We appreciate all of you who point out our errors.
Here's how to keep score: if you find six errors or more,
you're a professional; three to five errors makes you a trainee;
less than three makes you a desktop publisher.

Frank Romano
"Frank Talk",
TypeWorld (October 24, 1986)

TUGBOAT

THE TEX USERS GROUP NEWSLETTER
EDITOR BARBARA BEETON

VOLUME 8, NUMBER 2 • JULY, 1987
PROVIDENCE • RHODE ISLAND • U.S.A.

TUGboat

The communications of the TEX Users Group are published irregularly at Providence, Rhode Island, and are distributed as a benefit of membership both to individual and institutional members. Three issues of TUGboat are planned for 1987.

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

Submitting Items for Publication

The deadline for submitting items for Vol. 8, No. 3, is September 14, 1987; the issue will be mailed in November.

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

Contributions in camera copy form are encouraged, as is electronic submission of items on magnetic tape or diskette, via electronic mail, or transferred directly to the AMS computer; for instructions, write or call Barbara Beeton.

TUGboat Advertising and Mailing Lists

For information about advertising rates or the purchase of TUG mailing lists, write or call Ray Goucher.

Other TUG Publications

TUG is interested in considering for publication manuals or other documentation that might be useful to the TEX community in general. If you have any such items or know of any that you would like considered for publication, contact Ray Goucher at the TUG office.

TUGboat Editorial Committee

Barbara Beeton, *Editor*
Helmut Jürgensen, *Associate Editor for Software*
Maureen Eppstein, *Associate Editor for Applications*
Laurie Mann, *Associate Editor on Training issues*
Georgia K.M. Tobin, *Associate Editor of Font Forum*
Jackie Damrau, *Associate Editor for LATEX*
Patrick Ion, *Associate Editor for Macros and Problems*
Alan Hoenig and Mitch Pfeffer, *Associate Editors for Typesetting on Personal Computers*

See page 99 for addresses.

Addresses

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

James C. Alexander
Dept of Mathematics
Univ of Maryland
College Park, MD 20742
`alex@eneevax.umd.edu`

Wolfgang Appelt
Gesellschaft für Mathematik und Datenverarbeitung
Schloss Birlinghoven - PF 1240
D-5202 Sankt Augustin 1,
Federal Republic Germany
uucp:
`unido!gmdzi!zi.gmd.dbp.de!appelt`

Richard L. Aurbach
Monsanto Company
800 N Lindbergh Blvd
St Louis, MO 63167
314-694-5453

Elizabeth Barnhart
National EDP Dept
TV Guide
Radnor, PA 19088
215-293-8890

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

Barbara Beeton
American Mathematical Society
P. O. Box 6248
Providence, RI 02940
401-272-9500
`bnb@xx.lcs.MIT.Edu`,
`bb@Sail.Stanford.Edu`

Mike Black
Kingsdown Publishing Ltd
14 Osbaldeston Rd
London N16 7DP, England
01-806 5043; Telex: 937403 ONECOM

Malcolm Brown
ACIS/IRIS
Stanford University
Cypress Hall, Rm E7
Stanford, CA 94305
415-723-1055
`MBB@Portia.Stanford.Edu`

Anne Brüggemann-Klein
Institut für Informatik und Formale Beschreibungsverfahren
Postfach 6980
7500 Karlsruhe,
Federal Republic Germany
0721/608-3705
CSnet: `BRUEGGEM@GERMANY`

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

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

Adrian F. Clark
Analysis Group
British Aerospace
E-Block, Manor Rd
Hatfield
Hertfordshire AL10 9LL, England
070-72-62300 x8216

Maria Code
Data Processing Services
1371 Sydney Dr
Sunnyvale, CA 94087
408-735-8006

John M. Crawford
Computing Services Center
College of Business
Ohio State University
Columbus, OH 43210
614-292-1741
`Crawford-J@Ohio-State`
Bitnet: `TS0135@OHSTVMA`

Jackie Damrau
Dept of Math & Statistics
Univ of New Mexico
Albuquerque, NM 87131
505-277-4623
Bitnet: `damrau@unmb`
UUCP: `damrau@unmvax`

Alec Dunn
School of Electrical Engineering
University of Sydney
NSW 2006, Australia
(02) 692 2014
`alecd%facet.ee.su.oz@seismo.css.gov`

Maureen Eppstein
Administrative Publications
Stanford University
Encina Hall, Room 200
Stanford, CA 94305
415-725-1717
`as.mve@Forsythe.Stanford.Edu`

Michael J. Ferguson
INRS - Télécommunications
Université du Québec
3 Place du Commerce
Verdun (H3E 1H6), Québec Canada
514-765-7834
CSnet: `mike%tel.inrs.cdn@ubc`

Jim Fox
Academic Computing Center HG-45
University of Washington
3737 Brooklyn Ave NE
Seattle, WA 98105
206-543-4320
Bitnet: `fox7632@uwacdc`

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

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

Raymond E. Goucher
TEX Users Group
P. O. Box 9506
Providence, RI 02940
401-272-9500 x232

John Stewart Gourlay
Dept of Computer & Info Science
Ohio State University
2036 Neil Ave Mall
Columbus, OH 43210
614-292-6653
`GOURLAY-J%OSU-20@ohio-state`

Dean Guenther
Computer Service Center
Washington State University
Computer Science Building,
Room 2144
Pullman, WA 99164-1220
509-335-0411
BITnet: `Guenther@WSUVM1`

Doug Henderson
Division of Library Automation
Univ of California, Berkeley
186 University Hall
Berkeley, CA 94720
Bitnet: `dlatex@ucbcmsa`

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

Don Hosek
Platt Campus Center
Harvey Mudd College
Claremont, CA 91711
Bitnet: `dhosek@hmcvax`

Patrick D. Ion
Mathematical Reviews
416 Fourth Street
P. O. Box 8604
Ann Arbor, MI 48107
313-996-5273

Helmut Jürgensen
Dept of Computer Science
Univ of Western Ontario
London N6A 5B7, Ontario, Canada
519-661-3560
Bitnet: A505@UWOCC1
UUCP: helmut@deepthot

Alois Kabelschacht
Max-Planck-Institut für Physik
(Werner-Heisenberg-Institut)
Foehringer Ring 6
D-8000 München 40
Federal Republic Germany
(089) 31893-412

Arthur Keller
University of Texas at Austin
Department of Computer Science
Austin, TX 78712-1188
512-471-7316
ARK@SALLY.UTexas.Edu

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

Gerhard F. Kohlmayr
Mathmodel Press
80 Founders Rd
Glastonbury, CT 06033
203-633-5659

Gideon Koren, M.D.
Hospital for Sick Children
Toronto, Ontario M5G 1X8, Canada

Charles LeHardy
Summer Institute of Linguistics
Box 8987 CRB
Tucson, AZ 85738
602-791-2272
uucp: noao!azsil

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

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

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

Laurie Mann
Stratus Computer
55 Fairbanks Boulevard
Marlboro, MA 01752
617-460-2610
uucp: harvard!anvil!es!Mann

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

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

Gil Pierson
Computer Science Bldg
Washington State University
Pullman, WA 99164

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

Craig Platt
Dept of Math & Astronomy
Machray Hall
Univ of Manitoba
Winnipeg R3T 2N2, Manitoba, Canada
204-474-9832
CSnet: platt%cc.uofm.cdn@ubc

Pedro J. de Rezende
College of Computer Science
Northeastern University
360 Huntington Avenue
Boston, MA 02115
617-437-2078
rezende@corwin.ccs.northeastern.edu

Yasuki Saito
NTT Electrical Communications
Laboratories
NTT Corporation
3-9-11 Midori-cho Musashino-shi
Tokyo 180, Japan
+81 (422) 59-2537
yaski%ntt-20@sumex-aim.stanford.edu

John Sauter
801128 Bates Road
Merrimack, NH 03054
603-881-2301
sauter%dssdev.DEC@decwrl.DEC.COM

E. W. Sewell
3822 Hillsdale Lane
Garland, TX 75042
214-272-0515 x3553

Barry Smith
Kellerman & Smith
534 SW Third Ave
Portland, OR 97204
503-222-4234; TLX 9102404397
Usenet: tektronix!reed!barry

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

Rilla Thedford
Intergraph Corporation, MS HQ013
One Madison Industrial Park
Huntsville, AL 35807
205-772-2440

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

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

Glenn L. Vanderburg
Computing Services Center
Texas A & M University
College Station, TX 77843
409-845-8459
Bitnet: X230GV@TAMVM1

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

Ken Yap
Dept of Computer Science
University of Rochester
Rochester, NY 14627
Ken@Rochester
Usenet: ..!{allegra,decvax,seismo,
cmcl2,harvard,topaz}!rochester!ken

Hermann Zapf
Seitersweg 35
D-6100 Darmstadt
Federal Republic Germany

General Delivery

From the President

Bart Childs

The last issue of TUGboat (Vol. 8, No. 1) represents a measure of success in my mind. The number of contributions, their content, and all other measures of quality made it interesting and useful. I hesitate to mention any one paper, but a number of people have commented about their high interest on several of the papers. Let's keep up the good work.

Several people have been spreading the good word about TeX in national publications. We should publish a listing of these references soon.

Robert McGaffey's note in this issue (page 161) on the **Ideal TeX Driver** poses questions about standards that we need to address soon. Don Knuth created TeX to be portable, but the output drivers are of critical importance in making the system truly portable. I hope that we can have a significant session on this at the Seattle meeting.

Another topic that needs to be addressed is the use of fonts and magnification. It has been an active item in TeXhax. The particular item I am most concerned with is the extensive use of magnification in the LaTeX and SliTeX worlds. The cm family has the needed fonts in 12 and 17 point sizes. Shouldn't we always distribute only magnifications 0, half, 1 and 2? Maybe one or two fonts should have a lot of magnifications for use in titles? Come to Seattle and be ready to argue the points.

One more topic of this type is that we need to make a concerted effort to discard the old am family of fonts. Does anyone have a good reason to keep them around? With the exception of the amssmc fonts, almost all have such a simple change that it seems past due.*

We are looking forward to meeting in the great Northwest. Dean Guenther and Pierre MacKay are coordinating the usual TUG sessions and the TeX in the humanities sessions, respectively.

* Editor's note: We are pleased to announce that this issue of TUGboat has been set with the cm fonts resident on the Math Society's new Autologic APS-μ5 phototypesetter. These fonts are still being tested; however, testing should soon be complete, and they will then be made available from Autologic to other APS users.

Editor's note: The following item appeared in the *New England Journal of Medicine*, Nov. 13, 1986, and is reprinted with permission.

A Simple Way to Improve the Chances for Acceptance of your Scientific Paper

To the Editor: During the past few years we have witnessed a revolution in the way manuscripts, abstracts, and grant proposals are being typed. With improved typewriters and computer programs it is possible to produce manuscripts of typeset quality. It is generally assumed that data should be judged by its scientific quality and that this judgment should not be influenced by typing style.

I challenged this premise by analyzing the rate of acceptance of abstracts by a large national meeting. All abstracts submitted to the 1986 annual meeting of the American Pediatric Society and the Society of Pediatric Research (APS/SPR) appeared in Volume 20, No. 4 (Part 2) (April 1986) of *Pediatric Research.* Contrary to the practice of many other meetings, this volume also includes all the abstracts that were not accepted for presentation, and accepted papers are identified by symbols.

Abstracts were defined as "regularly typed" or "typeset printed." Each abstract was categorized as accepted if chosen for presentation or rejected.

A total of 1965 abstracts were evaluated. Excluded were 47 abstracts assigned for joint internal medicine–pediatric presentation, because the majority of them were submitted to the American Federation for Clinical Research, and there was no indication of their rejection rate; only those that had been accepted appeared in the APS/SPR book of abstracts.

Of the 1918 evaluable abstracts, 1706 were regularly typed and 212 were "typeset." The acceptance rate was significantly higher for the "typeset" abstracts: 107 of 212 (51.4 percent) vs. 747 of 1706 (44 percent) ($P < 0.05$).

Eighty-eight investigators submitted five or more abstracts to the meeting. Here, too, there was a higher rate of acceptance for the "typeset" abstracts (62 of 107; 57.9 percent) as compared with the regularly typed abstracts (184 of 451; 40.8 percent) ($P = 0.002$).

One may argue that investigators who can afford the new equipment for printing abstracts have more money and can afford better research, and therefore that their abstracts are accepted at

©1986 *New England Journal of Medicine*, reprinted with permission.

higher rates. To explore this possibility, I analyzed data on the 15 investigators who submitted five or more abstracts each and who used both typing methods. In this subgroup, 19 of 55 regularly typed abstracts were accepted (34.5 percent), whereas 31 of 53 of the "typeset" abstracts were accepted (58.5 percent) (P = 0.015).

These results demonstrate that the new "typeset" appearance of data increases the chance of acceptance. It may mean that "typeset" printing may cause the data to look more impressive. Alternatively, it may mean that the new printing makes it easier for reviewers to read the data and to appreciate its meaning.

Most important, it means that this technological innovation reduces the chance of success of those not currently using it.

GIDEON KOREN, M.D.
Hospital for Sick Children
Toronto, ON M5G 1X8, Canada

Software

Tib: a Reference Setting Package, Update

J. C. Alexander
University of Maryland

There have been a number of minor bug fixes and some refining of features of the TEX bibliography setter Tib (see TUGboat vol. 7, no. 3, for an article about Tib). Its version number has been incremented. Those people who asked to be put on my mailing list have been sent all the changes. However, I know from mail that there are a number of other users, presumably people who picked it up via anonymous ftp. Those people might want to check the file `CHANGES` and/or `READ.ME` via anonymous ftp from `eneevax:pub/tib`. Incidentally, I appreciate the kind comments and suggestions people have made. It seems Tib is proving to be a useful adjunct to TEX.

Portuguese Hyphenation Table for TEX

Pedro J. de Rezende
Northeastern University

I have compiled a Portuguese hyphenation table for TEX. It turns out to be a rather short table (compared to the one for English) because Portuguese has very concise rules for hyphenation. I'd like to make this table public and freely distributed. Even included in the distribution tapes. I have extensively tested it (with `patgen`) and haven't found any erroneous hyphenation. It does miss some hyphens but they are very, very few. It certainly does not hyphenate a word beyond an accent or a cedilla, but that's the way TEX handles hyphenation of words with intervening macros (see Appendix H of *The TEXbook*).

Editor's note: Arrangements are being made to include the Portuguese hyphenation table in the standard distribution. Hyphenation tables for languages other than English are frequently requested on TEXhax; anyone who knows of the existence of such tables is asked to send the relevant information to Barbara Beeton, so that a list can be compiled for the next issue of TUGboat.

A (Hopefully) Final Extension of Multilingual TÊX

Michael J. Ferguson
INRS-Télécommunications
Montréal, Canada

This note reports the, hopefully, final extension to TEX that allows for multilingual hyphenation reported in July 1985 (Vol. 6, No. 2, pp. 57–58) and March 1986 (Vol. 7, No. 1, page 16) of TUGboat. The key feature of the extension is that it accommodates **standard TEX fonts**, including words with accented letters. For details of the features the reader should refer to the July 1985 TUGboat. This note reports some recent extensions to accommodate certain typographical and input conventions in non-English text. These extensions are as follows:

- TeX will now hyphenate words that have an explicit `\discretionary`. Each part of the word including the discretionary is treated as a separate word for hyphenation purposes. This allows for the hyphenation of words such as "Wechselstromwecker" where the "ck" is represented by `\discretionary{k-}{k}{ck}`. The hyphens then given by `\showhyphens` are "Wech-sel-stromwek-ker".

  The discretionary hyphen approach also allows for the suppression of an unwanted ligature. This can be done by inserting a discretionary hyphen in the appropriate place. Thus the unwanted ligature in "auffrischen" is defeated with the insertion of `\-` after the first "f" to give "auffrischen". Note that the solution to exercise 5.1 in early editions of the TeXbook is incorrect as it will not survive a second pass of a paragraph. The second hyphen after the "i" remains with the extension.

  The extension is invoked by making the integer parameter `\dischyph` non-zero. Thus `\dischyph=1` will allow, on a paragraph-by-paragraph basis, hyphenation of words that have an embedded `\discretionary`. Note that by using an empty discretionary, a break is allowed without inserting a hyphen character.

- Two new integer parameters, `\starthyph` and `\stophyph`, have been defined. These allow the number of characters at the beginning and end of words that suppress hyphenation to be modified. The defaults are 2 and 3 respectively as in standard TeX. The minimum length of a word to be hyphenated is the sum of these two values. If necessary, a third independent integer parameter that specifies the minimum length of a hyphenated word could be added.

- In order to handle special keyboards with extended characters encoded outside the standard ASCII set, all characters with codes outside this set have been declared **permanently** active. This means that both the single character and the single character command may be separately defined. This would allow special discretionary sequences for various languages to be input easily.

  This extension has been in use on the VT-200 series of terminals by Digital and will work equally well for IBM-PCs. Since this modification takes effect at TeX's mouth, the extended characters never make it "inside". This means that they should not be used in definitions. For example if ü is one of these extended characters, then the definition `\Hühn{`⟨*text*⟩`}` actually defines `\H` and not the entire `\Hühn`. Interestingly enough, TeX will not complain. It just sticks the rest into the parameter argument.

Hopefully these extensions will be suitable for most other languages — either independently or together — assuming that the appropriate patterns and exceptions exist.

Report on jTeX: A Japanese TeX

Yasuki Saito
NTT Electrical Communications Laboratories
Japan

This is a short report of the current status of Japanese TeX, called jTeX. I do not try to give a detailed description of every nook and cranny. Instead, I will concentrate on giving the overview of what I have done to make TeX typeset Japanese text as well as English.

Example

First of all, look at the example input file and the corresponding output generated by jTeX in Appendix 1. (The input file listing was generated using the JLaTeX verbatim mode.) It is an excerpt from a famous textbook on analysis written by Teiji Takagi. jTeX is an upward compatible extension of TeX and everything in TeX is at your disposal. So you will find familiar control sequences in the input file. The only difference is of course that there are lots of Japanese characters in it! Actually, from the user's point of view, the fact that he can enter Japanese characters into the input file is the main difference although he must learn a few new control sequences to select fonts and to control spacing.

Two major problems

Many people think that it is difficult to make Japanese TeX, but it is not so. There were two problems to be solved in making jTeX. One was to make TeX's input mouth a little bit wider so that it can swallow Japanese characters. The second, and more serious problem, was to prepare the fonts for more than 6000 Japanese characters.

Knuth suggests a way to extend TeX to oriental languages in "TeX: The Program", page 57. His suggestion is to extend the data structure for a character so that TeX can handle more fonts each having more characters in it. I chose not to extend the data structure nor the font file format for various reasons. A GF file with information for 6000 Japanese characters in it is just too large to maintain. Another reason is that if you stick to the original data structure you can make modifications minimum. And with the ordinary font file format you can use various utility programs without modification.

Thus I divided Japanese characters into 33 subfonts each having at most 256 characters. TeX can handle maximum of 256 fonts at a time, and reserving 33 fonts for a single Japanese font may seem to be extravagant. However in actual use, one rarely uses all 6000 characters. A statistic says that the most frequent 2000 characters will cover 99% of ordinary Japanese text, so the actual requirements are much less than 33 subfonts.

Once decided on the font configuration, it is straightforward to modify TeX. TeX's input mouth is extended to eat Japanese characters and send an appropriate (subfont, character number) pair to its stomach. After that, TeX doesn't notice that it is actually handling Japanese characters!

As for the preparation of Japanese fonts, I didn't use METAFONT. Considering the amount of effort Knuth has spent to generate Computer Modern Typefaces, it would be a five- to ten-year project to devise a good METAFONT definition for all the Japanese characters. Although it will be necessary in the future, I am just content with available dot fonts for the time being and generated necessary font files from them directly.

Japanese character set

Before explaining the division of the Japanese character set into subfonts, it is necessary to explain what we have first. JIS (Japanese Industry Standard) C-6226 defines a "Code of the Japanese Graphic Character Set for Information Interchange". Here in Japan, we usually use this code (referred to as "JIS code" for short) to represent Japanese characters. It contains 6877 characters in total and uses two 7-bit bytes to represent a single character. These two bytes are called "ku" and "ten" in Japanese or simply "first byte" and "second byte". These bytes are taken from the non-control character part of the ASCII character set. Thus you can use ASCII control characters such as Tab, Carriage Return and Line Feed within a sequence of two-byte codes. See the table in Appendix 2 (This table is typeset by jTeX). In this 94 by 94 table, each Japanese character is positioned at the intersection point of first byte row and second byte column. Each byte is represented by corresponding ASCII character in the outermost column and row. Hexadecimal representation of each byte and "ku", "ten" numbers are added for convenience. All characters are grouped into natural categories as follows (we use the word "ku" to refer to a set of characters having the same first byte):

1-ku & 2-ku	symbols
3-ku	numerals & roman alphabets
4-ku	hiragana (phonetic symbols)
5-ku	katakana (phonetic symbols used to represent foreign words)
6-ku	greek alphabets
7-ku	russian alphabets
8-ku	line segments
16,...,47-ku	2965 first level kanji ordered according to their representative reading
48,...,84-ku	3388 second level kanji ordered by radicals and number of strokes

Here the separation of the kanji set into two levels is very important. The first level contains most frequently used kanji while the second level kanji are rarely used. A normal Japanese sentence consists of kanji, hiragana, katakana and some punctuation symbols. But you can freely mix foreign alphabets within a Japanese sentence, and we do write such a mixture from time to time, so various foreign alphabets are also included in this table.

There are several ways to represent a file with both ASCII and JIS characters in it. If your machine uses an 8-bit byte to represent an ASCII character, simply turning the most significant bit on for all two-byte codes enables you to distinguish ASCII and JIS code easily. This is used in VAX Kanji Code. Some Japanese word processors use so called Shift JIS code which also uses two 8-bit bytes. However the most widely used internal representation is to use escape sequences. JIS codes are simply represented by a sequence of two 7-bit bytes and a sequence of them are sandwiched between three-byte escape sequences ("`<esc>$@`" or "`<esc>$B`" to start and "`<esc>(J`" or "`<esc>(B`" to end.)

These various formats are easily interchangeable. So jTeX assumes that its input file is a sequence of 7-bit ASCII codes with JIS code parts surrounded by escape sequences.

Division into subfonts

The Japanese character set described in the previous section is divided into the following 33 subfonts. This division naturally corresponds to the categories mentioned above. The control sequence name for each subfont is used to refer to the individual characters in each subfont. Usually a user is not aware of the existence of subfonts, but if he wishes, he can specify, say, the second character in 4-ku, by "`{\jhira\char2}`".

`\jsy`	1-ku & 2-ku (symbols)
`\jroma`	3-ku (numerals & roman alphabets)
`\jhira`	4-ku (hiragana)
`\jkata`	5-ku (katakana)
`\jgreek`	6-ku (greek alphabets)
`\jrussian`	7-ku (russian alphabets)
`\jkeisen`	8-ku (line segments)
`\ja`,...,`\jl`	16-ku,...,47-ku (first level kanji)
`\jm`,...,`\jz`	48-ku,...,84-ku (second level kanji)

In each subfont, a character code corresponds to "ten" number except in kanji subfonts. 26 kanji subfonts (`\ja`,`\jb`,...,`\jz`) all have 256 kanji characters in them except `\jl` and `\jz`. Kanji in each level are densely packed into 256 character positions of each subfont in their order. So the last subfont in level 1 (`\jl`) has only 49 ($= 2965 - 256 \times 11$) kanji and the last one in level 2 (`\jz`) has only 60 ($= 3388 - 256 \times 13$). Appendix 3 shows the font tables for several subfonts generated by jTeX and the ordinary `testfont.tex`. Note that these control sequences are generic, i.e. these subfont selectors are assigned the actual subfont by a single control sequence defined in `jplain.tex` (plain file for jTeX, see below).

Font selection

jTeX provides several different fonts for Japanese and `jplain.tex` defines useful font selectors which switch all the necessary subfonts at once. For example, a default font is selected by a following control sequence (`\jstd`) in `jplain.tex` (This definition is simplified a little):

```
\font\djsystd=dnpjsy38
\font\djhirastd=dnpjhira38
\font\djkatastd=dnpjkata38
\font\djastd=dnpjka38
\jfont\djbstd=dnpjkb38 dnpjka38
\jfont\djcstd=dnpjkc38 dnpjka38
...
\jfont\djkstd=dnpjkk38 dnpjka38
\font\djlstd=dnpjkl38
\def\jstd{\let\jsy=\djsystd
\let\jhira=\djhirastd
\let\jkata=\djkatastd
\let\ja=\djastd \let\jb=\djbstd
...
\let\jk=\djkstd \let\jl=\djlstd
\baselineskip=18pt
\jintercharskip=0.0pt plus0.08pt
\jspaceskip=9.1542pt %38dots on 300dpi
\jasciikanjiskip=1.66667pt
 plus0.83333pt minus0.55556pt}
```

Note that only 15 subfonts (corresponding to `\jsy`, `\jhira`, `\jkata`, `\ja`, ..., `\jl`) are preloaded and switched by this command. Users must specify each subfont separately if they want to use foreign alphabets or level 2 kanji. However it is much better to use TeX's fonts for roman, greek and even russian alphabets in normal application. So we encourage people to use TeX's fonts instead of JIS foreign alphabets. `\jsmall` used in the example input file of Appendix 1 is another example of a font selector.

The control sequence `\jfont` is introduced to save jTeX's memory space for font information. Most kanji subfonts have identical TFM file and the use of this command:

```
\jfont\fontname=fontfile1 fontfile2
```

enables to load `fontfile1` as `\fontname` using the already loaded font information for `fontfile2`. Thus it does not consume any font space at all.

Modification to TeX's input mouth

Now we can state the task of jTeX's input mouth clearly. Treat every character in the input file as TeX does except for JIS codes surrounded by escape sequences. For those two-byte codes, deceive TeX as if it has seen the corresponding subfont selector and an appropriate `\char` command. For example, if you have the following line in the input file:

```
...<esc>$@$3$1$OF|K\81$G$9!#<esc>(J...
```

it should be seen as if they were:

```
...{\jhira\char19\char76\char47}%
{\ji\char111}{\jk\char37}{\jd\char59}%
{\jhira\char39\char25}{\jsy\3}...
```

(Try to decipher it using the font table in Appendix 2.) There is a little lie in this description. jTeX performs two other things to ensure the proper treatment of Japanese text. First, it inserts `\jintercharskip` between every pair of Japanese characters. Secondly, it inserts `\jasciikanjiskip` between an ASCII character and a Japanese one.

For the normal setting of these glues in `jplain.tex`, see the excerpt in the previous section.

The first glue ensures that Japanese sentences can break at any point except "Kinsoku Shori" explained below. And if it is necessary, this glue can stretch a bit to enable right justification. The second glue puts an appropriate amount of space between an English word and a Japanese character.

The internal data structure for tokens is also extended, but I do not describe it here.

Kinsoku Shori

Normal Japanese sentences can break at any point as I stated above. But there are exceptions. These exceptions are called "Kinsoku" in Japanese and the proper treatment of "Kinsoku" is "Kinsoku Shori". Certain characters cannot appear at the beginning of a line (such as close parenthesis and comma) and certain other characters (such as open parenthesis) cannot happen at the end of a line. These conditions are naturally met in TEX if you write these characters next to or just before the neighboring character without inserting space. But in jTEX, glues are put into every gap between Japanese characters so you need to get rid of this extra glue between a "Kinsoku" character and its neighbor.

Spacing

Although the number of characters are many, the saving feature of Japanese characters is that they all have the same width and height. There is no kerning, no ligatures, so typesetting is simpler than English in a sense. But there are a few characters you must pay attention to. They are punctuation marks such as period and comma. We have Japanese period "maru" (1-ku 3-ten) and Japanese comma (1-ku 4-ten). For these characters jTEX provides "Japanese space factor code" (`jsfcode`) whose function is similar to that of `sfcode` in TEX. Whenever jTEX encounters a Japanese character, this `jsfcode` is used instead of `sfcode`.

Carriage return is treated a little bit differently in jTEX. Single carriage return in JIS characters is not equivalent to space. So no extra glue is inserted there. But two or more consecutive carriage returns has the same effect of ending a paragraph as in TEX. ASCII space cannot appear among JIS characters but Tab can appear because it is one of the control characters. This Tab is simply dropped by jTEX. To put it in another words, single carriage return, tab or nothing between Japanese characters are all converted to `\jintercharskip` by jTEX except Kinsoku Shori.

JIS space (1-ku 1-ten, two byte code is "!!") is treated as a normal Japanese character (although it is invisible), so it gives you exactly one character-wide space on output.

Generation of font files from dot fonts

For TEX and device drivers to work, we need two kinds of font files: GF files and TFM files. I generated them from dot fonts by a simple LISP program.

For the first few months, I tried to gather as many Japanese fonts in dot format as possible to enhance jTEX's Japanese fonts. There were not many, but I have found two 24-dot fonts and two 32-dot fonts. One of the 24-dot fonts is part of the JIS standard for dot printers and one can freely copy and distribute it. In the beginning, there was no other way, so I mechanically generated 36, 48 and 72-dot fonts from this JIS 24-dot font to satisfy the need for larger fonts.

But recently we started collaboration with DNP (Dai Nippon Printing Co., one of the biggest printing companies in Japan), and they provide us with fonts of various sizes. We found out that a 38-dot font goes well with TEX's standard 10 point font, so we are preparing the fonts (both Mincho style and Gothic style) with the following dot sizes:

	5pt	6pt	7pt	8pt	9pt	10pt	12pt	17pt
1	19	23	27	30	34	38	46	65
$\sqrt{1.2}$	21	25	29	33	37	42	50	71
1.2	23	27	32	36	41	46	55	78
1.2^2	27	33	38	44	49	55	66	93
1.2^3	33	39	46	53	59	66	79	112
1.2^4	39	47	55	63	71	79	95	134
1.2^5	47	57	66	76	85	95	113	161

These are for the 300 dpi printers, so if you change the resolution you need different sizes as well.

Another important factor when generating Japanese fonts from dot fonts is where to draw the baseline. If you put a box surrounding a Japanese character just on the baseline, non-uppercase ASCII characters look sunk under the baseline. It is difficult to find the optimal point, but we experimentally settled on the following solution. Place the baseline one sixth of the box height above the bottom edge of the box.

Modification to device drivers

As I stated earlier, I tried to make the necessary modification as small as possible. But you need some modification to the device drivers if you want to run the whole system efficiently.

For example, we are now using DEC2065 and IMAGEN 8/300 and 3320 printers. A device driver for this combination is known as DVIIMP. This program loads all the information for a font when it first encounters a new font. And this becomes a great overhead if you use it on dvi files generated by jTEX. Japanese kanji are grouped into subfonts only by code order and there is no "working set" property ("use of one character in a font implies the use of other characters in the same font for a while") among them. So I modified this device driver to load the font information of each character one by one.

On UNIX machines (we use SUN-2 and SUN-3), there is no device driver which directly uses the GF file and most of them use old PXL files. So Japanese fonts are converted to extended PXL format (extended because it contains more than 128 characters, but the basic structure is the same), and device drivers are modified to accept this extended PXL formats.

I made a similar modification to the previewer in X-window system (xdvi) and it is running on our SUNs.

Restrictions

jTEX is quite general and can be used as widely as TEX itself. But there are still minor restrictions.

- You cannot use Japanese characters directly in math mode. But you can always escape to horizontal mode using `\hbox`, so this is not a real restriction.
- You cannot use Japanese characters in control sequence names. But no one has ever wanted to do that until now.
- The number of Japanese fonts usable in a job is limited. This limit is 17, because one Japanese font preloaded by jTEX normally consists of 15 subfonts and jTEX allows a maximum of 256 fonts. If you use other fonts of TEX or level 2 kanji or JIS foreign alphabets besides normally loaded Japanese fonts, you must be content with fewer Japanese fonts. But for ordinary purposes, this number is just enough. If you try to typeset a really big dictionary, you may reach this limit. But well before that you will face the following restriction.
- The IMAGEN print server we are using allows only 3072 different characters per job, and only 653302 bytes for font information per job. This is a real restriction for Japanese TEX. For example, Appendix 1 (only 4 pages!) cannot be output at once and you need to separate it into individual pages.

A bit of history

There have been several attempts to use TEX to format Japanese text. The first and pioneering work was done by Fujita [1]. It was based on TEX78. The modification of SAIL code together with the improvements on output device was carried out to make a usable system. I heard that it is still running at his lab, but now it is obsolete.

Another one was reported by Nagashima and Kawabata [2] at the second Japanese TEX Users Group meeting. (For information on the Japanese TEX Users Group, see TUGboat vol.7, no.3, p.192.) They preprocess a file containing Japanese text to feed it into TEX's mouth. I got hints from their work, so this preprocess is similar to jTEX's input processing, but they literally converted Japanese characters to font selector and `\char` pair producing an expanded intermediate file. They pointed out various problems: inability to use more than two Japanese fonts (they have been working with PXL font files), incompatibility with TEX's magnification sequence, to name a few.

After hearing their talk, I quickly realized that you can dispense with the preprocessor if you use the macro facility of TEX. So the first version of jTEX was realized as a macro package without changing TEX itself [3]. Although the quality of fonts was not so good in the beginning, people were amazed by the fact that TEX can typeset Japanese with only about 500 lines of macros! I did the first version just to see the feasibility of using TEX for Japanese, but people around us started to use jTEX as a daily tool and lots of them complained about its inefficiency and poor font quality.

To improve the efficiency I internalized what the macro does by modifying TEX itself. The number of changes required is not so great (about 40 change items are added to the change file) and the result is superb! The current version of jTEX processes a file with Japanese as fast as original TEX. Also LATEX, AMS-TEX etc. have been extended just by preloading `lplain.tex` and `amstex.tex` into jTEX.

To improve the font quality, we have just started collaboration with DNP (Dai Nippon Printing Co.). They have their own high quality font in vector format, and they kindly provide us with dot format versions of various sizes. An example output in Appendix 1 and sample font tables in Appendix 3 use these fonts from DNP.

Availability

jTEX is public domain software. It now works on DEC2065 under TOPS-20 version 6.1 and on various UNIX 4.2bsd systems. Several universities have began to use it on their UNIX machines. If you have a running TEX with a decent device driver, jTEX should work too without trouble. (You may need to modify your driver a bit.)

Font files generated from the JIS 24-dot font are also public and can be obtained from the author.

Several modified utility programs such as a previewer are also available.

Future work

What is necessary for using TEX to typeset Japanese is almost completed with jTEX. But there remain many things to be done if you consider jTEX as a total typesetting system.

- Some Japanese texts are written up to down. And we need to support that. But this is rather simple. Just rotate the font 90 degrees counterclockwise and adjust the centerline of each character if necessary.
- We need to build a collection of macro packages to facilitate the use of jTEX in various applications. Locally various forms are converted to jTEX format, and several Japanese academic societies show interest in using jTEX for the publication of their journals. You may see the emergence of JMS-jTEX in the near future....
- It may be necessary to enlarge the number of Japanese fonts usable in one document. This is not so difficult. I could have done so if I wished, and I am ready to do so if there are sufficient demands.
- Enhancement of Japanese fonts is really needed. To define all Japanese characters in METAFONT is a great challenge. And in the long run, someone or a group of people, preferably consisting of both font designers and computer scientists, must do it.

Acknowledgement

My thanks go to many people. Mr. Enari and Mr. Ishii of DNP for providing us with the good dot fonts in various sizes, Prof. Samuel for helping me to modify DVIIMP and giving me useful information on IMAGEN printers. Mr. Amamiya and Mr. Goto who provided the machine environment in which I can work, and many colleagues who used and criticized and found bugs in early versions of jTEX.

Finally I want to mention that without "TEX: The program" my endeavor to extend TEX to Japanese typesetting would have taken more time than I could afford. It has been both fun and exciting to read through the book seeking the best solution for the problem. So my thanks also go to Prof. Knuth.

References

[1] Hiroshi FUJITA: "Technical document typesetting system: TEX" (in Japanese), Information Processing, vol.25, no.8, pp.848–853 (Aug. 1984).

[2] Masaaki NAGASHIMA and Youichi KAWABATA: "Printing Japanese language using TEX" (in Japanese), handout of the 2nd Japanese TEX Users Group meeting (Jul. 1986).

[3] Yasuki SAITO: "Japanese TEX" (in Japanese), Working Group on Japanese Document Processing 10-3, IPJSS (Jan. 1987).

Appendix 1 Sample Input file and Output generted by jTEX

```
% 高木貞治 解析概論 改訂第三版 101ページより抜粋 %
\font\ninerm=cmr9 \font\ninei=cmmi9 \font\ninesy=cmsy9
\nopagenumbers \parindent=\jspaceskip
\def\small{\textfont0=\ninerm \scriptfont0=\sevenrm \textfont1=\ninei
\scriptfont1=\seveni \textfont2=\ninesy \scriptfont2=\sevensy}

定理35．$f(x)$が積分区間内の一点において連続ならば，その点において積分函数$F(x)$は微分可能で
$$ F'(x)=f(x). $$
[証] まず$h>0$として，前のように
$$ F(x+h)-F(x)=\int_{x}^{x+h}f(t)dt. $$
\noindent従って
$$ m \leq {{F(x+h)-F(x)}\over h} \leq M,\qquad\hbox{(\S31, ($7^\circ$)),}$$
\noindent$M, m$は$[x,x+h]$における$f(x)$の値の上限，下限である． 従って
$$ \biggl| {{F(x+h)-F(x)}\over h} - f(x) \biggr| \leq M-m. $$
\noindentさて$f(x)$の連続性によって，$\varepsilon>0$に対して$\delta$を十分小さく
取って，$h<\delta$のとき$M-f(x)<\varepsilon, f(x)-m<\varepsilon,$
従って$M-m<2\varepsilon$ならしめることができる．\par
$h<0$としても同様であるから$F'(x)=f(x)$.

{\jsmall\ninerm\parindent=1.5cm
\item{[注意1]} $\small x$において$\small f(x)$が右へ，あるいは左へ，連続ならば
$\small D^{+}F(x)=f(x)$あるいは$\small D^{-}F(x)=f(x)$.
\item{[注意2]} 同じ条件の下において，積分$\small \int_{x}^{b}f(x)dx$を下の限界に
関して微分すれば$\small -f(x)$を得る．それは$\small \int_{x}^{a}=-\int_{a}^{x}$だから，当
然である．}

逆に$f(x)$が連続で，その一つの原始函数$F(x)$が知られるときは，それを用いて
$f(x)$の積分が計算される．すなわちその場合，かりに
$$ F_1(x)=\int_{a}^{x} f(x)dx $$
\noindentと書けば，${F_1}'(x)=f(x), F'(x)-{F_1}'(x)=0$. 故に$F(x)-F_1(x)=C$は定数である．
すなわち$$ \int_{a}^{x}f(x)dx=F(x)+C.$$
\noindentここで$x=a$とすれば，$0=F(a)+C$, 故に$C=-F(a)$, 従って，$x=b$とすれば
$$ \int_{a}^{b}f(x)dx=F(b)-F(a). \eqno(1) $$
これを微分積分法の基本公式という．

{\jsmall\ninerm
積分$\small \int_{a}^{x}f(x)dx$の上の限界を変数とし，下の限界を任意の定数とすれば，
その定数をどうきめても，差は$\small x$に無関係である．すなわち$\small f(x)$が積分可能なる区間
に属する任意の定数$\small a, a'$に関して
$\small \int_{a'}^{x}=\int_{a}^{x}-\int_{a}^{a'}$
で，$\small \int_{a}^{a'}$は$x$に関係しない．このように積分の下の限界なる定数を
指定しない場合に，積分を限界なしに$\small \int f(x)dx$と書いて，それを不定積分という．
$\small f(x)$が連続函数ならば，不定積分は原始函数と同意語である．

基本公式(1)は，要約すれば連続函数に関する限り，微分と積分とが互いに逆な算法で
あることを意味する．もしも連続性を仮定しないならば，この関係は成立しない．
すなわち$\small F'(x)=f(x)$でも$\small f(x)$は必ずしも連続でなく，従って必ずしも積分可能でな
いが，また積分可能でも積分函数は$\small F(x)$と合致するとはいわれない．
$\small \int_{a}^{x}f(x)dx$は必ず連続であるけれども，それは必ずしも微分可能でなく，微分可能で
も微分商は$\small f(x)$と合致するとは限らない．連続函数以外では、微分積分法はむずかしい！
\vfill\eject}
\end
```

定理35. $f(x)$ が積分区間内の一点において連続ならば，その点において積分函数 $F(x)$ は微分可能で

$$F'(x) = f(x).$$

[証] まず $h > 0$ として，前のように

$$F(x+h) - F(x) = \int_x^{x+h} f(t)dt.$$

従って

$$m \leq \frac{F(x+h) - F(x)}{h} \leq M, \qquad (\S 31, (7^\circ)),$$

M, m は $[x, x+h]$ における $f(x)$ の値の上限，下限である．従って

$$\left| \frac{F(x+h) - F(x)}{h} - f(x) \right| \leq M - m.$$

さて $f(x)$ の連続性によって，$\varepsilon > 0$ に対して δ を十分小さく取って，$h < \delta$ のとき $M - f(x) < \varepsilon, f(x) - m < \varepsilon$, 従って $M - m < 2\varepsilon$ ならしめることができる．

$h < 0$ としても同様であるから $F'(x) = f(x)$.

[注意 1] x において $f(x)$ が右へ，あるいは左へ，連続ならば $D^+F(x) = f(x)$ あるいは $D^-F(x) = f(x)$.

[注意 2] 同じ条件の下において，積分 $\int_x^b f(x)dx$ を下の限界に関して微分すれば $-f(x)$ を得る．それは $\int_x^a = -\int_a^x$ だから，当然である．

逆に $f(x)$ が連続で，その一つの原始函数 $F(x)$ が知られるときは，それを用いて $f(x)$ の積分が計算される．すなわちその場合，かりに

$$F_1(x) = \int_a^x f(x)dx$$

と書けば，${F_1}'(x) = f(x), F'(x) - {F_1}'(x) = 0$. 故に $F(x) - F_1(x) = C$ は定数である．すなわち

$$\int_a^x f(x)dx = F(x) + C.$$

ここで $x = a$ とすれば，$0 = F(a) + C$, 故に $C = -F(a)$, 従って，$x = b$ とすれば

$$\int_a^b f(x)dx = F(b) - F(a). \tag{1}$$

これを微分積分法の基本公式という．

積分 $\int_a^x f(x)dx$ の上の限界を変数とし，下の限界を任意の定数とすれば，その定数をどうきめても，差は x に無関係である．すなわち $f(x)$ が積分可能なる区間に属する任意の定数 a, a' に関して $\int_{a'}^x = \int_a^x - \int_a^{a'}$ で，$\int_a^{a'}$ は x に関係しない．このように積分の下の限界なる定数を指定しない場合に，積分を限界なしに $\int f(x)dx$ と書いて，それを不定積分という．$f(x)$ が連続函数ならば，不定積分は原始函数と同意語である．

基本公式 (1) は，要約すれば連続函数に関する限り，微分と積分とが互いに逆な算法であることを意味する．もしも連続性を仮定しないならば，この関係は成立しない．すなわち $F'(x) = f(x)$ でも $f(x)$ は必ずしも連続でなく，従って必ずしも積分可能でないが，また積分可能でも積分函数は $F(x)$ と合致するとはいわれない．$\int_a^x f(x)dx$ は必ず連続であるけれども，それは必ずしも微分可能でなく，微分可能でも微分商は $f(x)$ と合致するとは限らない．連続函数以外では、微分積分法はむずかしい！

Appendix 2 JIS Code Table (top left quarter)

			!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
			21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
			01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
!	21	01		、	。	，	．	・	：	；	？	！	゛	゜	´	｀	¨	＾	￣	＿	ヽ	ヾ	ゝ	ゞ	〃	仝	々	〆	〇	ー	―	‐	／	＼	～	∥	｜	…	‥	‘	’	“	”	（	）	〔	〕	［	］
"	22	02	◆	□	■	△	▲	▽	▼	※	〒	→	←	↑	↓	〓												∈	∋	⊆	⊇	⊂	⊃	∪	∩									∧	∨	¬	⇒	⇔	∀
#	23	03																０	１	２	３	４	５	６	７	８	９								Ａ	Ｂ	Ｃ	Ｄ	Ｅ	Ｆ	Ｇ	Ｈ	Ｉ	Ｊ	Ｋ	Ｌ	Ｍ	Ｎ	Ｏ
$	24	04	ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
%	25	05	ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
&	26	06	Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω									α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
'	27	07	А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э	Ю	Я														
(	28	08	─	│	┌	┐	┘	└	├	┬	┤	┴	┼	━	┃	┏	┓	┛	┗	┣	┳	┫	┻	╋	┠	┯	┨	┷	┿	┝	┰	┥	┸	╂															
)	29	09																																															
*	2A	10																																															
+	2B	11																																															
,	2C	12																																															
-	2D	13																																															
.	2E	14																																															
/	2F	15																																															
0	30	16	亜	唖	娃	阿	哀	愛	挨	姶	逢	葵	茜	穐	悪	握	渥	旭	葦	芦	鯵	梓	圧	斡	扱	宛	姐	虻	飴	絢	綾	鮎	或	粟	袷	安	庵	按	暗	案	闇	鞍	杏	以	伊	位	依	偉	囲
1	31	17	院	陰	隠	韻	吋	右	宇	烏	羽	迂	雨	卯	鵜	窺	丑	碓	臼	渦	嘘	唄	欝	蔚	鰻	姥	厩	浦	瓜	閏	噂	云	運	雲	荏	餌	叡	営	嬰	影	映	曳	栄	永	泳	洩	瑛	盈	穎
2	32	18	押	旺	横	欧	殴	王	翁	襖	鴬	鴎	黄	岡	沖	荻	億	屋	憶	臆	桶	牡	乙	俺	卸	恩	温	穏	音	下	化	仮	何	伽	価	佳	加	可	嘉	夏	嫁	家	寡	科	暇	果	架	歌	河
3	33	19	魁	晦	械	海	灰	界	皆	絵	芥	蟹	開	階	貝	凱	劾	外	咳	害	崖	慨	概	涯	碍	蓋	街	該	鎧	骸	浬	馨	蛙	垣	柿	蛎	鈎	劃	嚇	各	廓	拡	撹	格	核	殻	獲	確	穫
4	34	20	粥	刈	苅	瓦	乾	侃	冠	寒	刊	勘	勧	巻	喚	堪	姦	完	官	寛	干	幹	患	感	慣	憾	換	敢	柑	桓	棺	款	歓	汗	漢	澗	潅	環	甘	監	看	竿	管	簡	緩	缶	翰	肝	艦
5	35	21	機	帰	毅	気	汽	畿	祈	季	稀	紀	徽	規	記	貴	起	軌	輝	飢	騎	鬼	亀	偽	儀	妓	宜	戯	技	擬	欺	犠	疑	祇	義	蟻	誼	議	掬	菊	鞠	吉	吃	喫	桔	橘	詰	砧	杵
6	36	22	供	侠	僑	兇	競	共	凶	協	匡	卿	叫	喬	境	峡	強	彊	怯	恐	恭	挟	教	橋	況	狂	狭	矯	胸	脅	興	蕎	郷	鏡	響	饗	驚	仰	凝	尭	暁	業	局	曲	極	玉	桐	粁	僅
7	37	23	掘	窟	沓	靴	轡	窪	熊	隈	粂	栗	繰	桑	鍬	勲	君	薫	訓	群	軍	郡	卦	袈	祁	係	傾	刑	兄	啓	圭	珪	型	契	形	径	恵	慶	慧	憩	掲	携	敬	景	桂	渓	畦	稽	系
8	38	24	検	権	牽	犬	献	研	硯	絹	県	肩	見	謙	賢	軒	遣	鍵	険	顕	験	鹸	元	原	厳	幻	弦	減	源	玄	現	絃	舷	言	諺	限	乎	個	古	呼	固	姑	孤	己	庫	弧	戸	故	枯
9	39	25	后	喉	坑	垢	好	孔	孝	宏	工	巧	巷	幸	広	庚	康	弘	恒	慌	抗	拘	控	攻	昂	晃	更	杭	校	梗	構	江	洪	浩	港	溝	甲	皇	硬	稿	糠	紅	紘	絞	綱	耕	考	肯	肱
:	3A	26	此	頃	今	困	坤	墾	婚	恨	懇	昏	昆	根	梱	混	痕	紺	艮	魂	些	佐	叉	唆	嵯	左	差	査	沙	瑳	砂	詐	鎖	裟	坐	座	挫	債	催	再	最	哉	塞	妻	宰	彩	才	採	栽
;	3B	27	察	拶	撮	擦	札	殺	薩	雑	皐	鯖	捌	錆	鮫	皿	晒	三	傘	参	山	惨	撒	散	桟	燦	珊	産	算	纂	蚕	讃	賛	酸	餐	斬	暫	残	仕	仔	伺	使	刺	司	史	嗣	四	士	始
<	3C	28	次	滋	治	爾	璽	痔	磁	示	而	耳	自	蒔	辞	汐	鹿	式	識	鴫	竺	軸	宍	雫	七	叱	執	失	嫉	室	悉	湿	漆	疾	質	実	蔀	篠	偲	柴	芝	屡	蕊	縞	舎	写	射	捨	赦
=	3D	29	宗	就	州	修	愁	拾	洲	秀	秋	終	繍	習	臭	舟	蒐	衆	襲	讐	蹴	輯	週	酋	酬	集	醜	什	住	充	十	従	戎	柔	汁	渋	獣	縦	重	銃	叔	夙	宿	淑	祝	縮	粛	塾	熟
>	3E	30	勝	匠	升	召	哨	商	唱	嘗	奨	妾	娼	宵	将	小	少	尚	庄	床	廠	彰	承	抄	招	掌	捷	昇	昌	昭	晶	松	梢	樟	樵	沼	消	渉	湘	焼	焦	照	症	省	硝	礁	祥	称	章
?	3F	31	拭	植	殖	燭	織	職	色	触	食	蝕	辱	尻	伸	信	侵	唇	娠	寝	審	心	慎	振	新	晋	森	榛	浸	深	申	疹	真	神	秦	紳	臣	芯	薪	親	診	身	辛	進	針	震	人	仁	刃
@	40	32	澄	摺	寸	世	瀬	畝	是	凄	制	勢	姓	征	性	成	政	整	星	晴	棲	栖	正	清	牲	生	盛	精	聖	声	製	西	誠	誓	請	逝	醒	青	静	斉	税	脆	隻	席	惜	戚	斥	昔	析
A	41	33	繊	羨	腺	舛	船	薦	詮	賎	践	選	遷	銭	銑	閃	鮮	前	善	漸	然	全	禅	繕	膳	糎	噌	塑	岨	措	曾	曽	楚	狙	疏	疎	礎	祖	租	粗	素	組	蘇	訴	阻	遡	鼠	僧	創
B	42	34	臓	蔵	贈	造	促	側	則	即	息	捉	束	測	足	速	俗	属	賊	族	続	卒	袖	其	揃	存	孫	尊	損	村	遜	他	多	太	汰	詑	唾	堕	妥	惰	打	柁	舵	楕	陀	駄	騨	体	堆
C	43	35	叩	但	達	辰	奪	脱	巽	竪	辿	棚	谷	狸	鱈	樽	誰	丹	単	嘆	坦	担	探	旦	歎	淡	湛	炭	短	端	箪	綻	耽	胆	蛋	誕	鍛	団	壇	弾	断	暖	檀	段	男	談	値	知	地
D	44	36	帖	帳	庁	弔	張	彫	徴	懲	挑	暢	朝	潮	牒	町	眺	聴	脹	腸	蝶	調	諜	超	跳	銚	長	頂	鳥	勅	捗	直	朕	沈	珍	賃	鎮	陳	津	墜	椎	槌	追	鎚	痛	通	塚	栂	掴
E	45	37	邸	鄭	釘	鼎	泥	摘	擢	敵	滴	的	笛	適	鏑	溺	哲	徹	撤	轍	迭	鉄	典	填	天	展	店	添	纏	甜	貼	転	顛	点	伝	殿	澱	田	電	兎	吐	堵	塗	妬	屠	徒	斗	杜	渡
F	46	38	董	蕩	藤	討	謄	豆	踏	逃	透	鐙	陶	頭	騰	闘	働	動	同	堂	導	憧	撞	洞	瞳	童	胴	萄	道	銅	峠	鴇	匿	得	徳	涜	特	督	禿	篤	毒	独	読	栃	橡	凸	突	椴	届
G	47	39	如	尿	韮	任	妊	忍	認	濡	禰	祢	寧	葱	猫	熱	年	念	捻	撚	燃	粘	乃	廼	之	埜	嚢	悩	濃	納	能	脳	膿	農	覗	蚤	巴	把	播	覇	杷	波	派	琶	破	婆	罵	芭	馬
H	48	40	函	箱	硲	箸	肇	筈	櫨	幡	肌	畑	畠	八	鉢	溌	発	醗	髪	伐	罰	抜	筏	閥	鳩	噺	塙	蛤	隼	伴	判	半	反	叛	帆	搬	斑	板	氾	汎	版	犯	班	畔	繁	般	藩	販	範
I	49	41	鼻	柊	稗	匹	疋	髭	彦	膝	菱	肘	弼	必	畢	筆	逼	桧	姫	媛	紐	百	謬	俵	彪	標	氷	漂	瓢	票	表	評	豹	廟	描	病	秒	苗	錨	鋲	蒜	蛭	鰭	品	彬	斌	浜	瀕	貧
J	4A	42	福	腹	複	覆	淵	弗	払	沸	仏	物	鮒	分	吻	噴	墳	憤	扮	焚	奮	粉	糞	紛	雰	文	聞	丙	併	兵	塀	幣	平	弊	柄	並	蔽	閉	陛	米	頁	僻	壁	癖	碧	別	瞥	蔑	箆
K	4B	43	法	泡	烹	砲	縫	胞	芳	萌	蓬	蜂	褒	訪	豊	邦	鋒	飽	鳳	鵬	乏	亡	傍	剖	坊	妨	帽	忘	忙	房	暴	望	某	棒	冒	紡	肪	膨	謀	貌	貿	鉾	防	吠	頬	北	僕	卜	墨
L	4C	44	漫	蔓	味	未	魅	巳	箕	岬	密	蜜	湊	蓑	稔	脈	妙	粍	民	眠	務	夢	無	牟	矛	霧	鵡	椋	婿	娘	冥	名	命	明	盟	迷	銘	鳴	姪	牝	滅	免	棉	綿	緬	面	麺	摸	模
M	4D	45	諭	輸	唯	佑	優	勇	友	宥	幽	悠	憂	揖	有	柚	湧	涌	猶	猷	由	祐	裕	誘	遊	邑	郵	雄	融	夕	予	余	与	誉	輿	預	傭	幼	妖	容	庸	揚	揺	擁	曜	楊	様	洋	溶
N	4E	46	痢	裏	裡	里	離	陸	律	率	立	葎	掠	略	劉	流	溜	琉	留	硫	粒	隆	竜	龍	侶	慮	旅	虜	了	亮	僚	両	凌	寮	料	梁	涼	猟	療	瞭	稜	糧	良	諒	遼	量	陵	領	力
O	4F	47	蓮	連	錬	呂	魯	櫓	炉	賂	路	露	労	婁	廊	弄	朗	楼	榔	浪	漏	牢	狼	篭	老	聾	蝋	郎	六	麓	禄	肋	録	論	倭	和	話	歪	賄	脇	惑	枠	鷲	亙	亘	鰐	詫	藁	蕨

JIS Code Table (top right quarter)

P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~			
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E			
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94			
{	}	〈	〉	《	》	「	」	『	』	【	】	+	−	±	×	÷	=	≠	<	>	≦	≧	∞	∴	♂	♀	°	′	″	℃	¥	$	¢	£	%	#	&	*	@	§	☆	★	○	●	◎	◇	01	21	!
∃												∠	⊥	⌒	∂	∇	≡	≒	≪	≫	√	∽	∝	∵	∫	∬								Å	‰	♯	♭	♪	†	‡	¶					◯	02	22	"
P	Q	R	S	T	U	V	W	X	Y	Z							a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z					03	23	#
ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ	ゐ	ゑ	を	ん												04	24	$
バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ									05	25	%
π	ρ	σ	τ	υ	φ	χ	ψ	ω																																							06	26	&
	а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э	ю	я														07	27	’
																																															08	28	(
																																															09	29	)
																																															10	2A	*
																																															11	2B	+
																																															12	2C	,
																																															13	2D	-
																																															14	2E	.
																																															15	2F	/
夷	委	威	尉	惟	意	慰	易	椅	為	畏	異	移	維	緯	胃	萎	衣	謂	違	遺	医	井	亥	域	育	郁	磯	一	壱	溢	逸	稲	茨	芋	鰯	允	印	咽	員	因	姻	引	飲	淫	胤	蔭	16	30	0
頴	英	衛	詠	鋭	液	疫	益	駅	悦	謁	越	閲	榎	厭	円	園	堰	奄	宴	延	怨	掩	援	沿	演	炎	焔	煙	燕	猿	縁	艶	苑	薗	遠	鉛	鴛	塩	於	汚	甥	凹	央	奥	往	応	17	31	1
火	珂	禍	禾	稼	箇	花	苛	茄	荷	華	菓	蝦	課	嘩	貨	迦	過	霞	蚊	俄	峨	我	牙	画	臥	芽	蛾	賀	雅	餓	駕	介	会	解	回	塊	壊	廻	快	怪	悔	恢	懐	戒	拐	改	18	32	2
覚	角	赫	較	郭	閣	隔	革	学	岳	楽	額	顎	掛	笠	樫	橿	梶	鰍	潟	割	喝	恰	括	活	渇	滑	葛	褐	轄	且	鰹	叶	椛	樺	鞄	株	兜	竃	蒲	釜	鎌	噛	鴨	栢	茅	萱	19	33	3
莞	観	諌	貫	還	鑑	間	閑	関	陥	韓	館	舘	丸	含	岸	巌	玩	癌	眼	岩	翫	贋	雁	頑	顔	願	企	伎	危	喜	器	基	奇	嬉	寄	岐	希	幾	忌	揮	机	旗	既	期	棋	棄	20	34	4
黍	却	客	脚	虐	逆	丘	久	仇	休	及	吸	宮	弓	急	救	朽	求	汲	泣	灸	球	究	窮	笈	級	糾	給	旧	牛	去	居	巨	拒	拠	挙	渠	虚	許	距	鋸	漁	禦	魚	亨	享	京	21	35	5
勤	均	巾	錦	斤	欣	欽	琴	禁	禽	筋	緊	芹	菌	衿	襟	謹	近	金	吟	銀	九	倶	句	区	狗	玖	矩	苦	躯	駆	駈	駒	具	愚	虞	喰	空	偶	寓	遇	隅	串	櫛	釧	屑	屈	22	36	6
経	継	繋	罫	茎	荊	蛍	計	詣	警	軽	頚	鶏	芸	迎	鯨	劇	戟	撃	激	隙	桁	傑	欠	決	潔	穴	結	血	訣	月	件	倹	倦	健	兼	券	剣	喧	圏	堅	嫌	建	憲	懸	拳	捲	23	37	7
湖	狐	糊	袴	股	胡	菰	虎	誇	跨	鈷	雇	顧	鼓	五	互	伍	午	呉	吾	娯	後	御	悟	梧	檎	瑚	碁	語	誤	護	醐	乙	鯉	交	佼	侯	候	倖	光	公	功	効	勾	厚	口	向	24	38	8
腔	膏	航	荒	行	衡	講	貢	購	郊	酵	鉱	砿	鋼	閤	降	項	香	高	鴻	剛	劫	号	合	壕	拷	濠	豪	轟	麹	克	刻	告	国	穀	酷	鵠	黒	獄	漉	腰	甑	忽	惚	骨	狛	込	25	39	9
歳	済	災	采	犀	砕	砦	祭	斎	細	菜	裁	載	際	剤	在	材	罪	財	冴	坂	阪	堺	榊	肴	咲	崎	埼	碕	鷺	作	削	咋	搾	昨	朔	柵	窄	策	索	錯	桜	鮭	笹	匙	冊	刷	26	3A	:
姉	姿	子	屍	市	師	志	思	指	支	孜	斯	施	旨	枝	止	死	氏	獅	祉	私	糸	紙	紫	肢	脂	至	視	詞	詩	試	誌	諮	資	賜	雌	飼	歯	事	似	侍	児	字	寺	慈	持	時	27	3B	;
斜	煮	社	紗	者	謝	車	遮	蛇	邪	借	勺	尺	杓	灼	爵	酌	釈	錫	若	寂	弱	惹	主	取	守	手	朱	殊	狩	珠	種	腫	趣	酒	首	儒	受	呪	寿	授	樹	綬	需	囚	収	周	28	3C	<
出	術	述	俊	峻	春	瞬	竣	舜	駿	准	循	旬	楯	殉	淳	準	潤	盾	純	巡	遵	醇	順	処	初	所	暑	曙	渚	庶	緒	署	書	薯	藷	諸	助	叙	女	序	徐	恕	鋤	除	傷	償	29	3D	=
笑	粧	紹	肖	菖	蒋	蕉	衝	裳	訟	証	詔	詳	象	賞	醤	鉦	鍾	鐘	障	鞘	上	丈	丞	乗	冗	剰	城	場	壌	嬢	常	情	擾	条	杖	浄	状	畳	穣	蒸	譲	醸	錠	嘱	埴	飾	30	3E	>
塵	壬	尋	甚	尽	腎	訊	迅	陣	靭	笥	諏	須	酢	図	厨	逗	吹	垂	帥	推	水	炊	睡	粋	翠	衰	遂	酔	錐	錘	随	瑞	髄	崇	嵩	数	枢	趨	雛	据	杉	椙	菅	頗	雀	裾	31	3F	?
石	積	籍	績	脊	責	赤	跡	蹟	碩	切	拙	接	摂	折	設	窃	節	説	雪	絶	舌	蝉	仙	先	千	占	宣	専	尖	川	戦	扇	撰	栓	栴	泉	浅	洗	染	潜	煎	煽	旋	穿	箭	線	32	40	@
双	叢	倉	喪	壮	奏	爽	宋	層	匝	惣	想	捜	掃	挿	掻	操	早	曹	巣	槍	槽	漕	燥	争	痩	相	窓	糟	総	綜	聡	草	荘	葬	蒼	藻	装	走	送	遭	鎗	霜	騒	像	増	憎	33	41	A
対	耐	岱	帯	待	怠	態	戴	替	泰	滞	胎	腿	苔	袋	貸	退	逮	隊	黛	鯛	代	台	大	第	醍	題	鷹	滝	瀧	卓	啄	宅	托	択	拓	沢	濯	琢	託	鐸	濁	諾	茸	凧	蛸	只	34	42	B
弛	恥	智	池	痴	稚	置	致	蜘	遅	馳	築	畜	竹	筑	蓄	逐	秩	窒	茶	嫡	着	中	仲	宙	忠	抽	昼	柱	注	虫	衷	註	酎	鋳	駐	樗	瀦	猪	苧	著	貯	丁	兆	凋	喋	寵	35	43	C
槻	佃	漬	柘	辻	蔦	綴	鍔	椿	潰	坪	壷	嬬	紬	爪	吊	釣	鶴	亭	低	停	偵	剃	貞	呈	堤	定	帝	底	庭	廷	弟	悌	抵	挺	提	梯	汀	碇	禎	程	締	艇	訂	諦	蹄	逓	36	44	D
登	菟	賭	途	都	鍍	砥	砺	努	度	土	奴	怒	倒	党	冬	凍	刀	唐	塔	塘	套	宕	島	嶋	悼	投	搭	東	桃	梼	棟	盗	淘	湯	涛	灯	燈	当	痘	祷	等	答	筒	糖	統	到	37	45	E
鳶	苫	寅	酉	瀞	噸	屯	惇	敦	沌	豚	遁	頓	呑	曇	鈍	奈	那	内	乍	凪	薙	謎	灘	捺	鍋	楢	馴	縄	畷	南	楠	軟	難	汝	二	尼	弐	迩	匂	賑	肉	虹	廿	日	乳	入	38	46	F
俳	廃	拝	排	敗	杯	盃	牌	背	肺	輩	配	倍	培	媒	梅	楳	煤	狽	買	売	賠	陪	這	蝿	秤	矧	萩	伯	剥	博	拍	柏	泊	白	箔	粕	舶	薄	迫	曝	漠	爆	縛	莫	駁	麦	39	47	G
釆	煩	頒	飯	挽	晩	番	盤	磐	蕃	蛮	匪	卑	否	妃	庇	彼	悲	扉	批	披	斐	比	泌	疲	皮	碑	秘	緋	罷	肥	被	誹	費	避	非	飛	樋	簸	備	尾	微	枇	毘	琵	眉	美	40	48	H
賓	頻	敏	瓶	不	付	埠	夫	婦	富	冨	布	府	怖	扶	敷	斧	普	浮	父	符	腐	膚	芙	譜	負	賦	赴	阜	附	侮	撫	武	舞	葡	蕪	部	封	楓	風	葺	蕗	伏	副	復	幅	服	41	49	I
偏	変	片	篇	編	辺	返	遍	便	勉	娩	弁	鞭	保	舗	鋪	圃	捕	歩	甫	補	輔	穂	募	墓	慕	戊	暮	母	簿	菩	倣	俸	包	呆	報	奉	宝	峰	峯	崩	庖	抱	捧	放	方	朋	42	4A	J
撲	朴	牧	睦	穆	釦	勃	没	殆	堀	幌	奔	本	翻	凡	盆	摩	磨	魔	麻	埋	妹	昧	枚	毎	哩	槙	幕	膜	枕	鮪	柾	鱒	桝	亦	俣	又	抹	末	沫	迄	侭	繭	麿	万	慢	満	43	4B	K
茂	妄	孟	毛	猛	盲	網	耗	蒙	儲	木	黙	目	杢	勿	餅	尤	戻	籾	貰	問	悶	紋	門	匁	也	冶	夜	爺	耶	野	弥	矢	厄	役	約	薬	訳	躍	靖	柳	薮	鑓	愉	愈	油	癒	44	4C	L
熔	用	窯	羊	耀	葉	蓉	要	謡	踊	遥	陽	養	慾	抑	欲	沃	浴	翌	翼	淀	羅	螺	裸	来	莱	頼	雷	洛	絡	落	酪	乱	卵	嵐	欄	濫	藍	蘭	覧	利	吏	履	李	梨	理	璃	45	4D	M
緑	倫	厘	林	淋	燐	琳	臨	輪	隣	鱗	麟	瑠	塁	涙	累	類	令	伶	例	冷	励	嶺	怜	玲	礼	苓	鈴	隷	零	霊	麗	齢	暦	歴	列	劣	烈	裂	廉	恋	憐	漣	煉	簾	練	聯	46	4E	N
椀	湾	碗	腕																																												47	4F	O

