Computerized Typesetting: TEX on the HP3000

Lance Carnes
Independent Consultant
Mill Valley, California

ABSTRACT

TEX is a program which allows the ordinary user to produce professional quality typeset output. TEX was developed by Donald E. Knuth of Stanford University and is currently used throughout the world for typesetting both technical and non-technical material. This paper will describe the use of TEX and show some examples of its output. The transportable version of TEX, written in PASCAL, has been successfully moved to the HP3000. The second part of the paper describes the tasks involved in this process.

INTRODUCTION

1. What is TEX?

Tau Epsilon Chi (TEX) is a system for typesetting technical books and papers. It can also be used for ordinary non-technical material. The system does not require the user to have a knowledge of typesetting rules or conventions.

The original TEX system was developed at Stanford University by Donald E. Knuth. Frustrated in his attempts to print a second edition of *The Art of Computer Programming* in the same printing style as the first edition, he looked for alternatives in the area of computerized typesetting. Finding nothing that suited him, he embarked on a project which was to become the TEX system. This system is described in detail in his informative and humorous book, *TEX and METAFONT* [Knut79].

The TEX system is currently used throughout the world, partly for technical work in mathematics and physics, and partly for various other uses. The *Journal of the American Mathematical Society* now accepts TEX input files for publication. Some major corporations and universities use it for typesetting their internal documentation, user manuals, newsletters, etcs. The TEX Users Group accepts articles and letters for their Journal in TEX format.

2. How Does It Work?

The TEX program accepts an input file consisting of text and control sequences, and generates a device independent output file (DVI file) which contains commands for driving a raster printer device. Once TEX has processed the input and produced a DVI file, it is up to a device driver program to interpret the commands in the DVI file and produce printed output. This sequence of events is shown in Figure 1.

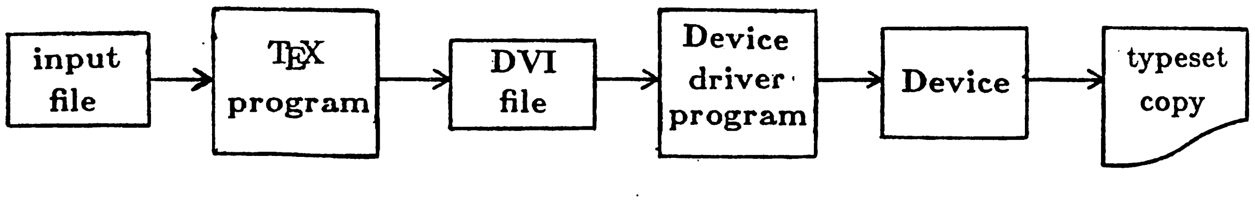

Figure 1. Functional Diagram of the TEX System

Most of the typesetting is done by TEX automatically. TEX operates on many levels, composing pages, paragraphs lines and words. All of these are interrelated, with the intention of producing professional quality printing. In cases where TEX needs to be guided, for example in printing the TEX logo, the user intervenes by specifying a control sequence (see 3 below).

The TEX system does not typeset a single word or a single line at a time. Rather, it typesets a page or more at a time. This is done for a variety of reasons. Mainly, we want the printed page to consist of pleasantly spaced paragraphs, lines and words. Also we want to avoid other unwanted phenomena, such as "widow" lines. A widow line is the first line of a paragraph appearing at the bottom of a page with the paragraph continuing on the next page. To eliminate widows, TEX returns to the paragraphs already layed out and expands them slightly so as to use one more line on the page. This forces the widow line to the top of the next page.

Paragraphs are composed to reduce the number of hyphenations and so as not to leave a single word stranded in the last line. In addition, the spacing between words is equalized throughout the paragraph.

Lines of text are composed of words and other symbols (e.g., mathematical formulas) with the space between words equalized.

Words are typeset with the letters placed one character width apart. Unlike standard computer printers which print all characters in the same width (usually 1/10 inch), typesetting separates characters by the exact width of the character, depending on the "font" or character style used. In addition, TEX will place characters closer together or farther apart in accordance with traditional typesetting rules. For example, when typesetting the word "AVIATOR" the "A" and "V" are placed closer together; this is called "kerning." Notice in the word "find" that the "f" and "i" are pushed together to form the "ligature" fi. These typesetting conventions and more are known to TEX, freeing the user from having to memorize them.

