The StarTEX source code

Dag Langmyhr
Department of Informatics
University of Oslo

dag@ifi.uio.no

Version 1.04, 11th March 1999

Contents

1 The StarTEX format

1 ⟨*code⟩

This file contains the documented source code of StarTEX, a simplified and more robust TEX format intended for students writing their first report or essay. For more information on StarTEX, see the other files accompanying this.

The main guidelines for this implementation are:

- StarTEX is implemented on top of plain TEX.

Value	State
`x`	Ordinary body text
`p`	Inside a `<psfig>` caption
`t`	Inside a `<table>` caption
`r`	Inside a `<table>` row

Table 1: The possible values of `\State`

- All names defined in the StarTeX implementation contain at least one uppercase letter, like `\Cdef` or `\NewEnvir`. This makes it easier to distinguish them from TeX's internal names and the names defined in plain TeX, nearly all of which consist of lowercase letters only.
- Most TeX macro packages tend to use a rather terse programming style, like

  ```
  \advance\var\@ne
  ```

  This improves processing speed and reduces storage, but make the code more difficult to read. Since this is a package which aims to be easily understandable and adaptable, I will use a more verbose style:

  ```
  \advance␣\var␣by␣1
  ```

To avoid possible confusion, StarTeX commands will be called *commands* in this document, while TeX commands will be called *macros*.

1.1 States

`\State` To avoid improper nesting, such as use of `<psfig>` within a `<table>`, we introduce a global state variable. Its values are given in table 1. The initial state is body text:

```
\let \State = x
```

1.2 Command handling

`\Command` StarTeX uses the syntax `<`*x*`>` for its commands rather than the usual `\`*x*. This is easily implemented by making `<` an active character calling the macro `\Command`. The command name is the text between the `<` and a subsequent `>`.

`\Command` first checks whether the command immediately follows a `<define>` in which case it is being defined; `\DefineCmd` handles that. Otherwise, `\Command` converts the command name into lowercase and checks whether it has been defined. If so, it is called; otherwise an error message is produced and the command is ignored.

```
\def \Command #1>{\ifDefining
    \def \Next {\lowercase{\DefineCmd{#1}}}%
  \else
    \lowercase{\expandafter\ifx \csname >#1\endcsname}\relax
      \Error{Unknown command <#1> ignored.}{}%
      \let \Next = \relax
    \else
      \def \Next {\lowercase{\csname >#1\endcsname}}%
```

```
    \fi
  \fi \Next }
```

The `<` character must be bound to `\Command`:

```
{\catcode `\< = \active  \global\let < = \Command }
```

1.2.1 Internal form of command names

All StarTeX commands are represented by a TeX macro whose name is constructed by `\csname`. To avoid confusion with predefined TeX macros, an initial `>`[1] is inserted; for example, the name of `<h1>` is represented by `\csname >h1\endcsname`.

`\Cdef` To simplify the declaration of StarTeX commands, the macro `\Cdef` is defined. It takes two parameters: `#1` is the command name (with no uppercase letters and no angle brackets), and `#2` is the command definition.

```
\def \Cdef #1#2{\expandafter\gdef \csname >#1\endcsname{#2}}
```

`\Ccall` In case we need to call StarTeX commands from TeX code, a `\Ccall` macro is introduced.

```
\def \Ccall #1{\csname >#1\endcsname}
```

1.2.2 User-defined commands

`<define>` `\ifDefining` The implementation of `<define>` is a little sneaky; it just sets a flag `\ifDefining`. When the following command name is found, the `\Command` routine checks the state of the flag and calls `\DefineCmd` when the flag is raised.[2]

```
\newif \ifDefining
\Cdef {define}{\Definingtrue}
```

`\DefineCmd` The command `\DefineCmd` is used to define a new user command; `#1` is the new command name (already translated into lowercase) and `#2` is the definition. The main job of `\DefineCmd` is to check whether the definition is legal. There are two reasons why it may not be legal:

- The definition may be nested, as in

  `<define><comm-1>...<define><comm-2>...`

  This is checked using a counter `\CallLevel` which is incremented whenever a user-defined command is called, and decremented on return; see the definition of `\Call` and `\Return` below.

- The command may already be defined.

If the definition is legal, `\FetchDef` is called to perform the actual defining.

```
\def \DefineCmd #1{\Definingfalse
  \ifnum \CallLevel>0
    \Error{Nested definitions are not allowed;}%
      {the definition of <#1> is ignored.}\let \Next = \relax
```

---

[1] The character `>` was chosen as it cannot possibly occur in a command name.

[2] This means that the user can insert text between the `<define>` and the command name, and this will be typeset as normal text. There is a slight chance that this might confuse some users, but I prefer this solution to using much more complicated code.

```
  \else
    \expandafter\ifx \csname>#1\endcsname\relax
      \def \Next {\begingroup \catcode‘\^^M = 12 \FetchDef{#1}}%
    \else
      \Error{Command <#1> already defined;}{this definition ignored.}%
      \def \Next {\begingroup \catcode‘\^^M = 12 \IgnoreDef{#1}}%
    \fi
  \fi \Next }
```

`\FetchDef`

`\FetchDef` defines the macro by using everything up to the end-of-line mark as the definition. This implies modifying the `\catcode` of the end-of-line character (`^^M`). Also note that calls on `\Call` and `\Return` are inserted into the definition.

```
\begingroup \catcode‘\^^M = 12
  \gdef \FetchDef #1#2^^M{\expandafter%
    \gdef\csname>#1\endcsname {\Call #2\Return }\endgroup }%
```

`\IgnoreDef`

I also define `\IgnoreDef` which is quite similar to `\FetchDef`, but no command is defined. It is used to ignore the rest of the line in case the user tries an illegal definition.

```
  \gdef \IgnoreDef #1#2^^M{\endgroup }%
\endgroup
```

`\CallLevel`

Finally, we must declare the user command level counter `\CallLevel`:

```
\newcount \CallLevel
```

`\Call`
`\Return`

We must also define the two commands used for incrementing and decrementing the counter:

```
\def \Call   {\global\advance \CallLevel by  1 }
\def \Return {\global\advance \CallLevel by -1 }
```

1.2.3 Catcode modifications

StarTeX uses only one special character: `<`, which must be active. All the other special characters of TeX and LaTeX are assigned suitable `\catcode`s turning them into ordinary characters.

