The alttex package

Arno L. Trautmann[*]

Version 0.c 2010/01/09

This is the package `alttex` which will try to give an experimental new way to write LaTeX code using the modern engines luaTeX or XeTeX. So far most of the code is done with very dirty code and actually it's a collection of things that come into my mind during boring lectures. Maybe someone will have fun with the following code fragments.

Contents

I Introduction **2**

1 Why this package? **2**

II Implementation: alttex.sty **3**

2 Textmode **3**
2.1 no escape 3
2.2 tabular 4
2.3 excel tabulars 4
2.4 tabbing 5

3 Math stuff **5**
3.1 braces 5
3.2 huge display math 6
3.3 unicode math 6
3.4 Lazy underscript and superscript 8
3.5 matrices 9

4 Lists and such things **10**
4.1 itemize with a single character 10
4.2 enumerate with a single character 12

5 Known Bugs **14**

[*]arno.trautmann@gmx.de

Part I
Introduction

1 Why this package?

The problem I have with LaTeX[1] is the antique way of typing. Because most people still use a hopelessly outdated keyboard layout (»qwerty« or slightly adapted versions of that), LaTeX doesn't make use of some cool features. I'm not talking about writing chinese or arabic text, but something like the following example:

In standard LaTeX, one has to write

```
This is the normal text, then comes the itemization:
\begin{itemize}
  \item text for first item
  \item \begin{itemize}
        \item this is an item inside an item…
        \item[$\Rightarrow$] Here an item with a formula: $\int_a^b x^2 dx$
        \end{itemize}
  \item and the outer itemize goes on…
\end{itemize}
```

That is quite some code to write and is not very good readable. Of course, a good editor has a shortcut to save time at typing, but the code still looks more like programming than like typesetting. Using this package and having a superior keyboard layout[2], you can simply write this code and get exactly the same output:[3]

```
This is the normal text, then comes the itemization:

• text for first item
•   ‣ this is an itemize inside an item
    ‣[⇒] Here an item with a formula: $∫_a^b x² dx$

• and the outer itemize goes on…

And your normal text goes on…
```

Well, actually I'm lying now because this is not fully implemented so far. But it's the aim of this package to provide this – besides many, many other funny and cool things. The aim is to offer a more „wysiwyg“ way, without loosing anything of logical markup. One still can re`\def`ine the • if he doesn't like the way his items look and so on. The above code is not meant to just typeset the character •, but this is rather a command itself to be there instead of the command `\item`.

[1] I'll write LaTeX instead of XeLaTeX or luaLaTeX—saves me two resp. three keystrokes. Most of the code below *only* works with XeLaTeX. If you need support for [utf8]inputenc, please contact the author. Support for luaLaTeX is work in progress.

[2] E. g. the ergonomic layout Neo: `http://neo-layout.org/`

[3] The lmodern font I'm using here does not have the symbol for the inner item , so we change to DejaVu Sans Mono here. If you compiled this document on your computer without having DejaVu Sans Mono at hand, there might be some characters missing. Please consult the documentation online, then.

Part II

Implementation: alttex.sty

Ok, enough blahblah, now comes the code. We begin with preamble stuff:

```
\ProvidesPackage{alttex}
  [2010/01/09 v 0.c alternative way of TeXing]
\RequirePackage{expl3}
\ExplSyntaxOn
\RequirePackage{amsmath}
```

Checking wether XƎTEX or luaTEX are used. (Not yet implemented, as we don't make use of any luaTEX-specific functionality so far. This will sure change in the future!)

`\usepackage` Now, this is the first highlight. It is an extremely simple and stupid approach to load missing packages on-the-fly, just like MikTEX does. We re`\def`ine the `\usepackage` and hope, it works. Only working with texlive! If you're using MikTEX, put a

```
\let\usepackage\altpackage
```

into your preamble, *directly* after loading `alttex`. If this does not work, delete the following lines from your `alttex.sty`.

```
\cs_set_eq:NwN\alt_oldpackage:n\usepackage
\cs_set_eq:NwN\altusepackage:n\usepackage % to restore at document level
\cs_set:Npn\usepackage#1{
  \file_if_exist:nTF{#1.sty}{
    \alt_oldpackage:n{#1}
    }{
    \immediate\write18{tlmgr~install~#1}
  }
}
```

So far, this code seems to be a bit buggy, but it should work anyhow.

