanted variant called Tinplate (*TnPl*) which can be used on the Apollo video display.)

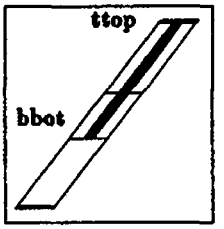
The second requirement entails setting up the proper heights for upper case characters (represented by the value of h), and lower case characters (represented by the value of m) and the proper descender depth (represented by the value of d). Kaufman and Homelsky specify that the baseline is three units above the descender; the tops of lower case letters extend two units above the baseline; and the tops of upper case characters extend three units above the tops of lower case letters. Total design size, then, equals eight units; therefore, h must equal 5/8 of design size, m must equal 2/5 of h, and d must equal 3/8 of design size to get the grid we want, namely:



The Copperplate Grid

Next, I defined a series of horizontal and vertical pens finer than those required by Chel. The horizontal pens are called w_1, w_2, w_3, w_4, w_5, w_6, and w_7; the vertical pens are w_{101}, w_{103}, and w_{105}. w_7 is the thickest pen required by *CuPl* and is used in most of the downward strokes of upper case characters. It gives a stroke of about 4.5 points when drawing a character with a design size of 80 points. w_5 is the pen *CuPl* used for most of the downward strokes in lower case characters; it produces a stroke of about 3.5 points when drawing a character with a design size of 80 points. w_1 and w_{101} are hairline-thin "slow grows"; that is, pens whose widths grow relatively slowly as boldness and/or expansion increases. These pens draw strokes wider than one pixel only at the greatest boldness and/or expansion.

A good deal of a font's character depends upon the subroutines with which it is drawn. In developing the *CutPt* subroutines, I took into account the fact that, according to Kaufman and Homelsky, there are nine basic strokes which occur in almost all the lower case letters and some of the upper case letters. Not all of these basic strokes became subroutines, and some subroutines were required for common shapes not described by those strokes; but all subroutines *are* described in fairly tiresome detail below.

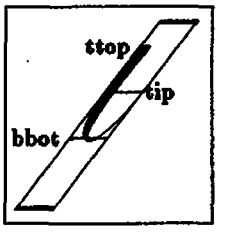


Basic Stroke One

Basic Stroke One is simply a straight line, but, because one or both of its ends may or may not taper, it is best produced by a call to either *Strone* or *Ucstrone*. *Strone* is used for lower case letters, i.e., it uses w_5 for most of the length of the stroke. It requires four parameters: the top point on the stroke (*ttop*), the bottom point on the stroke (*bbot*), the index of the pen used at the top and the index of the pen used at the bottom. These last two are needed to control taper at the ends. For example, the call

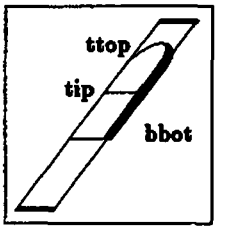
call *strone*(1, 2, 3, 3);

produces a vertical line from 1 to 2 which starts with a w_3, gradually widens to a w_5, and then tapers down again to a w_3. *Ucstrone* is used for upper case letters, i.e. it uses a w_7 for most of the stroke. It only requires two parameters, the top point and the bottom point, because tapering of the stroke from a w_5 is done automatically at both ends.



Basic Stroke Two

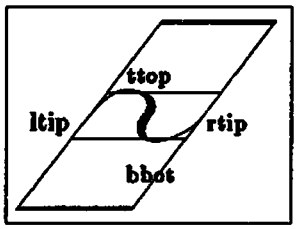
Basic Stroke Two can be produced by defining the coordinates of the topmost point on the stroke (*ttop*), the lowest point the rounded bottom reaches (*bbot*), the ending point on the stroke (*tip*), and the index of the pen used at the top (w_3 for tapering lower case letters, w_5 for non-tapering lower case letters or tapering upper case letters, w_7 for non-tapering upper case letters), and then passing those parameters to either *Strtwo* (for lower case letters) or *Ucstrtwo* (for upper case letters).



Basic Stroke Three

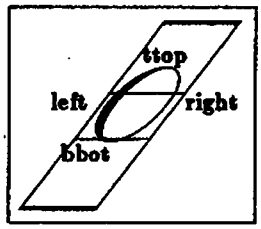
Basic Stroke Three is the reverse of Basic Stroke Two. It can be produced with a call to either *Strthree* or *Ucstrthree* for either lower or upper case, respectively. *Strthree* takes three parameters: the bottom point on

the straight stem of the stroke (*bbot*), the highest point the rounded top reaches (*ttop*), and the ending point on the stroke (*tip*). *Ucstrthree* requires those three parameters, as well as the index of the pen used at the bottom of the straight stem; w_5 is used if the stroke is to taper at the bottom, w_7 if it does not.



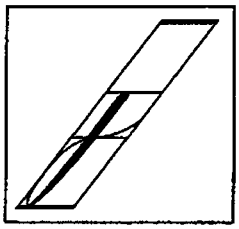
Basic Stroke Four

Basic Stroke Four is a combination of strokes two and three, and may be produced with a call to *Strfour* for lower case; this stroke does not occur in any upper case letters. *Strfour* requires four parameters: the highest point of the rounded top (*ttop*), the lowest point of the rounded bottom (*bbot*), the leftmost point where the stroke begins (*ltip*), and the rightmost point where it ends (*rtip*). All tapering required is handled by the subroutine (which is a nice way of saying that you have no control over the pens used.)

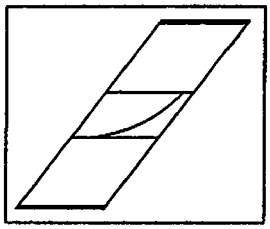


Basic Stroke Five

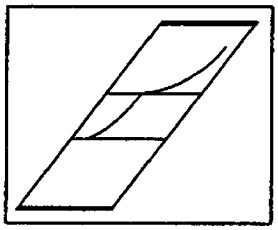
Basic Stroke Five is an oval which is thick on the left hand side and thin on the right. This stroke does not occur in any upper case letter. It may be produced by calling *Strfive*. The parameters required are: the highest point (*ttop*), the lowest point (*bbot*), the rightmost point (*right*), and the leftmost point (*left*). All tapering is handled by the subroutine.



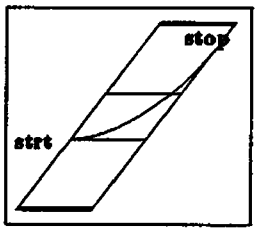
Basic Stroke Six



Basic Stroke Seven

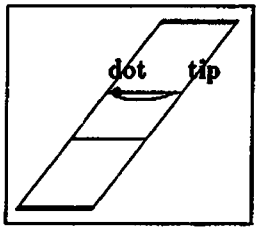


Basic Stroke Eight

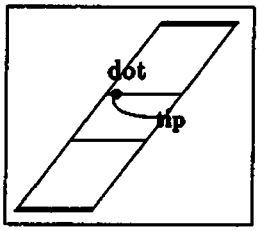


Hairarc

Basic Stroke Six does not occur often enough to merit its own subroutine, and is produced on an *ad hoc* basis. Basic Strokes Seven and Eight are adequately handled by, respectively, one or two calls to the subroutine *Hairarc*, which produces a concave arc from the first point it receives (*strt*) to the second point it receives (*stop*) using a hairline thin pen.