`\SpecialCatCodes`

`\SpecialCatCodes` sets the `\catcode`s to the StarTeX values:

```
\def \SpecialCatCodes {%
  \catcode ‘\\ = 12  \catcode ‘\{ = 12  \catcode ‘\} = 12
  \catcode ‘\$ = 12  \catcode ‘\& = 12  \catcode ‘\# = 12
  \catcode ‘\^ = 12  \catcode ‘\_ = 12  \catcode ‘\~ = 12
  \catcode ‘\% = 12  \catcode ‘\< = \active }
```

`\StandardCatCodes`

`\StandardCatCodes` restores the `\catcode`s to their normal values. This is necessary when reading for instance style files.

```
\def \StandardCatCodes {%
  \catcode ‘\\ = 0  \catcode ‘\{ = 1  \catcode ‘\} = 2
  \catcode ‘\$ = 3  \catcode ‘\& = 4  \catcode ‘\# = 6
  \catcode ‘\^ = 7  \catcode ‘\_ = 8  \catcode ‘\~ = \active
  \catcode ‘\% = 14 \catcode ‘\< = 12 }
```

1.2.4 Environments

A StarTeX environment is a piece of text enclosed in a `<x>`...`</x>` pair.

`\NewEnvir`

`\NewEnvir` is used to start a new environment. It has three parameters: `#1` is the name of the environment, `#2` is the TeX commands used to start the new environment (usually just a `\begingroup`), and `#3` is the TeX commands used to terminate the environment (usually just a `\endgroup`).

`\PrevEnv`
`\PrevEnvLine`
`\CurEnv`
`\CurEnvExit`
`\CurEnvLine`

`\NewEnvir` saves information about the current environment's name (in `\CurEnv`), the line on which it starts (in `\CurEnvLine`) and which command is used to exit it (in `\CurEnvExit`). It also keeps tracks of the outer environment's name (in `\PrevEnv`) and start line (in `\PrevEnvLine`) for better error reporting and error recovery.

```
\def \NewEnvir #1#2#3{#2\relax
  \let \PrevEnv = \CurEnv  \PrevEnvLine = \CurEnvLine
  \def \CurEnv {#1}\def \CurEnvExit {#3}\CurEnvLine = \inputlineno }
```

We need default definitions of `\CurEnv` and `\CurEnvExit`:

```
\def \CurEnv {}\def \CurEnvExit {\relax}
```

We also need to declare the two line counters:

```
\newcount \CurEnvLine  \newcount \PrevEnvLine
```

`\EndEnvir`

`\EndEnvir` is used to terminate an environment. It check that the correct environment is terminated, and tries to correct user mistakes. It checks for the following situations:

- If the name of the environment to be terminated is the same as that of the current environment, everything is OK, and we can safely leave the current environment.

- If the name of the terminated environment is not the name of the current one, but matches the name of the outer environment, we assume that the user has forgotten a `</x>` command. An error message is given, and both the current environment and the outer one are terminated.

- If the name of the terminated environment matches neither the current nor the outer environment, we assume that the user has just misspelled the command. The best thing we can do in this case is to give an error message and ignore the command. If the user had intended to terminate the current environment, we get erroneous processing of the following text, but the situation will normalize when the outer environment is terminated.

```
\def \EndEnvir #1{%
  \ifTextEqual{#1}{\CurEnv}\let \Next = \CurEnvExit
  \else \EnvirError{#1}\fi
  \Next }
```

`\EnvirError`

`\EnvirError` is an auxiliary command giving a proper error message and placing—in `\Next`—the best command to recover from the error.

```
\def \EnvirError #1{\ifTextEqual{#1}{\PrevEnv}%
    \Error{<\CurEnv> on line \the\CurEnvLine\space terminated by
      </#1>.}{An extra </\CurEnv> has been inserted.}%
    \def \Next {\CurEnvExit \CurEnvExit }%
  \else
    \Error{<\CurEnv> on line \the\CurEnvLine\space terminated by
      </#1>.}{The </#1> will be ignored.}%
    \let \Next = \relax
  \fi }
```

1.3 Document styles

`<style>` The command `<style>` is called to read a document style. It calls `\ReadStyle` to do the actual work.

```
\Cdef {style}{\IfNextChar{[}{\ReadStyle}%
  {\Error{No style name given;}{the syntax is: <style>[style name].}}}
```

`\ReadStyle` `\StyleLine` `\ReadStyle` reads the style file (with extension `.stx`) using the standard TEX `\catcode`s. Afterwards, the `<style>` command is redefined, since `<style>` should be called once only. Also, the current line number is saved (in `\StyleLine`) in case it is needed later in an error message.

```
\def \ReadStyle [#1]{\IfFileExists{#1.stx}%
    {\edef \StyleLine {\the\inputlineno}%
     \Cdef {style}{\Error{Command <style> already used on line
                    \StyleLine;}{this use of <style> is ignored.}}%
     \StandardCatCodes \input #1.stx
     \SpecialCatCodes }%
    {\Error{Style file ‘#1.stx’ could not be found;}%
          {style definition ignored.}}}
```

1.4 Fonts

StarTEX allows the user a limited amount of font selection:

- use of **bold text** using `<b>`. . . `</b>`,
- use of *italic text* using `<i>`. . . `</i>`, and
- use of `typewriter text` using `<tt>`. . . `</tt>`.

All other font changes, including font size changes, are done by the document style.

Since the font change commands mentioned above may be combined, StarTEX uses an internal font naming scheme in which a font name consists of five parts, as shown in table 2 on the following page. For example, the font "Standard bold roman italic normal-size" is represented internally as `F-RSBIN`.