JIS Code Table (bottom left quarter)

P	50	48	弌	丐	丕	个	丱	丶	丼	丿	乂	乖	乘	亂	亅	豫	亊	舒	弍	于	亞	亟	亠	亢	亰	亳	亶	从	仍	仄	仆	仂	仗	仞	仭	仟	价	伉	佚	估	佛	佝	佗	佇	佶	侈	侏	侘	佻
Q	51	49	僉	僊	傳	僂	僖	僞	僥	僭	僣	僮	價	僵	儉	儁	儂	儖	儕	儔	儚	儡	儺	儷	儼	儻	儿	兀	兒	兌	兔	兢	竸	兩	兪	兮	冀	冂	囘	册	冉	冏	冑	冓	冕	冖	冤	冦	冢
R	52	50	辧	劬	劭	劼	劵	勁	勍	勗	勞	勣	勦	飭	勠	勳	勵	勸	勹	匆	匈	甸	匍	匐	匏	匕	匚	匣	匯	匱	匳	匸	區	卆	卅	丗	卉	卍	凖	卞	卩	卮	夘	卻	卷	厂	厖	厠	厦
S	53	51	咫	哂	咤	咾	咼	哘	哥	哦	唏	唔	哽	哮	哭	哺	哢	唹	啀	啣	啌	售	啜	啅	啖	啗	唸	唳	啝	喙	喀	咯	喊	喟	啻	啾	喘	喞	單	啼	喃	喩	喇	喨	嗚	嗅	嗟	嗄	嗜
T	54	52	圀	國	圍	圓	團	圖	嗇	圜	圦	圷	圸	坎	圻	址	坏	坩	埀	垈	坡	坿	垉	垓	垠	垳	垤	垪	垰	埃	埆	埔	埒	埓	堊	埖	埣	堋	堙	堝	塲	堡	塢	塋	塰	毀	塒	堽	塹
U	55	53	奸	妁	妝	佞	侫	妣	妲	姆	姨	姜	妍	姙	姚	娥	娟	娑	娜	娉	娚	婀	婬	婉	娵	娶	婢	婪	媚	媼	媾	嫋	嫂	媽	嫣	嫗	嫦	嫩	嫖	嫺	嫻	嬌	嬋	嬖	嬲	嫐	嬪	嬶	嬾
V	56	54	屐	屏	孱	屬	屮	乢	屶	屹	岌	岑	岔	妛	岫	岻	岶	岼	岷	峅	岾	峇	峙	峩	峽	峺	峭	嶌	峪	崋	崕	崗	嵜	崟	崛	崑	崔	崢	崚	崙	崘	嵌	嵒	嵎	嵋	嵬	嵳	嵶	嶇
W	57	55	廖	廣	廝	廚	廛	廢	廡	廨	廩	廬	廱	廳	廰	廴	廸	廾	弃	弉	彝	彜	弋	弑	弖	弩	弭	弸	彁	彈	彌	彎	弯	彑	彖	彗	彙	彡	彭	彳	彷	徃	徂	彿	徊	很	徑	徇	從
X	58	56	悄	悛	悖	悗	悒	悧	悋	惡	悸	惠	惓	悴	忰	悽	惆	悵	惘	慍	愕	愆	惶	惷	愀	惴	惺	愃	愡	惻	惱	愍	愎	慇	愾	愨	愧	慊	愿	愼	愬	愴	愽	慂	慄	慳	慷	慘	慙
Y	59	57	戞	戡	截	戮	戰	戲	戳	扁	扎	扞	扣	扛	扠	扨	扼	抂	抉	找	抒	抓	抖	拔	抃	抔	拗	拑	抻	拏	拿	拆	擔	拈	拜	拌	拊	拂	拇	抛	拉	挌	拮	拱	挧	挂	挈	拯	拵
Z	5A	58	據	擒	擅	擇	撻	擘	擂	擱	擧	舉	擠	擡	抬	擣	擯	攬	擶	擴	擲	擺	攀	擽	攘	攜	攅	攤	攣	攫	攴	攵	攷	收	攸	畋	效	敖	敕	敍	敘	敞	敝	敲	數	斂	斃	變	斛
[	5B	59	暉	暸	曖	曚	曠	昿	曦	曩	曰	曵	曷	朏	朖	朞	朦	朧	霸	朮	朿	朶	杁	朸	朷	杆	杞	杠	杙	杣	杤	枉	杰	枩	杼	杪	枌	枋	枦	枡	枅	枷	柯	枴	柬	枳	柩	枸	柤
\	5C	60	梠	棧	棕	椶	椒	椄	棗	棣	椥	棹	棠	棯	椨	椪	椚	椣	椡	棆	楹	楷	楜	楸	楫	楔	楾	楮	椹	楴	椽	楙	椰	楡	楞	楝	榁	楪	榲	榮	槐	榿	槁	槓	榾	槎	寨	槊	槝
]	5D	61	檗	蘗	檻	櫃	櫂	檸	檳	檬	櫞	櫑	櫟	檪	櫚	櫪	櫻	欅	蘖	櫺	欒	欖	鬱	欟	欸	欷	盜	欹	飮	歇	歃	歉	歐	歙	歔	歛	歟	歡	歸	歹	歿	殀	殄	殃	殍	殘	殕	殞	殤
^	5E	62	沺	泛	泯	泙	泪	洟	衍	洶	洫	洽	洸	洙	洵	洳	洒	洌	浣	涓	浤	浚	浹	浙	涎	涕	濤	涅	淹	渕	渊	涵	淇	淦	涸	淆	淬	淞	淌	淨	淒	淅	淺	淙	淤	淕	淪	淮	渭
_	5F	63	漾	漓	滷	澆	潺	潸	澁	澀	潯	潛	濳	潭	澂	潼	潘	澎	澑	濂	潦	澳	澣	澡	澤	澹	濆	澪	濟	濕	濬	濔	濘	濱	濮	濛	瀉	瀋	濺	瀑	瀁	瀏	濾	瀛	瀚	潴	瀝	瀘	瀟
`	60	64	燹	燿	爍	爐	爛	爨	爭	爬	爰	爲	爻	爼	爿	牀	牆	牋	牘	牴	牾	犂	犁	犇	犒	犖	犢	犧	犹	犲	狃	狆	狄	狎	狒	狢	狠	狡	狹	狷	倏	猗	猊	猜	猖	猝	猴	猯	猩
a	61	65	瓠	瓣	瓧	瓩	瓮	瓲	瓰	瓱	瓸	瓷	甄	甃	甅	甌	甎	甍	甕	甓	甞	甦	甬	甼	畄	畍	畊	畉	畛	畆	畚	畩	畤	畧	畫	畭	畸	當	疆	疇	畴	疊	疉	疂	疔	疚	疝	疥	疣
b	62	66	癩	癶	癸	發	皀	皃	皈	皋	皎	皖	皓	皙	皚	皰	皴	皸	皹	皺	盂	盍	盖	盒	盞	盡	盥	盧	盪	蘯	盻	眈	眇	眄	眩	眤	眞	眥	眦	眛	眷	眸	睇	睚	睨	睫	睛	睥	睿
c	63	67	磧	磚	磽	磴	礇	礒	礑	礙	礬	礫	祀	祠	祗	祟	祚	祕	祓	祺	祿	禊	禝	禧	齋	禪	禮	禳	禹	禺	秉	秕	秧	秬	秡	秣	稈	稍	稘	稙	稠	稟	禀	稱	稻	稾	稷	穃	穗
d	64	68	筐	笄	筍	笋	筌	筅	筵	筥	筴	筧	筰	筱	筬	筮	箝	箘	箟	箍	箜	箚	箋	箒	箏	筝	箙	篋	篁	篌	篏	箴	篆	篝	篩	簑	簔	篦	篥	籠	簀	簇	簓	篳	篷	簗	簍	篶	簣
e	65	69	紂	紜	紕	紊	絅	絋	紮	紲	紿	紵	絆	絳	絖	絎	絲	絨	絮	絏	絣	經	綉	絛	綏	絽	綛	綺	綮	綣	綵	緇	綽	綫	總	綢	綯	緜	綸	綟	綰	緘	緝	緤	緞	緻	緲	緡	縅
f	66	70	罅	罌	罍	罎	罐	网	罕	罔	罘	罟	罠	罨	罩	罧	罸	羂	羆	羃	羈	羇	羌	羔	羞	羝	羚	羣	羯	羲	羹	羮	羶	羸	譱	翅	翆	翊	翕	翔	翡	翦	翩	翳	翹	飜	耆	耄	耋
g	67	71	隋	腆	脾	腓	腑	胼	腱	腮	腥	腦	腴	膃	膈	膊	膀	膂	膠	膕	膤	膣	腟	膓	膩	膰	膵	膾	膸	膽	臀	臂	膺	臉	臍	臑	臙	臘	臈	臚	臟	臠	臧	臺	臻	臾	舁	舂	舅
h	68	72	茵	茴	茖	茲	茱	荀	茹	荐	荅	茯	茫	茗	茘	莅	莚	莪	莟	莢	莖	茣	莎	莇	莊	荼	莵	荳	荵	莠	莉	莨	菴	萓	菫	菎	菽	萃	菘	萋	菁	菷	萇	菠	菲	萍	萢	萠	莽
i	69	73	蕁	蘂	蕋	蕕	薀	薤	薈	薑	薊	薨	蕭	薔	薛	藪	薇	薜	蕷	蕾	薐	藉	薺	藏	薹	藐	藕	藝	藥	藜	藹	蘊	蘓	蘋	藾	藺	蘆	蘢	蘚	蘰	蘿	虍	乕	虔	號	虧	虱	蚓	蚣
j	6A	74	蝓	蝣	蝪	蠅	螢	螟	螂	螯	蟋	螽	蟀	蟐	雖	螫	蟄	螳	蟇	蟆	螻	蟯	蟲	蟠	蠏	蠍	蟾	蟶	蟷	蠎	蟒	蠑	蠖	蠕	蠢	蠡	蠱	蠶	蠹	蠧	蠻	衄	衂	衒	衙	衞	衢	衫	袁
k	6B	75	襦	襤	襭	襪	襯	襴	襷	襾	覃	覈	覊	覓	覘	覡	覩	覦	覬	覯	覲	覺	覽	覿	觀	觚	觜	觝	觧	觴	觸	訃	訖	訐	訌	訛	訝	訥	訶	詁	詛	詒	詆	詈	詼	詭	詬	詢	誅
l	6C	76	譟	譬	譯	譴	譽	讀	讌	讎	讒	讓	讖	讙	讚	谺	豁	谿	豈	豌	豎	豐	豕	豢	豬	豸	豺	貂	貉	貅	貊	貍	貎	貔	豼	貘	戝	貭	貪	貽	貲	貳	貮	貶	賈	賁	賤	賣	賚
m	6D	77	蹇	蹉	蹌	蹐	蹈	蹙	蹤	蹠	踪	蹣	蹕	蹶	蹲	蹼	躁	躇	躅	躄	躋	躊	躓	躑	躔	躙	躪	躡	躬	躰	軆	躱	躾	軅	軈	軋	軛	軣	軼	軻	軫	軾	輊	輅	輕	輒	輙	輓	輜
n	6E	78	遏	遐	遑	遒	逎	遉	逾	遖	遘	遞	遨	遯	遶	隨	遲	邂	遽	邁	邀	邊	邉	邏	邨	邯	邱	邵	郢	郤	扈	郛	鄂	鄒	鄙	鄲	鄰	酊	酖	酘	酣	酥	酩	酳	酲	醋	醉	醂	醢
o	6F	79	錙	錢	錚	錣	錺	錵	錻	鍜	鍠	鍼	鍮	鍖	鎰	鎬	鎭	鎔	鎹	鏖	鏗	鏨	鏥	鏘	鏃	鏝	鏐	鏈	鏤	鐚	鐔	鐓	鐃	鐇	鐐	鐶	鐫	鐵	鐡	鐺	鑁	鑒	鑄	鑛	鑠	鑢	鑞	鑪	鈩
p	70	80	陜	陟	陦	陲	陬	隍	隘	隕	隗	險	隧	隱	隲	隰	隴	隶	隸	隹	雎	雋	雉	雍	襍	雜	霍	雕	雹	霄	霆	霈	霓	霎	霑	霏	霖	霙	霤	霪	霰	霹	霽	霾	靄	靆	靈	靂	靉
q	71	81	顱	顴	顳	颪	颯	颱	颶	飄	飃	飆	飩	飫	餃	餉	餒	餔	餘	餡	餝	餞	餤	餠	餬	餮	餽	餾	饂	饉	饅	饐	饋	饑	饒	饌	饕	馗	馘	馥	馭	馮	馼	駟	駛	駝	駘	駑	駭
r	72	82	髫	髮	髴	髱	髷	髻	鬆	鬘	鬚	鬟	鬢	鬣	鬥	鬧	鬨	鬩	鬪	鬮	鬯	鬲	魄	魃	魏	魍	魎	魑	魘	魴	鮓	鮃	鮑	鮖	鮗	鮟	鮠	鮨	鮴	鯀	鯊	鮹	鯆	鯏	鯑	鯒	鯣	鯢	鯤
s	73	83	鵝	鵞	鵤	鵑	鵐	鵙	鵲	鶉	鶇	鶫	鵯	鵺	鶚	鶤	鶩	鶲	鷄	鷁	鶻	鶸	鶺	鷆	鷏	鷂	鷙	鷓	鷸	鷦	鷭	鷯	鷽	鸚	鸛	鸞	鹵	鹹	鹽	麁	麈	麋	麌	麒	麕	麑	麝	麥	麩
t	74	84	堯	槇	遙	瑤																																											
u	75	85																																															
v	76	86																																															
w	77	87																																															
x	78	88																																															
y	79	89																																															
z	7A	90																																															
{	7B	91																																															
|	7C	92																																															
}	7D	93																																															
‾	7E	94																																															
			01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47
			21	22	23	24	25	26	27	28	29	2A	2B	2C	2D	2E	2F	30	31	32	33	34	35	36	37	38	39	3A	3B	3C	3D	3E	3F	40	41	42	43	44	45	46	47	48	49	4A	4B	4C	4D	4E	4F
			!	"	#	$	%	&	'	(	)	*	+	,	-	.	/	0	1	2	3	4	5	6	7	8	9	:	;	<	=	>	?	@	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O

JIS Code Table (bottom right quarter)

佩	佰	侑	佯	來	侖	儘	俔	俟	俎	俘	俛	俑	俚	俐	俤	俥	倚	倨	倔	倪	倥	倅	伜	俶	倡	倩	倬	俾	俯	們	倆	偃	假	會	偕	偐	偈	做	偖	偬	偸	傀	傚	傅	傴	傲	48	50	P
冩	冪	冫	决	冱	冲	冰	况	冽	凅	凉	凛	几	處	凩	凭	凰	凵	凾	刄	刋	刔	刎	刧	刪	刮	刳	刹	剏	剄	剋	剌	剞	剔	剪	剴	剩	剳	剿	剽	劍	劔	劒	剱	劈	劑	辨	49	51	Q
厥	厮	厰	厶	參	簒	雙	叟	曼	燮	叮	叨	叭	叺	吁	吽	呀	听	吭	吼	吮	吶	吩	吝	呎	咏	呵	咎	呟	呱	呷	呰	咒	呻	咀	呶	咄	咐	咆	哇	咢	咸	咥	咬	哄	哈	咨	50	52	R
嗤	嗔	嘔	嗷	嘖	嗾	嗽	嘛	嗹	噎	噐	營	嘴	嘶	嘲	嘸	噫	噤	嘯	噬	噪	嚆	嚀	嚊	嚠	嚔	嚏	嚥	嚮	嚶	嚴	囂	嚼	囁	囃	囀	囈	囎	囑	囓	囗	囮	囹	圀	囿	圄	圉	51	53	S
墅	墹	墟	墫	墺	壞	墻	墸	墮	壅	壓	壑	壗	壙	壘	壥	壜	壤	壟	壯	壺	壹	壻	壼	壽	夂	夊	夐	夛	梦	夥	夬	夭	夲	夸	夾	竒	奕	奐	奎	奚	奘	奢	奠	奧	奬	奩	52	54	T
孃	孅	孀	孑	孕	孚	孛	孥	孩	孰	孳	孵	學	斈	孺	宀	它	宦	宸	寃	寇	寉	寔	寐	寤	實	寢	寞	寥	寫	寰	寶	寳	尅	將	專	對	尓	尠	尢	尨	尸	尹	屁	屆	屎	屓	53	55	U
嶄	嶂	嶢	嶝	嶬	嶮	嶽	嶐	嶷	嶼	巉	巍	巓	巒	巖	巛	巫	已	巵	帋	帚	帙	帑	帛	帶	帷	幄	幃	幀	幎	幗	幔	幟	幢	幤	幇	幵	并	幺	麼	广	庠	廁	廂	廈	廐	廏	54	56	V
徙	徘	徠	徨	徭	徼	忖	忻	忤	忸	忱	忝	悳	忿	怡	恠	怙	怐	怩	怎	怱	怛	怕	怫	怦	怏	怺	恚	恁	恪	恷	恟	恊	恆	恍	恣	恃	恤	恂	恬	恫	恙	悁	悍	惧	悃	悚	55	57	W
慚	慫	慴	慯	慥	慱	慟	慝	慓	慵	憙	憖	憇	憬	憔	憚	憊	憑	憫	憮	懌	懊	應	懷	懈	懃	懆	憺	懋	罹	懍	懦	懣	懶	懺	懴	懿	懽	懼	懾	戀	戈	戉	戍	戌	戔	戛	56	58	X
捐	挾	捍	搜	捏	掖	掎	掀	掫	捶	掣	掏	掉	掟	掵	捫	捩	掾	揩	揀	揆	揣	揉	插	揶	揄	搖	搴	搆	搓	搦	搶	攝	搗	搨	搏	摧	摯	摶	摎	攪	撕	撓	撥	撩	撈	撼	57	59	Y
斟	斫	斷	旃	旆	旁	旄	旌	旒	旛	旙	无	旡	旱	杲	昊	昃	旻	杳	昵	昶	昴	昜	晏	晄	晉	晁	晞	晝	晤	晧	晨	晟	晢	晰	暃	暈	暎	暉	暄	暘	暝	曁	暹	曉	暾	暼	58	5A	Z
柞	柝	柢	柮	枹	柎	柆	柧	檜	栞	框	栩	桀	桍	栲	桎	梳	栫	桙	档	桷	桿	梟	梏	梭	梔	條	梛	梃	檮	梹	桴	梵	梠	梺	椏	梍	桾	椁	棊	椈	棘	椢	椦	棡	椌	棍	59	5B	[
榻	槃	榧	樮	榑	榠	榜	榕	榴	槞	槨	樂	樛	槿	權	槹	槲	槧	樅	榱	樞	槭	樔	槫	樊	樒	櫁	樣	樓	橄	樌	橲	樶	橸	橇	橢	橙	橦	橈	樸	樢	檐	檍	檠	檄	檢	檣	60	5C	\
殪	殫	殯	殲	殱	殳	殷	殼	毆	毋	毓	毟	毬	毫	毳	毯	麾	氈	氓	气	氛	氤	氣	汞	汕	汢	汪	沂	沍	沚	沁	沛	汾	汨	汳	沒	沐	泄	泱	泓	沽	泗	泅	泝	沮	沱	沾	61	5D	]
湮	渮	渙	湲	湟	渾	渣	湫	渫	湶	湍	渟	湃	渺	湎	渤	滿	渝	游	溂	溪	溘	滉	溷	滓	溽	溯	滄	溲	滔	滕	溏	溥	滂	溟	潁	漑	灌	滬	滸	滾	漿	滲	漱	滯	漲	滌	62	5E	^
瀰	瀾	瀲	灑	灣	炙	炒	炯	烱	炬	炸	炳	炮	烟	烋	烝	烙	焉	烽	焜	焙	煥	煕	熈	煦	煢	煌	煖	煬	熏	燻	熄	熕	熨	熬	燗	熹	熾	燒	燉	燔	燎	燠	燬	燧	燵	燼	63	5F	_
猥	猾	獎	獏	默	獗	獪	獨	獰	獸	獵	獻	獺	珈	玳	珎	玻	珀	珥	珮	珞	璢	琅	瑯	琥	珸	琲	琺	瑕	琿	瑟	瑙	瑁	瑜	瑩	瑰	瑣	瑪	瑶	瑾	璋	璞	璧	瓊	瓏	瓔	珱	64	60	‘
痂	疳	痃	疵	疽	疸	疼	疱	痍	痊	痒	痙	痣	痞	痾	痿	痼	瘁	痰	痺	痲	痳	瘋	瘍	瘉	瘟	瘧	瘠	瘡	瘢	瘤	瘴	瘰	瘻	癇	癈	癆	癜	癘	癡	癢	癨	癩	癪	癧	癬	癰	65	61	a
睾	睹	瞎	瞋	瞑	瞠	瞞	瞰	瞶	瞹	瞿	瞼	瞽	瞻	矇	矍	矗	矚	矜	矣	矮	矼	砌	砒	礦	砠	礪	硅	碎	硴	碆	硼	碚	碌	碣	碵	碪	碯	磑	磆	磋	磔	碾	碼	磅	磊	磬	66	62	b
穉	穡	穢	穩	龝	穰	穹	穽	窈	窗	窕	窘	窖	窩	竈	窰	窶	竅	竄	窿	邃	竇	竊	竍	竏	竕	竓	站	竚	竝	竡	竢	竦	竭	竰	笂	笏	笊	笆	笳	笘	笙	笞	笵	笨	笶	67	63	c
簣	簧	簪	簟	簷	簫	簽	籌	籃	籔	籏	籀	籐	籘	籟	籤	籖	籥	籬	籵	粃	粐	粤	粭	粢	粫	粡	粨	粳	粲	粱	粮	粹	粽	糀	糅	糂	糘	糒	糜	糢	鬻	糯	糲	糴	糶	糺	紆	68	64	d
縊	縣	縡	縒	縱	縟	縉	縋	縢	繆	繦	縻	縵	縹	繃	縷	縲	縺	繧	繝	繖	繞	繙	繚	繹	繪	繩	繼	繻	纃	緕	繽	辮	繿	纈	纉	續	纒	纐	纓	纔	纖	纎	纛	纜	缸	缺	69	65	e
耒	耘	耙	耜	耡	耨	耿	耻	聊	聆	聒	聘	聚	聟	聢	聨	聳	聲	聰	聶	聹	聽	聿	肄	肆	肅	肛	肓	肚	肭	冐	肬	胛	胥	胙	胝	胄	胚	胖	脉	胯	胱	脛	脩	脣	脯	腋	70	66	f
與	舊	舍	舐	舖	舩	舫	舸	舳	艀	艙	艘	艝	艚	艟	艤	艢	艨	艪	艫	舮	艱	艷	艸	艾	芍	芒	芫	芟	芻	芬	苡	苣	苟	苒	苴	苳	苺	莓	范	苻	苹	苞	茆	苜	茉	苙	71	67	g
萸	蔆	菻	葭	萪	萼	蕚	蒄	葷	葫	蒭	葮	蒂	葩	葆	萬	葯	葹	萵	蓊	葢	蒹	蒿	蒟	蓙	蓍	蒻	蓚	蓐	蓁	蓆	蓖	蒡	蔡	蓿	蓴	蔗	蔘	蔬	蔟	蔕	蔔	蓼	蕀	蕣	蕘	蕈	72	68	h
蚩	蚪	蚋	蚌	蚶	蚯	蛄	蛆	蚰	蛉	蠣	蚫	蛔	蛞	蛩	蛬	蛟	蛛	蛯	蜒	蜆	蜈	蜀	蜃	蛻	蜑	蜉	蜍	蛹	蜊	蜴	蜿	蜷	蜻	蜥	蜩	蜚	蝠	蝟	蝸	蝌	蝎	蝴	蝗	蝨	蝮	蝙	73	69	i
衾	袞	衵	衽	袵	衲	袂	袗	袒	袮	袙	袢	袍	袤	袰	袿	袱	裃	裄	裔	裘	裙	裝	裹	褂	裼	裴	裨	裲	褄	褌	褊	褓	襃	褞	褥	褪	褫	襁	襄	褻	褶	褸	襌	褝	襠	襞	74	6A	j
誂	誄	誨	誡	誑	誥	誦	誚	誣	諄	諍	諂	諚	諫	諳	諧	諤	諱	謔	諠	諢	諷	諞	諛	謌	謇	謚	諡	謖	謐	謗	謠	謳	鞫	謦	謫	謾	謨	譁	譌	譏	譎	證	譖	譛	譚	譫	75	6B	k
賽	賺	賻	贄	贅	贊	贇	贏	贍	贐	齎	贓	賍	贔	贖	赧	赭	赱	赳	趁	趙	跂	趾	趺	跏	跚	跖	跌	跛	跋	跪	跫	跟	跣	跼	踈	踉	跿	踝	踞	踐	踟	蹂	踵	踰	踴	蹊	76	6C	l
輟	輛	輌	輦	輳	輻	輹	轅	轂	輾	轌	轉	轆	轎	轗	轜	轢	轣	轤	辜	辟	辣	辭	辯	辷	迚	迥	迢	迪	迯	邇	迴	逅	迹	迺	逑	逕	逡	逍	逞	逖	逋	逧	逶	逵	逹	迸	77	6D	m
醫	醯	醪	醵	醴	醺	釀	釁	釉	釋	釐	釖	釟	釡	釛	釼	釵	釶	鈞	釿	鈔	鈬	鈕	鈑	鉞	鉗	鉅	鉉	鉤	鉈	銕	鈿	鉋	鉐	銜	銖	銓	銛	鉚	鋏	銹	銷	鋩	錏	鋺	鍄	錮	78	6E	n
鑰	鑵	鑷	鑽	鑚	鑼	鑾	钁	鑿	閂	閇	閊	閔	閖	閘	閙	閠	閨	閧	閭	閼	閻	閹	閾	闊	濶	闃	闍	闌	闕	闔	闖	關	闡	闥	闢	阡	阨	阮	阯	陂	陌	陏	陋	陷	陜	陞	79	6F	o
靜	靠	靤	靦	靨	勒	靫	靱	靹	鞅	靼	鞁	靺	鞆	鞋	鞏	鞐	鞜	鞨	鞦	鞣	鞳	鞴	韃	韆	韈	韋	韜	韭	齏	韲	竟	韶	韵	頏	頌	頸	頤	頡	頷	頽	顆	顏	顋	顫	顯	顰	80	70	p
駮	駱	駲	駻	駸	騁	騏	騅	駢	騙	騫	騷	驅	驂	驀	驃	騾	驕	驍	驛	驗	驟	驢	驥	驤	驩	驫	驪	骭	骰	骼	髀	髏	髑	髓	體	髞	髟	髢	髣	髦	髯	髫	髮	髴	髱	髷	81	71	q
鯰	鰕	鰔	鰉	鰓	鰌	鰆	鰈	鰒	鰊	鰄	鰮	鰛	鰥	鰤	鰡	鰰	鱇	鰲	鱆	鰾	鱚	鱠	鱧	鱶	鱸	鳧	鳬	鳰	鴉	鴈	鳫	鴃	鴆	鴪	鴦	鶯	鴣	鴟	鵄	鴕	鴒	鵁	鴿	鴾	鵆	鵈	82	72	r
麸	麪	麭	靡	黌	黎	黏	黐	黔	黜	點	黝	黠	黥	黨	黯	黴	黶	黷	黹	黻	黼	黽	鼇	鼈	皷	鼕	鼡	鼬	鼾	齊	齒	齔	齣	齟	齠	齡	齦	齧	齬	齪	齷	齲	齶	龕	龜	龠	83	73	s
																																															84	74	t
																																															85	75	u
																																															86	76	v
																																															87	77	w
																																															88	78	x
																																															89	79	y
																																															90	7A	z
																																															91	7B	{
																																															92	7C	|
																																															93	7D	}
																																															94	7E	~
48	49	50	51	52	53	54	55	56	57	58	59	60	61	62	63	64	65	66	67	68	69	70	71	72	73	74	75	76	77	78	79	80	81	82	83	84	85	86	87	88	89	90	91	92	93	94			
50	51	52	53	54	55	56	57	58	59	5A	5B	5C	5D	5E	5F	60	61	62	63	64	65	66	67	68	69	6A	6B	6C	6D	6E	6F	70	71	72	73	74	75	76	77	78	79	7A	7B	7C	7D	7E			
P	Q	R	S	T	U	V	W	X	Y	Z	[	\	]	^	_	‘	a	b	c	d	e	f	g	h	i	j	k	l	m	n	o	p	q	r	s	t	u	v	w	x	y	z	{	|	}	~			

Appendix 3 Sample Font Table for several subfonts

Test of dnpjhira38 on May 16, 1987 at 1254

	′0	′1	′2	′3	′4	′5	′6	′7	
′00x		ぁ	あ	ぃ	い	ぅ	う	ぇ	″0x
′01x	え	ぉ	お	か	が	き	ぎ	く	
′02x	ぐ	け	げ	こ	ご	さ	ざ	し	″1x
′03x	じ	す	ず	せ	ぜ	そ	ぞ	た	
′04x	だ	ち	ぢ	っ	つ	づ	て	で	″2x
′05x	と	ど	な	に	ぬ	ね	の	は	
′06x	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	″3x
′07x	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み	
′10x	む	め	も	ゃ	や	ゅ	ゆ	ょ	″4x
′11x	よ	ら	り	る	れ	ろ	ゎ	わ	
′12x	ゐ	ゑ	を	ん					″5x
′13x									
	″8	″9	″A	″B	″C	″D	″E	″F	

Test of dnpjkata38 on May 16, 1987 at 1254

	′0	′1	′2	′3	′4	′5	′6	′7	
′00x		ァ	ア	ィ	イ	ゥ	ウ	ェ	″0x
′01x	エ	ォ	オ	カ	ガ	キ	ギ	ク	
′02x	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	″1x
′03x	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ	
′04x	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	″2x
′05x	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ	
′06x	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	″3x
′07x	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ	
′10x	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	″4x
′11x	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ	
′12x	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ		″5x
′13x									
	″8	″9	″A	″B	″C	″D	″E	″F	

Test of dnpjka38 on May 16, 1987 at 1254

	′0	′1	′2	′3	′4	′5	′6	′7	
′00x	亜	唖	娃	阿	哀	愛	挨	姶	″0x
′01x	逢	葵	茜	穐	悪	握	渥	旭	
′02x	葦	芦	鯵	梓	圧	斡	扱	宛	″1x
′03x	姐	虻	飴	絢	綾	鮎	或	粟	
′04x	袷	安	庵	按	暗	案	闇	鞍	″2x
′05x	杏	以	伊	位	依	偉	囲	夷	
′06x	委	威	尉	惟	意	慰	易	椅	″3x
′07x	為	畏	異	移	維	緯	胃	萎	
′10x	衣	謂	違	遺	医	井	亥	域	″4x
′11x	育	郁	磯	一	壱	溢	逸	稲	
′12x	茨	芋	鰯	允	印	咽	員	因	″5x
′13x	姻	引	飲	淫	胤	蔭	院	陰	
′14x	隠	韻	吋	右	宇	烏	羽	迂	″6x
′15x	雨	卯	鵜	窺	丑	碓	臼	渦	
′16x	嘘	唄	欝	蔚	鰻	姥	厩	浦	″7x
′17x	瓜	閏	噂	云	運	雲	荏	餌	
′20x	叡	営	嬰	影	映	曳	栄	永	″8x
′21x	泳	洩	瑛	盈	穎	頴	英	衛	
′22x	詠	鋭	液	疫	益	駅	悦	謁	″9x
′23x	越	閲	榎	厭	円	園	堰	奄	
′24x	宴	延	怨	掩	援	沿	演	炎	″Ax
′25x	焔	煙	燕	猿	縁	艶	苑	薗	
′26x	遠	鉛	鴛	塩	於	汚	甥	凹	″Bx
′27x	央	奥	往	応	押	旺	横	欧	
′30x	殴	王	翁	襖	鴬	鴎	黄	岡	″Cx
′31x	沖	荻	億	屋	憶	臆	桶	牡	
′32x	乙	俺	卸	恩	温	穏	音	下	″Dx
′33x	化	仮	何	伽	価	佳	加	可	
′34x	嘉	夏	嫁	家	寡	科	暇	果	″Ex
′35x	架	歌	河	火	珂	禍	禾	稼	
′36x	箇	花	苛	茄	荷	華	菓	蝦	″Fx
′37x	課	嘩	貨	迦	過	霞	蚊	俄	
	″8	″9	″A	″B	″C	″D	″E	″F	

Multiple Changefiles: The Adventure Continues

E. W. (Wayne) Sewell
Software Engineering Specialist

The subject of multiple changefiles for the same WEB program has appeared several times in recent issues of TUGboat (Appelt and Korn, Vol 7, #1, and Guntermann and Rülling, Vol 7, #3). Even though the general TUG readership may be tiring of this subject, it is still a valid concern for anyone actively involved in WEB programming. This article describes yet another approach to the subject, embodied in a program called WEBMERGE.

A perfect example of the use of a program such as WEBMERGE is the Modula-2 WEB system described elsewhere in this issue (page 118). MWEB was implemented as a pair of changefiles applied to TANGLE and WEAVE, containing the modifications to allow WEB to work with the language Modula-2. However, virtually every implementation of a WEB program written for portability requires a changefile to tailor the program to the target system. These two sets of changes are independent and (hopefully) mutually exclusive, since the MWEB changes have to do with the program logic and the implementation changes deal primarily with the interface to the operating environment, but both must be applied to the same WEB source file. TANGLE and WEAVE, the WEB processors, expect only one changefile containing all of the changes to be incorporated. WEBMERGE can combine the two sets of changefiles into single files acceptable to the two programs.

Another valid use of WEBMERGE is in changes to be permanently applied to the main WEB file of a program, such as the updates to METAFONTware made by Tom Rokicki in the October 1986 TUGboat. These changefiles were not intended to be used as input to TANGLE and WEAVE, but were printed as if they were changefiles to guide the installer in making the changes directly to the WEB files with a text editor. WEBMERGE could have been used to process them like any other changefile to create a new WEB file.

The implementation of WEBMERGE is conceptually closer to the stand-alone TIE program described by Guntermann and Rülling than to the modifications Appelt and Korn made directly to TANGLE and WEAVE. Virtually all of Guntermann and Rülling's article applies equally well to WEBMERGE. Both programs apply multiple changefiles to a WEB file and generate either a new WEB or a composite changefile containing the combined changes.

The basic difference between TIE and WEBMERGE is in how the multiple changefiles relate to each other. The operation of the two programs should be pretty much the same when there is no change conflict (the case where more than one of the changefiles tries to modify the same lines of code), but the programs operate very differently when conflicts occur. In the sequential approach taken by Guntermann and Rülling, "the addition of changefile f_{i+1} behaves as if the changefiles f_1 to f_i had been merged into the WEB program before". The problem I see with this approach (assuming I am understanding it correctly) is that it requires the changefiles to be aware of the existence of each other. In other words, if changefiles f_1 and f_2 modify the same parts of a program, file f_2 must be written to modify f_1 rather than the WEB file itself. This precludes using f_2 without f_1. If the changes made by the two changefiles are truly independent, then it should be possible to handle them independently as well. It might be desirable to apply f_2 with a different f_1 or by itself. To reuse the MWEB example from above, changefiles WEAVE.VAX and MWEAVE.CH exist, both based on WEAVE.WEB. Applying MWEAVE.CH to WEAVE.WEB produces the generic version of MWEAVE (Modula-2 WEAVE). Applying WEAVE.VAX to WEAVE.WEB results in the VAX-specific version of regular WEAVE. Merging WEAVE.VAX and MWEAVE.CH together results in MWEAVE.VAX, which can then be used to create the VAX version of MWEAVE. Either of these changefiles can be used alone or with the other with no modifications. Also, MWEAVE.CH can be merged with a completely different implementation changefile to produce MWEAVE for another environment, without changing either file.

In contrast to TIE, WEBMERGE applies all of the changefiles to the original WEB file in parallel. If a conflict occurs, one of the changefiles is selected to apply that change, and the others are flushed (in this case, "flushing" a changefile means discarding the current change section for that file and moving ahead to the next one). A warning message is sent to the screen identifying which two files had a conflict, which file was flushed, and the source line on which the conflict occurred. Both the line number and the contents of the line are displayed so it is easy to determine exactly where the conflict occurred. Which file is used and which ones are flushed depends on how the conflict occurs. Two rules apply: a changefile with a matching operation already in progress has precedence over any others which match later lines; if no change is currently in progress and more than one file matches on the same line of the WEB file, the higher priority changefile is used. Priority refers to position within the list of changefiles (f_1 would have a higher priority than f_2).

Conflicts when merging changefiles are inevitable. While significant conflicts are not very likely, since the changes being merged are normally for different purposes and modify different portions of the code, conflicts of a trivial nature occur often. For instance, many WEB programs follow the example of Stanford and output a "banner line" to the terminal to identify the program and its version level, as in:

```
@d banner=='This is WEAVE,
   Version X.X'
```

Nearly all changefiles modify this line to reflect what change they are making to the program, such as:

```
@d banner=='This is WEAVE
   with hyperspace option, ...'
@d banner=='This is MWEAVE,
   Modula-2 WEAVE, ...'
```

for modifications to the logic of the program itself or

```
@d banner=='This is WEAVE,
   VAX/VMS Version ...'
@d banner=='This is WEAVE,
   Microsoft Pascal Version ...'
```

for the various implementation changefiles. However, when multiple changefiles are being merged, the banner line of none of them is correct, since the version of the program actually executing is a combination of the two:

```
@d banner=='This is MWEAVE,
   VAX/VMS Version ...'
```

The \title command in the "limbo" portion of a WEB program falls in the same category as the banner line, since it is also a target common to many changefiles.

The solution to this problem is to create a *third* changefile containing nothing but conflict resolutions. Its change sections would consist only of the composite banner line and title. It should be placed first in the list, so that its changes will override all of the others. Since the conflicts it addresses are expected, the warning messages can be ignored. (It goes without saying that any *unexpected* conflicts which surface must be analyzed to insure that they don't change the logic of the program to an uncompilable or unexecutable state.)

If the sequential approach of TIE is truly needed, the case where one changefile needs to be fully applied before the second one is applied to the result of the first, this can be accomplished serially by using WEBMERGE to create an intermediate WEB file and then applying the second changefile to it. Of course, this does require additional steps, but that's what batch files and command procedures are for.

Hopefully, WEBMERGE should be available from Stanford on the regular distribution tape by the time this reaches print. The WEB files and the VAX implementation files should be available from Stanford and additionally from Kellerman and Smith. For the people who have absolutely no way of reading a magnetic tape, the IBM PC version is available from me on PC floppies for a handling fee. Additionally, the original TANGLE and WEAVE, the MWEB system described elsewhere in this issue, and several of the TeX and METAFONT utility programs (sometimes referred to as TeXware and METAFONTware) are also available on floppy. All of these have change files targeted for Microsoft Pascal running under MS-DOS on the IBM PC, which is my development system. As far as other target computers are concerned, WEBMERGE was cannibalized from TANGLE, so it should be possible to adapt the current implementation-specific changefile for TANGLE without too much difficulty. If you have TANGLE running, you should have no trouble with WEBMERGE.

How to MANGLE Your Software: The WEB System for Modula-2

E. W. (Wayne) Sewell
Software Engineering Specialist

Standard Pascal is an incomplete language from a real-world production software point of view. This is not surprising, since the language was originally designed by Niklaus Wirth as a tool for teaching structured programming, and was never intended for development of production code. The only reason for the widespread use of Pascal is that the various implementors extended the language tremendously when they developed their compilers. VAX Pascal is a good example of a full-featured production compiler. Its many extensions to Pascal allow sophisticated systems to be developed with it. Virtually *every* implementation of Pascal has to extend it in some way, since standard Pascal (as described in Jensen & Wirth) is absolutely unusable, and ISO Pascal is not much better. While the extensions make Pascal a viable language, portability suffers because each of the implementors extended the language a different way, resulting in a Babel of dialects that is surpassed only by the BASIC language. Porting a program from one Pascal to another is a major effort, even on the same machine. Typical of the problems encountered is the **case** statement. The action to be performed if none of the cases match is not defined in standard Pascal. Since this is a major hole in the language, most implementations try to fill it. Some provide an **else** or **otherwise** clause, others use labels (such as *others:* or *otherwise:*). Whatever mechanism a compiler uses, it is different from what every other compiler uses.

The WEB system tries to counteract the portability problem by using macros for constructs that should have been addressed in the language and then redefining the macros in the implementation-specific change files to generate the correct code, allowing the generic WEB file to remain constant for all implementations. While this makes it possible to write portable Pascal programs, it would still be much less work if the language itself were more standardized.

While WEB does a tremendous job of overcoming the deficiencies of Pascal, there are limits to what can be accomplished. For instance, Pascal does not support separate compilation. A Pascal program is a monolithic block which must be compiled as a unit. *Include files*, which allow a program to be broken up into more than one source file, do not change this fact because the program is still logically one large block and must be compiled as such. Variables not local to a procedure are global to the entire program and are therefore available for accidental modification. Unrelated parts of the program can interact in unexpected ways, especially if the same variable names are used in more than one place. For example, forgetting to declare a variable which should be defined local to a procedure will be detected immediately by the compiler *unless* a variable of a compatible type with the same name is declared globally. The result is that the wrong variable, one unrelated to the procedure, will be modified. Errors of this nature can be very difficult to find. The WEB system can help detect this type of error (if the programmer happens to notice the inconsistencies in the cross-reference listing), but will not prevent it from happening.

The language Modula-2 was designed by Wirth to be the successor to Pascal. Unlike the original Pascal, it was designed to be used for developing real software. Most of the problems with Pascal are corrected by Modula-2, including the **case** problem mentioned above. The syntax is more straightforward, with less likelihood of ambiguities. The most important contribution of Modula-2 is that embodied in the name — the module concept. Modula-2 makes it possible to break up a large programming project into smaller independent pieces, called *modules*, each logically isolated from the others via the software engineering principle of *information hiding*.

The Modula-2 language is much more standardized than Pascal. Since the language is so much more powerful, there is less need to extend it. Input and output, the bane of portability, are completely removed from the language definition itself and are instead banished to library procedures that are more-or-less standardized.

While Modula-2 fixes most of the problems of Pascal and nearly all of the differences between Modula-2 and Pascal are improvements, a couple of the features of the language are steps backward, in my opinion. Case-sensitivity is one of the non-enhancements. In a Modula-2 program, *junk*, *Junk*, and *JUNK* would be considered three different variables. The reason for this change from Pascal, if any, is not obvious. I have never heard a *reasonable* explanation for it. Equally annoying, all of the Modula-2 reserved words are required to be in uppercase. This one almost makes sense, since having the reserved words stand out in this way would make a regular ASCII listing more readable. However, I don't feel that this slight benefit is worth the extra effort involved in writing a program. Using a powerful editor with macro and/or template capability which can fill in the reserved words on behalf of the programmer would make this less painful, but not necessarily enjoyable. I don't wish to give the impression that I am down on Modula-2 because of these issues. It is still my language of choice because the tremendous advantages it provides greatly outweigh the irritations.

MWEB is a version of the WEB system which has been customized for the language Modula-2. Many of the deficiencies of Pascal that are repaired by the WEB system are unnecessary in MWEB, since Modula-2 fixes most of them in the language definition itself. Some examples are the **else** clause on a **case** statement, the standard procedure to increment a variable (INC), and the **loop**, **exit**, and **return** instructions. To counteract the new problems introduced by the language, I designed MWEB to fix Modula-2 in the same way that Modula-2 and standard WEB fix Pascal. The effort expended by

MWEB in this effort is small compared to the lengths necessary to bring Pascal to a usable state. The result of the merger of Modula-2 and MWEB is a programming system that has the advantages of both and few of the disadvantages.

The transformation from WEB to MWEB was comparatively easy — Pascal and Modula-2 are so much alike to begin with, at least syntactically. In fact, Modula-2 is actually less complicated than Pascal and has a cleaner syntax with fewer ambiguities.

MANGLE and MWEAVE were created by modifying their regular WEB counterparts with a standard change file. I wanted to minimize the modifications to the code, limiting them to those absolutely necessary to process Modula-2.

Very few modifications were required to transform TANGLE into MANGLE. Many more changes had to be made to WEAVE to support Modula-2, since WEAVE has to know enough about the language to format it properly. Some changes could have been made with the built-in mechanisms of WEB, such as the formatting command

format *module* ≡ *program*

which creates a new reserved word **module** and causes it to be formatted as if it were **program**. The problem of this approach is that it has to be duplicated in every source file, putting the burden of implementing MWEB on the user rather than on the developer (myself). I decided to add the Modula-2 reserved words into the internal tables. Several new reserved words were added (**return**, **exit**, **by**, **import**, etc.) and others not needed for Modula-2 were dropped (**goto**, **label**, **downto**, **file**, and others).

The following issues surfaced during the implementation of MWEB:

- *Identifier length.* The size of an identifier had to be increased. The TANGLE limit was insufficient, since some of the standard Modula-2 library modules had identifiers far longer, and the truncated identifiers would not match. Unlike Pascal, Modula-2 does not specify a maximum identifier length; all characters in an identifier are considered significant. However, since it is difficult to use ∞ as a constant in a computer program, I just picked a number out of my hat — 31 characters maximum length, 20 for unambiguous length. It can be changed if needed. The length of reserved words also had to be increased so that words such as **definition** and **implementation** could be accommodated.
- *Comments.* All code related to comments had to be changed. While Pascal can have comments delimited by either (* *) or { }, Modula-2 uses only the former, since the braces are used elsewhere in the language definition (as set delimiters). Fortunately, this is a common and well-documented modification to TANGLE, since some of the more primitive Pascal systems have the same restriction. On the other hand, Modula-2 allows nested comments, so the comment-handling code in MANGLE could be simplified (the comment delimiters for the inner nest levels no longer have to be converted to [ ] for the program to compile). The metacomment delimiters are still @{ and @}, although they are converted to (* *) when output.
- *Case sensitivity.* The automatic forcing of everything to uppercase by MANGLE was a potential problem, since Modula-2 is case-sensitive. This mechanism could not be disabled, because the Modula-2 reserved words *do* have to be uppercase and MANGLE cannot differentiate reserved words from any other identifiers. I considered giving MANGLE a reserved word table like that of MWEAVE, but that was a more radical change to the code of TANGLE than I had planned. I finally decided this was a non-problem, since all occurrences of an identifier, definition and references alike, are forced to uppercase on an equal basis. If definition modules, implementation modules, and client modules are all MANGLED, all instances of the identifier will still match. This automatic forcing to uppercase removes the requirement in Modula-2 of reserved words being in uppercase in the source. As described above, the uppercase words are for readability, but the bold font used by MWEB is much more readable. Leaving MANGLE's uppercase mechanism intact disables the ability of Modula-2 to have multiple identifiers in a program differing only in case, (*junk*, *Junk*, and *JUNK*), but I consider this a poor practice anyway. (I will stop just short of saying that anyone who does it deserves whatever happens to them.) The only real problem with the uppercase characters occurs with imported modules which were not generated with the MWEB system (such as the library modules supplied with the compiler). For identifiers such as these, which *must* contain lower or mixed case, the WEB command to "pass through" Pascal

code without modification (@=*verbatim text*@>) must be used. For example:

```
from @= InOut @> import \\ @= WriteString @>;
```

Some predefined identifiers are all uppercase to begin with, such as the primary library module SYSTEM or the increment instruction INC. These can be left alone.

- *Vertical bar character.* The vertical bar character (|) had to be specially handled, since it is used by both Modula-2 and WEB for different purposes. WEB uses it to delimit Pascal code embedded within TEX code, such as

  ```
  The value of |good_stuff|
  should be output only if
  |buffer_index<=47|,
  otherwise ...
  ```

  while Modula-2 uses it to mark the end of the statement sequence following a case label (except for the last one), as in

  ```
  case junk of
  1: r := 10;
     m := 60   |
  2: k := 7    |
  3: m := 6
  end;
  ```

  A true conflict between the two usages is unlikely, because Pascal code within TEX code usually consists of short expressions or simple variable names rather than compound statements such as **case**. MWEAVE has been modified to identify the usage of the vertical bar by context. It will use the Modula-2 version in the code part of a section and the WEB version within TEX code (including module names and comments in the code section). If for some reason a **case** statement is needed within TEX code, two adjacent bar characters (||) are used to represent the Modula-2 case separator and are compressed by MWEAVE into an internal character which is output as the regular vertical bar character.

- *Underline character.* The usage of the underline character in identifiers, absent from the Modula-2 language definition as it is from standard Pascal, is provided by MWEB. I agree with Donald Knuth that *identifiers_several_words_long* are much more readable than *IdentifiersSeveralWordsLong*, which is Wirth's approach. MANGLE removes the underlines before passing the program to the compiler, like TANGLE does.

- *Other special characters.* Modula-2 adds some new special characters to optionally replace tokens which require a two-character combination or a reserved word in Pascal (# for <>, ~ for **not**, and & for **and**). Modifying MANGLE and MWEAVE to handle & and ~ was no problem, but # is already used by the WEB system for macro parameters. For example, the two definitions

  ```
  @d test(#)==m[#] <> x[j+#]
  @d test(#)==m[#] #  x[j+#]
  ```

  are logically equivalent from a Modula-2 standpoint, but the output generated by the regular TANGLE and WEAVE for the second would not be what the programmer expected. The parsers of the two programs would not be able to differentiate between the middle #, which means $\neq$, and those in the array index expressions, which are intended to be replaced by the macro parameter. To resolve this ambiguity, MANGLE and MWEAVE have been modified to accept the Modula-2 version of # anywhere in a WEB program *except* within a macro definition, where # will continue to represent the parameter.

  Of course, the old Pascal symbols still work. These new symbols, and the modifications to handle them, are largely irrelevant when the WEB system is being used, because MWEAVE will convert them to $\neq$, $\neg$, and $\wedge$ anyway.

- *Quotes.* Modula-2 allows strings to be delimited by either single or double quotes (' or "). While this is a definite improvement, it does conflict with WEB, in which single quotes delimit regular strings, while double quotes identify strings destined for the "string pool", a special WEB mechanism whereby the strings so designated are written to a separate text file to be read at run-time. Rather than disable the string pool, I reluctantly decided that the user would just have to continue using single quotes as in Pascal.

- *The macro package.* Surprisingly few modifications were required to the WEB macro package, WEBMAC.TEX. In fact, I decided not to modify it at all. A new file, MWEBMAC.TEX, inputs the original WEBMAC.TEX, then redefines one macro and adds one new one. The comment macro \C{...} was redefined to generate (* *) instead of { }, and \VB was added to generate the vertical bar character. Not counting blank lines, MWEBMAC.TEX is only five lines long.

The sample program provided with this article, SCANTEX, actually performs a useful function. It scans a TEX file generated by WEAVE (or MWEAVE, of course) and copies only the sections which have been modified by a change file to a new TEX file, resulting in an abbreviated program listing containing only the changes. The unchanged sections are not copied, nor are the cross-references, the section names, or the title page, which includes the table of contents. Typically, a WEB program running on a wide range of machines (such as TEX itself) has a great number of change files applied to it. For the most part, the main portion of the program is identical in all implementations and certain sections, containing "system dependencies", are different for each one. Since WEAVE generates a complete listing every time it is run, and a program the size of WEAVE or TANGLE runs to about a hundred pages (and that is small compared to TEX or METAFONT), a lot of paper is consumed printing several large listings that are essentially the same. Since writing SCANTEX, I have adopted the practice of printing one full listing generated from the pure WEB source (the way it comes from Stanford, with no change files applied), followed by an abbreviated listing generated by SCANTEX for each change file applied to that program. (In fact, in some cases I print *only* the changed sections, referring to published versions of the pure WEB source rather than printing it myself. In the case of TEX, I refer to the book *TEX: The Program*; most of the other Stanford-developed programs also exist as bound documents (available from Maria Code).

Experienced WEB users may wonder why I went to the trouble of writing a program to duplicate a function already provided by the WEB system itself, since the suppression of unchanged sections can be accomplished by placing

```
\let\maybe=\iffalse
```

into the limbo section of the WEB file. The reasons were:

1. SCANTEX does not print the index. Since the index contains entries for the full listing rather than just the abbreviated one, a program the size of TEX can have an index that is much longer than the rest of the listing.
2. Since SCANTEX is an external program, neither the WEB file nor the changefile need be modified to turn suppression on or off.
3. If the TEX file is to be saved, the reduced version generated by SCANTEX takes up much less disk space.
4. I was unaware that the builtin mechanism existed when I wrote SCANTEX (the real reason). Since it is buried deep in Appendix G of the WEB manual, it is easy to miss.

As can be seen from the SCANTEX listing, MWEB is not *that* different; on first glance, it could be mistaken for standard WEB. Closer inspection would reveal the differences in reserved words, in comments, and in compound statements. Note the use of the **elsif** statement. The boxes around words such as "WriteString" are an unforeseen side effect of use of the "pass through" WEB command described above to keep selected words from being forced to uppercase. While startling, it does point out which identifiers require special treatment. I highly recommend using the approach taken in SCANTEX: define simple macros equivalent to each of these external identifiers and use the macro everywhere else in the program, including the **import** statement. This isolates the boxes to the section containing the definitions rather than sprinkling them throughout the program.

Hopefully, MANGLE, MWEAVE, MWEBMAC.TEX, SCANTEX, and a few other sample MWEB programs should be available from Stanford on the regular distribution tape by the time this reaches print. The WEB files and the VAX implementation files should be available from Stanford and additionally from Kellerman and Smith. For people who have absolutely no way of reading a magnetic tape, the IBM PC version is available from me on PC floppies for a handling fee. Additionally, the original TANGLE and WEAVE, the WEBMERGE program described elsewhere in this issue (page 117), and several of the TEX and METAFONT utility programs (sometimes referred to as TEXware and METAFONTware) are also available on floppy. All of these have change files targeted for Microsoft Pascal running under MS-DOS on the IBM PC, which is my development system. As far as other target computers are concerned, MANGLE and MWEAVE are implemented as standard change files applied to TANGLE and WEAVE, so they can be merged with the current implementation-specific change files. WEBMERGE can be used for this purpose. If you have TANGLE and WEAVE running, you should have no trouble with MANGLE and MWEAVE.

The SCANTEX Processor

Version 1.0

1. Introduction. This program takes a TEX file generated by WEAVE and strips out the sections which have not been changed, outputting the changed sections to a second, greatly reduced TEX file. The index, section names, and table of contents are dropped as well.

The program uses a few features of the Modula-2 compiler used in its development (Logitech MS-DOS) that may need to be changed in other installations. System-dependent portions of SCANTEX can be identified by looking at the entries for 'system dependencies' in the index below.

The "banner line" defined here should be changed whenever SCANTEX is modified.

```
define
        banner ≡
        ´This␣is␣SCANTEX,␣Version␣1.0´
```

2. The program begins with a fairly normal header, made up of pieces that will mostly be filled in later. The TEX input comes from file *TeX_file* and the new TEX output goes to file *out_TeX_file*. Unlike Pascal, Modula-2 does not require the constant, type, and variable sections to be placed here in the program header in a rigidly specified order, but we will do it anyway, since WEB makes it so easy.

```
module scan_TeX; ⟨Import List 4⟩
  const ⟨Constants in the outer block 5⟩
  type ⟨Types in the outer block 6⟩
  var ⟨Globals in the outer block 7⟩
```

3. This procedure initializes the module.

```
procedure initialize;
  var ⟨Local variables for initialization 15⟩
  begin
    ⟨Set initial values 8⟩
    ⟨Initialize the file system 16⟩;
  end initialize;
```

4. A few macro definitions for low-level output instructions are introduced here. All of the terminal-oriented commands in the remainder of the module will be stated in terms of simple primitives. The boxes signify words that must not be forced to uppercase when the program is MANGLED, since Modula-2 is case-sensitive.

```
define pr_char ≡ ␣Write␣   (* library
          procedure to output a character *)
define pr_string ≡ ␣WriteString␣   (* library
          procedure to output a string *)
define rd_string ≡ ␣ReadString␣   (* library
          procedure to input a string *)
define pr_card ≡ ␣WriteCard␣
          (* library procedure to output a
          cardinal number *)
define new_line ≡ ␣WriteLn␣   (* a new line *)
define print_string(#) ≡ pr_string(#)   (* put a
          given string to the terminal *)
define read_string(#) ≡ rd_string(#)   (* read a
          given string from the terminal *)
define print_cardinal(#) ≡ pr_card(#, 1)   (* put
          a given cardinal to the terminal, in
          decimal notation, using only as many
          digit positions as necessary *)
define print_ln(#) ≡ pr_string(#); new_line;
          (* put a given string to the terminal,
          followed by a new line *)
define print_char(#) ≡ pr_char(#)   (* put a
          given character to the terminal *)
```

```
⟨Import List 4⟩ ≡
  from ␣InOut␣ import
      pr_string, rd_string, pr_char, pr_card, new_line;
```

See also sections 10 and 11.

This code is used in section 2.

5. Let's define a few constants.

```
⟨Constants in the outer block 5⟩ ≡
  buf_size = 1000;
      (* maximum length of input line *)
  file_name_len = 200;   (* length of a file name *)
```

This code is used in section 2.

6. A global variable called *history* will contain one of four values at the end of every run: *spotless* means that no unusual messages were printed; *harmless_message* means that a message of possible interest was printed but no real errors were detected; *error_message* means that at least one error was found; *fatal_message* means that the program terminated abnormally. The value of *history* does not influence the behavior of the program; it is simply computed for the convenience of systems that might want to use such information.

We don't really have to worry about errors too much in this particular program because the input is machine-generated (by WEAVE). The error likeliest to occur is failure during file opens.

```
define mark_harmless ≡
        if history = spotless then
          history ← harmless_message;
        end ;
define mark_error ≡ history ← error_message
define mark_fatal ≡ history ← fatal_message
define err_print(#) ≡ print_ln(#); mark_error;
```

```
⟨ Types in the outer block 6 ⟩ ≡
  error_level = (spotless, harmless_message,
      error_message, fatal_message);
```

This code is used in section 2.

7. ⟨ Globals in the outer block 7 ⟩ ≡

```
history: error_level;   (∗ how bad was this run? ∗)
```

See also sections 12 and 18.

This code is used in section 2.

8. ⟨ Set initial values 8 ⟩ ≡

```
  history ← spotless;
```

See also section 17.

This code is used in section 3.

9. Some implementations may wish to pass the value of the *history* variable to the operating system so that it can be used to govern whether or not other programs are started. The *doscall* procedure passes a program status value back to DOS. We use *fatal_error* to terminate the program abnormally.

```
define print_fatal_message ≡
            print_string('(That␣was␣a␣fatal␣');
        print_ln('error,␣my␣friend.)')
define fatal_error(#) ≡ mark_fatal; print_ln(#);
        print_fatal_message; doscall("4C, history);
```

```
⟨ Terminate program, converting history to program
      exit status 9 ⟩ ≡
  doscall("4C, history);
```

This code is used in section 24.

10. If we are going to use *doscall* from the Logitech library we have to import it from the module *system*.

```
⟨ Import List 4 ⟩ +≡
  from system import doscall;
```

11. File Handling. Here we define the symbols for use with file handling.

```
define lookup ≡ ␣Lookup␣
            (∗ library procedure to open a file ∗)
define close ≡ ␣Close␣
            (∗ library procedure to close a file ∗)
define failure(#) ≡ (#.res ≠ done)
            (∗ last file operation sucessful ? ∗)
define abort_if_open_error(#) ≡
        if failure(#) then
          print_string('unable␣to␣open␣');
          fatal_error(filename);
        end ;
define open_input_file(#) ≡ lookup(#, filename,
            false); abort_if_open_error(#)
define open_output_file(#) ≡ lookup(#, filename,
            true); abort_if_open_error(#)
define close_file(#) ≡ close(#);
define end_file ≡ eof
define null_char ≡ ␣nul␣
define end_of_line(#) ≡ (ch = eol)
define end_of_file(#) ≡ (#.end_file)
define read_char ≡ ␣ReadChar␣
define write_char ≡ ␣WriteChar␣
define input_char(#) ≡ read_char(#, ch);
define output_char(#) ≡ write_char(#, ch)
define read_ln(#) ≡
        while ¬end_of_line(#) do
          input_char(#);
        end ;
define write_ln(#) ≡ write_char(#, eol);
define text_file ≡ ␣File␣
```

```
⟨ Import List 4 ⟩ +≡
  from ␣FileSystem␣ import lookup, Response,
      read_char, write_char, text_file, close;
  from ascii import eol, null_char;
```

12. Input goes into an array called *buffer*.

```
⟨ Globals in the outer block 7 ⟩ +≡
buffer: array [0 .. buf_size] of char;
TeX_file, out_TeX_file: text_file;
```

13. The *input_ln* procedure brings the next line of input from the specified file into the *buffer* array and returns the value *true*, unless the file has already been entirely read, in which case it returns *false*. Under normal conditions, we will never reach true end of file, for reasons discussed in later sections, but we will handle it anyway. Trailing blanks are ignored and the global variable *limit* is set to the length of the line. The value of *limit* must be strictly less than *buf_size*.

```
procedure input_ln(var f : text_file): boolean;
        (* inputs a line or returns false *)
  var final_limit: [0 .. buf_size];
        (* limit without trailing blanks *)
    ch: char;  (* current input character *)
    line_pres: boolean;
        (* temporary result of procedure *)
  begin
    limit ← 0; final_limit ← 0;
    if end_of_file(f) then
      line_pres ← false
    else
      input_char(f);
      while ¬end_of_line(f) do
        if ch = null_char then
          return false
        end ;
        buffer[limit] ← ch; inc(limit);
        if buffer[limit − 1] ≠ '␣' then
          final_limit ← limit
        end ;
        if limit = buf_size then
          read_ln(f); dec(limit);
          err_print('!␣Input␣line␣too␣long');
          return true;
        end ;
        input_char(f);
      end ;
      read_ln(f); limit ← final_limit;
      line_pres ← true;
    end ;
    return line_pres;
  end input_ln;
```

14. The *output_ln* procedure writes the next line of output from the *buffer* array to the specified file.

```
procedure output_ln(var f : text_file);
        (* outputs a line *)
  var ch: char;  (* current output character *)
    temp: [0 .. buf_size];
  begin
    if limit > 0 then
      for temp ← 0 to limit − 1 do
        ch ← buffer[temp]; output_char(f);
      end ;
    end ;
    write_ln(f);
  end output_ln;
```

15. We define *filename* local to the initalization procedure because it is used only during file open.

```
⟨ Local variables for initialization 15 ⟩ ≡
filename: array [0 .. file_name_len − 1] of char;
```

This code is used in section 3.