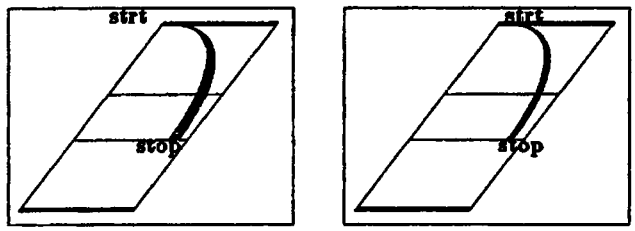
Basic Stroke Nine



Dotloop

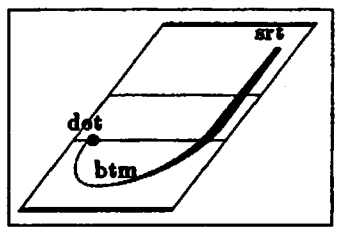
Basic Stroke Nine is produced by a call to *Strnine*. This subroutine requires two parameters: the point at which the dot appears (*dot*), and the ending point of the hairline tail (*tip*). A similar stroke is produced by *Dotloop*, which takes the same two parameters, but produces a tail with a shallower curve.

In addition to the basic strokes that Kaufman and Homelsky describe, there are a number of strokes which occur more or less frequently in Copperplate letters which are also handled by subroutines in *cubase*. These are described below.



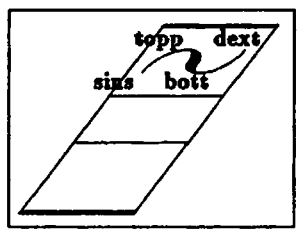
Arc **Growarc**

Arc is used to produce a tapering arc. It requires three parameters: the thinnest point on the arc (drawn with a w_1) (*strt*), the thickest point on the arc (*stop*), and the index of the pen used at the thickest point. *Growarc* also produces an arc given two points, *strt* and *stop*, but allows the user to specify both the pen with which we are to start drawing at *strt* and the pen with which we finish drawing at *stop*.



Dotstroke

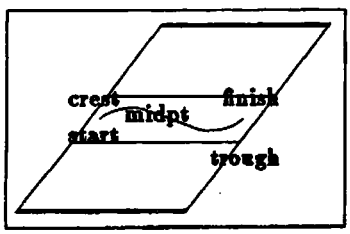
A curving stroke, which tapers at one end and concludes with a dot at the other, is produced by the subroutine *Dotstroke*. *Dotstroke* requires three parameters: the highest and rightmost point on the stroke (*srt*), the nadir of the stroke's rounded bottom (*btm*), and the end point where the dot appears (*dot*).



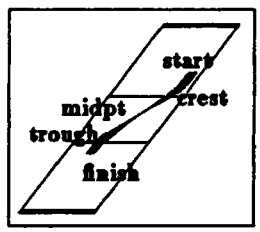
Flounce

This decorative stroke, which appears in several upper case letters, is produced by a call to *Flounce*. Four parameters are required: the leftmost point (*sins*), the rightmost point (*dext*), the highest point (*topp*) and the lowest point (*bott*). All required tapering is handled automatically within the subroutine.

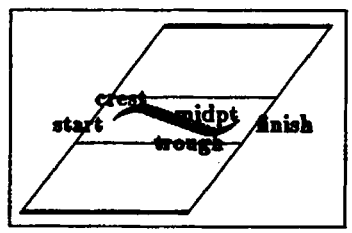
Several subroutines are used to produce smooth, symmetrical waves. Those which have a crest followed by a trough are produced by *Zhair*, *Zsquigl*, or *Qsquigl*.



Zhair

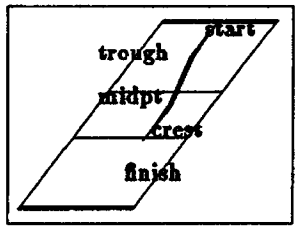


Qsquigl

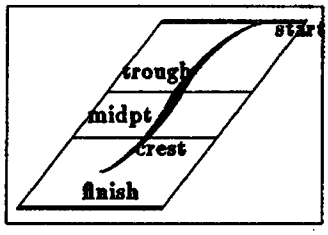


Zsquigl

Zsquigl, *Qsquigl* and *Zhair* all take the following parameters: the wave's starting point (*start*), the y-value of the turning point at the top of the crest (*crest*), the mid-point of the wave (*midpt*), the y-value of the turning point at the bottom of the crest (*trough*), the wave's ending point (*finish*), and a variable equal to the reciprocal of the ending slope. *Zhair* draws the entire wave thus defined with a hairline thin cpen, and so does not require any pen width parameters. *Qsquigl* and *Zsquigl* are passed two pen width parameters before the slope variable is given: the index of the pen with which we start drawing and the index of the pen with which we conclude. *Qsquigl* differs from *Zsquigl* in that *Zsquigl* is designed for waves with a horizontal orientation (i.e., it draws with a vpen), whereas *Qsquigl* is designed for waves with a vertical orientation (i.e., it draws with an hpen). Moreover, the *Qsquigl* wave tapers only at its ending point, whereas *Zsquigl* tapers at both the starting and ending points. All three are variations of *Zdraw* used in Don Knuth's *cmbase*, and they rely on a subroutine *Zcomp* (which is pilfered wholesale from *cmbase*) to handle the trigonometric nitty-gritty involved.



Msquigl



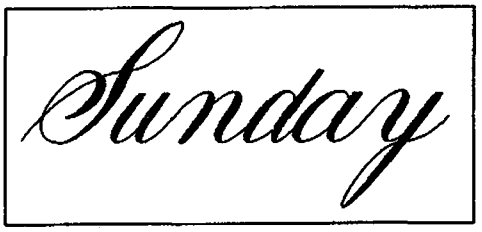
Tsquigl

Strokes which have a trough followed by a crest are produced by *Msquigl* or *Tsquigl*. *Msquigl* uses slightly thinner pens. Both subroutines require the following eight parameters: the wave's starting point (*start*), the x-coordinate of the turning point at the bottom of the trough (*trough*), the wave's midpoint (*midpt*), the x-coordinate of the turning point at the top of the crest (*crest*), the wave's ending point (*finish*), the index of the pen with which we start drawing, the index of the pen with which we conclude, and the slope at mid-wave. These subroutines are also adapted from *cmbase*. They call *Scomp* for computing values.

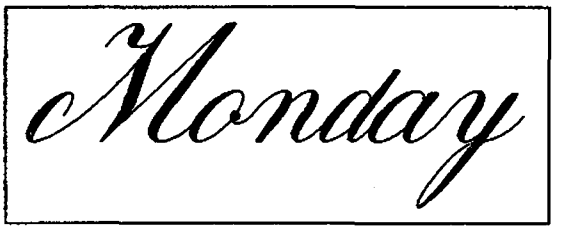
At one point, Kaufman and Homelsky remark, "The joining of letters into words is as important as the forming of the letters themselves". This nicely sums up another problem in designing *CuPt*, i.e., contriving to make discrete characters appear to join like flowing script. This is relatively simple for the lower case letters. I simply designed each so that the "beginning" of the character (that is, its left side) included the point with coordinates $(0,2/5h)$, and the "ending" of the character (that is, its right side) was precisely at the point with coordinates $(r, 2/5h)$. Occasionally, this produces a small bit of overlap:

The finishing upswing on the 'v' runs into the left side of the 'e', the finishing upswing on the 'r' runs into the left side of the 'l', and so on. Normally, this isn't apparent (I hope):

No single rule for beginning and ending points applies to the upper case letters, but then, none needs to: the upper case letters never need to be connected in sequence. What the precise value of an upper case letter's smallest x-coordinate is doesn't matter (as long, of course, as it is greater than zero). Some upper case letters (viz., H, J, K, M, R, U, X and Z) end in a hairline upswing that must connect to minuscules; for these, the right side must end at the point with coordinates $(r, 2/5h)$. The other upper case letters stand independently, and their right hand sides need only end at some point with an x-coordinate less than r.

