Dreamboat

$\mathcal{N}\mathcal{T}\mathcal{S}$ & ε-TEX: a status report

Philip Taylor

The $\mathcal{N}\mathcal{T}\mathcal{S}$ project project was created at the instigation of Joachim Lammarsch and under the aegis of DANTE e.V. at a DANTE meeting in Hamburg in 1992. The idea of the project was — and is — to perpetuate all that is best in TEX while being free of the constraints which Knuth has placed on the evolution of TEX itself. For that reason, the project was called the $\mathcal{N}\mathcal{T}\mathcal{S}$ project — $\mathcal{N}\mathcal{T}\mathcal{S}$ being short for New Typesetting System — to emphasise that we are not violating Knuth's wishes that TEX remain entirely his responsibility: indeed, we have received Knuth's blessing to pursue the ideas of $\mathcal{N}\mathcal{T}\mathcal{S}$ and ε-TEX, and he went so far as to make some suggestions which, as he put it, "he might have incorporated himself were it not for the fact that he had decided to freeze the evolution of TEX". We are, of course, endeavouring to incorporate those suggestions, although we have not yet fully succeeded.

Before getting into too much technical detail, I should like to explain why the group is called the "$\mathcal{N}\mathcal{T}\mathcal{S}$ group" yet the project on which this report is centered is called "ε-TEX". During the group's early deliberations, we took advice from Joachim Schrod who proposed that rather than attempt to modify TEX-in-Web, we first re-implement TEX using a modern rapid-prototyping language such as LISP, CLOS or PROLOG. Joachim's philosophy was that TEX-in-Web is essentially too poorly structured to allow for radical change, and a more highly structured implementation, with a well-specified interface between modules, was essential if we were to make more than cosmetic changes to TEX-the-system.

The group agreed that this philosophy was sound, but realised that the effort needed to re-implement TEX from scratch was greater than could be achieved with voluntary labour: if success was to be achieved within a sensible timescale, it was essential that the group be in a position to employ a programmer or small team of programmers who could undertake the re-implementation. As the group had no budget of its own, and as it was financially dependent on the goodwill of DANTE, it was reluctantly agreed that this re-implemention would have to be put on ice until such time as the group had adequate financial resources: in the meantime, the group agreed to pursue a rather more conservative approach, evolutionary rather than revolutionary, and to put all of their efforts into this less radical project.

I am delighted to be able to report that, whilst this paper was being written, the future of the $\mathcal{NTS}$ project became very much more certain: during the February meeting of DANTE in München, the Board and members of DANTE agreed to donate DM 30 000 to the $\mathcal{NTS}$ project, which we believe should enable us to employ a programmer for one year, full time, in the Czech Republic, working on the $\mathcal{NTS}$ project under the direct supervision of Professor Jiří Zlatuška (Dean of the Faculty of Informatics at Masaryk University in Brno).

Despite this very significant change in the status of the $\mathcal{NTS}$ project, the group have not abandoned or even reduced their efforts concerning ε-TEX, and we have agreed a tentative specification for ε-TEX V2 which we will be discussing and endeavouring to implement during the coming months. It is our intention to release ε-TEX V2 one year after the release of ε-TEX V1: that is to say, during the first two weeks of November 1997.

To clarify the distinction, "$\mathcal{NTS}$" refers to a future project to completely re-implement TEX in a modular fashion using a modern rapid-prototyping language, and to investigate each module in turn to see how it might be improved — examples of possible modules include the user interface, the user programming language, the typesetting engine itself, and/or components of the typesetting engine such as the line- and page-breaking algorithms.

"ε-TEX", on the other hand, refers to work-in-progress which seeks to develop the TEX-in-Web implementation in a way which is, and will remain, 100% compatible with TEX itself. The "ε" of "ε-TEX" can be thought of as representing "evolution", "extension", "enhancement" (and perhaps even "European"!); the varepsilon of its typeset form emphasises that it is just a small evolutionary step from TEX itself, not a fundamental paradigm shift.