1.4.1 Font definitions

`\FontDef` The macro `\FontDef` is used to define a StarTEX font pair. To be more precise, it defines which TEX font specifications to use for each internal StarTEX font name. `#1` gives the font kind, `#2` the boldness, `#3` whether the font is italic or not, and `#4` the font size. Two fonts are defined for each internal StarTEX name: the standard Cork encoded font (in `#5`) and the additional companion font (in `#6`).

```
\def \FontDef #1#2#3#4#5#6{%
  \expandafter\def \csname F-#1S#2#3#4\endcsname {#5}%
  \expandafter\def \csname F-#1A#2#3#4\endcsname {#6}}
```

Kind	**Alternate**	**Bold**	**Italic**	**Size**
`R` (Roman) `T` (Teletype)	`S` (Standard) `A` (Alternate)	`M` (Medium) `B` (Bold)	`U` (Upright) `I` (Italic)	`X` (Extra large) `L` (Large) `N` (Normal) `S` (Small)

Table 2: The five parts of the internal StarTeX text font name

Size	**Position**	**Family**
`X` (Extra large) `L` (Large) `N` (Normal) `S` (Small)	`N` (Normal) `S` (Script) `X` (Scriptscript)	`0` (Roman math) `1` (Italic math) `2` (Math symbols) `3` (Large math symbols) `4` (Bold math) `5` (Extended italic math) `6` (AMS symbols B)

Table 3: The three parts of the internal StarTeX math font name

`\MathFontDef`

Similarly, `\MathFontDef` is used to define a math font. `#1` is the size, `#2` is the family, and `#3`–`#5` are the three fonts to be used as math text font, script font and scriptscript font.

```
\def \MathFontDef #1#2#3#4#5{%
  \expandafter\def \csname M-#1N#2\endcsname {#3}%
  \expandafter\def \csname M-#1S#2\endcsname {#4}%
  \expandafter\def \csname M-#1X#2\endcsname {#5}}
```

1.4.2 Predefined fonts

The following fonts are predefined, which means that they are the fonts the user will get unless the document style dictates otherwise.

The fonts used are the so-called *EC fonts*. Since there are no bold typewriter characters in this set, ordinary typewriter characters are used instead.

Also defined are various size dependant values:

`\LineSkip`*x* line skip, and

`\CodeSkip`*x* line skip in code listings.

Normal size fonts An 11 point (actually 10.95 pt) font is the standard text font.

```
\def \XIpt { at 10.95pt}
\FontDef{R}{M}{U}{N}{ecrm1095}{tcrm1095}
\FontDef{R}{M}{I}{N}{ecti1095}{tcti1095}
\FontDef{R}{B}{U}{N}{ecbx1095}{tcbx1095}
\FontDef{R}{B}{I}{N}{ecbi1095}{tcbi1095}
\FontDef{T}{M}{U}{N}{ectt1095}{tctt1095}
\FontDef{T}{M}{I}{N}{ecit1095}{tcit1095}
\FontDef{T}{B}{U}{N}{ectt1095}{tctt1095}
```

```
\FontDef{T}{B}{I}{N}{ecit1095}{tcit1095}
\MathFontDef{N}{0}{cmr10\XIpt}{cmr8}{cmr6}
\MathFontDef{N}{1}{cmmi10\XIpt}{cmmi8}{cmmi6}
\MathFontDef{N}{2}{cmsy10\XIpt}{cmsy8}{cmsy6}
\MathFontDef{N}{3}{cmex10\XIpt}{cmex10\XIpt}{cmex10\XIpt}
\MathFontDef{N}{4}{cmbx10\XIpt}{cmbx8}{cmbx6}
\MathFontDef{N}{5}{ecti1095}{ecti0800}{ecti0600}
\MathFontDef{N}{6}{msbm10\XIpt}{msbm8}{msbm6}
\def \LineSkipN {12pt}  \def \CodeSkipN {11pt plus 0.0pt minus 0.1pt}
```

Large fonts The slightly larger characters are 14.4 pt.

```
\def \XIVpt { at 14.4pt}
\FontDef{R}{M}{U}{L}{ecrm1440}{tcrm1440}
\FontDef{R}{M}{I}{L}{ecti1440}{tcti1440}
\FontDef{R}{B}{U}{L}{ecbx1440}{tcbx1440}
\FontDef{R}{B}{I}{L}{ecbi1440}{tcbi1440}
\FontDef{T}{M}{U}{L}{ectt1440}{tctt1440}
\FontDef{T}{M}{I}{L}{ecit1440}{tcit1440}
\FontDef{T}{B}{U}{L}{ectt1440}{tctt1440}
\FontDef{T}{B}{I}{L}{ecit1440}{tcit1440}
\MathFontDef{L}{0}{cmr12\XIVpt}{cmr10}{cmr8}
\MathFontDef{L}{1}{cmmi12\XIVpt}{cmmi10}{cmmi8}
\MathFontDef{L}{2}{cmsy10\XIVpt}{cmsy10}{cmsy8}
\MathFontDef{L}{3}{cmex10\XIVpt}{cmex10\XIVpt}{cmex10\XIVpt}
\MathFontDef{L}{4}{cmbx12\XIVpt}{cmbx10}{cmbx8}
\MathFontDef{L}{5}{ecti1440}{ecti1000}{ecti0800}
\MathFontDef{L}{6}{msbm10\XIVpt}{msbm10}{msbm8}
\def \LineSkipL {15pt}
```

The extra large fonts The largest fonts are 17.28 pt.

```
\def \XVIIpt { at 17.28pt}
\FontDef{R}{M}{U}{X}{ecrm1728}{tcrm1728}
\FontDef{R}{M}{I}{X}{ecti1728}{tcti1728}
\FontDef{R}{B}{U}{X}{ecbx1728}{tcbx1728}
\FontDef{R}{B}{I}{X}{ecbi1728}{tcbi1728}
\FontDef{T}{M}{U}{X}{ectt1728}{tctt1728}
\FontDef{T}{M}{I}{X}{ecit1728}{tcit1728}
\FontDef{T}{B}{U}{X}{ectt1728}{tctt1728}
\FontDef{T}{B}{I}{X}{ecit1728}{tcit1728}
\MathFontDef{X}{0}{cmr17}{cmr12\XIVpt}{cmr9}
\MathFontDef{X}{1}{cmmi12\XVIIpt}{cmmi12}{cmmi9}
\MathFontDef{X}{2}{cmsy10\XVIIpt}{cmsy10 at 12pt}{cmsy9}
\MathFontDef{X}{3}{cmex10\XVIIpt}{cmex10\XVIIpt}{cmex10\XVIIpt}
\MathFontDef{X}{4}{cmbx12\XVIIpt}{cmbx12}{cmbx9}
\MathFontDef{X}{5}{ecti1728}{ecti1200}{ecti0900}
\MathFontDef{X}{6}{msbm10\XVIIpt}{msbm10 at 12pt}{msbm9}
\def \LineSkipX {17pt}
```