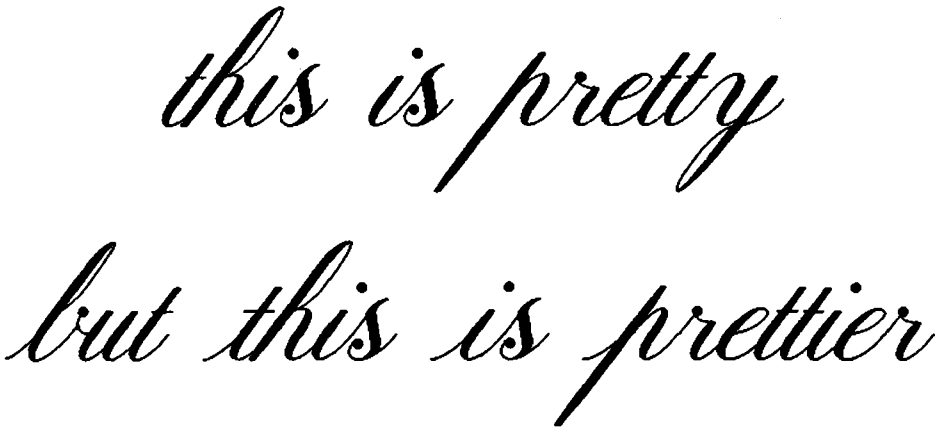


Non-Connecting Upper Case



Connecting Upper Case

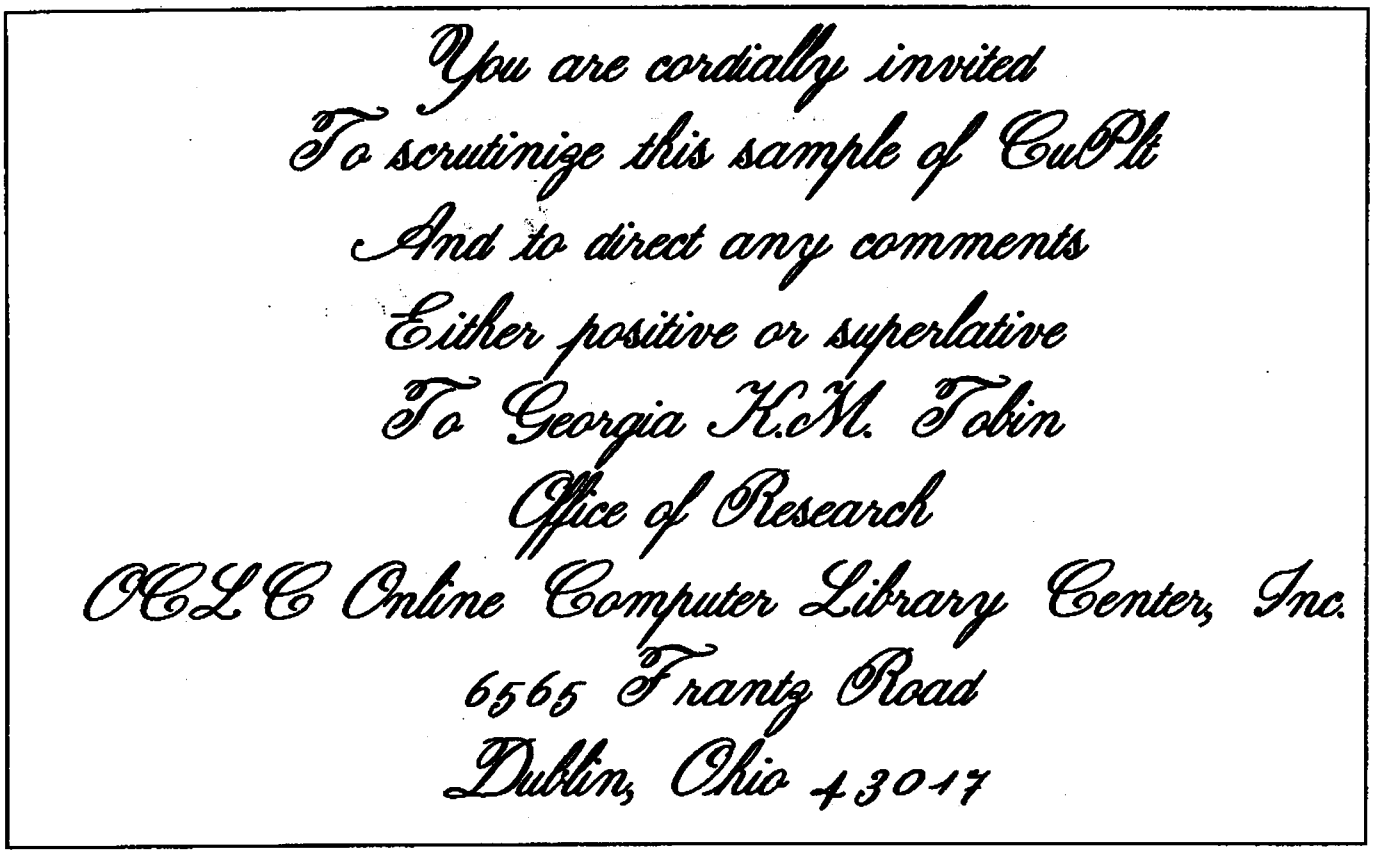
There are still a number of rough spots in *CuPt*. One problem is tied to the convention that the beginning of lower case letters include the point with coordinates $(0,2/5h)$. In certain cases, viz., b, h, i, j, k, l, p, r, s and t, the letter form which connects nicely is not always the most desirable form. When these letters do not follow another lower case letter, a form which begins with a lead-in hairline stroke from the baseline looks better than the usual combining form. That is, I claim that:



As of this writing, I have not found a way to put the information that (for instance) '001 is to replace 'b after a blank, or after certain upper case letters, or after certain punctuation marks. I am able to print these ligatures by manually inserting a dummy character (*) in front of letters to be replaced and including '* in the ligature table; but this approach is far too cumbersome and I hardly ever insert an * every place I need one.

At present, all upper and lower case letters, all digits and about a dozen punctuation marks are designed, and look acceptably good in *most* design sizes for *most* boldnesses and *most* expansions. I hope to carry out exhaustive testing to determine specific problems and to correct those, so that the same can be said of *all* design sizes at *all* boldnesses and expansions.

Meanwhile, here's a sample:



Bibliography

Hickey, Thomas B. "The Status of METAFONT at OCLC". *TUGboat*, Volume 2, No. 2, July 1981, pp. 35-38.

Hickey, Thomas B.; Tobin, Georgia K.M., *The Book of Chels*. Privately produced limited edition. 1982.

Kaufman, Herb; Homelsky, Geri. *Calligraphy in the Copperplate Style*. Dover Publications, Inc. New York, 1980.

Knuth, Donald E. *The Computer Modern Family of Typefaces*. Computer Science Department Report No. STAN-CS-80-780. January, 1980.