So, work on $\mathcal{NTS}$ will be started in the near future, but ε-TEX is now! After approximately three years in development and testing, we released ε-TEX at the Hamburg meeting of DANTE towards the end of 1996. A few cosmetic changes were made just subsequent to its official release, but despite the very large number of accesses made on the ε-TEX reference Web site, I am delighted to say that we have received no reports of bugs! This might, of course, simply mean that no-one is actually using ε-TEX, but I hope it means that ε-TEX is, as far as possible, bug free.

What are the features of ε-TEX? First and foremost, ε-TEX is 100% compatible with TEX: if you want to try ε-TEX, you can use it to process all of your legacy documents — it will produce results identical to TEX, right down to compatibility at the level of the TRIP test. When you are confident that ε-TEX can do everything that TEX does, you can try its extensions: there are approximately 30 of these, each intended to make the task of programming in TEX somewhat simpler; even when these extensions are enabled — that is, when ε-TEX is operating in so-called "extended mode" — ε-TEX will still process all existing TEX documents in a manner identical to TEX, provided of course that the document does not inadvertently refer to one of the new ε-TEX primitives. And finally, if you need to, you can use ε-TEX in "enhanced" mode, to get access to new features which are simply too radical to be 100% compatible with TEX: in the first release, there is only one enhancement, "TEX--XET", based on the earlier bi-directional typesetting system called "TEX-XET" by Don Knuth and Pierre MacKay. Unlike TEX-XET, TEX--XET requires no variant of DVI, no new pseudo device driver, and performs all of its operations internally.

To summarise, ε-TEX has three modes of operation. These are "compatibilty mode", in which ε-TEX behaves identically to TEX including full TRIP compatibility; "extended mode", in which ε-TEX provides approximately 30 new primitives but within which it continues to remain completely compatible with TEX, although it can no longer pass the TRIP test because of the presence of new primitives; and "enhanced mode", within which strict compatibility is sacrificed to allow additional features which are in some way fundamentally incompatible with TEX. The choice between compatibility or extended mode is made at the time a format is being generated: once the format is dumped, it contains within it a flag which indicates in which mode it is to operate, and it cannot be used in the alternative mode. The choice between extended and enhanced modes, on the other hand, is made at run time: even if an enhancement is enabled during format-creation, that enhancement will be automatically disabled before the format is dumped, and thus when the format is used the enhancement will initially be disabled. Enhanced mode can be entered only from extended mode, not from compatibility mode.

The extensions of ε-TEX V1

The extensions of the first version of ε-TEX can be classified into six distinct groups, plus a seventh which is accessible only in enhanced mode: these groups are

- Generalisation of the "`\mark`" concept;
- Additional control over expansion;
- Provision for re-scanning previously-read text;
- Environmental enquiries;
- Additional debugging features;
- Miscellaneous primitives;

and

- Bi-directional typesetting: the TEX--XET primitives.

A brief specification of the elements which compose these seven groups is given below, followed by a provisional specification of our plans for ε-TEX V2.

In ε-TEX V1, we took the initial step towards eliminating from ε-TEX the fixed limits which beset the current TEX language: we generalised the concept of a "`\mark`" into an array "`\marks`", with 256 elements in the first release. All of the related "`mark`" variables were similarly generalised into arrays of 256 elements.

We added a further twenty-three new primitives:

"`\protected`" is a prefix which can be used during macro definition. A macro defined as "`\protected`" will not expand during an "`\edef`", a "`\write`" or any similar operation in which expansion normally occurs. It will, however, expand if it reaches TEX's "stomach" (for example, during the actual typesetting process).

"`\detokenize`" is intended to be used just before a brace-delimited token list (a "general text", in the terminology of *The TEXbook*). It expands to yield a sequence of character tokens of category code 10 or 12 corresponding to the characters which compose the tokens of the unexpanded token list.

"`\unexpanded`" is also intended to be used just before a brace-delimited token list but it expands to yield the actual tokens of the token list. If TEX is performing an "`\edef`" or a "`\write`" or similar operation, no further expansion takes place, but if these tokens reach TEX's "stomach" (for example, during the typesetting process) then they will expand normally.

