ε-TEX: a 100%-compatible successor to TEX

Following humbly in the foosteps of the Grand Wizard

Philip Taylor

Royal Holloway & Bedford New College
University of London
United Kingdom
p.taylor@vms.rhbnc.ac.uk

1 Introduction

ε-TEX is the first concrete result of an international research & development project, the $\mathcal{NTS}$ Project, which was established under the ægis of DANTE during 1992. The aims of the project are to perpetuate and develop the spirit and philosophy of TEX, whilst respecting Knuth's wish that TEX itself should remain frozen.

The group were very concerned that unless there existed some evolutionary flexibility within which TEX could react to changing needs and environments, it might all too soon become eclipsed by more modern yet less sophisticated systems. Accordingly they agreed to investigate a possible successor or successors to TEX, successors which would enshrine and encapsulate all that was best in TEX whilst being freed from the evolutionary constraints which Knuth had placed on TEX itself. To avoid any suggestion that it was TEX which the group sought to develop against Knuth's wishes, a working title of $\mathcal{NTS}$ (for New Typesetting System) was chosen for the project.

During the initial meetings of the $\mathcal{NTS}$ group, it became clear that there were two possible approaches to developments based on TEX: an evolutionary path which would simply continue where Knuth had left off, and which would use as its basis the source code of TEX itself (i.e. `TeX.Web`); the other a revolutionary path which would be based on a completely new implementation of TEX, using a modern rapid-prototyping language which could allow individual components of the system to be modified or replaced in a simple and straightforward manner. The group agreed that the latter (revolutionary) approach had much greater potential, but were aware that the re-implementation would be non-trivial, and would require external funding to bring it to fruition in finite time;

accordingly they agreed to concentrate their initial efforts on the former (evolutionary) path, and set to work to specify and implement a direct derivative of TeX which became known as ε-TeX. The ε of ε-TeX may be read as *extended*, *enhanced*, *evolutionary* or *European* at will (!), and is also an acknowledgement of the parallel developments which have lead the LaTeX3 team to modify their initial goal and to release an interim LaTeX, LaTeX2ε, which is directly derived from the earlier LaTeX sources.

The group took as starting point for the development of ε-TeX the many contributions which had been made on `NTS-L` (the open mailing list on which discussions pertinent to ε-TeX & $\mathcal{NTS}$ take place), together with the extremely interesting list of ideas which Knuth gives at the end of `TeX82.Bug`, and which he describes as *'Possibly nice ideas that will not be implemented'* (and which he contrasts with *'Bad ideas that will not be implemented'*!). Individual members of the group also contributed ideas of their own which had not necessarily been discussed publicly. All proposals were then subjected to a rigorous vetting procedure to ensure that they conformed to the ε-TeX philosophy, which may be summarised as follows:

> ε-TeX will in all ways demonstrate its affinity to, and derivation from, Knuth's TeX; it will be implemented as a change-file to `TeX.Web`, and will not exploit features which could only be achieved by using a particular implementation, operating system or language; it will be capable of being used successfully on a machine as small as an 80286-based PC or similar.
>
> At format-generation time, a user will have the option of generating either a TeX-compatible format or an ε-TeX format; if the TeX-compatible format is subsequently used in conjunction with ε-TeX, the result will be *Trip-compatible¿* (i.e. indistinguishable from TeX proper). If an ε-TeX format is generated and used in conjunction with ε-TeX, then provided that none of the new ε-TeX primitives are used, the results will be identical to those which would be produced using TeX proper. If an ε-TeX format is used in conjunction with ε-TeX and if one or more of the new ε-TeX primitives are used, then those portions of the document which are affected by the new primitive(s) may be processed in a manner unique to ε-TeX; other portions of the document will be processed in a manner identical to that of TeX proper. Only if an ε-TeX format is used in conjunction with ε-TeX and if an explicit assignment is made to one of the *enhanced-mode* variables to enable that particular enhanced mode will ε-TeX behave in a manner which may be distinguishable from that of TeX even if no other reference to an ε-TeX primitive occurs anywhere in the document. (These modes of operation are referred to as *compatibility-mode*, *extended-mode* and *enhanced-mode* respectively.)
>
> All new ε-TeX primitives will be syntactically identical to existing TeX primitives: that is, they will be either *control-words* or *control-symbols* within a normal catcode regime. Where an analogous primitive exists within TeX, the corresponding ε-TeX primitive(s) will occupy the same syntactic niche. Every effort will be made

to ensure that new ε-TEX primitives fit into the existing set of TEX datatypes; no new datatype will be introduced unless it is absolutely essential.