Small fonts A smaller font for footnotes and other special text is 10 pt.

```
\FontDef{R}{M}{U}{S}{ecrm1000}{tcrm1000}
\FontDef{R}{M}{I}{S}{ecti1000}{tcti1000}
\FontDef{R}{B}{U}{S}{ecbx1000}{tcbx1000}
```

```
\FontDef{R}{B}{I}{S}{ecbi1000}{tcbi1000}
\FontDef{T}{M}{U}{S}{ectt1000}{tctt1000}
\FontDef{T}{M}{I}{S}{ecit1000}{tcit1000}
\FontDef{T}{B}{U}{S}{ectt1000}{tctt1000}
\FontDef{T}{B}{I}{S}{ecit1000}{tcit1000}
\MathFontDef{S}{0}{cmr10}{cmr7}{cmr5}
\MathFontDef{S}{1}{cmmi10}{cmmi7}{cmmi5}
\MathFontDef{S}{2}{cmsy10}{cmsy7}{cmsy5}
\MathFontDef{S}{3}{cmex10}{cmex10}{cmex10}
\MathFontDef{S}{4}{cmbx10}{cmbx7}{cmbx5}
\MathFontDef{S}{5}{ecti1000}{ecti0700}{ecti0500}
\MathFontDef{S}{6}{msbm10}{msbm7}{msbm5}
\def \LineSkipS {11pt}
```

1.4.3 Font selection

`\FontAlt`
`\FontBold`
`\FontItal`
`\FontKind`
`\FontSize`

Selecting a StarTeX font is a two-step procedure:

1. Define `\FontKind`, `\FontAlt`, `\FontBold`, `\FontItal` and `\FontSize` to contain the correct letters, as shown in table 2 on page 8.
2. Call `\SelectFont`.

(The reason for this procedure is to avoid loading unused fonts.)

`\SelectFont`

As mentioned, `\SelectFont` selects a font according to the specifications given.

```
\def \SelectFont {%
  \edef \ThisFont {F-\FontKind\FontAlt\FontBold\FontItal\FontSize}%
  \font \CurFont = \csname\ThisFont\endcsname \CurFont }
```

`\ResetFont`

`\ResetFont` provides standard settings for the various StarTeX font parameters.

```
\def \ResetFont {%
  \def \FontKind{R}\def \FontAlt{S}\def \FontBold{M}%
  \def \FontItal{U}\def \FontSize{N}}
```

`\CheckItCorr`

`\CheckItCorr` is called after every group of italic text. Unless the following character is a low character like '`.`' or '`,`', an italic correction command `\/` is inserted.

```
\def \CheckItCorr {%
  \IfNextCharTwo{.}{,}{}{\/}}
```

1.4.4 Font size selection

`\SetSize`

`\SetSize` is used to set various parameters connected with a size change. It also contains a call on `\SelectFont`.

```
\def \SetSize #1{\def \FontSize{#1}\SelectFont
  \baselineskip = \csname LineSkip#1\endcsname
  \ParIndent = \StdParIndent }
```

1.4.5 StarTeX commands for font changes

`<b>`
`</b>`
`<i>`
`</i>`
`<tt>`
`</tt>`

The following commands are used to indicate bold, italic, or typewriter text.

```
\Cdef {b}{\NewEnvir{b}{\begingroup}{\endgroup}%
  \def\FontBold{B}\SelectFont}
```

```
\Cdef {/b}{\EndEnvir{b}}
\Cdef {i}{\NewEnvir{i}{\begingroup}{\endgroup\CheckItCorr}%
  \def\FontItal{I}\SelectFont}
\Cdef {/i}{\EndEnvir{i}}
\Cdef {tt}{\NewEnvir{tt}{\begingroup}{\endgroup}%
  \def\FontKind{T}\SelectFont}
\Cdef {/tt}{\EndEnvir{tt}}
```

1.5 Body text

A StarTeX file is enclosed in a `<body>`... `</body>` environment. The main reason for this is to be able to detect the end of the document and terminate properly.[3] It is also used to print an introductory message.

`<body>` `<body>` just redefines itself to produce an error message if used a second time; this is to ensure that it is only used once.

```
\Cdef {body}{\NewEnvir{body}{\begingroup}{\endgroup}%
  \message{^^JThis is StarTeX, version \CodeVersion^^J}%
  \global\everypar = {\NewPar }%
  \xdef \BodyLine {\the\inputlineno}%
  \Cdef {body}{\Error{Command <body> already used on line \BodyLine;}%
    {this use of <body> was ignored.}}}
```

`\BodyError` If the user has forgotten the `<body>`... `</body>` environment, the first text paragraph will trigger the `\BodyError` macro.

```
\def \BodyError {\message{^^JThis is StarTeX, version \CodeVersion^^J}%
  \Error{A <body>...</body> environment should surround
    the whole document.}{A missing <body> was inserted.}%
  \global\everypar = {\NewPar }}
```

`</body>` `</body>` is used to terminate the document. This involves two things:

1. the current page must be terminated, and
2. the `.aux` file must be check to determine whether any references have changed. The `.aux` file must be checked if both
   - an `.aux` file has been read (`\ifAuxRead` is true), and
   - a new `.aux` file has been created (`\ifAuxOpen` is true).

   If we already know that a rerun is necessary (because `\ifRerun` is true), there is no need to check the `.aux` file.

```
\Cdef {/body}{\EndEnvir{body}\endgraf\vfill\supereject
  \let \Next = \CheckAux
  \ifRerun \let \Next = \relax \fi
  \ifAuxRead \else \let \Next = \relax \fi
  \ifAuxOpen \else \let \Next = \relax \fi
  \Next
```

---

[3] An `\everyeof` command—had it existed—would have made the `<body>`... `</body>` environment superfluous, thus removing yet another possible error source. (ε-TeX[4] does provide such a primitive, but I have chosen not to apply it as use of ε-TeX is not yet very widespread.)