The basic concepts TEX uses are "boxes" and "glue." A box contains something which is to be printed, and glue specifies the spacing between boxes. For example, a character is a box, a word is a collection of character boxes, a line is a group of word boxes, a paragraph is a collection of line boxes, and a page is a box composed of paragraph boxes. The space between boxes can expand or contract by carefully defined amounts, called the stretchability or shrinkability of the glue. For example, when TEX composes a paragraph that has a hyphenation it tries to back up and redistribute the spacing of the words in the paragraph to avoid the hyphenation. It does this by increasing or shrinking the space or glue between the boxes by allowable amounts.

For further details on the inner workings of TEX, see [Knut79] or [Spiv80].

3. Submitting an Input File to TEX

The input file for TEX is edited using any text editor. The text and any control sequences are contained in this file. When TEX is run, this file is designated as the input file.

Basically, text is entered in a standard fashion with spaces between words, and one blank line between paragraphs. The input need not be formatted in any particular manner beyond this. Control sequences are defined as a " " followed by a word or symbol. They allow the user to specify a special command. For example, "\it IMPORTANT" would cause the world IMPORTANT to be set in italic font.

The TEX system can be run in either interactive or batch mode. In interactive mode, if TEX finds an error the user is allowed to make modifications on the fly. For example,

```
!Undefined control sequence
\iy
    IMPORTANT
```

The TEX program is indicating that it does not know the control sequence "\iy" and shows what it has scanned on the first line, and what it has not yet scanned on the following line. At this point the user may correct the input by typing "1" to erase one symbol or control sequence, and then "I" to insert the correct sequence "\it". Any corrections made in this manner are recorded in an errors file for future reference.

As TEX is processing the input, it is writing to the DVI file. After the input is successfully processed, the DVI file is ready for the device driver program.

Two other important facilities are available with TEX. These are alternate input files and macro definitions. Alternate input files are TEX input files which are read in conjunction with another input file. For example, if a paper has an abstract and three sections, and each is in a separate file, a main file would draw them all together as follows:

```
% paper on TEX for the HP3000
\input basic % basic control sequences
\input texabs % abstract file
\input sect1
\input sect2
\input sect3
\end
```

Each of the alternate input files could have had \input commands also. The maximum nesting depth is nine.

Macro definitions allow the user to specify a common sequence by defining it and giving it a name. For example, the logo TEX was specified by inserting "\TEX". \TEX was previously defined as

```
\def TEX{\hbox{lowercase{\:a
\uppercase{T} hskip-2pt\lower1.94pt
\hbox{\uppercase{E}}\hskip-2pt \uppercase{X}}}}
```

It is much easier to write "\TEX" than to insert the above expansion.

4. Fonts

A font is a specific design of an alphabet and associated symbols. Most typewrites have Pica or Elite type fonts. The different "balls" or "daisy wheels" on some printers allow the user to change fonts.

The TEX system allows up to 64 different fonts to be specified within the same job. A control sequence is given to switch from one to the other. Naturally you must have a device which can support all of these different fonts.

Knuth also wanted to define his own fonts and created a system called METAFONT to do this. Using METAFONT one can design a font which is coded into a file for use by TEX. For more information on METAFONT see [Knut79].

5. The DVI File

The DVI file consists of a series of 8-bit codes which tells a device driver how to typeset the job. The format of the DVI files is given in Appendix B.

Basically, a DVI file command is of the form "set the letter d and advance the character width" or "change to font 3" or "advance vertically 12 rsu's". No inherent intelligence on the part of the device is assumed. In fact, TEX gets along best with devices which have no internal programming, such as proportional spacing or typesetting firmware.

6. Device drivers.

The assumed printer is a raster scan printing device. This implies that all spacing between characters and lines is user specified. A typical computer line printer is not a raster device since it will always print 10 character/inch, and six lines/inch (or some variation of this). Most of the daisy wheel terminals available now can be used as raster devices. The actual device TEX is aimed at is a commercial computer-driven typesetting device, such as a Xerox Graphics Printer, a Mergenthaler Linotron 202, or an HP2680 Laser Printer.

The TEX program has no knowledge of any particular printing device. It creates the same DVI file regardless of the output device. It is the job of the Device Driver program to interpret the DVI commands and produce output on a specific device. While there is only one TEX program, there will be one Device Driver program for each output device.