16. In this section we open both of the files.

```
  define next_file(#) ≡ filename[0] ← '␣';
        print_ln(#); read_string(filename);
        new_line; print_ln(filename); new_line;

⟨ Initialize the file system 16 ⟩ ≡
  next_file('TeX␣file:');
  open_input_file(TeX_file);
  next_file('output␣TeX␣file:');
  open_output_file(out_TeX_file);
```

This code is used in section 3.

17. Here we initialize most of the variables. The *output_enabled* flag is initialized to *true* so that the lines in the header of the WEAVE-generated TEX file, known as "limbo text", are picked up in addition to the changed sections.

```
⟨ Set initial values 8 ⟩ +≡
  TeX_line ← 0; out_TeX_line ← 0; limit ← 0;
  buffer[0] ← '␣'; input_has_ended ← false;
  output_enabled ← true;
```

18.

```
⟨ Globals in the outer block 7 ⟩ +≡
TeX_line: cardinal;  (* the number of the current
        line in the main TEX file *)
out_TeX_line: cardinal;  (* the number of the line
        in the output TEX file *)
limit: [0 .. buf_size];  (* the last character position
        occupied in the buffer *)
input_has_ended: boolean;
        (* there is no more input *)
output_enabled: boolean;
        (* we are copying input lines to output *)
```

19. Main Input Loop. This is the main processing loop of SCANTEX. We simply read lines until end of file. The *get_line* procedure will determine the setting of the *output_enabled* flag. If set, we copy the line to the output file.

```
⟨ Read the input 19 ⟩ ≡
  while ¬input_has_ended do
    get_line;
    if output_enabled then
      output_ln(out_TeX_file); inc(out_TeX_line);
    end ;
  end ;
```

This code is used in section 24.

20. The *get_line* procedure is called to read in the next line and scan it. We will output an "I'm alive!" dot to the terminal every 100 input lines and a new line every 2000.

```
procedure get_line;   (* inputs the next line *)
  var keep_looking: boolean; temp_index: cardinal;
  begin
    input_has_ended ← ¬input_ln(TeX_file);
    if input_has_ended then
      output_enabled ← false; return ;
    else
      inc(TeX_line);
      if (TeX_line mod 100) = 0 then
        print_char(´.´);
        if (TeX_line mod 2000) = 0 then
          new_line;
        end ;
      end ;
      ⟨Scan the line 21⟩;
      buffer[limit] ← ´␣´;
    end ;
  end get_line;
```

21. In this section we determine whether the current line is the beginning of a section (`\M` or `\N` at beginning of line, followed immediately by a section number) and, if so, whether the section has been modified (`\*.` following the number). We update the *output_enabled* flag according. Additionally, the index section (`\inx`) is considered end of file. If it is detected, we set the flag *input_has_ended* to terminate the program and set *output_enabled* to false to keep the `\inx` command from being copied to the output file.

```
  define numeric_digit_at(#) ≡ ((buffer[#] ≤ ´9´)
            ∧ (buffer[#] ≥ ´0´))
  define third_char_matches(#) ≡
            (buffer[temp_index + 2] = #)
  define second_char_matches(#) ≡
            (buffer[temp_index + 1] = #) ∧
            third_char_matches
  define char_matches(#) ≡ (buffer[temp_index] =
            #)
  define three_chars_match(#) ≡ char_matches(#)
            ∧ second_char_matches

⟨Scan the line 21⟩ ≡
  temp_index ← 1;
  if (limit > 3) ∧ (buffer[0] = ´\´) then
    if (char_matches(´M´) ∨ char_matches(´N´)) ∧
          numeric_digit_at(2) then
      ⟨Search for ‘\*.’; set output_enabled if
          found 22⟩;
    elsif three_chars_match(´i´)(´n´)(´x´) then
      output_enabled ← false;
      input_has_ended ← true;
    end ;
  end ;
```

This code is used in section 20.

22. Starting at the first digit of the section number, search for `\*.`, which indicates that this is a changed section. Discontinue the search if `\*.`is found or the current position is no longer a numeric digit, which means we have moved past the section number without finding it.

```
⟨Search for ‘\*.’; set output_enabled if found 22⟩ ≡
  output_enabled ← false; keep_looking ← true;
  temp_index ← 3;
  while (¬output_enabled) ∧ keep_looking do
    output_enabled ←
          three_chars_match(´\´)(´*´)(´.´);
    keep_looking ← numeric_digit_at(temp_index);
    inc(temp_index);
  end ;
```

This code is used in section 21.

23. The command to generate the table of contents (`\con`) is normally the last line in a TeX file generated by WEAVE. Part of its function is to terminate TeX gracefully by generating a `\bye` command or equivalent after generating the contents. Since we are dropping the `\con` command, we must issue the `\bye` command directly, just before closing the input and output files.

```
⟨Add ‘\bye’ command to end of output and close
      both files 23⟩ ≡
  buffer[0] ← ´\´; buffer[1] ← ´b´;
  buffer[2] ← ´y´; buffer[3] ← ´e´; limit ← 4;
  output_ln(out_TeX_file); inc(out_TeX_line);
  close_file(TeX_file); close_file(out_TeX_file);
```

This code is used in section 24.

24. Main Program. This is the main program.

```
begin
  print_ln(banner); initialize;
  ⟨Read the input 19⟩;
  ⟨Add ‘\bye’ command to end of output and
      close both files 23⟩;
  ⟨Print statistics about line counts 26⟩;
  ⟨Print the job history 25⟩;
  ⟨Terminate program, converting history to
      program exit status 9⟩;
end scan_TeX.
```

25. Here we simply report the history to the user.

```
⟨Print the job history 25⟩ ≡
  case history of
  spotless: print_ln(´(No␣errors␣were␣found.)´)
        |
  harmless_message:
        print_string(´(Did␣you␣see␣the␣´);
    print_ln('warning␣message␣above?')  |
  error_message:
        print_string('(Pardon␣me,␣but␣I␣think␣I␣');
    print_ln('spotted␣something␣wrong.)')  |
  fatal_message: print_fatal_message
end ;  (* there are no other cases *)
```

This code is used in section 24.

26. ⟨Print statistics about line counts 26⟩ ≡

```
  new_line; print_ln('Line␣count␣statistics:');
  print_cardinal(TeX_line);
  print_ln('␣lines␣in␣input␣TeX␣file');
  print_cardinal(out_TeX_line);
  print_ln('␣lines␣in␣output␣TeX␣file');
```

This code is used in section 24.

27. Index.

⟨ Add '\bye' command to end of output and close both files 23 ⟩ Used in section 24.
⟨ Constants in the outer block 5 ⟩ Used in section 2.
⟨ Globals in the outer block 7, 12, 18 ⟩ Used in section 2.
⟨ Import List 4, 10, 11 ⟩ Used in section 2.
⟨ Initialize the file system 16 ⟩ Used in section 3.
⟨ Local variables for initialization 15 ⟩ Used in section 3.
⟨ Print statistics about line counts 26 ⟩ Used in section 24.
⟨ Print the job *history* 25 ⟩ Used in section 24.
⟨ Read the input 19 ⟩ Used in section 24.
⟨ Scan the line 21 ⟩ Used in section 20.
⟨ Search for '*.'; set *output_enabled* if found 22 ⟩ Used in section 21.
⟨ Set initial values 8, 17 ⟩ Used in section 3.
⟨ Terminate program, converting *history* to program exit status 9 ⟩ Used in section 24.
⟨ Types in the outer block 6 ⟩ Used in section 2.

Fonts

Blacker Thoughts

John S. Gourlay
Ohio State University

Like many owners of write-white laser printers, I found a few months ago that the "cm" series of Computer Modern fonts, as distributed, is unacceptably faint on my Xerox 2700. I began my search for more suitable METAFONT parameter settings with the "conjectural" settings for QMS printers, which share the same print engine as the 2700. I was immediately disappointed, however. Printed, the new bitmaps were acceptably black, but they didn't look anything like the Computer Modern in *Computer Modern Typefaces*, and not even very much like the original bitmaps printed on a write-black Canon engine.

Laser printers work by producing patterns of electric charge on a piece of paper. The charge attracts particles of black "toner," which eventually forms a permanent printed image. Write-black laser printers start with an uncharged piece of paper and in effect use a laser to place spots of charge on the paper. Write-white laser printers start with a fully charged piece of paper and then use a laser to remove the charge in places where the final image should remain white. In both cases the round spot produced by the laser is slightly larger than a pixel so that no gaps are left between spots in solid regions of black or white. For this reason, lines drawn on a write-black laser printer tend to be slightly thicker than one would expect given their width in pixels, and lines drawn on a write-white printer tend to be slightly thinner (the "white lines" are thicker).

The plain base file of METAFONT anticipates this kind of systematic difference between printers by providing a parameter called *blacker* whose value can be added to the thickness of pen strokes to compensate for any thinning inherent in the printing process. After some experimentation with various settings of *blacker* I decided empirically that the higher I made the value of *blacker* the smaller I found such lowercase letters as o and e to become. Also decreasing were the sizes of the bowls of such letters as p and b, the widths of m, n, and the lower part of h. The overall impression was that the "x-height" of the font was decreasing as *blacker* increased. At the conjectured setting of $blacker = .75$, the effect was great enough to make the font look entirely different and much less legible than the model in *Computer Modern Typefaces*.

Once I saw the problem it wasn't hard to see why it was happening. Looking at the METAFONT code for the roman lowercase o, one can see that it is drawn with a variable-width pen moving along a path through four points at the character's top, left, bottom, and right. Concentrating on point 1, the top point of the o, the relevant METAFONT statements are

$$penpos_1(vair, 90);$$

and

$$y_{1r} = h + \mathrm{vround}\,1.5oo;$$

The first says that the pen at point 1 has a nib of width *vair* and it is held vertically with the "right" edge of the nib at the top. The second says that the right (or top) edge of the pen should be at a distance $h + \mathrm{vround}\,1.5oo$ from the baseline. The parameter *blacker* figures into this because the pen width, *vair*, increases as *blacker* increases. Since the location of the top edge is fixed, an increase in *blacker* causes the whole pen to move down, and all the extra width appears at the bottom edge of the pen stroke. The same thing happens at the sides and bottom of the o, so the overall effect of an increase in *blacker* is a decrease in the diameter of the path forming the o.

An increase in *blacker* should cause the pen width to increase, but it should not cause the pen's path to change. The extra width should be distributed equally on both sides of the stroke on the assumption that the printer will erode both sides of the stroke equally. Unfortunately, I do not see any easy way to modify the METAFONT code to make it behave this way. The best results would be obtained by moving all the points that are positioned relative to a pen edge outward by .5*blacker*. For example, we could change the second statement above to

$$y_{1r} - .5\mathit{blacker} = h + \mathrm{vround}\,1.5\mathit{oo};$$

This would require an enormous amount of error prone work, however, because every point in the full set of Computer Modern fonts would have to be studied and a large proportion changed. A more tractable approach would be to remove all references to *blacker* in the definitions of pen widths and to modify such commands as **penstroke** and **filldraw stroke** to broaden their strokes by *blacker* automatically. This would limit the number of changes that would have to be made to the code, but it would have the disadvantage of nullifying the carefully planned rounding of pen widths, perhaps ruining the fonts in other ways.

Not wishing to tackle either of these projects immediately, I decided to live within the limits of the existing parameters. After many experiments I arrived at a compromise set of parameters:

$$\mathit{blacker} := .6;$$
$$\mathit{fillin} := -.3;$$
$$\mathit{o\_correction} := .6;$$

At this value of *blacker*, most of the characters keep their original sizes, but it is not quite enough to compensate for the thinning inherent in the printer. The rather extreme setting of *fillin*, which thickens diagonals, seems to correct the remaining faint spots. (At one point I tried to use *o_correction* to enlarge the shrunken bowls of *blacker* = .75, but with hindsight I should have known better. The results were not the kind of thing that I would take outside the privacy of my own home.) There are still some ugly features in the resulting fonts, particularly an inconsistency in the weights of characters. Nevertheless, I feel that this set of parameters is considerably better than the ones that result from the "conjectural" parameters, and also better than the "am" fonts they replace. There is room for improvement, however, as well as a challenge for the next generation of METAFONT designers.

Copies of these fonts ready for downloading to the Xerox 2700 can be obtained from me or preferably from

Margot Nelligan
Xerox Printing Systems Division
880 Apollo Street
El Segundo, CA 90245.

Updated Computer Modern Fonts for the LN03

John Sauter

Included with TUGboat volume 8 number 1 was the usual errata sheet for the TeX programs and documentation. Among the corrections to *Computers and Typesetting, volume E*, were changes to the parameters. The effect of these changes is to change the shapes of some of the Computer Modern characters.

I have been making available "alternative" versions of the Computer Modern parameter files since TUGboat volume 7 number 4, so these changes to volume E make my files obsolete. Fortunately, the files are easily fixed. Anyone who got a tape from me not marked METAFONT version 1.3 (or later) please make the following changes. I have taken some liberties with the spacing in order to fit the corrections into TUGboat's columns. You can use whatever spacing you wish when you change the files, except that a comment that starts with % must end on the same line as the %.

In COMPUTE_CMR.MF, starting at line 128, change four lines from

```
%elseif design_size < 12:
    ((design_size*15)+150)
else:  ((0.020812520812*
    design_size*design_size) +
(14.5421245421*design_size) +
(152.49750249))fi)/360pt#;
```

to

```
elseif design_size < 40:
    ((-0.23934398934*design_size*
    design_size) +
(20.265567765*design_size) +
(121.278721278))
else:  (548.951048934)fi)/360pt#;
```

Replace line 167, which looks like this:

```
else:  ((design_size*9.4696969696)+
    236.36363637)fi)/360pt#;
```

with these four lines:

```
elseif design_size < 30:
    ((-0.4995004995*design_size*
    design_size) +
(25.989010989*design_size) +
(110.059940059))
else:  (440.179820179)fi)/360pt#;
```

At line 173, change these three lines

```
else:  ((0.020812520812*design_size*
    design_size) +
(14.5421245421*design_size) +
(222.49750249))fi)/360pt#;
```

to the following four lines:

```
elseif design_size < 45:
    ((-0.23934398934*design_size*
    design_size) +
(20.265567765*design_size) +
(191.278721278))
else:  (618.557692303)fi)/360pt#;
```

Lastly, change five lines starting at line 269 from

```
serif_drop#:=
    % vertical drop of sloped serifs
    (17.28 pt looks strange)
(if design_size < 12:  (design_size*4)
else:  ((design_size*design_size*
    2.62445887445) -
(design_size * 53.738095238) +
314.935064935)fi)/360pt#;
```

to

```
serif_drop#:=
    % vertical drop of sloped serifs
(if design_size < 12:  (design_size*4)
else:  ((design_size*design_size*
    0.0228937728937) +
(design_size * 3.49633699633) +
2.74725274725)fi)/360pt#;
```

In COMPUTE_CMSS.MF there is only one change: replace three lines starting at line 47 with two lines. The original lines look like this:

```
(if design_size < 8:
    ((design_size*235)+10)
elseif design_size < 9:
    ((design_size*470)-1870)
elseif design_size < 10:  (2360)
```

and the new lines look like this:

```
(if design_size < 9:
    ((design_size*230)+50)
elseif design_size < 10:
    ((design_size*240)-40)
```

These changes affect only the Computer Modern Roman font at 17 point and the Computer Modern Sans-serif and Slanted Sans-serif fonts at 9 point, so only the CMR17, CMSS9 and CMSSI9 fonts need to be recompiled.

The errata sheet in TUGboat volume 8 number 1 also contained some changes to METAFONT, turning METAFONT version 1.0 into METAFONT version 1.3. At the time I wrote this article I had not yet finished testing all of the Computer Modern fonts with the LN03, but from what I have seen so far METAFONT version 1.3 produces slightly thinner diagonal lines than version 1.0 in some cases, which seems to improve the appearance of the Computer Modern fonts on the LN03.

It is still my intention to put these alternative parameter files on the Stanford tape, along with the pixel and GF files for the LN03. I have recently gotten an encouraging message from David Fuchs, but at the time I wrote this I was still not certain that I would be successful. Therefore, I renew my offer to send anyone running VAX/VMS who has an LN03 and the Stanford tape a VMS Backup copy of everything necessary to print documents using the Computer Modern fonts on the DEC LN03: command files, alternative parameter files, and the resulting .TFM and pixel files for the DEC LN03. If you write me I'll send you a magnetic tape by return mail. If you can't read 6250 BPI tapes be sure to let me know, since that is my default density: it lets me use a smaller tape.

If I succeed in getting these files onto the Stanford tape I'll write another TUGboat article withdrawing this offer; to avoid hassle for the TEX community I'll fill requests until that article is published.

MFtool: A Description of a METAFONT Script-Driven Processing Facility

John M. Crawford
Ohio State University

I thought I might take a moment to describe some of the elements of an environment I've hacked together which helps me in generating fonts using METAFONT. I wanted to develop a mechanism which would allow me to generate new fonts with METAFONT, on demand, without much overhead on my part. What I've come up with is a file-driven facility which allows me to generate fonts from a set of what I'll call script files. Using the command procedure language available with my operating system (Primos), I've put together this processing facility which provides the ability to make repeated invocations of METAFONT while varying specified elements for each call. Perhaps a similar solution could be designed for any given operating environment. Particular elements of this system were added as individual needs arose, so the system design is not necessarily very elegant.

A major element of this scripting file definition describes the font and magnifications desired. This information is provided as a text prefix of the font to be generated, followed (optionally) by the magstep at which the font is to be generated. Multiple requests for a font at various magsteps can be scripted by specifying magsteps within parentheses. A file might then contain

```
      { tfm loaded by plain TeX }
cmr10 (0 0.5 1 2 3 4 5)
cmr12 (0 0.5 1 2)
cmr17 (0 0.5 1 2)
```

This example would invoke METAFONT fifteen times. We also see an example of a comment. Comments may be indicated in two ways: Text preceded by a left curly bracket is discarded. If any given line of text has a space in column 1, then that line is also treated as a comment and ignored.

This system also allows one to select specific base files, or include METAFONT specifications to be fed to METAFONT. For example,

```
spec:\mode=qms
base:&cm
```

could be specified; this must be done before the fonts are chosen. I've also included labels and an "ignore until label" goto facility. Additionally, to improve checkpointing of output, text can be displayed in screen traffic, with a "type" directive. With these elements, I've been able to build specification files and font family files which allow me to generate various sets of METAFONT fonts quickly.

As an example, I wanted to generate the METAFONT logo font for use with a personal computer preview package, mainly as a test. I chose to create a new base file, which specified some new *mode_def*'s I wanted to try. After creating with INIMF a base file called JMC (my initials) containing three specifications for preview fonts, I ran a file similar to the following:

```
base:&jmc
spec:\mode=preview
logo10 (0 0.5 1 2)
spec:\mode=previeww
logo10 (0 0.5 1 2)
spec:\mode=previewww
logo10 (0 0.5 1 2)
```

With that, I'd generated the GF files and TFM file which I'd need to later use the LOGO font on the PC.

In actuality, I can specify several script files to process. I generally divide files into FONT, BASE and SPEC script files. This allows greater flexibility when building a script for a new run. By extending and modifying this basic font generation scheme, I've been able to build various sets of fonts easily.

Update: METAFONT mode_def Settings for Various TeX Output Devices

Barbara Beeton

An earlier article by this title appeared in TUGboat Vol. 8, No. 1, page 33. Almost immediately, corrections and new information started arriving. The present iteration attempts to correct the errors of the previous article and present additional information now available. If interest warrants, this may become an "annual" column.

As Neenie Billawala has explained (TUGboat Vol. 8, No. 1, pages 29–32), the marking characteristics of different print engines must be taken into account in order to assure legible, attractive output. For the Computer Modern family, this is done by tuning several parameters built into the METAFONT design. The settings for all printers used at Stanford appear in the file `WAITS.MF`. Other settings are frequently requested and (less frequently) communicated in TeXhax or Laser-Lovers.

Here is a typical **mode_def** setting, adapted from `PLAIN.MF` (*The METAFONTbook*, page 270) for 200 dpi devices (such as the Xerox XGP, the original TeX output device); it has been augmented by the parameter *aspect_ratio* (required for non-square rasters; the default value is given).

```
mode_def lowres =
 proofing:=0;        % not making proofs
 fontmaking:=1;      % we are making a font
 tracingtitles:=0;  % don't show titles
 pixels_per_inch:=200;
 blacker:=.65;    % make pens a bit blacker
 fillin:=.2;  % adjust for diagonal fillin
 o_correction:=.4;  % less overshoot
 aspect_ratio:=1/1; % vertical/horizontal
 enddef;
```

For all font "production", typical settings are *proofing* = 0 and *fontmaking* = 1. *tracingtitles* is usually set to 0 for low-resolution fonts (400 dpi or less) and to 1 for higher-resolution fonts, to reassure one that the computer is still in operation and to indicate how far it has progressed during a long job. The standard proof settings can be found in `PLAIN.MF` as already noted.

For more guidance, see *Adapting to local conditions, The METAFONTbook*, page 278.

Stan Osborne has observed in TeXhax that "Anyone interested in understanding these parameters should read *The METAFONTbook* and experiment by setting sentences and paragraphs with many sizes of their new fonts. The look, blackness, readability, feel, taste, etc., of the variations of new fonts should be compared with the samples found in the cmr book." This excellent advice should not be ignored.

The table on the next page contains a summary of the relevant settings gleaned from available sources. Most of the print engines cited in the table are listed below, along with an indication of whether they are write-black (wb) or write-white (ww), if known, and the names of some of the output devices into which they have been built.

Canon CX (wb)	Apple LaserWriter, Cordata, HP LaserJet, Imagen 8/300, QMS and Talaris 8 ppm printers
Canon LBP-10	Imagen 10/240
Canon (wb)	Imagen 3320, Imagen 7320
Ricoh 4080 (ww)	DEC LN03; TI OmniLaser 2115
Ricoh LP4120 (ww)	HP 2688A, Imagen 12/300
Ricoh 4150 (ww)	Talaris 1500
Xerox XP-12 (ww)	DEC LN01, QMS 1200, Talaris 1200, Xerox 2700
Xerox XP-24 (ww)	Imagen 24/300, QMS 2400, Talaris 2400, Xerox 3700

As always, additions and corrections to this list are solicited.

A late note from John Lavagnino of Brandeis University warns against assuming that "improved" models of printers, or even printers from different manufacturers based on the same print engine, will produce equivalent output:

> We have discovered that the LN03 and the LN03-Plus don't print the same way: a font that looks fine on the LN03 will look lighter on the LN03-Plus. In fact it isn't necessary to download fonts to observe this: even the internal fonts look different.
>
> We've been badgering DEC about this for some time, and they have finally agreed that this is the case. The current story is that they "made the pixels smaller" on the LN03-Plus, "to make it look more like a typewriter."
>
> Be aware, then, that a good **mode_def** for one will only be a poor approximation for the other.

Consider yourselves warned.

Typical **mode_def** parameter settings for CM fonts

Source of information		*pixels_per_inch*	*blacker*	*fillin*	*o_correction*	*aspect_ratio*
`PLAIN.MF`						
`proof`		2601.72	0	0	1	
`lowres`		200	.65	.2	.4	
`WAITS.MF`						
`dover`	(Xerox Dover)	384	1.2	0	.6	
`imagen`	(Canon CX)	300	0	.2	.6	
`qms`	(Xerox XP-12E)	300	.75*	0*	.5*	
`aps`	(APS-Micro5)	722.909	.2	.2	1	
`crs`	(Alphatype CRS)	4000+4000/3	.4	0	1	
`boise`	(HP 2680A)	180	.55	.1	.3	
`DD`	(DataDisc terminal)	70	0	0	.2	
`canon`	(Canon LBP-10)	240	.2	.2	.4	
`newDD`	(DataDisc terminal)	70	0	0	.2	4/3
`cg`	(Compugraphic 8600)	1301.5	.2	.2	1	1569/1301.5
`epson`		240	0	0	.2	9/10
Charles Karney, TEXhax 86#4, Oct 86 [Note 1]						
`qms`	(Xerox XP-12E)	300	.8	.2	.4	
John Gourlay, May 87 [Note 2; page 128]						
`xeroxxxvii`	(Xerox XP-12)	300	.6	−.3	.6	
Charles LaBrec, TEXhax 86#6, Oct 86 [Note 3]						
`decln`	(Ricoh 4080)	300	.9	−.2	.5	
Stan Osborne, Apr 87 [Note 4]						
`decln`	(Ricoh 4080)	300	.2	−.4	.5	
Janene Winter, May 87 [Note 5; page 178]						
`ibm`	(IBM 3820) (ww)	240	.65	−.2	.45	
`ibm-a`	(IBM 3812) (ww)	240	.4	−.2	.4	
`ibm-b`	(IBM 3800) (ww)	240	.2	−.1	.6	
`ibm-c`	(IBM 4250) (ww)	600	.05	0	.6	
`sherpa`	(IBM 6670) (wb)	240	1	1	.6	
Matthias Feyerabend, GSI, Darmstadt, May 87 [Note 6]						
`ibmlaser`	(IBM 3820)	240	.3	−1	.4	
`bensmall`	(Benson 9211)	200	−.5	0	.4	
`bensmall`	[alternate settings for problem fonts]	200	0	0	.4	
`benbig`	(Benson 9436)	254	−.8	0	.4	
`benbig`	[alternate settings for problem fonts]	254	−.1	0	.4	

* A note in `WAITS.MF` states that these settings are conjectural.

Notes:

1. Charles Karney states, "...I haven't fully explored the parameter space. If anyone knows of a better (or 'authorized') solution, I'd appreciate hearing about it."
   [`Karney%PPC.MFEnet@LLL-MFE.Arpa`]
2. John Gourlay has diagnosed an unexpected modification to the pen path as *blacker* increases, causing the diameter of such letters as "o" to decrease; the details are discussed in his article on page 128. The parameter values given here are a compromise, allowing most characters to keep their original sizes, although the value of *blacker* "is not quite enough to compensate for the thinning inherent in the printer." There is still "an inconsistency in the weights of characters. Nevertheless, [Gourlay] feel[s] that this set of parameters is considerably better than the ones that result from the 'conjectural' parameters, and also better than the 'am' fonts they replace."
   `Gourlay.Ohio-State@csnet-relay`
3. Charles LaBrec's comments: "I have twiddled the parameters a bit, and this seems to produce good 12 point cm fonts. I am a bit unsure because changing *blacker*, *fillin*, or *o_correction* seem to make no difference for quite a large range of values. I can't remember exactly, but you will get the same results as [these] for $.4 < blacker < .9$, $-.8 < fillin < -.1$, and $0 < o\_correction < .7$. But this probably makes a good starting point."
   [`crl@newton.physics.purdue.edu`]
   [Editor's note: The value given in TUGboat 8#1 for `decln` *fillin* should have been −.2, not +.2.]
4. Stan Osborne: "The `decln` mode [Mr. LaBrec] suggested did not *fillin* correctly and was too black for the smaller point sizes. His choice of settings produces small sized fonts that are much blacker than the small cmr's found in the cmr book (Vol E). ... I found the [above] values of *blacker* and *fillin* to produce readable small fonts for an LN03.... These values were not carefully tested for larger point sizes. (I stopped experimenting when I got something I liked and I had verified that larger sizes were also usable.) [`...!ucbvax!dual!dbi!stan`]
5. Janene Winter has found these settings "to be optimal for the IBM printers". This information was transmitted by Dean Guenther along with his site report (page 178).
6. Matthias Feyerabend: "Fonts tested are CMR5, CMR10, CMR12 and CMSSI17 for a full range of settings for *blacker* and *fillin*."

Fonts for Digital Halftones

Donald E. Knuth
Stanford University

Small pictures can be "typeset" on raster devices in a way that simulates the screens used to print fine books on photography. The purpose of this note is to discuss some experiments in which METAFONT has created fonts from which halftones can be generated easily on laser printers. High levels of quality are not possible at low resolution, and large pictures will overflow TEX's memory at high resolution; yet these fonts have proved to be useful in several applications, and their design involves a number of interesting issues.

I began this investigation several years ago when about a dozen of Stanford's grad students were working on a project to create "high-tech self-portraits" [see Ramsey Haddad and Donald E. Knuth, "A programming and problem-solving seminar," *Stanford Computer Science Report 1055* (Stanford, California, June 1985), pp. 88–103]. The students were manipulating digitized graphic images in many ingenious ways, but Stanford had no output devices by which the computed images could be converted to hardcopy. Therefore I decided to create a font by which halftones could be produced using TEX.

Such a font is necessarily device-dependent. For example, a laser printer with 300 pixels per inch cannot mimic the behavior of another with 240 pixels per inch, if we are trying to control the patterns of pixels. I decided to use our 300-per-inch Imagen laserprinter because it gave better control over pixel quality than any other machine we had.

It seemed best at first to design a font whose "characters" were tiny 8×8 squares of pixels. The idea was to have 65 characters for 65 different levels of brightness: For $0 \le k \le 64$ there would be one character with exactly k black pixels and $64 - k$ white pixels.

Indeed, it seemed best to find some permutation p of the 64 pixels in an 8×8 square so that the black pixels of character k would be $p_0, p_1, \ldots, p_{k-1}$. My first instinct was to try to keep positions $p_0, p_1, p_2, \ldots$ as far apart from each other as possible. So my first METAFONT program painted pixels black by ordering the positions as follows:

45	29	34	18	46	30	33	17
13	61	2	50	14	62	1	49
39	23	40	24	36	20	43	27
7	55	8	56	4	52	11	59
47	31	32	16	44	28	35	19
15	63	0	48	12	60	3	51
37	21	42	26	38	22	41	25
5	53	10	58	6	54	9	57

[This is essentially the "ordered dither" matrix of B. E. Bayer; see the survey paper by Jarvis, Judice, and Ninke in *Computer Graphics and Image Processing* **5** (1976), 22–27.]

It turns out to be easy to create such a font with METAFONT:

```
% halftone font with 65 levels of gray, characters "0" (white) to "p" (black)
pair p[]; % the pixels in order (first p0 becomes black, then p1, etc.)
pair d[]; d[0]=(0,0); d[1]=(1,1); d[2]=(0,1); d[3]=(1,0); % dither control
def wrap(expr z)=(xpart z mod 8,ypart z mod 8) enddef;
for i=0 upto 3: for j=0 upto 3: for k=0 upto 3:
 p[16i+4j+k]=wrap(4d[k]+2d[j]+d[i]+(2,2)); endfor endfor endfor
w#:=8/pt; % that's 8 pixels
font_quad:=w#; designsize:=8w#;
picture prevchar; prevchar=nullpicture; % the pixels blackened so far
for i=0 upto 64:
 beginchar(i+ASCII"0",w#,w#,0); currentpicture:=prevchar;
 if i>0: addto currentpicture also unitpixel shifted p[i-1]; fi
 prevchar:=currentpicture; endchar;
 endfor
```

This file was called dt.mf; I used it to make a font called 'dt300' by applying METAFONT in the usual way to the following file dt300.mf:

```
% Halftone font for Imagen, dithered
mode_setup;
if (pixels_per_inch<>300) or (mag<>1):
 errmessage "Sorry, this font is only for resolution 300";
 errmessage "Abort the run now or you'll clobber the TFM file";
 forever: endfor
else: input dt fi
end.
```

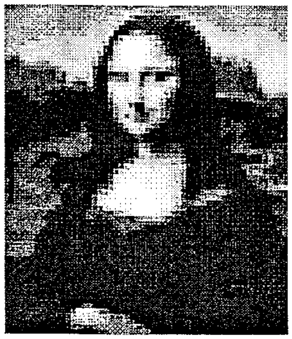
(The purpose of dt300.mf is to enforce the device-dependence of this font.)

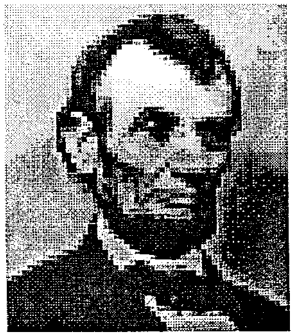
It's fairly easy to typeset pictures with dt300 if you input the following macro file hf65.tex in a TeX document:

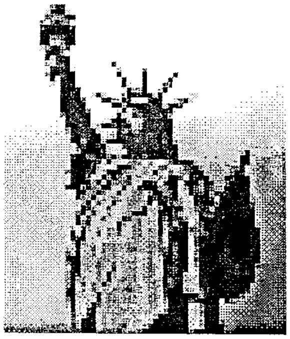
```
\font\halftone=dt300  % for halftones on the Imagen 300
\chardef\other=12

\def\beginhalftone{\vbox\bgroup\offinterlineskip\halftone
 \catcode`\\=\other \catcode`\^=\other \catcode`\_=\other
 \catcode`\.=\active \starthalftone}
{\catcode`\.=\active \catcode`\/=0 \catcode`\\=\other
 /gdef/starthalftone#1\endhalftone{/let.=/endhalftoneline
  /beginhalftoneline#1/endhalftone}}
\def\beginhalftoneline{\hbox\bgroup\ignorespaces}
\def\endhalftoneline{\egroup\beginhalftoneline}
\def\endhalftone{\egroup\setbox0=\lastbox\unskip\egroup}

% Example of use:
% \beginhalftone
% chars for top line of picture.
% chars for second line of picture.
% ...
% chars for bottom line of picture.
% \endhalftone
```

(These macros are a bit tricky because '\' is one of the legal characters in dt300; we must make backslashes revert temporarily to the status of ordinary symbols.)

Unfortunately, the results with dt300 weren't very good. For example, here are three typical pictures, shown full size as they came off the machine:*



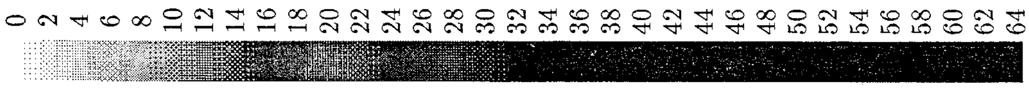




The squareness of the pixels is much too prominent.

* Asterisks are used throughout this paper to denote places where output from the 300-pixels-per-inch Imagen printer has been pasted in. Elsewhere, the typesetting is by an APS Micro-5, which has a resolution of about 723 pixels per inch.

Moreover, the laser printer does strange things when it is given pixel patterns like those in dt300:*



Although character k has more black pixels than character $k-1$, the characters do not increase their darkness monotonically! Character 6 seems darker than character 7; this is an optical illusion. Character 32 is darker than many of the characters that follow, and in this case the effect is not illusory: Examination with a magnifying glass shows that the machine deposits its toner in a very curious fashion.

Another defect of this approach is that most of the characters are quite dark; 50% density is reached already at about character number 16. Hence dt300 overemphasizes light tones.

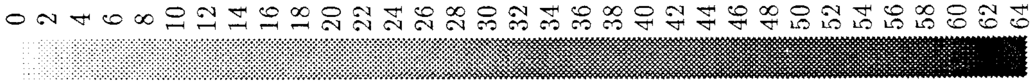
My next attempt was to look at halftone pictures in books and newspapers, in order to discover the secret of their success. Aha! These were done by making bigger and bigger black dots; in other words, the order of pixels p_0, p_1, ... was designed to keep black pixels *close together* instead of far apart. Also, the dots usually appear in a grid that has been rotated 45°, since human eyes don't notice the dottiness at this angle as much as they do when a grid is rectilinear. Therefore I decided to blacken pixels in the following order:

							63				
						39	47	55			
						28	15	23	31	35	
					52	20	4	7	11	43	59
			62	60	44	12	1	3	19	51	
		38	46	54	36	25	17	9	27		
		29	14	22	30	34	49	41	33		
	53	21	5	6	10	42	58	57			
61	45	13	0	2	18	50					
	37	24	16	8	26						
			48	40	32						
				56							

Here I decided not to stick to an 8 × 8 square; this nonsquare set of pixel positions still "tiles" the plane in Escher-like fashion, if we replicate it at 8-pixel intervals. The characters are considered to be 8 pixels wide and 8 pixels tall, as before, but they are no longer confined to an 8 × 8 bounding box. The reference point is the lower left corner of position 24.

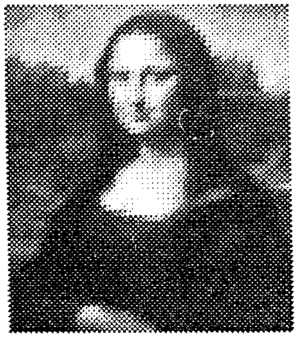
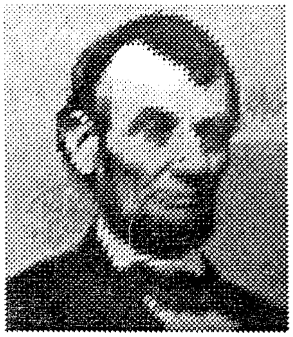
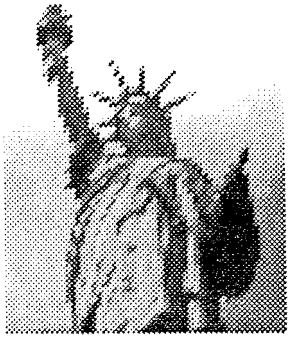
The matrix above is actually better than the one I first came up with, but I've forgotten what that one was. John Hobby took a look at mine and suggested this alternative, because he wanted the pattern of *black* pixels in character k to be essentially the same as the pattern of *white* pixels in character $64-k$. (Commercial halftone schemes start with small black dots on a white background; then the dots grow until they form a checkerboard of black and white; then the white dots begin to shrink into their black background.) The matrix above has this symmetry property, because the sum of the entries in positions (i,j) and $(i,j+4)$ is 63 for all i and j, if you consider "wraparound" by computing indices modulo 8.

John and I used this new ordering of pixel positions to make a font called dot300, analogous to dt300. It has the following gray levels:*



Now we have a pleasantly uniform gradation, except for an inevitable anomaly between characters 62 and 63. The density reaches 50% somewhere around character number 45, and we can compensate for this by preprocessing the data to be printed.

The three images that were displayed with dt300 above look like this when dot300 is used:*

My students were able to use dot300 successfully, so I stopped working on halftones and resumed my normal activities.

However, I realized later that dot300 can easily be improved, because each of its characters is made up of two dots that are about the same size. There's no reason why the dots of a halftone image need to be paired up in such a way. With just a bit more work, we can typeset each dot independently!

Thus, I made a font hf300 with just 33 characters (not 65 as before), using the matrix

			31				
		19	23	27			
		14	7	11	15	17	
	26	10	2	3	5	21	29
30	22	6	0	1	9	25	
	18	12	8	4	13		
			24	20	16		
				28			

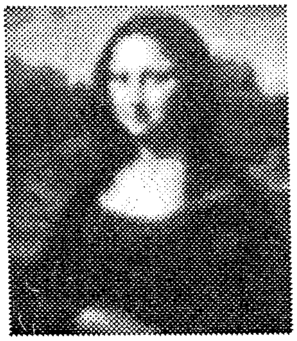
to control the order in which pixels are blackened. (This matrix corresponds to just one of the two dots in the larger matrix above.) The characters are still regarded as 8 pixels wide, but they are now only 4 pixels tall. When a picture is typeset, the odd-numbered rows are to be offset horizontally by 4 pixels.

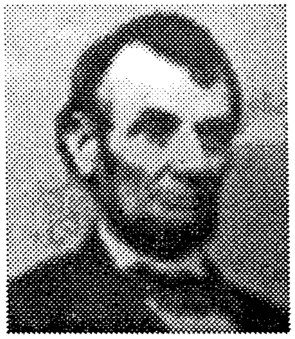
Here is the METAFONT file hf.mf that was used to generate the single-dot font:

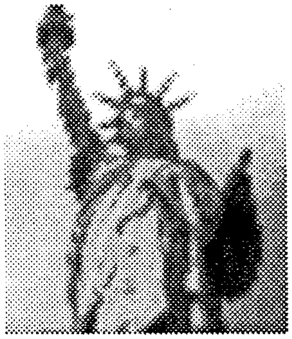
```
% halftone font with 33 levels of gray, characters "0" (white) to "P" (black)
pair p[]; % the pixels in order (first p0 becomes black, then p1, etc.)
p0=(1,1); p4=(2,0); p8=(1,0); p12=(0,0);
p16=(3,-1); p20=(2,-1); p24=(1,-1); p28=(2,-2);
transform r; r=identity rotatedaround ((1.5,1.5),90);
for i=0 step 4 until 28:
 p[i+1]=p[i] transformed r;
 p[i+3]=p[i+1] transformed r;
 p[i+2]=p[i+3] transformed r;
 endfor
w#:=8/pt; % that's 8 pixels
font_quad:=w#; designsize:=8w#;
picture prevchar; prevchar=nullpicture; % the pixels blackened so far
for i=0 upto 32:
 beginchar(i+ASCII"0",w#,.5w#,0); currentpicture:=prevchar;
 if i>0: addto currentpicture also unitpixel shifted p[i-1]; fi
 prevchar:=currentpicture; endchar;
 endfor
```

(There's also a file hf300.mf, analogous to the file dt300.mf above.)

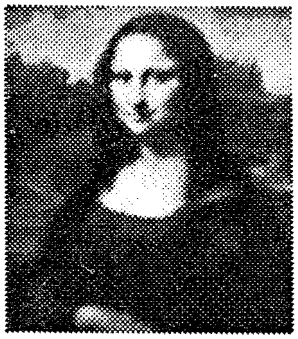
Here's how the three example images look when they're rendered by font hf300:*

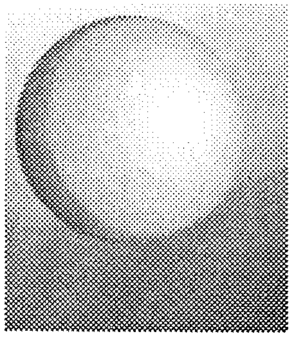






They are somewhat blurry because they were generated second-hand from data intended for square pixels; sharper results are possible if the data is expressly prepared for a 45° grid. For example, here is a sharper Mona Lisa, and an image whose dots were computed directly by mathematical formulas:*



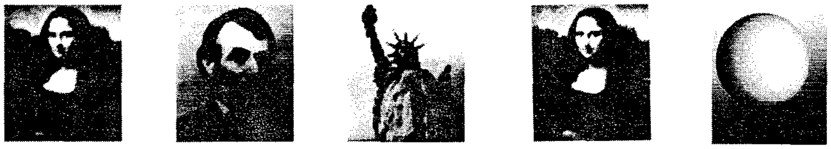


The TeX macros hf65.tex shown above must be replaced by another set hf33.tex when independent dots are used:

```
\font\halftone=hf300  % for halftones on the Imagen 300, each dot independent
\chardef\other=12

\newif\ifshifted
\def\shift{\moveright.5em}
\def\beginhalftone{\vbox\bgroup\offinterlineskip\halftone
 \catcode`\.=\active\shiftedtrue\shift\hbox\bgroup}
{\catcode`\.=\active \gdef.{\egroup
 \ifshifted\shiftedfalse\else\shiftedtrue\shift\fi\hbox\bgroup\ignorespaces}}
\def\endhalftone{\egroup\setbox0=\lastbox\egroup}

% Example of use:
% \beginhalftone
% chars for top halfline of picture. (shifted right 4 pixels)
% chars for second halfline of picture. (not shifted right)
% chars for third halfline of picture. (shifted right 4 pixels)
% ...
% chars for bottom halfline of picture. (possibly shifted right)
% \endhalftone
```

These macros are much simpler than those of hf65, because the 33 ASCII characters "0" to "P" have no special meaning to plain TeX.

We can also create an analogous font hf723 for the high-resolution APS, in which case the pictures come out looking like this:

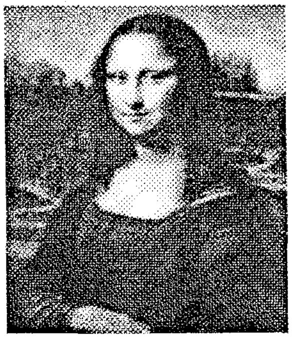


The same TEX macros were used, but font \halftone was defined to be hf723 instead of hf300. Now the pictures are smaller, because the font characters are still 8 pixels wide, and the pixels have gotten smaller. At this resolution the halftones look "real," except that they are too dark. This problem can be fixed by adjusting the densities in a preprocessing program. Also, small deficiencies in the APS's analog-to-digital conversion hardware become apparent when such tiny characters are typeset.

What resolution is needed? It is traditional to measure the quality of a halftone screen by counting the number of dots per inch in the corresponding unrotated grid, and it's easy to do this with a magnifying glass. The photographs in a newspaper like the *International Herald Tribune* use a 72-line screen, rotated 45°; this is approximately the resolution $50\sqrt{2}$ that we would obtain with the hf400 font on a laser printer with 400 dots per inch. (The 300-per-inch font hf300 gives a rotated screen with only $37.5\sqrt{2} \approx 53$ dots per inch.) The photographs on the book jacket of *Computers & Typesetting* have a 133-line screen, again rotated 45°; this is almost identical to the resolution of hf723. But this is not the upper limit: A book that reproduces photographs with exceptionally high quality, such as *Portraits of Success* by Carolyn Caddes (Portola Valley: Tioga Press, 1986), has a screen of about 270 lines per inch, in this case rotated 30°.

Let's turn now to another problem: Suppose we have an image for which we want to obtain the best possible representation on a laser printer of medium resolution, because we will be using that image many times—for example, in a letterhead. In such cases it is clearly desirable to create a special font for that image alone; instead of using a general-purpose font for halftones, we'll want to control every pixel. The desired image can then be typeset from a special-purpose font of "characters" that represent rectangular subsections of the whole.

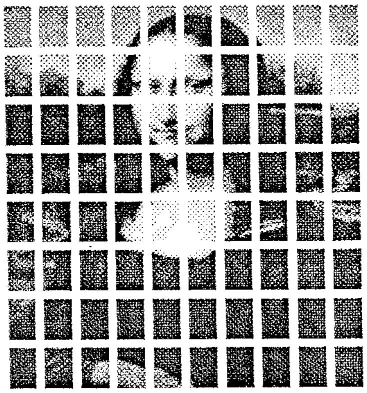
The examples above were produced on an Imagen printer as 64 lines of 55 columns per line, with 8 pixels in each line and each column. To get an equivalent picture with every pixel selected individually, we can make a font that has, say, 80 characters, each 64 pixels tall and 44 pixels wide. By typesetting eight rows of ten characters each, we'll have the desired image. For example, the following picture was done in that way:*

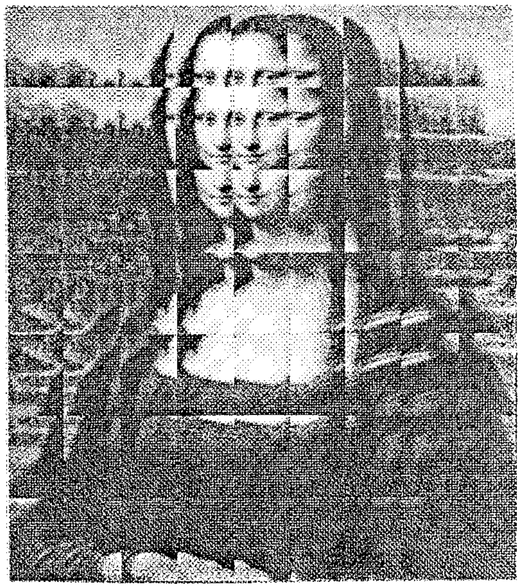


TEX will typeset such an image if we say \monalisa after making the following definitions:

```
\font\mona=mona300[hf,dek]
\newcount\m \newcount\n
\def\monalisa{\vbox{\mona \offinterlineskip \n=0
 \loop \hbox{\m=0 \loop \char\n \global\advance\n by 1
   \advance\m by 1 \ifnum\m<10 \repeat}
  \ifnum\n<80 \repeat}}
```

And once we have the individual pieces, we can combine them to get unusual effects:*





The font mona300 shown above was generated from a file mona.mf that began like this:

```
row(1); cols(1,5,9,13,15,17,21,24,30,32,39,46,56,62,70,
             78,86,95,103,110,118,120,127,135,142,151,159,167,175,183,
             191,198,207,215,223,230,238,246,254,263,271,279,287,295,302,
             311,318,328,334,342,350,358,366,367,375,382,383,390,392,398,
             400,405,408,414,416,421,424,430,432,439);
row(2); cols(4,7,12,20,23,28,30,37,38,40,45,48,53,61,64,
```

... and so on, until 512 rows had been specified. The parameter file mona300.mf was

```
% Mona Lisa for Imagen 300
mode_setup;
if (pixels_per_inch<>300) or (mag<>1):    ...  ⟨error messages as before⟩
else: input picfont
 width:=44; height:=64; m:=8; n:=10; filename:="mona";
 do_it; fi
end.
```

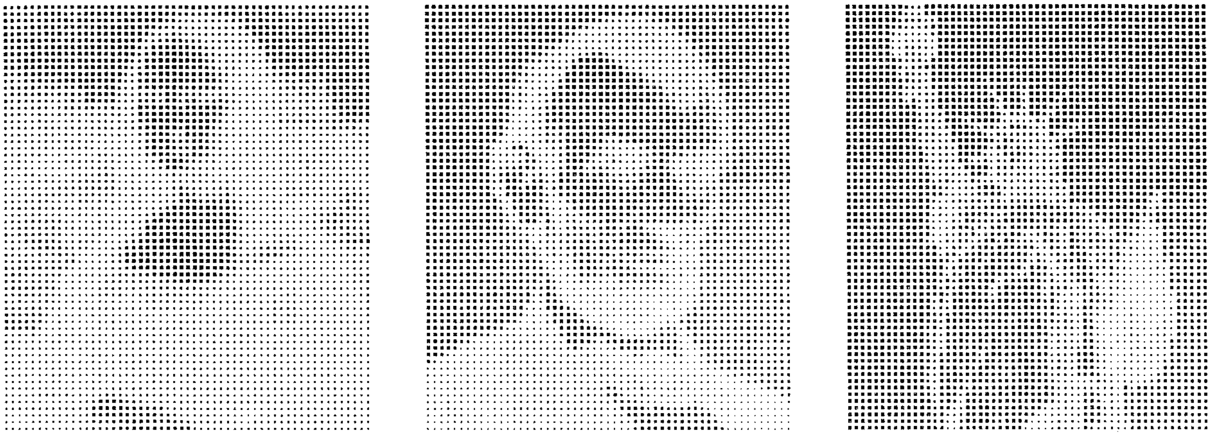
and the driver file picfont.mf was

```
def do_it=
 for j=0 upto n-1: jj:=width*j; jjj:=jj+width; jjjj:=j;
 scantokens("input "&filename); endfor enddef;
string filename;
def row(expr x) =
 cc:=(x-1)div height; rr:=height-1-((x-1)mod height);
 if rr=height-1: beginchar(cc*n+jjjj,width/pt,height/pt,0); fi enddef;
def cols(text t) =
 for tt:=t: exitif tt>=jjj; if tt>=jj:
  addto currentpicture also unitpixel shifted (tt,rr); fi endfor
 if rr=0: xoffset:=-jj; endchar; fi enddef;
```

This is not very efficient, but it's interesting and it seems to work.

Ken Knowlton and Leon Harmon have shown that surprising effects are possible once a picture has been digitized [see *Computer Graphics and Image Processing* **1** (1972), 1–20]. Continuing this tradition, I found that it's fun to combine the TEX macros above with new fonts that frankly acknowledge their digital nature. One needn't always try to compete with commercial halftone screens!

For example, we can use `hf65.tex` with a '`negdot`' font that makes negative images out of square dots:

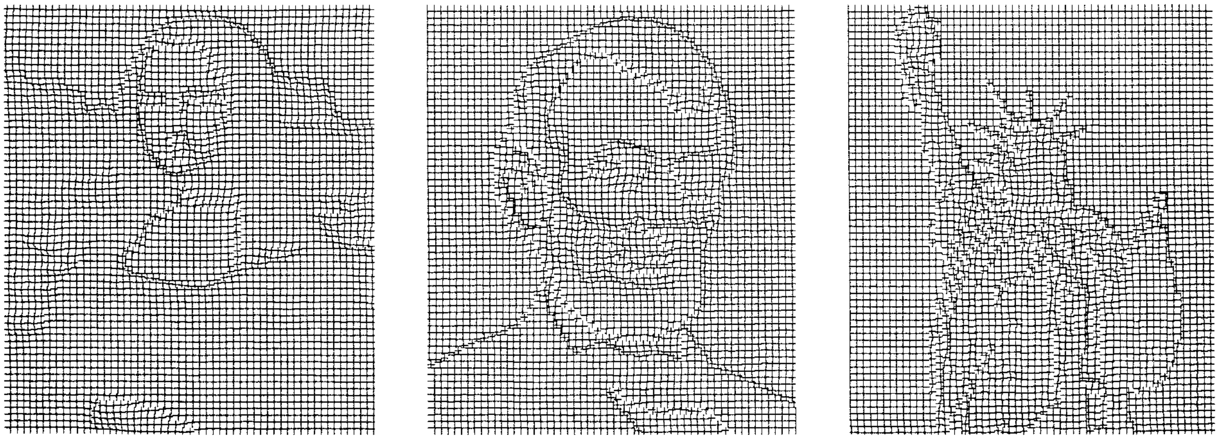


The METAFONT file `negdot.mf` that generated this font is quite simple:

```
% negative pseudo-halftone font with 65 sizes of square dots
mode_setup;
w#:=2.5pt#; font_quad:=w#; designsize:=8w#;
for i=0 upto 64:
 beginchar(i+ASCII"0",w#,w#,0);
 r#:=sqrt(.9w#*(1-i/80)); define_pixels(r);
 fill unitsquare scaled r shifted(.5w,.5h);
 endchar;
 endfor
end.
```

Unlike the previous fonts we have considered, this one is device-independent.

It's even possible to perceive images when each character of the halftone font has exactly the same number of black pixels. Here, for example, is what happens when the three images above are typeset with a font in which each character consists of a vertical line and a horizontal line; the lines move up and to the right as the pixel gets darker, but they retain a uniform thickness. We perceive lighter and darker features only because adjacent lines get closer together or further apart.



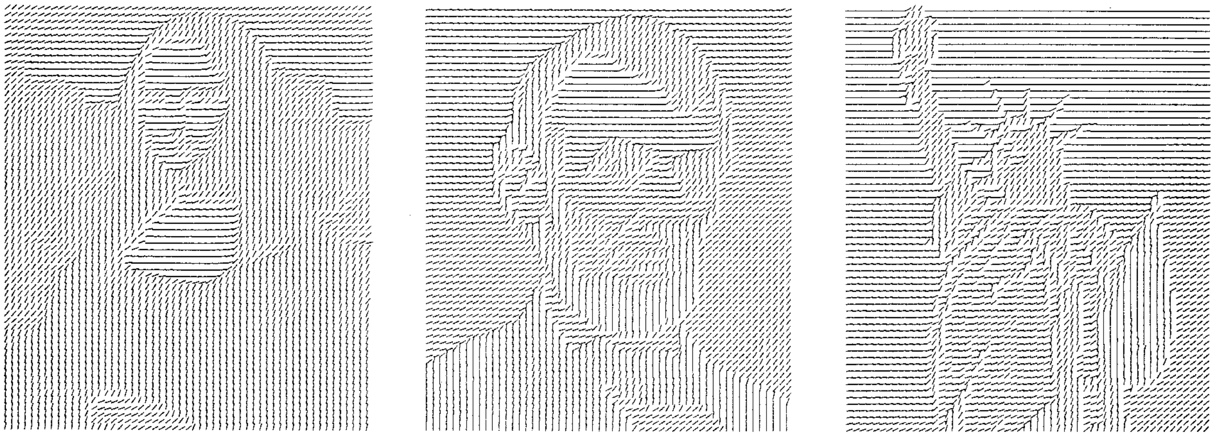
The METAFONT file lines.mf for this device-independent font is:

```
% pseudo-halftone font with 65 lines that move right and up
mode_setup; q:=savepen;
w#:=2.5pt#; font_quad:=w#; designsize:=8w#;
for i=0 upto 64: beginchar(i+ASCII"0",w#,w#,0); pickup q;
 draw (0,h*i/64)--(w,h*i/64); draw(w*i/64,0)--(w*i/64,h); endchar;
 endfor end.
```

Yet another possibility is the font produced by angles.mf; here each character is a single line of radius 2.5 pt that rotates from horizontal to vertical as the density increases:

```
% pseudo-halftone font with 65 radii that move counterclockwise
mode_setup; q:=savepen;
w#:=2.5pt#; font_quad:=w#; designsize:=8w#;
for i=0 upto 64: beginchar(i+ASCII"0",w#,w#,0); pickup q;
 draw ((0,0)--(w,0)) rotated (90*i/64); endchar;
 endfor end.
```

The images are still amazingly easy to identify:



(We can think of a large array of dials whose hands record the local light levels.) It is amusing to view these images by tilting the page up until your eyes are almost parallel to the paper.

As a final example, let's consider a 33-character font that's designed to be used with hf33.tex instead of hf65.tex. Readers who like puzzles are invited to try to guess what this METAFONT code will do, before looking at the image of Mona Lisa that was typeset with the corresponding font. [*Hint:* The name of the METAFONT file is hex.mf.]

```
% pseudo-halftone font with 33 more-or-less hexagonal patterns
mode_setup; q:=savepen;
w#:=7.5pt#; font_quad:=w#; designsize:=8w#;
for i=0 upto 32: beginchar(i+ASCII"0",w#,.5w#,0); pickup q;
 alpha:=.5-i/72; z0=(.5w,.5h);
 z1=alpha[(5/6w,.5h),z0]; z2=alpha[(2/3w,-.5h),z0];
 z0=.5[z2,z5]=.5[z3,z6]=.5[z1,z4]; x2=x6; y5=y6;
 draw z1--z2; draw .5[z1,z2]--z0;
 draw z3--z4; draw .5[z3,z4]--z0;
 draw z5--z6; draw .5[z5,z6]--z0; endchar;
 endfor
end.
```

The answer to this puzzle can be seen in the illustration at the very end of this paper (following the appendices).