Give a warning if rerunning StarTEX is necessary.

```
\ifRerun \Warning{Cross-references are not correct;}%
  {please run StarTeX again.}\fi
```

And now, we can finally terminate the run.

```
\end}
```

1.5.1 Paragraph breaking

`<p>` The StarTEX command `<p>` is used to separate paragraphs:

```
\Cdef {p}{\endgraf}
```

Blank lines are just ignored:

```
\def \par {}
```

StarTEX uses parameters which allow rather sloppy paragraph setting:

```
\pretolerance = 2500 \tolerance = 9999  \hbadness = 10000
\emergencystretch = 3cm
```

The reasons for this are:

- If a good set of breaks is possible, TEX will still choose that.
- If no good breaks are possible, it is better to have a sloppily set paragraph than problem lines sticking into the margin. The potential users of StarTEX tend to just ignore messages about overfull boxes.

Note, however, that `\hfuzz` has not been increased, as such a change is immediately visible and gives a poor impression.

1.5.2 Paragraph indentation

The paragraph handling mechanism must be able to let some paragraphs be indented and others not; for instance, the first paragraph following a title should not be indented, while the subsequent ones should be.

In StarTEX this is solved by disabling the original `\parindent` and instead insert a call on `\NewPar` at the start of every paragraph. However, `\everypar` is initialized to `\BodyError` to detect a missing `<body>`... `</body>` environment; the `<body>` command will set `\everypar` correctly.

```
\parindent = 0pt \everypar = {\BodyError }
```

`\NewPar` This `\NewPar` inserts an indentation `\ParIndent` wide if `\ifIndent` is true.[4] It also sets `\ifIndent` to true, ensuring the indentation of subsequent paragraphs. `\NewPar` sets `\parskip` to its standard current value `\CurParSkip`.

```
\def \NewPar {\ifIndent \kern \ParIndent \fi  \Indenttrue
  \global\parskip = \CurParSkip }
```

(In some environments, like `<list>`... `</list>`, no paragraphs are indented. By setting `\ParIndent` to 0 pt, the whole paragraph indentation mechanism is disabled.)

---

[4] We cannot use the standard `\parindent` to this, as that is inserted when the paragraph is started, in other words before `\NewPar` is called.

`\ParIndent` `\ifIndent` `\StdParIndent`

We must declare `\ParIndent` and `\ifIndent`, and also `\StdParIndent` which holds the value to which `\ParIndent` is set when it is not 0:

```
\newskip \ParIndent  \def \StdParIndent {1em}
\newif \ifIndent
```

1.5.3 Paragraph separation

Sometimes it is necessary to omit the paragraph separation (the vertical space inserted between every paragraph), for instance after having inserted some special vertical space. This is done by setting `\parskip` to 0 pt. The following paragraph will reset this to `\CurParSkip` so that subsequent paragraphs will have the standard separation; see the definition of `\NewPar` above.

`\CurParSkip`

`\CurParSkip` must be defined:

```
\newskip \CurParSkip
```

`\AddVspace`

Several insertions of vertical space (with `\vskip`) should not just accumulate; this will lead to excessive vertical spacing. Instead, `\AddVspace` should be used; it inserts the maximum of the last space (if any) and the current one, and it also sets `\parskip` to 0 pt so that the automatic paragraph separation will not interfere.

```
\def \AddVspace #1{\ifvmode \else \endgraf \fi
  \skip1 = #1\relax
  \ifdim \lastskip < \skip1 \relax
    \ifdim \lastskip > 0pt \vskip -\lastskip \fi
    \vskip \skip1
  \fi \parskip = 0pt \relax }
```

1.5.4 The document head

This section implements the various commands used at the start of the document.

`<title>` `</title>`

`<title>`...`</title>` is used to set the document title. It is centered, and uses the biggest fonts available.

```
\Cdef {title}{\AddVspace{30pt plus 10pt}
  \NewEnvir{title}{\begingroup}{\endgraf\endgroup}
  \leftskip = 2cm plus 1fill  \rightskip = \leftskip
  \ParIndent = 0pt  \CurParSkip = 0pt
  \ResetFont \SetSize{X}}
\Cdef {/title}{\EndEnvir{title}%
  \AddVspace{20pt plus 4pt}}
```

`<author>` `</author>`

`<author>`...`</author>` is used for the author's name. It is set like `<title>`... `</title>`, but uses a smaller font.

```
\Cdef {author}{\AddVspace{10pt plus 3pt}
  \NewEnvir{author}{\begingroup}{\endgraf\endgroup}
  \leftskip = 2cm plus 1fill  \rightskip = \leftskip
  \ParIndent = 0pt  \CurParSkip = 0pt
  \ResetFont \SetSize{L}}
\Cdef {/author}{\EndEnvir{author}%
  \AddVspace{20pt plus 4pt}}
```

`<info>` `</info>` `<info>`...`</info>` is used for additional information, like the date. It is centered, but uses the normal text font.

```
\Cdef {info}{\AddVspace{10pt plus 3pt}
  \NewEnvir{info}{\begingroup}{\endgraf\endgroup}
  \leftskip = 2cm plus 1fill  \rightskip = \leftskip
  \ParIndent = 0pt  \CurParSkip = 0pt
  \ResetFont \SetSize{N}}
\Cdef {/info}{\EndEnvir{info}%
  \AddVspace{20pt plus 4pt}}
```

`<abstract>` `</abstract>` `<abstract>`...`</abstract>` is used for the short abstract usually given with each document. It is set with indented margins, and the type is smaller than the text type. The word 'Abstract' is set in boldface.

```
\Cdef {abstract}{\AddVspace{10pt plus 3pt}
  \NewEnvir{abstract}{\begingroup}{\endgraf\endgroup}
  \ResetFont \def \FontBold{B} \SetSize{S}  \CurParSkip = 0pt
  \leftskip = 2cm  \rightskip = \leftskip  \Indentfalse
  \centerline{\AbstractName}
  \ResetFont \SetSize{S}\Indentfalse }
\Cdef {/abstract}{\EndEnvir{abstract}%
  \AddVspace{10pt plus 2pt}}
```

1.5.5 Sectioning commands

To keep track of the section numbers, we need four counters:

```
\newcount \SectI    \newcount \SectII
\newcount \SectIII  \newcount \SectIV
```