Now load some nice packages and testing wether you're running XƎLATEX or not.

```
\RequirePackage{exscale}
\RequirePackage{hhline}
```

We need `exscale` to write really big formulae, and `ifxetex` to check wether one uses the correct engine.

2 Textmode

2.1 no escape

`\noescape` You want to write plain text. Maybe you're annoyed by always escaping characters like _ # & { } $ ~ and so on. `\noescape` allows you to never escape anything—except the \, which still might be used for `\textit{}` or so. Or maybe not… because the { } are not escaped. Have to think about this one. Maybe the \ will be redefined to define { } by itself.

```
\cs_new:Npn\noescape{
  \char_set_catcode:w`\_= 11%
  \char_set_catcode:w`\^= 11%
  \char_set_catcode:w`\#= 11%
  \char_set_catcode:w`\&= 11%
  %\char_set_catcode:w`\{= 11%
  %\char_set_catcode:w`\}= 11%
  \char_set_catcode:w`\$= 11%
  \char_set_catcode:w`\~= 11%
  \char_set_catcode:w`\@= 11%
  \char_set_catcode:w`\%= 11
}
```

Changing `@` is not necessary here, but it fitted in nicely, so I leave it. Using luaTEX, one would just call another catcode table instead of these manual changes.

`\oldescape` Of course this has to be reset when doing anything like formula, tabular etc. Maybe I will be able to change the behaviour automatically. This idea has been inspired by a discussion on the ConTEXt mailinglist.

```
\cs_new:Npn\oldescape{
  \char_set_catcode:w`\%= 14%
  \char_set_catcode:w`\_= 8%
  \char_set_catcode:w`\^= 7%
  \char_set_catcode:w`\#= 6%
  \char_set_catcode:w`\&= 4%
  %\char_set_catcode:w`\{= 1%
  %\char_set_catcode:w`\}= 2%
  \char_set_catcode:w`\$= 3%
  \char_set_catcode:w`\~= 13%
  \makeatother%
}
```

2.2 tabular

The way one has to type extensive tabulars is quite complex – and the resulting code is often not easy to read. I don't have good ideas how to change this, but I'm thinking about it. Mail me any suggestions for this!

This will be the first attempt to make tabulars easier: Mostly you want an `\hline` after an `\\`. So let's try something like:

I will try to implement cool stuff from the `hhline`-package.

`\§ for \\\hhline` Type `\–` (an en-dash) at the end of a line, and you get an `\hhline`. Type `\=` to get a double line

```
\cs_set:Npn\–{\hhline}
\cs_set:Npn\={\hhline}
```

This is shurely not a good symbol for this purpose, but I don't have a better idea so far. At least it's a ”bar“, so one can guess what it should do.

2.3 excel tabulars

`\exceltabular` Often one usese a program to calculate tabulars of numbers. To insert it into LATEX, one has to do some work. Here we try to copy-paste the tabular from excel, Calc or any other program to a file mytabular.txt (or any other ending). Then you say `\exceltabular{mytabular}` (you do not need the ending, therefor it doesn't matter) and you get the tabular in a standard format. I will extend this to enable caption, variable number of columns, kind of rule used etc. This is just a very first test.

This is the definition of the command:

```
\cs_new:Npn\exceltabular#1{
  \char_set_catcode:w`\^^I=4\tex_relax:D
  \alt_eolintabular:%
  \begin{tabular}{|c|c|c|}\hline%
  \file_input:n{#1}%
  \end{tabular}%
  \char_set_catcode:w`\^^M=5\tex_relax:D
}
```

And a little helper function to make the <enter> `\active`. Again, thanks to the people on the mailinglists.

```
\cs_new:Npn\alt_linebreak:{\\\hline}
\tex_begingroup:D
  \tex_lccode:D`\~=`\^^M%
\tex_lowercase:D{%
  \tex_endgroup:D
  \cs_new:Npn\alt_eolintabular:{%
    \char_set_catcode:w`\^^M=\active
    \cs_set_eq:NwN~\alt_linebreak
}%
}
```

2.4 tabbing

This will be analog to the `\exceltabular`. You write your tabbing using tabs and <enter>. That's it :)

`\alttabbing` Not yet implemented!

