Brno
June 1995

LaTeX 2ε Encoding Interfaces

Purpose, Concepts, and Open Problems

Frank Mittelbach

Zedernweg 62

55128 Mainz

Germany

Contents

1 Overview

- Input encodings (keyboard encodings) introduced with LaTeX 2$_\varepsilon$ 1994/12/01
- Output encodings (font encodings) introduced with NFSS
- Standard internal representations (internal encoding)
- Future tasks

This talk will describe the models used by LaTeX 2$_\varepsilon$ to translate input characters in a source document via internal representations to glyphs in a font. This is important for everybody writing in a language other than English and/or using a TeX system that allows 8-bit input.

The talk will cover

- input encodings (keyboard encodings) introduced with LaTeX 2$_\varepsilon$ 1994/12/01
- output encodings (font encodings) introduced with NFSS
- standard internal representations

Future issues discussed will be a proposal for 8-bit math encoding developed by the LaTeX3 Project and a general mechanism for providing "short references" to ease coding tasks such as `"a` → `\"{a}` as they are already available in certain language files.

Short refs are clearly part of input encodings: indeed, inputenc could be looked on as making all 128 upper input chars into short refs.

* * *

To set the stage ...

2 What's that?

GrŤáe

That's the German word "Größe" (height) typeset on a German PC and displayed using T1 fonts.

3 And what's that?

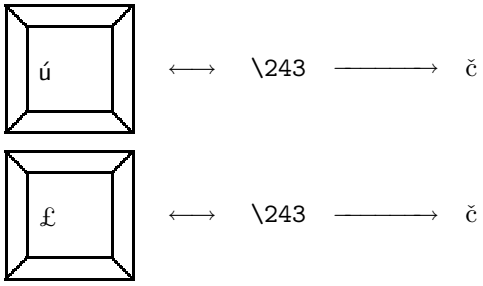
$\$$

That's somebody trying to typeset a pound sign but getting a dollar sign.

* * *

Let's look at the encoding problems in some more detail ...

4 One Encoding Problem



Key . . . In file (and on screen) . . . Typeset output: oops!

Top line is (IBM) code page cp850.

Second line is ISO-Latin1.

Both with no input or output translation: both with T1 fonts.

Needs
Input encoding: translates what is in the input file into:
Internal (LaTeX) representation;
Output encoding: translates to correct code (for the font) in the dvi file.

5 Input Encodings

- The mapping between keyboard glyphs and character numbers in the `.tex` source file
- There are up to 256 character numbers available (8-bit)
- It is possible that a source file is composed using several input encodings

The input encoding describes the relationship between the 8-bit characters in your input file and their meaning, i.e. the glyphs you will see if you display the source file with a browser (if that browser uses the same encoding.

6 Input Encodings: Possible Realisations

- Translate characters using hardwired tables inside TEX the program
- Translate characters to a standard representation using external programs
- Translate characters inside LATEX by declaring translation tables

The first solution will result in source files which are not portable and whose non-portability will only be visible by looking at the typeset output.

The second solution produces portable documents if you consider the translated source file as the major source.

Neither of the two allows to mix input encodings within one document.

Only the third solution will result in fully portable documents which will produce identical output at different sites. However, it has the disadvantage of taking up space within the LATEX format and also of needing extra processing time.

7 Input Encodings: The LaTeX 2_ε solution

- 7-bit printable ASCII is considered to be essentially transparent
- Input encodings are declared by name.
- Each named encoding defines the mapping for (a subset of) the character numbers between 128–255 to the internal encoding of LaTeX 2_ε
- Input encodings are declared for a whole document or for parts of it

Exceptions are the standard escape characters for LaTeX (since they are not passed through) and the non-printable ASCII characters at the moment.

This means, for example, that encodings like EBCDIC (in any variant are not supported. This isn't totally surprising because to be able to support an input encoding it must at least have the characters that form TeX's command language in well-defined positions.

8 Another Encoding Problem

`\textrm{Hung\H{a}rian umlaut}` —cmr10→ Hungărian umlaut

`\texttt{Hung\H{a}rian umlaut}` —cmtt10→ `Hungarian umlaut`

This shows that the encoding of the original TEX fonts (`OT1`) is unfortunately not completely fixed.