Knuth, Donald E. TEX *and* METAFONT*: New Directions in Typesetting*. American Mathematical Society/Digital Press. 1979.

The Fifth ATypI Working Seminar

The Computer and the Hand in Type Design: The Aesthetics and Technology of Digital Letterforms

During the week of July 31–August 7, at Stanford University, the Committee for Education and Research in Letterforms of the Association Typographique Internationale (ATypI) is sponsoring an International Working Seminar on electronic and traditional methods of letter design. The program includes workshops, seminars and illustrated lectures, and it will conclude with a typographical excursion to San Francisco.

The Seminar will begin on Monday morning, August 1, with the keynote address, "A Turning Point in Type Design", by John Dreyfus, Honorary President of ATypI. Other speakers include the type designers Hermann Zapf, Matthew Carter, Andre Guertler, Christian Mengelt, Gerard Unger and Bram de Does; typographers Jack Stauffacher and Charles Bigelow; lettering artists David Kindersley and John Benson; type punch-cutter Henk Drost; designer Veronika Elsher; and computer scientists Donald Knuth, Patrick Baudelaire and Neil Wiseman.

The theme and purpose of the Seminar are:

- To acquaint educators and designers with the new computerized methods of type production and to review certain traditional lettering crafts, including punch-cutting and stonecutting.
- To provide practical experience with computer-aided design systems.
- To bring designers and engineers together for future cooperation and creation in type design.

Working installations of the IKARUS, METAFONT, ALTO, CAMEX, and other systems will be available for use during the Seminar. Demonstrations and assistance in using the computers will be provided. Morning seminar sessions will be devoted to working with computer systems, and the systems will be available at other times of the day and night for further exploration.

Prior to the Seminar, information materials on each computer-aided design system, with samples of selected design problems, will be sent to each registered participant. This is to acquaint participants with the systems before their arrival at the Seminar.

The Seminar is intended for design research, not scientific research. Participants do not need scientific training. The emphasis will be on the practice of design with the new computer technology, and with traditional hand technology.

The Seminar language is English, with translators for German and French.

Fees of $950 per person include Seminar, shared double room, meals, reception, and excursion. The rooms and meals are in a Stanford Residence Hall on the Stanford Campus. For a private, single room, the total Seminar fee is $1025 per person. [For partial bookings, e.g. seminar without room and meals, or a stay of less than the full week, send inquiries to the address below.]

To reserve a position at the Seminar, or for additional information, write to

Charles Bigelow
President, ATYPI Committee on
Education and Research in Letterforms
Department of Computer Science
Stanford University
Stanford, California 94305 USA

Reservations must be accompanied by the full fee.

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

MACRO COLUMN

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

TUGBOAT MACRO INDEX

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

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

* * * * * * * * * * *

HOW TO BUILD A \STRUT

Barbara N. Beeton
American Mathematical Society

Struts are things that keep objects a fixed distance apart, like the wings of a biplane. Because of the way that TEX puts boxes together vertically, struts are sometimes needed to maintain the desired distance. The concept was introduced in the definition and explanation of \| on pp. 108–109 of the TEX manual [*TEX and Metafont*]: "TEX doesn't use `\baselineskip` and `\lineskip` before and after horizontal rules." The failure of `\baselineskip` to apply the desired spacing also affects adjacent `\hbox` pars which contain more than one line, but in that case, a visible vertical rule would not be a satisfactory remedy. As it happens, an *invisible* vertical rule is just the thing, but first let us look at the problem in more detail.

In text, `\baselineskip` is typically set at 2 points greater than the text body size: 10-on-12, 9-on-11, 8-on-10. For some special work, though, it may be desirable to set material more densely, even "solid" — 10-on-10, etc. Only rarely are lines of text set any closer than that, and struts won't help with that problem in any case, so it will be ignored here. A strut for solid text should be the same height and depth as the tallest and deepest characters in the font; in METAFONT text fonts, a parenthesis ( ) or square bracket [ ] qualifies, so adjacent vertical boxes containing one of these on the last and first lines respectively will be separated by the desired distance. Consider the following example, which consists of three `\hbox` pars: the first junction lacks sufficient ascenders and descenders to force the baselines apart to the `\baselineskip` distance (this is `\tenpoint\rm \baselineskip 10pt`), but the second junction looks no different from two lines in the middle of a paragraph.

This paragraph has no
descenders in the last line.
one can scarce see an
answer to this (let's cheat).
(This example may be
contrived, but it works.)

Now, define a strut with the maximum height and depth of any character in the font, and insert it at the beginning of the first and end of the last line in each paragraph:

This paragraph has no
descenders in the last line.
one can scarce see an
answer to this (let's cheat).
(This example may be
contrived, but it works.)

Finally, reset `\baselineskip 12pt` and apply a strut that is 2 points longer:

This paragraph has no
descenders in the last line.
one can scarce see an
answer to this (let's cheat).
(This example may be
contrived, but it works.)

There are probably many ways actually to define struts, but only two will be shown here. In one approach, strut is defined within the range of each "size" definition (this is Knuth's approach):

```
\def \tenpoint{\baselineskip 12pt ...
    \def\strut{\lower 3.5pt\vbox to 12pt{}}
    ... }
```

The following is equivalent for `\tenpoint`, but more efficient (because rules take less memory space

than boxes):

```
\def \strut{\vrule
    height 8.5pt depth 3.5pt width 0pt}
```

The preceding defined a strut of the "second" kind, which at AMS we call a \strutt. We also use "throwaway" definitions, in which the size of the strut is calculated on the basis of the current text font, so that only two definitions are necessary, rather than one for each size (some applications include as many as 6 "text" sizes, including titles, footnotes, *et al.*). These probably run less efficiently, but permit more generality in the construction of \tenpoint and its friends, a memory-saver when there are frequent size changes.

```
\def \strut{\save7\hbox{()\vrule
    height 1ht7 depth 1dp7 width 0pt
    \save7\hbox{}}
\def \strutt{\save7\hbox
    {(\lower2pt\hbox{()}\vrule
    height 1ht7 depth 1dp7 width 0pt
    \save7\hbox{}}
```

Just because \hbox pars go away in TEX82 doesn't mean that the need for struts will vanish; it won't. But there will be a more efficient way to define them. See item 137 in the current list of differences between TEX80 and TEX82.

* * * * * * * * * * *

DETERMINING HASHTABLE SIZE AND OTHER QUANTITIES

Barbara N. Beeton
American Mathematical Society

The hashtable, in which names of control sequences are recorded, is limited in size. In TEX80, a common size is 1009 (= MIX); it will be larger in TEX82, but the number of primitives and names in PLAIN.TEX is also larger than the basic set. And once a name is loaded, it is never removed. So it's not too hard to run out of space for new names.

At AMS, a file called HOWMANY.MAC (a shortened version is shown below) is used to help determine how much space is left after all the header file \defs for a job have been installed. To use it, load in the header(s), then \input howmany.mac; this will rapidly fill the hashtable and blow up when there's no more room, returning with the error message the number of control sequence names that were able to be loaded before space ran out.

```
\chcode`176=11 % ~ - make it a letter

\def \~~~{0 }

\def \~AA{1 }
\def \~AB{2 }
\def \~AC{3 }
   . . .
\def \~AS{19 }
\def \~AT{20 }
\def \~BA{21 }
\def \~BB{22 }
   . . .
\def \~OS{299 }
\def \~OT{300 }
```

The control sequence names used in such a file must be unique. In HOWMANY.MAC, only 20 letters of the alphabet were used, so that the file is easily extensible with the help of a few commands to an editor. This also made checking the count easier. If the count is wrong, or a name in this file duplicates that of an existing control sequence, results will be unreliable.