Appendix 1: Source data for the examples

The examples in this paper are mostly derived from the basic pixel values shown below. This data uses a convention taken from the book *Digital Image Processing* by Rafael C. Gonzalez and Paul Wintz (Addison-Wesley, 1977): The 32 characters `0123456789ABCDEFGHIJKLMNOPQRSTUV` represent densities from 1.0 down

```
(Lisa)
FFHHIJKKJJJKLKKLLLKLLLLLMLLLLMMMLLLLLLMLKJJJJJJIIIHHGHH.
IIIIJKKKKJJJKLMLMMMLMNNOMLJHGFGIKMMMMMMLKKJJKJJIJIJJHJJ.
JJIIJKLLLKJKMMMMLNNNOOOOHCA988999DJNONONMLKKLLKKKKKKJLK.
IJJJLMMMMLLLMMMNNOOOPPLC9AA88888788EOPQONMMMNNMMMMMMLMM.
KJKLMNNNNMMMMMMNOOPQQKA89CA988877777BORRQONNNONPOOONMNN.
MLLMMMNNNNNNONNNOPQQOA9DHLIFC98777776CQRQPOOOOOPPQQOOMN.
MLLMNMNOOOOOOOOOPQRQFALSTUTRNG97666666GSSSRQQQQQRRPPPNN.
LKMMOOPPPOPPOOPPQRRL9JTVVVVUTNE97666668OSSSRSRSSRQRQQOO.
LLMOPPQQQQQPPQPORSQBAPUVVVVVTQH97665666FSRRRSSSSRRSSRPP.
LKKOPQRSSRQPQRPPRSK8BRUVVVVUTPJC86655558NMPQOQQRRRSSSRR.
IFGNPRQORRRQQRQQSOB6GSUVVVVUTROJE9655556DGLKKOMORSTSSSR.
DEEHLMKJKQRRSSSTSJ77LTUVVVVUTTRPI9555555ACFEIKKIQTTTSSS.
BDEFGGHGGGJOSSTTQD66JSQRUUQJKLLHC855555589CGGFIEMSTTTTS.
BCCFDCDBCDHFQSSSM967BEGFQSECHD989965555579ABEFEBHQTTTTS.
CBCCBCC9BCGEPPKNL866HJIFQRCLODDIMB655555789ABBAAEPTTTSR.
BABA9AA8ABDBIEBCH857QSQPTRHQROOROB554455689ABA9ACMTTSRQ.
99988978989789A8A856PUVUURNTUVUQI85555556999A9889FQQQPP.
87778988777777777666LUVUURNTUUTLC65555556CCEDDDDDEHKKJI.
76677998877766665666GTUSUOJRUTOF9554444556678899BDEFEBA.
77666678777766655555ANSSLA9OTQJC855544555666677678889A9.
67666667667765555558JOSQDEJKMHB85544444567766666667899.
77666766566665555556HOMOHCEMOHA755444445677777777778987.
766666555566656655559LSPHFHKIC8655444445687777777777667.
66665555567777786555569NUTOJFB7655544444568789866776777.
656655555567777965555559OQKD876555544444568889877777777.
65566555556778AC845445557A87666555544445558778766667677.
56566777789ACBBB944445555786666667654444557778866666667.
5556788899BBBBBA9645444559GA8778A96444444457789887777888.
6678899999ABBB998644445558NJEBBBED8444444467999ABA9CCDA.
7789AB9777799897654444556AMOKHHIIHB544454445555876DHDB9.
779899877956666554444458FOPQOOOPNMF644554445555666BHC9A.
76789878886665765444446DKQTTSRRSSRPJ85554444567AEFGGFA99.
668999ABA766657654445FPTUUUUTTUTSPI655544444457FJHCDCBA.
7799A9BC8766757644448OTUUUUUUUUTSOJ754544444456ADFGFFEF.
979CDCDB977777754456FSUUVVUUUUUTSOG6545445654457ABEHCHF.
878DGGEDB8798644459CLUUVVVUUUUUTROF655556886544579DHJKN.
76889DHFC988644447HMTUVVVVVVVUUTSOE7569CEED964444569BBC.
978889BECDB744445BPTUUVUUUVVVUUURNC879CHGB754444456777A.
879A99BBBEC444446BQUVVUUUVVVVVUUSNF9ADA654444444444457778.
99BCCDEFFB54444578FQUUTUUVVVVVUURIC86444444444444446888.
BDCBBBBA84444446899AEJLSUVVVVUTPEA74444444444444444599A.
BCB9BAA854444457987898ABEHKMNNIB87544444544444444445689.
ACDCDDEA44444467776779A9AA99998856433445544444444444578.
8ABBBBB844444466665568989987677544434445556655544444589.
889889964333446555555788766666544444444467765555444457B.
99999974443334554555566665565544444444445677655444454457.
CDCCDC64333334544544555555555544444444345676654444455457.
EFFEFC53333334444444444445554444443343456765433444455457.
CCCDC8333333334444444444444444434334555544433444455556.
77886433333333344443344444444444443333444443333444445545.
6655333333333333333334444444444444333344444333333344444445.
664333333433333333333334444444444333333443333333444444444.
5433434444433333333333344444444444333333333333333344444444.
43344444444333333333333334444444433333333333333333344444444.
3333444333333333333333344334444433333333333333344444444444.
33334433333333333333333333344333333333333333333444444444.
3334555444443333333333333333333333333333333333333444444.
33345544444333333333333333333333333333333333333334444.
3334457646555459975333333333333333333333333333333334444.
3334555454566BLNLJFB744333333333333333333333333334444.
33335445344448IMNMLJIFED943333333333332233333333333333334.
3223334333348HJKJKKJHHHFA6433333333322233333333333333334.
3222333333346DEFFHIIIIIHGFB74333333322333333333333333334.
3222232222234788ADFHIIIJHHIG74334433345444333333333334.
```

```
(Lincoln)
RRRQRRQRQQQQRQQPRQPQQPPQPQQQ
RRRRRQQRRSRQRRQQQRQQQQQQQQRR
RSRRSRRRRSRQPRQQQQQQQQQRPNHC
QRRRSSSSSSSQQQRQRRRRQQOMD555
QQRSSSSSSTRRSRSRSSTSQMD55555
RSSSSSSSTTSUTTTSSSSSKE555555
RRRSSRSSTSTTSUSTSPKD76454555
RRRSSSSSSRSSSTRTNIA66549EKK9
SSRRSSSSQRRRRSSNDA7654DVVVVT
SSRRRRRRRRQRRRQ9846977VVVVVV
TSRRSRRRQRRSRSI953454RVVVVVV
SSSRRRRSRSRSRQ9A5443GVVVVVVV
SSSSSRRRRRRQQGE52435SVVVVVVV
TSSSRQRRQQQRPAE4285KUVVVVVVV
SRRRQRQRQRQRN7A2259PSUUUUVVV
SQRQQQQQQQRRN49238GOQTTTTUUU
RQQQPQQQQQRPK4213ANPOUUTTTTT
RQQQQQPPQQPQK22238NPPTTTTTSS
RPPQRQPPQPPQ911357IJKSSTSSST
QQQQQRPPPQPL201246HFHQRSSRRR
RRQPPQPQPPM6122247CBFOQRQQLB
RQQPPPPQPQ424K074357EOQPRNDA
RQPPPPPPPL3753906335AOQQODD6
RPPOOOOPN33659JK333ANRQI74B
QQPPPOPOON5B9FLME633FOPL89EB
RQQPPOOOOM7ILLL69C58MQNKFJMI
QQPOOONNMLADPPH5NN79OONONNNJ
PPOOONNMMN53QQ9PQO49NNNOPONM
PPOONMNMM072KOLPOL73GMMNOONN
OONMNMMMLNN62MOOLJ824FKMNNNN
ONNMMMMLMLLB2NOMMGB125GKLMNN
NNNMMMMMLMLJ780NMHG2287CGIJM
NNMMMMNMLLLKC7HNL2K31AB6ADFK
NMMMLLMLKKLKFF8621G659D9A9CG
NMLMLKKKKKKIGB612BB9CFEF9GH
NMMLLKJKKJJJJHD24B79AJGHBHA
MMLKKKKJJIJKLJIE93A67CJIIFIJ
MLKKKKJKKJJKJKJIH56A78GJKFIC
LLKKKJKJKJJKJJJII94485GIKEFB
MLKLKKKJJKJKJJKJIF71369JKHCH
MLKLJKLKJJJKIIIIIGD1244HEDBB
MLLLLKKJKKJJJIJII7H622369332
MLKLKKLKJJJKKJJJFCLB12223223
LLLKKKLKJKKJKJJK57NF42221111
LLLKKJKJJKKJJJJH23NMB2111111
LLLKKJKKJKJKKKK222KNMD211111
LLLLLKKJJJJKKKC2223NMMI41011
LLMLKKKKKJKLJF22223HMMML8110
LLLLLLLKLKJC32222232NMMMMD62
LLLKLLLJFA3133222223DNMMMLLK
LLMMMJA22222333333232MNMH85F
MMNLD1112122333333436J73322
MMG3111111223334343333234432
G511111112223444543443335543
1111112112224434543443333343
1211111111222344455444334333
1112111222222444554443343444
11221112222234554444333436J
122221212212244544444333433D
1223222111122444434433333434
1223222111112444433333333333
2023232110112222222323222332
1011112100011111121222122222
0001011000001001111211112112
```

to 0.0 (i.e., '0' is black and 'V' is white). The Lisa data was digitized by a TV camera in Stanford's robotics lab. The Lincoln and Liberty data come mostly from Appendix B in the Gonzales-Wintz book, although I decided to change several dozen of the pixel values found there.

```
QQRRRQPPRQQQQQQQQPPPPOPOOMM.
QKJONMPQRQQQQQQQPQPPPPOOOON.
966989EKPQQQQQRQQQPPQPOONNN.
66689877AMPRQQRRQPPPPOONMNM.
6669999898HORQRQQPPQOOONOMM.
66678898878BQRRRQQQPOPOOPNM.
666769998887GRRRQPOOPPOOONM.
5667667788778IRQPPOPPPOONOL.
F6676777677779IQQOOPONOOOOM.
VKA767778777679PQPPOOMMMMML.
VVTKC7676667677IPPOOPNMMMNL.
VVVUQHD77668787AOPOPNONNNML.
VVVVVURG77787F87KONNNNNLMML.
VVVVVVVSAE777EE6AOONNNMMMMM.
VVVVVVUUHD968EK77KONMMLLLLL.
UVUVUUTTTG89GLKB6IMMMLLLLLK.
TUUUUUTTSSPKNNLJ6FMLKLJLLKK.
TTTTSSSSSRRQOOLH89NLKKJKKKL.
TTTTSSRRRSTRPPLH97MKJKIKKKL.
QTTTTSRRRSTRQOMH77KJKJJJKJJ.
9EHORRQRRTSRSSOG67LJJKJJJJL.
6555BKORPSPONJIE76IKJKJKJKK.
55456BKQLMG89B8769IKKLKKLKL.
F6467BMRI76566566CJKMLLKKLJ.
74458LOQI55775559BKLLLLLKLK.
B97EINLQR9897555DLLLLLLLLLK.
HECEOOOQQMBCA767LLKKKKKKKKK.
MJFKNOOOQOAEBBA9JJJIJIJJJJI.
LLMNNNMNOOH9A8DAIIIGIIIJJIH.
MMNNMLMMNNJIJIG9GGGFGGGHHHH.
MNNKJLIMMNLILJF7FGFFFGGHHHH.
MMKDKNMLMMKHKF89GFFEFGGGHGH.
LICJGGIFFGD8E83EGFFEFFEFFGG.
H8KMD332233263BGFEFEFFEEFFF.
FJLLKHB3213632FGFEFEEEEEEEE.
HKKKKKJ9514827FGFEEEFEEEEEE.
FCFGHHHHB4342EFFEFFEDEEEEDE.
6FHGD97922122CFFEEEEDDDDDDE.
HFADGGHF83142BEEEFFEEDDDCDD.
JJG8333211433EEEEFEEDDEDCCC.
GJIIHHFB65622CFFEFEEEDDDCCC.
4B8EFFEA81211BEEEEEEEEDCCCC.
3574685341101BEEEEEDDDDCCCB.
2422231111111EEEEEEDDDDDCBB.
1121110211003EFEEDDDDDCCCCC.
11111000101BFFEEEDDDDCCCBB.
1111100002CGFFEEEEEDDCCCCC.
0111100052FGFFEEEEDEDCCCCCC.
10000016EA4GFFEEEDDDDCCCBBB.
D74325BHGG08GFEFEDEDDDCCCCC.
LMLKJKJIIH20CFEEEEDDDDAACCB.
8IIHGDEHII501EFEEEDDDB9BBBB.
222972116F5004FFEEEEDBBABAB.
22111111011001BEFFFDDD9BBBB.
3221110000000113AEEFECBCBBB.
5543221111111120017DECCCCCD.
355642422332211100016BCEDDD.
B5569434344322111000004BDDD.
L754642322221111110000004BD.
GFD973322112410111000000007.
4HLKH8IF969EF82011000000000.
33IKJJKKJJIJJJG201000000000.
234JJJIIIIIHHHHF10000000000.
121BHHHHGGGGGGGGB0000000000.
```

```
(Liberty)
VVVVVVVVVJABVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV.
VVVVVVVVPB76OVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV.
VVVVVVVVGHEDAEVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV.
VVVVVVVKL97ADTVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV.
VVVVVVVC55656FVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV.
VVVVVVV4457765VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV.
VVVVVVV35CB974VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV.
VVVVVVV44JIB55VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV.
VVVVVVVQHEDG8VVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV.
VVVVVVVVE7CHSVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV.
VVVVVVVV9LDDAVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV.
VVVVVVVVNMCCAVVVVVVVVHVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVVV.
VVVVVVVVU99ABTVVVVVVVVNVVVVMVVVVVVVVVVVVVVVVVVVVVVVVVVV.
VVVVVVVVM86ADCVVVVVVVV9VVVVBVVVVVIVVVVVVVVVVVVVVVVVVVVV.
VVVVVVVVV5L8DDVVVVVVVVVAVVV9VVVVGVVVVVVVVVVVVVVVVVVVVVV.
VVVVVVVVVQV9EEVVVVVVVVV8TVV9VVVJLVVVVVVVVVVVVVVVVVVVVUU.
VVVVVVVVVVVBDEFVVVVVVVVS8MJ9VVNBVVVVVVVVVVVVVVVVVVUUUUU.
VVVVVVVVVVVCEFBVVQHQVVVK78978M8VVVVUQOVVVVVVVVVVUVUUUTU.
VVVVVVVVVVVDEDAIVVVG9LN7BRRL97AVVSLCRVVVVVVVVVVUTUUUTTU.
VVVVVVVVVVVDCDBBVVVVVC8JBDCCDA7QEAIVVVVVVVVVVUUUUTTTSTT.
VVVVVVVVVVVD9DABMVVVVVNAPHEB9DA9ETVVVVVVVVVUUUUUTSSSSST.
VVVVVVVVVVVI9ABA8VVVVU8LQAHDCBD8KVVUVUVUUUUUUUTTSSRSSRS.
VVVVVVVVVVUVAB9B8VVTPA59KNNFCBA99VUUTTTTTTTTTTTTSSSSSRS.
VVVVUUVVVUUVA79BBS08BG5EA0JDBABA8UTTSSTSSSTSSSSSRRRRRRR.
VVUUUUUUUUUUL9AA98PUUU5HFHFDBABBBTSSRRRSSSSSSSSRRRRRQQR.
UUUUUTUTTTTTA8C977USJ5CEBCA997CCTSRRQRQRRRQRRRRQQPPPQQ.
UTUTSTTTTTTTSA9J866JB977AAB9AA7E7LRQPQQPQPPPQQQQPQOOOPP.
TTTTSTSSSSRRSOEB5796NSQE899BAB8B9AQPPPOPOOPPPPQOOOOONON.
TTTSSSSSSSRSSQG6767EMPRP49ADAA87B7QQPPOOOOPOPPK3NOOONNM.
TTSSSSRSRRRRRA55BC7DJ8JC5EEGDDB8B9OPPP0000000053ONMMMLM.
SSRRSRRRRRRRS7645AFLDMMMIHMPMOOLHAEAELOONNONOL4ANMMLMLM.
SSSSRRRRRRRRQ5645EKONMDBEKPMNNHHMLKGF6KONNNOIG9MMMLMLKK.
SSRRSRSRQRRRI444AAOLCCDIKNOKNCKOHLJIFE3LNNNMG49LMMLKKKJ.
RRRRRRRRQRQRP644DNLBJ7IK9OLMJKIKBAGLDFD3MNMDD515MLLLKKJ.
QRQQQPQQRRRRQ3367IDL6JIIGOL78FLF98PEC9I68NMF6211LLKKJJI.
RQRQQQQQQQRB44F8HJ6EOOBONAE75HIA7OJEGJDGJJ53232LKKKIII.
RQRRQQQRRRRQQQ36HH6HOODPOLILG57HF6OM9FGAJC881242KLKKJII.
QRQQQQRRRRRRQQ4HI5LLPLNPNKMM5E3BF8KOL6BBFL312432IKJJIIH.
RRQQQQRRRQRRQQMI3MHOODSPMKLJ6L62FIIPL4DCDF311113AJJIIHI.
QRQQQQQRQRRQRQQ7FKOQHQSOMLK6MKG24CDNLGEAI75011232JJIIII.
RQQPQQQQQRRQRQH5MEPPMQQMJDBEOLE93BJLLJ6BC65000111JJIHHH.
QQQPQQPQPQRQQQAMEOPIQOM8GLKONM9G4AAINK31861000001KJIIII.
RQQQQQQQQQQRRP9J9POOPO4JPMLPNM7JBI55LIA2650000000IJIIHI.
RRRQQQQRRQQQQKOFLQNSMM8OPLMPMM8KB386FGC56200000011JIHIH.
RRQRQQQQQRQQPBM7OOLRMMGPOMMONLCL6631JG9960000000010JIIIH.
RRQRQQQQQRQQQ8F7OOQQNNNOOLNNNLGM8573LC56400100001 1IIIIH.
RSQRQQRRRRRQOJ9JNNSMMMMMKLNNMLIL8C75ND600001000022IIHHG.
RRQQQQRRRRRQMKANOLRMMLLLHFMNLLGH7E74NF000004000042IIHGG.
RRRRRQQQRQQQGK7NNMQLMLNKJLOOLOI96D65OH200014000142JIHGG.
SRRRRSRRQQQQDC5LNNMKMLNLMLOOLOK67864LE400013101132IIHGH.
SRRQRRRRRQQQIB5MNNHNNHPKLLPNLNL47874LG40011230133EHHGGH.
SRRRQRRRQQRQMB8MNPJNNCPLLKQNLNL48977NK41100245153IIHHHG.
SSRRRRQRRRQQJ9ENPQLO7LOMKKPNMNL5887KNK42000302221IIHHGG.
TSSRRRRQQQQQI9KKROLMCOMIJLPNNMJ6C88KNL74001111227IHHGGG.
TSSRRRQQQQQQI8OHSINCKONLIMNPNLK6C86LOL75002122 11IIHHGGG.
SSRRQQQQQQQQF8NMRHNMNILIFKNPNLM9764IJLA51002201AJIHHHGG.
SSRRQQQQQPQQD7NNOKDNLLNLLMQOMLNC654246C6313123BJIHHGGG.
SRRQQQPQQPPPH5NPMM6NJMNLLNOPMJOH6663984662L5GIJIIHHHHGF.
RRQRPPPPPPOPN7NOK7HNLNNPNPNPLMOJ6653LKA472KJJJIIHHGGG.
SRRRQQPOPPOPPBMOD5PNNNMLNNNMMNKC6763LLF262JJIIIIHHHGHGG.
QQQQPQQPPPPPFKK57QMPNLNNMOLNH324673LOH573JJIIIHHGHHGGF.
QPOPOPONOOOOOHK73AQMPLMJKKLMH2334653KNH643FIIHHGGGGGFFF.
NNNNNNNMMMMLMEJ52DQLOLIENLLH12126462KLH6546HHFFFFFEFFFE.
LLLLLLKJKKKKKHH51DMJLJCKMKG101036341G734622FFDEDDDDDDDD.
```

Appendix 2: 65-level data for typesetting

The 32-level data in Appendix 1 was converted to 65-level data by a computer program similar to (but simpler than) the program in Appendix 3. In this case the representation is different: ‘0’ means white and ‘p’ means black. The 65 codes follow standard ASCII order: `0123456789:;<=>?@ABCDEFGHIJKLMNOPQRSTUV`

~~~
(Lisa)
MMIHFDBADDDA?BA???B?????=????===??????=?BDDDDDDFFFIHKIH.
FFFFDBABADDDB?=?===?=;;8=?DIKMKFB======?BADDADDFDFDDHDD.
DDFFDB???BDA====?;:;8888ITY[]^[[[RD;8;8;=?BA??BABBABD?B.
FDDD?====???===;:89877?T[YY]^]]^`]]P8768:===;;======?==.
BDA?=;:;:======:98766AY^[TY[]^]````_W85468;:;8:7888;=:;.
=??===;:;;:;8;:;87668Y[RH?FMT[^_`_``bT656788889776688=;.
=?@=:=;8888888887646MY?33125;K[`bbbbbK33456666645777:;.
?B==887778779877655?[D2000012;0[`bbbbb]9433535335646688.
??=8776666677678536VY710000026I[`bbdbbbM355443435534477.
?BA876443567647753B]V510000137DT]bbeded^;=7686654534355.
FMK;75685455656639VbK3100001258D0[bdeddbRK?AB8=85324344.
RPOI?=ADA65533424C``?31000012257F[eddedeXTMPFABF6232434.
VROMKKHKKKC843226RbbD365116DB??IT^dededd^[TKKMFO=422224.
WTTMRTRVTRIM6443=[b`VPKM640TIR[][[bdeded`[YVPMPVI623233.
TVTTWTT[WTKO77A;?^bbIDFM64T?8RRF=Wbddded`][YVVYYO722245.
WYWY[YY]YVRVFOWTH]e_636725I658858Vedggddb^[YWY[YT=23356.
[[[]][`][^[`^[Y]Y^db710115:21016F]edededb[[[Y[]][M66677.
]```][^]```_`````bbb?10115:3112?TbdeddedbTTPQRRRRPHAADF.
`bb``[[^]`_`bbbbdbbbK21418D5129M[edggggdebb`]^[[VRPMPVY.
``bbbb`]````bbbdededY;43@Y[826DT^dedgfdedbbbb``b`]]^[Y[.
b`bbbbb`bb`_bdeddede]D846QPDB=IV]edggggfeb``bbbbbbb`][[.
`_bbb`bbdbbbbededdddbI8=8HTO=8HY`ddgfgggdb``````````][^`.
`bbbbbdedebbbdbbedeed[?37IMIAFT^bdeggfgfeb]`_`_`_```bb`.
bbbbededdb`````]bdddeb[;129DMW`bdedgfgggdb]`][^bb`_b``_.
bdbbdededeb`_``[bededde[96AR]`bdeddgggfgdb^]^[]`````_``.
bedbbdddedb``]YT]gdggddd`Y]`bbbddedgfggded]``]`bbbb`b``.
dbebb````][YTVWV[ggggdedd`]bbbbbb`bdggggde```]^bbbbbbb`.
eddb`]^][[WVWVWY[bgdgggde[KY]``^Y[bggfgfgd`_][]]````]]].
bb`]^[[[[[YVVW[[]bgfgfded];DPVVVPR]gggggggb`[[[YVY[TTRY.
``][YV[````[[][`bdggggddbY=8AIIFFHWdgfgdfggdede]`bRHRV[.
``[^[[^`_[dbbbbdeggfggd^M8768887;=MbggddggfeddebbbVIT[Y.
`b`][]`]^]bbbe`bdgfggbRA6224553447D]ddeggfgdb`YOMKKMY[[.
bb][[[YVY`bbbd`bdgggdM721111221247Fbeddgggggggd`MDITRTWY.
``[[Y[WT]`bb`d`bggfg]8212111112248D`dgdgfgfggdbYRMKMMOM.
[`[TRTRV[``````dggdbM4110012111239Kbdgdggdbdggd`YVOITIM.
]`]RKKORW]`[]bggfd[T?1100011111358Mbdeddb^]bdggd`[RHDA;.
`b]^[RIMT[]^bgfgg`I=210000000112480`db[TPOR[bggfgdb[WVT.
[`]]][VPTRV`gfggdV721201110001215:T^`[TIKV`egfgggdb```Y.
]`[Y[[WVVPTggggfbW610011100000113;M[YRYbdggfggfgggd``_^.
[[WTTROMMWdgfgfe`]M61131100000115FT]bgfgggfggggfgggb]^].
WRTVWVWY]ggggggb][[YPD?4100001370Y`gggggfgggfgfggfgd[[Y.
VTW[VYY]efgfgfd`[^`][]YVPIB=;;FV^`dgfgfgdgfgggggfggdb^[.
YTRTRROYggggggb``_b``[Y[YX[[[[]^dbgiiggdeggfgfgfggggd`].
]YVWVWV^fgfgfgbbbbdeb][][[^`b``dggfigfgdddbbdedggfgfe][.
^][]^[[bgiiiggbdededd`^]`bbbbbdggfggggggb``beddefgggd`V.
[[[[[[`gfgiiifedgddedbbbbdebdegfgggfgfgdb``bdeggggdggd`.
TRTTRTbgiiiiigdggdggdedededdefgggfgggigdb`bbdgfgfgddgd`.
PMMPMTdiiiiiiggfgggfgggfgdedggfgggiifigdb`bdgiigggdegd`.
TTTRT^iiiiiiiiggfggggfggggfggggfgifiigdeddgggiigfgdddeb.
``]^bgiiiiiiiigfggiifggfgggfgfggfiiiigggggiiiigfgggdefd.
bbddiiiiiiiiiiiiiiiiigggfgggggfggiiiigfgfiiiiiiggfggggd.
bbgiiiiiigiiiiiiiiiifggfgfgfgiiiiiiggiiiiiiiiigfgggfggg.
dgiigigggfgiiiiiiiiiigggggggggiiiiiiiiiiiiiiiiiggfggfgf.
giiggfgfggiiiiiiiiiiiiifgfgfgfiiiiiiiiiiiiiiiiigfggfggg.
iiiigggiiiiiiiiiiiiiiggiiggggiiiiiiiiiiiiiiiigggfgggggfg.
iiiigfiiiiiiiiiiiiiiiiiiifgiiiiiiiiiiiiiiiiiiigggfgfggg.
iiigdedggggiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiigggfgf.
iiigddggfgfiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiigggg.
iiiggd`bgbdedgd[[`diiiiiiiiiiiiiiiiiiiiiiiiiiiiiiiifgfg.
iiigddefdgdbbW?;?DMW`ggiiiiiiiiiiiiiiiiiiiiiiiiiiiigggg.
iiiidggdiggg]F=;=?DFMOR[giiiiiiiiiiilkiiiiiiiiiiiiiiiif.
ikliiigiiiig]IDADBADIHIMYbgiiiiiiiiikkliiiiiiiiiiiiiiig.
ikkkiiiiiiigbROMMIFFFFFIKMV`giiiiiilkiiiiiiiiiiiiiiiiig.
ilklkiklkkkig`]^YRMIFFFDHIFK`giiggiiigdgggiiiiiiiiiiiig.
~~~

~~~
(Lincoln)
55465564666665667567667767666
4554556553555466556656666645
5355355453567566665666647:IT
654534343436655555455658=Rddd
665343434254453534236=Rdeded
43434343224122243434APdedded
554435342422414247AR`bgdgded
5453434344434252:FYbbdg[PAB[
345534346555443;RY`bdgR00002
435455454555546[]gb[``000000
24453554655353G[eigdg5000000
34355453535356[YdgfiK0000000
4343445554566JPdkgid30000000
2434465466647YOgk^dB10000000
355465656465:`Ykke[741111000
464666566645:g[li]K862323111
466676665657BfkniY;781122232
566656776676Alkki];772323243
577656776776[nnie`FDA4324342
65665577767?kpmkgbHMI6534455
5567767677<bnklkg`TWM86566?V
4667777576fkgA8`gid`08674;RY
567777777?i`di[8biidY8668RQb
577888887;iibe[DBiiiY:56F`gV
657778788;dW[M?=ObiiM87?][OW
566778898=`F???b[Te]=6:BMD=F
667888;:=?YR77Id;;`[88;8;:;D
77888;;==;di66[768g[;;:878;=
7788;=:==8`kA8?78¿iK==;88;:
88;=:===?;:bl=88?D]kgMA=;;:;
8;:====?=??Wk;8==KWnkdKB?=;:
;:;=====?=?D`]8;=HKkk^`TKFD=
;:====:=???BT`I:?lAinYVbYRMA
;===??=?BB?BMM]blmKbd[R[Y[TK
:=?=?BBABABAFKWbnkWV[TMPM[KH
;==??BDABDDDDDHRkgV`[YDKHWHY
==?BABADDFDA?DFP[iYb`TDFFMFD
=?BABADBADDBDBDFHebY`]KDBMFT
??BABDADBDDADDDFF[gf^dKFAPMW
=?B?BABDDADBDDADFM`nib[DBHTH
=?B?DB?ADDDAFFFFFKRmkggHPRWV
=????ABDABDDDFDFG`Hblkib[iik
=?B?BB?BDDDABDDDMT?Wmlkkikli
???BAB?ADABDADDAe`:Mgklknnmn
?@?BADBDDABDDDDHki;=Vlmnnnnn
???BADABDBDABABklkB:=Rknmnmn
?????BADDDDABATklki;==Fgnpnm
?@=?BBABADA?DMlkkliH===?]nmp
???????B?BDTikklkkik;====Rbl
??@B???DMYiniilklkliR:===??A
??===DYkkkkkiiiiiikik=;=H^dM
==;?RnnnknlkiiiiiiigibD`iikl
==Kimnmnnnkliiigigiiiikiggik
Kdnnnnnmnkkkigggdgiggiiiddgi
nnmnmnknnklkggifefigfiiiiigi
nknnmnnmnkklifggdegfgiigiiii
nmnknmnkklkkkgggddgggiigiggg
nnkknnnnkklklifdegfgfiiigibD
nkklknknklmlkggdgggggiiigiiR
nklikkknnnnkkgggfifgiiiiigig
nkkiklknmnmnkgfggiiiiiiiiii
kplikiknnpnnkklkkklikiklkiik
npnnnnknpppmnnnnnknklknkklkl
pppmpnnpppppnppmnnmlmnnnknnk
~~~

WXYZ[\]^_‘abcdefghijklmnop. The density of character k was assumed to be approximately $k/36$ for $0 \le k \le 8$, and approximately $k/72 + 1/9$ for $8 \le k \le 64$.

```
66545677566666666677778788==.
6AD8;=76466566567677778988;.
[bb[][0B7666665665776788;;:.
bbb][^‘_Y<7466456777788;=:=.
bbb[[[[][^H946556776888;8==.
bbb‘^][^]‘]W6545666787887;=.
bbb‘b[[[^]^_K554678877898:=.
ebb‘bb‘‘]^‘‘]F5677877788;8?.
Mbb‘b‘‘‘b_‘‘‘[F668878;8889=.
0BY‘b‘_‘]‘‘_b‘[767788=====?.
002BT‘b‘bbb‘b‘_F77887;===:?.
00016HR‘‘bb]‘]‘Y8787;8:;:=?.
0000015J‘‘‘]‘M]‘A8;:;:;?==?.
00000003Y0‘‘‘0PbY88;;:=====.
00000011IR[b]PA‘‘B8;==?????.
101011322K][K?BVbF===??@??B.
31111123347B:;?DbM=?A?D??BA.
22324434354688?I][;?BBDBAB?.
32233355533577?I[‘=ADAFBAB?.
62223354532568=H‘‘ADBDDDBDD.
[PI855654335348Kb‘?DDADDDD?.
bddeVA857378;CFP_bFBDBDADAB.
degdbWA6?=K][W]‘b[FBA?AB?B?.
Mbgb‘V=5F‘bebbdbbTDA=??BA?D.
‘gfd^?86Fdd‘‘ded[WB??@??B?B.
V[‘PF;?65[][‘ddeR?????????B.
I0T088866=VTY‘b‘??BBBBBBBAB.
=DMB;88868YPVWY[DDDFDFDDDDF.
??=;:;=:88H[Y]RYFFFKFFFDDFH.
==;:=?==;;DFDFK[KKKMKKKHHII.
=;:BD?F==:?F?DM‘MKMMMKKIHIH.
==ARA;=?==BIBM][KMMPMKKKHKI.
?FTDKKFMMKR]0^iPKMMOMMOMMKK.
I]B=RiilkiikbiVKMOMPMMPPMMM.
MD??BHWiknibilMKMPMPOPOOPOP.
IABABAD[dng]l_MKMOPOMPOPOPO.
MTMKIHIHWgigkPMMPMMPROPOPRO.
bMIKR[‘[kknkkTMMPOPORRRRRRO.
IMYRKKHM^ingkWPOPMMPORRQTRR.
DDK]iiiknmgiiOPOOMPORRORTTT.
KDFFIHMWbdbklTMMPMOPORRRTTT.
gV^OMMPY]nknnVPOPOPOPORTTTT.
id‘gb^dignnpmWOPOPORRRRTTTW.
kgkkkinnmnmnnOPOPOPRRRRRTWV.
nnknnnpknnppiOMPORRQRQTTTTT.
mnnnmnpppmpnVMMPOPRRRRTTTWV.
nmnmnnppppkTKMMPOOOPRRTTTTT.
pnnnmpppdlMKMMPOPPRORTTTTTT.
npppppnbOYgKMMPOPQRRRTTTWWV.
R‘fikdWHKKp]KMPMORORRRTTTTT.
?=?BDBDFFIkpTMOPPORRRRYYTTW.
]FFIKQPHFFepnPMOPORRQV[VWVV.
lkl[‘knnbMdppgMMPOPORWVYVYW.
klmnmnnmpnnppmWOMMMRRR[WVWV.
ikknnmpppppppnniYPOMOTVTWVW.
degiklnnnnmnnmkppn‘RPTTTTTR.
iddbgkgkkiikknnnpppmbWTPRRR.
Vedb[gigiggiklmnmpppppgVRRR.
¿dgbgkikkklnmnnnnppppppgVR.
KMR[‘iiklnnkgnpmnmpppppppp‘.
fI?BH]FM[b[OM]kpnnppppppppp.
iiFBDDBBDDFDDDKkpnppppppppp.
kigDDDFFFFFIHIHMnppppppppp.
nknVHIHIKKKKKKKKVpppppppppp.
```

```
(Liberty)
000000000DYV0000000000000000000000000000000000000000000.
000000007V‘b8000000000000000000000000000000000000000000.
0000000KIORYP000000000000000000000000000000000000000000.
0000000B?[‘YR200000000000000000000000000000000000000000.
0000000TddbdbM00000000000000000000000000000000000000000.
0000000ggd‘‘be00000000000000000000000000000000000000000.
0000000idTW[‘f00000000000000000000000000000000000000000.
0000000ggDFVde00000000000000000000000000000000000000000.
00000006HPRK]000000000000000000000000000000000000000000.
000000000‘TI3000000000000000000000000000000000000000000.
00000000[?RRY000000000000000000000000000000000000000000.
00000000;=TTY00000000I000000000000000000000000000000000.
000000001[[YV200000000:0000=000000000000000000000000000.
00000000=]bYRT00000000[0000V00000F000000000000000000000.
00000000e?]RR000000000Y000[0000K00000000000000000000000.
00000000060[P0000000000]200[000D?0000000000000000000011.
0000000000VRPM000000004]=D[00;V000000000000000000011111.
0000000000TOMW006H6000A‘^[‘]=]0000168000000000001011131.
0000000000RPRYF000K[?;‘V54@[‘Y004?T50000000000121112221.
0000000000RTRVV00000T]DVRTTRY‘60YF000000000001112232423.
0000000000R[RYV=00000;Y7IOW[RX[020000000000111112434342.
0000000000F[YVY^00001]?6YHRTVR^B00101011111112233444353.
00000000010YV[W]0027Ye[A;:MTWY[[01123223232322224343444.
000011000110Y‘[VW39]WKdPY8DRWXVY]1324333432434435545554.
001111111111?[YY[^7111dIMHMRVYWVW2435544343433454554665.
1111213132323X^T[‘‘14DdTPVTY[[‘TT2445656545655456677766.
131242222233Y[D]bbDV[‘‘YYW[YY‘0‘?567657677766667588877.
223233344454480Wd‘[b;360][[VYV]W[Y677787887777588989:8:.
23243434335346Kb‘b‘0=757g[YRYY]‘V‘657788887877Bi;888;;=.
2243435355545XddWT‘RD]DTdPOKRRV^V[877789898888di8;===?=.
4355355545453‘bgdYM?R===FI=7=88?IY0YP?88;:8;8?gY;==?=?=.
3433545455556dbgdPA8;=RVPA7=:;IH=?BKMbB8;;;8FK[===?=?BA.
435535356454FgggYY8?TTRFA;8B:TB8I?DFMPi?:;:=Kg[?==?BABD.
5545555465657bgfR:?WD‘FB[8?=DBFAWYK?RMRi=;=RRdnd=???BAD.
6466676645456iib‘FR?bDFFK8¿]M?M[]70T[Fb];=Mbknn??BADDF.
565565666664VgfM^HDbP88W8;Y0‘eHGY‘8DPKDRKDDdikik?BABFFF.
46556664545666ibHIbI88R78?F?Kd‘HMb8=[MKYDT]^nkglA?BBDFF.
65566555455466fIFd??7?;7;A==e0iWM^A8?bVVM?inkgikFBDDFFI.
54666645564566=Fi=I88R37=B?Db?bkMFF7@gRTRMinnmniYDDFFIF.
656656646456466_MB86I648=?Bb=BKkgTR;?K0YF‘dpnnkikDDFFFF.
55676666654656He=077=66<DRVP8?P[iWD??DbWTbepppnnnDDFIHI.
66676676765656Y=087F68=]K?B8;=[KgXYF;Ain]bnppppmBDFFFF.
55665665665557[D[78878gD7=?7:=‘DVFed?FYkbdppppppFDFFIF.
5546666546666A8M?6;3==]87?=7==]BVi]bMKTeblppppppnmDFIFI.
4565656664667V=‘88?5==K78==8;?T?bbinDK[[bppppppnpDFFFH.
5464666565656]M‘8966:;:89?;:;?K=^d‘i?TdbgppnppppmnFFFGH.
5365664554568D[D;:3=====A?;:=?F?]T‘d;RbppppmppppklFFIHK.
556656545546=BY:8@5==???IM=;??KI‘P‘g:MpppppgppppgkFFIKK.
454556664666JB‘;:=6?=?;BD?88?9F[bRbd8IkpppngpppngkDFIKK.
445453546666RTd?;:=A=?;?=?89?8Bb‘]bg?PgpppminpnmikFFHKH.
355645554665FWd=;:I;:I7A??7;?;?g_^‘g?KgppnnkipniiPIIKKI.
445465456646=V^=:7D:;T7?@B6:?;?g][‘_;AgmnppkgdndiFFHIHK.
345545645466D[0;76?8‘?8=AA7;=:?d^]‘B:BgkpppipkklnFFIHKK.
233554566666F[BA58?=T8=FD?7:;=DbT^]B:¿gppnnnnkk‘FIHKKK.
343545666666F]8I3F;TB8;?F=;7;?BbT]b?8¿dppknkknnFFIHKKK.
344566656566M];=5H;=:F?FNA;7:?=[‘bgFD?YdnppkkpnYDFHIHKK.
345466666766Q‘;:8BR:@?;??=68=?;TbdgkgbTbininkiVDFIHIKKK.
445665766777Hd;7==b;D=;??;87=D8Ibbbi[^gbbk?dKFDFFIHIHKM.
546477777787;‘:8B_I:?:;7;7:7?=8Dbbei?AYg‘kBDDDDFFIHIKKK.
3555667877877V=8Rd7;;;=?:;;==;ATb‘bi??MkbkDDFFFFIHIKHKK.
6665766777777MBBd‘6=7:?;;=8?;Hilgb‘i?9Id‘iDDFFGHIKHIKKM.
6787878:88888IA‘iY6<7?=DAA?=HliifbdiB:IbgiMFFIHKKKKKMMM.
:;:;;:;====?=0DdkR6?8?FP;??InknkbgbkB?HbdgbIHMMMMMOMMMP.
??????BDABABBHIdnR=D?DTA=BKnpnpibignK‘igbkkMMRPRRRRRRR.
```

Appendix 3: Transforming the pixel data

The following WEB program illustrates how to convert data like that of Appendix 1 into the form required by the fonts and macros described earlier.

1. Introduction. This program prepares 33-level halftone images for use in TEX files. The input is assumed to be a sequence of pictures expressed in the form

m n
⟨ first line of pixel data, n characters long ⟩
. . .
⟨ mth line of pixel data, n characters long ⟩

terminated by a line that says simply '0'. The pixel data consists of the characters "0" to "9" and "A" to "V", representing 32 levels of darkness from black to white. [See Appendix 1.]

The output is the same set of pictures, expressed in a simple format used for 33-level halftones, with ASCII characters "0" to "P" representing darkness levels from white to black. The levels are adjusted to compensate for the idiosyncrasies of Canon LBP-CX laser-printing engines. Two dots are typeset for each pixel of input; hence there are $2m$ "halflines" of n-character data in the output.

2. Here's an outline of the entire Pascal program:

```
program halftones(input, output);
  label ⟨Labels in the outer block 4⟩
  const ⟨Constants in the outer block 3⟩
  type ⟨Types in the outer block 5⟩
  var ⟨Global variables 6⟩
  procedure initialize;  { this procedure gets things started properly }
    var ⟨Local variables for initialization 8⟩
    begin ⟨Set initial values 7⟩
    end;
    begin initialize; ⟨The main program 23⟩;
    end.
```

3. Each picture in the input data must contain fewer than *max_m* rows and *max_n* columns.

```
⟨Constants in the outer block 3⟩ ≡
  max_m = 200;  { m should be less than this }
  max_n = 200;  { n should be less than this }
```

This code is used in section 2.

4. The main program has one statement label, namely *cleanup_and_terminate*.

```
  define cleanup_and_terminate = 9998
  define finish ≡ goto cleanup_and_terminate  { do this when all the pictures have been output }
⟨Labels in the outer block 4⟩ ≡
  cleanup_and_terminate;
```

This code is used in section 2.

5. The character set. We need translation tables between ASCII and the actual character set, in order to make this program portable. The standard conventions of *TEX: The Program* are copied here, essentially verbatim.

```
  define text_char ≡ char  { the data type of characters in text files }
  define first_text_char = 0  { ordinal number of the smallest element of text_char }
  define last_text_char = 127  { ordinal number of the largest element of text_char }
⟨Types in the outer block 5⟩ ≡
  ASCII_code = 0 . . 127;  { seven-bit numbers }
```

This code is used in section 2.

6. ⟨ Global variables 6 ⟩ ≡

```
xord: array [text_char] of ASCII_code;  { specifies conversion of input characters }
xchr: array [ASCII_code] of text_char;  { specifies conversion of output characters }
```

See also sections 10, 13, 14, and 22.

This code is used in section 2.

7. ⟨ Set initial values 7 ⟩ ≡

```
  xchr['40] ← '␣'; xchr['41] ← '!'; xchr['42] ← '"'; xchr['43] ← '#'; xchr['44] ← '$'; xchr['45] ← '%';
  xchr['46] ← '&'; xchr['47] ← '''';
  xchr['50] ← '('; xchr['51] ← ')'; xchr['52] ← '*'; xchr['53] ← '+'; xchr['54] ← ','; xchr['55] ← '-';
  xchr['56] ← '.'; xchr['57] ← '/';
  xchr['60] ← '0'; xchr['61] ← '1'; xchr['62] ← '2'; xchr['63] ← '3'; xchr['64] ← '4'; xchr['65] ← '5';
  xchr['66] ← '6'; xchr['67] ← '7';
  xchr['70] ← '8'; xchr['71] ← '9'; xchr['72] ← ':'; xchr['73] ← ';'; xchr['74] ← '<'; xchr['75] ← '=';
  xchr['76] ← '>'; xchr['77] ← '?';
  xchr['100] ← '@'; xchr['101] ← 'A'; xchr['102] ← 'B'; xchr['103] ← 'C'; xchr['104] ← 'D';
  xchr['105] ← 'E'; xchr['106] ← 'F'; xchr['107] ← 'G';
  xchr['110] ← 'H'; xchr['111] ← 'I'; xchr['112] ← 'J'; xchr['113] ← 'K'; xchr['114] ← 'L';
  xchr['115] ← 'M'; xchr['116] ← 'N'; xchr['117] ← 'O';
  xchr['120] ← 'P'; xchr['121] ← 'Q'; xchr['122] ← 'R'; xchr['123] ← 'S'; xchr['124] ← 'T';
  xchr['125] ← 'U'; xchr['126] ← 'V'; xchr['127] ← 'W';
  xchr['130] ← 'X'; xchr['131] ← 'Y'; xchr['132] ← 'Z'; xchr['133] ← '['; xchr['134] ← '\';
  xchr['135] ← ']'; xchr['136] ← '^'; xchr['137] ← '_';
  xchr['140] ← '`'; xchr['141] ← 'a'; xchr['142] ← 'b'; xchr['143] ← 'c'; xchr['144] ← 'd';
  xchr['145] ← 'e'; xchr['146] ← 'f'; xchr['147] ← 'g';
  xchr['150] ← 'h'; xchr['151] ← 'i'; xchr['152] ← 'j'; xchr['153] ← 'k'; xchr['154] ← 'l';
  xchr['155] ← 'm'; xchr['156] ← 'n'; xchr['157] ← 'o';
  xchr['160] ← 'p'; xchr['161] ← 'q'; xchr['162] ← 'r'; xchr['163] ← 's'; xchr['164] ← 't';
  xchr['165] ← 'u'; xchr['166] ← 'v'; xchr['167] ← 'w';
  xchr['170] ← 'x'; xchr['171] ← 'y'; xchr['172] ← 'z'; xchr['173] ← '{'; xchr['174] ← '|';
  xchr['175] ← '}'; xchr['176] ← '~';
  xchr[0] ← '␣'; xchr['177] ← '␣';  { ASCII codes 0 and '177 do not appear in text }
```

See also sections 9, 11, 15, and 17.

This code is used in section 2.

8. ⟨ Local variables for initialization 8 ⟩ ≡

```
i: 0 .. last_text_char;
```

This code is used in section 2.

9. ⟨ Set initial values 7 ⟩ +≡

```
  for i ← 1 to '37 do xchr[i] ← '␣';
  for i ← first_text_char to last_text_char do xord[chr(i)] ← '177;
  for i ← 1 to '176 do xord[xchr[i]] ← i;
```

10. Inputting the data. We keep the pixel values in a big global array called *v*. The variables *m* and *n* keep track of the current number of rows and columns in use.

The *dd* table contains density values assumed for the input, indexed by single-character codes.

⟨ Global variables 6 ⟩ +≡

```
v: array [0 .. max_m, 0 .. max_n] of real;  { pixel darknesses, from 0.0 to 1.0 }
m: integer;  { rows 0 .. m + 1 of v should contain relevant data }
n: integer;  { columns 0 .. n + 1 of v should contain relevant data }
dd: array [text_char] of real;
```

11. All input codes give zero density, except "0" to "9" and "A" to "V".

```
⟨Set initial values 7⟩ +≡
  for $i \gets first\_text\_char$ to $last\_text\_char$ do $dd[chr(i)] \gets 0.0$;
  for $i \gets$ "0" to "9" do $dd[chr(i)] \gets 1.0 - (i -$ "0"$)/31.0$;
  for $i \gets$ "A" to "V" do $dd[chr(i)] \gets 1.0 - (i -$ "A" $+ 10)/31.0$;
```

12. The process of inputting pixel values is quite simple. We terminate the program if anomalous values of m and n occur. Boundary values are added at the top, left, right, and bottom in order to provide "padding" that will be convenient in the pixel transformation process. Each boundary value is equal to one of its adjacent neighbors.

```
⟨Input a picture, or terminate the program 12⟩ ≡
  $read(m)$; if $(m \le 0) \lor (m \ge max\_m)$ then $finish$;
  $read\_ln(n)$; if $(n \le 0) \lor (n \ge max\_n)$ then $finish$;
  for $i \gets 1$ to $m$ do
    begin for $j \gets 1$ to $n$ do
      begin $read(c)$; $v[i,j] \gets dd[c]$;
      end;
    $v[i,0] \gets v[i,1]$; $v[i,n+1] \gets v[i,n]$;
    $read\_ln$;
    end;
  for $j \gets 0$ to $n+1$ do
    begin $v[0,j] \gets v[1,j]$; $v[m+1,j] \gets v[m,j]$;
    end
```

This code is used in section 23.

13. The code just written makes use of three temporary registers that must be declared:

```
⟨Global variables 6⟩ +≡
$i, j$: $integer$;  { current row and column }
$c$: $char$;  { character read from input }
```

14. Pixel compensation. The 33-level output of this program is assumed to be printed by a font that contains 4×8 characters, where each character has 0 to 32 black bits. Physical properties of output devices cause distortions, so that a character with k black bits does not have an apparent density of $k/32$. We therefore maintain a table of apparent density values.

```
  define $max\_l = 32$  { maximum output level }
⟨Global variables 6⟩ +≡
$d$: array $[0 \,..\, max\_l]$ of $real$;  { apparent densities, from 0.0 to 1.0 }
```

15. This table is based on some densitometer measurements that are not especially reliable. The amount of toner seems to vary between the top of a page and the bottom; also blocks of the character "N" seem to appear darker than blocks of the character "O", because of some property of xerography, although the "O" has one more bit turned on. Such anomalies have been smoothed out here, since the resulting values should prove good enough in practice.

```
⟨Set initial values 7⟩ +≡
  $d[0] \gets 0.0$; $d[1] \gets 0.06$; $d[2] \gets 0.095$; $d[3] \gets 0.125$; $d[4] \gets 0.155$;
  $d[5] \gets 0.175$; $d[6] \gets 0.215$; $d[7] \gets 0.245$; $d[8] \gets 0.27$; $d[9] \gets 0.29$;
  $d[10] \gets 0.3$; $d[11] \gets 0.31$; $d[12] \gets 0.32$; $d[13] \gets 0.33$; $d[14] \gets 0.34$;
  $d[15] \gets 0.35$; $d[16] \gets 0.36$; $d[17] \gets 0.37$; $d[18] \gets 0.38$; $d[19] \gets 0.4$;
  $d[20] \gets 0.42$; $d[21] \gets 0.44$; $d[22] \gets 0.47$; $d[23] \gets 0.5$; $d[24] \gets 0.53$;
  $d[25] \gets 0.57$; $d[26] \gets 0.61$; $d[27] \gets 0.66$; $d[28] \gets 0.72$; $d[29] \gets 0.80$;
  $d[30] \gets 0.88$; $d[31] \gets 0.96$; $d[32] \gets 1.0$;
```

16. We convert the pixel values by using a variant of the Floyd-Steinberg algorithm for adaptive grayscale [Society for Information Display, *SID 75 Digest*, 36–37]. The idea is to find the best available density, then to diffuse the error into adjacent pixels that haven't yet been processed.

The following code assumes that x is the desired density value in column j of the current halfline. It outputs one 33-level density, then updates x and j in preparation for the next column. Adjustments to the densities in the two next halflines are accumulated in auxiliary arrays *next1* and *next2*; this will compensate for errors in the current halfline.

We assume that *next1* $[j]$, *next1* $[j+1]$, and *next2* $[j]$ correspond to the dots that are adjacent to *current* $[j]$.

```
⟨Output one value and move to the next column 16⟩ ≡
  ⟨Find $l$ so that $d[l]$ is as close as possible to $x$ 21⟩;
  $write(xchr["0" + l])$;  $err \leftarrow x - d[l]$;
  $next1[j] \leftarrow next1[j] + alpha * err$;
  $next2[j] \leftarrow beta * err$;
  $j \leftarrow j + 1$;   { move right }
  $next1[j] \leftarrow next1[j] + gamma * err$;
  $x \leftarrow current[j] + delta * err$
```

This code is used in sections 18 and 18.

17. The constants *alpha* .. *delta* control the distribution of errors to adjacent dot positions.

```
⟨Set initial values 7⟩ +≡
  $alpha \leftarrow 7/16$;  { error diffusion to SW neighbor }
  $beta \leftarrow 1/16$;  { error diffusion to S neighbor }
  $gamma \leftarrow 5/16$;  { error diffusion to SE neighbor }
  $delta \leftarrow 3/16$;  { error diffusion to E neighbor }
```

18. Here is the overall control of the process. Every halfline of the picture being output is a sequence of ASCII characters from "0" to "P", terminated by ".".

```
⟨Output the picture 18⟩ ≡
  for $j \leftarrow 1$ to $n + 1$ do
    begin $next1[j] \leftarrow 0.0$;  $next2[j] \leftarrow 0.0$;
    end;
  for $i \leftarrow 1$ to $m$ do
    begin ⟨Set the current halfline data for the upper row of dots in line $i$ 19⟩;
    $j \leftarrow 1$;  $x \leftarrow current[1]$;
    repeat ⟨Output one value and move to the next column 16⟩;
    until $j > n$;
    write_ln('.');  ⟨Set the current halfline data for the lower row of dots in line $i$ 20⟩;
    $j \leftarrow 1$;  $x \leftarrow current[1]$;
    repeat ⟨Output one value and move to the next column 16⟩;
    until $j > n$;
    write_ln('.');
    end
```

This code is used in section 23.

19. The density value for dot j in the upper halfline of line i is obtained as a weighted average of the input values in rows $i - 1$ and i, columns j and $j + 1$. The upper halfline is skewed to the right, so we must shift *next1* and *next2* appropriately.

```
⟨Set the current halfline data for the upper row of dots in line $i$ 19⟩ ≡
  for $j \leftarrow 1$ to $n$ do
    begin $current[j] \leftarrow (9 * v[i, j] + 3 * v[i, j + 1] + 3 * v[i - 1, j] + v[i - 1, j + 1])/16 + next1[j + 1]$;
    $next1[j] \leftarrow next2[j]$;
    end;
  $next1[n + 1] \leftarrow 0.0$
```

This code is used in section 18.

20. The lower halfline is similar, but in this case there is leftward skew; we use rows i and $i + 1$, columns $j - 1$ and j.

```
⟨ Set the current halfline data for the lower row of dots in line $i$ 20 ⟩ ≡
  for $j \gets 1$ to $n$ do
    begin $current[j] \gets (9 * v[i,j] + 3 * v[i,j-1] + 3 * v[i+1,j] + v[i+1,j-1])/16 + next1[j]$;
    $next1[j+1] \gets next2[j]$;
    end;
  $next1[1] \gets 0.0$
```

This code is used in section 18.

21. The algorithm is now complete except for the part that chooses the closest possible dot size. A straightforward binary search works well for this purpose:

```
⟨ Find $l$ so that $d[l]$ is as close as possible to $x$ 21 ⟩ ≡
  if $x \le 0.0$ then $l \gets 0$
  else if $x \ge 1.0$ then $l \gets max\_l$
    else begin $low\_l \gets 0$; $high\_l \gets max\_l$;  { we have $d[low\_l] \le x < d[high\_l]$ }
      while $high\_l - low\_l > 1$ do
        begin $mid\_l \gets (low\_l + high\_l)$ div 2;
        if $x \ge d[mid\_l]$ then $low\_l \gets mid\_l$
        else $high\_l \gets mid\_l$;
        end;
      if $x - d[low\_l] \le d[high\_l] - x$ then $l \gets low\_l$ else $l \gets high\_l$;
      end
```

This code is used in section 16.

22. We had better declare the variables we've been using.

```
⟨ Global variables 6 ⟩ +≡
$x$: real;  { current pixel density }
$err$: real;  { difference between $x$ and the best we can achieve }
$current$: array [0 .. $max\_n$] of real;  { desired densities in current halfline }
$next1, next2$: array [0 .. $max\_n$] of real;  { corrections to subsequent densities }
$alpha, beta, gamma, delta$: real;  { constants of error diffusion }
$l, low\_l, mid\_l, high\_l$: 0 .. $max\_l$;  { trial density levels }
```

23. The main program. Now we're ready to put all the pieces together.

```
⟨ The main program 23 ⟩ ≡
  write_ln('\input␣hf33'); write_ln;
  while true do
    begin ⟨ Input a picture, or terminate the program 12 ⟩;
    write_ln('\beginhalftone'); ⟨ Output the picture 18 ⟩;
    write_ln('\endhalftone'); write_ln;
    end;
cleanup_and_terminate:
```

This code is used in section 2.

Appendix 4: Pixel optimization

Here is another short WEB program. It was used to generate the special font for Mona Lisa.

1. Introduction. This program prepares a METAFONT program for a special-purpose font that will approximate a given picture. The input is assumed to be a binary file that contains one byte of density information per pixel. The output will be a sequence of lines like

```
row(10); cols(3,15,16,17);
```

this means that bits 3, 15, 16, and 17 of the character for row 10 should be black.

2. Here's an outline of the entire Pascal program:

```
program picfont(bytes_in, output);
  type ⟨ Types in the outer block 5 ⟩
  var ⟨ Global variables 6 ⟩
    ⟨ Basic procedures 10 ⟩
    begin ⟨ The main program 26 ⟩;
    end.
```

3. The picture in the input data is assumed to contain *mm* rows and *nn* columns.

```
  define mm = 512  { this many rows }
  define nn = 440  { this many columns }
```

4. It's convenient to declare a macro for incrementation.

```
  define incr(#) ≡ # ← # + 1
```

5. Inputting and outputting the data. The input appears in a file of 8-bit bytes, with 00 representing black and FF representing white. There are *mm* × *nn* bytes; they appear in order from top to bottom and left to right just as we normally read a page of text.

```
⟨ Types in the outer block 5 ⟩ ≡
  eight_bits = 0 .. 255;  { unsigned one-byte quantity }
  byte_file = packed file of eight_bits;  { files that contain binary data }
```

This code is used in section 2.

6. ⟨ Global variables 6 ⟩ ≡

```
bytes_in: byte_file;
```

See also sections 9, 14, 16, 22, and 25.

This code is used in section 2.

7. Different Pascal systems have different ways of dealing with binary files. Here is one common way.

```
⟨ Open the input file 7 ⟩ ≡
  reset(bytes_in, '', '/B:8')
```

This code is used in section 26.

8. We shall use the following model for estimating the effect of a given bit pattern: If a pixel is black, the darkness is 1.0; if it is white but at least one of its four neighbors is black, the darkness is *zeta*; if it is white and has four white neighbors, the darkness is zero.

```
  define white = 0  { code for a white pixel with all white neighbors }
  define gray = 1  { code for a white pixel with 1, 2, 3, or 4 black neighbors }
  define black = 2  { code for a black pixel }
  define zeta ≡ 0.2  { assumed darkness of white pixel with a black neighbor }
```

9. There isn't room to store all the input bytes in memory at once, but it suffices to keep buffers for about a dozen rows near the current area being computed.

```
⟨Global variables 6⟩ +≡
ii: integer;  { the buffer holds rows $8ii - 7$ through $8ii + 4$ }
buffer: array [−2 .. 9, 0 .. nn + 1] of real;  { densities in twelve current rows }
darkness: array [−3 .. 9, 0 .. nn + 1] of white .. black;  { darknesses in buffer rows }
new_row: array [0 .. nn + 1] of real;  { densities in row being input }
```

10. The *get_in* procedure computes the densities in a specified row and puts them in *new_row*. This procedure is called successively for $i = 1, 2, \ldots$.

```
⟨Basic procedures 10⟩ ≡
procedure get_in(i : integer);
  var j: integer; t: eight_bits;  { byte of input }
  begin new_row[0] ← 0.0;
  if i > mm then
    for j ← 1 to nn do new_row[j] ← 0.0
  else for j ← 1 to nn do
      begin read(bytes_in, t); new_row[j] ← (255.5 − t)/256.0;
      end;
  new_row[nn + 1] ← 0.0;
  end;
```

See also sections 11 and 20.

This code is used in section 2.

11. Here is a procedure that "rolls" the buffer down eight lines:

```
⟨Basic procedures 10⟩ +≡
procedure roll;
  var j: 0 .. nn + 1; i: 2 .. 9; k: integer;
  begin for i ← 6 to 9 do
    for j ← 0 to nn + 1 do
      begin buffer[i − 8, j] ← buffer[i, j]; darkness[i − 8, j] ← darkness[i, j];
      end;
  for j ← 0 to nn + 1 do darkness[−3, j] ← darkness[5, j];
  incr(ii);
  for i ← 2 to 9 do
    begin get_in(8 ∗ ii + i − 5);
    for j ← 0 to nn + 1 do
      begin buffer[i, j] ← new_row[j]; darkness[i, j] ← white;
      end;
    end;
  end;
```

12. It's tedious but not difficult to get everything started. We put zeros above the top lines in the picture.

```
⟨Initialize the buffers 12⟩ ≡
  ii ← 0;
  for i ← 6 to 9 do
    begin get_in(i − 5);
    for j ← 0 to nn + 1 do
      begin buffer[i, j] ← new_row[j]; darkness[i, j] ← white;
      end;
    end;
  for i ← −2 to 5 do
    for j ← 0 to nn + 1 do
      begin buffer[i, j] ← 0.0; darkness[i, j] ← white;
      end;
  for j ← 0 to nn + 1 do darkness[−3, j] ← white
```

This code is used in section 26.

13. It's easy to output the current darkness values. Here we output eight consecutive rows.

```
⟨Output the pixel values for the top eight rows of the buffer 13⟩ ≡
  for i ← −2 to 5 do
    begin write('row(',8 * ii − 5 + i : 1,');␣cols('); cols_out ← 0;
    for j ← 1 to nn do
      if darkness[i, j] = black then
        begin if cols_out < 15 then
          begin if cols_out > 0 then write(',');
          incr(cols_out);
          end
        else begin write_ln(','); write('␣␣␣␣␣␣␣␣␣␣␣␣␣␣␣'); cols_out ← 1;
          end;
        write(j : 1);
        end;
    write_ln(');')
    end
```

This code is used in section 26.

14. ⟨Global variables 6⟩ +≡
cols_out: 0 . . 15: { the number of columns output so far on this line }

15. Dot diffusion. The pixels are divided into 64 classes, numbered from 0 to 63. We convert the pixel values to darknesses by using a method called "dot diffusion." Values are assigned first to all the pixels of class 0, then to all the pixels of class 1, etc.; the error incurred at each step is distributed to the neighbors whose class numbers are higher. This is done by means of precomputed tables *class_row*, *class_col*, *start*, *del_i*, *del_j*, and *alpha* whose function is easy to deduce from the following code:

```
⟨Choose pixel values and diffuse the errors in the buffer 15⟩ ≡
  for k ← 0 to 63 do
    begin i ← class_row[k]; j ← class_col[k];
    while j ≤ nn do
      begin ⟨Decide the color of pixel [i, j] and the resulting err 17⟩;
      for l ← start[k] to start[k + 1] − 1 do
        begin u ← i + del_i[l]; v ← j + del_j[l]; buffer[u, v] ← buffer[u, v] + err * alpha[l];
        end;
      j ← j + 8;
      end;
    end
```

This code is used in section 26.

16. ⟨Global variables 6⟩ +≡
class_row: **array** [0 . . 63] **of** −2 . . 8: { buffer row containing pixels of a given class }
class_col: **array** [0 . . 63] **of** 1 . . 8: { first column containing pixels of a given class }
class_number: **array** [−2 . . 9, 0 . . 9] **of** 0 . . 63; { number of a given position }
err: *real*; { error introduced at the current position }
err_black: *real*; { error introduced at the current position if black chosen }
black_diff: *real*; { difference between *err* and *err_black* for gray pixel }
l: 0 . . 256; { index into diffusion tables }
start: **array** [0 . . 64] **of** 0 . . 256; { first entry of diffusion table for a given class }
del_i, *del_j*: **array** [0 . . 256] **of** −1 . . 1; { neighboring location for diffusion }
alpha: **array** [0 . . 256] **of** *real*; { constant of proportionality for diffusion }

17. Here we choose white or black, whichever minimizes the magnitude of the error. Potentially *gray* values of this pixel and its neighbors make this calculation slightly tricky, as we must subtract *zeta* when a gray pixel is created and add *zeta* when it is destroyed.

```
⟨Decide the color of pixel $[i,j]$ and the resulting $err$ 17⟩ ≡
  if $darkness[i,j] = gray$ then
    begin $err \leftarrow buffer[i,j] - zeta$; $err\_black \leftarrow err - black\_diff$;
    end
  else begin $err \leftarrow buffer[i,j]$; $err\_black \leftarrow err - 1.0$;
    end;
  if $darkness[i-1,j] = white$ then $err\_black \leftarrow err\_black - zeta$;
  if $darkness[i,j-1] = white$ then $err\_black \leftarrow err\_black - zeta$;
  if $darkness[i,j+1] = white$ then $err\_black \leftarrow err\_black - zeta$;
  if $darkness[i+1,j] = white$ then $err\_black \leftarrow err\_black - zeta$;
  if $err\_black + err > 0$ then
    begin $err \leftarrow err\_black$; $darkness[i,j] \leftarrow black$;
    if $darkness[i-1,j] = white$ then $darkness[i-1,j] \leftarrow gray$;
    if $darkness[i,j-1] = white$ then $darkness[i,j-1] \leftarrow gray$;
    if $darkness[i,j+1] = white$ then $darkness[i,j+1] \leftarrow gray$;
    if $darkness[i+1,j] = white$ then $darkness[i+1,j] \leftarrow gray$;
    end
```

This code is used in section 15.

18.

```
⟨Initialize the diffusion tables 18⟩ ≡
  $black\_diff \leftarrow 1.0 - 2.0 * zeta$;
```

See also section 19.

This code is used in section 26.

19. Computing the diffusion tables. The tables for dot diffusion could be specified by a large number of boring assignment statements, but it is more fun to compute them by a method that shows some of the mysterious underlying structure.

```
⟨Initialize the diffusion tables 18⟩ +≡
  ⟨Initialize the class number matrix 21⟩;
  ⟨Compile "instructions" for the diffusion operations 23⟩
```

20. The order of classes used here is the order in which pixels might be blackened in a font for halftones based on dots in a 45° grid. In fact, this is precisely the pattern used in the —dot300— font that was described earlier.

```
⟨Basic procedures 10⟩ +≡
procedure $store(i,j : integer)$;  { establish new $class\_row$, $class\_col$ }
  begin if $i < 1$ then $i \leftarrow i + 8$ else if $i > 8$ then $i \leftarrow i - 8$;
  if $j < 1$ then $j \leftarrow j + 8$ else if $j > 8$ then $j \leftarrow j - 8$;
  $class\_number[i,j] \leftarrow k$; $class\_row[k] \leftarrow i$; $class\_col[k] \leftarrow j$; $incr(k)$;
  end;

procedure $store\_eight(i,j : integer)$;  { rotate and shift for eight classes }
  begin $store(i,j)$; $store(i-4, j+4)$; $store(5-j, i)$; $store(1-j, i-4)$;
  $store(4+j, 1-i)$; $store(j, 5-i)$; $store(5-i, 5-j)$; $store(1-i, 1-j)$;
  end;
```

21. ⟨Initialize the class number matrix 21⟩ ≡

```
k ← 0; store_eight(7, 2); store_eight(8, 3); store_eight(8, 2); store_eight(8, 1);
store_eight(1, 4); store_eight(1, 3); store_eight(1, 2); store_eight(2, 3);
for i ← 1 to 8 do
  begin class_number[i, 0] ← class_number[i, 8]; class_number[i, 9] ← class_number[i, 1];
  end;
for j ← 0 to 9 do
  begin class_number[−2, j] ← class_number[6, j]; class_number[−1, j] ← class_number[7, j];
  class_number[0, j] ← class_number[8, j]; class_number[9, j] ← class_number[1, j];
  end
```

This code is used in section 19.

22. The tricky part of this process is the fact that some values near the bottom of the buffer aren't ready for processing until errors have been diffused from the next bufferload. In such cases we go up eight rows to process a value that has been held over.

⟨Global variables 6⟩ +≡

```
hold: array [0 .. 9, 0 .. 9] of boolean;
        { is this value too close to the bottom of the buffer to allow immediate processing? }
```

23. The "compilation" in this step simulates going through the diffusion process the slow way, and records the actions it performs (so that they can all be done at high speed later).