In brief, this implies that ε-TEX will follow the principle of least surprise: an existing TEX user, on using ε-TEX for the first time, should not be surprised by ε-TEX's behaviour, and should be able to take advantage of new ε-TEX features without having either to unlearn some aspects of TEX or to learn some new ε-TEX philosophy.

2 Installation

It is intended that ε-TEX be available ready-compiled for those systems for which precompiled binaries are the norm (e.g. MS-DOS, VMS, ...); for other systems such as Unix™, ε-TEX is supplied as a change-file which will need to be applied to `TeX.Web` in the normal way. However, since there will already be an implementation-specific change-file for the system of interest, some means will be required of merging `TeX.Web` with not one but (at least) two change-files; possibilities include `PatchWeb`, `Tie`, etc. , but if none of these is available then `WebMerge`, a TEX script, is supplied and can be used as a slower but satisfactory alternative. In practice, two or three change-files may be needed: the ε-TEX system-independent change-file, the TEX system-dependent change-file, and perhaps a small ε-TEX system-dependent change-file. The system-independent ε-TEX change-file is supplied as part of the ε-TEX kit, and sample system-dependent ε-TEX change-files are also supplied which may be used as a guide to those places at which system-dependent interactions are to be expected: an experienced implementor should have little difficulty in modifying one of these to produce an ε-TEX system-dependent change-file for the system of interest. Once ε-TEX has been tangled and woven, it should be compiled and linked in the normal way.

Once a working binary (or binaries, for those systems which have separate executables for `IniTeX` and `VirTeX`) has been acquired or produced, the next step will be to generate a suitable format file or files. Whilst ε-TEX can be used in conjunction with `Plain.TeX` to produce a Plain *e-format*, it is better to use the supplied `e-Plain.TeX` file which suplements the ε-TEX primitives with additional useful control sequences. When generating the format file, and regardless of the format source used, one fundamental decision must be made: is ε-TEX to generate a *compatibility mode* format, or an *extended mode* format? If the former, *all* ε-TEX extensions and enhancements will be disabled, the format will contain only the TEX-defined set of primitives, and any subsequent use of the format in conjunction with ε-TEX will result in completely TEX-compatible behaviour and semantics, including compatibility at the level of the `Trip` test. If the latter option, however, is selected, then all extensions present in ε-TEX will automatically be activated, and the format file will contain not only the TEX-defined set of primitives but also those defined by ε-TEX itself; any subsequent use of such a format in conjunction with ε-TEX will result in ε-TEX operating in *extended mode*; documents which contains no references to any of the ε-TEX-defined primitives will continue to generate results identical to those which would have been produced using TEX, but compatibility at the `Trip`-test level can no longer be accomplished, and of course any document which makes reference to an ε-TEX primitive will generate results which could not have been accomplished using TEX. It should be noted that neither a *compatibility mode* format nor an *extended mode* format may be used in conjunction with TEX itself; they are only suitable for use in conjunction with ε-TEX, since formats are not in general portable. Finally it should be emphasised that even if an *extended mode* format is generated, any document processed using such a format but not referencing any ε-TEX-defined primitive will produce results identical to those which would have been produced had the same document been processed using TEX; only if the document makes an explicit assignment to one of the *enhanced mode* state variables (`\TeXXeTstate` is the only instance of these in V1 of ε-TEX) will compatibility with TEX be compromised: ε-TEX is then said to be operating in *enhanced mode* rather than *extended mode*.