3 Math stuff

3.1 braces

`\newbraces` `\oldbraces` Now this is something most LaTeX-beginners don't recognize and wonder why the formula looks so ugly: The braces () do not fit to the hight of the formula. This can be achieved by putting `\left` and `\right` in front of the braces. But actually, this is annoying! In almost any case you want this behaviour, so this should be the standard. So we redefine the way braces are handled. With `\newbraces` the ( ) always fit. If you prefer the normal LaTeX way, use `\oldbraces` to reset everything. This new behaviour should be extended to other characters like `| [ { <` and so on. Maybe in some later version.

The newbraces does *not* work at the moment!

I would have never been able to implement this without the help of the mailinglist members of `tex-d-l@listserv.dfn.de`!

Maybe one could "temporarily hardcode" the hight of [and then use this...

The redefinition of `\mathstrut` is necessary when using amsmath (you will use amsmath when typesetting formulae, won't you?), because the hight of formulae is determinated by the hight of a brace. But using ( ) as `\active` characters, we need another brace here. So we take `[`. This will probably also change. But the code is working fine for ( ).

```
\def\resetMathstrut@{%
    \setbox\z@\hbox{%
      \mathchardef\@tempa\mathcode`\[\relax
      \def\@tempb##1"##2##3{\the\textfont"##3\char"}%
      \expandafter\@tempb\meaning\@tempa \relax
}%
  \ht\Mathstrutbox@\ht\z@ \dp\Mathstrutbox@\dp\z@
}

{\catcode`(\active \xdef({\left\string(}}
{\catcode`)\active \xdef){\right\string)}}

\cs_new:Npn\newbraces{
  \char_set_mathcode:w`("8000
  \char_set_mathcode:w`)"8000
}

\cs_new:Npx\oldbraces{
  \char_set_mathcode:w`(\char_value_mathcode:w`(
  \char_set_mathcode:w`)\char_value_mathcode:w`)
}
```

3.2 huge display math

`hugedisplaymath` Sometimes, especially in presentations, you might need an really big formula. Imagine two hours of struggle with transformations—and finally there is the beautiful formula. Now you can say

```
\begin{hugedisplaymath} E = mc^2 \end{hugedisplaymath}
```

There should be several steps of size, maybe.

```
\cs_new:Npn\hugedisplaymath{
  \Huge
    \begin{equation*}
}
\cs_new:Npn\endhugedisplaymath{
  \end{equation*}
}
```

3.3 unicode math

This section will soon be given up as there is the great unicode-math package by Will Robertson which does all the math stuff much more elaborate than it is hacked here. However, I let it stand for a while as unicode-math is under development.

Typing math in TEX is no great fun – you have to write things like `\int` instead of ∫ and so on. Have a look at the following formula:

`\int_\infty^\infty \sum_a`

The code again is stolen and I don't understand, why it does what it does, but it does it: The first argument is the character you want to use for "unicode math", the second one is the TEX-command.

```
\ExplSyntaxOff
\def\altmath#1#2{%
```

```
  \expandafter\ifx\csname cc\string#1\endcsname\relax
    \add@special{#1}%
    \expandafter
    \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
    \begingroup
      \catcode`\~\active  \lccode`\~`#1%
      \lowercase{%
      \global\expandafter\let
         \csname ac\string#1\endcsname~%
      \expandafter\gdef\expandafter~\expandafter{#2}}%
    \endgroup
    \global\catcode`#1\active
  \else
  \fi
}
\ExplSyntaxOn
```

We will make a switch to turn this stuff on or off, so it does not interfere with the unicode-math package. This list will increase by time. If you are missing a symbol, just send me the `\altmath{X}{\Xcode}`-line. I would be very thankful if anybody could send me a whole list of symbols!

```
\cs_new:Npn\makealtmath{
  \altmath{α}\alpha
  \altmath{β}\beta
  \altmath{γ}\gamma
  \altmath{δ}\delta
  \altmath{ε}\varepsilon
  \altmath{χ}\chi
  \altmath{λ}\lambda
  \altmath{μ}\mu
  \altmath{π}\pi
  \altmath{σ}\sigma
  \altmath{ζ}\zeta

  \altmath{⇒}\Rightarrow
  \altmath{⇐}\Leftarrow
  \altmath{⇔}\Leftrightarrow

  \altmath{∫}\int
  \altmath{∀}\forall
  \altmath{∃}\exists
  \altmath{∞}\infinity

  \altmath{₁}{_1}
  \altmath{₂}{_2}
  \altmath{₃}{_3}
  \altmath{₄}{_4}
  \altmath{₅}{_5}
  \altmath{₆}{_6}
  \altmath{₇}{_7}
  \altmath{₈}{_8}
  \altmath{₉}{_9}
  \altmath{₀}{_0}
}
```

3.4 Lazy underscript and superscript

An underscore at the end of an inline-formula has to be ended with } or egroup. That is not nice...