"`\readline`" is analogous to "`\read`", but treats each character as if it were currently of category code 10 or 12; the text read can then be scanned and re-scanned in different catcode environments using another new primitive "`\scantokens`".

"`\scantokens`" is intended to be followed by a brace-delimited token list, and decomposes the token list into the corresponding sequence of characters as if the token list were written to a file without expansion. It then applies TEX's "`\input`" mechanism to this sequence of characters, re-tokenising them according to the current catcode environment.

"`\currentgrouplevel`" is an internal read-only integer which returns the current group level at the point of call: in other words, it returns the current depth of group nesting.

"`\currentgrouptype`" is an internal read-only integer which returns the type of the innermost group as an integer in the range 0 to 16. These numbers can be converted to text strings using definitions provided in the ε-TEX macro library.

"`\ifcsname`" has the same syntax as TEX's "`\csname`" but is a Boolean "`\if`", yielding "`true`" if and only if the putative control sequence is already known to ε-TEX.

"`\ifdefined`" is analogous to "`\ifcsname`" but takes as parameter a control sequence or active character: it yields "`true`" if and only if the control sequence or active character is already known to ε-TEX.

"`\lastnodetype`" is an internal read-only integer which returns the type of the last node on the current list as an integer in the range −1 to 15 (the upper bound may be increased in a future version). These numbers can be converted to text strings using definitions provided in the ε-TEX macro library.

"`\eTeXversion`" is an internal read-only integer which contains the integral component of the combined version/revision number.

"`\eTeXrevision`" is a primitive which expands to yield a sequence of character tokens of category code 12 representing the fractional component of the combined version/revision number.

"`\showtokens`" is intended to be followed by a brace-delimited token list, and provides a simple way of displaying the contents of a particular element of the "`\marks`" family of arrays; it has many other potential applications.

"`\interactionmode`" provides read/write access to the current interaction mode. Assigning a numeric value sets the associated mode, while the current mode may be ascertained by interrogating its value. Symbolic definitions of these values are provided in the ε-TEX macro library.

"`\showgroups`" causes ε-TEX to display the level and type of all active groups from the point at which it was called.

"`\tracingassigns`", if set to a positive non-zero value, causes ε-TEX to display the value of registers both before and after assignment. Standard TEX displays only the new value, not the old.

"`\tracinggroups`" is an aid to debugging runaway-group problems. If it is set positive and non-zero, ε-TEX traces entry and exit to every group.

"`\tracingifs`" is an aid to debugging the expansion of conditionals. If it is set to a positive non-zero value, ε-TEX traces the flow of control through conditional statements.

"`\tracingscantokens`", when set to a positive non-zero value, causes ε-TEX to display an open-parenthesis and space whenever "`\scantokens`" is invoked; the matching close-parenthesis will be displayed when the scan is complete.

"`\tracingcommands`" is defined in TEX to provide different degrees of verbosity as its value is increased from zero to two; in ε-TEX, we provide greater verbosity and detail when it is set to values greater than two.

"`\everyeof`" is one of Knuth's "possibly good ideas", listed at the end of `tex82.bug`. It is analogous to the other "`\every...`" primitives, and takes as parameter a brace-delimited token list the tokens of which are inserted when the end of a file is reached.

"`\middle`" is analogous to TEX's "`\left`" and "`\right`" primitives: it specifies that the following delimiter is to serve both as a right and left delimiter. It will be set with spacing appropriate to a right delimiter with respect to the preceding atom, and with spacing appropriate to a left delimiter with respect to the succeeding atom.

"`\unless`" allows the sense of all Boolean conditionals to be inverted. For example, "`\unless \ifeof`" yields "`true`" if and only if end-of-file has not yet been reached.

The final class of primitives is composed of those that are accessible only when ε-TEX is operating in enhanced mode. An ε-TEX program enters enhanced mode when it assigns a positive non-zero value to one of the "enhanced state" variables, of which the only one in the first release of ε-TEX is "`\TeXXeTstate`". Once this has been assigned a positive non-zero value, five other primitives become accessible: "`\beginL`", "`\beginR`", "`\endL`", "`\endR`" and "`\predisplaydirection`". If these primitives are used when ε-TEX is not operating in enhanced mode, an error is reported.

