TEX-PASCAL has been distributed to more centers: Stanford Linear Accelerator Center, University of Aarhus in Denmark, Universities of Milan and Pisa in Italy, University of Valencia in Spain, etc. The ones mentioned above have given the most feedback.

At this time, it seems that most pioneer installations are free of compilation problems and are now trying to obtain adequately interfaced output devices, together with fonts and font information files suitable for them.

The only fully operational PTEX system is still this at Stanford CSD, but we expect to be printing DVI files produced by the CIT installation very soon.

Compiler issues:

TEX-PASCAL was developed using the Hamburg PASCAL compiler for the PDP-10 by Kisicki and Nagel. Some compiler maintenance was needed during the debugging of PTEX. We have found this to be a rather powerful and permissive compiler.

There have only been three system requirements on PTEX hosts and these were explicit since the beginning of the project:

- The system must have enough addressable memory to store the large arrays employed by PTEX (about 128K words of 32 bits).
- The compiler should be able to really pack fields of a PACKED RECORD and overlap multiple variants of packed records. If this requisite is not satisfied, PTEX will require at least four times as much memory.
- The compiler should be able to handle large case statements (say over 64 actual cases in a range [—500..500]) and have a default case (this is non-standard in PASCAL but available in most compilers).

Additionally, PTEX requires an EXTERNAL (or separate) compilation facility. If no such thing is available, the SYSDEP module has to be inserted both in TEX and in TEXPRE by hand. Also, if there is no compile time variable initialization, the INITPROCEDURE appearing in the program has to be changed into an ordinary procedure.

We have fought not to add more requirements and have changed the program to facilitate the installation with simpler or more restrictive compilers. Encountered problems have been common to most pioneer installations:

- lines of code were too long
- octal constants were not accepted
- identifiers containing the underscore character were not accepted
- some identifiers were too long
- sometimes two different identifiers were equal in the first eight characters
- fields of packed records could not be passed as procedure arguments
- loop counters had to be local variables
- all declared labels had to be used
- use of GOTOs was restricted: not even allowed from the body of a procedure out to the block in which the procedure was declared
- there were discrepancies in the treatment of nested WITH statements
- the compiler lacked the standard MAX and MIN functions
- procedures had to be kept small (less than 400 statements)

The program has been modified to avoid them. Currently, the code is all uppercase in lines that are never longer than 72 characters. All identifiers are shorter than 16 characters and differ in the first 8 characters. Octal variables appear only in SYSDEP.

Distribution:

Currently, TEX-PASCAL can be obtained from the TEX group at the CS Dept. at Stanford. Anyone asking for the system will get a tape containing the files TEX.PAS, TEXPRE.PAS, SYSDEP.PAS, TEX.STR, TEXPRE.STR, and SYSDEP.STR, which is about everything that is needed to have PTEX running. The distribution package also contains a short installation guide, a description of the DVI format of the output file of TEX, and extensively documented listings of TEX, TEXPRE and SYSDEP. (Of course, the ultimate documentation on TEX is the TEX manual.) All these files (not the listings) are available on-line in the directory (TEX.PASCAL)%SCORE, accessible via the ARPANET.

Fonts and font information files may also be provided on request (in the format employed here at SAIL). These files are very system-and-output-device-dependent and of restricted general value for that reason.

* * * * * * * * * *

THE FORMAT OF TEX'S DVI FILES

David Fuchs

DVI files contain information about where characters go on pages. The format is such that there are those who claim that almost any reasonable device can be driven by a program that takes DVI files as input. In particular, DVI files can be sent to the Xerox Graphics Printer (XGP), Versatec, Canon or

Alphatype at the Stanford CS Dept., depending on what spooler it is passed to. The format follows.

The basic unit of information in a DVI file comes in an 8-bit chunk. Here at Stanford, they are packed four per word, in the lower-order 32 bits of each word, and the highest-order chunk is considered to be before the others, etc.

The DVI file contains a number of Pages followed by a Postamble. Each Page starts with a BOP command, has lots of other commands, and ends with an EOP command. Each EOP command is immediately followed by another BOP command, or the PST command, which means that there are no more Pages in the file, and the Postamble follows. See below for details on all the commands that occur in Pages, and what goes in the Postamble.

Each Page consists of a number of Commands that specify what characters should be typeset where. Who- or what-ever reads these Pages should have a Stack that can hold, say, 200 coordinates (i.e. integers) to be on the (very) safe side.

There is a notion of the "current position on the page", which is specified by its horizontal and vertical coordinates. Moving rightwards on a page is represented by an increase in the H-coordinate, while moving down is an increase in V, and the upper-left-hand corner of the page is 0,0 (i.e. it's slightly non-cartesian). Coordinates are given in *rsu*'s (ridiculously small units), where $1 rsu = 1/2^{16}$ *points*. This is so that accumulated errors will be undetectable even in the worst imaginable case (a "box" many feet long). Whenever a character or rule is set, it gets put at the current position on the page. The current position on the page is changed by explicit move commands (their names begin with W, X, Y, and Z). It can also change as a side effect of setting a character or rule (the 0–127 and VERTRULE commands). The w-, x-, y-, and z-amounts are not locations, but distances (in *rsu*'s). Some commands change their values, and some cause the current H- or V-coordinate to be incremented by one of their current values.