The redefinition of hat does not work because TeX uses it for definition of catcodes. There has to be a really tricky way to get around that.

Sometimes one has to make extensive use of subscripts and superscripts, e. g. when typing long formulae including tensors. Then it is a bit annoying to always write the `{}`, especially when there are only two letters in the sub/superscript. So let's try to implement the possibility to type `$F_μν F^μν $`.

First, store the actual meaning of _ and ^ in `\oldunderscore` and `\oldhat`.

```
\cs_set_eq:NwN\oldunderscore_\tex_relax:D
\cs_set_eq:NwN\oldhat^\tex_relax:D
```

Now set _ as `\active` char and define it the way we want it to behave. For this, we need the space char and end-of-line char to be an egroup char. So the underscript group is ended by space or eol and we don't need to close it explicitly. (no expl3 code here, as we are dealing with the underscore. That would mess up everything ...)

```
\ExplSyntaxOff
\catcode`\_=13
\def_{%
  \ifmmode
    \catcode`\ =2\relax%
    \catcode`\^^M=2\relax%
    \expandafter\oldunderscore\bgroup%
  \else%
    \textunderscore%
  \fi%
}

\iffalse
This does not work so far…
\catcode`\^=13
\def^{%
  \ifmmode
    \catcode`\ =2\relax%
    \catcode`\^^M=2\relax%
    \expandafter\oldhat\bgroup%
  \else%
    \oldhat%
  \fi%
}
\fi
```

To give the possibility to swith between normal and `alttex` behaviour, store the new underscore.

```
\let\advancedunderscore_
```

The newUnder does not work so far.

And the switches. By default, _ is active. Type `\oldUnder` to get the normal _.

```
\def\oldUnder{
  \global\catcode`\_=8\relax
}
\def\newUnder{
  \global\let_\advancedunderscore
}
\ExplSyntaxOn
```

3.5 matrices

This is a nice idea by Alexander Koch on `diskussion@neo-layout.org`. Using the unicode glyphs for writing matrices, we can make writing and readig of big matrices much easier. (In Neo, one can use the compose function to write the whole matrix by 4–5 keystrokes and then fill in the elements.) For example, say in the source:

```
                          ⎧a & b⎫
⎛a & b⎞     ⎡a & b⎤       ⎪c & d⎪
⎜c & d⎟  or ⎢d & e⎥   or  ⎨e & f⎬
⎝e & f⎠     ⎣f & g⎦       ⎪g & h⎪
                          ⎩i & j⎭
```

and the result will be a `bmatrix`, a `pmatrix` or a `\right\{ matrix \end{matrix}`, respectively. As TEX is assumed to read from left-top to right-bottom, the matrices must not stand in a line, i. e. the following notation is *not* (yet?) possible:

```
    ⎛a & b⎞
A = ⎜c & d⎟ = B
    ⎝e & f⎠
```

but rather you have to write

```
A = ⎛a & b⎞
    ⎜c & d⎟
    ⎝e & f⎠ = B
```

If you have a suggestion how to enable the upper solution, please contact me, that would be an awesome thing!

One has to pay greatest attention to the different characters looking like ⎟ ⎥. They are in fact *different* for the three matrices! (But not in every case; I just hope the following code really works.)

```
\char_set_catcode:w`\⎛13
\char_set_catcode:w`\⎞13
\char_set_catcode:w`\⎟13
\char_set_catcode:w`\⎝13
\char_set_catcode:w`\⎠13
\cs_new:Npn⎛{\begin{pmatrix}}
\cs_new:Npn⎞{\\}
\cs_new:Npn⎝{}
\cs_new:Npn⎠{\end{pmatrix}}
\cs_new:Npn⎟{\\}

\catcode`\⎡13
\catcode`\⎤13
\catcode`\⎥13
\catcode`\⎣13
\catcode`\⎦13
\cs_new:Npn⎥{\\}
\cs_new:Npn⎡{\begin{bmatrix}}
\cs_new:Npn⎤{\\}
```

```
\cs_new:Npn⎣{}
\cs_new:Npn⎦{\end{bmatrix}}

\catcode`\⎧13
\catcode`\⎫13
\catcode`\⎪13
\catcode`\⎩13
\catcode`\⎭13
\catcode`\⎨13
\cs_new:Npn⎧{\left\{\begin{matrix}}
\cs_new:Npn⎫{\\\use_none:n}
\cs_new:Npn⎩{}
\cs_new:Npn⎭{\end{matrix}\right\}}
\cs_new:Npn⎪{\\\use_none:n}
\cs_new:Npn⎨{\\\use_none:n}
```

The codepoints have to be checked very carefully! This is not what a robust solution does look like!