"`\TeXXeTstate`" is an internal read/write integer which, when set to a positive non-zero value, enables use of the TEX--XET primitives;

"`\beginL`" indicates the start of a region which should be set left-to-right;

"`\endL`" indicates the end of a region which should be set left-to-right;

"`\beginR`" indicates the start of a region which should be set right-to-left;

"`\endR`" indicates the end of a region which should be set right-to-left;

"`\predisplaydirection`" is an internal read/write integer which is initialised by ε-TEX to indicate the direction of the last partial paragraph before a maths display; it is used to control the placement of elements such as equation numbers, and it can be directly set to alter this placement when appropriate.

The ε-TEX macro library

Although the majority of the effort in developing ε-TEX has been put into adding new functionality to the TEX language, we also spent a little time in developing a small macro library to accompany ε-TEX. In essence, this is a wrapper for the Plain format source, augmenting the existing definitions where appropriate to support the new primitives.

We also took the opportunity to add two features which we thought might be appreciated by the TEX community at large: we delayed the reading of patterns and exceptions until a natural language handling mechanism had been defined (so now patterns and exceptions are associated with a particular language, rather than being defined in limbo), and we added support for ε-TEX library files so that one can now load modules as an alternative to loading complete files.

The language handling mechanism is not predicated on the use of any particular natural language support system: it can be linked to Babel, for example, or to any other natural language handling system. In addition, hooks are provided so that user code can be threaded before and after language selection and in this way we hope to provide a sufficiently flexible language handling environment to support the needs of the majority of national TEX user groups. This is not to say that we do not foresee a rôle for Omega: on the contrary, it is clear from the Omega discussion list that there is a very real need for a system of Omega's complexity. However, we believe that for typesetting environments in which access is needed only to the major Western languages, ε-TEX will prove sufficient.

Availability

When ε-TEX was announced in late 1996, two reference implementations were available: Peter Breitenlohner's PubliC ε-TEX, which is a Turbo Pascal implementation for the IBM PC family running MS/DOS and Christian Spieler's VMS ε-TEX, a Pascal implementation for the VAX/VMS and

AXP/VMS family of machines. Since ε-TeX's release, it was first ported to the Commodore Amiga and then to the Windows 95/NT environments. During late February 1997, Bernd Raichle announced that his port of ε-TeX to Web2C Version 7 was also available for release; Bernd, who works at Stuttgart University, tells me that Eberhard Mattes is planning to release a combined ε-TeX/ML-TeX for the IBM PC family in the near future.

That summarises the present state of ε-TeX: in the remainder of this paper I will concentrate on the ideas which we are considering for ε-TeX V2.

Ideas for ε-TeX V2

We are looking at facilities for testing that a particular character exists within a given font:

"`\iffontchar`" takes two parameters, a font and a character number, and yields "true" if and only if the character exists within the font. It is easy to define a macro "`\ifchar`" which tests for the existence of a character within the current font.

There are four related primitives for finding the dimensions of a particular character in a given font: "`\fontcharwd`", "`\fontchardp`", "`\fontcharht`", and "`\fontcharic`" each take two parameters, a font and a character number, and return the width, depth, height and italic correction respectively. The effect cannot be achieved by setting a character in a box and then measuring the box because one or more of the dimensions may be negative. As with "`\iffontchar`", it is simple to define macros which performs the analogous tasks for the current font.

We are debating whether to incorporate a control sequence "`\iffont`", which would take as parameter the external name of a font and return "`true`" if and only if a font metric file of the same name can be opened: not all members of the group are convinced of the need for this, so we would welcome your comments on this idea.

In ε-TeX V1, we provided additional diagnostics which report the line number at which a group was opened if it has not been closed at end of program: in V2, we propose to report in addition the name of the file, although implementation differences may prevent us from returning the path to the file. We are also considering reporting if a group mis-match is detected at end of file, although this may be limited to a fairly simple test of group level rather than a full check to ensure that the file is left at the identical group to that at which it was entered.