⟨Compile "instructions" for the diffusion operations 23⟩ ≡

```
for j ← 0 to 9 do hold[9, j] ← true;
for i ← 0 to 8 do
  for j ← 0 to 9 do hold[i, j] ← false;
l ← 0; k ← 0;
repeat i ← class_row[k]; j ← class_col[k]; w ← 0; start[k] ← l;
  for u ← i − 1 to i + 1 do
    for v ← j − 1 to j + 1 do
      if class_number[u, v] > k then
        begin del_i[l] ← u − i; del_j[l] ← v − j; incr(l);
        if u = i then w ← w + 2   { neighbors in the same row get weight 2 }
        else if v = j then w ← w + 2   { neighbors in the same column get weight 2 }
          else incr(w);   { diagonal neighbors get weight 1 }
        end
      else if hold[u, v] then hold[i, j] ← true;
  if hold[i, j] then class_row[k] ← i − 8;
  ⟨Compute the alpha values for class k, given the total weight w 24⟩;
  incr(k);
until k = 64;
start[64] ← l
```

This code is used in section 19.

24. ⟨Compute the *alpha* values for class *k*, given the total weight *w* 24⟩ ≡

```
for ll ← start[k] to l − 1 do
  begin if del_i[ll] = 0 then alpha[ll] ← 2.0/w
  else if del_j[ll] = 0 then alpha[ll] ← 2.0/w
    else alpha[ll] ← 1.0/w;
  end
```

This code is used in section 23.

25. ⟨Global variables 6⟩ +≡

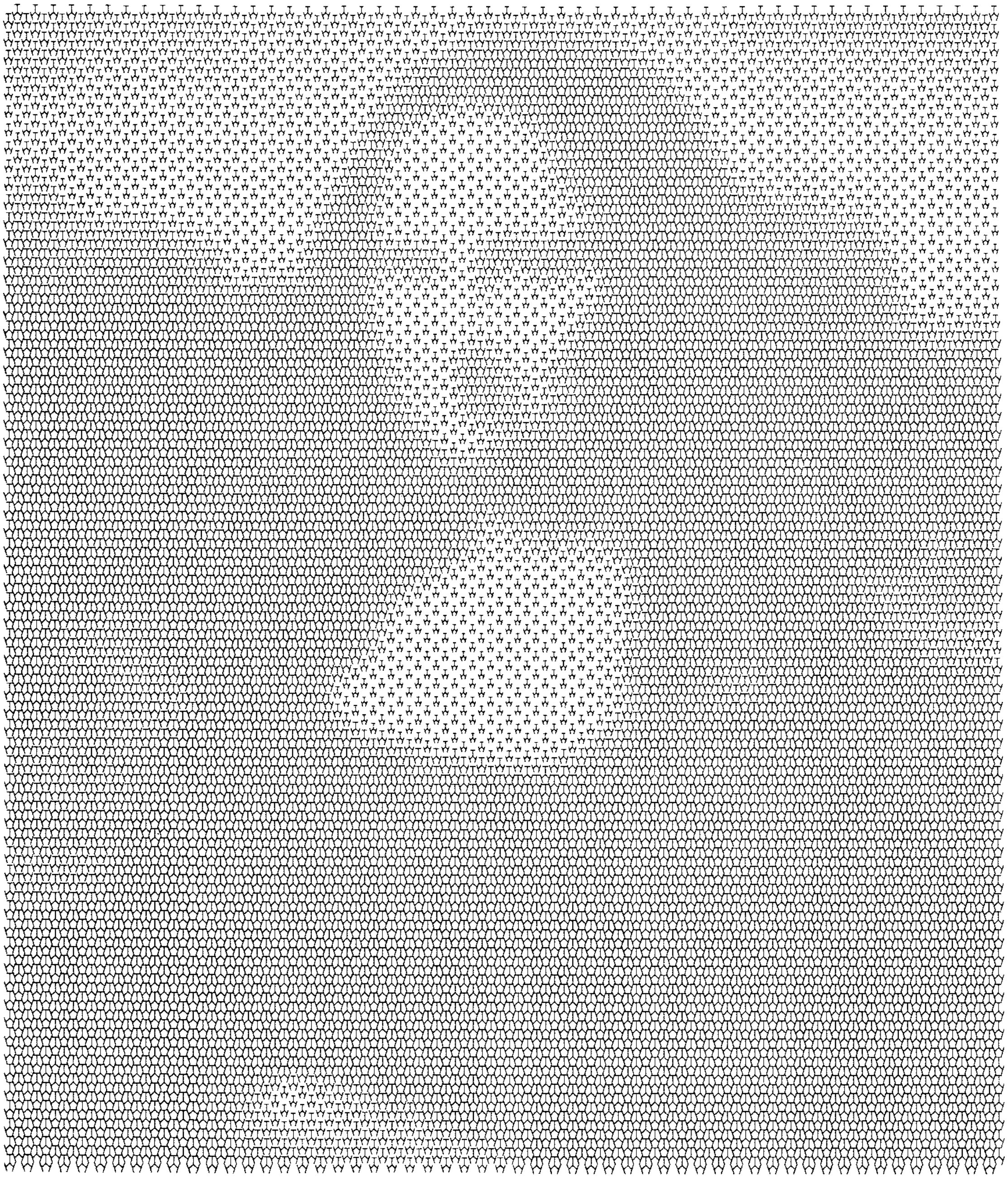
```
ll: 0 .. 256;  { loop index }
u, v: integer;  { neighbors of i and j }
w: integer;  { the weighted number of high-class neighbors }
i, j: integer;  { the current pixel position being considered }
k: 0 .. 64;  { the current class being considered }
```

26. The main program. Finally we're ready to get it all together.

```
⟨The main program 26⟩ ≡
  ⟨Initialize the diffusion tables 18⟩;
  ⟨Open the input file 7⟩;
  ⟨Initialize the buffers 12⟩;
  repeat ⟨Choose pixel values and diffuse the errors in the buffer 15⟩;
    if $ii > 0$ then ⟨Output the pixel values for the top eight rows of the buffer 13⟩;
    roll;
  until $8 * ii > mm$
```

This code is used in section 2.

Acknowledgements. The research described in this paper was supported in part by the System Development Foundation and in part by National Science Foundation grants IST-8201926, MCS-8300984, and DCR-8308109.



Addendum

Stop the presses! When I wrote the preceding pages (and had them typeset), I was unaware of a "well known" method that should have been included for comparison. So far this paper has considered (1) a halftone font with 65 levels of gray, in which each 8 × 8 character essentially contributes two dots to a picture; and (2) a halftone font with 33 levels of gray, in which each 4 × 8 character contributes one dot to a picture. It's also possible to construct (3) a halftone font with 17 levels of gray, in which each 4 × 4 character essentially contributes half of a dot (actually two quarter-dots) to a picture. This third method is based on an idea due to Robert L. Gard [*Computer Graphics and Image Processing* **5** (1976), 151–171].

The kth level of gray in the half-dot scheme is obtained by blackening cells 0 to $k-1$ in the array

1	5	10	14
3	7	8	12
13	9	6	2
15	11	4	0

or

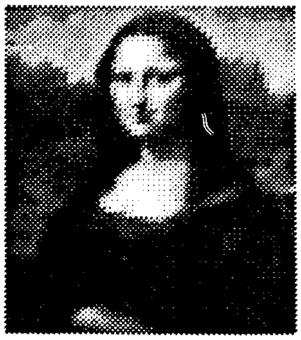
14	10	5	1
12	8	7	3
2	6	9	13
0	4	11	15

.

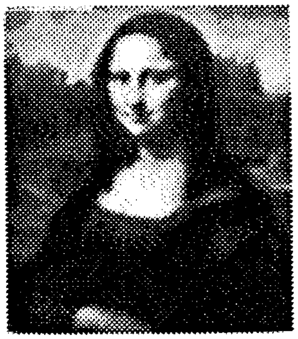
(We actually make two sets of characters, one the mirror image of the other, and alternate between them as a picture is typeset.) The following METAFONT file will generate such a font hd300, in essentially the same way that the other fonts dot300 and hf300 were generated earlier:

```
% halftone font with 17 levels of gray, characters "A" (white) to "Q" (black)
% includes also the mirror-reflected characters "a" (white) to "q" (black)
pair p[]; % the pixels in order (first p0 becomes black, then p1, etc.)
p0=(3,0); p4=(2,0); p8=(2,2); p12=(3,2);
transform r; r=identity rotatedaround ((1.5,1.5),180);
for i=0 step 4 until 12: p[i+1]=p[i] transformed r;
 p[i+2]=p[i] shifted (0,1); p[i+3]=p[i+2] transformed r; endfor
w#:=4/pt; % that's 4 pixels
font_quad:=w#; designsize:=8w#;
r:=identity reflectedabout ((2,0),(2,3));
picture prevchar; prevchar=nullpicture; % the pixels blackened so far
for i=0 upto 16:
 beginchar(i+ASCII"A",w#,w#,0); currentpicture:=prevchar;
 if i>0: addto currentpicture also unitpixel shifted p[i-1]; fi
 prevchar:=currentpicture; endchar;
 beginchar(i+ASCII"a",w#,w#,0);
 currentpicture:=prevchar transformed r; endchar;
 endfor
```

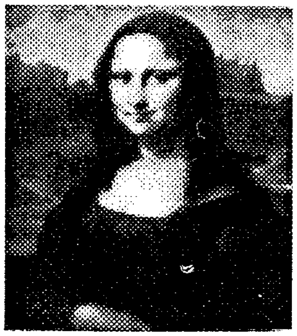
Here are four pictures for comparison, showing also the result of the elaborate "dot diffusion" method discussed at the end of my paper:



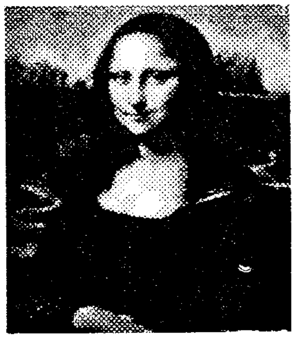
double dot



single dot



half dot



dot diffusion

(Each of these was printed on a Canon laser printer with 300 pixels per inch.) Gard's half-dot method clearly improves the quality of single-dot pictures; it also has the advantage that its characters are square instead of diamond-shaped, hence the data is easier to compute. On the other hand, it does require twice as much data. Indeed, the double-dot picture shown here was typeset from 64 rows of 55 characters each; the single-dot picture was typeset from 128 rows of 55 characters each; and the half-dot picture was typeset from 128 rows of 110 characters each.

The upper left corner of the half-dot data for Mona Lisa looks like this:

```
\beginhalftone
mMmMmLlLkKjJiIiIjJjJjJiIgGiJjJiGgGgIiIgGgGgGgGgIgFgGgGgGgFfFeFfFeGgGgGgGgGgFfFgIi ...
LmMlLkKlLkJiIiHiIiJjJkJiGgIiIiGgFgGgGgGgGfFfGgFfEfFgFfFfGgEgGfEfGgGgGgGfFfFgGeGiH ...
kKkKkKkKkJiIiHiIiIjJkKkJiIgGgFgIgFfFfFgGfEeEeEeEfGeGgGgHiJjJjIgGiGfGgGgFeFeGgFgGg ...
JjJjKkKkKjIiGiIgGiJkJjJiIiIgGfGiGfEfEfGfFeEdEeDdEeFiKlMnNnNnNnMlKiGfFfFfFeFfFeFgG ...
jJkJkKkLkJiHgGgGgIiJjJjIgGgFeFeGgGfEeEfEeDdDdEdDeJlNnOnOoOoOoOmOoNmKgEeDeEeEeDeEf ...
```

Uppercase and lowercase letters alternate in a checkerboard fashion, so that the reflected characters will appear in the correct positions. The \beginhalftone macro is the same for half dots as for double dots; only the font name and the data encoding scheme are different.

Output Devices

The Ideal TEX Driver

Robert W. McGaffey
Martin Marietta Energy Systems, Inc.
Oak Ridge, Tennessee

It is intended that this article encourage discussion of features of output drivers with a view towards both selecting the best features currently available and labelling those as standards to be met by driver writers. TEX has been available long enough and output drivers have matured to the point that it is possible and reasonable to decide upon a set of minimal features. (This is not yet true of TEX previewers.) We hope that there will be enough interest in setting standards to result in a session on output standardization at the TUG meeting this summer.

A driver should provide page-level control over:

1. Page orientation. In particular, both portrait mode and landscape mode must be supported. In addition, devices which support PostScript should be able to orient a page at any angle desired. Zero degrees should be equivalent to portrait mode, 90° equivalent to landscape, 45° halfway between, 180° upside down, and so forth (assuming increasing angles measured counter-clockwise).
2. Magnify a page to any magnification desired. We realize that the availability of fonts will affect some devices, but others such as the APSμ5 and the FR80 should be able to handle all magnifications within a given range. TEX does not restrict users to certain `\magnifications` and neither should drivers unless the fonts are missing. We recognize that this directly contradicts the restriction in TEX that there be only one document-level `\magnification` per TEX file and that it will cause trouble when `true` dimensions are used.
3. Modify the margins. TEX's `\output` routine creates a `box` and then calls upon `\shipout` to send the `box` to the `.dvi` file. The only standard that *has* been adopted (that we know of) concerns the margins surrounding the `\vbox` that TEX ships out. In TUGboat, Volume 5, No. 1, David Fuchs states in an article (page 22): "I would urge everyone to adopt the convention used at Stanford: all DVI-reading programs allow the user to specify an extra top or left margin, but these values default to 1 inch if not explicitly specified." We second the motion. The user should be able to replace these values with his own values.
4. Exceed the page boundaries without wraparound, overstriking, or some other catastrophe. Stated positively, any material which will be printed off the page should be indicated in an error message and then not printed. Ideally, if part of a figure, a rule, or even a character would fit on the page, it would be good if that part of it were included. But we realize that doing this in some cases would cause severe loss in computer efficiency.

With respect to fonts, drivers should be able to:

1. Handle all of the `CMR` fonts. (Not the `AMR` fonts as a substitute.)
2. Allow user specifiable control over handling unavailable fonts. Users should be able to select from the following list:
   a. Replace the missing characters with white space exactly matching the metric dimensions of the characters.
   b. Replace the missing characters with black ink precisely matching the metric dimensions of the characters.
   c. Substitute a smaller or larger font if the difference in size is less than 0.5 pixels on the output device and inform the user that this was done. For example, a ten point font on a 300 dot/in laser printer uses approximately 41.511 pixels for its character heights. (10 points divided by 72.27 points/in multiplied by 300 dots/in.) If we assume that the magnification for this font was 1000, then 41.011 pixels would indicate a magnification of 987.955 and 42.011 pixels comes out to 1012.045. Thus, if this font were requested at any magnification in the range 988–1012 inclusive, we can be sure that using the font at magnification 1000 would at most produce an error of one pixel in either direction. We include this feature because once (in trying to calculate magnifications) we requested a font at magnification 1096 only to be turned away. So we shifted the numerator in our calculation and got turned away for asking for 1094. There was no simple way to get 1095 and yet that was what we needed and wanted. If this feature were implemented,

we would have been in business.* In no case should a font be replaced with another font whose topology is different (such as CMB10 for CMR10).

d. "Jiggle" the pixels of the font closest in magnification to make it smaller or larger until it is the right size. This method would actually be useful for draft copies if the final output device is capable of "real" magnification. This technique would also allow pixel-bound devices to support orientations of any angle.

3. Refer to pixel files with device-dependent names. On our VAX we have an LN03 driver which refers to the pixel files using the name TEX$PXLDIR. This causes difficulties if our QMS driver uses the same name. Yet if each driver used a name like TEXLN03_PXLDIR, there would be no problem.

With respect to font loading:

1. Drivers should download only the characters specifically needed to produce the document. This will save both device memory and CPU time for both the device and the computer which drives the device.
2. Drivers should be written to be executed in three stages. The first procedure should write a file giving the font and character information needed by the document as well as the character positions. The second procedure should download the fonts and the third must set the characters on the page. The reason for this approach is to enable a font manager to replace the second procedure. This font manager could keep track of what fonts and characters are already loaded in the device and thus enable more efficient processing.

With respect to rules:

1. All \hrules with exact heights (plus depths) must span the same number of pixels on the paper, assuming devices of the same resolution. (We have seen this not occur. The results are distracting at best, and have caused us to make some macros device-dependent.) This can be accomplished by calculating the rule thickness in pixels, rounding to the nearest number of pixels, and finally, using one pixel if the result turns out to be zero. Of course, the horizontal movement should depend upon sps and not pixels.
2. Similar arguments apply to \vrules and their widths.

With respect to outputting pages:

1. There should be at least three ways to order the output:
   a. The pages may be output in the same order they were shipped out. A starting page and number of pages should be allowed here.
   b. The pages may be output in reverse order. A starting page and number of pages should be allowed here also.
   c. The pages may be output in page number order. In this mode \count0 should be more significant than \count1 which should be more significant than \count2 etc. Thus page 4.6 should be printed before page 4.7 and page 4.5 before page 5.4. Pages numbered with roman numerals should also come out in their real order (i, ii, iii, ...). In this mode drivers should allow ranges of pages such as (-2.4,3.2) for pages ii.4 through 3.2. If there should be two or more pages with the same set of \counts and that page number is specified, all of those pages should be printed.

With respect to the \special command drivers should be able to:

1. Pull in a plot from a plot file which is written in some standard format which the output device can read. This may be a .tkf file for a Tektronix plotter, a Quic file for a QMS laser printer, etc.
2. Since the above is often impractical, every driver should call a subroutine named SPECIAL and send it the character string contained in the .dvi file when it does not understand the string itself. This would allow users of the driver to expand the capabilities of their driver without knowing or having the source code available. This subroutine would be supplied as a null routine, and of course the driver would have to be sent as object code rather than executable code so that the subroutine special could be linked to the driver.

With respect to the driver parameters a driver should:

1. Be able to read an initialization file which can be made to contain the instructions and

* Editor's note: TeX itself can generate such an "inexact" magnification: for a font at \magstep4 with a document magnification of 1200, a magnification of 2489 will be requested instead of the 2488 defined by plain for \magstep5.

parameters which are usually used by the end users. Thus, if a user commonly uses landscape mode he/she may create such a file and save the effort of typing in the landscape command every time.

2. In addition to the initialization file, a driver should be able to write the commands it receives from a user into a file which can be read by the driver and executed. Thus if a user is debugging a particular set of pages, he/she need not reenter the set of commands but may instead refer the driver to the file containing the needed set of commands.
3. Drivers should also be able to run from batch mode as well as interactive mode.

With respect to METAFONT, drivers should be able to do either of the following:

1. Have the ability to download fonts produced by METAFONT.
2. Come with a program able to convert standard METAFONT output into a form needed by the driver. This would enable users to create different sizes of the CMR fonts as well as logos, special symbols, etc.

Editor's note: Robert M^c^Gaffey has agreed to chair a committee to define standards for TEX output drivers, and will be holding an organizational meeting at the Seattle TUG meeting. Persons interested in participating in this effort should write to Robert, with a copy of the letter to Bart Childs, to assist in planning.

TEX Output Devices

Don Hosek

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

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

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

Corrections, updates, and new information for the list are welcome; send them to Don Hosek, Bitnet DHOSEK@HMCVAX (postal address on page 99).

Sources

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

Adelaide Adelaide University, Australia

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

AMS American Mathematical Society, Barbara Beeton, 401-272-9500 Arpanet: BNB@XX.LCS.MIT.Edu

Arbor ArborText, Inc., Bruce Baker, 313-996-3566, Arpanet: bwb%arbortext@umix.cc.umich.edu

ArborText's software is proprietary and ranges in price from $150 to $3000. The drivers for PostScript printers, the HP LaserJet Plus, the QMS Lasergrafix, and Imagen printers are part of their DVILASER series. The drivers all support graphics and include other special features such as use of resident fonts or landscape printing when supported by the individual printers.

Printing on the Autologic APS-5 and μ-5 phototypesetters with DVIAPS includes support of Autologic standard library fonts and Logo processing.

A-W Addison-Wesley, Brian Skidmore,
617-944-3700, ext. 2253

Addison-Wesley supports graphics on all Macintosh software, and on Imagen, PostScript, and QMS laser printers on the IBM PC.

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

Caltech1 California Institute of Technology,
Glen Gribble, 818-356-6988

Caltech2 California Institute of Technology,
Chuck Lane, Bitnet: CEL@CITHEX

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

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

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

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

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

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

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

GA Tech GA Technologies

GMD Gesellschaft für Mathematik und Datenverarbeitung, Federal Republic of Germany, Dr. Wolfgang Appelt, uucp: `seismo!unido!gmdzi!zi.gmd.dbp.de!appelt`

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

IAM Institut für Angewandte Math, Univ of Bonn, Federal Republic of Germany, Bernd Shulze, 0228-733427, Bitnet: `BESCHU@DBNUAMA1`

INFN INFN/CNAF, Bologna, Italy, Maria Luisa Luvisetto, 51-498286, BITnet: `MILTEX@IBOINFN`

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

Intergraph Intergraph, Mike Cunningham, 205-772-2000

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

K&S Kellerman and Smith, Barry Smith, 503-222-4234

The MacIntosh drivers and the VAX/VMS Imagen driver support graphics.

LLL Lawrence Livermore Laboratory

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

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

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

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

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

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

MPAE Max-Planck-Institut für Aeronomie, H. Kopka, (49) 556-41451, Bitnet: `MI040L@D606WD01`

MR Math Reviews, Patrick Ion, 313-996-5273

NJIT New Jersey Institute of Technology, Bill Cheswick, 201-596-2900, Arpanet: `cheswick@jvnca.csc.org`

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

OSU2 Ohio State University, John Gourlay, 614-422-1741, `gourlay.ohio-state@csnet-relay`

Pers Personal TEX, Inc., Lance Carnes, 415-388-8853

Graphic output is supported on Imagen, PostScript, and QMS printers.

PPC Princeton Plasma Physics Lab, Charles Karney, ARPAnet: `Karney%PPC.MFENET@NMFECC.ARPA`

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

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

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

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

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

SLAC Stanford Linear Accelerator Center, 415-854-3300

The SLAC drivers are on the standard CMS distribution tape.

SRI SRI International

Stanford Stanford University

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

Sun Sun, Inc.

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

Talaris Talaris, Rick Brown, 619-587-0787

All of the Talaris drivers support graphics.

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

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

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

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

The QMS driver supports inclusion of QUIC graphics commands via specials as well as landscape printing.

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

The TEXrox package includes a DVI driver (TEXrox), a GF/PXL to Xerox font converter (PXLrox2), and a utility to build TFM files from licensed Xerox fonts (Xetrix). The programs are all written in C. Xetrix currently runs only under UNIX.

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

Tools Tools GmbH Bonn, Edgar Fuß,
Kaiserstraße 48, 5300 Bonn, Federal Republic of
Germany

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

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

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

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

UCIrv1 University of California, Irvine,
David Benjamin

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

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

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

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

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

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

The UNIX Imagen driver is on the UNIX distribution tape.

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

UNI.C Aarhus University, Regional Computer
Center

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

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

The Beebe family of drivers was described in *TUGboat, Vol. 8, No. 1.* Graphics is supported only in the DVIALW (PostScript) driver.

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

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

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

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

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

Wash St Washington State University, Dean
Guenther, 509-335-0411, Bitnet: `Guenther@WSUVM1`

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

Screen Previewers

■ Data General MV

T A&M1 See above for contact name.

■ IBM MVS

Milan1 See above for contact name.

Drives Tektronix 4014 terminal.

GMD See above for contact name.

■ Siemens BS2000

GMD See above for contact name.

■ UNIX

Adelaide Programs are on distribution tape.

The DVItoVDU program is capable of driving the following terminals: AED 512; ANSI-compatible; DEC ReGIS; DEC VT100; DEC VT220; Tektronix 4014; and Visual 500, 550.

Talaris See above for contact name.

The Talaris driver supports the Talaris 7800 terminal. Tektronix graphics are supported on-screen.

Utah See above for contact name.

The Beebe driver family includes a driver for the BBN Bitgraph display.

■ VAX VMS

Adelaide Programs are on distribution tape.

The DVItoVDU program is capable of driving the following terminals: AED 512; ANSI-compatible; DEC ReGIS; DEC VT100; DEC VT220; Tektronix 4014; and Visual 500, 550.

INFN See above for contact name.

The INFN drivers include support for DEC VT125 and Tektronix 4014 terminals.

Talaris See above for contact name.

The Talaris driver supports the Talaris 7800 terminal. Tektronix graphics are supported on-screen.

Utah See above for contact name.

The Beebe driver family includes a driver for the BBN Bitgraph display.

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

	Amdahl (MTS)	CDC Cyber	Data General MV	DEC-10	DEC-20	HP9000 500	IBM MVS	IBM VM/CMS	IBM VM/UTS	Prime	Siemens BS2000	Symbolics Lisp	UNIX	VAX VMS
Agfa P400								IAM						
Canon					Utah	Utah							Canon Utah	Utah
DEC LN01													U Wash1	NJIT
DEC LN03														K&S Procyon DEC
HP LaserJet Plus					Utah	Utah							Arbor Utah	Arbor Utah
IBM 38xx, 4250, Sherpa								SLAC Wash St						
Imagen	Arbor UBC		T A&M1	Stanford Vander	Columb. SRI Utah	Utah	Arbor	Arbor SLAC W'mann				MIT	Arbor U Md Utah	Arbor K&S Utah
PostScript printers					Utah	Adelaide Arbor Utah		Arbor				MIT	Arbor Carleton MIT Utah	Utah
QMS Lasergrafix	Arbor	U Wash2	T A&M1			T A&M2	GMD	Arbor		T A&M3	GMD	MIT	Arbor U Wash1	Arbor GA Tech T A&M3
Symbolics					U Wash1								U Wash1	U Mass
Talaris							Talaris	Wash St					Talaris	Talaris
Xerox Dover					CMU								Stanford	
Xerox 2700II		Bochum			OSU2								OSU2	
Xerox 9700	Arbor U Mich						Arbor T A&M4	Arbor T A&M4	T A&M4				U Del	ACC Arbor T A&M4

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

	CDC Cyber	Cray	Data General MV	DEC-10	DEC-20	HP9000 500	IBM MVS	IBM VM	Prime	VAX UNIX	VAX VMS
Apple ImageWriter					Utah	Utah				Utah	LSU Utah
DEC LP100					OSU2						
Facit 4542											INFN
Florida Data					MR						
MPI Sprinter					Utah	Utah				Utah	Utah
NDK 7700								IAM			
Okidata					Utah	Utah				Utah	Utah
Printronix			T A&M1		Utah	Utah				Utah	Utah
Toshiba			T A&M1		Utah	Utah				Utah	Procyon Utah
Varian											Sci Ap
Versatec	U Köln	PPC	T A&M1	GA Tech Vander	U Wash1		GMD U Milan2	Weizmann	LLL	U Wash1	Caltech2 K&S

Low-Resolution Printers on Microcomputers and Workstations — Laser Xerograp

	Apollo	Apple Macintosh	Atari ST	HP1000	HP3000	HP9000 200	IBM PC	ICL Perq	Integra Solutio
Canon							Utah	GMD	Utah
Cordata LP300							Pers		
GE 3000	COS								
HP 2680				JDJW	Pers				
HP 2688A				JDJW		Caltech1 HP			
HP Laserjet Plus			Tools	TRC Finl'd		MPAE	Arbor Utah		Utah
Imagen	Arbor OCLC						A-W Arbor Pers Utah		Utah
PostScript printers	Arbor	A-W K&S				Arbor	A-W Arbor Pers Utah		Utah
QMS Lasergrafix	Arbor Scan						A-W Arbor Pers		
Talaris							Talaris		
Xerox 9700	COS Scan								

Low-Resolution Printers on Microcomputers and Workstations — Impact, Electros

	Apollo	Apple Macintosh	Atari ST	Cadmus 9200	HP1000	HP3000	IBM PC	Integrated Solutions	SUN
Apple ImageWriter		A-W K&S					MR Utah	Utah	Utah
Diablo						Pers			
Epson			Tools		JDJW	U Shef	A-W Milan1 Pers U Shef		
Fujitsu				U Köln					
GE 3000	COS								
MPI Sprinter							Utah	Utah	Utah
Printronix							Utah	Utah	Utah
Texas Instruments 855									
Toshiba							A-W Pers Utah	Utah	Utah
Video display	Arbor	A-W K&S	U Köln	Tools			A-W Arbor Pers	UCIrv1 Utah	Arbor UCB UCIrv2

Typesetters

	Amdahl (MTS)	Apollo	CDC Cyber	DEC-20	HP3000	HP9000 200; 500	IBM MVS	IBM PC	IBM VM
Allied Linotype CRTronic									
Allied Linotype L100, L300P	Arbor	Arbor				Arbor		A-W Arbor Pers	
Allied Linotype L202								Pers	
Alphatype CRS				AMS					
Autologic APS-5, Micro-5	Arbor	Arbor COS Scan		Arbor	Arbor			Arbor Pers	Arbor
Compugraphic 8400					U Shef			Pers	
Compugraphic 8600			UNI.C				Wash St	Pers	Wash St
Harris 7500									
Hell Digiset							GMD		

Using PostScript with TeX

Alec Dunn, University of Sydney*

Several articles in TUGboat have demonstrated methods of drawing simple graphics using dots and horizontal or vertical rules. LaTeX provides a complex `\picture` environment which uses special fonts, containing arcs and diagonal lines, to draw more elaborate figures. These methods have the advantage of portability because they use only facilities common to all TeX implementations and most dvi processors. But they have the disadvantages of limited scope, difficulty of use, TeX memory limitations, and lack of an interface to other graphical systems. A more versatile approach is to link a graphical language, such as PostScript, into TeX, using the `\special` command. A recent TUGboat article [1] describes this approach, and several other sites have used this method.

This article describes work on these lines at the University of Sydney. Some small improvements on the methods of [1] make the TeX–PostScript interface simpler and more foolproof for users. Also, the use of this system with a general-purpose graphics package and with a specially-written Macintosh PostScript generator are discussed.

1 The PostScript language

This section briefly describes the features of PostScript relevant to its usage with TeX, for the benefit of readers unfamiliar with the language.

PostScript is a language for programming two-dimensional graphical and typesetting operations. It is independent of any brand of printer, and it has been implemented on several models of laser printers and typesetters. Unlike TeX, PostScript is a proprietary product, owned by Adobe Systems Incorporated. The following small example demonstrates the operation of PostScript:

```
newpath 113 92 moveto
116 96 lineto 113 100 lineto
110 96 lineto closepath stroke
```

To understand this example you should know that commands apply to the numbers preceding them, and you can safely ignore the `newpath` and `stroke` commands. The example code draws a diamond shape (◇) starting from coordinates $(113, 92)$ and with sides of length 5 units.

Note that we haven't specified what the units are. PostScript lets you define and re-define your own units, so the diamond could be drawn at any size by suitable definition of the units. This is one of the most important features of PostScript (for present purposes) — everything, including text, is perfectly scalable. Similarly, the entire coordinate system can be shifted at any time, so the point $(113, 92)$ can be placed anywhere on the page.

Other valuable features of PostScript are demonstrated by the example. Nowhere in an ordinary PostScript program is there any reference to the printing hardware: the example will produce the same results on a 300 dots/inch laser printer and on a phototypesetter; raster conversion is performed in the printer. And the code is entirely in visible ASCII characters and so is fully portable and communicable between different computers and operating systems. PostScript is just as portable as TeX.

2 \special in TeX

The natural way to combine TeX and PostScript is to use the TeX `\special` command to pass PostScript code to the dvi processor (TeX itself has no use for PostScript, since TeX is only concerned with placing objects on pages, not with actually imaging those objects).

To obtain a PostScript graphic, using our system, the TeX user puts a command of the form

*Comments should be sent to A Dunn, School of Electrical Engineering, University of Sydney, NSW 2006, Australia, or via ACSnet to alecd@facet.ee.su.oz.

```
\special{PF filename height width}
```

into his or her TeX file at the point at which the lower left corner of the graphic is to be placed (which may be inside a float). The `PF` is just a keyword, chosen by us, to distinguish this kind of `\special` command from any others which may be used (using a keyword for this purpose is recommended in *The TeXBook* page 228). The PostScript code is in file *filename* — we don't insert the PostScript code itself because it tends to be verbose and the `\special` command uses TeX memory. The *height* and *width* arguments define a bounding box for the graphic. (Historical reasons compelled the unfortunate choice of *height, width* order instead of PostScript's x, y order). An example of this form of `\special` command is:

```
\special{PF diamond 4mm 3mm}
```

which is how the diamond example above was drawn.

The `\special` command occupies no height or width (since TeX can't interpret its contents), so in practice it must be supplemented by glue commands, for example:

```
\hbox to width
   {\vrule height height width 0pt
    \special{PF filename height width}\hfil}
```

Of course, this can be simplified for users by a suitable macro.

This is the full extent of TeX's role — most of the work in combining TeX and PostScript is performed by the dvi processor. In this case the processor, called Dvi/PS, was written by us for Vax/VMS machines and is proprietary to the University of Sydney. If you have the source code to a different dvi processor you may be able to adapt it to handle the `\special` command (naturally, it must be driving a PostScript device!).

3 \special in Dvi/PS

When Dvi/PS encounters a `\special` command, it already knows the coordinates of the lower left corner of the graphic (by the same mechanisms by which it knows where text is to be placed); the `PF` keyword and the *filename*, *height*, and *width* arguments follow in the dvi file. Dvi/PS then scans the PostScript file to find its bounding box, which, if the file conforms to the Adobe structuring conventions [2], will be given in a specially-formatted comment line.

Knowing the bounding box of the graphic in its own PostScript coordinate space, and the desired location and bounding box in the page's space, Dvi/PS computes a suitable transformation matrix and sends this to the printer before sending the contents of the PostScript file. This relieves the user from having to know anything about the size of the PostScript graphic, or about its coordinate system — the graphic will always appear where the user asked for it and at the size he or she asked for.

It is possible, even likely, that the user-specified bounding box and the graphic's PostScript bounding box will have different aspect ratios. PostScript allows different scale factors horizontally and vertically, so the graphic could be fitted exactly to the user's bounding box. But most users don't want their graphics distorted in this way, so Dvi/PS computes only one scale factor, according to the limiting dimension (horizontal or vertical), and applies that to both dimensions, adjusting the origin transformation so that the graphic will be centered in the non-limiting dimension.

An example may make the whole process clearer. Suppose the TeX file contains

```
\special{PF cat 180pt 200pt}
```

so the user is asking for a graphic of 200pt×180pt (in normal x, y order). Also suppose the PostScript file `cat` has a bounding box of (110, 50) to (190, 140), after conversion to TeX points. The PostScript width is 80 pt and height 90 pt. The limiting dimension is the vertical, allowing a scale factor of 2×, which leaves 40 pt of white space to be taken up in the horizontal. So Dvi/PS emits PostScript code to shift the origin horizontally by $-110 + 40$ pt and vertically by -50 pt and then to magnify by 2. The graphic, as printed, will be 180 pt high and 160 pt wide, horizontally centered in its bounding box (see Figure 1).

If either dimension is given as zero in the `\special` command Dvi/PS ignores the corresponding PostScript dimension in its scaling calculation and doesn't center the graphic, leaving it left- or bottom-justified. Combinations of zero and non-zero arguments, in suitable macros, give most of the facilities users want.

It is not an error for the PostScript code to draw outside of its supposed bounding box, since users

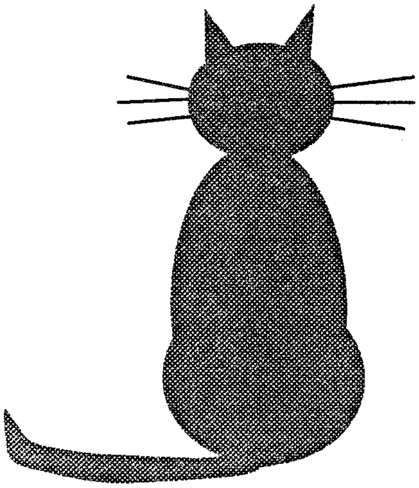


Figure 1: cat

may want to achieve special effects this way. Dvi/PS doesn't produce code to clip the image, nor does it draw the bounding box onto the page.

A PostScript file to be used in a `\special` command should not depend on another such file, since it is not known in what order Dvi/PS will process them. Also, text should use only native PostScript fonts, not downloaded fonts, which Dvi/PS may not yet have loaded. In practice these are not serious limitations.

4 Error handling

Of course, errors are possible: the PostScript file may be missing, or it may not conform to the structuring conventions and so not contain a bounding box specification. Dvi/PS copes with such problems by leaving white space if the file is missing, or not performing any coordinate transformation if the bounding box is unspecified.

But it is impractical for Dvi/PS to attempt to control errors in the PostScript file itself — once Dvi/PS sends the PostScript file to the printer it effectively hands over all control to that file. For the purposes of this system we can define a *well-behaved* PostScript file as one which leaves the printer in the same state as it found it, except for the coordinate system (which Dvi/PS always restores) and marks added to the page (which was the purpose of sending the file). For example, a file which executes the PostScript `showpage` command (which prints and ejects the page) is not well-behaved.

Unfortunately, few PostScript files are well-behaved! Most software with PostScript output aims to produce a complete specification of the final hard copy, which is the purpose for which PostScript was conceived, but here we are using it for "graphical procedures". This problem is the major limitation in using PostScript with TEX — to get well-behaved PostScript has required writing our own software.

5 Sources of PostScript code

You can write PostScript code yourself with any text editor, but this is impractical for graphics of any complexity. At the School of Electrical Engineering we have extended our general-purpose graphics package so that it can produce a well-behaved PostScript file instead of driving a graphics device. Most of the graphically-oriented software written in the School in the last five years can now be used together with TEX, and several theses and reports have been printed almost entirely without cutting and pasting.

Graphics produced by applying programs to data are very useful in engineering, but we also need to be able to just *draw* and have the drawing translated into PostScript. *MacDraw* on the Macintosh is ideal for simple engineering drawing, and *MacPaint* for freehand drawing. But it is tricky to make the Macintosh produce a PostScript file, and almost impossible to convert that file into well-behaved PostScript, so we have written a program, *PostScript from Mac*, which can convert *MacDraw* and *MacPaint* files directly into PostScript files suitable for inclusion in TEX documents. (The `cat` example, above, was drawn with *MacDraw* and converted by *PostScript from Mac* into a PostScript file of about 1 Kbyte.)

PostScript from Mac has proved surprisingly popular with the academic staff of the School and has revealed an unexpected demand for a means of *easily* including good quality graphics into TEX documents.

References

[1] H. Varian and J. Sterken, "MacDraw Pictures in TEX Documents", *TugBoat*, vol 7 no 1.

[2] *PostScript Language Reference Manual*, Addison-Wesley, Reading, Mass. Appendix C, pp 263ff.

Site Reports

TEX-L Access for Bitnet Users

Glenn Vanderburg
Texas A & M University

Texas A & M operates a TeX file- and list-server for those with Bitnet access. To use the server, send commands to LISTSERV@TAMVM1. Commands can be sent either interactively or in mail files. To get a list of the available TeX-related files, send the command "`get tex filelist`". You can then get any of the files listed with "`get filename filetype`". The server accepts many other commands; issue an "`info`" command to find out more.

The list of available files is currently small; TeXhax archives and files, WEB sources, and the LaTeX style repository. We hope eventually to offer most of the files that those with ARPAnet access can get from Score (sources, change files for various systems, macro packages and documentation, etc.). Suggestions are welcome, and should be sent to X230GV@TAMVM1.BITNET (along with complaints and problems). And if anyone would like to donate a disk pack or two, that would help out a lot!

Finally, TEX-L is a TeX discussion list and TeXhax redistribution. To subscribe, send this command to LISTSERV@TAMVM1 with your name: "`subscribe tex-l firstname lastname`" and you will begin receiving TEX-L and TeXhax from a nearby Bitnet list server.

TeX and Training
A Case Study

Laurie Mann
Stratus Computer
Marlboro, MA

Over the last seven years, many "Site Reports" have appeared in these pages. This is a site report with a difference — I'm not a programmer or a system administrator. This is a look at the issue of TeX training at Stratus, and how we hope to improve training for new TeX users.

For the last $3^1/_2$ years, Stratus Computer has been using TeX to produce software manuals and other publications. Between 1981 and 1983, Stratus had published about twenty manuals, with all the production work being sub-contracted out. But Ron Webber, the Publications manager, wanted to bring production in-house. He looked into several production systems, and decided to port TeX to the Stratus and use the Compugraphic 8400 as the output device. With some help from Eric Janszen, he ported TeX, wrote a driver, wrote a format file and taught four technical writers the basics about formatting manuals with TeX.

I was hired back in November, 1983 as the first Publications Specialist. The first few weeks were very interesting and very frustrating. My previous computer experience was pretty limited. I knew how to log in. I knew how to use Emacs. Ron spent some time the first day getting me acquainted with the Stratus equipment, and giving me some introduction to TeX. But being fairly non-technical, I found everything very confusing. It took me a while to remember that a file was not permanently changed **until** I'd written it out. I had difficulty comprehending the differences between executing a command macro and working in an Emacs buffer (something I hate to admit today).

Some of TeX looked easy. Before Ron hired me, he showed me the format file he'd written, and some of that even read like English. Besides, at first I was only supposed to be debugging TeX files that the writer had formatted. Compiling and binding programs is easy, right? But debugging TeX files is a not a trivial activity. I remember the first time I saw the error,

```
TeX capacity exceeded, sorry
If you really absolutely need more capacity,
you can ask a wizard to enlarge me.
```

I was absolutely lost. Ron explained what most of the error messages meant, including some I'd seen that he'd never seen when he was experimenting with TeX. But, over the winter, the error messages became less frequent. And I generally ceased to panic when I did see an unfamiliar error. I started to learn how to format manuals, so the writers could concentrate on writing rather than formatting. Formatting, particularly the challenge of producing nice tables and usable indexes, really appealed to me.

I read everything I could about TeX, but there really wasn't very much material available. I had a line-printer copy of *The TeXbook*. While the first few chapters were absolutely invaluable, most of it seemed to be aimed at programmers. Ron's written introductory material on TeX read the same way.

Though I struggled with TEX a lot as I was first learning it, within a few months I was TEXnician enough to format manuals and send off completed manuals to our printer every month.

The fledgling production group added a second book producer in April, 1984. Carol Klos also had little computer experience before coming to Stratus, but was interested in learning about manual production. She also found learning TEX difficult, but was soon producing high-quality manuals. That fall, I became the Production group's manager. By mid-85 there were five Publications Specialists and one full-time Typesetter in the Production group. We also added a Compugraphic 8600 typesetter and an MCS High Speed Processor. The department produced something over sixty manuals, some newsletters, and some long Marketing projects in 1984 and 1985.

One of first things I was supposed to do upon becoming the group's manager was to assemble an in-house production manual. Well, one thing led to another, and the manual sits in assorted pieces in my directories. Over the last few years, I restructured Ron's basic TEX information and included lots of examples. I worked with Mary Fusoni, the second technical writer at Stratus, to develop formatting and indexing guidelines for writers. I wrote up many brief descriptions of how to use tools and equipment, but this information is disorganized and is sometimes later found to be wrong (I've since strongly encouraged people to edit what I write). While I'm a TEX resource, I'm not a very good teacher, and the training materials we have are somewhat lacking.

Back in 1985, when three new people joined the group in a six week period, I worked with each person individually for a time but pretty much encouraged her to learn things on her own. The worst mistake here was giving new hires too much, too fast. What I should have done instead was to introduce new employees to the computer system, and have them spend a few days learning Emacs and basic operating system (VOS) commands. I should have waited before introducing TEX, working with the batch queue, debugging files, using the phototypesetter, etc. But despite the fact that everyone had early difficulties, the entire group has now been together for over two years. For the last two years, manual production has been so efficient that manuals were completed and printed before the software release tapes were cut.

In-house manual production saves time and money. Not long after Ron brought Production in-house, he created positions for Editors and Project Leaders. He also had us strengthen department procedures, and develop a team approach to manual publication. Our department now has five project leaders, each one responsible for an area of Stratus manuals (Communications, End-User, Languages, etc). Each team consists of a Project Leader, Writer, Editor and Publication Specialist. The Publication Specialist is responsible for most of the formatting, illustrating, and production of the manuals. The Writer writes the manual, but does minimal TEX formatting. The Editors and Projects Leaders usually don't touch the source files at all, and work only on typeset galleys. We all believe this team approach has helped us improve the quality of our manuals. Our customers have been giving us good marks over the last two years. Three manuals were published in 1985 won local Society for Technical Communication (STC) awards, and one of those took an International STC award.

Ron was a member of TUG in the early '80s, and I joined in 1986, after spending a year in the STC. But our only contact with TUG was through TUGBOAT. When I went to the TEX conference at Tufts last summer, I wasn't sure what to expect. Would I be the only non-programmer there? Was TEX training an issue for other sites? It turns out that **all kinds of people use TEX**, from secretaries to technical writers, from graphic designers, to, yes, computer programmers. A number of us got together to discuss how to improve TEX training, and we all felt it was something we should pursue in the future.

Over the winter, I came to the conclusion that as much as I liked working with TEX, I really didn't much like the managerial side of the job, so I switched jobs with Carol. This has given me more time to study TEX, as I've been implementing a new format file. Additionally, I have started to organize material for that production manual. This is important now because Carol will be hiring more Production Specialists over the next few months. Carol and I want to take a more structured approach to the training of new people. Additionally, we are going to see if we can hire people with prior publishing and/or computer experience (send your resumes to Carol if you are interested — we may also be hiring a tools programmer later this year).

While hiring non-technical people to work with TEX succeeds in the long run, it can cause a lot of job frustration, particularly in the first months. There are several reasons for this:

1. Insufficient instruction on how to use VOS.
2. Insufficient instruction on how to use Emacs.
3. Insufficient material on basic TEX information for non-programmers.

4. Insufficient organized material on formatting standards and production tools.

Stratus now has good introductory manuals about VOS and about Emacs. We will need to help new people get adjusted to using computer equipment, and have taken this into consideration as an important part of new-hire training. Until new hires are comfortable using VOS and Emacs, it's counter-productive to present much about TeX.

We want to take a more structured approach to teaching production skills in general and TeX in particular. Initially, only the basics about TeX will be presented. It's been my personal experience that overloading TeXnical detail early on is very confusing to most non-programmers. (It's very interesting once you've adapted to using TeX, but not immediately.) Once the new hires master simple TeX concepts (like font designations, spacing, running TeX) they can go on to learn about how TeX is used to create "beautiful books."

Since last summer, with the help of others in the Publications department, I've been assembling an outline for the production manual. Actually, the planned manual is now two — a tutorial and a reference manual. Here's the proposed outline for the tutorial:

Introduction

Section 1: VOS Tutorial
- **A.** VOS and Stratus Computers
- **B.** Using VOS
  - **1.** Logging-In and Logging-Out
- **C.** Using Emacs
  - **1.** Moving the Cursor
  - **2.** Moving Text
  - **3.** Saving Text
- **D.** File Management
- **E.** Basic VOS Commands

Section 2: Introductory TeX Reference Material
- **A.** Basic Printing Concepts
  - **1.** Fonts
    - **a.** Characters
  - **2.** Spacing
  - **3.** Page Make-up
- **B.** Typesetting Languages and TeX
- **C.** Stratus Implementation of TeX
  - **1.** Type Fonts and Type Faces
  - **2.** Point Sizes
  - **3.** Spacing
    - **a.** Examples of Vertical Spacing
    - **b.** Examples of Horizontal Spacing
    - **c.** Special Characters
  - **4.** Running TeX
    - **a.** Debugging Log Files
    - **b.** Preparing Files for the Typesetter
  - **5.** Introduction to Machines

Section 3: TeX Tutorial
- **A.** Exercise 1: A Letter
- **B.** Exercise 2: A Sign
- **C.** Exercise 3: A Phonelist
- **D.** Exercise 4: A Flyer
- **E.** Exercise 5: A Form
- **F.** Exercise 6: A Table
- **G.** Exercise 8: A Report

Note on presentation of samples: Each exercise will begin with the specifications for a job and an unformatted file (the manual will also direct the reader to an on-line, unformatted file). In the back of the book, the final, formatted file will appear on a left-hand page and the typeset output will appear on a right-hand page.

Appendix A: Formatted file and typeset output for each exercise

When people are brought in, they will have five experienced Publications Specialists to help train them. We also hope to have this tutorial in place in the next few weeks, with the reference manual to follow later in the summer. I'll let you know how things go.

Data General Site Report

Bart Childs

We have updated to TeX 2.1 and now support DG's 4558 laser printer, which is a pretty plain Canon engine. I mentioned some of its limitations in the last issue. We seem to have little or no problem with it except that the size of the font downloading files is inordinately large. It can still operate near rated capacity with parallel or 19.2k serial interfaces.

We also have graphics inclusion. The source of the driver is the same as for our QMS driver; thus we are making it by use of a change file and have a large number of "device dependencies" as well as the "system dependencies". We reserve the latter for the operating system and compilers.

Graphics inclusion in the QMS uses the QUIC language and Canon-compatible instructions on the 4558. We use `QIC` and `LBP` for the file extensions. I have written a `WEB` code for the translation of

QIC files to LBP files. It is limited at this time to translation of bitmap patterns.

We require that the TeX sources allocate the space for the graphics to be included. The graphic named drawing is included by use of the TeX command:

```
\special{copy{drawing}}
```

This implies that drawing.qic is read by the QMS driver and that drawing.lbp is read by the 4558 driver. This is a step toward the goals mentioned by Leslie Lamport in the last issue without committing to PostScript only.

I suggest that we introduce a companion file with the extension .siz that would be created by any graphics-producing program or translator. This would give TeX the information necessary to allocate the space for graphics.

Enhancements to TeX on the VAX

Adrian F. Clark
British Aerospace

This note describes two enhancements to TeX under VAX/VMS. These are believed to be equally applicable to the two VMS implementations, by David Fuchs (the Stanford distribution) and by Kellerman and Smith.

The first enhancement allows TeX to determine automatically whether or not it was invoked by an interactive user. If it was, TeX runs in **error_stop_mode**, where it will prompt the user for instructions if an error is detected. When used non-interactively, TeX runs in **batch_mode**; errors are recorded in the transcript file but execution is not affected. This facility was absent from the last Stanford distribution received by the author (TeX 1.3) while Kellerman and Smith use a command line qualifier to set batch mode.

The interactive or batch nature of a job is determined by testing whether the process-permanent logical name SYS$INPUT translates to a terminal device. (SYS$INPUT is used because of the prompting mechanism of LIB$GET_FOREIGN.) This means that, when executed from a command procedure by an interactive user, TeX runs in **batch_mode**; this is similar to other VMS utilities. The technique works for "virtual" and networked terminals as well as directly-connected devices. It has been tested over both DECnet and VAX/PSI (X–25) links.

The second enhancement is to interface TeX to an editor, a feature which is missing from both VMS implementations. When TeX spots an error in an input file and prompts the (interactive) user with "?," a reply of "E" invokes the editor and positions the cursor at the place in the input file where TeX spotted the error. After correcting the text, exiting the editor causes the file to be written to disc; the context of the error is again displayed and the "?" prompt re-issued.

The editor which has been interfaced to TeX is callable TPU, since it allows the cursor to be positioned at the appropriate part of the file with ease. However, the user is not restricted to the default EVE editing interface. Customised editors, such as the EDT emulator, can be selected in the usual way, simply by assigning the appropriate section file to the logical name TPUSECINI.

To position the cursor at the correct point in the text, a file is created in SYS$SCRATCH containing the required TPU commands. This is executed as the editor starts up and deleted as it exits. It is possible to add commands to this initialisation file; this could be used for, say, loading TeX-related keypad definitions. To do this, the TPU commands should be stored in a file and its name assigned to TeX$TPU_INI. The contents of the file are then written into the TPU initialisation file before the cursor-positioning commands.

As an example, the author uses the **EVEplus** editor (available through DECUS) with an extension to provide an EDT-like keypad layout. The commands to set this up are (on the author's VAX):

```
$ DEFINE TPUSECINI   BAE$SYSTEM:EVEplus
$ DEFINE TeX$TPU_INI SYS$LOGIN:EVEplus.INI
```

where EVEplus.INI contains the line:

```
set_edt_keypad;
```

The enhancements are largely made via TeX's VMS change file, although a little of the code is contained in a separately-compiled file written in Fortran. The change file is based on David Fuchs' TeX 1.3 implementation, modified by the author for TeX 2.0. The interaction-testing code works on any VMS 4 version; the editor interface was written under VMS 4.5 but should work on VMS 4.2 and later systems. The files which implement these enhancements are available to the TeX community; they can be obtained, for those with access to JANET, by MAILing a request to UK.AC.KCL.PH.IPG::ALIEN.

IBM VM/CMS Site Report

Dean Guenther
Washington State University

Back in January, a new IBM VM/CMS distribution tape was made available through Maria Code. This tape corrected one problem in the CMS version of TeX. The problem showed itself in documents which used `\write` statements. Specifically, a `\write` with an argument of more than 256 characters was being truncated. The max length has now been increased to 2048.

On this tape were extensive corrections to the drivers for the IBM 3812, 3820, 3800–3, 4250, and APA6670 (Sherpa). Thanks for these are due to Bob Creasy. These drivers now can imbed inline GDDM or SAS/Graph graphic page segment. For example,

`\special{IPS` *filename*`}`

where *filename* has the filetype of PSEG3820. Refer to the following article by Gil Pierson on creating SAS files for imbedding into TeX. Thanks to Gil for getting this going.

Another contribution from Bob was a utility, PFS, which creates the Computer Modern font set for the 3812, 3820, 4250, APA6670 or 3800–3. It uses standard METAFONT and sets the blackness and other parameters depending on which printer it is creating fonts for. It does take quite a bit of CPU time, so I would recommend that you use VM/Batch for this job.

Agnes Hsu added the missing filemode specification in the CMS changes file for BIBTEX.

The CMS version of METAFONT now accepts information from the command line. Corrections were made to these MF files: CMBASE, CMTT12, CMTT10, CMR5, GREEKU, CMSS9, and CMSSI9. This brings these 7 files up to date. A new MF file, CMHINCH (half inch) has been included for those printers that cannot handle CMINCH.

Other changes:

- A utility to convert old 2K TFMs to the new 1K format has been added;
- The changes file for INITeX is now included;
- Faster versions of the CMS ⇒ MVS transfer programs (APART/TOGETHER);
- Finally, GFtoDVI now works properly on CMS.

The same week this tape was sent to Maria, the message "It happened..." came across the network. (See TUGboat, Vol. 8, No. 1, pg 6.) I didn't have time to install the fixes to make TeX 2.1 until April. Along with TeX 2.1, this tape contains a CMS changes file for GFtoPK. The first tests of GFtoPK seem to work fine. I also modified the PSIZZL macros to use CM instead of AM fonts. Finally, the file for testing IBM fonts (IBMFONT.TEX) was fixed. As of May, Maria has the VM/CMS TeX 2.1 distribution.

As for IBM font tapes, by the time this is published, Maria will have MVS font tapes for the IBM 3800–3, 3820, and 3812. Maria also has CMS dump format tapes for the 3800–3, 3820, 3812 and 4250. No font tape is available for the APA6670.

Janene Winter has put together a collection of *mode_def* settings that she has found to be optimal for the IBM printers cited above. These settings appear along with comparable settings for other printers in the overview beginning on page 132.

SAS Merged with TeX

Gil Pierson
Washington State University

If you want to merge SAS/Graph output into TeX output on the IBM printers, you can do so fairly easily with the VM/CMS distribution driver programs. First you build a SAS program with the GDDMNICKNAME of IBM3820. Specify the GSFNAME to be the filename of the input file. The GDDMTOKEN must reflect the resolution of the printer, in this case IMG240X. Consider this example which uses the filename of DRUM3:

```
OPTIONS NOTEXT82;
GOPTIONS DEVICE=GDDMFAM4 GSFNAME=DRUM3
GSFLEN=264 GSFMODE=REPLACE
GDDMNICKNAME=IBM3820 GDDMTOKEN=IMG240X
NODISPLAY;
DATA A;
   M=2;  N=5;  PI=3.1416;
   DO Y=0 TO 1 BY .02;
      DO X=0 TO 1 BY .02;
         Z=2*(SIN(M*PI*X)*SIN(N*PI*Y)+
             SIN(N*PI*X)*SIN(M*PI*Y));
         OUTPUT;
         END;
      END;
   FORMAT Z 4.1; RUN;
PROC G3D;
   PLOT Y*X=Z / CTOP=GREEN CBOTTOM=RED;
   TITLE C=WHITE F=SIMPLEX 'M=2  N=5';
```

Next, you must create a PROFILE ADMDEFS file which must have the filename included, in this case DRUM3.

```
NICKNAME NAME=IBM3820,
   TONAME=(DRUM3,PSEG3820),
   PROCOPT=((CDPFTYPE,SEC),
            (HRIFORMT,CDPF),
   (HRISPILL,NO),(HRISWATH,1),
   (HRIPSIZE,30,30,TENTHS),
   (PRINTCTL,6,0,1,66,0,0,0))
```

Then call SAS to create the file:

```
GLOBAL TXTLIB ADMGLIB ADMPLIB
SAS DRUM3
```

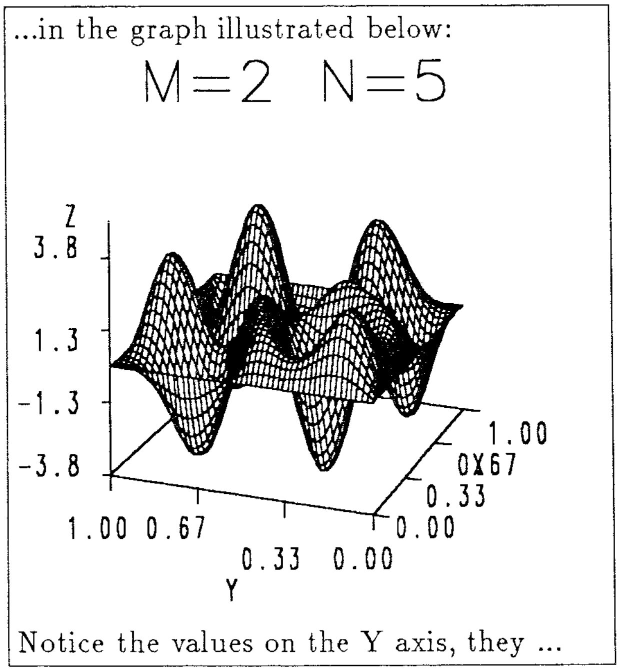
The file created is DRUM3 PSEG3820, which can be included with a `\special`. It has the form,

`\special{IPS` *filename*`}`

where *filename* is DRUM3 in this case. For example,

```
\noindent
....in the graph illustrated below:
...in the graph illustrated below:

\special{IPS drum3}
\vskip 3in
\noindent
Notice the values on the Y axis, they ...
```

will print as:



...in the graph illustrated below:

M=2 N=5
Z
3.8
1.3
-1.3
-3.8
1.00
0X67
0.33
0.00
1.00 0.67
0.33 0.00
Y

Notice the values on the Y axis, they ...

The above sample was printed on a 240dpi IBM 3820.

UnixTEX

Pierre MacKay
University of Washington

The most significant news for this Unix TEX site report is that a System V port for the AT&T 3B2 has been completed by Lou Salkind of NYU, and is now part of the regular Unix distribution. This port depends on the System V version of the BSD4.x pc compiler, and is still limited by the fundamentalist rejection of the default, or "others" case in Pascal case statements. It is therefore necessary to find a source for pxp (the BSD4.x Pascal source prettyprinter) and to make use of the still undocumented `-O -f` flags which wrap up every case statement in a set-inclusion test. We can now hope that other machine-specific System V ports will follow. It is not clear yet whether the version of Berkeley pc will migrate to non-AT&T systems or whether we should look for new ports perhaps developed with commercially available Pascal compilers. In either case, the System V barrier is at last down, and the other half of the Unix world can now look forward to the availability of TEX.

A second approach to System V TEX is Pat Monardo's Common TEX in C, being developed at Berkeley. Working versions of this are already known at a number of sites, and it is currently being upgraded to correspond with TEX Version 2.1. Common TEX is coded in traditional C fashion, without the rich documentation of a WEB source, and without the convention of a pool file, but on many architectures it is likely to produce somewhat tighter and faster executable code. It is not yet a part of the Unix TEX distribution.

The move from BSD4.2 to BSD4.3 on VAXen has caused very few serious difficulties. Recompilation is necessary owing to the change in the format of `a.out` files, which also makes it impossible to use the old version of undump.

On Suns, the changes are more radical. There is a large gulf between Sun operating system 3.x and its predecessors. This is further complicated by the difference between Sun-2 and Sun-3 versions of the new O. S. Sun-3 users (68020 CPU) have no serious trouble with any version of O. S. 3.x, but Sun-2 users must upgrade to O. S. 3.2 before trying to compile TEX. It is a measure of the breadth of the Unix TEX community that I first learned from Peter Ilieve in Scotland of some possible improvements in the Sun compilation. The pc that comes with the new Sun O. S. has adopted the "otherwise" heresy, and provides some other nice features not formerly available. Moreover, the "optimization" loop in the assembler has been cleaned up, and compilation no longer takes a week of elapsed time. It is now possible to compile the output from tangle directly, without the use of either pxp or the old sed script for breaking up the source file.

We are still working to track down ports for Celerity, Sequent, Intelligent Systems, Gould and others.

Output drivers have remained stable. (Too stable in the case of the dvi-to-PostScript driver.) The best news is that Matt Thomas has rewritten the LN03 driver to support all the varieties of METAFONT output. His driver is now the standard offering on the distribution tape. A good deal of work has recently been done on dvi-to-PostScript drivers. Richard Furuta is presently looking at one possible upgrade developed in Canada, and I am looking at the Unix relevant parts of Nelson Beebe's collection of drivers available from Utah. (The Utah drivers are accompanied by a thorough manual which can be read with profit by anyone concerned with dvi output.) We hope to improve the TEX devices directory in the summer by incorporating a good part of the work mentioned here. We hope also to offer the first genuinely open access to a true high-resolution phototypesetter before the year is out.

Contents of the distribution

Current versions of standard programs in the distribution:

(in the `./tex82` path)
- TEX 2.1 (implying use of the new cm fonts) (plain.tex version 2.3)
- LATEX 2.09 (release of 19 April, 1986, with corrections to 22 February, 1987) (also SliTEX 2.09)
- tangle 2.8
- weave 2.9
- dvitype 2.8
- pltotf 2.3
- tftopl 2.5

(in the `./mf84` path)
- METAFONT 1.3
- chtopx 1
- gftodvi 1.7
- gftopk 1.3 (changes as published in TUGboat)
- gftopxl 2.1
- gftype 2.2
- pktopx 2.3 (changes as published in TUGboat)
- pktype 2.2 (changes as published in TUGboat)
- pxtoch 1.1 (with change contributed by William LeFebvre)
- pxtopk 2.3 (changes as published in TUGboat)
- mft 0.3 (a formatting program for METAFONT source files)

Fonts:
- Fonts in `mf` source format. The full Computer Modern as released from Stanford. Utility fonts for character proofs etc. (not made with `cmbase.mf`). LATEX and SliTEX fonts are also supplied in `mf` source format.
- Fonts in `gf` format. Just about all standard shapes and sizes in 118, 200, 240, and 300 `gf`(dpi) series. The `gf` files for the principal LATEX symbol fonts are in a separate list. The Euler fonts are supplied in `gf` format only. (If you want `[euler].mf` source files, you should get in touch with the American Mathematical Society, which will make these available under license.)
- Fonts in `pk` format. AMS fonts — Cyrillic, and special symbols (created with old METAFONT-in-SAIL). All but the `*.590pxl`, `*.1000pxl`, `*.1200pxl` and `*.1500pxl` fonts are in `*.*pk` format. The relevant directories include `sh` scripts (e.g. `makepxl300.sh`) for expanding the entire list. See the `*.list` files in each font subdirectory for an idea of the size of the expanded font directory.
- Fonts in `pxl` format. Only the bare minimum set declared in `plain.tex`, in 590 1000 1200 and 1500 sizes, as mentioned above. See also `mcyr`, `msxm`, `msym` (amsfonts).