`\Heading` Since most of the work of a sectioning command is the same irrespective of its level, the common code has been place in `\Heading` which has four parameters: `#1` is the vertical space before the section, `#2` is the font size (`N`, `L` or `X`), `#3` tells whether to use bold type or not (`B` or `M`), and `#4` is the TEX command to produce the section number.

```
\def \Heading #1#2#3#4{\AddVspace{#1}
  \def \FontBold {#3} \SetSize{#2}
  \setbox0 = \hbox{#4\kern 0.5\baselineskip}
  \hangindent = \wd0  \hangafter = 1  \raggedright
  \ParIndent = 0pt  \CurParSkip = 0pt  \leavevmode  \box0 }
```

`<h1>` `<H1>` must update the counters and add vertical space. The rest of the work is done by `\Heading`.

```
\Cdef {h1}{\endgraf
  \global\advance \SectI by 1
  \global\SectII = 0  \global\SectIII = 0  \global\SectIV = 0
  \edef \CurDef {\SectIim}
  \NewEnvir{h1}{\begingroup}{\endgroup}
  \Heading{24pt plus 12pt minus 3pt}{X}{B}{\SectIim}}
```

`</h1>` `</H1>` terminates the heading. It adds some vertical space (protcted with a top `\penalty` so that a page break will never occur) and sets `\Indentfalse` to prevent indentation of the following line.

```
\Cdef {/h1}{\endgraf
```

```
  \nobreak\vskip 6pt plus 1.5pt
  \EndEnvir{h1}  \Indentfalse }
```

`<h2>` `</h2>` <H2> and `</h2>` work more or less like `<h1>` and `</h1>`:

```
\Cdef {h2}{\endgraf
  \global\advance \SectII by 1
  \global\SectIII = 0  \global\SectIV = 0
  \edef \CurDef {\SectIIim}
  \NewEnvir{h2}{\begingroup}{\endgroup}
  \Heading{14pt plus 7pt minus 2pt}{L}{B}{\SectIIim}}
\Cdef {/h2}{\endgraf
  \nobreak\vskip 4pt plus 1pt
  \EndEnvir{h2}\Indentfalse }
```

`<h3>` `</h3>` <H3> and `</h3>` work more or less like the two preceding ones:

```
\Cdef {h3}{\endgraf
  \global\advance \SectIII by 1  \global\SectIV = 0
  \edef \CurDef {\SectIIIim}
  \NewEnvir{h3}{\begingroup}{\endgroup}
  \Heading{10pt plus 5pt minus 1pt}{N}{B}{\SectIIIim}}
\Cdef {/h3}{\endgraf
  \nobreak\vskip 2pt plus 1pt
  \EndEnvir{h3}\Indentfalse }
```

`<h4>` `</h4>` <H4> and `</h4>` are slightly different from the previous ones in that they use a run-in title (the text continues on the same line as the title):

```
\Cdef {h4}{\AddVspace{8pt plus 4pt minus 1pt}
  \global\advance \SectIV by 1
  \edef \CurDef {\SectIVim}
  \NewEnvir{h4}{\begingroup}{\endgroup}
  \def \FontBold{B}\SelectFont \Indentfalse \SectIVim }
\Cdef {/h4}{\kern 0.5\baselineskip
  \EndEnvir{h4}}
```

1.5.6 Lists

`<list>` `</list>` The `<list>`...`</list>` environment increases the left margin and sets the item counter to 0; everything else is handled by the various kinds of `\item`.

```
\Cdef {list}{\AddVspace{\ListSkip}
  \NewEnvir{list}{\begingroup}{\endgroup}
  \advance \leftskip by \ListIndent
  \Indentfalse  \CurParSkip = \ListSkip  \ParIndent = 0pt
  \ItemCount = 0 }
\Cdef {/list}{\endgraf
  \EndEnvir{list}  \Indentfalse }
```

`<item>` `<item>` inserts a bulleted item.

```
\Cdef {item}{\endgraf  \Indentfalse
  \leavevmode\llap{\BulletItemFormat}\ignorespaces }
```

`<numitem>` `<numitem>` gives a numbered item.

```
\Cdef {numitem}{\endgraf
  \advance \ItemCount by 1  \edef \CurDef {\the\ItemCount}
  \Indentfalse
  \leavevmode\llap{\NumItemFormat{\ItemCount}}\ignorespaces }
```

`\ItemCount`

The item counter `\ItemCount` must also be declared:

```
\newcount \ItemCount
```

`<textitem>` `</textitem>`

The `<textitem>`...`</textitem>` environment starts another item with a bold text.

```
\Cdef {textitem}{\endgraf
  \NewEnvir{textitem}{\begingroup}{\endgroup}
  \def \FontBold {B}\SelectFont \leavevmode \kern -\ListIndent
  \ignorespaces }
\Cdef {/textitem}{\unskip
  \EndEnvir{textitem}%
  \hskip 1em  \ignorespaces }
```

1.5.7 Displays

`<display>` `</display>`

A `<display>`... `</display>` environment is implemented by increasing both margins.

```
\Cdef {display}{\AddVspace{\DisplayPreSkip}
  \NewEnvir{display}{\begingroup}{\endgroup}%
  \advance \leftskip  by \DisplayIndent
  \advance \rightskip by \DisplayIndent
  \CurParSkip = \DisplayParSkip \ParIndent = 0pt \relax }
\Cdef {/display}{\AddVspace{\DisplayPostSkip}
  \EndEnvir{display}}
```

1.5.8 Code

Handling code like computer programs is quite easy. It involves

- changing <'s `\catcode` into that of an ordinary character,
- making space and end-of-lines into active characters with a suitable definition, and
- selecting a fixed-width font and a suitable leading.