The choice between generating a *compatibility mode* format and an *extended mode* format is made at the point of specifying the format source file: assuming that the operating system supports command-line entry with parameters, then a normal TEX format-generation command would probably resemble:

```
IniTeX Plain \dump
```

or if the more verbose interactive form is preferred:

```
IniTeX
**Plain
*\dump
```

With ε-TEX, exactly the same command will achieve exactly the same effect, and the format generated will be a *compatibility-mode* format; thus assuming that the Ini-version of ε-TEX is invoked with the command `eIniTeX`, the following will both generate *compatibility-mode* formats:

```
eIniTeX Plain \dump
```

and

```
eIniTeX
**Plain
*\dump
```

In order to generate an *extended mode* format, the file-specification for the format source file must be preceded by an asterisk (`*`); whilst this may seem an inelegant mechanism, it has the great advantage that it avoids almost all system dependencies (GUI systems excepted, of course), and the asterisk as a component element of a filename is a very remote possibility (most filing systems reserve the asterisk as a ‘wild card’ character, which can therefore not form a part of a real file name *per se*). Thus to generate an *extended mode* Plain format, the following dialogue may be used:

```
IniTeX *Plain \dump
```

or

```
eIniTeX
***Plain
*\dump
```

and to generate an *extended mode* `e-Plain` format, the following instead:

```
eIniTeX e-Plain \dump
```

or

```
eIniTeX
***e-Plain
*\dump
```

Once suitable formats have been generated, they can then be used in conjunction both with `e-IniTeX` and `e-VirTeX` without further formality: in particular, no asterisk is needed (nor should be used!) if a format is specified, since the format implicitly defines (depending as its mode of generation) in which mode (compatibility or extended) ε-TEX will operate. Thus, for example, if a `Plain` format had been generated in *compatibility mode*, and an `e-Plain` format had been generated in *extended mode*, then both:

```
eIniTeX &Plain
```

and

```
eVirTeX &Plain
```

will cause ε-TEX to process any subsequent commands in *compatibility mode*. On the other hand, both

```
eIniTeX &e-Plain
```

and

```
eVirTeX &e-Plain
```

will cause ε-TEX to process any subsequent commands in *extended mode*, but only because the `e-Plain` format was generated in *extended mode*: it is not the *name* of the format, nor is it the contents of the *source* of the format, which determine the mode of operation – it is the *mode of operation* which was used when the format was generated. Any format generated in *compatibility mode* will cause ε-TEX to operate in *compatibility* *mode* whenever it is used, whilst the same format generated in *extended mode* will cause ε-TEX to operate in *extended mode* whenever it is used.

Although ε-TEX is completely TEX-compatible, and there is therefore no real reason why any system should need both TEX and ε-TEX, it is anticipated that until complete confidence exists in the compatibility of ε-TEX many sites and users will prefer to retain instances of each. For this reason the supplied change-files and binaries will ensure that both TEX and ε-TEX can happily co-exist on any system by a careful choice of non-overlapping name-spaces. This might, for example, by achieved by changing the default extension for *e-format* files to (say) `.efm` rather than `.fmt`, or by referencing a different format directory and/or environment variable (for example, `eTeX_formats` rather than `TeX_formats`).

3 The new features

Bearing in mind the contraints outlined in the introduction, the group identified approximately 30 new primitives which they believed would give added functionality to ε-TEX without compromising its compatibility with TEX; of the 30 new primitives, 25 are extensions (which by definition do not affect the semantics of existing TEX documents), whilst just six (all concerned with the implementation of TEX–XET) are associated with an enhancement. In addition to the new primitives, additional functionality was added to some existing primitives, and TEX's behaviour in some unusual boundary conditions was made more robust (this last has been subsumed in the most recent version of TEX, so this is no longer ε-TEX-specific).

The new features are listed and briefly described below, clustered together to indicate related functionality; it is intended that a full description of each together with appropriate examples will be published in *The ε-TEX Manual*, which it is hoped will become the definitive reference manual for ε-TEX.