A lower-case character with a bracketed number following a command means that the command has a parameter that is that many bytes long. Thus, the BOP command, for instance, is 9 bytes long, the first byte of which has the decimal value 129, the second through fifth of which give the page number (high order byte first), and the sixth through ninth being another number which is explained below. These numbers are in two's complement, so they should be sign-extended on the left when they are read.

The commands are:

Command	Description
0 to 127	Set the appropriate character from the current font such that its reference point is at the current H,V location, and then increment the current H-coordinate by the character's width.
128 NOP	No-op, do nothing, ignore.
129 BOP n<4> p<4>	Beginning of page n, with pointer p to the BOP command of the *previous* page. By "pointer" is meant the relative byte number within the DVI file, where the first byte (the BOP of the first page) is byte number zero. (ex.: If the first page had only a BOP and EOP, the *third* page's pointer would be 9, because the BOP command takes bytes 0 to 7, the EOP is 8, so the *second* page's BOP is in byte 9. Get it?). The *first* page has a —1 for a pointer; the second, a zero. Start the H- and V-coordinates out at 0, as well as the w-, x-, y-, and z-amounts. The stack should be empty, and no characters will be set before a FONT(NUM) command occurs. Remember that n can be < 0, if the page was Roman Numbered. Also the pages need not come in the proper order in the file, depending on who's doing the TEXing.
130 EOP	The end of all commands for the page has been reached. The next page, or the postamble, starts in the next byte.
131 PST	The postamble starts here. See below for the full explanation of what goes in the postamble.
132 PUSH	Push the current values of the H- and V-coordinates, 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, had it been given just before the PUSH).
133 POP	Pop the z-, y-, x-, and w-amounts, and the V- and H-coordinates off the stack.
134 VERTRULE h<4> w<4>	Same as HORZRULE, but also increment the current H-coordinate by w when done (even if $h \leq 0$ or $w \leq 0$).
135 HORZRULE h<4> w<4>	Typeset a rule of height h and width w, with its bottom left corner at the current H,V position. If $h \leq 0$ or $w \leq 0$, no rule should be set.

Command	Description
136 HORZCHAR c<1>	Set character c just as above, but don't change the current value of the H-coordinate (or V-coordinate, either).
137 FONT f<4>	From now on, set characters from font number f. Note that this command is not currently used by TEX—it is only needed if f is greater than 63. See FONTNUM commands below.
144 X2 m<2>	Move right m *rsu*'s by adding m to the H-coordinate, and put m into the current x-amount. Note that m is in 2s complement, so this could actually be a move to the left.
143 X3 m<3>	As above.
142 X4 m<4>	As above.
145 X0	Move right the current x-amount (which can be negative, etc).
140 W2 m<2>	The same as the X commands (i.e. alters H-coordinate), but alter w-amount rather than x-amount, so that doing a W0 command can have different results than doing an X0 command.
139 W3 m<3>	As above.
138 W4 m<4>	As above.
141 W0	Move right the current w-amount.
148 Y2 n<2>	Same idea, but now it's "down" rather than "right", so the V-coordinate changes, as does the y-amount.
147 Y3 n<3>	As above.
146 Y4 n<4>	As above.
149 Y0	Guess.
152 Z2 m<2>	Another downer. Affects the V-coordinate and z-amount.
151 Z3 m<3>	
150 Z4 m<4>	
153 Z0	Guess again.
154 to 217 FONTNUM's	Make $0, 1, \ldots, 63$ the current font.
218 to 255 are currently undefined	and will not be output by TEX.

Pages need not be sequential by number, but any blank or non-existent page might not be represented, so page —5's pointer to the "previous page" might point to page 34, for instance (remember that TEX uses negative numbers for roman-numbered pages). The first page in the file has a "previous page" pointer of —1.

The postamble begins with a PST command, followed by four bytes of previous-page pointer to the last real page, followed by four bytes of the height of the tallest page (in *rsu*'s), followed by four bytes of the width of the widest. Next come some Font Definitions (maybe none, if you're an authoritarian), each of which has a Font ID in the first 4 bytes, followed by 4 bytes of Font Number, followed by any character not in the font name, followed by the Font Name, one character per byte for as many bytes as necessary, followed by that same character that was not in the Font Name (a quote is probably a good choice for such a character). The end of the font definitions is marked by an ID of —1 (which will not be followed by font number, etc). The four bytes following this phony ID are a pointer to the PST command (i.e. the begining of the postamble), which is followed by a zero byte, which is followed by at least 4 bytes containing the number 223_{10} (which is '337 octal). The reason for some of the above weirdness is twofold: We are producing DVI files with a Pascal program, and to avoid doing any non-serial I/O, the postamble pointer has to go at the end of the file. Of course, most programs that read these files need not be generally transportable, and can do a random seek to the end of the file, and then another to get right to the postamble. The fact that page-pointers point backwards is in the same spirit, but this also allows the file to be read in backwards-page-order efficiently. This, in turn, will allow for further efficiencies in communicating with your device, depending on how clever it (and you) is (are).

Stanford University July 10, 1980.

* * * * * * * * * * *

UNIVERSITY OF MINNESOTA CDC SITE REPORT

Thea Hodge

We have succeeded in compiling TEX-in-PASCAL on our Cyber 172 but cannot yet run it. TEXPRE, which should generate the required table file, has some problem relative to our system. Michael Frisch, our manager of user libraries and graphics software, is working on that. We are awaiting