It is necessary to remember whether the code followed a `<p>`. If so, some vertical space will be added before and after the code.

```
\newif \ifCodePar
```

`\CodeSetup`

All the initialization is handled by the `\CodeSetup` macro:

```
\def \CodeSetup {\ifvmode \CodePartrue \AddVspace{\DisplayPreSkip}
     \else \CodeParfalse \fi
  \def\FontKind{T}\SelectFont
  \baselineskip = \CodeSkipN  \CurParSkip = 0pt  \ParIndent = 0pt
  \catcode`\< = 12  \frenchspacing   \obeylines  \obeyspaces }
```

`\obeylines`

Plain TEX's definition of `\obeylines` must be redefined since `\par` has been changed:

```
\begingroup
```

```
  \catcode‘\^^M = \active %
  \gdef\obeylines{\catcode‘\^^M=\active \def^^M{\endgraf\leavevmode}}%
\endgroup
```

`\obeyspaces` The appearance of a space when `\obeyspaces` is in effect must also be modified. By inserting a `\leavevmode`, we will not lose spaces at the beginning of a line.

```
\begingroup
  \obeyspaces\gdef {\leavevmode\space}%
\endgroup
```

`\CodeFinish` After the code, `\CodeFinish` is called to terminate the mode. It adds the needed vertical space.

```
\def \CodeFinish {\ifCodePar
    \def \Next {\endgraf  \vskip -\baselineskip  \vskip \DisplayPostSkip
      \global\Indentfalse }%
  \else
    \let \Next = \relax
  \fi \Next }
```

`<code>` Time has come to write the user interface. The `<code>`...`</code>` environment reads the code as parameter to the command `\ReadCode`.

```
\Cdef {code}{\NewEnvir{code}{\begingroup}{\endgroup}%
  \CodeSetup \ReadCode }
```

`\ReadCode` `</code>` `\ReadCode` just inserts the code (which is parameter `#1`) and terminates the environment.

```
\begingroup
  \catcode ‘\< = 12
  \gdef \ReadCode #1</code>{#1\CodeFinish\EndEnvir{code}}%
\endgroup
```

`<codefile>` `<codefile>` checks for correct syntax. If the file name is given properly, `\ReadCodeFile` is called to perform the actual reading.

```
\Cdef {codefile}{\IfNextChar{[}%
    {\ReadCodeFile}%
    {\Error{No code file name given;}%
      {the syntax is <codefile>[file name].}}}
```

`\ReadCodeFile` `\ReadCodeFile` just reads the specified file

```
\def \ReadCodeFile [#1]{\endgraf \begingroup
  \CodeSetup
  \IfFileExists{#1}{\input #1}%
    {\Error{Code file ‘#1’ cound not be found.}{}}
  \CodeFinish \endgroup }
```

1.5.9 Footnotes

`<footnote>` `</footnote>` Implementing footnotes is easy, as the mechanism is ready in plain TEX.

```
\Cdef {footnote}{\NewEnvir{footnote}%
    {\global\advance \FootnoteCnt by 1
    \footnote{\FootnoteIm{\FootnoteCnt}}\bgroup
      \edef \CurDef {\the\FootnoteCnt}}%
    {\egroup}%
  \ResetFont \SetSize{S}}
\Cdef {/footnote}{\EndEnvir{footnote}}
```

`\FootnoteCnt`

We must remember to declare the footnote counter:

```
\newcount \FootnoteCnt
```

1.5.10 Page headers and footers

By default, StarTeX has the same headers and footer as TeX. The footer must, however, be redefined so that it always uses the same font as the body text.

```
\footline = {\ResetFont\SelectFont \hfil \folio \hfil}
```

1.6 Customization

This section contains definitions that are likely to be changed in local adaptions.

1.6.1 Sizes

`\DisplayIndent`
`\DisplayParSkip`
`\DisplayPostSkip`
`\DisplayPreskip`

The following four sizes determine the appearance of displays.

```
\def \DisplayIndent    {\ListIndent}
\def \DisplayParSkip   {\ListParSkip}
\def \DisplayPostSkip  {5pt plus 2pt minus 1pt\relax}
\def \DisplayPreSkip   {5pt plus 2pt minus 1pt\relax}
```

`\ListIndent`
`\ListParSkip`
`\ListSkip`

The next three are used when setting lists.

```
\def \ListIndent       {25pt\relax}
\def \ListParSkip      {\ListSkip}
\def \ListSkip         {10pt plus 2pt minus 1pt\relax}
```

1.6.2 Formats

`\SectIim`
`\SectIIim`
`\SectIIIim`
`\SectIVim`

`\SetcIim`, `\SectIIim`, `\SectIIIim` and `\SectIVim` define what the section number at various levels should look like.

```
\def \SectIim {\the\SectI}
\def \SectIIim {\SectIim.\the\SectII}
\def \SectIIIim {\SectIIim.\the\SectIII}
\def \SectIVim {\SectIIIim.\the\SectIV}
```

`\FootnoteIm`

`\FootnoteIm` defines the appearance of the footnote mark; `#1` is the footnote number.

```
\def \FootnoteIm #1{$^{\the #1}$}
```

`\FigIm`
`\TabIm`

`\FigIm` and `\TabIm` define the appearance of the figure and table counters, respectively.

```
\def \FigIm {\the\FigCnt}
\def \TabIm {\the\TabCnt}
```

`\BulletItemFormat`
`\NumItemFormat`

The two commands `\BulletItemFormat` and `\NumItemFormat` define the appearance of the marks used by `<item>` and `<numitem>`, respectively.

```
\def \BulletItemFormat {$\bullet$\kern 6pt\relax}
\def \NumItemFormat  #1{\the#1.\kern 4pt\relax}
```

1.6.3 Character set

This version of StarTeX uses ISO 8859-1 as its standard character set. (Users of other character sets will have to modify this.)

`\E` `\E` is used to redefine the encoding of a character; this is done by making the character active. `#1` is the character's number, and `#2` is the TeX code to which it is to be defined. The implementation of `\E` uses the standard technique of defining a `\uccode` and then calling `\uppercase` to insert the correct character where it is defined.)

```
\def \E #1#2{\catcode#1 = \active
  \begingroup \uccode`\~ = #1\uppercase{\endgroup \def ~{#2}}}
```

There are five different kinds of reencoding:

- Some characters are reencoded to a different character in the same font using the `\char` primitive; examples are 161 (¡) and 163 (£).

```
\E{161}{\char189 } \E{163}{\char191 } \E{167}{\char159 }
\E{171}{\char19 }  \E{184}{\char11 }  \E{187}{\char20 }
\E{191}{\char190 } \E{223}{\char255 } \E{255}{\char184 }
```