9 Output Encodings

- The mapping between glyphs and character numbers in the `.dvi` file
- There are up to 256 character numbers available for TEX fonts
- Usually typesetting involves several fonts, not all with the same output encoding

The output encoding describes the relationship between the 8-bit characters put into the `.dvi` by TEX and the glyphs that should be produced for them.

10 Output Encodings: Possible Realisations

- Translate characters using hardwired tables inside TEX the program
- Use a TEX extension: MLTEX's `\charsubdef`, Omega, NTS ...
- Translate characters inside LATEX by declaring translation tables

The first solution only works for characters written to plain files not to character numbers written to the `.dvi` file and thus is not usable.

None of the TEX extensions is generally available. MLTEX doesn't give a general solution. Omega might (**?**) do — but I haven't studied it yet.

Only the third solution will right now result in fully portable documents which will produce identical output at different sites. However, it has the disadvantage of taking up space within the LATEX format and also of needing extra processing time.

11 Output Encodings: The LaTeX 2_ε solution

- Provide a standard internal encoding
- Translate characters inside LaTeX by declaring translation tables from the internal encoding to any output encodings

So have a look at the internal encoding which will lead us eventually back to the implementation of output encodings ...

12 Internal Encoding

- Must mediate between the input and output encodings
- Must be 7-bit to ensure system independence while reading and writing external files
- Should be independent of the input encoding(s) used
- Should be independent of the font(s) used for typesetting

Internal representation:
Must be 7-bit printing-chars since LaTeX uses external files to store information between runs;
reading and writing to these files must be system-independent.

13 Internal Encoding: Possible Realisations

- Transparent: the plain TEX & LATEX 2.09 solution
- Mediating: the LATEX 2_ε solution

The character number of the input character (placed into the source file) is passed unchanged to the `.dvi` file; commands always generate the same output character position, e.g. `\it\$` →£ — in other words: essentially no internal encoding.

This does work as long as input, internal, and output encoding are essentially the same, as happened with the early versions of TEX/LATEX with only a limited number of fonts and only 7-bit input.

14 Internal Encoding: The LaTeX 2_ε solution

- 7-bit printable ASCII
- Encoding specific commands
  - without arguments, e.g. `\textquoteleft`
  - with arguments containing 7-bit printable ASCII or further commands, e.g. `\"{a}` or `\^{\i}`

7-bit printable ASCII is considered to be essentially transparent

Encoding specific commands are commands that change their meaning when the output encoding changes.

We decided to represent accented characters by commands with arguments a) to avoid taking up an enormous name space, b) to allow for accented characters which are not in the font, and c) to allow for the use of `\uppercase` an issue which will be discussed later on.

For the really dedicated I suggest having a look at the actual implementation in `ltoutenc.dtx` which contains quite complicated code to handle spacing, ligatures and kerns correctly.

15 Encoding Specific Commands: Concepts

- Commands change their meanings if used with different output encodings
- Change happens, when the command gets used not when the output encoding changes
- Commands are robust, i.e. they stay unchanged during LaTeX's internal processing, including writing out to external files and reading back in

Lots to say here ...

* * *

Let us now turn to the problem of changing the case of letters and its relation to hyphenation within TeX ...

16 The `\lowercase` and the `\lccode` Table

The `\lowercase` command:

- Changes character code but keeps category codes (e.g. `\active`)
- Doesn't act on commands

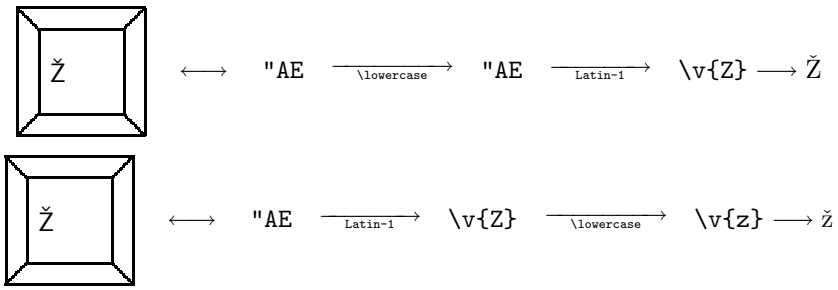
The `\lccode` table:

- Used by the `\lowercase` primitive
- Used to translate words before applying hyphenation patterns
- For hyphenation: table is consulted only at the end of the paragraph

Let's have a look at these issues in detail.