TEX ON THE HP3000

1. TEX in PASCAL

The TEX system was originally written in a language called SAIL (Stanford University Artificial Intelligence Language). The SAIL compiler and the original TEX system ran on the DEC-20 computer only. TEX is in the public domain, but was not even remotely transportable. Due to the popularity of the system, a project was undertaken to translate the TEX system into a computer language which was available on most modern computer systems.

The language chosen for the transportable system was PASCAL. The method for translating the system was as follows. First, a well documented pseudo-PASCAL source was developed. This source has only a slight resemblance to a PASCAL program and was intended to serve mostly as documentation, and to give all the algorithms. This file is often referred to as the DOC file.

The second step was to produce syntactically correct PASCAL source code from the DOC file. There is a program called UNDOC which performs this step. The resulting PASCAL source is distributed to anyone wanting to transport TEX to another computer.

The DOC file is actually typeset, and a photocopy is provided with the distribution tape. The PASCAL source is almost unreadable, but will compile. Examples of both of these files is in Appendix C.

2. Moving TEX to the HP3000

The Stanford TEX-in-PASCAL project brought the system to a point where it could be transported to other computer systems. The transportation process, however, requires a good deal of time and a patient systems programmer.

At the time of writing, this author has successfully transported TEX to the HP3000. The project was by no means trivial, as will be shown.

Bringing TEX to the HP3000 had a lot of problems right from the outset. First, there was no supported PASCAL compiler at the time this project was begun. Second, the design of the TEX program assumes a large address space, something on the order of 600K words of addressable memory.

The tasks broke down as follows:

a. Edit the PASCAL sources. While the system was translated to a "Standard" PASCAL, there are still many variations and assumed extensions which had to be accounted for. With 23,000 lines of PASCAL source this took considerable time and effort.

b. Rewrite the "System dependent" routines. These are the procedures and functions which interface TEX with the file system, terminal I/O and other traits particular to the host system. About 25 routines had to be modified or rewritten.

c. Implement a virtual memory scheme. TEX references several large arrays throughout, some as large as 50,000 elements with 4 32-bit words per element. An addressing scheme was developed to allow the array contents to reside in secondary storage.

d. Revise the PASCAL compiler to allow 32-bit integers and to compile large array references. TEX assumes 32-bit integers throughout, and the Portable P4 compiler from the HP Users Contributed Library was modified to allow them.

e. Optimize the performance of the system. When the above tasks were completed and the system first ran on the HP3000, it was incredibly slow. Where the original TEX system at Stanford processed a document in less than two minutes, the initial HP3000 TEX took 40 minutes. By analyzing TEX's operation, some optimizations have been made reducing the run time to about 6 minutes. Additional optimizations will be made to allow the system to run as fast as possible. One tool which has been particularly useful for identifying inefficient code is APG/3000 from Wick Hill Associates.

3. Device drivers

A device driver for a daisy wheel printer has been developed for use on the HP3000. While only one font is available at a time with this device, satisfactory results have been obtained. The output is suitable for internal documentation, and for proofing a document. Future plans are to develop a driver for the HP2680 Laser printer.

However, it is not necessary to have a high quality printing device on-site. There is one commercial printing house in San Francisco which uses TEX for typesetting on a Mergenthaler Linotron 202; the output from this device is camera ready. DVI files produced by TEX on the HP3000, once proofed on the daisy wheel printer, will be sent to this commercial printer.

CONCLUSIONS

This is a truly remarkable system. It gives the ordinary person the ability to print professional quality copy. The user will not have to explain to a typographer what is wanted, but will have personal control.

The HP3000 implementation of TEX will be a boon for any organization desiring to improve the quality of documentation, user manuals and other printed materials. Good results can be obtained with an inexpensive daisy wheel printer. Where camera-ready copy is desired, several higher quality devices are commercially available.

Hopefully more organizations will begin to use TEX for documentation, manuals, annual reports and newsletters. Perhaps one day soon the HP General Systems Users Group will accept papers for publication in TEX format.

ACKNOWLEDGEMENT

My thanks to Prof. Luis Trabb-Pardo and Charles Restivo of Stanford University for their assistance in learning the TEX system; and to GENTRY, INC. of Oakland, California, for providing time on the HP3000.

REFERENCES

[Knut79] Donald E. Knuth, *TEX and METAFONT*. New Directions in Typesetting. Digital Press, 1979.
This is a beautifully printed book, an acknowledgement of the TEX system. Don Knuth's writing style is at once brilliant and witty. It contains a User's Guide to the TEX and METAFONT systems and a paper on Mathematical Typography.