The distribution has grown to the point of needing a small support organization, which has received a grant from the National Endowment for the Humanities to continue and expand the service of software distribution. Even so, it has become necessary to raise the price slightly. Amateur economists may be amused by the fact that this is one area where volume does not bring lower costs.

How to order

To order a full distribution of TeX, send $100.00 (foreign sites $110.00, to cover the extra postage) payable to the University of Washington to:

Pierre A. MacKay
Northwest Computer Support Group, DW-10
University of Washington
Seattle, Washington 98195

The normal distribution is a tar tape, blocked 20, 1600 bpi. If you need 1/4 inch streamer cartridges for the SUN, be sure to tell us. Although we have had problems previously with cartridge tapes, we can usually hope to get them out quite fast now.

Typesetting on Personal Computers

A Bug in TEXTURES v0.95 Prerelease

Gerhard F. Kohlmayr
Mathmodel Press

ABSTRACT. Using a current version of TEXTURES™, a discrepancy between Macintosh[1] Plus screen display and ImageWriter[2] printout is documented.

The purpose of this note is to call attention to an implementation error[3] in TEXTURES[4] v0.95 prerelease. The control sequences (see [1] and [2]) '\ne', '\neq', '\not=', '\not\equiv', '\not\in', and '\not\subset' are **correctly displayed** on the Mac+ screen, but **incorrectly printed** by the ImageWriter. For example, the *correctly displayed* inequality '$1 \neq 2$' is *incorrectly printed* as '$1 = 2$' and the *correctly displayed* negated membership relation '$x \notin y$' is *incorrectly printed* as '$x \in y$'. (These symbols are, no doubt, of great interest to the average mathematician.) No attempt has been made to find all control sequences which lead to corrupted ImageWriter printouts.

How can one resist the temptation to introduce the term WYSINAWYG (what-you-see-is-*not-always*-what-you-get)?

Barry Smith of Kellerman & Smith has supplied a work-around for this problem, as described in the following note. I am grateful to him for responding within a few days to a first draft of this note by offering, over the telephone, a hands-on demonstration of the ImageWriter bug-fix. I tried it and it worked to my satisfaction.

BIBLIOGRAPHY

1. Donald E. Knuth, "The TeXbook,' American Mathematical Society and Addison-Wesley Publishing Company, Providence, Rhode Island and Reading, Massachusetts, 1984.
2. M. D. Spivak, Ph.D., "The Joy of TeX," American Mathematical Society, Providence, Rhode Island, 1986.

I wish to thank P. D. F. Ion for sending me a preliminary version of the AMS-TeX macro package for the Macintosh Plus and for his assistance over the telephone when I had a problem getting AMS-TeX running. AMS-TeX™ American Mathematical Society.

[1] Macintosh is a trademark licensed to Apple Computer, Inc.

[2] ImageWriter is a trademark of Apple Computer, Inc.

[3] Also called simply a bug.

[4] ©1987 Kellerman & Smith • ™Addison-Wesley.

Work-around for an ImageWriter Problem Affecting Output from TEXTURES

Barry Smith

The failure of the \not character to print in output to the ImageWriter is due to the ImageWriter's improper handling of characters having zero width. There are two such characters in the Computer Modern fonts, both in cmsy: the \not character already cited (character "36) and the piece character that forms the left end of the \mapsto symbol (↦, character "37).

The work-around is fairly straightforward, but requires a special piece of software — a font editor: either Resedit or Fontastic will do the job. The change required to make zero-width characters appear is to reset their width to 1 pixel. This is sufficient to avoid the ImageWriter bug, but not enough to cause problems with character positioning or appearance.

If Apple ever gets around to handling zero-width characters in a more intelligent manner, this problem will go away.

Update: Real Typesetting from Your Personal Computer

Alan Hoenig and Mitch Pfeffer

In Vol. 7, No. 3, we listed some sources of typesetting services for TEX users who need output from real typesetting machines. Some new information has turned up, and we are listing these new facts and figures according to the same rules applied in the original article: no favorites. And we ask the same consideration from readers — if anyone knows of other organizations that offer TEX typesetting services, please get in touch.

American Mathematical Society, P O Box 6248, Providence, RI 02940; (401) 272-9500 The AMS is now using an Autologic APS Micro-5 for its typesetting. At the present time the Society has AM, CM and Times Roman fonts available and within the next few months it expects to have many more typefaces from the Autologic library. Turnaround time varies depending on the size of the job but should be no more than a week for up to a 500 page job.

The AMS has also simplified its pricing structure. The charge for producing typeset output from your DVI file is $5 per page for the first 100 pages, $2.50 per page for all additional pages. The minimum charge for any job is $30. Files can be sent on VAX/VMS tapes or on IBM PC or Macintosh diskettes (although no PostScript extensions can be handled).

For scheduling purposes, the AMS asks that you contact them before submitting any jobs; talk to Regina Girouard.

Macros

Macros with Keyword Parameters

Wolfgang Appelt
Gesellschaft für Mathematik und
Datenverarbeitung, Sankt Augustin

TEX uses positional parameters for passing arguments to a macro. This has, as it is usually the case with positional parameters, two consequences: When calling a macro, which was defined to have parameters (#1, #2, ...),

1. the order of the arguments is important, i. e. it usually gives different results, if you write `\a{b}{c}` or `\a{c}{b}`, and
2. the number of the arguments must match the number of parameters in the definition of the macro.

There are, however, sometimes situations where the concept of keyword parameters is more adequate. Consider, for example, a `SetStyle` macro which may be used for modifying the formatting environment. The arguments, that can be passed to the macro, can be selected from a set of predefined keywords, say {RAGGEDRIGHT, BOLD, ITALIC, NOINDENT, ...}. The order of the arguments shall be unimportant and the number of the arguments shall be variable, i. e. the macro may be used as

```
\SetStyle(ITALIC)      or
\SetStyle(NOINDENT;ITALIC;RAGGEDRIGHT)
```

(If several arguments are present, they must be separated by a delimiter, for example by a ";".)

The following macros solve the problem (explanations afterwards):

```
\newif\if@more@keywords
\newif\if@keyword@matched

\def\SetStyle (#1){%
    \def\arguments{#1;\end}%
    \@more@keywordstrue
    \loop\expandafter\next@style@keyword
         \arguments
    \if@more@keywords\repeat}

\def\next@style@keyword #1;#2\end{%
    \def\next{#2}%
    \ifx\next\empty\@more@keywordsfalse
        \else\def\arguments{#2\end}\fi
    \@keyword@matchedfalse
    \compare@with@keyword #1<BOLD><\bf>%
    \if@keyword@matched
       \else\compare@with@keyword #1%
            <RAGGEDRIGHT><\raggedright>\fi
    \if@keyword@matched
       \else\compare@with@keyword #1%
            <ITALIC><\it>\fi
    \if@keyword@matched
       \else\compare@with@keyword #1%
            <NOINDENT><\parindent=0pt>\fi
    \if@keyword@matched\else
        \message{Unknown keyword #1!}\fi}

\def\compare@with@keyword #1<#2><#3>{%
    \def\next{#1}\def\keyword{#2}%
    \ifx\next\keyword
        \@keyword@matchedtrue #3\fi}
```

First we define two switches (`\if@more@keywords` and `\if@keyword@matched`) which are used within the macros. The `\SetStyle` macro has one parameter which is a keyword or a list of keywords, separated by a ";". The argument passed to the macro is saved in the macro `\argument` with a "`;\end`" added at the end. This will later on be used to detect the end of the argument. Then the macro starts calling `\next@style@keyword` within a loop as long as the switch `\@more@keywords` is `true`. The argument to `\next@style@keyword` is expanded before the macro is actually called, i. e. what the macro "sees" is something like

```
NOINDENT;ITALIC;RAGGEDRIGHT;\end
```

The macro `\next@style@keyword` splits the list of keywords into the first keyword, which is anything up to the first ";", and into the rest, which is anything up to the final `\end`. If the rest is empty, the switch `\@more@keywords` is set to `false`, so the loop in `\SetStyle` will stop. Otherwise the macro `\arguments` is redefined to contain the rest of the keywords with the "`\end`" added at the end again.

Then we start comparing the current keyword `#1` with the list of predefined keywords. This is done by calling the macro `\compare@with@keyword` several times, each time with a different specific keyword which is regarded a valid argument to `\SetStyle`. To avoid unnecessary calls of this macro when the current keyword was already found, we use the switch `\@keyword@matched`. If the current keyword was not recognized at all, when the list of the specific keywords is exhausted, an error message is written. The macro `\compare@with@keyword` is called with three arguments: the current keyword, a specific keyword and (usually) an action, that is to be performed, when the current keyword matches the specific one. The definition of `\compare@with@keyword` is simple: The first two parameters are compared. If they are identical, the switch `\@more@keywords` is set to `true` and the third parameter is "executed".

Expanding the set of valid keywords for the `\SetStyle` macro is trivial: It is only necessary to add furthers calls of `\compare@with@keyword`, each time with a new specific keyword, within the definition of `\next@style@keyword`. There is no restriction on the number of keywords, that can be used, i. e. the restriction, that a TEX macro must not have more than nine (positional) parameters, does not hold for keyword parameters.

In some applications a slightly different kind of keyword parameters is necessary. Consider, for example, a `\SetDimensions` macro, which shall be used for modifying some parameters controlling the size and positioning of a page. The macro may, for example, be used as

`\SetDimensions(VSIZE=1.5\char92hsize)` or
`\SetDimensions(HSIZE=20pc;HOFFSET=10pt)`

Now each keyword has an associated value, which shall be passed to some "attribute", denoted by the keyword. This case can be handled by the following macros:

```
\def\SetDimensions (#1){%
    \def\arguments{#1;\end}%
    \@more@keywordstrue
    \loop\expandafter\next@setdim@keyword
         \arguments
    \if@more@keywords\repeat}

\def\next@setdim@keyword #1;#2\end{%
    \def\next{#2}%
    \ifx\next\empty\@more@keywordsfalse
        \else\def\arguments{#2\end}\fi
    \@keyword@matchedfalse
    \compare@with@attribute #1%
            <HSIZE><\hsize=\value>%
    \if@keyword@matched
       \else\compare@with@attribute #1%
            <VSIZE><\vsize=\value>\fi
    \if@keyword@matched
       \else\compare@with@attribute #1%
            <HOFFSET><\hoffset=\value>\fi
    \if@keyword@matched
       \else\compare@with@attribute #1%
            <VOFFSET><\voffset=\value>\fi
    \if@keyword@matched\else
        \message{Unknown keyword #1!}\fi}

\def\compare@with@attribute#1=#2<#3><#4>{%
    \def\next{#1}\def\keyword{#3}%
    \ifx\next\keyword\@keyword@matchedtrue
        \def\value{#2}#4\fi}
```

The macros are rather similar to the previous ones. The main difference to the example above is the macro `\compare@with@attribute` which is used instead of the former `\compare@with@keyword`. The macro is called with three arguments as in the previous example, but it has four parameters in its definition, the first two being separated by a "=". This serves for splitting the first argument, which might, for example, be

```
VSIZE=1.5\hsize
```

into the "attribute name" and the "attribute value". If the attribute name matches the third parameter (or — looking at the call of the macro — the second argument), the switch `\@keyword@matched` is set to `true` and the meaning of `\value` is defined as the attribute value.

Notice the macro `\value`: When it is passed as an argument to `\compare@with@attribute` it is still undefined. In other words, we have the funny case of a macro which — to some extent — defines the arguments, that it receives, itself.

The two examples above show rather simple applications of keyword parameters without great practical value. They should primarily be regarded as an explanation of the basic ideas how such macros can be written. In practice further extensions may be necessary. One extension may be the mixture of positional and keyword parameters, another one the definition of macros, where the keywords in the argument list may have to be reordered before they get interpreted.

The discussion on positional versus keyword parameters has a long tradition in computer science and common understanding is probably, that keyword parameters are preferable to positional ones in many cases. Also several document processing systems, e. g. Reid's *SCRIBE* system (B. K. Reid: *SCRIBE — Introductory User's Manual*, Unilogic Ltd., Pittsburgh, 1980), make use of keyword parameters to some extent. (There are even a few features in LaTeX which look like keyword parameters though Lamport does not use this terminology. See, for example, the *options* that can be given with a `\documentstyle` command.)

Using the concept of keywords parameters can probably lead to macro packages with user interfaces, which look quite different from existing ones and might be preferred by many users. Maybe even the writing of "bridgeware" macro packages to other formatting languages, for example a macro package that makes (at least certain classes of) *SCRIBE* documents processable by TeX, might become easier.

When I first thought about keyword parameters I was surprised, that it took only a few hours to write down some macros that solved the problem. So, if after all the examples above may show nothing, they at least prove once again the flexibility and power of TeX's macro language.

\expandafter vs. \let and \def in Conditionals and a Generalization of PLAIN's \loop

Alois Kabelschacht
Max-Planck-Institut für Physik

Conditionals with \expandafter

Sometimes the replacement text for a TeX macro should end with one or another macro call, depending on a condition. The trivial solution

```
... \if... ... \aa \else ... \bb \fi
```

works only if neither `\aa` nor `\bb` needs an argument. Otherwise a more complicated construction such as the following example from `plain.tex` is needed:

```
\def\ph@nt{\ifmmode
    \def\next{\mathpalette\mathph@nt}%
  \else\let\next\makeph@nt\fi\next}
```

There is the alternative:

```
\def\ph@nt{\ifmmode
    \expandafter\mathpalette
    \expandafter\mathph@nt
  \else\expandafter\makeph@nt\fi}
```

which uses the fact that the expansion of both `\else ... \fi` and `\fi` is empty. This alternative is definitely shorter (by 4 tokens) and as far as I can see not slower. It has the further advantage that it also works if expandable tokens are expanded but no commands are digested (e.g. in the replacement text for `\edef`). The alternative construction is clearly even more economical in such cases where one of the branches would otherwise contain a '`\let\next\relax`'.

A generalization of PLAIN's \loop macro

Using the above idea one could e.g. replace PLAIN's definition of `\iterate` (used in conjunction with `\loop`):

```
\def\loop#1\repeat{\def\body{#1}\iterate}
\def\iterate{\body \let\next\iterate
    \else\let\next\relax\fi \next}
\let\repeat=\fi % this makes
      % \loop...\if...\repeat skippable
```

by

```
\def\iterate{\body
    \expandafter\iterate\else\fi}
```

Finally, omitting the `\else` and rearranging things a bit one obtains

```
\def\loop#1\repeat{\def\iterate
      {#1\expandafter\iterate\fi}%
  \iterate \let\iterate\relax}
```

which allows constructions such as

```
    \loop ... \if... ... \else ... \repeat
    \loop ... \ifcase ... \or ...
                \else ... \repeat
```

The final '\let\iterate\relax' throws away the token list for the body of the loop which could be quite long.

TEX in the Commercial Environment Setting Multi-Column Output

Elizabeth M. Barnhart

A little more than two years ago, **TV GUIDE** magazine started to investigate the possibility of using the TEX typesetting language to compose both the national feature and local program-listing sections of the magazine. The idea of vendor independence was one of the most attractive attributes of using this as our composition language.

Academic vs. the Commercial Environment

As we started to get more involved, we discovered that a large percentage of the TEX community consisted of academic users of TEX in colleges and universities around the country, but that few commercial typesetting applications were using TEX.

The academic user is usually involved with a relatively small quantity of output — from a few pages to perhaps several hundred pages. In contrast, **TV GUIDE** publishes over 100 editions in the United States and Canada for each weekly issue. The output comes to approximately 15,000 pages per week, presenting quite a different processing problem.

In the typical academic environment, one person might key in text through a word processor or PC editor and handle the style and output of the text by the inserting of typesetting commands directly into the text. In our environment, the same keystrokes are captured once, and may repeat in several areas and in many editions of the magazine. No one single person enters the text that makes up a page of the magazine. Editors for each local station gather the programming information and send it to the main office in Radnor, Pa. Output is handled by feeding items through pre-defined typesetting-specification files.

Specific Problems

Although TEX has many positive features, we have encountered some problems as we experiment with a variety of the type elements that compose **TV GUIDE**.

One problem is that TEX was designed for much wider columns than the ones called for by our typesetting specifications. We have been able to get around this with adjustments of the `\tolerance` and penalties that control the line breaking algorithm in TEX. It would be infeasible to use the TEX defaults for these penalties, which would require frequent interacting with the copy to eliminate the many "*overfull boxes*" that would occur.

Another problem is that TEX is a paragraph setting composition language; all other composition languages that our staff had been exposed to set type line by line. In line-by-line systems, once a line of type has been hyphenated and justified, it is closed and will not be changed; TEX can rework a paragraph completely differently when one word is eliminated. This has been presenting some problems in our environment, since knowing exactly where a line breaks is important to us. Our text often includes optional copy, and we need to know how lines will fit together if optional copy in the center of a paragraph is eliminated. Taking text measurements from the longest version of the copy has been our solution to this problem.

We have also encountered difficulties with the fact that when TEX produces a `.dvi` file, the fonts involved lose their identity. They are assigned a number in the font table contained in the "postamble" of the `.dvi` file. We need to be able to convert the text back to the original format, so we must be able to reconstruct the font calls made in the original text. We are experimenting with dealing with this problem by forcing system-specific font calls into the `.dvi` file using the `\special` command.

Another one of the big problems we have encountered is the complexity of defining page layouts with "output" routines. Each section of **TV GUIDE** is different and within those sections each page can be different. One example would be switching from a three-column to a two-column format within an article. Another layout requirement is leaving drops for photographs or artwork that occupy portions of more than one column of type. We are experimenting using `\parshape` commands within output routines to deal with this problem. We may find ourselves developing a front-end page description language to make this usable in a production environment that has a short turn-around time.

Input/Output Example

There are many different sorts of typeset items that go into the production of **TV GUIDE** magazine, for example, features, prime-time grids, program listings, close-up articles, cable-movie guides, etc. One of the problems encountered in our type of application concerns the switching from a single column format to a two, three or more column format.

A multi-column format is presented here, that can handle output from one to five columns. For the sample page, the counts and dimensions referred to in the commented section of the macro file are defined as follows:

`\global\dimen40=150pt`	`\hsize` value.
`\global\dimen50=477pt`	`\vsize` value.
`\global\dimen60=314pt`	Width of all columns.
`\global\count30=2`	Number of columns.
`\global\count70=1`	Turn on outside rules.
`\global\count71=1`	Turn on inside rules.
`\pageno=16`	First page of the batch.
`\def\startpage{\pageno=16}`	Gets no running head.
`\def\editdate{April 4, 1987}`	For running heads.

These values will set 2-column pages with 150-point columns and rules to either side of the columns. Changing any one of these values will affect the output. For example, changing `\count30` from a 2 to a 3 will change the output to 3-column pages. This will work without adjusting any of the other values.

The macro file shown addresses only one aspect of the typeset page, but the sample page shown illustrates how each page must be tailored to cut rules for photographs, use `\parshapes` to leave areas to drop in photos and have figure-caption text inserted. The handling of these problems is not discussed in the macro-file listed below.

The TEX input would have to be refined to include these details before being used in a production environment. Below is a sample of TEX input for a multi-column output routine.

We are in the beginning stages of adopting TEX into the editorial system that collects the feature and program text that makes up the magazine. We invite other commercial TEX users to discuss problems and solutions of their typesetting problems in the *TUGboat* and would like to know if there is interest in establishing a commercial TEX users group.

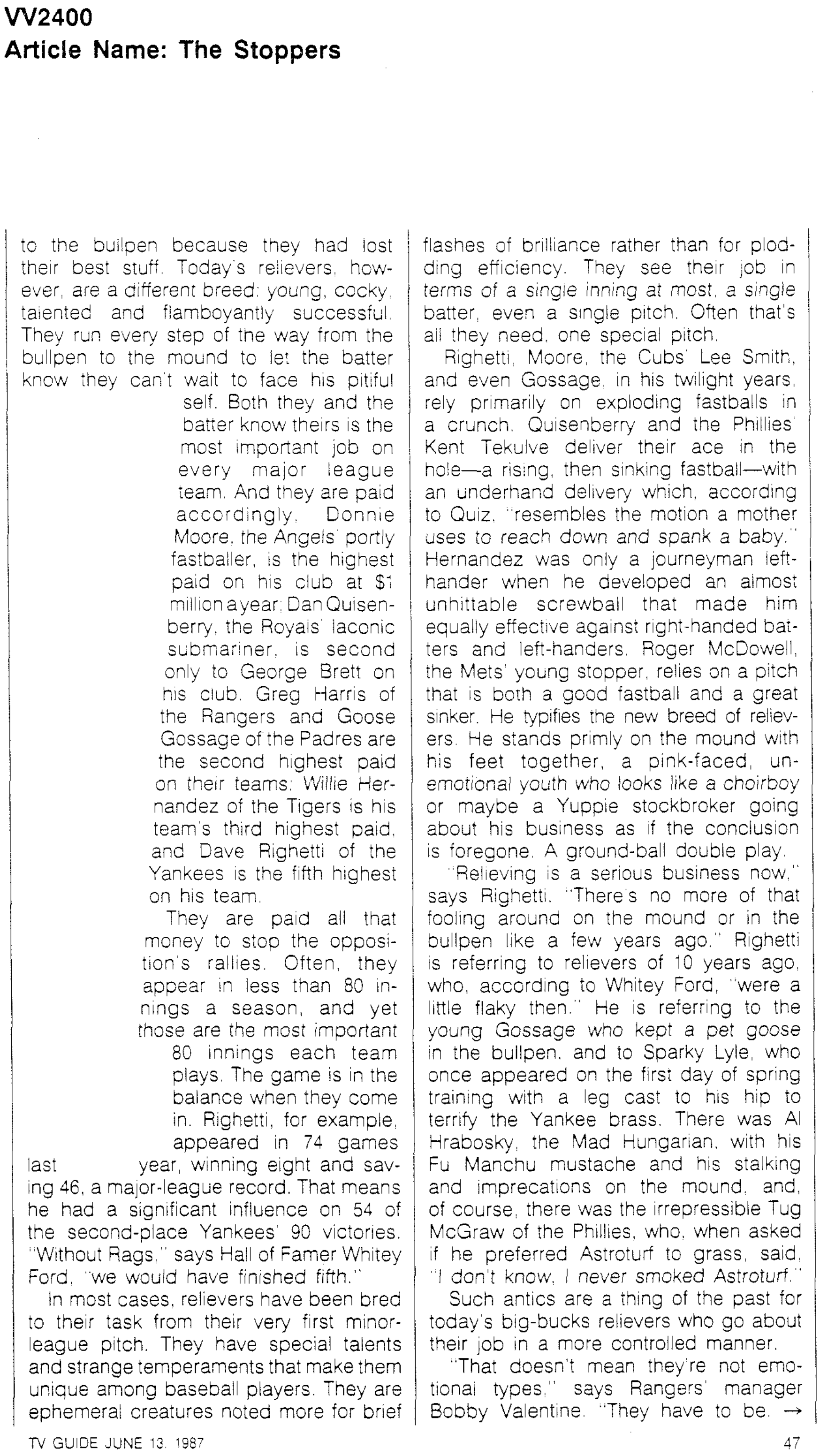
Multi-Column Macros

```
%
% COLRHRF.MAC
% Multi-column layout for 1 to 5 columns,
% with 2-piece running heads that switch
% for odd and even pages, and an outside
% edge folio for a running foot.
%
% To use this macro-file, the following
% definitions must be inserted at the top
% of your input file:
%
% \global\dimen40=00pt (Becomes \hsize value)
% \global\dimen50=00pt (Becomes \vsize value)
% \global\dimen60=00pt
% \global\count30=1   (Used for 1 to 5 columns)
% \global\count70=1   (rule on [1], off [0])
% \global\count71=1   (rule on [1], off [0])
% \pageno=1   (Set to first page number wanted)
%
% (``startpage'' will not get a running head)
% \def\startpage{\pageno=1}
%
% The next definition is fed into
% the running head calls in the output.
%
% \def\editdate{ }
%
\newdimen\fullhsize
\newdimen\pagewidth
%
% ``fullhsize'' is the full page width
% including the space between the rules
% and the text, and the gutter space
% between columns.
%
\fullhsize=\dimen60
\pagewidth=\fullhsize
\hsize=\dimen40        % Column Width
\vsize=\dimen50        % Page Depth
\def\fullline{\hbox to \fullhsize}
\def\space{\ }
\def\TVG{{\bf TV\GUIDE}}
\def\arrow{{\sy\char65}}
%
% The tolerance is defined very high for
% narrow columns, it allows more hyphenation
% and bigger word-spacing.
%
\tolerance=10000
\doublehyphendemerits=0
\finalhyphendemerits=10000
\hyphenpenalty=100
%
% These are in-house TV Guide fonts and
% must be changed if you want to use
% this macro file.
```

```
%
\font\helvlight=tvg8hlr at 8pt
\font\helvlightit=tvg8hls at 8pt
\font\runlight=tvg7hlr at 7pt
\font\sy=tvg7pi at 7pt
%
% Running head layout definitions
% ``Blankheadline'' is used for the
% starting page. Only lefthand turnover
% pages get a ``continued'' line.
%
\def\bheadline{{\hsize=\dimen60
 \leftline{\quad }}}
\def\blankheadline{{\baselineskip=7pt
 \vskip0pt\bheadline}}
%
\def\lheadline{{\hsize=\dimen60
 \leftline{\quad \runlight continued}}}
\def\leftheadline{{\baselineskip=7pt\vskip0pt
 \lheadline}}
%
\def\rheadline{{\hsize=\dimen60
 \leftline{\quad }}}
\def\rightheadline{{\baselineskip=7pt
 \vskip0pt\rheadline}}
%
% Running feet layout definitions.
%
\def\lfootline{\hsize=\dimen60
 \line{\quad \runlight \folio\hfill
 TV GUIDE \editdate\quad}}
\def\leftfootline{{\baselineskip=7pt
 \vskip 7pt{\runlight \lfootline}}}
\%
\def\rfootline{\hsize=\dimen60
 \line{{\quad \runlight TV GUIDE \editdate
 \hfill \folio\quad }}
\def\rightfootline{{\baselineskip=7pt
 \vskip 7pt{\runlight \rfootline}}}
%
% Define a BOX to hold each column while
% building the page.
%
\let\lr=A \newbox\onecolumn \newbox\twocolumn
\newbox\threecolumn \newbox\fourcolumn
\def\columnbox{\leftline{\pagebody}}
\def\midrule{\hfil\vrule width0.5pt\hfil}
\def\openrule{\vrule width0.5pt\hfil}
\def\closerule{\hfil\vrule width0.5pt}
%
\ifnum\count30=1
\output={
 \oneformat \global\let\lr=A
 \ifnum\outputpenalty>-20000
 \else\dosupereject\fi}
%
```

```
\else\ifnum\count30=2
\output={\if A\lr
 \global\setbox\onecolumn=\columnbox
 \global\let\lr=B
 \else \twoformat \global\let\lr=A\fi
 \ifnum\outputpenalty>-20000
 \else\dosupereject\fi}
%
\else\ifnum\count30=3
\output={\if A\lr
 \global\setbox\onecolumn=\columnbox
 \global\let\lr=B
 \else\if B\lr
 \global\setbox\twocolumn=\columnbox
 \global\let\lr=C
 \else \threeformat \global\let\lr=A
 \fi\fi
 \ifnum\outputpenalty>-20000
 \else\dosupereject\fi}
%
\else\ifnum\count30=4
\output={\if A\lr
 \global\setbox\onecolumn=\columnbox
 \global\let\lr=B
 \else\if B\lr
 \global\setbox\twocolumn=\columnbox
 \global\let\lr=C
 \else\if C\lr
 \global\setbox\threecolumn=\columnbox
 \global\let\lr=D
 \else \fourformat \global\let\lr=A
 \fi\fi\fi
 \ifnum\outputpenalty>-20000
 \else\dosupereject\fi}
%
\else\ifnum\count30=5
\output={\if A\lr
 \global\setbox\onecolumn=\columnbox
 \global\let\lr=B
 \else\if B\lr
 \global\setbox\twocolumn=\columnbox
 \global\let\lr=C
 \else\if C\lr
 \global\setbox\threecolumn=\columnbox
 \global\let\lr=D
 \else\if D\lr
 \global\setbox\fourcolumn=\columnbox
 \global\let\lr=E
 \else \fiveformat \global\let\lr=A
 \fi\fi\fi\fi
 \ifnum\outputpenalty>-20000
 \else\dosupereject\fi}
\fi\fi\fi\fi\fi
%
```

```
% The output routines test for the
% starting-page value and do not put a
% running head on that page. All subsequent
% lefthand pages will get a running head.
% The running foot layout flips as you go
% from odd page numbers to even page numbers.
%
% The output routines also test to see if
% you want to put rules between the columns
% (\count71) and/or before and after the
% first and last column (\count70).
%
\def\oneformat{\shipout\vbox{
 \ifnum\startpage\blankheadline
 \else \ifodd\pageno\rightheadline
 \else \leftheadline\fi\fi
 \nointerlineskip
 \fullline{\ifnum\count70=1\openrule
 \else\hfil\fi
 \columnbox\ifnum\count70=1\closerule
 \else \ hfil\fi}
 \ifodd\pageno\rightfootline
 \else \leftfootline\fi
 \vfill}\advancepageno}
%
\def\twoformat{\shipout\vbox{
 \ifnum\startpage\blankheadline
 \else \ifodd\pageno\rightheadline
 \else \leftheadline\fi\fi
 \nointerlineskip
 \fullline{\ifnum\count70=1\openrule
 \else\hfil\fi
 \box\onecolumn\ifnum\count71=1\midrule
 \else\hfil\fi
 \columnbox\ifnum\count70=1\closerule
 \else \hfil\fi}
 \ifodd\pageno\rightfootline
 \else \leftfootline\fi
 \vfill}\advancepageno}
%
\def\threeformat{\shipout\vbox{
 \ifnum\startpage\blankheadline
 \else \ifodd\pageno\rightheadline
 \else \leftheadline \fi\fi
 \nointerlineskip
 \fullline{\ifnum\count70=1\openrule
 \else\back hfil\fi
 \box\onecolumn\ifnum\count71=1\midrule
 \else\back hfil\fi
 \box\twocolumn\ifnum\count71=1\midrule
 \else\back hfil\fi
 \columnbox\ifnum\count70=1\closerule
 \else \hfil\ fi}
 \ifodd\pageno\rightfootline
 \else \leftfootline\ fi
 \vfill}\advancepageno}
%
\def\fourformat{\shipout\vbox{
 \ifnum\startpage\blankheadline
 \else \ifodd\pageno\rightheadline
 \else \leftheadline\fi\fi
 \nointerlineskip
 \fullline{\ifnum\count70=1\openrule
 \else\hfil\fi
 \box\onecolumn\ifnum\count71=1\midrule
 \else\back hfil\fi
 \box\twocolumn\ifnum\count71=1\midrule
 \else\back hfil\fi
 \box\threecolumn\ifnum\count71=1\midrule
 \else\back hfil\fi
 \columnbox\ifnum\count70=1\closerule
 \else \hfil\ fi}
 \ifodd\pageno\rightfootline
 \else \leftfootline\fi
 \vfill}\advancepageno}
%
\def\fiveformat{\shipout\vbox{
 \ifnum\startpage\blankheadline
 \else \ifodd\pageno\rightheadline
 \else \leftheadline\fi\ fi
 \nointerlineskip
 \fullline{\ifnum\count70=1\openrule
 \else \hfil\ fi
 \box\onecolumn\ifnum\count71=1\midrule
 \else \hfil\fi
 \box\twocolumn\ifnum\count71=1\midrule
 \else \hfil\fi
 \box\threecolumn\ifnum\count71=1\midrule
 \else \hfil\fi
 \box\fourcolumn\ifnum\count71=1\midrule
 \else \hfil\fi
 \columnbox\ifnum\count70=1\closerule
 \else \hfil \fi}
 \ifodd\pageno\rightfootline
 \else\leftfootline \fi
 \vfill} \advancepageno}
%
```



VV2400
Article Name: The Stoppers

to the bullpen because they had lost their best stuff. Today's relievers, however, are a different breed: young, cocky, talented and flamboyantly successful. They run every step of the way from the bullpen to the mound to let the batter know they can't wait to face his pitiful self. Both they and the batter know theirs is the most important job on every major league team. And they are paid accordingly. Donnie Moore, the Angels' portly fastballer, is the highest paid on his club at $1 million a year; Dan Quisenberry, the Royals' laconic submariner, is second only to George Brett on his club. Greg Harris of the Rangers and Goose Gossage of the Padres are the second highest paid on their teams; Willie Hernandez of the Tigers is his team's third highest paid, and Dave Righetti of the Yankees is the fifth highest on his team.

They are paid all that money to stop the opposition's rallies. Often, they appear in less than 80 innings a season, and yet those are the most important 80 innings each team plays. The game is in the balance when they come in. Righetti, for example, appeared in 74 games last year, winning eight and saving 46, a major-league record. That means he had a significant influence on 54 of the second-place Yankees' 90 victories. "Without Rags," says Hall of Famer Whitey Ford, "we would have finished fifth."

In most cases, relievers have been bred to their task from their very first minor-league pitch. They have special talents and strange temperaments that make them unique among baseball players. They are ephemeral creatures noted more for brief flashes of brilliance rather than for plodding efficiency. They see their job in terms of a single inning at most, a single batter, even a single pitch. Often that's all they need, one special pitch.

Righetti, Moore, the Cubs' Lee Smith, and even Gossage, in his twilight years, rely primarily on exploding fastballs in a crunch. Quisenberry and the Phillies' Kent Tekulve deliver their ace in the hole—a rising, then sinking fastball—with an underhand delivery which, according to Quiz, "resembles the motion a mother uses to reach down and spank a baby." Hernandez was only a journeyman left-hander when he developed an almost unhittable screwball that made him equally effective against right-handed batters and left-handers. Roger McDowell, the Mets' young stopper, relies on a pitch that is both a good fastball and a great sinker. He typifies the new breed of relievers. He stands primly on the mound with his feet together, a pink-faced, unemotional youth who looks like a choirboy or maybe a Yuppie stockbroker going about his business as if the conclusion is foregone. A ground-ball double play.

"Relieving is a serious business now," says Righetti. "There's no more of that fooling around on the mound or in the bullpen like a few years ago." Righetti is referring to relievers of 10 years ago, who, according to Whitey Ford, "were a little flaky then." He is referring to the young Gossage who kept a pet goose in the bullpen, and to Sparky Lyle, who once appeared on the first day of spring training with a leg cast to his hip to terrify the Yankee brass. There was Al Hrabosky, the Mad Hungarian, with his Fu Manchu mustache and his stalking and imprecations on the mound, and, of course, there was the irrepressible Tug McGraw of the Phillies, who, when asked if he preferred Astroturf to grass, said, "I don't know, I never smoked Astroturf."

Such antics are a thing of the past for today's big-bucks relievers who go about their job in a more controlled manner.

"That doesn't mean they're not emotional types," says Rangers' manager Bobby Valentine. "They have to be. →

TV GUIDE JUNE 13, 1987 47

Figure 1. Page of **TV GUIDE** set using `COLRHRF.MAC`

Diglot Typesetting

by Charles Lehardy
The Summer Institute of Linguistics

This article describes TEX macros that print Scripture in parallel-column diglot form. The program has been used to typeset the New Testament in the San Bartolomé Zoogocho dialect of Zapotec, a language spoken in southern Mexico.

Diglot means "two languages". A diglot book contains a running translation of the primary text in some secondary language. In the context of our work with the indigenous peoples of Mexico, the primary text (the "idiom") is a language such as Zapotec, and the secondary text (the "diglot") is Spanish. Printing the two languages side-by-side serves a number of purposes. It gives them equal status, sometimes helping to settle questions about the "legitimacy" of one language or the other. It helps speakers of one of the languages make comparisons to the other. And it has the practical effect of producing shorter lines, which is important to those who are not skilled readers.

Traditionally, our parallel-column diglot versions of the New Testament were done largely by hand. The idiom and the Spanish were typeset in two separate runs. The pages were created by cutting the appropriate strips of the idiom and diglot copy and pasting them side by side. As might be expected, the process was slow and quite expensive.

Because of a growing interest in diglot New Testaments, we have created TEX macros that page the two languages together automatically. What follows is a simplified description of how the **diglot.tex** program does its work.

The basic feature of TEX being exploited by **diglot.tex** is its ability to read from and/or write to auxiliary files while typesetting. The Zoogocho Zapotec book of Matthew is contained in a single 325k-byte file. At the beginning of chapter one, the file contains the command `\df{svpmat1}`. This tells TEX that the diglot material to be used with this portion of the idiom is found under the filename **svpmat1.tex**.

The `\df{`*diglot-filename*`}` command opens the auxiliary input channel `\diglotfile` and searches for **svpmat1.tex**. The Spanish Versión Popular edition of Matthew is a 200k-byte file. We've found that smaller files are easier to handle, so we use chapter-sized files for the diglot text, which for Matthew average a comfortable 7k-bytes each. A `\df` command placed in the idiom text at the beginning of each chapter calls up the appropriate diglot file as typesetting progresses. After `\diglotfile` is opened, verse processing begins.

Each time a new idiom verse is encountered, several things happen. First, the verse number is saved. Verses are usually marked by a single number, but sometimes the translator elects to create a synthesis of two or more verses, in which case the verses are marked by two numbers separated by a comma or hyphen (*e.g.*, 12–14). The verse command has the form `\iv` $m.n$, where m is the starting verse number and n is the (optional) ending verse number. m is compared to the current verse number in an effort to guard against missing or duplicated verses. If all is well, the `\getdiglot` routine is called.

The listing below is a simplified version of the `\getdiglot` routine. TEX treats each verse in `\diglotfile` as a unit. `\getdiglot` pulls a verse from `\diglotfile`, increases the value of `\diglotverse` by one, makes an insert of the diglot verse, and compares the verse number of the idiom text with the diglot. If the idiom verse was a synthesis of several verses, the routine repeats until the verse counts for both the idiom and the diglot match. When `\getdiglot` has finished inserting the diglot material, the verse routine contributes the idiom text to the current page.

Pages are constructed using the routines called "marginal hacks" in the *TEXbook* (pp. 415–416). The `\pagecontents` macro has been altered as shown below to place the diglot material

```
% Get a verse from the diglotfile and make an insertion of it
\def\getdiglot{\loop\read\diglotfile to \diglottext
    \ifeof\diglotfile\closein\diglotfile
    \else
        \advance\diglotverse by 1
        \insert\diglotins{\diglottext}
    \fi
    \ifnum\diglotverse < \lastverse
    \repeat}
```

on the page as an insertion, much like a footnote. TEX really sees the page as a fairly narrow column of text with an extraordinarily large right margin into which we are dumping the contents of \diglotins.

```
% Redefine pagecontents to allow for diglotins
\def\pagecontents{
    \hrule width\pagewidth
    \global\bardepth = \ht\diglotins
    \rlap{\kern\hsize\tbar
        \vbox to 0pt{\vglue 6pt\unvbox\diglotins \vss}}
    \ifvoid\topins\else\unvbox\topins\fi
    \global\pagedim = \dp255 \unvbox255
    ...}
```

The new \pagecontents macro draws a horizontal line just below the header. It then stores the height of \diglotins which is used to draw the vertical line between the columns. Next, it kerns to the right \hsize (the width of the idiom text column) and prints the vertical line (using the \tbar macro) followed by the diglot text. After the diglot text is printed, \pagecontents continues as it would under normal conditions.

Because of the narrow columns used, line-breaks can become difficult. We help the program along by allowing it to stretch the white space on a line a bit more than usual. We often avoid hyphenation and justification because whole words and ragged margins are a help to beginning readers. The Zoogocho Zapotec New Testament is an example of this style. Some languages, however, have much longer words and are impossible to typeset without hyphenation. Had hyphenation been required, it would have been necessary to hyphenate the idiom and diglot texts simultaneously using two sets of rules. Unfortunately, TEX was not designed to hyphenate two different languages at the same time. TEX's hyphenation rules (for English) could be changed to work correctly for Zapotec, but then the Spanish text would also be hyphenated using Zapotec rules.

The easiest solution seems to be to combine the use of discretionary hyphens with penalties that make hyphenation somewhat limited. Before typesetting, we run both texts through programs that insert discretionary hyphens in each word according to the rules of that particular language. In any word containing a discretionary hyphen, TEX will suspend its hyphenation rules and break the word where the user has indicated. This works, but we are looking for other solutions.

The sample shown here is a page from the Zoogocho Zapotec Matthew. Each column has attributes that can be altered independently of the other. These include: column width, typesize, typestyle, leading, hyphenation (on or off) and justification (on or off).

yodaoˈ c̱hegaquenˈ. Naˈ ḻeca besyebandeˈ
len xtiženˈ naˈ goseˈ ḻježeˈ:

—¿Ga jasedeˈ yeḻaˈ sinaˈ? naˈ
¿nacxec̱henˈ chac choneˈ yeḻaˈ guac
caˈ? **55** Nombiaˈchoneˈ, ḻe naqueˈ xiˈiṉ
benˈ chonšagüeˈ yag. Naˈ xṉeˈenaˈ lieˈ
María, naˈ Jacobo, José, Simón naˈ
Judas zjanaqueˈ beṉeˈ bišeˈ. **56** Naˈ ḻecze
beṉeˈ zaneˈ caˈ niteˈ lažcho nga. ¿Nacxe
chaquenˈ choˈe dižaˈ caˈ naˈ chacteˈ choneˈ
yeḻaˈ guac?

57 Naˈ danˈ goson eˈ xbab nac
Jesúsenˈ con to beṉeˈ gualaž c̱hegaqueˈ, bi
gosaclažeˈ yesejḻeˈe c̱heˈ. Naˈ gož Jesúsenˈ
ḻegaqueˈ:

—Yogoˈze beṉeˈ chonḻaˈaṉeˈ beṉeˈ
choeˈ xtižaˈ Diosenˈ, perw beṉeˈ
gualaž c̱heˈ naˈ beṉeˈ ḻoˈ yoˈo c̱heˈ bi
chesonḻaˈaṉeˈ ḻeˈ.

58 Naˈ to c̱hopze yeḻaˈ guac ben
Jesúsenˈ Nazaretenˈ danˈ bitw gosejḻeˈe
c̱heˈ.

Quinganˈ goc gosoteˈ Juan benˈ bzoa beṉeˈ nis

(Mr. 6.14-29; Lc. 9.7-9)

14 Naˈ ca tiempenˈ Herodes benˈ
naqueˈ gobernador c̱he distritw
Galileanˈ, bendeˈ dižaˈ ca nac yogoˈḻoḻ
danˈ chon Jesúsenˈ. **2** Nach gožeˈ xmoseˈ
caˈ:

—Benganˈ daˈ Juanṉaˈ, benˈ bzoa
beṉeˈ nis. Ba bebaneˈ ladjo beṉeˈ guat
caˈ, daˈnanˈ chaqueˈ choneˈ yeḻaˈ guac.

3-4 Herodesenˈ gwneˈ caˈ danˈ beneˈ
mandadw gosoteˈ daˈ Juanṉaˈ. Quinganˈ
goc: Herodesenˈ bequeˈe Herodías ca
xoˈoleˈ ḻaˈczḻaˈ Herodías naqueˈ xoˈole
beṉeˈ bišeˈ Felipe. Naˈ Juanṉaˈ gožeˈ
Herodesenˈ:

—Bi cheyaḻaˈ soalen noˈol c̱he beṉeˈ
bišonˈ.

55 Pues este es el hijo
del carpintero, y su madre
es María. Es hermano de
Jacobo, José, Simón y
Judas,
56 y sus hermanas también
viven aquí entre nosotros.
¿De dónde, pues, sabe todo
esto?

57 Por eso no quisieron
hacerle caso. Pero Jesús les
dijo:

—Todos aprecian a un
profeta, menos los de su
propia tierra y los de su casa.

58 Y no hizo muchos milagros
allí, porque ellos no
creían en él.

La muerte de Juan el Bautista

(Mr. 6.14-29; Lc. 9.7-9)

14 **1** *En ese tiempo* Herodes,
el que gobernaba en
Galilea, tuvo noticias de
Jesús,
2 y dijo a los que estaban
con él:

—Ese es Juan el
Bautista, que ha sido resucitado
de la muerte. Por eso
tiene este poder milagroso.

3 Es que Herodes había
hecho detener a Juan y llevarlo
atado a la cárcel. Lo
hizo por causa de Herodías,
que era esposa de su hermano
Felipe.
4 Juan había dicho a
Herodes:

—No debes tenerla como
tu mujer.

First Line Special Handling with TEX

Anne Brüggemann-Klein
Institut für Angewandte Informatik
und Formale Beschreibungsverfahren
Postfach 6980
7500 Karlsruhe, West Germany

Jim Sterken in TUGboat 4, no. 2, and James Alexander in TUGboat 7, no. 2, ask a question about fancy first line processing: Can TEX automatically set the first line of a paragraph in a special font? Yes, indeed, TEX can! But before I tell you a solution, let me describe some wrong ways I tried before, because this explains some assumptions and restrictions I made. For short, I call paragraphs which have their first lines set in a special font, *special* paragraphs, in contrast to the *normal* ones.

The first line problem in its strongest form means the following: Can TEX make *optimal* special paragraphs in the same sense as it does optimal line breaking for normal paragraphs? In my opinion, the only solution for this strong form is to alter TEX's line breaking algorithm, but this is strictly opposite to the spirit of the TEX community.

Therefore I give a somewhat weaker formulation of the first line problem: Can TEX find an acceptable breakpoint for the first line of a special paragraph and then do optimal linebreaking as usual for the rest of it? The question is now: How can you find an acceptable breakpoint for the first line of the paragraph?

My first idea was to impose this job on TEX's line breaking algorithm: First let TEX set the whole paragraph in the special font for the first line and store it in a box register. Then take the first line of this box as the first line of the paragrah, "unbox" the rest of the box and set it again, but in the normal font.

This nice idea didn't work, and the reason is that TEX's digestive process is a one-way street. There is no problem to set the whole paragraph in the special first line font and store it in a box register, and you can easily detract the first line from this box by a `\vsplit`-command. It's somewhat harder to "unbox" the rest of the box by the help of the commands `\vsplit`, `\lastbox`, and `\unhbox`. But even suppose you can manage this, all what you get is a list of character boxes, and there is no way to change this back into a token list in order to set the stuff a second time, but in *another* font.

So I had to help TEX in finding the breakpoint for the first line. My idea was to put one word after another into an `\hbox`, using the special font for the first line, until — after some stretching or shrinking — the width of this box matches the `\hsize`. My intention was to regulate the amount of the stretching or shrinking by testing the "badness" of the according line. But unfortunately the badness of a line is only *reported* in your log-file. You cannot *use* it as an internal parameter inside of TEX.

Therefore, I had to calculate explicitly whether the box fits into a line by itself. For this purpose, I had to calculate the total stretchablility and shrinkability of the box. By the help of `\sfactor` and `fontdimen3` and `4` this can be done.

But I decided to make things easier. To me it seems not necessary to do tricky spacing in a heading-like special first line. Therefore I set the first line in a `\frenchspacing` style. Then I only had to count the spaces in the box and multiply `\fontdimen3` and 4 of the first line font by this factor to obtain the stretchability and shrinkability of this box.

Now the algorithm which finds an appropriate breakpoint for the first line works as follows. One word after the other is appended to a box until a feasible breakpoint is found. When a word has been appended, a test is made whether the box fills a line: If the width of the box is smaller than the `\hsize`, we have two cases: In the first case the box cannot be stretched to the `\hsize`. In this case the box is too short, i.e. another word must be appended and the process is iterated. In the second case, when the box can be stretched to the `\hsize`, this box will be the first line of the paragraph. In this case the glue set ratio is less or equal 1, i.e. the badness of the line is less or equal 100.

If, on the other hand, the width of the box has become greater than the `\hsize`, there are two cases again: In the first case, the box can shrink to the `\hsize`, and again we have a badness less or equal 100 if we take this box as the first line of the paragraph. In the second case the box is too large to shrink to the `\hsize`. In this case, the last word of the box is removed, and the rest of the box is my first line. Then the badness of the line is greater than 100, and it might be an underfull one.

The algorithm prefers to stretch the first line instead of shrinking it, which fits well to the heading-like character of the application. It seems better to me to stretch the first line than to hyphenate the last word of it (hyphenation could be included, but the user had to indicate possible hyphenation points in the input). Furthermore, there is no problem to multiply the stretchability and shrinkability of the first line by a factor to allow for a higher badness.

Some features of TEX don't work in the context of the macros described here. One of these is migrating material from the first line (`\vadjust`, e.g.). Special caution is necessary with macros. Spaces inside of macros and their arguments must be protected by grouping symbols. Some macros, for example the `\verbatim`-macro from the TEXbook, don't work at all near the beginning of a special paragraph, because the token converting process is disturbed. Finally, my macros don't work if the paragraph doesn't take at least two lines of text.

The "user interface" of the macros is very simple. You can use the commands `\firstlinefont{<filename>}` to define the font for the first line of a special paragraph (default: `\firstlinefont{cmcsc10}`) and `\firstlineindent{<dimen>}` to define its indentation (default: `\firstlineindent{0pt}`). With `\iffirstlineinner` you can test whether you are in the first line of a special paragraph or not, and `\firstlinespecial` in vertical mode starts a special paragraph. `\firstlinespecial` works also inside an `\everypar`-command, but be shure to remove the indentation box first.

I **f you are a mathematician (like me), these macros can add a playful or narrative** element to your severe subject. But if you are a philologist, why don't you improve the aesthetic effect by combining fancy first line printing with fancy printing of initials?

```
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%                                                          %%%%%
%%%%%           FIRST LINE SPECIAL HANDLING WITH TeX           %%%%%
%%%%%                                                          %%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% These macros can set the first line of a paragraph
% in a special font. You can determine this font by the
% command \firstlinefont{<filename>}.
% The default value is \firstlinefont{cmcsc10}.
% You can define the indentation of such a paragraph by
% the command \firstlineindent{<dimen>}. The default value is
% \firstlineindent{0pt}.
% There is a if-register \iffirstlineinner, which is true if and
% only if you are in the first line of a special paragraph.
% Then start the paragraph with the command \firstlinespecial
% in vertical mode.

\catcode`\@=11 % `@' is used as a letter to hide command names
               % from the user.

%%% Allocate some internal registers:
%%% The box \oldb@x contains the material which has been
%%% collected for the first line so far.
%%% \newb@x tries to append another
%%% word to \oldb@x.
%%% The counter \spacec@unt counts the number of interword spaces
%%% in \oldb@x.
%%% The dimen-register \firstlineind@nt stores the indentation of
%%% the special paragraphs.
%%% The if-register \iffirstw@rd stores whether the first word
%%% of the special line has been read already or not.
%%% The dimen-registers \sp@cestretch and \sp@ceshrink store
%%% the strechability and the shrinkability of the normal interword
%%% space in the font \firstlinef@nt.
%%% The dimen-register \boxw@dth stores the width of the box \newb@x.
%%% The if-register \iffirstlineinner is true iff you are in the
%%% first line of a special paragraph.

\newbox\oldb@x
\newbox\newb@x
\newcount\spacec@unt
\newdimen\firstlineind@nt
\newif\iffirstw@rd
\newdimen\sp@cestretch
\newdimen\sp@ceshrink
\newdimen\boxw@dth
```

```
\newif\iffirstlineinner \firstlineinnerfalse

%%% \firstlinefont determines which font has to be used
%%% for the first line of a paragraph.
%%% This font gets the internal name \firstlinef@nt.
%%% Default is \firstlinefont{cmcsc10}.
%%% \firstlineindent defines the indentation of a paragraph which
%%% has been started with \firstlinespecial.
%%% This dimension is stored in the register \firstlineind@nt.
%%% Default is \firstlineindent{0pt}.

\def\firstlinefont#1{\font\firstlinef@nt#1}
\firstlinefont{cmcsc10}
\def\firstlineindent#1{\firstlineind@nt #1}
\firstlineindent{0pt}

%%% \firstlinespecial initializes some registers, stores the current
%%% font, switches to \firstlinef@nt,
%%% and starts the command \n@xt which does the real work.

\def\firstlinespecial{%
     \firstlineinnertrue
     \setbox\oldb@x\hbox{\hskip\firstlineind@nt}%
     \setbox\newb@x\copy\oldb@x
     \spacec@unt 0
     \firstw@rdtrue
     \xdef\oldf@nt{\the\font}%
     \firstlinef@nt
     \sp@cestretch\fontdimen3\firstlinef@nt
     \sp@ceshrink\fontdimen4\firstlinef@nt
     \n@xt}

%%% The command \append copies \oldb@x to \newb@x and appends
%%% its argument to \newb@x, preceeded by a space, depending on
%%% \iffirstw@rd.

\def\app@nd#1{%
     \setbox\newb@x\hbox{\unhcopy\oldb@x
     \iffirstw@rd\else\space\global\advance\spacec@unt by 1\fi
     #1}\firstw@rdfalse}

%%% The command \n@@xt is the heart of the whole. At the
%%% beginning of a \firstlinespecial-command, \n@xt has the
%%% same meaning as \n@@xt. When \n@@xt comes to work, \oldb@x
%%% contains the material which has been collected so far for the
%%% first line of the paragraph, i.e.~\oldb@x is an initial part of
%%% the first line.
%%% \n@@xt reads the next word of the paragraph and appends it to
%%% \newb@x, which is a copy of \oldb@x. As its last command, \n@@xt
%%% calls in \n@xt again, which iterates the process.
%%% But how to stop the process? After \n@@xt has
%%% appended a word to \newb@x, it makes a test: If the
%%% width of \newb@x is still less than \hsize we have two cases:
%%% In the first one the line is still too short and the process must
%%% be iterated. In the second one, the interword glue of the box
%%% can be stretched to match \hsize. In this case we can take
%%% \newb@x as the first line of the paragraph, and \n@xt must be
%%% redefined to stop the process.
%%% On the other hand, if the width of \newb@x has become greater or
%%% equal to \hsize, we can try to shrink \newb@x to \hsize.
%%% If \newb@x doesn't shrink to
%%% \hsize, we have to remove the last word from \newb@x. In fact, in
%%% this case we return to \oldb@x and take it as the first line.
%%% Here you may get a message about an underfull \hbox. Then
%%% the second line starts with the last word and \n@xt is
%%% redefined as before.

\def\n@@xt#1 {% Note that the argument is delimited by a blank!
     \app@nd{#1}%
     \boxw@dth\wd\newb@x
     \ifdim\boxw@dth < \hsize
          \advance\boxw@dth by \spacec@unt\sp@cestretch
          \ifdim\boxw@dth < \hsize
                % \message{1 }% not yet full enough
                \else
                \message{2 }% stretchable to \hsize
                \noindent\line{\unhbox\newb@x}\penalty -10000
                \def\n@xt{\firstlineinnerfalse\let\n@xt\n@@xt}%
                \oldf@nt
                \fi
          \else
          \advance\boxw@dth by -\spacec@unt\sp@ceshrink
          \ifdim\boxw@dth > \hsize
                \message{3 }% last word in next line, shrinkable
                \noindent\line{\unhbox\oldb@x}\penalty -10000
                \def\n@xt{\firstlineinnerfalse\let\n@xt\n@@xt}%
                \oldf@nt
                #1 % keep this blank!
                \else
                \message{4 }% shrinkable to hsize
                \noindent\line{\unhbox\newb@x}\penalty -10000
                \def\n@xt{\firstlineinnerfalse\let\n@xt\n@@xt}%
                \oldf@nt
                \fi
          \fi
     \setbox\oldb@x\box\newb@x
     \n@xt}
\let\n@xt\n@@xt
\catcode`\@=12
```

LaTeX

Contents of LaTeX Style Collection as of 17th June 1987

Ken Yap
University of Rochester

The LaTeX style collection now contains the files listed below. They are available for anonymous ftp from Rochester.Arpa in directory public/latex-style. You should retrieve the file 00index first to obtain a brief description of current directory contents.

File	Description
00directory	
00index	
00readme	
a4.sty	Set page size to A4
a4wide.sty	Adjusts width too to suit A4
*aaai-instructions.tex	Instructions to authors
*aaai-named.bst	BibTeX style to accompany aaai.sty
*aaai.sty	Style file for AAAI conference 1987
acm.bst	ACM BibTeX style
agugrl.sty	AGU Geophysical Research Letters style
agujgr.sty	AGU Journal of Geophysical Research style
amssymbols.sty	Load AMS symbol fonts
biihead.sty	Underlined heading
*ctex.readme	Notes on ctex
*ctex.shar1	Sources to Pat Monardo's Common-TeX in C
...	
ctex.shar9	9 shar files
cyrillic.sty	Load cyrillic font
*dayofweek.tex	Macros to compute day of week and phase of moon
	Examples of how to use TeX arithmetic capabilities
deproc.sty	DECUS Proceedings style
deprocldc.tex	Paper that describes the above
docsty.c	Program to convert .doc to .sty by stripping comments
docsty.readme	
doublespace.sty	Double spacing in text
drafthead.sty	Prints DRAFT in heading
dvidoc.shar1	Sh archive of DVIDOC, DVI to character device filter for Unix BSD systems
dvidoc.shar2	
*dvidoc.sty	Style file to substitute all fonts with doc font
epic.shar1	Sh archive of extended picture environment
epic.shar2	
format.sty	Print FP numbers in fixed format
fullpage.doc	Get more out of a page
fullpage.sty	
geophysics.sty	Geophysics journal style
ieeetr.bst	IEEE Transactions BibTeX style
*ist21.sty	IST21 document style option for cover page
*latex.bug	latest listing of bugs found in LaTeX
layout.readme	Prints nice diagram showing page parameters
layout.tex	
lcustom.tex	Useful macros and definitions for LaTeX
lfonts_ams.readme	Use AMS symbols in LaTeX
lfonts_ams.tex	
lgraph.shar	Sh archive of data to graph command filter in Pascal
*local.suppl	Supplement to local guide; describes a4, tgrind, sfwmac, trademark, lcustom, and vdm
natsci.bst	Natural sciences generic BibTeX style
newalpha.bst	Modified alphabetic BibTeX style
nopagenumbers.doc	Remove page numbers
nopagenumbers.sty	
remark.sty	like newtheorem but no \it
*sc21.sty	ISO/TC97/SC21 document style
*sc21-wg1.sty	option for cover page
*sfwmac.sty	Useful macros for Unix documentation
siam.bib	SIAM BibTeX style
siam.bst	
siam.doc	SIAM LaTeX style
siam.sty	
siam.tex	
siam10.doc	
siam10.sty	
siam11.sty	
siam12.sty	
slem.doc	Change \sl to \em
slem.sty	
spacecites.doc	Modified to give spacing between citations
spacecites.sty	
suthesis.doc	Stanford U thesis style
suthesis.sty	
texindex.doc	Style file and processor for index entries. Works under VMS.
texindex.pas	
texindex.sty	

texnames.doc	Define a couple more TEX names
texnames.sty	
*tgrind.sty	Tgrind macros for LATEX instead of TEX
threepart.sty	Three part page headers
*trademark.sty	Definitions of common trademarks
uct10.doc	U of California thesis style
uct11.doc	
uct12.doc	
ucthesis.doc	
ucthesis.readme	
vdm.doc	Vienna Development Method LATEX style
vdm.sty	
vdm.tex	
ws87.p	Wordstar 8 bit filter
wsltex.c	Wordstar to LATEX filter, C version
wsltex.p	Wordstar to LATEX filter
*xxxcustom.tex	Supplementary macros for xxx-tex, for some xxx
*xxxslides.sty	Supplementary macros for SLITEX, includes slides.sty

New entries since the last TUGboat listing are marked with an *. More submissions are very welcome. Send them to

Ken
LaTeX-Style@Rochester.Arpa
LaTeX-Style@cs.rochester.edu
..!rochester!latex-style

Editor's note: People sending future submissions should note that some gateways to Bitnet strip off everything beyond 80 columns, and perhaps corrupt some other data as well (ASCII tabs may or may not remain intact). Please structure your file so that it will survive.

For Internet users: how to ftp

An example session is shown below. Disclaimer: ftp syntax varies from host to host. Your syntax may be different. The syntax presented here is that of Unix ftp. Comments in parentheses.

Non-Internet users: how to retrieve by mail

An archive server for LATEX files has been installed. Send a piece of mail to LaTeX-Style (@rochester.arpa, @cs.rochester.edu, via uucp or your favourite gateway) in the following format.

- Subject line should contain the phrase "@file request".
- The body of the mail should start with a line containing only an @ (at) sign.

Important! The first line following the "at" line should be a mail address **from** Rochester **to** you. (Undeliverable mail will be silently dropped on the floor.)

- Follow your return address by the names of the files you want, either one to each line, or many to each line, separated by spaces.
- End with a line containing only an @ sign.
- Case is not significant.

For example, if you are user at site.bitnet, this is what you should send:

```
To: latex-style@rochester.arpa
Subject: @file request

@
user%site.bitnet@wiscvm.wisc.edu
00readme
00index
@
```

A word to the wise: it is best to fully qualify your mail address. Our mailer knows about some gateways but not all. Examples:

```
user%site.bitnet@wiscvm.wisc.edu
user%site.csnet@relay.cs.net
```

Sample FTP session for Internet users

```
% ftp cayuga.cs.rochester.edu   (a.k.a. rochester.arpa, a.k.a. 192.5.53.209)
 ...                            (general blurb)
user: anonymous
password: <any non-null string>
ftp> cd public/latex-style      (where the files are)
ftp> ls                         (to see what is there)
 ...                            (lots of output)
ftp> get 00index
 ...                            (more blurb)
ftp> quit
```

Do not include any messages in the mail. It will not be seen by human eyes. Be patient as the server is actually a batch program run once a day. Files will be sent in batches, each not exceeding 100kbytes in size.

IBM PC and clone users: how to get a distribution

David Hopper of Toronto, Canada, is offering copies of the style collection on diskettes. This is not a commercial enterprise. David is doing this in his own time as a favour to the TEX community. The entire set of style files, not including the C-TEX files, as of June 1st, fits on one 1.2 MB diskette or three 360KB diskettes. No subsetting, please. Send David

1. Formatted diskettes,
2. Indication of the format required,
3. A self-addressed mailer, and
4. A $5.00 donation per set of files, to cover postage and equipment wear & tear. (If you live outside North America, airmail delivery will probably require more postage. You should probably contact David for details.)

David's address:

David W. Hopper
446 Main Street
Toronto, Ontario
Canada M4C 4Y2
Thanks, David.

Editor's note: Traffic on the network servers and gateways has been very high recently, and in order to provide improved service, there have been some volunteers to maintain local "slave" repositories of the LATEX style collection. There is usually a geographic or network restriction requested, since the idea is to cut down traffic, not add to it. The following areas will be covered by the volunteers listed.

- Bitnet users: Texas A&M maintains a list- and file-server which is already handling (with TEX-L) much of the Bitnet distribution of TEXhax. An inquiry via listserv will retrieve a list of all TEX-related files:
  `tell listserv at tamvm1 get tex filelist`
- United Kingdom, for users of JANET or uucp: Stephen Page, `sdpage@uk.ac.ox.prg` or `...!ukc!ox-prg!sdpage`
- European users of BITnet: Christoph Gatzka, `zrgc002@dtuzdv5a.Bitnet`

Additional volunteers should contact Ken.

Automated Index Generation for LaTeX

Richard L. Aurbach
Monsanto Company
St. Louis, Missouri

Abstract

LaTeX includes partial support for the generation of an Index in a document. It contains commands which enable index terms and the pages on which they appear to be captured in an auxiliary file. However, special processing (external to LaTeX itself) is required to translate this information into a pleasingly-formatted index.