3.1 Additional control over expansion

- `\protected`
- `\detokenize`
- `\unexpanded`

`\protected` is a prefix, analogous to `\long`, `\outer`, and `\global`; it associates with the macro being defined an attribute which inhibits expansion of the macro in expansion-only contexts (for example, within the parameter text of a `\write` or `\edef`); if, however, the parser or command processor (TEX's 'œsophagous' and 'stomach', in Knuth's alimentary paradigm) is currently demanding a *command*, then the `\protected` macro will expand in the normal way. This behaviour is identical to that displayed by the explicit expansion of a token-list register through the use of `\the`; the same model is used elsewhere in ε-TEX to achieve a consistent paradigm for `partial expansion`.

\detokenize, when followed by a *general text*, expands to yield a sequence of character tokens of catcode 10 (*space*) or 12 (*other*) corresponding to a decomposition of the tokens of the *balanced text* of the unexpanded *general text*; c.f. \showtokens. The effect is rather as if \scantokens (q.v.) were applied to the *general text* within a regime in which only \catcodes 10 and 12 existed. Note that in order to preserve the boundaries between *control words* and any following *letter*, a *space* is yielded after each control word including the last.

\unexpanded, when followed by a *general text*, expands to yield the *balanced text* of the unexpanded *general text*. No further expansion will occur if ε-TEX is currently performing a \write, \edef, etc. , but further expansion will occur if the parser or command processor is currently demanding a *command*. The effect is as if the *general text* were assigned to a token list register, and the latter were then partially expanded using \the, but no assignment actually takes place; thus \unexpanded can be used in expansion-only contexts.

3.2 Provision for re-scanning already read text

- \readline
- \scantokens

\readline is analogous to \read, but treats each character as if it were currently of catcode 10 (*space*) or 12 (*other*); the text thus read is therefore suitable for being scanned and re-scanned (using \scantokens, q.v.) under different catcode regimes.

\scantokens, when followed by a *general text*, decomposes the *balanced text* of the *general text* into the corresponding sequence of characters as if the *balanced text* were written unexpanded to a file; it then uses TEX's \input mechanism to re-process these characters under the current catcode regime. As the \input mechanism is used, even hex notation (^^xy) will be re-interpreted. Parentheses and a single space representing the pseudo-file will be displayed if \tracingscantokens (q.v.) is positive and non-zero.

3.3 Environmental enquiries

- \eTeXrevision
- \eTeXversion
- \grouplevel
- \grouptype
- \ifcsname
- \ifdefined
- \lastnodetype

\eTeXrevision: an primitive which expands to yield a sequence of character tokens of catcode 12; these represent the minor component of the combined version/revision number. Pre-release versions will be characterised by an initial minus sign (-), whilst post-release versions will be implicitly positive; both will contain an explicit leading decimal point, which will follow any minus sign present.

`\eTeXversion`: an internal read-only integer representing the major component of the combined version/revision number.

`\grouplevel`: an internal read-only integer which returns the current group level (i.e. depth of nesting).

`\grouptype`: an internal read-only integer which returns the type of the innermost group as an integer in the range 0..16. Textual definitions of these types are provided through the an associated macro library, but it is intended that these definitions shall be easily replaceable by national language versions in environments within which English language texts are sub-optimal.

`\ifcsname`: similar in effect to the sequence
`\unless \expandafter \ifx \expandafter \relax \csname`
but avoids the side-effect of the `cs-name` being ascribed the value `\relax`, and also does not rely on `\relax` having its canonical meaning. No hash-table entry is used if `cs-name` does not exist. (`\unless` is explained below.)

`\ifdefined`: similar in effect to `\unless \ifx \undefined`, but does not require `\undefined` to actually be undefined, since no explicit comparison is made with any particular control sequence.

`\lastnodetype`: an internal read-only integer which returns the type of the last node on the current list as an integer in the range $-1..15+$ (only values $-1..15$ are defined in the first release, but future releases may define additional values). Textual definitions of these types are provided through an associated macro library, but it is intended that these definitions shall be easily replaceable by national language equivalents for use in environments within which the use of English language texts is sub-optimal.