The following test ran on a SAIL version of TEX80 on a DEC 2060.

```
@tex header.fil
[input from: notices]
AMS TeX varsize of 11500 created
  Wednesday, August 18, 1982 16:07:51
this run of TeX begun:
  Tuesday, March 1, 1983 23:00:32

*
(PS:<BNB>HEADER.FIL.1)
*\input howmany.mac

(PS:<AMSTEX>HOWMANY.MAC.3 1
! TEX capacity exceeded, sorry [hashsize=1009].
p.1,l.67 \def \~DC
                {63 }
No output file.
```

It is useful to run a new set of header files through this test before putting them into production, and some other useful checks can be performed at the same time. Use a 'slow' version of TEX (one compiled in such a way that statistics are saved for such things as memory size), and insert \ddt (for TEX80) before \input howmany.mac. This will report the location on the page and the current nesting level, all of which should ordinarily be 0 if there aren't any braces missing in the header. It will also tell you how much memory the header requires—if total available memory is 22,000 words, and the header alone occupies 19,000, some pruning is in order.

* * * * * * * * * * *

SOME LAYOUT MACROS

August Mohr

Editor's note: The author of this note is the editor of *CommUNIXations*, an elaborately laid-out newsletter for the UNIX community. His original plan was to use his layout macros to describe themselves. But the deadline arrived before the manuscript ....

So, for better or worse, here's a few macros that you might find interesting. This one has a structure that has been useful in several variations.

```
\def \by #1\\ {\def \OF{}\gdef \of ##1::{\gdef
                    \OF{\unskip, {\it ##1\/}}}
      {\ragged 1000\hbox par size
           {\unskip\bf By #1\OF\par}}}
```

This one can be used as either

```
\by Author N. Ame\\
```

or as

```
\by Arthur N. Ames\of Groupth, Inc.::\\
```

to produce a paragraph-form author-credit.

Note that the macro \of will redefine \OF when it is used. Otherwise, \OF remains {}. \by is closed with \\, and \of is closed with ::. (I'm not very pleased with the ::\\ construction; at the time I tried to do without braces à la Mike Spivak.)

The same general pattern is used in \cont and \contf, which produce the "Continued on ... " and "Continued from ... " references at the tops and bottoms of columns. Of course, to get them at the top or bottom, they have to be referenced by the \output routine, but that's another subject. The special thing about them, like \by above, is that they have optional parts. They may be used in simple or complex ways such as,

```
\cont{page 12}
```

or

```
\contf{previous page\col3\ti{A Simple Title}}
```

The key to making the pieces optional, such as \col and \ti, is that they are always given one argument, which may have more than one part.

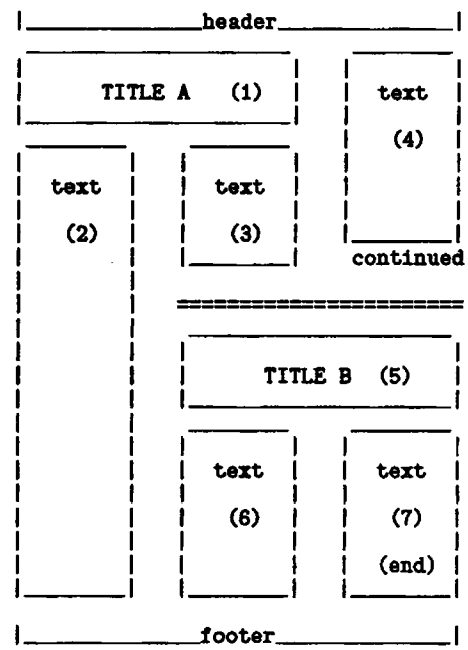
Here are the macros:

```
% \contfont is a font, 9pt italic.
% \strut is a vrule of width 0 and
%       height and depth to match the font.

% Article continuation macros
\def \cont#1{\def\COL{}\def \col##1{\gdef
                  \COL{\unskip, Column ##1}}
     \vskip2pt\hbox to size{\hfil
        \contfont Continued on #1\COL\hfil}}

% Usage: \contf{ {Page #} <\col{column#}>
%        <\ti{title}> }  (<...> is optional)

\def \contf#1{\def\TI{}\def
                  \COL{}\def \col##1{\gdef
                   \COL{\unskip, Column ##1}}
     \def \ti##1{\gdef\TI{\hbox to size{\hfil
                  \contfont ##1\strut\hfil}}}
     \xdef\PG{#1}
     \TI\hbox to size{\hfil\contfont
          Continued from \PG\COL\strut\hfil}}
```

Both \col and \ti are optional, and their order is not important, but the page number should come before them. If \ti is used, the result is two lines, the first containing the "Title".

The really tricky part is \xdef\PG, which is given the entire input parameter string, and allows \col and \ti, if they are present, to do their work of redefining \COL and \TI.

These macros are most useful in output routines or inside other macros, since it would be rare in straight text to know when and where the page will break. The structure, however, is useful in many cases.



In another issue, I'd like to describe how I get TEX to do some of the trickier kinds of 3-column layout. In a page like this, pieces 3 and 4 must balance, allowing for title (1) and the continuation line, and their sizes also depend on the length of article B.

In my solution, the output routine keeps changing the \vsize, and saving the pages in successive \boxes. When the proper number of boxes have been filled, it puts out the completely assembled page.

If there are other TUGboat readers who have tried similar constructions, I'd like to hear from

them. I'm also interested in corresponding with people who'd like to use a layout schema like this to help me get the macros to be less idiosyncratic.

* * * * * * * * * * *

TESTING THE WIDTHS OF A FONT

Robert M. McClure
Unidot, Inc.
Communicated by August Mohr

This macro takes the characters of a font and makes a string of 10 of them, then draws a box around the string. The box's size is determined by TEX, from the font-metric file. The character string is generated by the typesetter. When the width is right, the box will match the characters, but when the width is off, the box will show the difference. Using 10 characters in a row multiplies the difference by 10, which makes it much easier to be very accurate about the width.

The width file is based on a "design size" of 18pt, so testing with a 9pt font means the error is only multiplied by 5, relative to the width file.