[Spiv80] Michael Spivak, *The Joy of TEX*. A Gourmet Guide to Typesetting Technical Text by Computer. Verson -1. American Mathematical Society, 1980.
This is a real book. It gives a lighthearted introduction to the use of AMS-TEX, the version of TEX used by the AMS.

TUGboat, The TEX Users Group Newsletter. Published by the American Mathematical Society.
The TEX Users Group is small currently, but enthusiastic and helpful. For information on membership write to:

TEX Users Group
c/o American Mathematical Society
P.O. Box 6248
Providence, Rhode Island 02940

```
\noindent {\bf ABSTRACT:} \TEX\ is a program which allows
the ordinary user to produce professional quality
typeset output.
\TEX\ was developed by Donald E. Knuth of Stanford
University and is currently used throughout the world
for typesetting both technical and non-technical material.
This paper will describe the use of \TEX\ and show
some examples of its output.
The transportable version of TEX, written in Pascal,
has been successfully moved to the HP3000.
The second part of the paper describes the tasks involved
in this process.

\vskip 0.4 cm
\noindent {\bf I. INTRODUCTION}

\vskip 0.3 cm
\noindent {\bf 1. What is \TEX\ ?}

\vskip 0.1 cm
{\it Tau Epsilon Chi} (\TEX\ ) is a system for typesetting
technical books an dpapers.
It can also be used for ordinary non-technical material.
The system does not require the user to have a knowledge of
typesetting rules or conventions.

The original \TEX\ system was developed at Stanford University
by Donald E. Knuth.
Frustrated in his attempts to print a second edition of
{\it The Art of Computer Programming} in the same printing
style as the first edition, he looked for alternatives in the
area of computerized typesetting.
Finding nothing that suited him, he embarked on a project
```

ABSTRACT: TEX is a program which allows the ordinary user to produce professional quality typeset output. TEX was developed by Donald E. Knuth of Stanford University and is currently used throughout the world for typesetting both technical and non-technical material. This paper will describe the use of TEX and show some examples of its output. The transportable version of TEX, written in Pascal, has been successfully moved to the HP3000. The second part of the paper describes the tasks involved in this process.

I. INTRODUCTION

1. What is TEX ?

Tau Epsilon Chi (TEX) is a system for typesetting technical books and papers. It can also be used for ordinary non-technical material. The system does not require the user to have a knowledge of typesetting rules or conventions.

The original TEX system was developed at Stanford University by Donald E. Knuth. Frustrated in his attempts to print a second edition of *The Art of Computer Programming* in the same printing style as the first edition, he looked for alternatives in the area of computerized typesetting. Finding nothing that suited him, he embarked on a project

APPENDIX A
A PORTION OF THE TEX INPUT FILE

Command Name	Command Bytes Description
VERTCHAR0	0 Set character number 0 from the current font such that its reference point is at the current position on the page, and then increment horizontal coordinate by the character's width.
VERTCHAR1	1 Set character number 1, etc.
⋮	⋮
VERTCHAR127	127 Set character number 127, etc.
NOP	128 No-op, do nothing, ignore. Note that NOPs come *between* commands, they may not come between a command and its parameters, or between two parameters.
BOP	129 c0[4] c1[4] ... c9[4] p[4] Beginning of page. The parameter p is a pointer to the BOP command of the *previous* page in the .DVI file (where the *first* BOP in a .DVI file has a p of —1, by convention). The ten c's hold the values of TEX's ten \counters at the time this page was output.
EOP	130 The end of all commands for the page has been reached. The number of PUSH commands on this page should equal the number of POPs.
PUSH	132 Push the current values of horizontal coordinate and vertical coordinate, and the current w-, x-, y-, and z-amounts onto the stack, but don't alter them (so an X0 after a PUSH will get to the same spot that it would have had it had been given just before the PUSH).
POP	133 Pop the z-, y-, x-, and w-amounts, and vertical coordinate and horizontal coordinate off the stack. At no point in a .DVI file will there have been more POPs than PUSHes.
HORZRULE	135 h[4] w[4] Typeset a rule of height h and width w, with its bottom left corner at the current position on the page. If either h ≤ 0 or w ≤ 0, no rule should be set.