3.4 Generalisation of the \mark concept: a class of \marks

- `\marks`
- `\botmarks`
- `\firstmarks`
- `\topmarks`
- `\splitfirstmarks`
- `\splitbotmarks`

`\marks`: whereas TEX has only one `\mark`, which has to be over-loaded if more than one class of information is to be saved (e.g. over-loading is necessary if separate information for recto and verso pages is to be maintained), ε-TEX has a whole class of `\marks` (16, in the first release); thus rather than writing `\mark` *general text* as in TEX, in ε-TEX one writes `\mark` `4-bit number` *general text*. For example, `\marks 0` could be used to retain information for the verso page, whilst `\marks 1` could retain information for the recto.

There are equivalent classes for the five `\marks` variables `\botmarks`, `\firstmarks`, `\topmarks`, `\splitfirstmarks` and `\splitbotmarks`.

3.5 Bi-directional typesetting: the TEX–XET primitives

- `\TeXXeTstate`
- `\beginL`
- `\beginR`
- `\endL`
- `\endR`
- `\predisplaydirection`

TEX–XET was developed by Peter Breitenlohner based on the original `TeX-XeT` of Donald Knuth and Pierre MacKay; whereas `TeX-XeT` generated non-standard `DVI` files, TEX–XET generates perfectly normal `DVI` files which can therefore be processed by standard `DVI` drivers (assuming, of course, that the necessary fonts are available). Both systems permit the direction of typesetting (conventionally left-to-right in Western documents) to be reversed for part or all of a document, which is particularly useful when setting languages such as Hebrew or Arabic.

`\beginL`: indicates the start of a region (e.g. a section of text, or a pre-constructed box) which should be set left-to-right;

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

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

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

`\TeXXeTstate`: an internal read/write integer, its value is zero or negative to indicate that TEX–XET features are not to be used; a positive value indicates that they may be used. As the internal data structures built by TEX–XET differ from those built by TEX, and as the typesetting of a document by TEX–XET may therefore differ from that performed by TEX, `\TeXXeTstate` defaults to zero, and even if set positive during format creation will be re-set to zero before the format is dumped. Explicit user action is therefore required to enable TEX–XET semantics, and TEX–XET is thereby classed as an *enhancement*, not simply an *extension*.

3.6 Additional debugging features

- `\interactionmode`
- `\showgroups`
- `\showtokens`
- `\tracingassigns`
- `\tracinggroups`
- `\tracingifs`
- `\tracingscantokens`
- Additional detail for `\tracingcommands`

`\interactionmode`: whereas in TEX there exist only explicit commands such as `\scrollmode`, `\errorstopmode`, etc. , in ε-TEX read/write access is provided via `\interactionmode` (an internal integer); assigning a numeric value sets the associated mode, whilst the current mode may be ascertained by interrogating its value. Symbolic definitions of these values are provided through an associated macro library, but it is intended that these definitions shall be easily replaced by national-language equivalents in environments within which the use of English is sub-optimal.

`\showgroups`: `(e-)TeX` has many different classes of group, which should normally be properly balanced and nested; if a nesting or imbalance error occurs, it can be *very* difficult to track down the source of the problem. `\showgroups` causes ε-TEX to display the level and type of all active groups from the point within which it was called.

`\showtokens`, when followed by a *general text*, displays a sequence of characters corresponding to the decomposition of the `balanced text` of the unexpanded *general text*; c.f. `\detokenize`.

`\tracinggroups`: a further aid to debugging runaway-group problems, the command `\tracinggroups` (an internal read/write integer) causes ε-TEX to trace entry and exit to every group while set to a positive non-zero value.

`\tracingscantokens`: an internal read/write integer, assigning it a positive non-zero value will cause an open-parenthesis and space to be displayed whenever `\scantokens` is invoked; the matching close-parenthesis will be recorded when the scan is complete. If a traceback occurs during the expansion of `\scantokens`, the first displayed line number will reflect the logical line number of the pseudo-file created from the parameter to `\scantokens`; thus enabling `\tracingscantokens` can assist in identifying why an seemingly irrational line number is shewn as the source of error (the traceback always continues until the line number of the actual source file is displayed).