Here's the file "`fontest.x`":

```
\chcode'173=1    \chcode'175=2
\chcode'44=3     \chcode'100=4
\chcode'45=5     \chcode'43=6
\chcode'136=7    \chcode'137=8

\def\null{\hbox{}}

\def\boxit#1{\vbox{\hrule
     \hbox{\vrule #1\vrule}\hrule}}
\def\vboxit#1{\hbox{\vrule
     \vbox{\hrule\hbox{#1}\hrule}\vrule}}
\def\need#1{\vskip #1 minus 1fil
     \penalty 0\vskip -#1 minus -1fil}

\output{ \vskip .1in \page \vskip .1in }

\hsize 7in
\vsize 9in
\maxdepth 2pt
\topbaseline 10pt
\lineskip 0pt
\parskip 0pt    \parindent 0pt

\setcount0 1
\chpar0=2

% --- Standard TeX fonts ---
\font A=life at 9pt  % *** Font under test
\font H=helv at 10pt
\font I=helv at 8pt
\font J=helv at 6pt
\font K=helv at 5pt
\font L=helv at 9pt

\def\x#1{\hbox{\:K#1}\hskip .1in
     \hbox to 3in{\hbox to .26in{\hbox{\char'#1}\hss}
     \hbox to .26in{\vboxit{\char'#1}\hss}
     \vboxit{\char'#1\char'#1\char'#1\char'#1\char'#1
      \char'#1\char'#1\char'#1\char'#1\char'#1}\hss}}
\def\vsk{\vskip .in plus .1in}

\baselineskip 0pt
\:A

\hbox{\:H life- 115 }
% *** Name and number of font being tested
\vsk
\hbox{\x{000}\x{001}} \vsk \hbox{\x{002}\x{003}} \vsk
\hbox{\x{004}\x{005}} \vsk \hbox{\x{006}\x{007}} \vsk
\hbox{\x{010}\x{011}} \vsk \hbox{\x{012}\x{013}} \vsk
\hbox{\x{014}\x{015}} \vsk \hbox{\x{016}\x{017}} \vsk
\hbox{\x{020}\x{021}} \vsk \hbox{\x{022}\x{023}} \vsk
\hbox{\x{024}\x{025}} \vsk \hbox{\x{026}\x{027}} \vsk
\hbox{\x{030}\x{031}} \vsk \hbox{\x{032}\x{033}} \vsk
\hbox{\x{034}\x{035}} \vsk \hbox{\x{036}\x{037}} \vsk
\hbox{\x{040}\x{041}} \vsk \hbox{\x{042}\x{043}} \vsk
\hbox{\x{044}\x{045}} \vsk \hbox{\x{046}\x{047}} \vsk
\hbox{\x{050}\x{051}} \vsk \hbox{\x{052}\x{053}} \vsk
\hbox{\x{054}\x{055}} \vsk \hbox{\x{056}\x{057}} \vsk
\hbox{\x{060}\x{061}} \vsk \hbox{\x{062}\x{063}} \vsk
\hbox{\x{064}\x{065}} \vsk \hbox{\x{066}\x{067}} \vsk
\hbox{\x{070}\x{071}} \vsk \hbox{\x{072}\x{073}} \vsk
\hbox{\x{074}\x{075}} \vsk \hbox{\x{076}\x{077}} \vsk
\hbox{\x{100}\x{101}} \vsk \hbox{\x{102}\x{103}} \vsk
\hbox{\x{104}\x{105}} \vsk \hbox{\x{106}\x{107}} \vsk
\hbox{\x{110}\x{111}} \vsk \hbox{\x{112}\x{113}} \vsk
\hbox{\x{114}\x{115}} \vsk \hbox{\x{116}\x{117}} \vsk
\hbox{\x{120}\x{121}} \vsk \hbox{\x{122}\x{123}} \vsk
\hbox{\x{124}\x{125}} \vsk \hbox{\x{126}\x{127}} \vsk
\hbox{\x{130}\x{131}} \vsk \hbox{\x{132}\x{133}} \vsk
\hbox{\x{134}\x{135}} \vsk \hbox{\x{136}\x{137}} \vsk
\hbox{\x{140}\x{141}} \vsk \hbox{\x{142}\x{143}} \vsk
\hbox{\x{144}\x{145}} \vsk \hbox{\x{146}\x{147}} \vsk
\hbox{\x{150}\x{151}} \vsk \hbox{\x{152}\x{153}} \vsk
\hbox{\x{154}\x{155}} \vsk \hbox{\x{156}\x{157}} \vsk
\hbox{\x{160}\x{161}} \vsk \hbox{\x{162}\x{163}} \vsk
\hbox{\x{164}\x{165}} \vsk \hbox{\x{166}\x{167}} \vsk
\hbox{\x{170}\x{171}} \vsk \hbox{\x{172}\x{173}} \vsk
\hbox{\x{174}\x{175}} \vsk \hbox{\x{176}\x{177}}
\vfill
\eject
\end
```

* * * * * * * * * * *

Problems

* * * * * * * * * * *

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

Hanging punctuation

In the last issue, Problem #2 gave this example of a paragraph in which (nearly) all line-initial and line-final punctuation symbols extended into the margin, so that the text itself is aligned:

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

The problem: prepare a file PUNCTUATION.TEX containing definitions necessary to make punctuation characters behave as shown. The following set of definitions is one possible solution.

```
\spaceskip .3em plus .4em minus .1em

\def\,#1{{\def\'{'}
         \xdef\blah{#1}
         \def\space{ }
         \ifx \blah\space{\comma}
         \else{,}
         }#1}

\def\comma{\save0\hbox
     {,\hskip .3em}\hbox to 0pt{,\hss}\hskip
          1wd0 plus .5em minus .08em}

\def\.#1{{\def\'{'}
         \xdef\blah{#1}
         \def\space{ }
         \ifx \blah\space{\period}
         \else{.}
         }#1}

\def\period{\save0\hbox
     {.\hskip .3em}\hbox to 0pt{.\hss}\hskip
          1wd0 plus 1.2em minus .033em}

\def\'#1{{\def\'{test for double quote}
         \xdef\blah{#1}
         \ifx \blah\'{\rightquote{''}}
         \else{\rightquote{'}#1}
          }}

\def\`#1{{\def\`{test for double quote}
         \xdef\blah{#1}
         \ifx \blah\`{\leftquote{``}}
         \else{\leftquote{`}#1}
          }}

\def\leftquote#1{\save0\hbox
         {#1}\ifvmode{$ $\hskip -1wd0}\else{}
  \hskip 1wd0\null\penalty1000\hskip -1wd0 #1}

\def\rightquote#1{\save0\hbox
         {#1\hskip .3em}\hbox to
   0pt{#1\hss}\hskip 1wd0 plus .4em minus .1em}

\chcode'54=13   % Make , automatically invoke \,
\chcode'39=13   % Make ' automatically invoke \'
\chcode'140=13  % Make ` automatically invoke \`
\chcode'46=13   % Make . automatically invoke \.
```

When a comma or period is encountered, the macros check whether the next character is a space (macros \, and \.). If so, appropriate glue is inserted (macros \comma and \period) to produce the desired effect if the character should occur at the end of a line.

When a single left or right quotation mark is encountered, macros \` and \' test to see if the next character is another occurrence of the same symbol and then call respectively \leftquote or \rightquote to insert the symbol and appropriate glue. The \leftquote macro must also test to see if the open quote symbol is the first character in a new paragraph. This approach is not completely general, because it assumes that the closing quotation marks will be followed by white space. This does not account for apostrophes, for example, or quotation marks followed by parentheses. To solve this problem, the quotation mark macros can be extended, by analogy to the comma and period macros, to test whether they are followed by spaces.

Users of versions of TEX that do not provide the \ifx control sequence may want to modify the input slightly, by inserting backslash characters before punctuation symbols, or by using the "long form" (\comma) directly, as was necessary for preparing this copy. (At the AMS, the version of TEX being used is an antique—a SAIL version current as of March 1981. We hope the next issue of TUGboat will be a demonstration of TEX82.)

* * * * * * * * * * *

Letters et alia

* * * * * * * * * * *

To the Editor:

New releases of TEX82 have been made since the last announcement in TUGboat, and I thought your readers would like to know about this.

The present release is number .96. There have been five releases in the past year.

The entire Stanford font library is now available for distribution. It is broken into 6 tapes according to the pixel densities.