17 Example: Using \lowercase on input

Keybord encoding latin2:



`"AE` is the character "ő" (hungarian umlaut over o) in the T1 encoding, the encoding which is used by default to set up the lower and upper case tables.

In other input encodings one might end up with some active character which isn't even defined.

18 Example: Hyphenation and the `\lccode` Table

MANŒUVRES Manœuvres manœuvres

- No hyphenation if any char has `\lccode` of zero
- Incorrect hyphenation for first word if `\lccode` of "M" isn't "m", the one for "Œ" isn't "œ", etc.
- Incorrect hyphenation for second word if `\lccode` of "M" isn't "m", the one for "œ" isn't "œ", etc.
- Incorrect hyphenation for third word if `\lccode` of any character used is different from the character itself

Now in what encoding are these characters at this point when TEX is trying to apply hyphenation to them?

ANSWER: in the output encoding of the font in which they will be typeset. They must be, since after hyphenation the char codes are simply written to the dvi file and if they aren't the char codes that match the position of "Œ", "œ", "u", "v", "r", "s", etc. then you will find garbage on your printed page.

This is a slight oversimplification because TEX works harder by breaking up ligatures and possibly reinserting new ones etc.

* * *

The characters must be in the output encoding of the current font otherwise one will end with something ... like this.

19 Example: Hyphenation and the `\lccode` Table

МАНŒУВРЕС Манœуврес манœуврес

- Но хыпхенатион иф аны чар хас `\lccode` оф зеро
- Инцоррецт хыпхенатион фор фирст щорд иф `\lccode` оф “М” исн’т “м”, тхе оне фор “Œ” исн’т “œ”, етц.
- Инцоррецт хыпхенатион фор сецонд щорд иф `\lccode` оф “М” исн’т “м”, тхе оне фор “œ” исн’т “œ”, етц.
- Инцоррецт хыпхенатион фор тхирд щорд иф `\lccode` оф аны чарацтер усед ис дифферент фром тхе чарацтер ицелф

This is what happens if the encoding used doesn’t match the font (output) encoding.

The typewriter comes out correctly because `\verb` switches back to the typewriter font in the main document encoding.

20 Example: Hyphenation in a Paragraph

```
Some text with a \russian{Russian phrase} in the middle.
```

TEX3 enables

- Both parts of the text to be hyphenated with the correct set of hyphenation patterns for that language

But

- Both parts are changed to lowercase using the same `\lccode` table: the one that is current at the end of the paragraph

Thus

- Hyphenation will be wrong if output encodings are used which need different `\lccode` tables

21 Consequences

- All output encodings used by LaTeX 2_ε must have the same `\lccode` table (Cork T1 encoding)
- Direct use of `\uppercase` and `\lowercase` (on the input encoding) is not supported
- Instead, LaTeX 2_ε provides `\MakeUppercase` and `\MakeLowercase` which operate on the internal encoding

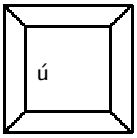
Keep in mind that the fixing the `\lccode` tables restricts font encodings only slightly. The T1 encoding is far from perfect but it is the agreed on TeX standard.

The alternative would be to make up to 600 assignments each time an encoding changes (and that doesn't solve the problem of multiple encodings within a paragraph).

But the `\Make*` commands will not always do exactly what you want: eg maths letters, private commands, ...

22 The LaTeX 2_ε Encoding Solution

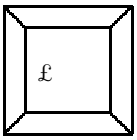
```
\usepackage[cp850]{inputenc}
```



$\longleftrightarrow$ `\243` $\xrightarrow[\text{cp850}]{}$ `\'{u}` $\xrightarrow[\text{T1}]{}$ `\char250` $\longrightarrow$ ú

$\xrightarrow[\text{OT1}]{}$ `{\accent 19 u}` $\longrightarrow$ ú

```
\usepackage[latin1]{inputenc}
```



$\longleftrightarrow$ `\243` $\xrightarrow[\text{Latin-1}]{}$ `\textsterling` $\xrightarrow[\text{T1}]{}$ `\char191` $\longrightarrow$ £

$\xrightarrow[\text{U}]{}$ `\Error...` $\longrightarrow$

23 Future tasks

- Math font encoding (8-bit)
- All caps fonts (new NFSS axis?)
- Short references, e.g.

  `"a` → ä or
  `->` → `\rightarrow`