We need to `\use_none:n` (ignore) the next character only in this case, as the left-hand bar characters seem to be the same as the right-hand and so cause additional line breaks. This way it is robust against every strange codepoint the left-hand may have.

4 Lists and such things

4.1 itemize with a single character

`• instead of \item`

Here we use an active character (mostly a unicode character bullet •) for the whole construct. And another one for nested itemizations (like a triangular bullet ‣).

This works quite fine for most LaTeX classes, but *not for beamer*! There, the end of `itemize` has to be given explicitly. For this, just say in the preable of your document, after loading this package: `\def\•{\end{itemize}}` and use the `\•` to end the itemization.

`\newitemi`
`\newitemii`

The following ugly peace of code is writen by me, defining the conditional insertion of the `\begin{itemize}`. This will be assigned to an active character using `\makeitemi` and `\makeitemii`, respectively.

```
\cs_new:Npn\outside{o}
\cs_new:Npn\inside{i}
\cs_set_eq:NwN\insideitemizei\outside
\cs_set_eq:NwN\insideitemizeii\outside
```

The end of itemizei and itemizeii:

```
\cs_new:Npn\altenditemize{
  \if\altlastitem 1%
    \let\altlastitem0%
  \else%
    \end{itemize}%
    \let\insideitemizei\outside%
  \fi%
}
```

Dealing with the ~ char, therefore no expl3 syntax here:

```
%
\ExplSyntaxOff
\begingroup
```

```
  \lccode`\~=`\^^M%
\lowercase{%
  \endgroup
  \def\makeenteractive{%
    \catcode`\^^M=\active
    \let~\altenditemize
}%
}
\ExplSyntaxOn

\cs_new:Npn\newitemi{%
  \if_meaning:w\insideitemizei\inside%
    \cs_set_eq:NN\altlastitem1%
    \exp_after:wN\item%
  \else:%
    \begin{itemize}%
    \cs_set_eq:NN\insideitemizei\inside%
    \cs_set_eq:NN\altlastitem1%
    \makeenteractive%
    \exp_after:wN\item%
  \fi:
}

\cs_new:Npn\newitemii{
  \if_meaning:w\insideitemizeii\inside
    \exp_after:wN\item%
  \else:
    \begin{itemize}
      \cs_set_eq:NN\insideitemizeii\inside
      \exp_after:wN\item%
  \fi:
}
```

Ok, the following code is stolen from the `shortvrb` package, and I don't understand anything of it. But I keep on trying… nevertheless, it's working fine, as far as I can see.

`\makeitemi` `\makeitemii` With this macro, you can define the character you want to use for first-level itemize. (Guess the sense of `\makeitemii`…) Default ist • for first-level and ‣ for second-level. Maybe this will be extended till fourth level. More doesn‘t seem to make any sense.

Again, no expl3 due to usage of ~ here. :(

```
\ExplSyntaxOff
\makeatletter
\def\makeitemi#1{%
  \expandafter\ifx\csname cc\string#1\endcsname\relax
    \add@special{#1}%
    \expandafter
    \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
    \begingroup
      \catcode`\~\active  \lccode`\~`#1%
      \lowercase{%
      \global\expandafter\let
         \csname ac\string#1\endcsname~%
      \expandafter\gdef\expandafter~\expandafter{\newitemi}}%
    \endgroup
    \global\catcode`#1\active
  \else
  \fi
}

\def\makeitemii#1{%
  \expandafter\ifx\csname cc\string#1\endcsname\relax
    \add@special{#1}%
    \expandafter
    \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
    \begingroup
      \catcode`\~\active  \lccode`\~`#1%
      \lowercase{%
      \global\expandafter\let
         \csname ac\string#1\endcsname~%
      \expandafter\gdef\expandafter~\expandafter{\newitemii}}%
    \endgroup
    \global\catcode`#1\active
  \else
  \fi
}
\ExplSyntaxOn
```