METAFONT is also available, but it is written in SAIL and recorded with the non-standard version of ASCII used at the Stanford AI center.

I am enclosing an order form for your convenience. Please get rid of any old forms you may have.

It is our wish and that of the TEX group at Stanford to make these materials available at a reasonable price to all interested persons. We hope that those who have ordered tapes from us have been satisfied with our service in the past and will continue to deal with us in the future.

Maria Code
DP Services
1371 Sydney Drive
Sunnyvale, CA 94087

Editor's note: Tapes containing TEX82 source code and fonts are not available from TUG. The "standard" files are being distributed by DP Services as described above. Versions already tailored for various computer architectures may be available as well; check with the relevant Site Coordinator. For the convenience of those wishing to obtain the official Stanford versions, an order form is bound into the back of this issue.

* * * * * * * * * * *

MANIPULATION DE DOCUMENTS JOURNÉES FRANCOPHONES

Rennes, 4–6 Mai 1983

Editor's note: Although the deadline has passed to submit a paper for presentation at this conference, it may still be possible to attend or to arrange to receive copies of the proceedings.

Objectifs

Le but de ces journées est de faire le point sur les systèmes informatisés de manipulation de documents (textes au sens large du terme). Ces journées viennent après celles de Lausanne (Février 1981) et de Portland (Juin 1981), et s'adressent à un **public francophone**. L'accent sera plus particulièrement mis sur l'impact des nouvelles technologies logicielles (systèmes experts, etc.) et matérielles (disques optiques numériques, imprimantes à laser, etc.) sur le **traitement des textes contenant des informations graphiques** (formules, dessins, schémas etc.). L'utilisation de tels systèmes par les professionels des arts graphiques et de l'édition sera aussi considérée.

Il s'agira de "Journées" s'addressant aux techniciens déja spécialistes du sujet (concepteurs, implémenteurs, utilisateurs ayant acquis une expérience dans ce domaine). L'auditoire sera limité à une centaine de personnes. Ces Journées se dérouleront sur deux modes:

- Exposés et table rondes
- Démonstrations, non commerciales, de prototypes

Thèmes des journées

- Typographie
- Systèmes interactifs de manipulation de documents
- Manipulation dans un texte des diverses informations graphiques (formules mathématiques ou chimiques, schémas, dessins, photos etc.)
- Mise en page de textes illustrés
- Mise en page spécifique aux terminaux vidéotex
- Manipulation de documents vocaux
- Structuration et codage des documents illustrés
- Emploi de bases d'informations généralisées
- Utilisation des nouvelles technologies (imprimantes sans impact, disques optiques numériques, etc.)
- Editeurs intelligents et systèmes experts
- Ergonomie des matériels et des interactivités homme-machine
- Normes (CCITT, ISO, etc.)
- Reproduction et diffusion d'ouvrages réalisés par des systèmes informatisés de manipulation de documents
- Expériences de réalisations informatiques dans les milieux professionels des arts graphiques et de l'édition.

Renseignement et soumission des communications

Jacques André
IRISA/INRIA – Rennes
Campus Universitaire de Beaulieu
35042 Rennes Cédex, France
tel. (99)36 20 00; télex: 950 473F

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

Dreamboat

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

TEX AS A PROGRAMMING LANGUAGE?

A. E. Siegman
Professor of Electrical Engineering and
Director of the Edward L. Ginzton Laboratory, Stanford University

Many of the future users of TEX will not be computer professionals. The majority of future users, in fact, will be "knowledge workers"— scientists and engineers, editors and writers, humanities scholars, university faculty members, secretaries, and even business professionals.

Many of these users will want to do all their text composition, communication, document preparations, and (very important) their record-keeping using only a local screen editor, an electronic mail system, and TEX— nothing else. One reason for keeping this list of computer tools as short as possible is that the working user of a computer system — in contrast to a computer professional — usually wishes to learn as few systems or languages as possible. This also minimizes the number of file formats to be concerned with, and the number of programs that must be provided on the system.

An important implication of this is that it would be very desirable to develop a TEX *programming language*, separate and distinct from the TEX *typesetting program*. That is, one would like to have a programming language which would retain essentially the same syntax and the same programming capabilities as TEX (plus additional capabilities), but which would work only with ASCII characters and produce only ASCII output. What's important of course is not so much the ASCII aspect, but retaining (and even supplementing) most of the programming language and macro capabilities of TEX, while eliminating everything related to the more sophisticated typesetting and printing aspects.

As one illustration of what this implies, consider a TEX user who keeps bibliographies, address lists, and other sets of records with each record stored in a format like

```
\author{J. Jones}
\title{History of TeX}
\date{December 1982}
```

By defining suitable macros for \title, \author and so forth, it is possible at present not only to introduce these records into documents to be typeset, but also to use TEX to manipulate, rearrange, and reformat these records in a variety of ways, i.e. to use TEX as a programming and record-manipulating language.

Suppose however that the user only needs to manipulate these records, and then output a reordered subset of each record in ASCII lineprinter format, not as typeset output. Reasons for doing this might be to get output on a local hardcopy terminal; or perhaps not printed output at all, just reformatted output to be sent to another file or electronically transferred to a non-TEX-using colleague. This can be done at present by using \send to transfer the formatted output to another file and discarding the pages that TEX produces; but this solution is inelegant, incomplete, and expensive.

A supplementary TEX-like language, which did not require learning an entire new syntax, and which might offer added arithmetic and logical capabilities, would be extremely useful in this situation. It's not necessary here to explore just which capabilities should be retained or added and which should be discarded in such a TEX programming language. Clearly the line-breaking and page-making capabilities of TEX might go; the entire concept of glue might be eliminated; and so forth. A side benefit is that a simplified TEX programming language might be a much less expensive program to run. (This may not seem important to computer professionals in working environments where extensive computer resources are readily available. In environments where users must pay for CPU cycles on a fully costed basis, however, TEX is in fact an expensive program to run).

All this is obviously not something that has much connection with Don Knuth's original objectives in writing TEX. If however TEX is to become a widely used language — as I believe it is — for people who work with words and ideas, then the concept that TEX and its derivatives should meet essentially all the needs for those people becomes important. The importance of minimizing the number of languages that a lay computer user has to learn should not be minimized. This note is to suggest to the Tugboat community that an offshoot of TEX, which uses the same notation and syntax to manipulate strings, numbers, and files, without doing any typesetting at all, is well worth thinking about.

* * * * * * * * * * *

Advertisements

* * * * * * * * * * *

TEXTSET, INC. — A SERVICE FOR TEX USERS

Paul Grosso
David Rodgers
James Sterken

Textset, Inc. can offer TEX Users:

- Final-quality phototypesetting from DVI files using a variety of font styles.
- Consulting assistance with installation of TEX82, with developing DVI-to-printer driver programs, and with developing specialized TEX applications.

Textset, Inc. is a newly formed company offering direct-typesetting to customers who have documents stored on a computer system or word processor and need a path to highest-quality photocomposition. We have adopted TEX82 as our typesetting language and use an Autologic APS-5 phototypesetter for final pages.

Manuscripts can be transmitted by telephone or standard magnetic tape. Ordinarily, 72-hour turnaround is available for production typesetting. In special cases, 24-hour turnaround is possible.

Textset's production system can support existing Autologic fonts and Computer Modern Fonts as soon as they become available. Stanford and Autologic have reached an agreement that will make Computer Modern fonts available as soon as Autologic makes the necessary packaging and pricing decisions.