The IdxTeX program provides this additional processing, and generates a file of LaTeX source which may be included in a document to produce the desired index.

This paper describes how IdxTeX provides a full range of services for index generation and discusses issues related to the development of programs which provide auxiliary processing for LaTeX documents.

The IdxTeX Project

The LaTeX text formatting program is an extremely versatile tool for the generation of high-quality documents. However, its handling of an Index is incomplete. LaTeX provides the `\index` command which (in conjunction with the `\makeindex` command) generates an auxiliary file containing index terms and references to the pages on which the terms appear. However, it is left to the writer to develop this information into an appropriately formatted Index.

As part of a project to develop new document styles, I became interested in the automation of this process of Index generation and developed the IdxTeX program[1] to complement the capabilities of LaTeX. The IdxTeX project had several goals:

- to provide a fully-automated mechanism for Index generation which produces an Index with the same level of quality as the LaTeX document in which it appears.

- to help make indexing sufficiently easy to encourage authors to build effective and helpful indices into their documents.

- to provide a full set of indexing features, so that even complex indices (such as the Index of *The TeXbook*[2]) could be generated.

[1] The IdxTeX program has been submitted to both the TeX Users Group and to the DECUS Library for distribution to interested parties. The distribution includes a Users Guide, which describes how to use the program, an executable VAX/VMS image, and complete sources in the C language. Since the program uses VAX/VMS services, it will only run in that environment. However, I believe that it could be ported to other environments with modest effort.

[2] Knuth, Donald E., *The TeXbook*, Addison-Wesley and the American Mathematical Society, 1984.

- to provide support for all of the indexing capabilities inherent in LaTeX, such as three-level indexing, without requiring any additional enhancements to LaTeX itself.

- to include support for the generation of a Master Index for a set of documents.

The Indexing Problem

Generating an index in the LaTeX context provides a number of interesting challenges.

Index Levels

LaTeX supports a three-level index (items, subitems, and subsubitems). However, the `\index` command accepts only one argument. It is necessary to adopt a convention within the text of its argument to specify the level of the term being indexed. In IdxTeX, the `>` symbol is used to separate the item from the subitem and the subitem from the subsubitem.

Spelling

Obviously, the index must appear in alphabetical order. However, it must be possible to include LaTeX commands within an index term, to optimize the visual appearance of the Index. That is, an author should be able to specify "`\index{{\em Special\/} Commands}`", for example, and have the index item appear as expected. This means that IdxTeX must understand LaTeX syntax, so that the term can be properly placed in alphabetic order.

Page Ranges

If an item is indexed on a series of consecutive pages, the index entry should display the range of pages, rather than a list of consecutive numbers. That is, an item which is indexed on pages 11, 12, and 13, for example, should appear in the index with a page reference of 11–13.

Cross References

It is not uncommon to see an item in an index which refers to one or more other items in the index. To support this, syntactic conventions in the `\index` command and special processing are necessary.

Master Index

To generate a Master Index, the program must be able to process more than one auxiliary file, and keep track of which volume of the volume set is associated with each item. The output of the program must include labels which identify the volume associated with each index item.

The following sections provide insights into how each of these issues was resolved in IdxTeX.

Indexing Conventions

The LaTeX `\index` command takes a single argument. In an automatic index generation environment, that argument represents the only mechanism by which the author can communicate information to IdxTEX about how the term should be handled. To allow for the multitude of index features supported, it was necessary to impose a set of conventions on the use of this command.

Two principles were important to the design of these conventions:

1. The conventions should be (as much as possible) mnemonic, so that they are easy to remember.
2. The conventions should be easily recognizable as such. That is, the program must be able to distinguish unambiguously between characters which are used as part of a convention and characters which are part of the term being indexed.

Conventions were chosen which are not valid LATEX syntax — they would generate LATEX errors if they occurred naturally. Since IdxTEX is sensitive to LATEX syntax, this assures that there will be no cases in which IdxTEX confuses a part of its conventions for legitimate text entry.

The following conventions are used in IdxTEX.

Level Separators

The > character is used to separate items from subitems and subitems from subsubitems.[3] For example,

```
\index{Aaa>Bbb>Ccc}
```

specifies an index entry with an item of "Aaa", a subitem of "Bbb", and a subsubitem of "Ccc". Of course,

```
\index{Aaa>Bbb}
```

or

```
\index{Aaa}
```

are also acceptable.

Page Reference Highlights

The first character of an index entry may be used to specify special formatting for its page reference. The following table lists the capabilities which are available.

Format	**Meaning**	**Example**
`\index{^Foo}`	**boldface**	Foo, **11**
`\index{_Foo}`	underline	Foo, 11
`\index{~Foo}`	*italics*	Foo, *11*
`\index{#Foo}`	"and following"	Foo, 11ff

Cross References

Cross references are specified using the & character. For example,

```
\index{Aaa&Bbb}
```

will generate a cross reference of the form

"Aaa, *see* Bbb"

[3] Note that the RUNOFF text formatting system uses the same convention. Since we expected to convert a number of RUNOFF documents, this choice was obvious.

Cross reference processing allows for a combination of real page references and cross references, so that a combination of entries such as

```
\index{Aaa} \index{Aaa&Bbb}
```

will generate

“Aaa, 11, *see also* Bbb”

Master Index

Master index processing uses a new type of auxiliary file to provide the information IdxTEX needs to understand which document indices to use when building the Master Index and what labels to use when displaying information from different volumes. This will be discussed in more detail below.

Data Structures

The Index of a large document or the Master Index of a large document set may be quite extensive. To avoid limitations on the number of items which IdxTEX could handle, all internal data structures are allocated from dynamic memory. Therefore, the size of an Index is limited only by the user's virtual page quota.

Since the three-level structure of the index implies a tree-like organization, the basic data structures selected for internal storage of index information in IdxTEX were linked lists. While linked lists are not optimally efficient in this application, their simplicity compensates for the minor loss of performance.[4]

The basic data structure for each index item, subitem, or subsubitem is called a NODE. Using the notation of the C language, a NODE can be defined as

```
typedef struct node
    {
    struct node             *link;
    struct dsc$descriptor   item;
    struct dsc$descriptor   spell;
    struct node             *subhead;
    struct pgnode           *pghead;
    struct pgnode           *cfhead;
    } NODE;
```

In this structure, *link* is the forward linkage pointer to the next node in the list; *item* is a VAX/VMS dynamic string descriptor[5] which describes the text string associated with the index item; and *spell* is another string descriptor for the *spell-string*. The *spell-string* is used when alphabetizing index

[4] Since IdxTEX is not run often, its cost is an inconsequential fraction of the total cost of generating a document.

[5] VAX/VMS dynamic strings were used (rather than the ASCIZ strings which are more natural in a C-language implementation) because the VMS services which work with them handle all details of dynamic memory allocation and deallocation.

entries and helps resolve the spelling problems discussed previously. It is discussed in more detail below.

The *subhead* is the listhead for a linked list of NODEs for any subitems associated with this index item. The recursive nature of this data structure made handling the three levels of indexing simple.

The *pghead* and *cfhead* variables are listheads for linked lists of PGNODE structures. Each PGNODE structure includes information about a single reference to the particular index entry. The list chained from *pghead* contains numeric page references, while the list chained from *cfhead* contains cross references.

Using C language notation, a PGNODE has the following structure

```
typedef struct pgnode
    {
    struct pgnode          *link;
    struct dsc$descriptor  *vol;
    struct dsc$descriptor  page_dsc;
    char                   highlight;
    } PGNODE;
```

Once again, *link* is the forward linkage pointer for the linked list. The *vol* variable is used in Master Index processing to point to the dynamic string descriptor for the label to be associated with the volume from which the reference came. The *page_dsc* describes the page reference string, while *highlight* is a flag used to indicate what type of page reference highlighting is associated with this page reference.

One virtue of this type of internal data organization is that each distinct item, subitem, or subsubitem uses only a single NODE structure. If the entry has a number of page references, then one PGNODE structure (which is fairly small) is used for each. If more than one index reference occurs on the same page, only a single PGNODE is allocated. This approach conserves dynamic memory.

NODEs are linked together in alphabetical order (by *spell-string*). PGNODEs for numeric page references are linked together in the order they appear in the auxiliary file produced by LaTeX, which automatically puts them into numerical order. PGNODEs for cross references are linked together alphabetically. This means that the internal representation of the index is built in sorted order, simplifying back-end processing.

Spell Strings and Alphabetization

As discussed previously, putting index entries into alphabetical order is a complex task, because the entry may contain LaTeX commands which are meant to enhance the visual appearance of the index, but which must not be included when the term is placed in alphabetical order. In IdxTeX, the concept of a *spell-string* was introduced to handle this problem.

The basic idea is that each NODE of the internal data structure contains descriptors for two copies of the index entry — the *item* and the *spell-string.*

- The *item* string contains the original form of the index entry, including all LaTeX commands. It is used to generate the formatted output and is not used when placing the entry in proper alphabetical order.

- The *spell-string* originally contains a copy of the index entry. However, during spelling processing, it is modified to remove everything which should not be included when the entry is placed in alphabetical order. It is not used for any other purpose.

Therefore, spelling processing consists of a number of steps which recognize various forms of LaTeX syntax and remove them from the *spell-string*. After this has been done, the *spell-string* is in a form suitable for alphabetizing the index entry, while the *item* string remains untouched.

In some special documents, it may be desirable to place TeX or LaTeX commands themselves in the index.[6] To accommodate this possibility, spelling processing skips any text contained within a `\verb` or `\verb*` construct. This means, for example, that

```
\index{\em Command}
```

will be treated as if it were spelled as "`Command`", but

```
\index{\verb+\em+ Command}
```

will be treated as if it were spelled as "`\em Command`".

The spelling processing performs the following operations (in order)

- Accents are processed. All of the special characters associated with the accents are removed. For example, in the *spell-string*, `se\~{n}or` is translated to `senor`.

- Emphasis commands are removed from the *spell-string*. Examples of emphasis commands are `\rm`, `\bf`, `\large`, etc.

- Grouping and mode commands are removed from the *spell-string*. That is, `{`, `}`, and `$` are removed. However, `\{`, `\}`, and `\$` are retained, since they do not represent grouping commands.

- Backslashes are removed from the *spell-string*. The logic which skips processing in `\verb` and `\verb*` constructs prevents the "`\`" in "`\verb`" from being removed.

- `\verb` and `\verb*` constructs are cleaned up. For example, "`\verb+foo+`" is translated to "`foo`".

- The *spell-string* is converted to upper case, all unnecessary whitespace is removed, and a few minor corrections are made to handle special cases.

  For example, the *spell-strings* of index items which begin with non-alphanumeric characters are adjusted so that all such terms will appear in the index before any items which begin with any alphanumeric character.

[6] Obvious examples of this are the indexes of documents about text processing.

Also, special logic is used to assure that any index term which begins with a `\verb` or a `\verb*` is placed in the proper place in the index. This includes adjustments to the *spell-string* which prevent references for items such as "`input`" and "`\verb+\input+`" from being confused.

This approach has proven to be effective in developing an index which uses LaTeX commands liberally, but retains proper alphabetical order. There are, however, aspects of spelling processing which can be debated.

- In *The TeXbook*, native TeX commands are displayed with a leading asterisk, but are alphabetized as if the asterisk were not present. IdxTeX does not currently handle this case.

- IdxTeX is case blind. That is, `\index{Large}` and `\index{large}` are considered two instances of the same item.[7] The case displayed in the index matches that of the first item seen. This is usually desirable — it prevents some typographical errors from generating unwanted index entries. However, there may be some cases in which case sensitivity would be preferred.

Page Ranges

Another issue which appears simple, but has a number of interesting complications is the handling of page ranges. Indeed, the simplest case (converting references on pages 11, 12, and 13, for example, to a reference to "11–13") does not present any significant difficulties. However, the general case is not that simple.

- Since we support page reference highlighting, it is necessary that the system recognize that 11, 12, and 13 constitute a page range, but that **11**, *12*, and 13 must be handled differently.

- A reference such as "20ff" represents a different type of page range. If a term is also indexed on page 19, then the index entry should read "19ff" rather than "19, 20ff".

- Some document styles use chapter oriented (or other complex) page numbers. The algorithm which determines whether pages are adjacent must be able to handle page numbers such as "5–2" or "Glossary–4".

- In a Master Index context, the algorithm must also be able to determine that a reference to page 11 from Volume I is not adjacent to a reference to page 12 in Volume II.

It turns out that solving these complications is unreasonably difficult during the initial building of the internal data structures. Therefore, a special processing step is used to handle page ranges.

For each linked list of page references, an array of special data structures[8] is dynamically allocated and the information from the linked list is moved to the array. Each page reference text string is parsed into a volume string, a chapter string (if any), a page number, and a highlight flag. Two pages are adjacent if they have the same volume, chapter, and highlight, and consecutive page numbers.

[7] In fact, **any** items which have the same *spell-string*, according to the syntax rules above, will be considered instances of the same item. The displayed text will be that of the first index reference seen.

[8] My thanks to my colleague, Donald R. Gummow, for suggestions concerning this internal array.

Page references which are parts of ranges are flagged as the beginning, middle, or end of the range. Since the "*and following*" notation is handled internally as a highlight, it is relatively easy to handle special cases involving this type of page reference within a page range.

Once this analysis is complete and one or more page ranges is discovered in the list of page references, the initial list is deleted and a new page reference list is built based on the information in the array. Since this approach concentrates all of the page range logic in one place, the routines which format the output require no special logic. Also, given the amount of information stored in the array, it is trivial to provide special touches, such as formatting a range of simple page numbers as "`11--13`", while handling a range of complex page numbers as "`2--6 to 2--10`".

Other Features

A number of other features of the program deserve some mention.

Cross References

Handling cross references turned out to be surprisingly easy, once I realized that they should be segregated from page number references in their own linked list. This allowed multiple cross references to be listed in alphabetical order, and eliminated problems associated with mixtures of page number references and cross references for the same term.

At present, IdxTEX does not check to verify that an index entry actually exists for each cross reference, but this desirable feature could be added without great difficulty.

Master Index

Some special processing is required to generate a Master Index.

- There must be a mechanism to inform IdxTEX of which auxiliary files to process to build the Master Index (and in which order to process them). This problem was solved by creating a new auxiliary file (an `.mdx` file) which lists the `.idx` files to be processed. A special qualifier to the `IdxTeX` command is used to specify that Master Index processing is to be performed.
- For the Master Index to be useful, it is necessary that the formatted index include *labels* which identify the volumes from which the page references come. The `.mdx` file is the obvious place for these labels to be defined. As noted above, a pointer to these label strings is included as part of the PGNODE structure, so that the labels may be easily included in the output.

Output Format

Since the internal data structures contain all of the information needed to generate the Index, creation of the output file is a simple matter. All that is necessary is to walk the linked lists, generating appropriate LATEX code as we go.

The most interesting problem which occurs during output generation concerns the headings which precede the index entries which begin with a new letter of the alphabet. In the first version of the program, the heading for a new letter could appear at the bottom of one column, with the first entry for that letter appearing at the top of the next column. This was clearly undesirable.

To solve this problem, I defined a new LaTeX indexing command

```
\makeatletter
\def\indexhead#1#2#3{\par\if@nobreak \everypar{}
   \else\addpenalty{\@secpenalty}\addvspace{#1}\fi
   \begingroup #3\par \endgroup  \@xsect{#2}}
\makeatother
```

IdxTeX includes this code at the beginning of every output file it generates.

This macro was derived from LaTeX's section processing logic, where the same type of orphan problem exists. The first parameter is the amount of space to leave before the heading. The second parameter is the amount of space to leave after the heading. The third parameter is the text used to generate the heading. This macro uses `\nobreak`, `\everypar`, and `\clubpenalty` to produce the desired effect.

Summary

The IdxTeX program has been used to generate indexes in a substantial number of documents at Monsanto. We have found that its indexes are effective and attractive — well in keeping with the general quality of the documents in which they appear. It has no difficulty handling large indexes — in fact, I estimate that a document containing 25,000 `\index` commands should be well within the virtual page quotas normally found on VAX/VMS systems optimized for scientific computing environments.

While IdxTeX has basically met its design goals, a simple change in the LaTeX document styles (which was beyond the scope of this project) would allow it to do even more. At the beginning of the start of the Index to *The TeXbook*, for example, there are several paragraphs of one-column text which describe how to use the Index. The current definition of `\begin{theindex}` precludes this type of usage. I believe that a simple change to the definition of this environment (taking advantage of the optional argument of the `\twocolumn` command) would contribute to even better, more effective indexes.

Of course, IdxTeX fails to deal with the most difficult part of building an Index that communicates effectively — it does not insert the `\index` commands in the document. I leave that part of the problem to the AI experts.

Problems

Problem for a Saturday Morning — A Solution

Donald E. Knuth
Stanford University

```
% Problem for a Saturday Morning --- A Solution
\newdimen\z \z=0pt
\font\big=cmbx10 scaled \magstep5
\newbox\query \setbox\query=\hbox{\raise6pt\hbox{\big\thinspace?\thinspace}}
\newdimen\leftedge \newdimen\rightedge
\leftedge=\hsize \advance\leftedge by-\wd\query \divide\leftedge by 2
\rightedge=\leftedge \advance\rightedge by\wd\query
\parshape 10  \z\hsize  \z\hsize  \z\hsize
 \z\leftedge \rightedge\leftedge  \z\leftedge \rightedge\leftedge
 \z\leftedge \rightedge\leftedge  \z\hsize
\newbox\partialpage \newcount\n
\newdimen\savedvsize \savedvsize=\vsize
\newtoks\savedoutput \savedoutput=\output
\newdimen\savedprevdepth \savedprevdepth=\prevdepth
\output={\global\setbox\partialpage=\vbox{\unvbox255\unskip}}\vfill\break
\topskip=\ht\strutbox \vsize=\topskip
\n=245 % we will store nine lines of text in boxes 246--254
\output={\global\advance\n by 1
 \ifnum\n<255 \global\setbox\n=\box255
 \else \unvbox\partialpage \prevdepth=\savedprevdepth \vskip\parskip
  \box246  \box247  \box248
  \box249 \vskip-\baselineskip \box250
  \box251 \vskip-\baselineskip \box252
  \box253 \vskip-\baselineskip \box254
  \vskip-\baselineskip \moveright\leftedge\hbox{\smash{\box\query}}
  \box255 \global\vsize=\maxdimen \fi}
\noindent This puzzle was suggested to me by Sape Mullender, of the Centre for
Mathematics and Computer Science in Amsterdam. He told me his belief that
``the general design of \TeX\ is better than that of {\it troff}, but the
real guru can make {\it troff\/} do things that you could never do in \TeX.''
As an example, he showed me a page on which {\it troff\/} had typeset a
picture in the middle of a paragraph, with the text going around the picture.
``It's not pretty, but it can be done, and that's what counts,'' he said.
Well, I have to admit that I didn't think of a simple solution until the
next Saturday morning; and I didn't finish debugging it until that
Saturday afternoon. Can you guess how I typeset the paragraph you're
now reading?  (The answer will appear in the next issue. It doesn't
demonstrate the superiority of \TeX\ to {\it troff}, but it does have
some interesting and instructive features.)
\savedprevdepth=\prevdepth
\output{\global\setbox\partialpage=\vbox{\unvbox255\unskip}}\vfill\break
\vsize=\savedvsize \output=\savedoutput
\unvbox\partialpage \prevdepth=\savedprevdepth
% Improvements to this solution are welcome!
```

TeX Does Windows — The Conclusion

Alan Hoenig

In the last issue, readers were invited to think about creating TeX macros to "do windows"—have TeX leave rectangular cutouts horizontally centered within paragraphs and in which you can insert artwork, or whatever. The solution I discuss in this issue differs significantly from the one I had in mind at the time I threw down the gauntlet last time. The window macros have become leaner and more robust. (I am grateful to my colleague Mitch Pfeffer who supplied me with one crucial idea, and to Barbara Beeton for a valuable insight. Barbara also helped keep me honest.) I present the macros below, as well as two extensions, which allow TeX to set rectangular cutouts which aren't horizontally centered, and which force TeX to set cutouts of *arbitrary* shape. I do make several limiting assumptions: the cutout fits entirely within a single paragraph, and the `\baselineskip` remains constant within that paragraph. I believe you can modify these macros with little additional work, however. There is one known bug, which I was unable to fix in time to meet the submission date. When the ratio of baselineskip to design font size reaches decreases to a certain critical value, the cutout is not properly formed. You'll be okay if you keep the baselineskip at least 2 points greater than the design size.

Using the Macros

Here's how to use the macros. First, some anatomy. The several lines of material *above* the window we'll call the *lintel*. The material below it is the *sill*, and the text to its right and left form its *sides*. (Of course, some of these regions may be empty depending on your placement of the window.)

We'll talk about the macro that generates a horizontally-centered window first. Before you invoke the window-making macro, place the code which reserves the registers that the macros use. These definitions form figure 1.

```
\newcount\l \newcount\d \newdimen\lftside \newdimen\rtside \newtoks\a
\newbox\rawtext \newbox\holder \newbox\window \newcount\n
\newbox\finaltext \newbox\aslice \newbox\bslice
\newdimen\topheight
\newdimen\ilg % InterLine Glue
```

FIGURE 1. The boxes, counts, dimens, and so on you need for the window-making macro.

Next, place the actual macro definitions in your document file. There are a slew of such macros—you'll need them all. They appear in figure 2.

```
\def\openwindow\down#1\in#2\for#3\lines{%
% #1 is an integer---no.  of lines down from par top
% #2 is a dimension---amount from left where window begins
% #3 is an integer---no.  of lines for which window opening
%    persists.
\d=#1 \l=#3 \lftside=#2 \rtside=\lftside \a={}
\createparshapespec
\d=#1 \l=#3  % reset these
\setbox\rawtext=\vbox\bgroup
\parshape=\n \the\a }
%
\def\endwindowtext{%
\egroup \parshape=0 % reset parshape; end \box\rawtext
\computeilg % find ILG using current font.
\setbox\finaltext=\vsplit\rawtext to\d\baselineskip
\topheight=\baselineskip \multiply\topheight by\l
\multiply \topheight by 2
\setbox\holder=\vsplit\rawtext to\topheight
% \holder contains the narrowed text for window sides
\decompose\holder\to\window % slice up \holder
\setbox\finaltext=\vbox{\unvbox\finaltext\vskip\ilg\unvbox\window%
\vskip\ilg\unvbox\rawtext}
\box\finaltext} % finito
%
\def\decompose#1\to#2{%
\loop\advance\l-1
\setbox\aslice=\vsplit#1 to\baselineskip
\setbox\bslice=\vsplit#1 to\baselineskip %get 2 struts
\prune\aslice\lftside \prune\bslice\rtside
\setbox#2=\vbox{\unvbox#2\hbox
to\hsize{\box\aslice\hfil\box\bslice}}
\ifnum\l>0\repeat
}
%
\def\prune#1#2{ % take a \vbox containing a single \hbox,
% \unvbox it, and cancel the \lastskip
% put in a \hbox of width #2
\unvbox#1 \setbox#1=\lastbox %\box#1 now is an \hbox
\setbox#1=\hbox to#2{\strut\unhbox#1\unskip}
}
%
\def\createparshapespec{%
\n=\l \multiply \n by2 \advance\n by\d \advance\n by1
\loop\a=\expandafter{\the\a 0pt \hsize}\advance\d-1
\ifnum\d>0\repeat
\loop\a=\expandafter{\the\a 0pt \lftside 0pt \rtside}\advance\l-1
\ifnum\l>0\repeat
\a=\expandafter{\the\a 0pt \hsize}
}
%
\def\computeilg{% compute the interline glue
\ilg=\baselineskip
\setbox0=\hbox{()} \advance\ilg-\ht0 \advance\ilg-\dp0
}
```

FIGURE 2. The window-making macros which generate horizontally centered rectangular windows in a paragraph of text.

Apart from identifying the window text, there are 3 parameters you need to determine: The number of lines *down* for the the top of the paragraph (that is, the thickness of the lintel), the amount in from the left (the width of the sides), and the number of lines for which the window persists (the thickness of the sides). If w is the width of the side (the value of parameter #2 in the `openwindow` macro), then the width of the window is `hsize` $- 2w$. You see in figure 4 the exact way in which I created the window in *this* paragraph.

```
\openwindow\down 1\in 15pc \for 2\lines
Apart from identifying ...
  ...window in {\sl this} this paragraph.
\endwindowtext
```

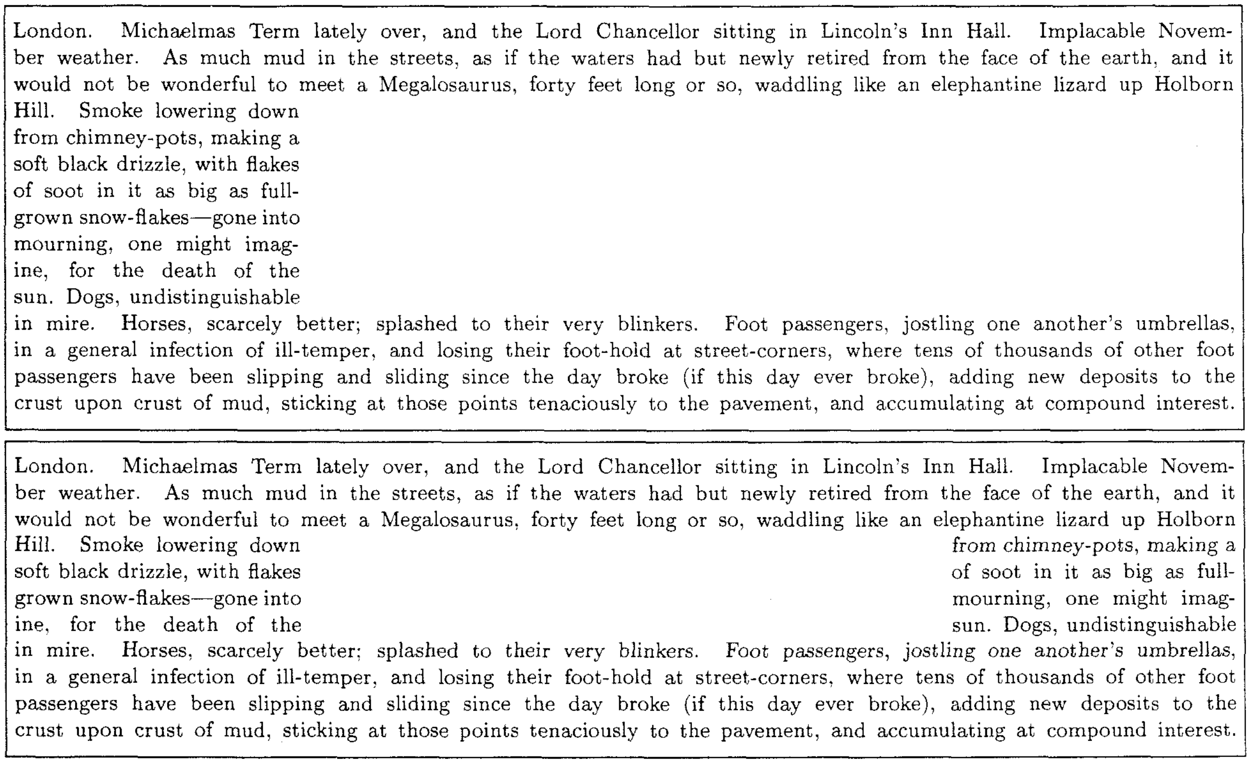
FIGURE 3. Opening up a window in text.

Horizontally Centered Windows

Here is the basic idea behind these macros. We seek to use the command `\parshape` to create an odd-shaped paragraph consisting of a top portion identical to the lintel, a bottom portion identical to the sill, and a lengthy and narrow middle portion with the width of the side text. Then, take this typeset text, and slice it like a roast beef. These "slices " will contain lines of text in `\vbox`es which we rearrange to get the text we want, cutout and all. In figure 4, you see the intermediate position of some text *before* and *after* this rearrangement.

Only a few macros require any special comment. Macro \createparshapespec computes the value n that the \parshape command looks for. Next, it builds up a token \a which contains the $2n$ line lengths and indentations. (I take all the indentations to be 0pt, but varying this can add to your palette of special effects.) Some of this token manipulation is a tad tricky. You construct tokens by enclosing the token list in braces, but if you don't prefix this with an \expandafter, TEX will construct \a out of the component names, rather than out of their meanings. The last line is necessary because of the way \parshape works. If the paragraph is sufficiently long, TEX uses the final line length for the remainder of the text. If we set the final line length to be \hsize, then this is what we need to set the sill text.

Note that these macros use \struts to help maintain proper vertical line spacing. If you use type at other than the usual 10 point design size, *make sure to redefine the* \strut *to reflect this change.*



London. Michaelmas Term lately over, and the Lord Chancellor sitting in Lincoln's Inn Hall. Implacable November weather. As much mud in the streets, as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn
Hill. Smoke lowering down
from chimney-pots, making a
soft black drizzle, with flakes
of soot in it as big as full-
grown snow-flakes—gone into
mourning, one might imag-
ine, for the death of the
sun. Dogs, undistinguishable
in mire. Horses, scarcely better; splashed to their very blinkers. Foot passengers, jostling one another's umbrellas, in a general infection of ill-temper, and losing their foot-hold at street-corners, where tens of thousands of other foot passengers have been slipping and sliding since the day broke (if this day ever broke), adding new deposits to the crust upon crust of mud, sticking at those points tenaciously to the pavement, and accumulating at compound interest.

London. Michaelmas Term lately over, and the Lord Chancellor sitting in Lincoln's Inn Hall. Implacable November weather. As much mud in the streets, as if the waters had but newly retired from the face of the earth, and it would not be wonderful to meet a Megalosaurus, forty feet long or so, waddling like an elephantine lizard up Holborn
Hill. Smoke lowering down from chimney-pots, making a
soft black drizzle, with flakes of soot in it as big as full-
grown snow-flakes—gone into mourning, one might imag-
ine, for the death of the sun. Dogs, undistinguishable
in mire. Horses, scarcely better; splashed to their very blinkers. Foot passengers, jostling one another's umbrellas, in a general infection of ill-temper, and losing their foot-hold at street-corners, where tens of thousands of other foot passengers have been slipping and sliding since the day broke (if this day ever broke), adding new deposits to the crust upon crust of mud, sticking at those points tenaciously to the pavement, and accumulating at compound interest.

FIGURE 4. The window macro generates a funny-looking paragraph (top) using the 'parshape' command, which it then rearranges to form the window (bottom).

The actual "deli slicing" is done by the \decompose macro, and the slicing mechanism is done via the \vsplit primitive. Decomposition also calls for the \pruneing of each slice—removing the glue TEX places on the right of a short, \parshape line.

On Beyond Centering: Off-Center Windows

All the hard work in cutting out windows is already present in these macros. It's almost trivial to generalize to the case where a rectangular window is to be off-center. The only real change is to the definition of \openwindow to allow for another parameter—the width of the right side of the window. Here, in figure 5, is the way I've redefined the beginning of \openwindow.

```
\def\openwindow\down#1 \in #2 \fromright #3 \for #4\lines{%
\d=#1 \l=#4 \lftside=#2 \a={} \rtside=#3
```

FIGURE 5. The opening for the openwindow macro to account for windows that can be off-centered horizontally. Parameter 3 is now to be the distance from the right margin.

On Beyond Rectangles: Arbitrary Shapes

The most interesting part of writing these macros was their extension to windows of *arbitrary* shape. Users specify the "shapespec" for their cutout by creating a list of the right and left line lengths, each of which is separated by a double backslash `\\`; this list is then passed to the macro as a single parameter. Changes to the macros to handle this shape spec are localized to a few places. As before, the beginning `\openwindow` statement needs modifying to accept a different parameter list. The `\parshape` specifications are built up in a slightly different way, and `\createparshapespec` is itself slightly different. The most extensive changes are in the `\decompose` macro, which uses the `\lop` macro (of Appendix D in the *TeXbook*) to chop off the individual line lengths from the shape spec. At the risk of appearing repetitious, all the listings for all the macros in this version of `\openwindow` appear in figure 6. The shapespec itself is handled as a list, à *la* the suggestions of Appendix D, pages 378–380, in *The TeXbook*. The general shapespec is of the form `\\`l_1`\\`r_1`\\`l_2`\\`r_2`\\`...`\\`l_n`\\`r_n`\\` where there are n lines in the cutout area. Each l_i should be the length of the left side, and each r_i is the length of the right side for the i^{th} line of the cutout. Note well the double backslash which both begins *and* ends the shapespec. The shapespec for this paragraph is `\\ 168\x \\ 168\x \\ 154\x \\ 154\x \\ 145\x \\ 145\x \\ 138\x \\ 138\x \\ 134\x \\ 134\x \\ 132\x \\ 132\x \\ 131\x \\ 131\x \\ 132\x \\ 132\x \\ 134\x \\ 134\x \\ 138\x \\ 138\x \\ 144\x \\ 144\x \\ 154\x \\ 154\x \\ 168\x \\ 168\x \\` where `\x` is a special `\dimen` register containing the value 1.22pt.

The area of a circle is a mean proportional between any two regular and similar polygons of which one circumscribes it and the other is isoperimetric with it. In addition, the are of the circle is less than that of any circumscribed polygon and greater than that of any isoperimetric polygon. And further, of these circumscribed polygons, the one that has the greater number of sides has a smaller area than the one that has a lesser number; but, on the other hand, the isoperimetric polygon that has the greater number of sides is the larger. [Galileo, 1638]

```
%
% Special registers
%
\newcount\l \newcount\d \newdimen\lftside
\newdimen\rtside \newtoks\a \newtoks\b
\newbox\rawtext \newbox\holder \newbox\window \newcount\n
\newbox\finaltext \newbox\aslice \newbox\bslice
\newdimen\topheight
\newdimen\ilg % InterLine Glue
\newtoks\c
%
%
% Here are the special WINDOW MACROS.
%
\long\def\openwindow\down#1\for#2\linesand#3as_shape_spec{%
% #1 is an integer---no.  of lines down from par top
% #2 is an integer---no.  of lines for which window opening
% #3 is the shapespec token list
\d=#1 \l=#2 \def\b{#3} \def\tail{\hsize }
\edef\\{0 pt}
\createparshapespec
\d=#1 \l=#2  % reset these
\setbox\rawtext=\vbox\bgroup
\parshape=\n \the\c \b \tail
}
%
\def\endwindowtext{%
\egroup \parshape=0
% reset parshape; end \box\rawtext
\computeilg % find ILG using current font.
```

```
\setbox\finaltext=\vsplit\rawtext to\d\baselineskip
\topheight=\baselineskip \multiply\topheight by\l
\multiply \topheight by 2
\setbox\holder=\vsplit\rawtext to\topheight
% \holder contains the narrowed text for window sides
\decompose\holder\to\window % slice up \holder
\setbox\finaltext=\vbox{\unvbox\finaltext\vskip\ilg\unvbox\window%
\vskip\ilg\unvbox\rawtext}
\box\finaltext} % finito
%
\def\lop#1\to#2{\expandafter\lopoff#1\lopoff#1#2}
\long\def\lopoff\\#1\\#2\lopoff#3#4{\def#4{#1}\def#3{\\#2}}
%
\def\decompose#1\to#2{%
\loop\advance\l-1
\lop\b\to\lft \lftside=\lft
\lop\b\to\rt \rtside=\rt
\setbox\aslice=\vsplit#1 to\baselineskip
\setbox\bslice=\vsplit#1 to\baselineskip %get 2 struts
\prune\aslice\lftside \prune\bslice\rtside
\setbox#2=\vbox{\unvbox#2\hbox
to\hsize{\box\aslice\hfil\box\bslice}}
\ifnum\l>0\repeat
}
%
\def\prune#1#2{ % take a \vbox containing a single \hbox,
% \unvbox it, and cancel the \lastskip
% put in a \hbox of width #2
\unvbox#1 \setbox#1=\lastbox %\box#1 now is an \hbox
\setbox#1=\hbox to#2{\strut\unhbox#1\unskip}
}
%
\def\createparshapespec{%
\n=\l \multiply\n by 2
\advance\n by1 \advance\n by\d
\c={}
\loop\c=\expandafter{\the\c 0in \hsize}\advance\d-1
\ifnum\d>0\repeat
}
%
\def\computeilg{% compute the interline glue
\ilg=\baselineskip
\setbox0=\hbox{(} \advance\ilg-\ht0 \advance\ilg-\dp0
}
```

FIGURE 6. These are the macros you will need to generate cutouts of arbitrary text within your document.

A few final comments. Use these macros sparingly; it's difficult and mighty uncomfortable to read across a large gap in text. If you do use these macros, try not to let the width of the sides get too narrow or you'll have lots of 'overfull' messages. You may therefore want to up the `\tolerance` of your document. Also, the use of these macros may greatly increase the TEX compilation time. This may be especially noticeable on microcomputers like an IBM PC. Wait a good few minutes before you decide the system has hung and it's time to re-boot.

Comment on "TEX Does Windows"

Jim Fox

The emphasis of the "window paragraphs" article in the March TUGboat seems to be somewhat misdirected. It is not so big a deal that TEX can be made to format such paragraphs, indeed the 14th and 15th chapters of the *TEXbook* describe completely and in detail the means both to specify paragraph shape and to split off parts of said paragraphs—thereby rendering these *holed* paragraphs essentially trivial. Nor should the emphasis be placed on the paragraphs themselves (the hole being distracting and making the paragraphs hard to read, especially when the hole produces not only a gap between words but a hyphenated word across the gap as well. The emphasis instead belongs on the TEX itself, and the very fact that it can format such idiosyncratic paragraphs with ease. And one should note that the holes can be of arbitrary shape, and there can be several of them in the paragraph.

Queries

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

The following items, which appear elsewhere in this issue, are in response to, or otherwise relevant to, previous queries.

- First-line special handling (James Alexander, Vol. 7, No. 2, page 110), see page 193.
- Indexing with LATEX (Jim Ludden, Vol. 7, No. 2, page 111), see page 201.
- Setting parallel texts (John Stovall, Vol. 2, No. 2, page 57), see page 190.

Time Line Macro

This query elicited no response when it was published in TEXhax, so I will try the TUGboat audience. In addition to being quite useful for its (admittedly specialized) purpose, it would seem to be a challenging exercise for an expert — something along the lines of some of the esoteric exercises in the *TEXbook* or the tree-making macro of last year's TUGboat. I offer it as such a challenge.

I would like a macro which makes a "time line". It would read a file which consists of entries of the form

$$\langle\textit{date}\rangle\ \langle\textit{event}\rangle$$

(presorted if necessary) and produce a vertical line of some preassigned length with tick marks so that the top of the line represents the first date (or #1 in the macro call) and the bottom represents the last date (or #2). Down the line, with vertical spacing mimicking (and that is the key point) time intervals, the dates and events are printed horizontally out to the right. One problem is to do something intelligent when two or more of the dates cluster too closely (e.g. two events on the same date). One can see the general idea, but also many TEXnical details. Alternately (perhaps less interestingly), one could write a preprocessor in C or Pascal.

Sometimes the time scale is linear (e.g. for the history of the USA); sometimes a logarithmic scale is appropriate (e.g. cosmological events since the beginning of the universe — as much happened in the first second or so as since — or, compressing in the opposite direction, the chronology of life on earth). Such time lines are a useful semi-pictorial way of presenting chronologies, but are somewhat awkward to create with conventional typesetting. Any takers?

James Alexander
University of Maryland

Reply: Printing Out Selected Pages

In TUGboat Vol. 7, No. 3, Helen Horstman asked, "Is there some way by which one can select only a page (or pages) of printout?"

I recently put some new lines, shown below, into MANMAC (the macros of Appendix E that generated Volumes A and E), so that I could put only selected pages into the DVI file. The method should work if you use it at the end of almost any macro file. (Or, if necessary, at the front of a source document.)

The idea is to make TeX look for a file called pages.tex. If such a file doesn't exist, everything works as before. Otherwise the file should contain a list of page numbers, one per line, in the order they will be generated. After the last page number has been matched, all further pages will be printed. Thus, if you want to print page 123 and all pages from 300 onwards, your file pages.tex should say

```
123
300
```

but if you want to print pages 123 and 300 only the, file should say, e.g.,

```
123
300
-9999999999 % impossible number
```

so that the end of file will never occur.

You should rename the pages.tex file after you're done with it; otherwise it will continue to affect the output.

The macros cause TeX to announce that fact that it's doing something special.

Donald Knuth
Stanford University

Using the Windows Environment

We currently run TeX on IBM PC/XT and AT's and have recently adopted Microsoft's Windows environment to provide us with a Mac-like interface. At present MicroTeX will run without modification under Windows but without pull-down menus and the like. I would be very interested to hear from anyone who either has a .DVI file previewer that will work under Windows or who is interested in developing such a previewer (or any TeX product that runs under Windows).

As Windows is about to be upgraded and will form the presentation manager of OS/2 for the new range of IBM Personal System computers, this would seem to be where the future is for those of us who live in the world of IBM compatibility.

Mike Black
Kingsdown Publishing Ltd.
London N16 7DP

Macro for printing out selected pages

```
\let\Shipout=\shipout
\newread\pages \newcount\nextpage \openin\pages=pages
\def\getnextpage{\ifeof\pages\else
 {\endlinechar=-1\read\pages to\next
  \ifx\next\empty % in this case we should have eof now
  \else\global\nextpage=\next\fi}\fi}
\ifeof\pages\else\message{OK, I'll ship only the requested pages!}
 \getnextpage\fi
\def\shipout{\ifeof\pages\let\next=\Shipout
 \else\ifnum\pageno=\nextpage\getnextpage\let\next=\Shipout
  \else\let\next=\Tosspage\fi\fi \next}
\newbox\garbage \def\Tosspage{\deadcycles=0\setbox\garbage=}
```

TeX Users Group

University of Washington, Seattle, Wash.

August 23 – 26, 1987

Program as of June 25, 1987

Sunday – August 23

6:30 – 10:15 pm	Dinner/Cruise – Lake Washington[1]

Monday – August 24

7:45 – 9:00 am	Registration
8:00 – 9:00	Introduction to TeX and TUG for new users – Bart Childs
9:00 – 9:15	Introductions: officers, site coordinators, others
9:15 – 10:00	Publishing ventures for linguistics and the humanities – Christina Thiele
10:00 – 10:30	Burton's Anatomy of Melancholy – Dean Guenther
10:30 – 10:45	Break
10:45 – 11:30	Classical Greek – Silvio Levy
11:30 – 1:00 pm	Lunch
1:00 – 2:00	Publishing in Turkish – Walter Andrews and Pierre MacKay
2:00 – 3:00	Japanese TeX [JTeX] – Yaski Saito
3:00 – 4:15	Developing TeX DVI driver standards – Robert McGaffey
4:15 – 4:30	Orientation – things to do in and around Seattle
6:00 – ???	Barbecue – entertainment provided by the Seattle Banjo Club hosted by Addison-Wesley

Tuesday – August 25

9:00 – 10:30 am	Output device manufacturer/exhibitor presentations[2] Addison-Wesley Publishing Co.; ArborText, Inc.; Computer Composition Corp.; FTL systems Inc.; Imagen Corp.; Kellerman & Smith; Personal TeX; Talaris Systems Inc.; TeXnology, Inc.
10:30 – 10:45	Break

10:45 – 11:30 am	Site Coordinators' status reports (*preliminary listing*)
	DG MV8000, Prime 750 – Bart Childs; HP 3000 – Lance Carnes;
	IBM VM/CMS – Dean Guenther; UNIX – Pierre MacKay;
	VAX (VMS) – Barry Smith

11:30 – 1:00 pm	Lunch
1:00 – 1:45	Building a modular DVI driver – Nelson Beebe
1:45 – 2:45	Using TeX in a non-academic environment – Elizabeth Barnhart
2:45 – 3:00	Break
3:00 – 3:45	CWEB – Silvio Levy
3:45 – 4:00	Update from the standards committee
4:00 – ???	Birds-of-a-Feather sessions
5:00 – ???	Wine & cheese – hosted by Personal TeX

Wednesday – August 26

9:00 – 9:30 am	TeX as the standard for Maryland lawyers – Allen Dyer
9:30 – 10:00	PostScript outline fonts from METAFONT – Les Carr
10:00 – 11:30	TUG business meeting
11:30 – 1:00 pm	Lunch
1:00 – 2:00	TeX problems help session – Barbara Beeton et al.
2:00 – 3:30	Report from the standards committee
3:30 – 4:00	General wrap-up and closing – Bart Childs et al.

[1] Barbecued Salmon or Cajun Shiskabob, along with baked potato, salad, dessert, beer, wine and soft drinks will be served.

[2] *Preliminary listing.* Representatives are scheduled to be available throughout the meeting. Exhibit rooms will be open from 10:30 am, Tuesday, until 3:00 pm, Wednesday.

An IBM PC and floppy discs will be available so that members might exchange software.

•

Program coordinator: Dean Guenther, Washington State University

Calendar

1987

University of Exeter, England

Jul 6–10 Intensive Beginning/Intermediate TeX

Jul 13–17 Advanced TeX/Macro Writing

* * * * *

Jul 8–10 International Conference on Text and Image Processing, Hotel Maritim, Würzburg, Federal Republic of Germany; several sessions by TeX users.

Jul 13–17 Advanced TeX/Macro Writing; University of New Mexico, Albuquerque

Rijksuniversiteit Groningen, The Netherlands

Jul 20–24 Intensive Beginning/Intermediate TeX

Jul 27–31 Advanced TeX/Macro Writing

* * * * *

Jul 27–30 TeX Wizard Course; TV Guide Magazine Headquarters, Radnor, Pa.

Jul 27–31 SIGGRAPH, Anaheim, Calif. For information, contact the ACM Conference Office, Chicago, Ill.; 312-644-6610

Rutgers University, Busch Campus, Piscataway, N. J.

Aug 3–4 Macro Writing

Aug 5–7 LaTeX Style Files

Aug 10–14 Beginning TeX

Aug 10–14 Intensive Beginning/Intermediate TeX

* * * * *

Aug 4–7 Sixth International Conference on Mathematical Modelling, Washington University, St. Louis, Mo.; TeX exhibits and presentations.

Texas A & M University, College Station

Aug 10–14 Beginning TeX

Aug 10–14 Intensive Beginning/Intermediate TeX

TeX Users Group 1987 Conference University of Washington, Seattle

Aug 17–21 Beginning TeX

Aug 17–21 Intensive Beginning/Intermediate TeX

Aug 24–26 **TUG Annual Meeting** preliminary program, page 218

Aug 27–28 Short course: Macro writing

Aug 27–28 Short course: Output routines

Stanford University, Palo Alto

Aug 17–21 Intensive Course in LaTeX

Aug 24–28 Beginning LaTeX

* * * * *

Aug 28–Sep 4 American Chemical Society, National Meeting, New Orleans, La.; two symposia on problems of journal publication: "Methods for the electronic submission of manuscripts for publication", and "Problems and solutions in the generation of scientific manuscripts". For information on the symposia, contact Peter Lykos, Illinois Institute of Technology, 312-567-3430, Bitnet: `chempgl@iitvax`. TeX exhibits and presentations.

Sep 9–12 Seybold Seminars' *Desktop Publishing Conference*, Santa Clara, Calif. For information, contact Seybold Seminars, Malibu, Calif.; 213-457-5850

Sep 14 **TUGboat Volume 8, No. 3:** Deadline for submission of manuscripts

Status as of 15 June 1987

University of Illinois, Chicago

Sep 14–18	Intermediate TeX
Sep 14–18	Advanced TeX/Macro Writing

Bergen Scientific Center, Norway

Sep 21–25	Intensive Beginning/Intermediate TeX
Sep 28–Oct 2	Advanced TeX/Macro Writing

* * * * *

Sep 28–Oct 1	Conference on Electronic/Desktop Publishing, San Francisco, Calif. For information, contact National Computer Graphics Association, Fairfax, Va.; 703-698-9600
Oct 8–9	Annual meeting, Deutsche TeX-Interessenten, University of Münster, Federal Republic of Germany; for information, contact Joachim Lammarsch, University of Heidelberg (`$33$DHDURZ1.BITNET`), or Wolfgang Kaspar, University of Münster (`URZ86@DMSWWU1A.BITNET`)
Oct 19–22	Protext IV, Boston, Mass.; TeX Seminar, Monday, Oct. 19 (see announcement, page 220)

1988

Jan 5–9	Joint Mathematics Meeting, Atlanta, Ga. TeX Short Course, Tuesday, Jan. 5
Apr 20–22	International Conference on Electronic Publishing, Document Manipulation and Typography, Nice, France (see announcement, TUGboat Vol. 8, No. 1, page 78)
Jul 18–20	TeX Conference, University of Exeter, England.

For additional information on the events listed above, contact the TUG office (401-272-9500, ext. 232) unless otherwise noted.

PROTEXT IV

Boston, Massachusetts
20–22 October 1987

The Fourth International Conference on Text Processing Systems will be held in the Boston Park Plaza Hotel and Towers. The conference is being held under the auspices of the Institute for Numerical Computation and Analysis, which is a non-profit research corporation licensed under the laws of the Republic of Ireland. Six related short courses will be offered on the day preceding the conference, 19 October 1987; except as noted below, the short courses are held under the same auspices.

Conference, 20–22 October 1987

The conference provides a forum for discussion of the latest research on computer-aided generalized text processing. Keynote speakers include

John Collins (Bitstream, Cambridge)
Richard Furuta (University of Maryland, College Park)
Shiu Chang Loh (Chinese University, Hong Kong)
Pierre MacKay (University of Washington, Seattle)
Marc Nanard (CRIM, Montpellier)
Luis Trabb Pardo (Imagen Corporation, Santa Clara)
Xuan Wang (Peking University, Beijing)

The Programme Committee is headed by John Miller (Trinity College, Dublin) and Robert Morris (University of Massachusetts/Boston).

Short Courses, 19 October 1987

The following one-day short courses are being held in parallel on the day before the conference begins.

A Issues in Generalized Text Processing
Richard Furuta, Shiu Chang Loh, Pierre MacKay, Marc Nanard

B TeX for Scientific Documentation
Bart Childs (Texas A&M University, College Station) in cooperation with the TeX Users Group

C **An Introduction to SGML**
W. W. Davis (Internal Revenue Service, Washington, DC)

D **Issues in Digital Typography**
Richard Rubenstein (Digital Equipment Corp., Hudson, MA)

E **Introduction to PostScript – A Graphics Solution**
Yvonne Perry (Adobe Systems, Palo Alto, CA)

F **Document Databases and Technical Publishing**
Geoffrey James (Honeywell Information Systems, Los Angeles, CA) under the auspices and organized by the University of Massachusetts/Boston and University of California/Los Angeles Extension Programs (to be confirmed)

Further Information

To obtain further information, request a form from the Conference Organizer:

Paulene McKeever
Conference Management Services
P. O. Box 5
51 Sandycove Road
Dún Laoghaire
Co. Dublin, Ireland
(+353-1) 452081
Telex: 30547 SHCN EI (Ref. Boole)

Please note that there is an early rate that applies to all fees *received* by the Conference Organizer before 1 September 1987. Reservations for accommodations must be made directly with the hotel before 15 September 1987 to ensure availability; a block of rooms has been reserved at a special conference rate for participants. Details of all fees and other arrangements will be on the form.

6th German TeX Meeting

On October 8th and 9th, 1987, German TeX Users and other people interested in TeX will meet in Münster (Westphalia) at the University Computing Centre.

For the first day we plan — for people who just had their first contacts with TeX — introductory lectures about installing TeX and using LaTeX and other macro packages.

The second day we want to give information about further developments related to TeX. Also, we are going to deal with problems specifically concerning the German language.

During both days demonstrations will be offered on several Output Device Drivers and TeX Implementations on workstations and PCs.

For further information, please contact:

W. Kaspar
Univ of Münster
Computing Center
Einsteinstraße 60
D-4400 Münster
Fed Rep Germany

Late-Breaking News

SGML and TeX

Lynne A. Price
Hewlett-Packard, Palo Alto, CA

The Standard Generalized Markup Language, SGML, is defined in International Standard 8879, published in October, 1986. This paper gives an overview of SGML, discussing its relationship with other text processing tools such as TeX, PostScript, and WYSIWYG systems. It gives examples of applications for SGML. It concludes with a description of the way SGML and TeX are used together in one particular environment.

What is SGML?

The Standard Generalized Markup Language (SGML) is a notation for representing documents and making their inherent structure explicit. Various forms of automatic processing can be performed on documents coded in SGML; they can be formatted, loaded into online databases, or analyzed for various linguistic properties. SGML is defined in International Standard 8879.

SGML evolved from macro-based word-processing and text-formatting tools. Like a TeX macro package, it encourages a writer to use *descriptive markup*, identifying structures within a document, rather than *procedural markup*, specifying processing. For example, "this is a section heading" is preferred to "center this line in boldface". As the word "generalized" implies, documents prepared with SGML can be processed in various ways. For example, the same markup tags used to prepare a book's index might also be used by an information retrieval application to locate text relevant to selected terms.

SGML views a document as a hierarchy of *structural elements*. For example, a manual may be composed of front matter, some chapters, optional appendices, and an index. Similarly, a chapter may be a series of sections, while a section is composed of text and optional figures, tables, lists, and so on.

No finite set of structural elements can account for the vast flexibility permitted in written texts. SGML therefore provides features for defining types of documents and then coding particular documents that belong to the defined types. Possible document types include reference manuals, journal articles, term papers, short stories, third-grade book reports, memos, letters, and employee performance evaluations. A document type is formally defined with a *document-type definition* that itemizes the structural elements permitted in documents of that type and defines the contexts in which each element can occur. Most document-type definitions are prepared by a small group of individuals for many more people to use with a large number of documents. Thus, most users of SGML are concerned with creating and maintaining documents rather than document-type definitions.

Document-type definitions frequently distinguish elements that are formatted in similar fashion. For example, newly introduced terms and titles of books may both be typeset in italics. However, logically they are different structures. Markup that distinguishes between them enables software that supports glossary and bibliography maintenance.

Context-Sensitive Interpretation of Markup

The document-type definition can control context-sensitive interpretation of parts of a document. For instance, an asterisk may be interpreted as a code for the multiplication symbol inside a mathematical formula but as a footnote indicator elsewhere. Context-sensitive knowledge of valid document structure also permits various abbreviations of SGML constructs, called *markup minimization*. If it is known, for example, that every chapter begins with a chapter title, the SGML processor can recognize the first words in a new chapter as the title whether or not the writer has explicitly coded them as such.

Most SGML markup consists of identifying the beginning or end of structural elements. The most common convention (which can be overridden) is to mark the beginning of an element with the element name enclosed in angle brackets and to mark the end of an element similarly, but with the element name preceded with a slash. These delimiters are illustrated in the (deliberately verbose) example shown in Figure 1.

This example assumes that the document-type definition specifies rules for creating glossaries. Glossaries in this context are assumed to have titles and to contain multiple entries. Each entry has a term followed by a definition. Definitions may contain cross-references to other terms in the glossary.

The document-type definition may also specify context-sensitive text-entry conventions. For example, glossaries may be defined so that the title and terms never extend past the end of a line and that entries are separated by blank lines. With these definitions, SGML treats the example in Figure 2 exactly like the more complete form in Figure 1.

Since SGML knows which document-type definition is being used, the start-tag `<glossary>` can be omitted. The start-tag `<title>` is optional because all glossaries must start with a title. The end-tag `</title>` is optional because the title cannot extend for more than one line. The blank line after the title is ignored because no text characters have been encountered to start the element expected after the title. At the word "aardvark", SGML recognizes that one or more start-tags have been omitted. The start-tag `<entry>` is implied since `<entry>` is the only element allowed after a title. The start-tag `<term>` is then implied since every entry begins with a term. The term cannot extend past the end of the line, so the material on the next line must be something else. Since every entry consists of a term followed by a definition, this must be a definition and SGML infers a <definition> start-tag. The blank line after the definition ends both the entry and the definition contained within it. The end-tag </glossary> is implied by the end of the input file.

SGML's knowledge of context can also be used in some forms of error checking. Most TEX users can sympathize with the user who inadvertantly omits the closing brace after an emphasized phrase and generates several pages printed in a boldface font or who neglects to close an indented list and discovers the rest of the document in narrow columns. SGML, referencing the appropriate document-type definition, knows that an emphasized phrase cannot span multiple paragraphs and that an indented list cannot cross a chapter boundary. When such markup occurs, the effect can be limited to a single paragraph or chapter and appropriate error messages issued. This context-checking is an inherent property of SGML rather than something that must be laboriously built into individual macros.

SGML and Other Tools

SGML is used with classes of related documents, rather than one-of-a-kind texts. The language is carefully and deliberately defined independently of any application. The International Standard specifies possible input of SGML source files without the corresponding output. Unlike TEX and various what-you-see-is-what-you-get systems, the purpose

```
<glossary>
<title>Glossary of Animals</title>
<entry>
  {\it Aardvark\/}
  <definition>The first animal listed in a
              dictionary.</definition>
<entry>
  {\it Cat>\/}
  <definition>A carnivorous mammal long domesticated and kept by
              man as a pet or for catching mice (Webster's New
              Collegiate Dictionary, 1973).
  <definition>
<entry>
  {\it Dog\/}
  <definition>A domesticated <xref>canine</xref>.</definition>
</entry>
          ...
</glossary>
```

Figure 1. Example of glossary data with full markup

```
Glossary of Animals

Aardvark
The first animal listed in a dictionary.

Cat
A carnivorous mammal long domesticated and kept by
man as a pet or for catching mice (Webster's New
Collegiate Dictionary, 1973).

Dog
A domesticated <xref>canine</xref>.

...
```

Figure 2. Example of glossary data after markup minimization

of SGML is not to determine how to arrange characters on paper. Nor is SGML a page-description language like PostScript. The purpose of SGML is to describe the logical structure of a document in a way that can be used by different processes.

Of course, individual uses of SGML have a particular goal, which may be document formatting or page layout. Software to support these applications from SGML can be written. The advantages of doing so are the advantages of context-sensitive markup described above and the ability to use the same source file for other applications as well. These advantages can be preserved whether the application code is written specifically for use with SGML or SGML is used as a front-end to independent tools.

Applications of SGML

Although little SGML software is commercially available as yet, there are several ongoing development efforts. A project is under way at the Institute of Computer Sciences and Technology at the National Bureau of Standards (NBS) to develop an SGML validation suite. The validation suite is being built, along with a public-domain SGML parser, under the Computer-Aided Logistic Support (CALS) program of the Department of Defense. While the purpose of the test suite is to validate SGML parsers intended to process documents that conform to Department of Defense SGML requirements, its examples of correct syntax may also be useful to individuals learning the language.

The Association of American Publishers has defined several SGML document-type definitions for use by authors using machine-readable media to submit books and articles for publication. *The Chicago Guide to Preparing Electronic Manuscripts* (University of Chicago Press, 1987) describes similar markup for use by authors submitting material to the University of Chicago Press. Their guidelines also form a template for publishers defining their own requirements for submission of electronic material.

One particular publisher beginning to use SGML is the Internal Revenue Service. Potential SGML applications at the IRS include embedding the text of relevant sections of the tax code in explanatory material. The tax code can then be printed by an application that generates copy for legal review, but suppressed by the application that prints the information for the taxpayer. SGML can also be used to supply text to tax information services that can in turn distribute it to tax preparers and taxpayers with no government expense.

SGML as a Preprocessor for TeX

Over fifty independent writing departments located throughout the world produce user guides and reference manuals for Hewlett-Packard computers, software, and electronic instruments. Electronic interchange of material between departments is often desirable; for example, a manual written in one country may be translated to the local language in another. Interchange is complicated by the assortment of text processing tools used (which includes TeX) and the corresponding differences in markup as well as by the diverse hardware on which the tools are installed. The same problems hamper communication between the staff of different writing groups.

SGML supplies a means of standardizing markup conventions throughout the company, thereby allowing interchange of files without requiring replacement of all existing text processing software and the corresponding hardware. A shared markup technique also provides a vehicle for discussion among writers in different groups.

As an internal tool to aid in the production of user documentation, Hewlett-Packard has therefore developed an SGML parser and application generator called MARKUP. MARKUP allows SGML to be used as a front-end to other text-processing systems. A document-type definition that represents the structure of Hewlett-Packard user documentation has been developed and successfully compared to segments of different published manuals. The first MARKUP application, scheduled for beta testing mid-summer of 1987, uses TeX to print documentation from source files coded in SGML.

A MARKUP application is specified by a table which indicates the processing to be performed for each instance of every element included in the document-type definition. Table entries specify actions to be taken at the beginning of the element, within it, and at its end. When the MARKUP application generates a TeX source file analogous to the original SGML input, the actions usually consist of the TeX markup corresponding to the SGML construct. For example, the string “`{\it`” might be generated at the beginning of a book title, an introduced term, or a variable component in a computer command, while “`}`” is generated at the end of these structures. When a quotation mark occurs within normal text, the TeX open-quote convention “‘‘” is generated; when a quotation mark occurs within a quote element, the close-quote sequence “’’” is output.

When necessary, actions can also be entered as C code to be executed when the corresponding structure occurs. For example, C code is used to process forward and backward cross-references and to verify that every term introduced in the text is entered in the glossary.

Use of TEX with MARKUP differs from use of TEX by itself. For example, consider the TEX code used to start a chapter. Should a macro be defined for this purpose? Not necessarily. Macros are used to give a convenient label to sequences of instructions that are needed repeatedly. In this case, the code is isolated in the start-chapter cell of MARKUP's definition table. MARKUP invokes it as often as needed and, in effect, it has already been given a logical name (start-chapter).

However, debugging is simplified when macros are used; the TEX source file generated by MARKUP is more readable when it contains macro calls. The macros are not parameterized as they would be if the calls were user-written instead of automatically generated. For example, suppose that the chapter title is normally printed on the inside margin of the page header, but that the user can specify a different header if the chapter title is too long to fit in available space. User-invoked macros should be designed for the usual case. The chapter macro needs one parameter, the chapter title. To override the default page header, the user can call a second macro. When macros are automatically invoked, however, the chapter macro can have two parameters, the chapter title and the header specification, even though the values are usually identical. This repetition is not tedious to the user, since he enters the chapter title only once. Furthermore, there is no risk that two copies intended to be identical will in fact differ.

TUG and the Standards Community

In the United States, standards work on SGML began in ANSI Committee X3J6 and then moved to X3V1. TUG has maintained liaison with these committees since 1982, and TUGBoat regularly publishes liaison reports. Larry Beck is the current representative.

When TUG first sent me to an X3J6 meeting in January, 1982, my goal was to explain TEX concepts such as boxes and glue to committee members. I would like to belatedly thank TUG for my current involvement with SGML and am delighted to take this opportunity to convey information in the other direction.

