[3] Pehong Chen. Gnu emacs BIBTeX-*mode*. Technical Report 87/317, Computer Science Division, University of California, Berkeley, California, 1986.

[4] Pehong Chen. Gnu emacs *TeX-mode*. Technical Report 87/316, Computer Science Division, University of California, Berkeley, California, 1986.

[5] Pehong Chen, J. Coker, Michael A. Harrison, J. McCarrell, and S. Procter. An Improved User Environment for TeX. In J. Desarménien, editor, *TeX for Scientific Documentation, Proc. 2nd European Conf. Strasbourg, France. June 19–21*, 1986. Lecture Notes in Computer Science, No. 236, pp. 32–44 (1986).

[6] Pehong Chen, John L. Coker, Michael A. Harrison, Jeffrey W. McCarrell, and Steven J. Procter. The VORTEX document preparation environment. In J. Desarménien, editor, *TeX for Scientific Documentation, Proc. 2nd European Conf. Strasbourg, France. June 19–21*, 1986. Lecture Notes in Computer Science, No. 236, pp. 45–54 (1986).

[7] Pehong Chen and Michael A. Harrison. Multiple representation document development. *IEEE Computer*, 21(1):15–31, January 1987.

[8] Pehong Chen and Michael A. Harrison. Index preparation and processing. *Software, Practice and Experience*, 18(9):897–915, September 1988.

[9] Pehong Chen, Michael A. Harrison, and Ikuo Minakata. Incremental document processing. In *Proceedings of the ACM Conference on Document Processing*, New York, December 1988. Association for Computing Machinery.

[10] Donald E. Knuth. A torture test for TeX. Technical Report STAN-CS-84-1027, Computer Science Department, Stanford University, Stanford, California, November 1984.

[11] Ethan V. Munson. Enhancements to Bibliography Management. Master's thesis, Computer Science Div., University of California at Berkeley, 1988.

[12] Robert W. Scheifler and Jim Gettys. The X window system. *ACM Transactions on Graphics*, 5(2):79–109, April 1986.

[13] Typographics, Ltd., Jerusalem. *TYPO Font Editing System, Reference Manual*, 1988.

An Enhanced TeX-Editor Interface for VMS

Adrian F. Clark
Essex University

The author described various enhancements to the Stanford distribution of TeX under VAX/VMS in a couple of previous articles. The most significant of these was to allow an editor to be invoked (by typing '`e`' or '`E`') when TeX spotted an error in its input file. The editor which was interfaced to TeX at that time was `TPU`. Since then, a number of requests have been made for interfaces to other editors, usually `EDT`. The author is pleased to announce that this has now been done, in a way which is fully compatible with the previous editor interface. This note discusses the interfaces to the editors usually encountered under VMS and shows how other editors can also be used.

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

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

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

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

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

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

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

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

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

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

The Virtual Memory Management of PubliC TEX

Klaus Thull

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

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

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

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

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

The entire sources are publicly available at LISTSERV@DHDURZ1.BITNET.

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

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