Textset, Inc. will distribute a WEB DVI-to-printer driver program for the APS-5 family of phototypesetters that was jointly developed with Dave Fuchs and for which **Textset** will provide support and maintenance. We have experience installing TEX under four different operating system environments (MVS, VM/CMS, UNIX, and MTS) and experience supporting several printing devices.

Textset's computer system is a Motorola 68000-based SUN workstation with Berkeley UNIX(4.2). We have dial-in and dial-out communications capability at 300 and 1200 baud and can process standard magnetic tape at 800, 1600, and 6250 bpi. Proof-quality galleys are produced on a Florida Data OSP-130 printer. We have access to TEX capabilities that are available at the University of Michigan Computing Center: TEX82, a WEB DVI-to-printer driver program for the Xerox 9700 page printer, and a Fortran implementation of METAFONT. The Computing Center will acquire a phototypesetter later in 1983.

Inquiries can be directed to **Textset, Inc.** at 1612 Anderson, Ann Arbor, MI 48104 [Telephone: (313) 996-3566].

TEX82 ORDER FORM

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

Eight different tapes are available. The standard distribution tape contains the source of TEX82 and WEB, the test program, a few "change" files, the collection of fonts in TFM format, and other miscellaneous materials. Six tapes are PXL font collections, recorded in magnifications of 1000, 1100, 1200, 1300, 1400 and 1500 respectively (there is too much to fit on a single tape). The last tape contains the SAIL source for METAFONT and associated materials, including .MF source files.

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

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

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

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

The order form contains a place to record the name and address of the person who should be notified of new TEX releases. This should be the TEX user, not someone in the purchasing department.

Your order will be filled with the most recent versions of software and manuals available from Stanford at the time your order is received. If you are waiting for some future release, please indicate this. Orders are normally filled within 48 hours. There may be periods (like short vacations) when it will take longer. You will be notified of any serious delays. if you want to inquire about your order you may call Maria Code at (408) 735-8006 preferably between 9:30 a.m. and 2:30 p.m. West Coast time.

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

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

TEX82 ORDER FORM

** TAPES ** density (6250, 1600 or 800) = _____

characters (ASCII or EBCDIC) = _____

_____ TEX standard distribution tape.

_____ Font library (1000PXL)

_____ Font library (1100PXL)

_____ Font library (1200PXL)

_____ Font library (1300PXL)

_____ Font library (1400PXL)

_____ Font library (1500PXL)

(1000 = 200 pixels/inch, 1100 = 220 pixels/inch, etc.)

_____ METAFONT (available only in SAIL ASCII)

_____ Total number of tapes.

Tape costs: $82.00 for first tape; $62.00 for each additional.

Tape cost = $ _____

Media costs: $10.00 for each tape required.

Media cost = $ _____

** MANUALS **

_____ TEX82 – $20.00 _____ Test Manual – $8.00

_____ WEB – $10.00 _____ TEXware – $8.00

Manuals cost = $ _____

California orders only: add sales tax = $ _____

Domestic book rate: no charge.
Domestic first class: $2.50 for each tape and each manual.
Export surface mail: $2.50 for each tape and each manual.
Export air mail to North America: $4.00 each.
Export air mail to Europe: $7.00 each.
Export air mail to other areas: $10.00 each.

Postage cost = $ _____

(make checks payable to Maria Code)

Total order = $ _____

Name and address for shipment:

Person to contact (if different):

Telephone _____

Send to: Maria Code, DP Services, 1371 Sydney Dr., Sunnyvale, CA 94087

3/83

Request for Information

The TEX Users Group publishes a membership list containing information about the types of equipment on which members' organizations plan to or have installed TEX, and about the applications for which TEX would be used. It is important that this information be complete and up-to-date.

Please answer the questions below, and also those on the other side of this form, obtaining information from the most knowledgeable person at your installation if necessary. Some sites have more than one computer system on which TEX has been or might be installed. Please list all such machines below. Output device information should be given on the other side.

If you need more space than is provided here, feel free to use additional paper. Your cooperation is appreciated.

- *Send completed form with remittance* (checks, money orders, UNESCO coupons) to:
  TEX Users Group
  c/o American Mathematical Society
  P.O. Box 1571, Annex Station
  Providence, Rhode Island 02901, U.S.A.
- *For foreign bank transfers* the name and address of the AMS bank is:
  Rhode Island Hospital Trust National Bank
  One Hospital Trust Plaza
  Providence, Rhode Island 02903, U.S.A.
- *General correspondence* about TUG should be addressed to:
  TEX Users Group
  c/o American Mathematical Society
  P.O. Box 6248
  Providence, Rhode Island 02940, U.S.A.

Name: _____ Mail to (if different): _____

Address: _____ _____

_____ _____

_____ _____

_____ _____

QTY	ITEM	AMOUNT
	TUG Membership – 1983 (includes TUGboat subscription) @ $20.00 each *	
	TUGboat Library Subscription – 1983 @ $20.00 each *	
	TUGboat Foreign Air Mail Postage Option @ $12.00 each * (see page 3, Vol. 4, No. 1)	
	TUGboat Back Issues 1980 #1 (1 copy, no charge) 1981 @ $10.00/issue 1982 @ $15.00/issue circle issue(s) desired: additional @ $10.00/copy #1, #2, #3 #1, #2	
	The Joy of TEX (revised preliminary edition, 1982) @ $10.00 each	
	TEX Lectures on Tape (see cover 3, Vol. 4, No. 1)	
	Max Díaz's *Fácil TEX* (macro package supplement; revised 10/81) @ $8.00 each	

*Foreign air mail postage option is available for members/subscribers outside North America. (Surface mail postage is included.)

TOTAL ENCLOSED: _____
(*Prepayment in U.S. dollars required*)

* * *

Membership List Information

Institution (if not part of address):

Title:
Phone:
Specific applications or reason for interest in TEX:

My installation can offer the following software or technical support to TUG:

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

Date:
Status of TEX: [ ] Being installed
[ ] Up and running since
[ ] Under consideration
Version of TEX: [ ] SAIL
Pascal: [ ] TEX82 [ ] TEX80
[ ] Other (describe)

From whom obtained:

Approximate number of users:
Computer system(s):

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

Name _____ Institution _____

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

B. Computer to which this device is interfaced
1. Computer name
2. Model
3. Type of architecture*
4. Operating system

C. Output device driver software
[ ] Obtained from Stanford
[ ] Written in-house
[ ] Other (explain)

D. Separate interface hardware (if any) between host computer and output device (e.g. Z80)
1. Separate interface hardware not needed because:
[ ] Output device is run off-line
[ ] O/D contains user-programmable micro
[ ] Decided to drive O/D direct from host
2. Name of interface device (if more than one, specify for each)
3. Manufacturer information
a. Manufacturer name
Contact person
Address
Telephone
b. Delivery time
c. Purchase price
4. Modifications
[ ] Specified by Stanford
[ ] Designed/built in-house
[ ] Other (explain)
5. Software for interface device
[ ] Obtained from Stanford
[ ] Written in-house
[ ] Other (explain)

E. Fonts being used
[ ] Computer Modern: ( ) .tfm ( ) .tfx
[ ] Fonts supplied by manufacturer
[ ] Other (explain)
1. From whom were fonts obtained?
2. Are you using Metafont? [ ] Yes [ ] No

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

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

H. Comments – overview of output device

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