We hope to avoid many of the problems which currently beset writers of output routines by providing access to the items which are discarded when a page break is taken. These will be made available until the next page break in a new reserved box called "`\pagediscards`". We do not consider it possible at the present time to provide similar access to material discarded during line-breaking.

In a similar area, we hope to make "`\vsplit`" more useful by enabling the programmer to remove the "`\topsplitglue`" which is currently inserted by TeX: we propose to implement this in a very general manner by providing four list destructors, one of which is already present in ε-TeX V1. The four destructors are "`\firstnodetype`", "`\lastnodetype`", "`\removefirstnode`" and "`\removelastnode`". In a future implementation we may allow access to these nodes as well as the option of simply removing them.

In TeX, "`\parshape`" is really a write-only quantity: only the number of entries in the current "`\parshape`" can be subsequently interrogated. In ε-TeX V2, we intend to provide read access to all of the dimensions of "`\parshape`", although we are not yet certain whether we will do this through a single control sequence "`\parshapedimen`" or through a pair of control sequences "`\parshapewidth`" and "`\parshapeindent`". No matter which is implemented, the control sequence will be indexed by an integer to return a particular dimension from the current "`\parshape`".

In a manner analogous to ε-TeX V1's "`\currentgrouplevel`" and "`\currentgrouptype`", we intended to provide in V2 new control sequences for interrogating the current flow of control through conditionals: "`\currentiflevel`" and "`\currentiftype`". We are also considering a third, "`\currentifbranch`", which will enable the programmer to determine whether the thread of expansion is currently the "`*\then`" or the "`\else`" branch, and also to determine if the current "`\if`" has not yet received sufficient tokens to decide which branch to take. It may also be possible to use this control sequence to determine which "`\or`" has been taken in an "`\ifcase`", but we are not yet sure that this is possible.

Again by analogy with ε-TeX V1's "`\showgroups`", we intend to provide "`\showifs`" in V2.

We are considering, but have not yet agreed upon, a new class of alignment, "`\malign`": this, if implemented, would provide a "maths alignment" primitive.

Continuing on the theme of removing fixed limits from ε-TeX, we are looking into the possibility of removing fixed bounds on the number of count registers, dimen registers, skip registers, muskip registers, token-list registers and boxes. If we are successful in implementing these, then we will probably remove the fixed limit on the number of marks at the same time.

During discussions in Brno, Don asked us to consider providing control over the looseness of the last line in a paragraph: although TeX typesets this to its natural length (if "`\parfillskip`" is infinite, as it usually is), Don said that traditionally this line was set to the same looseness as the previous line. We are looking into ways of providing not only these two boundary conditions, but at a continuum between those extremes, possibly through a control sequence "`\finaladjdemerits`".

We are trying to save stack space in ε-TeX V2 by eliminating redundant assignments (that is, values which will be restored to exactly the same value on exit from a loop). No new primitive will be required for this: it is simply an internal optimisation.

We feel that the "lost chars" message which currently goes into the log file is sufficiently important that it should appear on the console as well; accordingly we are extending the semantics of "`\tracinglostchars`" so that a value greater than 1 will cause the message to appear on the console as well as in the log.

We are looking into an idea which would enable a programmer to add material (for example crop marks) to a box being shipped out, even if the output routine has been rendered inaccessible (for example, by a complex format such as LaTeX). We will probably implement this through a control sequence "`\outpage`", analogous to "`\output`", the routine associated with which will be entered at the point at which a box is to be shipped out. This will also allow the programmer to overlay the box with material to be placed at fixed points on the page.

We are still discussing the implementation of "`\outpage`", and would welcome your advice as to whether you feel it should be recursive: that is, should a "`\shipout`" called from within an "`\outpage`" automatically invoke a further instantiation of "`\outpage`" unless the outer "`\outpage`" routine has already cleared the "`\outpage`" token-list register?