If `\tracingcommands` is greater than 2, additional information is displayed.

3.7 Miscellaneous primitives

- `\everyeof`
- `\middle`
- `\unless`

`\everyeof`: this is one of Knuth's 'possibly good ideas', listed at the end of `TeX82.Bug`; analogous to the other `\every...` primitives, it takes as parameter a *balanced text*, the tokens of which are inserted when the end of a file (either real or virtual, if `\scantokens` is used) is reached. This allows `\input` statements to be used within the replacement text of `\edef`s, and allows totally arbitrary files to be `\input` within a ε-TEX conditional, since the necessary `\fi` can be inserted before ε-TEX complains that it has fallen off the end of the file.

`\middle`: analogous to TEX's `\left` and `\right`, `\middle` 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 w.r.t. the preceding atom(s), and with spacing appropriate to a left delimiter w.r.t. the succeeding atom(s).

`\unless`: TEX has, by design, a rather sparse set of conditional primitives: `\ifeof`, `\ifodd`, `\ifvoid`, etc. , have no complementary counterparts. Whilst this normally poses no problems since each accepts both a `\then` (implicit) and an `\else` (explicit) part, problems occur when one is used as the final `\if`... of a `\loop ... \if ... \repeat` construct, since no `\else` is allowed after the final `\if`.... `\unless` allows the sense of all Boolean conditionals to be inverted, and thus (for example) `\unless \ifeof` yields `true` iff end-of-file has *not* yet been reached.

4 What next?

At the time of writing, ε-TEX version 1 is ready to go to `TeX-Implementors`, although work remains to be done on the `eTrip` test. Whether it will have been released to the implementors before the conference cannot be predicted, since I discovered today that I have only four working days left before I leave for Europe, and it will not be possible to release it once I am away. The version being prepared for the implementors is termed Version 1β (the $\mathcal{NTS}$ team themeselves acted as α-testers). Once the implementors have given us the go-ahead and said that in their opinion ε-TEX is a viable alternative to TEX (by which I mean that it is completely compatible, and functions according to the accompanying documentation), we will release it to the TEX world as a whole. We will react as quickly as possible to any bug reports (we sincerely hope that there will be few!), and we will then concentrate on new features for version 2. We certainly intend to work as closely as possible with the LATEX2ε team, not because we believe that LATEX2ε is necessarily right for everybody, but because (a) we respect the intellect and knowledge of the members of the LATEX2ε team, and (b) because it might be possible to enable them to achieve things with LATEX and ε-TEX which would either be impossible or extraordinarily difficult with LATEX and TEX. We have a very long list of suggestions from Nelson, we still have many of Knuth's 'possibly good ideas' to consider, and we have an enormous number of suggestions made on NTS-L: we are unlikely to run out of ideas for many years yet!

5 Acknowledgements

I would like to thank many people without whom both the project and this paper would simply never have come to fruition. I would like to thank above all Don Knuth, for creating TEX and for making it so freely available; for sparing us the time to discuss the project when we met last year at Stanford; and for his willingness to allow us to base ε-TEX on TEX. I would also like to thank all those who have contributed ideas, either via personal communication or via the NTS list, and I would single out Nelson

Beebe for presenting us with an extremely well thought through and well presented set of proposals. I would like to thank DANTE, without whose financial support we could never have afforded to meet (and experience has shewn that *unless* we meet fairly regularly, very little gets done!). And finally I would like to thank all the members of the team, who have contributed ideas, time, enthusiasm and expertise: they are Joachim Lammarsch (Managing Director), Peter Breitenlohner (project leader, ε-TEX), Jiří Zlatuška (project leader, $\mathcal{NTS}$), Bernd Raichle (2-i/c, ε-TEX & $\mathcal{NTS}$ projects), and Friedhelm Sowa (color, and user interfaces). And I would like to single out for a special vote of thanks Peter Breitenlohner: without his expertise in `TeX.Web`, there is absolutely no doubt whatsoever in my mind that this project would *never* have been possible: thank you Don, thank you Peter, thank you everyone.