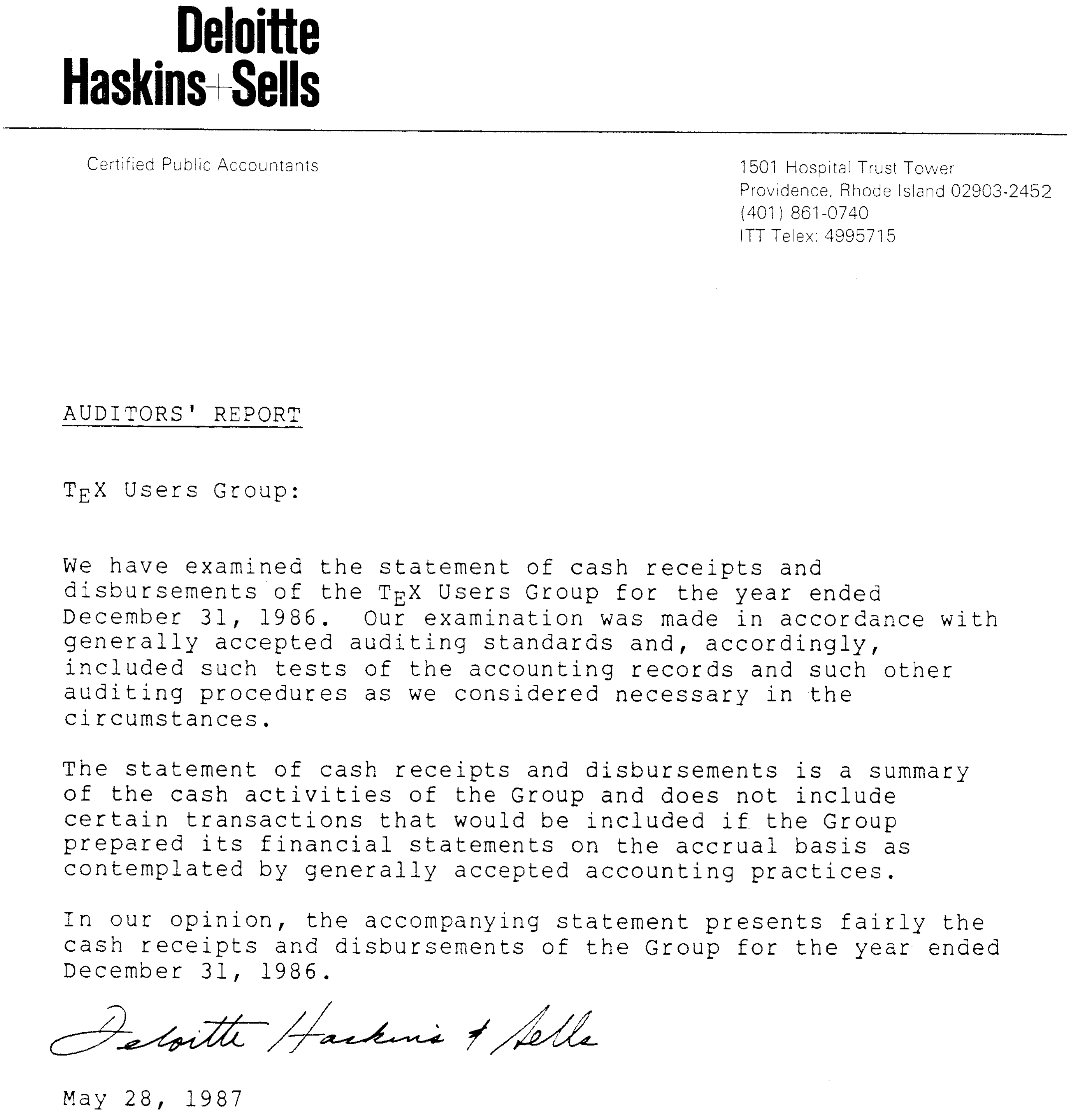
TUG Business

Treasurer's Report

For the first time, TUG's financial statements have been audited. We invited the firm of Deloitte Haskins & Sells to examine our records. DH&S have stated their conclusions in the report which appears on the following pages. The report shows that cash receipts during 1986 exceeded cash disbursements by $98,000. The auditors' letter reflects their opinion that TUG's accounts might be more fairly stated were they reported on an accrual, rather than a cash basis, a change TUG plans for its 1987 reports (for example, although TUG did have an excellent year in 1986, a large portion of the ending cash balance represents 1987 membership dues, paid in advance during 1986).

It should be noted that one of TUG's fiscal goals is the building of an available reserve equal to one year's operating budget, a policy consistent with the practice of other non-profit societies, including the American Mathematical Society. Cash reserves at the end of 1986 totaled $143,000, as compared with operating expenses of $406,000.

Samuel B. Whidden, Treasurer



Deloitte
Haskins+Sells

Certified Public Accountants

1501 Hospital Trust Tower
Providence, Rhode Island 02903-2452
(401) 861-0740
ITT Telex: 4995715

AUDITORS' REPORT

TeX Users Group:

We have examined the statement of cash receipts and disbursements of the TeX Users Group for the year ended December 31, 1986. Our examination was made in accordance with generally accepted auditing standards and, accordingly, included such tests of the accounting records and such other auditing procedures as we considered necessary in the circumstances.

The statement of cash receipts and disbursements is a summary of the cash activities of the Group and does not include certain transactions that would be included if the Group prepared its financial statements on the accrual basis as contemplated by generally accepted accounting practices.

In our opinion, the accompanying statement presents fairly the cash receipts and disbursements of the Group for the year ended December 31, 1986.

Deloitte Haskins & Sells

May 28, 1987

TEX USERS GROUP

STATEMENT OF CASH RECEIPTS AND DISBURSEMENTS
FOR THE YEAR ENDED DECEMBER 31, 1986

	1986
RECEIPTS:	
Dues	$ 98,404
Meetings and courses	213,194
Publications	170,342
Advertising	7,999
Royalties	8,807
Interest	5,298
Miscellaneous	1,190
Total	505,234
DISBURSEMENTS:	
Newsletter	29,698
Meetings and courses	88,500
Cost of publications	121,034
Administrative costs	134,881
Contribution	18,750
Refunds	11,999
Exhibits	1,453
Total	406,315
EXCESS OF RECEIPTS OVER DISBURSEMENTS	98,919
CASH BALANCE, BEGINNING OF THE YEAR	44,584
CASH BALANCE, END OF THE YEAR	$143,503

See note to statement of cash
receipts and disbursements.

NOTE TO STATEMENT OF CASH RECEIPTS AND DISBURSEMENTS

The TEX Users Group is a not-for-profit membership organization which provides information and technical assistance to the users of TEX, a sophisticated typesetting computer application.

Institutional Members

Addison-Wesley Publishing Company, *Reading, Massachusetts*

The Aerospace Corporation, *El Segundo, California*

Allied-Signal Canada, Inc., *Mississauga, Ontario, Canada*

American Mathematical Society, *Providence, Rhode Island*

ArborText, Inc., *Ann Arbor, Michigan*

ASCII Corporation, *Tokyo, Japan*

Aston University, *Birmingham, England*

California Institute of Technology, *Pasadena, California*

CALMA, *Sunnyvale, California*

Calvin College, *Grand Rapids, Michigan*

Canon, Inc., Office System Center, *Tokyo, Japan*

Carleton University, *Ottawa, Ontario, Canada*

CDS/WordWorks, *Davenport, Iowa*

Centre Inter-Régional de Calcul Électronique, CNRS, *Orsay, France*

Centro Internacional De Mejoramiento De Maiz Y Trigo (CIMMYT), *México, D.F., Mexico*

City University of New York, *New York, New York*

College of St. Thomas, Computing Center, *St. Paul, Minnesota*

College of William & Mary, Department of Computer Science, *Williamsburg, Virginia*

Columbia University, Center for Computing Activities, *New York, New York*

COS Information, *Montreal, P. Q., Canada*

DECUS, *Marlboro, Massachusetts*

Data General Corporation, *Westboro, Massachusetts*

Digital Equipment Corporation, *Nashua, New Hampshire*

Dowell Schlumberger Inc., *Tulsa, Oklahoma*

Edinboro University of Pennsylvania, *Edinboro, Pennsylvania*

Electricité de France, *Clamart, France*

Environmental Research Institute of Michigan, *Ann Arbor, Michigan*

European Southern Observatory, *Garching bei München, Federal Republic of Germany*

Ford Aerospace & Communications Corporation, *Palo Alto, California*

Försvarets Materielverk, *Stockholm, Sweden*

FTL systems Incorporated, *Toronto, Ontario, Canada*

General Motors Research Laboratories, *Warren, Michigan*

Geophysical Company of Norway A/S, *Stavanger, Norway*

Grinnell College, Computer Services, *Grinnell, Iowa*

Grumman Corporation, *Bethpage, New York*

GTE Laboratories, *Waltham, Massachusetts*

Hart Information Systems, *Austin, Texas*

Hartford Graduate Center, *Hartford, Connecticut*

Harvard University, Computer Services, *Cambridge, Massachusetts*

Hewlett-Packard Co., *Boise, Idaho*

Hobart & William Smith Colleges, *Geneva, New York*

Hutchinson Community College, *Hutchinson, Kansas*

Humboldt State University, *Arcata, California*

IBM Corporation, Scientific Center, *Palo Alto, California*

Illinois Bell Telephone, *Chicago, Illinois*

Illinois Institute of Technology, *Chicago, Illinois*

Imagen, *Santa Clara, California*

Institute for Advanced Study, *Princeton, New Jersey*

Institute for Defense Analyses, Communications Research Division, *Princeton, New Jersey*

Intergraph Corporation, *Huntsville, Alabama*

Intevep S. A., *Caracas, Venezuela*

Iowa State University, *Ames, Iowa*

Istituto di Cibernetica, Università degli Studi, *Milan, Italy*

Kuwait Institute for Scientific Research, *Safat, Kuwait*

Los Alamos National Laboratory, University of California, *Los Alamos, New Mexico*

Louisiana State University, *Baton Rouge, Louisiana*

Marquette University, Department of Mathematics, Statistics, and Computer Science, *Milwaukee, Wisconsin*

Massachusetts Institute of Technology, Artificial Intelligence Laboratory, *Cambridge, Massachusetts*

Mathematical Reviews, American Mathematical Society, *Ann Arbor, Michigan*

Max Planck Institute Stuttgart, *Stuttgart, Federal Republic of Germany*

McGill University, *Montreal, Quebec, Canada*

McGraw-Hill, Inc., *Englewood, Colorado*

National Research Council Canada, Computation Centre, *Ottawa, Ontario, Canada*

New Jersey Institute of Technology, *Newark, New Jersey*

New York University, Academic Computing Facility, *New York, New York*

Northeastern University, Academic Computing Services, *Boston, Massachusetts*

Online Computer Library Center, Inc. (OCLC), *Dublin, Ohio*

Pennsylvania State University, Computation Center, *University Park, Pennsylvania*

Personal TEX, Incorporated, *Mill Valley, California*

Purdue University, *West Lafayette, Indiana*

QMS, Inc, *Mobile, Alabama*

Queens College, *Flushing, New York*

Research Triangle Institute, *Research Triangle Park, North Carolina*

RE/SPEC, Inc., *Rapid City, South Dakota*

Ruhr Universität Bochum, *Bochum, Federal Republic of Germany*

Rutgers University, Hill Center, *Piscataway, New Jersey*

St. Albans School, *Mount St. Alban, Washington, D.C.*

Sandia National Laboratories, *Albuquerque, New Mexico*

SAS Institute, *Cary, North Carolina*

Schlumberger Offshore Services, *New Orleans, Louisiana*

Schlumberger Well Services, *Houston, Texas*

Science Applications International Corp., *Oak Ridge, Tennessee*

I. P. Sharp Associates, *Palo Alto, California*

Smithsonian Astrophysical Observatory, Computation Facility, *Cambridge, Massachusetts*

Software Research Associates, *Tokyo, Japan*

Sony Corporation, *Atsugi, Japan*

Space Telescope Science Institute, *Baltimore, Maryland*

Springer-Verlag, *Heidelberg, Federal Republic of Germany*

Stanford Linear Accelerator Center (SLAC), *Stanford, California*

Stanford University, Computer Science Department, *Stanford, California*

Stanford University, ITS Graphics & Computer Systems, *Stanford, California*

State University of New York, Department of Computer Science, *Stony Brook, New York*

Stratus Computer, Inc., *Marlboro, Massachusetts*

Syracuse University, *Syracuse, New York*

Talaris Systems, Inc., *San Diego, California*

Texas A & M University, Department of Computer Science, *College Station, Texas*

Texas A & M University, Computing Services Center, *College Station, Texas*

Texas Accelerator Center, *The Woodlands, Texas*

TRW, Inc., *Redondo Beach, California*

Tufts University, *Medford, Massachusetts*

TV Guide, *Radnor, Pennsylvania*

TYX Corporation, *Reston, Virginia*

UNI.C, Danmarks EDB-Center, *Aarhus, Denmark*

University of Alabama, *Tuscaloosa, Alabama*

University of British Columbia, *Vancouver, British Columbia, Canada*

University of Calgary, *Calgary, Alberta, Canada*

University of California, Berkeley, Computer Science Division, *Berkeley, California*

University of California, San Diego, *La Jolla, California*

University of California, San Francisco, *San Francisco, California*

University of Chicago, Computation Center, *Chicago, Illinois*

University of Chicago, Computer Science Department, *Chicago, Illinois*

University of Chicago, Graduate School of Business, *Chicago, Illinois*

University of Delaware, *Newark, Delaware*

University of Glasgow, *Glasgow, Scotland*

University of Groningen, *Groningen, The Netherlands*

University of Illinois at Chicago, Computer Center, *Chicago, Illinois*

University of Kansas, Academic Computing Services, *Lawrence, Kansas*

University of Maryland, *College Park, Maryland*

University of Massachusetts, *Amherst, Massachusetts*

University of North Carolina, School of Public Health, *Chapel Hill, North Carolina*

University of Oslo, Institute of Informatics, *Blindern, Oslo, Norway*

University of Ottawa, *Ottawa, Ontario, Canada*

University of Southern California, Information Sciences Institute, *Marina del Rey, California*

University of Tennessee at Knoxville, Department of Electrical Engineering, *Knoxville, Tennessee*

University of Texas at Austin, Physics Department, *Austin, Texas*

University of Texas at Dallas, Center for Space Science, *Dallas, Texas*

University of Washington, Department of Computer Science, *Seattle, Washington*

University of Western Australia, Regional Computing Centre, *Nedlands, Australia*

University of Wisconsin, Academic Computing Center, *Madison, Wisconsin*

Vanderbilt University, *Nashville, Tennessee*

Vereinigte Aluminium-Werke AG, *Bonn, Federal Republic of Germany*

Villanova University, *Villanova, Pennsylvania*

Washington State University, *Pullman, Washington*

Widener University, Computing Services, *Chester, Pennsylvania*

Worcester Polytechnic Institute, *Worcester, Massachusetts*

Yale University, Department of Computer Science, *New Haven, Connecticut*

TeX Users Group **1987** **Membership Form**

Request for Information

The TeX Users Group maintains a database and publishes a membership list containing information about the equipment on which members' organizations plan to or have installed TeX, and about the applications for which TeX would be used. This list is updated periodically and distributed to members with TUGboat, to permit them to identify others with similar interests. Thus, it is important that the information be complete and up-to-date.

Please answer the questions below, in particular those regarding the status of TeX and the hardware on which it runs or is being installed. (Operating system information is particularly important in the case of IBM mainframes and VAX.) This hardware information is used to group members in the listings by computer and output device.

If accurate information has already been provided by another TUG member at your site, you may indicate that member's name, and the information will be repeated.

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

- *Send completed form with remittance*
  (checks, money orders, UNESCO coupons) to:
  TeX Users Group
  P. O. Box 594
  Providence, Rhode Island 02901, U.S.A.
- *For foreign bank transfers*
  direct payment to the TeX Users Group, account #002-031375, at:
  Rhode Island Hospital Trust National Bank
  One Hospital Trust Plaza
  Providence, Rhode Island 02903-2449, U.S.A.
- *General correspondence*
  about TUG should be addressed to:
  TeX Users Group
  P. O. Box 9506
  Providence, Rhode Island 02940-9506, U.S.A.

Name: ______________________________

Home [ ]
Bus. [ ] Address: ______________________________

____

____

____

QTY	ITEM	AMOUNT
	1987 TUGboat Subscription/TUG Membership (Jan.–Dec.) – **North America** New (first-time): [ ] $30.00 each Renewal: [ ] $40.00; [ ] $30.00 – reduced rate if renewed before January 31, 1987	
	1987 TUGboat Subscription/TUG Membership (Jan.–Dec.) – **Outside North America** New (first-time): [ ] $40.00 each Renewal: [ ] $45.00; [ ] $40.00 – reduced rate if renewed before January 31, 1987	
	TUGboat back issues, $15.00 per issue (16), circle issue(s) desired: 1980 (v. 1) #1 1981 (v. 2) #1, 2, 3 1982 (v. 3) #1, 2 1983 (v. 4) #1, 2 1984 (v. 5) #1, 2 1985 (v. 6) #1, 2, 3 1986 (v. 7) #1, 2, 3	

Air mail postage is included in the rates for all subscriptions
and memberships outside North America.
Quantity discounts available on request.

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

* * * *

Membership List Information

Institution (if not part of address):

Title:

Phone:

Network address: [ ] Arpanet [ ] BITnet
[ ] CSnet [ ] uucp

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: [ ] Under consideration
[ ] Being installed
[ ] Up and running since
Approximate number of users:

Version of TeX: [ ] SAIL
Pascal: [ ] TeX82 [ ] TeX80
[ ] Other (describe)

From whom obtained:

Hardware on which TeX is to be used:

Computer(s) Operating system(s) Output device(s)

Please answer the following questions regarding output devices used with TEX
if this form has never been filled out for your site, or if you have new information.
Use a separate form for each output device.

Name ______________________________ Institution ______________________________

A. Output device information
Device name
Model
1. Knowledgeable contact at your site
Name
Telephone
2. Device resolution (dots/inch)
3. Print speed (average feet/minute in graphics mode)
4. Physical size of device (height, width, depth)

5. Purchase price
6. Device type
[ ] photographic [ ] electrostatic
[ ] impact [ ] other (describe)

7. Paper feed [ ] tractor feed
[ ] friction, continuous form
[ ] friction, sheet feed [ ] other (describe)

8. Paper characteristics
a. Paper type required by device
[ ] plain [ ] electrostatic
[ ] photographic [ ] other (describe)

b. Special forms that can be used [ ] none
[ ] preprinted one-part [ ] multi-part
[ ] card stock [ ] other (describe)

c. Paper dimensions (width, length)
maximum
usable
9. Print mode
[ ] Character: ( ) Ascii ( ) Other
[ ] Graphics [ ] Both char/graphics
10. Reliability of device
[ ] Good [ ] Fair [ ] Poor
11. Maintenance required
[ ] Heavy [ ] Medium [ ] Light
12. Recommended usage level
[ ] Heavy [ ] Medium [ ] Light
13. Manufacturer information
a. Manufacturer name
Contact person
Address

Telephone
b. Delivery time
c. Service [ ] Reliable [ ] Unreliable

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

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

D. Separate interface hardware (if any) between host computer and output device (e.g. Z80)
1. Separate interface hardware not needed because:
[ ] Output device is run off-line
[ ] O/D contains user-programmable micro
[ ] Decided to drive O/D direct from host
2. Name of interface device (if more than one, specify for each)

3. Manufacturer information
a. Manufacturer name
Contact person
Address

Telephone
b. Delivery time
c. Purchase price
4. Modifications
[ ] Specified by Stanford
[ ] Designed/built in-house
[ ] Other (explain)

5. Software for interface device
[ ] Obtained from Stanford
[ ] Written in-house
[ ] Other (explain)

E. Fonts being used
[ ] Computer Modern
[ ] Fonts supplied by manufacturer
[ ] Other (explain)

1. From whom were fonts obtained?

2. Are you using Metafont? [ ] Yes [ ] No

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

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

H. Comments – overview of output device

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

TEX Users Group **1987** **Membership Form**

Request for Information

The TEX Users Group maintains a database and publishes a membership list containing information about the equipment on which members' organizations plan to or have installed TEX, and about the applications for which TEX would be used. This list is updated periodically and distributed to members with TUGboat, to permit them to identify others with similar interests. Thus, it is important that the information be complete and up-to-date.

Please answer the questions below, in particular those regarding the status of TEX and the hardware on which it runs or is being installed. (Operating system information is particularly important in the case of IBM mainframes and VAX.) This hardware information is used to group members in the listings by computer and output device.

If accurate information has already been provided by another TUG member at your site, you may indicate that member's name, and the information will be repeated.

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

- *Send completed form with remittance* (checks, money orders, UNESCO coupons) to:
  TEX Users Group
  P. O. Box 594
  Providence, Rhode Island 02901, U.S.A.
- *For foreign bank transfers* direct payment to the TEX Users Group, account #002-031375, at:
  Rhode Island Hospital Trust National Bank
  One Hospital Trust Plaza
  Providence, Rhode Island 02903-2449, U.S.A.
- *General correspondence* about TUG should be addressed to:
  TEX Users Group
  P. O. Box 9506
  Providence, Rhode Island 02940-9506, U.S.A.

Name: ____________________

Home [ ]
Bus. [ ] Address: ____________________

____

____

____

QTY	ITEM	AMOUNT
	1987 TUGboat Subscription/TUG Membership (Jan.–Dec.) – **North America** New (first-time): [ ] $30.00 each Renewal: [ ] $40.00; [ ] $30.00 – reduced rate if renewed before January 31, 1987	
	1987 TUGboat Subscription/TUG Membership (Jan.–Dec.) – **Outside North America** New (first-time): [ ] $40.00 each Renewal: [ ] $45.00; [ ] $40.00 – reduced rate if renewed before January 31, 1987	
	TUGboat back issues, $15.00 per issue (16), circle issue(s) desired: 1980 (v. 1) #1 1981 (v. 2) #1, 2, 3 1982 (v. 3) #1, 2 1983 (v. 4) #1, 2 1984 (v. 5) #1, 2 1985 (v. 6) #1, 2, 3 1986 (v. 7) #1, 2, 3	

Air mail postage is included in the rates for all subscriptions and memberships outside North America.
Quantity discounts available on request.

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

* * * *

Membership List Information

Institution (if not part of address):

Title:

Phone:

Network address: [ ] Arpanet [ ] BITnet
[ ] CSnet [ ] uucp

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: [ ] Under consideration
[ ] Being installed
[ ] Up and running since
Approximate number of users:

Version of TEX: [ ] SAIL
Pascal: [ ] TEX82 [ ] TEX80
[ ] Other (describe)

From whom obtained:

Hardware on which TEX is to be used:

Computer(s)	Operating system(s)	Output device(s)

Please answer the following questions regarding output devices used with TEX
if this form has never been filled out for your site, or if you have new information.
Use a separate form for each output device.

Name ______________________ Institution ______________________

A. Output device information
   Device name
   Model
1. Knowledgeable contact at your site
   Name
   Telephone
2. Device resolution (dots/inch)
3. Print speed (average feet/minute in graphics mode)
4. Physical size of device (height, width, depth)

5. Purchase price
6. Device type
   [ ] photographic [ ] electrostatic
   [ ] impact [ ] other (describe)

7. Paper feed [ ] tractor feed
   [ ] friction, continuous form
   [ ] friction, sheet feed [ ] other (describe)

8. Paper characteristics
   a. Paper type required by device
      [ ] plain [ ] electrostatic
      [ ] photographic [ ] other (describe)

   b. Special forms that can be used [ ] *none*
      [ ] preprinted one-part [ ] multi-part
      [ ] card stock [ ] other (describe)

   c. Paper dimensions (width, length)
      maximum
      usable
9. Print mode
   [ ] Character: ( ) Ascii ( ) Other
   [ ] Graphics [ ] Both char/graphics
10. Reliability of device
   [ ] Good [ ] Fair [ ] Poor
11. Maintenance required
   [ ] Heavy [ ] Medium [ ] Light
12. Recommended usage level
   [ ] Heavy [ ] Medium [ ] Light
13. Manufacturer information
   a. Manufacturer name
      Contact person
      Address

      Telephone
   b. Delivery time
   c. Service [ ] Reliable [ ] Unreliable

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

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

D. Separate interface hardware (if any) between host computer and output device (e.g. Z80)
1. Separate interface hardware not needed because:
   [ ] Output device is run off-line
   [ ] O/D contains user-programmable micro
   [ ] Decided to drive O/D direct from host
2. Name of interface device (if more than one, specify for each)

3. Manufacturer information
   a. Manufacturer name
      Contact person
      Address

      Telephone
   b. Delivery time
   c. Purchase price
4. Modifications
   [ ] Specified by Stanford
   [ ] Designed/built in-house
   [ ] Other (explain)

5. Software for interface device
   [ ] Obtained from Stanford
   [ ] Written in-house
   [ ] Other (explain)

E. Fonts being used
   [ ] Computer Modern
   [ ] Fonts supplied by manufacturer
   [ ] Other (explain)

1. From whom were fonts obtained?

2. Are you using Metafont? [ ] Yes [ ] No

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

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

H. Comments – overview of output device

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

TeX Order Form

The current versions of the public domain TeX software, as produced by Stanford University, are available from Maria Code by special arrangement with the Computer Science Department.

Several versions of the distribution tape are available. The generic ASCII and EBCDIC tapes will require a Pascal compiler at your installation. Each tape contains the source of TeX, and WEB (a precompiler language in which TeX is written), and **METAFONT**. It also contains font descriptions for Computer Modern, macros for AMS-TeX, LaTeX, SliTeX and HP TeX, some sample "change files", and many other odds and ends.

Ready-to-run versions of TeX are available for DEC VAX/VMS, IBM VM/CMS and DEC 20/TOPS-20 formats. They contain everything on the generic tape as well as the compiled programs. This means that you will not need a Pascal compiler unless you want to make source changes. Order these tapes if and only if you have one of these systems.

The font tapes contain GF files for the Computer Modern fonts. While it is possible to generate these files yourself, it will save you a lot of CPU time to get them on the tape.

The price of the tapes now includes the cost of the tape reels. Either 1200′ or 2400′ reels will be used depending on the needed capacity. If you order a distribution tape and a font tape, they will most likely be put on a single 2400′ foot reel. All tapes are 1600 bpi.

Please take care to fill in the order form carefully. Note that postage (other than domestic book rate, which is free) is based on the total weight and postal class which you select. Sales tax is added for orders with a shipping address in California.

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

Make checks payable to *Maria Code*. Export orders must send checks which are drawn on a US bank. International money orders are fine. Purchase orders are accepted if your company has a policy of prompt payment (30 days maximum).

Your order will be filled with the current versions of software and manuals at the time it is received. Since some versions are "pre-announced", please indicate if you want to wait for a specific version.

Telephone calls are discouraged, but if you must call, please do so between 9:30 a.m. and 2:30 p.m. West Coast time. The number for Maria Code is (408) 735-8006. Do not call for advice or technical assistance since no one is there who can help you. You may try Stanford or some other of the helpful people whose names appear in TUGboat.

TeX Order Form

Distribution tapes:

ASCII generic format ____
EBCDIC generic format ____
VAX/VMS Backup format ____
DEC 20/TOPS-20 Dumper format ____
IBM VM/CMS format ____

Tape prices: $92 for first tape,
$72 for each additional tape

Font tapes (GF files):

200/240 dpi CM fonts ____
300 dpi CM fonts ____

Allow 2 lbs shipping weight for each tape ordered.

Documents:

	Price $	Weight	Quantity
TeXbook (vol. A) softcover	25.00	2	____
TeX: The Program (vol. B)	37.00	4	____
METAFONTbook (vol. C) softcover	22.00	2	____
METAFONT the Program (vol. D)	37.00	4	____
Computer Modern Typefaces (vol. E)	37.00	4	____
WEB language *	12.00	1	____
TeXware *	10.00	1	____
BibTeX *	10.00	1	____
Torture Test for TeX *	8.00	1	____
Torture Test for **METAFONT** *	8.00	1	____
LaTeX - document preparation system	25.00	2	____

* published by Stanford University

Payment calculation:

Number of tapes ordered ________ Total price for tapes ____________
Number of documents ordered ________ Total price for documents ____________
Add the 2 lines above ____________
Orders from within California: Add **sales tax** for your location ____________

Shipping charges: (for domestic book rate, skip this section)

Total weight of tapes and books ________ lbs.
Check type of shipping and note rate:

____ domestic priority mail: rate $ 1.00/lb.
____ air mail to Canada and Mexico: rate $ 1.50/lb.
____ export surface mail (all countries): rate $ 1.00/lb.
____ air mail to Europe, South America: rate $ 4.00/lb.
____ air mail to Far East, Africa, Israel: rate $ 6.00/lb.

Multiply total weight by shipping rate. Enter **shipping charges:** ____________

Total charges: (add charges for materials, tax and shipping) ____________

Methods of payment: Check drawn on a US bank. Make payable to Maria Code.
International money order.
Purchase order (maximum 30 days allowed for payment)

Send order to: Maria Code, Data Processing Services,
1371 Sydney Drive, Sunnyvale, CA 94087

Name and address for shipment: ______________________________

____

____

Contact person (if different from above): ______________________________

Telephone: ______________________________

Output Devices

Index to Sample Output from Various Devices

Camera copy for the following items in this issue of TUGboat was prepared on the devices indicated, and can be taken as representative of the output produced by those devices. Some items (noted below) were received as copy larger than 100%; these were reduced photographically using the PMT process. The bulk of this issue has been prepared at the American Mathematical Society, on a VAX 8600 (VMS) and output on an APS-μ5 using a newly-installed set of resident CM fonts.

- Apple LaserWriter (300 dpi):
  Alec Dunn, *Using PostScript with TeX*, pp. 171*ff*; VAX 11/780 (VMS).
  - FTL systems Inc. advertisement, cover 3.
  - ArborText advertisement, p. 244.
- Autologic APS-μ5 (1440 dpi):
  Donald E. Knuth, *Digital halftones*, pp. 135*ff*, except for the last two pages, which were set at AMS; DEC 10 and DEC 2065.
- Canon CX (300 dpi):
  Metafoundry advertisement, p. 252.
- Corona LP300 (300dpi):
  Personal TeX advertisement, p. 238.
- IBM 3820 (240 dpi):
  Gil Pierson, *SAS merged with TeX*, p. 178; Amdahl 470V/8 (MVS).
- Imagen 8/300 (300dpi):
  Donald E. Knuth, some of the illustrations (as marked) in *Digital halftones*, pp. 135*ff*.
  - Charles Lehardy, *Diglot typesetting*, pp. 190*ff* (article reduced to 95%, example reduced to 90%); Integrated Solutions 6810 (Unix).
  - Yasuki Saito, *Report on jTeX: A Japanese TeX*, pp. 103*ff*; DEC 2065.

Coming next issue

Generating an APL font

	0	1	2	3	4	5	6	7
'000		⎕	∇	∆	⍺	⍵	∊	¨
'010	∨	⋄	≤	≥	⍴	⍳	∘	○
'020	∪	∩	⊂	⊃	_	¯	~	≠
'030	⊤	⊥	⊣	⊢	⌊	⌈	→	↓
'040		!	⍝	×	$	÷	∧	'
'050	(	)	⋆	+	,	−	.	/
'060	0	1	2	3	4	5	6	7
'070	8	9	:	;	<	=	>	?
'100		*A*	*B*	*C*	*D*	*E*	*F*	*G*
'110	*H*	*I*	*J*	*K*	*L*	*M*	*N*	*O*
'120	*P*	*Q*	*R*	*S*	*T*	*U*	*V*	*W*
'130	*X*	*Y*	*Z*	[	\	]	↑	←
'140		a	b	c	d	e	f	g
'150	h	i	j	k	l	m	n	o
'160	p	q	r	s	t	u	v	w
'170	x	y	z	{	|	}		

Personal
TEX
Inc

Now for the PC!
TEX 2.0, METAFONT 1.0!

... now offers a list of software, fonts and hardware so that we can be your complete TEX outfitter for PC and AT workstations. We have joined forces with ArborText, n^2 Computer Consultants, the Metafoundry, FTL Systems, and Aurion Technology to bring you these products:

SOFTWARE:

PCTEX® A full TEX82, version 2.0, including INITEX, LATEX 2.09, LATEX User's Guide, AMS-TEX, and Mike Spivak's PCTEX Manual and VANILLA macro package. 33% faster than version 1.01! Runs LATEX in 512K RAM! **$249.**

PC•MF A full METAFONT, version 1.0, for the PC/XT, AT and compatibles. Includes The METAFONTbook and a complete set of Computer Modern typeface description files. Useful for scaling existing fonts and creating new ones. **$195.**

PCDOT Device drivers for dot-matrix printers. **$95. each.**

PCLaser Device drivers for laser printers, including HP LaserJet Plus and PostScript devices. **$175.** to **$225. each.**

Preview ArborText's popular Preview for the PC, now with a host of new features, including side-by-side page viewing and vertical scrolling through a document. **$175.**

MAXview A new screen preview program, written by Max Diaz of Aurion Technology. This program has a good basic set of features and works with 13 different graphics adapters, including CGA, EGA and Hercules. **$125.**

DVISCRN A new screen preview program from n^2 Computer Consultants. This program also has a good basic set of features, and works with a variety of graphics adapters, including CGA, EGA and Hercules. **$125.**

MacTEX TEX for the Macintosh, from FTL Systems. Includes Editor, Preview and PostScript driver. **$750.** plus **$100.** for each additional license.

FONTS:

MF Medley Chel fonts (Computer Helvetica, shown here), and Copperplate, Black Letter and Schoolbook headline fonts. **$100.**

HARDWARE:

Cordata Laser Printer Includes PCTEX, driver and fonts. Only **$2695.**

JLASER Makes any Canon LBP-CX laser engine a TEX device. From **$699.**

Join thousands of satisfied PCTEX users. Write or call us today for complete product information. Inquire about educational and corporate discounts and site licensing.
PCTEX Bulletin Board: (415)388-1708. 300/1200/2400 baud.

System requirements: DOS 2.0 or later, 512K RAM, 10M hard disk. Screen preview programs require appropriate graphics adapter. Corona Laser Printer requires additional 512K RAM disk. Prices are for U.S. only and do not include shipping. Orders outside the U.S. must be placed through distributors listed below. MasterCard, Visa accepted.

Personal
TEX
Inc

12 Madrona Street
Mill Valley, CA 94941
(415) 388-8853 Telex 510-601-0672

Distributors: **Canada:** Docusoft, Vancouver, (604) 687-0354; **UK:** UniTEX Systems, Sheffield, (0742) 351 489; **Germany:** Kettler EDV-Consulting, Lenggries, (08042) 8081; **France:** Probe Informatique, Trappes, (01) 3062 2603; **Ireland:** UniTEX, Dublin, 772941 x1983; **Australia:** The Wordworks, ACT, (062) 572893; **Mexico:** Aurion, Mexico City, (905) 531-9748; **Scandinavia:** Interbase, Vedbæk DK, (02) 894847.

Trademarks: PCTEX, Personal TEX, Inc.; MacTEX, FTL Systems; TEX, American Mathematical Society. Manufacturer's product names are trademarks of individual manufacturers. This ad produced using PCTEX, and printed on a Cordata LP-300.

MacroTEX Table Making . . .

```
\verticallines
\horizontallines
\autosizetable
This is&Autosizetable\cr
The Width&of Each Column\cr
is automatically determined&
by the width of the
contents.\cr
\endautosizetable
```

This is	**Autosizetable**
The Width	of Each Column
is automatically determined	by the width of the contents.

```
\horizontallines
\verticallines
\autowraptable{.5\hsize}
{.5\hsize}*
This is&Autowraptable\cr
Contents of each column will
automatically wrap within
the column.&
As you see in these
examples\dots\cr
\begincode
Verbatim text can
be included
~ % ^ & \
\endcode
&Display Equations:
$$\lambda=\lambda'+i\lambda
\eqno{(3.4)}$$\cr
\autocenter
Autosizetable and
Autowraptable
will

continue over
many pages
automatically
\endautocenter
&Macro\TeX\ Listing:
\beginlist{\outline}
\list*
Here is a
\list* Tiny List

More things
\endlistlevel
Back again
\endlist\endautowraptable
```

This is	**Autowraptable**
Contents of each column will automatically wrap within the column.	As you see in these examples...
Verbatim text can be included ~ % ^ & \	Display Equations: $\lambda = \lambda' + i\lambda \qquad (3.4)$
Autosizetable and Autowraptable will continue over many pages automatically	MacroTEX Listing: I. Here is a A. Tiny List B. More things II. Back again

MacroTEX :

A TEX toolkit, containing Macros for Book and Software Documentation Formats, Table of Contents generation, Indexing, Cross-Referencing, Glossary production, and many more.

For information and order form, please write or call
Amy Hendrickson
57 Longwood Avenue, Brookline MA 02146
(617) 738-8029

TEXnology Inc.

The Joy of TeX

A Gourmet Guide to Typesetting with the $\mathcal{AMS}$-TeX macro package

M. D. SPIVAK, Ph.D.

The Joy of TeX is the user-friendly user's guide for $\mathcal{AMS}$-TeX, an extension of TeX, Donald Knuth's revolutionary program for typesetting technical material. $\mathcal{AMS}$-TeX was designed to simplify the input of mathematical material in particular, and to format the output according to any of various preset style specifications.

There are two primary features of the TeX system: it is a computer system for typesetting technical text, especially text containing a great deal of mathematics; and it is a system for producing beautiful text, comparable to the work of the finest printers.

Most importantly, TeX's capabilities are not available only to TeXperts. While mathematicians and experienced technical typists will find that TeX allows them to specify mathematical formulas with great accuracy and still have control over the finished product, even novice technical typists will find the manual easy to use in helping them produce beautiful technical TeXt.

This book is designed as a user's guide to the $\mathcal{AMS}$-TeX macro package and details many features of this extremely useful text processing package. Parts 1 and 2, entitled "Starters" and "Main Courses," teach the reader how to typeset most normally encountered text and mathematics. "Sauces and Pickles," the third section, treats more exotic problems and includes a 60-page dictionary of special TeXniques.

Exercises sprinkled generously through each chapter encourage the reader to sit down at a terminal and learn through experimentation. Appendixes list summaries of frequently used and more esoteric symbols as well as answers to the exercises.

ISBN 0-8218-2999-8, LC 85-7506
290 pages (softcover), 1986
AMS Indiv. Memb. $26, AMS Inst. Memb. $30, List price $33
To order specify JOYT/T
Shipping/Handling: 1st book $2, each add'l $1, max. $25; by air, 1st book $5, each add'l $3, max. $100

PREPAYMENT REQUIRED. Order from
American Mathematical Society
PO Box 1571
Annex Station
Providence, RI 02901-9930
or call 800-556-7774 to use VISA or MasterCard.

Prices subject to change.

When quality, price and service are important, TEXSource has the answer.

High quality typeset output

We're the leader in offering low prices and in providing fast turnaround for your DVI files. We can process your work for as little as $2.50 per page with overnight turnaround.

Typeset output for CMR fonts

Available soon. TEXSource will be able to output your CMR files on high quality resin-coated (RC) paper.

Times Roman proofing fonts

Text with math. See your Times Roman output on your laser printer or screen previewer before you send it to us for high quality output. We provide pixel (.pxl) and TEX font metric (.tfm) files.

Call for more information about our products and services.

(713) 520-7206

TEXSource

3333 West Alabama, Suite 111 / Houston, Texas 77098

TEXSource is pleased to announce that it is represented in Scandinavia by KAOS A/S in Oslo, Norway. Mr. Bruce Wolman is the person to contact and he can be reached at the following address:

KAOS A/S, Box 3169 Elisenberg, 0208 Oslo 2, NORWAY
Telephone: 2-59-02-94

The copy for this ad was set at TEXSource using Times Roman fonts.

COMPLETE
TYPESETTING
SERVICES

Math and Technical Book Publishers . . .

If you are creating your book files with TeX, Computer Composition Corporation can now offer the following services:

— Converting TeX DVI or source files to the fully paginated typeset page in either Computer Modern (from DVI files) or true Times Roman typefaces (from source files).

— Providing 300 dpi laser-printed page proofs (when source files are submitted) which simulate the typeset page exactly.

— Keyboarding services, from traditionally-prepared manuscripts via the TeX processing system.

— Camera work services, including half-tone, line-art, screens, and full page negatives.

Call or write us for sample pages in both Computer Modern and Times Roman.

COMPUTER COMPOSITION CORPORATION
1401 West Girard • Madison Heights, Michigan 48071 • (313) 545-4330

Cooke Publications

MathWriter™ & MW2TeX™

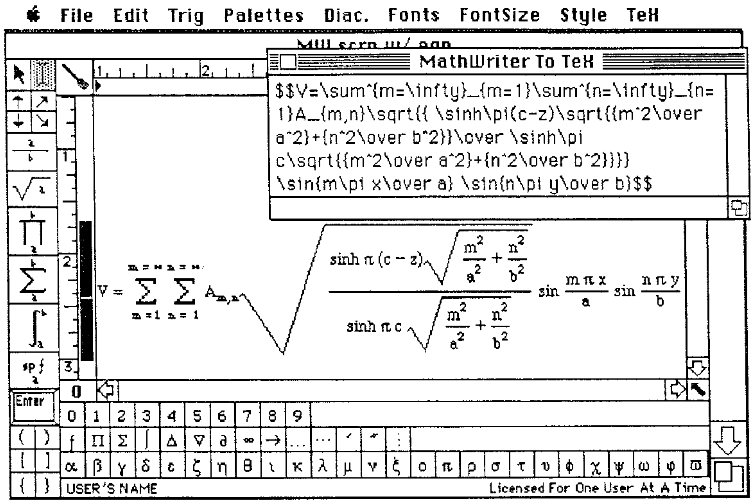
P.O. Box 4448, Ithaca, NY 14852 (800) 482-4438, ext 15, in NYS (800) 435-4438, ext 15

MathWriter, the elegant tool for creating mathematical expressions, has just become more useful to TUGboat readers. With the new Macintosh desk accessory, MW2TeX, you can automatically translate the 'wysiwyg' MathWriter data structures into standard TeX code! These two programs provide the much-needed visual tool for creating TeX code for mathematical expressions. Imagine being able to learn to produce TeX code for complex expressions in a few hours!

Features:

- Point and click or key-stroke entry of expressions in conventional notation
- Immediate visual feedback with interactive screen refresh
- Greek and user-selected alphabet on-screen
- Automatic handling of font sizing and character placement
- Automatic sizing of parentheses, braces, brackets, etc.
- Up to 10 expressions can be stored for repeated usage
- Extensive editing capability
- Choice of output printed : ImageWriter, LaserWriter
- Choice of external storage: clipboard, MacDraw, MacPaint and MathWriter files, and with MW2TeX you can also produce ASCII files of TeX code.

MathWriter™



```
$$V=\sum^{m=\infty}_{m=1}\sum^{n=\infty}_{n=
1}A_{m,n}\sqrt{{ \sinh\pi(c-z)\sqrt{{m^2\over
a^2}+{n^2\over b^2}}\over \sinh\pi
c\sqrt{{m^2\over a^2}+{n^2\over b^2}}}}
\sin{m\pi x\over a} \sin{n\pi y\over b}$$
```

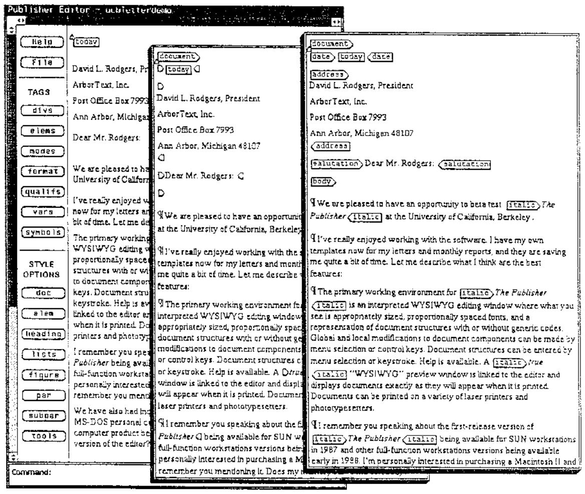
INEXPENSIVE - $99.90 license ($49.95 each) includes UPS shipping within US, 100+ page manual, Switcher™, and system files including printer drivers. Macintosh 512K and larger; a Mac II version is also available. US $; NYS residents include tax.

MathWriter and MW2TeX are trademarks of Cooke Publications. TeX is a trademark of the Amer. Math. Soc.

$$\left[\begin{pmatrix} N_x & N_{xy} \\ N_{yx} & N_y \end{pmatrix}\begin{pmatrix} \frac{\partial u}{\partial x} & \frac{1}{2}\left(\frac{\partial u}{\partial y}+\frac{\partial v}{\partial x}\right) \\ \frac{1}{2}\left(\frac{\partial v}{\partial x}+\frac{\partial u}{\partial y}\right) & \frac{\partial v}{\partial y} \end{pmatrix} + \frac{1}{2}\begin{pmatrix} N_x & N_{xy} \\ N_{yx} & N_y \end{pmatrix}\begin{pmatrix} \frac{\partial w}{\partial x} \\ \frac{\partial w}{\partial y} \end{pmatrix}\begin{pmatrix} \frac{\partial w}{\partial x} & \frac{\partial w}{\partial y} \end{pmatrix}\right]$$

MW2TeX™ DA

```
$$\left[\left(\matrix{{N_x}&{N_{xy}}\cr{N_{yx}}&{N_y}\cr}\right)\left(\matrix{{{\partial  u\over\partial
x}}&{{1\over2}\left({\partial u\over\partial y}+{\partial v\over\partial x}\right)}\cr{{1\over2}\left({\partial
v\over\partial x}+{\partial u\over\partial y}\right)}&{{\partial v\over\partial
y}}\cr}\right)+{1\over2}\left(\matrix{{N_x}&{N_{xy}}\cr{N_{yx}}&{N_y}\cr}\right)\left(\matrix{{{\partial  w\over\partial
x}}\cr{{\partial w\over\partial y}}\cr}\right)\left(\matrix{{{\partial w\over\partial x}}&{{\partial w\over\partial
y}}\cr}\right)\right]$$
```



document
date today date
address
David L. Rodgers, President

ArborText, Inc.

Post Office Box 7993

Ann Arbor, Michigan 48107
address

salutation Dear Mr. Rodgers: salutation

body

¶ We are pleased to have an opportunity to beta test italic *The Publisher* italic at the University of California, Berkeley .

¶ I've really enjoyed working with the software. I have my own templates now for my letters and monthly reports, and they are saving me quite a bit of time. Let me describe what I think are the best features:

¶ The primary working environment for italic *The Publisher* italic is an interpreted WYSIWYG editing window where what you see is appropriately sized, proportionally spaced fonts, and a representation of document structures with or without generic codes. Global and local modifications to document components can be made by menu selection or control keys. Document structures can be entered by menu selection or keystroke. Help is available. A italic *true* italic "WYSIWYG" preview window is linked to the editor and displays documents exactly as they will appear when it is printed. Documents can be printed on a variety of laser printers and phototypesetters.

¶ I remember you speaking about the first-release version of italic *The Publisher* italic being available for SUN workstations in 1987 and other full-function workstations versions being available early in 1988. I'm personally interested in purchasing a Macintosh II and

WHAT YOU SEE IS

With *The Publisher*$^{\mathrm{TM}}$, what you see is a great deal more than with other publishing systems. On the screen, you see a full function, sophisticated composition system with WYSIWYG math and tables, general purpose paint and draw software, an easy-to-use editor with mouse and keyboard commands, and a flexible set of style options for complex page layouts. On the printed page, you see TEX's high quality typography, top notch hyphenation, complicated page layouts, and PostScript fonts.

TEX

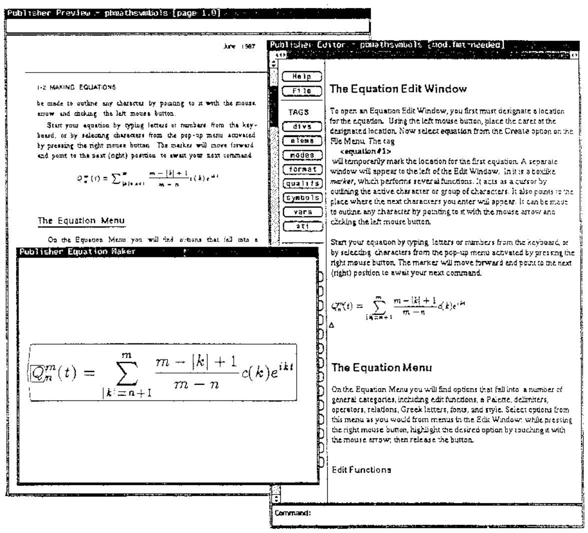
Hidden from most users of *The Publisher*, TEX provides a level of sophistication and device independence in formatting that very few systems can match. *The Publisher* uses a version of TEX that is upwardly compatible with all standard implementations of TEX and produces standard .dvi output files; however, *The Publisher*'s version of TEX allows for interactive page-by-page processing and faster document production.

Editing

The Publisher features an easy-to-use, full-featured, unpaginated WYSIWYG editor. Within the editor, you can choose how the formatting tags are displayed—whether they are off, on, or represented by icons. In the left window of the graphic to the left, all tags marking document structures are suppressed. In the middle window, most tags are replaced by icons. The right window displays all tags. When you want to see the exact representation of the document on-screen, just format it and a second window will show you the fonts and character placement as they will appear on the printed page. You can even edit or change style options while your document is formatting.

The Equation Maker

The main feature of the What-You-See-Is-What-You-Get Equation Maker pictured below is a separate Equation Edit Window that allows you simultaneously to enter symbols and to view your equation. Among other options provided are a pop-up menu from which you can select a wide range of mathematical symbols and functions, and a customizable character palette. Standard keyboard characters can be used in the Equation Maker as well. When you exit from the Equation Maker, the equation appears in the Edit Window.


Publisher Preview - pmathsymbols [page 1.0]
Publisher Equation Maker
Publisher Editor - pmathsymbols [mod.fmt needed]
Help
File
TAGS
divs
elems
modes
format
qualifs
symbols
vars
ati
Command:

June 1987

1-2 MAKING EQUATIONS

be made to outline any character by pointing to it with the mouse arrow and clicking the left mouse button.

Start your equation by typing letters or numbers from the keyboard, or by selecting characters from the pop-up menu activated by pressing the right mouse button. The marker will move forward and point to the next (right) position to await your next command.

$$Q_n^m(t) = \sum_{|k|=n+1}^{m} \frac{m-|k|+1}{m-n} c(k)e^{ikt}$$

The Equation Menu

On the Equation Menu you will find options that fall into a

$$\boxed{Q}_n^m(t) = \sum_{|k|=n+1}^{m} \frac{m-|k|+1}{m-n} c(k)e^{ikt}$$

The Equation Edit Window

To open an Equation Edit Window, you first must designate a location for the equation. Using the left mouse button, place the caret at the designated location. Now select **equation** from the Create option on the File Menu. The tag

<equation#1>

will temporarily mark the location for the first equation. A separate window will appear to the left of the Edit Window. In it is a boxlike *marker*, which performs several functions. It acts as a cursor by outlining the active character or group of characters. It also points to the place where the next characters you enter will appear. It can be made to outline any character by pointing to it with the mouse arrow and clicking the left mouse button.

Start your equation by typing letters or numbers from the keyboard, or by selecting characters from the pop-up menu activated by pressing the right mouse button. The marker will move forward and point to the next (right) position to await your next command.

$$Q_n^m(t) = \sum_{|k|=n+1}^{m} \frac{m-|k|+1}{m-n} c(k)e^{ikt}$$

△

The Equation Menu

On the Equation Menu you will find options that fall into a number of general categories, including edit functions, a Palette, delimiters, operators, relations, Greek letters, fonts, and style. Select options from this menu as you would from menus in the Edit Window: while pressing the right mouse button, highlight the desired option by touching it with the mouse arrow; then release the button.

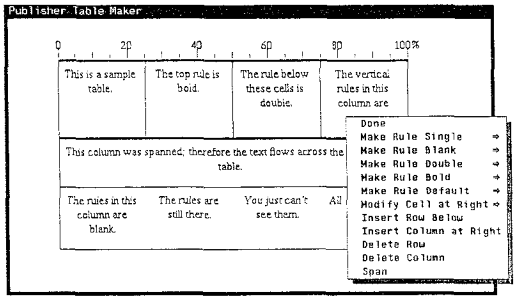
Edit Functions

WHAT YOU WANT.

The Table Maker

The What-You-See-Is-What-You-Get Table Maker allows you to design complex tables for your *Publisher* documents that can be edited and re-edited with the click of a mouse button. Now for the first time, complex tables can be created by everyone without entering a single backslash. *You* design your WYSIWYG table and let *The Publisher* take care of the rest.

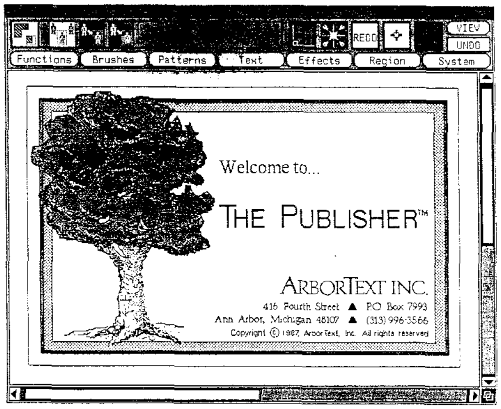
The Table Maker features a separate window in which to choose any number of rows and columns, design the appearance of the rules, and enter and edit text. To widen a cell while creating a table, just hold the left mouse button down with the mouse arrow over a vertical rule and "drag" the rule left or right using the on-screen ruler as a width gauge. The finished table is automatically placed in-line in the *Publisher* Edit Window.



This is a sample table.	The top rule is bold.	The rule below these cells is double.	The vertical rules in this column are
This column was spanned; therefore the text flows across the table.			
The rules in this column are blank.	The rules are still there.	You just can't see them.	All

Graphics

The Publisher features integrated, full function paint and draw programs. You can rotate, shade, merge, and import/export graphics; draw circles, squares, curves, and arcs; magnify images; design your own patterns and brushes; and label your drawings with PostScript fonts. And the list goes on. Also included is Pubgrab, a program that allows you to "grab" any region of the Sun Workstation screen and include it in your document. *The Publisher* can then resize the graphic to your specified magnification.



Welcome to...

THE PUBLISHER™

ARBORTEXT INC.
416 Fourth Street ▲ P.O Box 7993
Ann Arbor, Michigan 48107 ▲ (313) 996-3566
Copyright © 1987, ArborText, Inc. All rights reserved

Special Offer

During the beta period, we are offering *The Publisher* at half off the regular price. This offer ends when we officially release version 1.0 in the fall. If you are interested in *The Publisher*, just call or send in the form below.

TEX is a trademark of the American Mathematical Society. PostScript is a trademark of Adobe Systems, Inc. Sun is a trademark of Sun Microsystems, Inc.

Yes, I am interested in receiving:

□ **a call.** □ **literature.**

Name ____________________

Title ____________________

Company/School ____________________

Address ____________________

City **State** **Zip** ____________________

Country ____________________

Business Phone (**)** ____________________

Computer Hardware ____________________

Printer(s) ____________________

ARBORTEXT INC.
P.O. Box 7993
Ann Arbor, Michigan 48107
(313) 996–3566

easyTEX

easyTEX

easyTEX

easyTEX

easyTEX

easyTEX

easyTEX

$$\sqrt[3]{\frac{\sum_{i=1}^{n} f(x_i)}{\left\{\begin{matrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{1n} & a_{2n} & \cdots & a_{nn} \end{matrix}\right\}}}$$

easyTEX

easyTEX

easyTEX

easyTEX

easyTEX

easyTEX

easyTEX

Why to lose time and to find you have made a mistake?

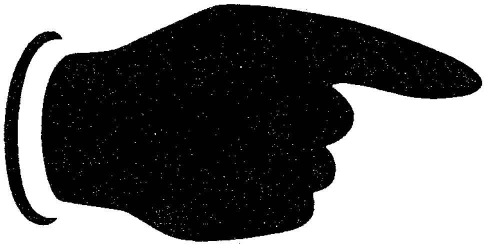
Today TEX can be easier to use

easyTEX Release 1.0.
the interactive Formula Editor running on PC–IBM compatibles
saving formulae typesetting time using TEX.

TE CO GRAF Via Plinio, 11 20129 Milano ITALIA. Tel. 02 208150 – 278063. Telex 340160 Per Tecograf.

Please send me more information about easyTEX.
Name ....
....
Address ....
....

This Page Is Not Copy-Protected



People don't like to tear up their copies of TUGboat. They're much too valuable. But don't let that deter you from ordering our products, including TeXtures v.1.0 and the new CM Drivers for MicroTeX.

So please, take this coupon to your nearest copier and zapf a copy to us. Or give us a call at 617-944-6795.

ADDISON-WESLEY

EMS Division, Reading, MA 01867
Site Licensing, Network Licensing,
Training, Service, and Support.

Addison-Wesley TeX Products Order Form

To order TeXtures,™ MicroTeX, or related products, fill out the following information and return this form. For further information call (617) 944-6795. Send this form to Addison-Wesley, 12 Jacob Way, Reading, MA 01867, Attn: EMSD Sales.

TeXtures™ Professional Typesetting for the Macintosh

□ TeXtures	$495.00
TeXtures Style Masters (Macro packages with pre-defined formats):	
● LaTeX Style Masters (Available summer 1987)	$44.95
□ AMS-TeX Style Masters (Available summer 1987)	$44.95

MicroTeX Professional Typesetting System™ for the IBM PC and Compatibles

□ MicroTeX, Preview and an Epson/IBM dot-matrix printer driver	$469.95
□ MicroTeX, Preview and one laser printer driver below	$595.00
□ PostScript printer driver	
□ HP LaserJet+ printer driver	
□ QMS QUIC printer driver	
□ Imagen Impress printer driver	
Additional drivers may be purchased separately:	
□ PostScript printer driver	$300.00
□ HP LaserJet+ printer driver	$300.00
□ QMS QUIC printer driver	$200.00
□ Imagen IMPRESS printer driver	$200.00
□ IBM/EPSON printer driver	$100.00
□ Preview driver	$250.00
MicroTeX Style Masters (Macro packages with pre-defined formats):	
□ LaTeX Style Masters	$44.95
□ AMS-TeX Style Masters (Available summer 1987)	$44.95

□ MY CHECK OR MONEY ORDER IS ENCLOSED
Addison-Wesley will pay postage and handling (please include local sales tax).

□ CHARGE IT TO MY CREDIT CARD
I'd like to charge my order. I understand I'll be charged for local sales tax, plus shipping and handling.

□ VISA □ MasterCard □ American Express Acct # ____________________

(MasterCard Interbank # ____________________) Exp. Date ____________________

Signature ____________________

□ PURCHASE ORDER ENCLOSED

PLEASE SHIP VIA: □ UPS □ U.S. Mail Phone (______) ____________________

Name ____________________ Title ____________________

Address ____________________

Company ____________________

City ____________________ State ____________________ Zip ____________________

Add-on Packages to Make TEX Work Easier

Polyglot, Inc. has solved those annoying little problems that prevent you from being maximally efficient when using TEX with foreign languages.

Hy-TEX ™- The Multilingual Hyphenator
To-TEX ™- The Multilingual Accentor
JET:SPELL- The Multilingual Spelling Checker
A series of advanced linguistic products from Polyglot

Hy-TEX

Hy-TEX correctly hyphenates files in 13 languages, acting as a filter between the editing stage and the TEX stage. **Hy-TEX** reads your TEX file from standard input, inserts soft hyphens (\-) at the appropriate places in all words longer than 5 characters, and writes the result in standard output. And **Hy-TEX** works completely transparently for the user without modifying your input file. Available for:

American-English	British-English	French	Québécois-French	Italian
German	Spanish	Dutch	Danish	Swedish
Norwegian	Finnish	Portuguese		

To-TEX

To-TEX converts accented characters as well as Greek and Mathematical symbols allowed by the IBM extended character set, into valid TEX control sequences.

JET:SPELL

The only standalone multilingual spelling checker available anywhere, **JET:SPELL** is so fast and powerful that one reviewer thought it wasn't even working! The program uses sophisticated morphological analysis to provide accuracy and completeness of verification far beyond what most spelling checkers can offer. In addition to its *eleven* languages, **JET:SPELL** includes nine user-tailorable dictionaries in which to store names and terminology particular to your work. Available for:

American-English	British-English	French	Québécois-French	Italian
German	Spanish	Dutch	Danish	Swedish
Norwegian				

All Polyglot programs run on any IBM-PC or truly compatible computer and are accompanied by complete user instructions. Hy-TEX needs 256KB of RAM; JET:SPELL needs 384KB of RAM.

ORDER NOW

Hy-TEX (20% discount on orders of five or more languages)	$49.95/language
To-TEX	$29.95
JET:SPELL (Multilingual System disk, one language dictionary)	$109.95
JET:SPELL Dictionaries	$59.95/language

PAYMENT

- VISA, MasterCard, checks; for overseas orders we accept bank drafts in US$ on a US bank.
- Colorado residents add appropriate sales tax.
- All prices include standard shipping to cities within the continental United States. Rush delivery and overseas airmail, add $10.

ADDRESS INQUIRIES AND ORDERS TO:
Polyglot, Inc., 2450 Central Avenue, Suite P4, Boulder, Colorado 80301 USA
Tel: 303/449-7002; Telex: 353069; Telecopier: 303/449-2897; MCI Mail: 266-0370
TEX is a trademark of the American Mathematical Society.
JET:SPELL and Polyglot are registered trademarks of Polyglot, Inc.

TeX Typesetting Services

The American Mathematical Society can produce typeset pages from your DVI or source files. Features of our services include:

- **QUALITY** - We use an Autologic APS Micro-5 typesetter.
- **FONTS** - We offer AM, CM and Times Roman. Several more Autologic typefaces will be added in the near future.
- **LOW-COST** - We charge only $5 per page for the first 100 pages; $2.50 per page for additional pages.
- **SPEED** - Turnaround time is no more than one week for up to a 500 page job.
- **EXPERIENCE** - If you have a problem with a DVI or source file, we can usually solve it with our staff who are trained in TeX.
- **FULL-SERVICE** - We also offer keyboarding, camera work, printing and binding services.

For more information, or to schedule a job, please contact Regina Girouard

American Mathematical Society	(401) 272-9500
PO Box 6248	800-556-7774
Providence, RI 02940	

WANTED

TV GUIDE, the nation's best-selling weekly magazine, is on the front edge of an exciting adventure in the use of TeX in a commerical environment. The suburban Philadelphia-based publishing company is looking for applicants with TeXpertise to create, via TeX, an automated operation supporting the varying page formats and layouts of the magazine. As such, the individual will be involved in:

- *creating TeX macros;*
- building fonts and television listing grids;
- creating a page description enviroment to interface with and support the feature and listings sections of the magazine;
- automation of page layout functions, integrating images and text;
- building interfaces to TeX which will allow its operation in a production environment.

The successful applicant will be someone who:

- has extensive TeXperience, including use of PC TeX;
- has programming experience, perferably in C or PASCAL;
- is familiar with micro computer technology;
- has had exposure to and use of Metafont and other TeX add ons.

A B.S. in computer science, or the equivalent, is required. Salary will be commensurate with experience. Please send resume, including salary requirements to:
Margaret A. Beard, Personnel Administrator, TV GUIDE, Radnor, PA 19088.

TV GUIDE
TRIANGLE PUBLICATIONS INC

metafont
metafont
metafont
metafont
metafont
metafont

NEENIE BILLAWALA Metafont Consulting

(415) 965-0643 2014 Colony #8, Mt View CA 94043

Font News

☞ The Newest in TeX-Compatible Fonts ☜

WE'RE MAKING HEADLINES.

Dublin – Metafoundry fonts are making headlines, invitations, mastheads, newsletters and a myriad of other documents, without giving up any of the power and precision of TeX. First, there are our *complete* families (including math symbols, math italic and extensible) in a crisp sans serif and in a Roman that is lighter and more compact than CMR.

In addition, we are now proud to offer a Decorative package made up of specialty fonts and symbols to add spice to your TeX creations. Outline fonts. Black letter fonts. Even a Copperplate script font.

Finally, we offer a Slavic package of Cyrillic and slavic characters in Roman and sans serif styles compatible with our English fonts.

These fonts are useable with most versions of TeX. All are supported for 300 dpi positive imaging machines.

We think you'll be impressed with the quality and versatility of our fonts. For more information, contact one of these companies who distribute to members of the TeX users group:

ASCII Corporation
Tokyo, Japan

Comp-U-Calc, Inc.
Great Bend, Kansas

Personal TeX, Inc.
Sunnyvale, California

UniTeX Systems
Sheffield, England

To receive copies of our four product catalogues, send a check (payable to OCLC, Inc.) to cover shipping and handling to *The Metafoundry, 6565 Frantz Road, Dublin, Ohio 43017.* Cost is $6.00 if shipping address is in the U.S. or Canada, $15.00 elsewhere.

The Metafoundry™
Digital Typography & Font Design

Index of Advertisers