To allow for formats very different to the Plain/LaTeX/$\mathcal{AMS}$-TeX/L$\mathcal{AMS}$-TeX family (for example, ATML[1]), we are considering providing an alternative to the current default of appending "`.tex`" to an otherwise unqualified file specification on an "`\input`", "`\openin`" or "`\openout`" command. We envisage allowing the default extension to be specified explicitly. Here again your comments would be welcome: do you believe that this would be useful?

A recurring proposal, but one which we have still not entirely agreed upon, is the idea of an "`\evaluate`" primitive, which would allow arithmetic to be carried out in ε-TeX's mouth. Although we are all agreed that "`\evaluate`" would be very beneficial, we also realise that to implement it in a way that will guarantee repeatability across all platforms will require that it be implemented without recourse to the host's floating-point arithmetic. This in turn implies either a very limited implementation, or requires considerable time to implement a full floating-point package in software. We are still discussing which of these to adopt (if either) for ε-TeX V2.

During our discussions in Brno, Don asked us to investigate the idea of providing greater control over the spacing of fractions so as to permit, for example, less dependency on the use of kludges such as "`\sub \strut`". We have not yet developed a suitable model for this, but we are continuing to investigate the possibilities.

Finally two fairly major proposals: ML-TeX, and pdfTeX: Mike Ferguson, the author/creator of ML-TeX, has given Bernd Raichle free rein to oversee ML-TeX's future. Bernd and Peter Breitenlohner have made us aware that the present implementation has some deficiencies, particularly in terms of the timing of certain operations. We hope to be able to provide a better implementation, but will ensure that users of ML-TeX are able to contribute to discussions on its specification before any decisions are made.

As to pdfTeX, this is a fairly recent project, undertaken by Han The Thanh at Masaryk University in Brno. We are very impressed with Thanh's work, but are not convinced that the model which he has adopted is necessarily the best way to proceed: in particular, we are concerned that changes of the magnitude required to support his present implementation could introduce subtle bugs into TeX which may be hard to detect and which would adversely affect its stability. Accordingly we are interested in investigating a simpler model which would defer much of the processing to a post-processor similar to Sergey Lesenko's dvi2pdf, but as Thanh has already pointed out, this would be incompatible

[1] "A TeX Markup Language", presented at EuroTeX'95, Papendal.

with adopting a variant of the HZ algorithm[2] which he believes could be incorporated into his present implementation. We are still considering the options in this area, but would very much like to support the concept of pdfTEX in some form.

Finally we would like to express our thanks to all who have made this project possible:

- To Professor Don Knuth, without whom many of us would never have met; for his foresight in creating TEX; and for his willingness to discuss ideas for ε-TEX despite his incredibly busy schedule. And to both him and his wife Jill for making me feel so welcome to join them during their week in the Czech Republic last year.
- To Joachim Lammarsch, for instigating the project;
- To DANTE e.V., for underwriting it;
- To Rainer Schöpf and Joachim Schrod, for their invaluable contributions during the first year of the project;
- To Peter Breitenlohner, without whom ε-TEX simply would not exist: Peter has written virtually all of the Web code, and has been responsible for many of the ideas now in ε-TEX;
- To Bernd Raichle, who has also been responsible for many of the ideas in ε-TEX; and who volunteered to write the `e-TRIP` test, which is a daunting task;
- To Jiří Zlatuška, who as Project Leader for the $\mathcal{NTS}$ project has been very patient in waiting for the project to get the financial backing it needs, and who has in the meantime contributed much to the development of ε-TEX;
- To Friedhelm Sowa, who is continuing to investigate colour and user interfaces, and who acts as treasurer for the project;
- To all the members of DANTE, for making me so welcome each time I attend their meetings, for their courtesy in speaking to me in English, and for their encouragement whenever I try to speak a few faltering words of German. And for their magnificent donation of DM 30 000 to the $\mathcal{NTS}$ project, which will enable the project to finally become a reality.

Philip Taylor,
March 1997.

[2] Electronic Publishing, Vol 6(3), 283–288 (September 1993) "About micro-typography and the hz-program", Herman Zapf.