Now there are the two helperfunctions – no guess what they are really doing.

```
\cs_new:Npn\add@special#1{%
  \rem@special{#1}%
  \exp_after:wN\cs_gset_nopar:Npn\exp_after:wN\dospecials\exp_after:wN
{\dospecials \do #1}%
  \exp_after:wN\cs_gset_nopar:Npn\exp_after:wN\@sanitize\exp_after:wN
{\@sanitize \@makeother #1}}
\cs_new:Npn\rem@special#1{%
  \cs_set:Npn\do##1{%
    \if_num:w`#1=`##1 \else: \exp_not:N\do\exp_not:N##1\fi:}%
  \cs_set_nopar:Npx\dospecials{\dospecials}%
  \tex_begingroup:D
    \cs_set:Npn\@makeother##1{%
      \if_num:w`#1=`##1 \else: \exp_not:N\@makeother\exp_not:N##1\fi:}%
    \cs_set_nopar:Npx\@sanitize{\@sanitize}%
  \tex_endgroup:D}
```

4.2 enumerate with a single character

¹, ² And we do just the same stuff with `\enumerate`. But here we take the character ¹ as first level item, the ²[4] as second level etc. This may be confusing some way, but just try it.

For the implementation: copy-pasted the code above, nothing interesting so far.

```
\cs_new:Npn\¹{\end{enumerate}}
```

[4] Maybe this is a very stupid idea, because now the ² cannot be used as a square in mathmode. Of course there could be a test `ifmmode`, but I rather would like to find a better character for `enumerate`.

```
\cs_new:Npn\²{\end{enumerate}}

\cs_set_eq:NN\insideenumi\outside
\cs_set_eq:NN\insideenumii\outside

\cs_new:Npn\newenumi{
  \if_meaning:w\insideenumi\inside
    \exp_after:wN\item%
  \else:
    \begin{enumerate}
      \cs_set_eq:NN\insideenumi\inside
      \exp_after:wN\item%
  \fi:
}

\cs_new:Npn\newenumii{
  \if_meaning:w\insideenumii\inside
    \exp_after:wN\item%
  \else:
    \begin{enumerate}
      \cs_set_eq:NN\insideenumii\inside
      \exp_after:wN\item%
  \fi:
}
```

We use the same methods as above, still not understanding, what they are doing. Just changing two lines of code and hoping, everything will be fine. Again, no expl3.

```
\ExplSyntaxOff
\makeatletter
\def\makeenumi#1{%
  \expandafter\ifx\csname cc\string#1\endcsname\relax
    \add@special{#1}%
    \expandafter
    \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
    \begingroup
      \catcode`\~\active  \lccode`\~`#1%
      \lowercase{%
      \global\expandafter\let
         \csname ac\string#1\endcsname~%
      \expandafter\gdef\expandafter~\expandafter{\newenumi}}%
    \endgroup
    \global\catcode`#1\active
  \else
  \fi
}

\def\makeenumii#1{%
  \expandafter\ifx\csname cc\string#1\endcsname\relax
    \add@special{#1}%
    \expandafter
    \xdef\csname cc\string#1\endcsname{\the\catcode`#1}%
    \begingroup
      \catcode`\~\active  \lccode`\~`#1%
      \lowercase{%
      \global\expandafter\let
         \csname ac\string#1\endcsname~%
      \expandafter\gdef\expandafter~\expandafter{\newenumii}}%
    \endgroup
    \global\catcode`#1\active
  \else
  \fi
}
\makeatother
\ExplSyntaxOn
```

Finally, we set the default characters for the items and enumerations:

```
\makeitemi•
\makeitemii‣
\makeenumi¹
\makeenumii²
```

And that's it.

Happy altTeXing!

5 Known Bugs

This should be a list of serious bugs. Please report any of them to me!

- Itemize does not work correctly in beamer. Use `\•` at the end of your itemize. (see section 4.1)
- `\exceltabular` is broken.

todo

Here a section with some ideas that could be implemented.

- Change `\usepackage` only with a given package-option to make it more robust.
- Use `²` as square in mathmode and possibly `¹` as `\footnote`?
- Do something to enable easy tabular
- If there is only one char after an `_`, there should no space be needed.
- Maybe there could be a ConTeXt-version of this file.