VERTRULE	134 h[4] w[4] Same as HORZRULE, but also increment horizontal coordinate by w when done (even if $h \leq 0$ or $w \leq 0$).
HORZCHAR	136 c[1] Set character c just as if we'd gotten the VERTCHARc command, but don't change the current position on the page. Note that c must be in the range [0..127].
FONT	137 f[4] Set current font to f. Note that this command is not currently used by TEX—it is only needed if f is greater than 63, because of the FONTNUM commands below. Large font numbers are intended for use with oriental alphabets and for (possibly large) illustrations that are to appear in a document; the maximum legal number is $2^{32} - 2$.
X2	144 m[2] Move right m rsu's by adding m to horizontal coordinate, and put m into x-amount. Note that m is in 2's complement, so this could actually be a move to the left.
X3	143 m[3] Same as X2 (but has a 3 byte long m parameter).
X4	142 m[4] Same as X2 (but has a 4 byte long m parameter).
X0	145 Move right x-amount (which can be negative, etc).
W2	140 m[2] The same as the X2 command (i.e., alters horizontal coordinate), but alter w-amount rather than x-amount, so that doing a W0 command can have different results than doing an X0 command.
W3	139 m[3] As above.
W4	138 m[4] As above.
W0	141 Move right w-amount.
Y2	148 n[2] Same idea, but now it's "down" rather than "right", so vertical coordinate changes, as does y-amount.

Y3	147 n[3]	As above.
Y4	146 n[4]	As above.
Y0	149	Guess.
Z2	152 m[2]	Another downer. Affects vertical coordinate and z-amount.
Z3	151 m[3]	
Z4	150 m[4]	
Z0	153	Guess again.
FONTNUM0	154	Set current font to 0.
FONTNUM1	155	Set current font to 1.
⋮	⋮	
FONTNUM63	217	Set current font to 63.

36. The procedure *Print* takes an integer as argument and prints the corresponding *strngpool* entry both in the terminal and in the errors file.

```
procedure Print(mes : integer);
  var i : integer; { index in the string }
    c : asciiCode;
  begin i := strng[mes]; c := strngpool[i];
  while c <> null do
    begin terOut↑ := chr(c); errfil↑ := chr(c); put(terOut); put(errfil);
    Increment(i); c := strngpool[i]
    end;
  end;
procedure PrintLn(mes : integer);
        { Like Print, but beginning at a new line. }
  begin terOut↑ := chr(carriagereturn); errfil↑ := terOut↑; put(terOut);
  put(errfil); terOut↑ := chr(linefeed); errfil↑ := terOut↑; put(terOut);
  put(errfil); Print(mes)
  end;
```

```
1224 PROCEDURE PRINT(MES: INTEGER);
1224 VAR I: INTEGER;
   3 S: ASCIICODE;
   5 B
1224 I:=
1233 C:=ST
1247 BEGIN
1247 TEROUT^:= HR( );
1251 ER FIL^:= HR( );
1255 PUT(T P UT);
1257 PUT (ERPFIL);
1259 I:=I+1;
1263 C:=STRNGPOOL[1]
1270 END;
1274 END;
1275 PROCEDURE PRINTLN(MES: INTEGER);
1275 BEGIN
1275 PRINT(MES)
1279 ;WRITELN(TEROUT);
1282 WRITELN(ERRFIL);
1284 END;
```

APPENDIX C
FRAGMENTS OF TEX DOC AND PASCAL SOURCE

25. Find the equation of the plane passing through the end points of the three vectors $\mathbf{A} = 3\mathbf{i} - \mathbf{j} + \mathbf{k}$, $\mathbf{B} = \mathbf{i} + 2\mathbf{j} - \mathbf{k}$, and $\mathbf{C} = \mathbf{i} + \mathbf{j} + \mathbf{k}$, supposed to be drawn from the origin.

26. Show that the plane through the three points (x_1, y_1, z_1), (x_2, y_2, z_2), and (x_3, y_3, z_3) is given by

$$\begin{vmatrix} x_1 - x & y_1 - y & z_1 - z \\ x_2 - x & y_2 - y & z_2 - z \\ x_3 - x & y_3 - y & z_3 - z \end{vmatrix} = 0.$$

Solution. Let $w = f(x, y, z) = x^2 + y^2 - z$, so that the equation of the surface has the form

$$f(x, y, z) = \text{constant},$$