`\CC`

- Other characters can be found in the *Companion font*, known to StarTeX as the Alternate font. The macro `\CC` is used for this; its parameter `#1` is the character's position in the companion font. Examples are 162 (¢) and 164 (¤).

```
\def \CC #1{{\def\FontAlt{A}\SelectFont \char#1}}
\E{162}{\CC{162}} \E{164}{\CC{164}} \E{165}{\CC{165}}
\E{166}{\CC{166}} \E{168}{\CC{168}} \E{169}{\CC{169}}
\E{170}{\CC{170}} \E{172}{\CC{172}} \E{174}{\CC{174}}
\E{175}{\CC{175}} \E{176}{\CC{176}} \E{180}{\CC{180}}
\E{182}{\CC{182}} \E{186}{\CC{186}}
```

`\MC`

- A few characters are math characters, like 185 (^{1}) and 246 ($\div$). They are defined using the macro `\MC` which has two parameters: `#1` is the character's position in the companion font to be used in text mode, and `#2` is some TeX code to be used in math mode.

```
\def \MC #1#2{\ifmmode #2 \else \CC{#1}\fi }
\E{177}{\MC{177}{\pm}}        \E{178}{\MC{178}{^2{}}}
\E{179}{\MC{179}{^3{}}}       \E{181}{\MC{181}{\mu}}
\E{183}{\MC{183}{\cdot}}      \E{185}{\MC{185}{^1{}}}
\E{188}{\MC{188}{{1\over4}}} \E{189}{\MC{189}{{1\over2}}}
\E{190}{\MC{190}{{3\over4}}} \E{215}{\MC{214}{\times}}
\E{247}{\MC{246}{\div}}
```

- Character 160 is a "non-break space" which is treated just like a ~ in TeX.

```
\E{160}{\nobreak\ }
```

- Character 173 is a "soft hyphen" which is treated like `\-` in TeX.

```
\E{173}{\-}
```

1.6.4 Language-specific definitions

`\Abstractname` `\Figurename` `\Tablename` These three names vary from one language to another.

```
\def \AbstractName {Abstract}
\def \FigureName   {Figure}
\def \TableName    {Table}
```

`\TimeSep` Not all languages use a “:” to separate the hours, minutes, and seconds.

```
\def \TimeSep      {:}
```

`<today>` `\Month` `\Th` The `<today>` command should be redefined for each language. To simplify the code, two auxiliary macros are defined: `\Month` gives the name of the month, and `\Th` gives the correct suffix for the day of the month.

```
\Cdef {today}{\the\day\Th{\day} \Month\space\the\year}
\def \Month {\ifcase \month \or January\or February\or March\or
  April\or May\or June\or July\or August\or September\or
  October\or November\or December\fi }
\def \Th #1{%
  \ifnum #1=1 st\else\ifnum #1=21 st\else\ifnum #1=31 st\else
  \ifnum #1=2 nd\else\ifnum #1=22 nd\else
  \ifnum #1=3 rd\else\ifnum #1=23 rd\else th\fi\fi\fi\fi\fi\fi\fi }
```

`<now>` The `<now>` command will quite probably remain unchanged for most languages (except for the `\TimeSep`; see above), but is given here just the same.

```
\Cdef {now}{\Minutes = \time  \Hours = \Minutes
  \divide \Hours by 60  \Htemp = \Hours  \multiply \Htemp by -60
  \advance \Minutes by \Htemp
  \the\Hours \TimeSep \ifnum \Minutes > 9 \else 0\fi \the\Minutes }
```

The three counters used by `<now>` must be defined.

```
\newcount \Minutes  \newcount \Hours  \newcount \Htemp
```

1.7 Math mode

A few modifications to the TEX math mode are necessary.

The `<` character must be active also in math mode.

```
\mathcode‘< = "8000
```

`<math>` `</math>` The inline math mode `$...$` is called `<math>...</math>` in StarTEX.

```
\Cdef {math}{\MathFonts \NewEnvir{math}{$}{$}}
\Cdef {/math}{\EndEnvir{math}}
```

`<displaymath>` `</displaymath>` The display math environment `$$...$$` is also available in StarTEX using the notation `<displaymath>...</displaymath>`.

```
\Cdef {displaymath}{\endgraf \MathFonts \NewEnvir{displaymath}{$$}{$$}}
\Cdef {/displaymath}{\EndEnvir{displaymath}}
```

1.7.1 Math fonts

`\MSetFont` The `\MSetFont` macro is used to simplify the definition of math fonts. It takes two parameters: `#1` is the math font element being defined (i.e., `\textfont0`), and `#2` is the font name.

```
\def \MSetFont #1#2{\font \NewMFont = \csname #2\endcsname
  #1 = \NewMFont }
```

`\MathFonts` The `\MathFonts` macro is called whenever math mode is entered. It will define all the required math fonts. To avoid unnecessary work, new fonts will only be assigned if the font size has changed since the last time.

```
\def \MathFonts {\if \FontSize \LastMathSize \else
  \MSetFont{\textfont0}{M-\FontSize N0}%
  \MSetFont{\scriptfont0}{M-\FontSize S0}%
  \MSetFont{\scriptscriptfont0}{M-\FontSize X0}%
  \MSetFont{\textfont1}{M-\FontSize N1}%
  \MSetFont{\scriptfont1}{M-\FontSize S1}%
  \MSetFont{\scriptscriptfont1}{M-\FontSize X1}%
  \MSetFont{\textfont2}{M-\FontSize N2}%
  \MSetFont{\scriptfont2}{M-\FontSize S2}%
  \MSetFont{\scriptscriptfont2}{M-\FontSize X2}%
  \MSetFont{\textfont3}{M-\FontSize N3}%
  \MSetFont{\scriptfont3}{M-\FontSize S3}%
  \MSetFont{\scriptscriptfont3}{M-\FontSize X3}%
  \MSetFont{\textfont4}{M-\FontSize N4}%
  \MSetFont{\scriptfont4}{M-\FontSize S4}%
  \MSetFont{\scriptscriptfont4}{M-\FontSize X4}%
  \MSetFont{\textfont5}{M-\FontSize N5}%
  \MSetFont{\scriptfont5}{M-\FontSize S5}%
  \MSetFont{\scriptscriptfont5}{M-\FontSize X5}%
  \MSetFont{\textfont6}{M-\FontSize N6}%
  \MSetFont{\scriptfont6}{M-\FontSize S6}%
  \MSetFont{\scriptscriptfont6}{M-\FontSize X6}%
  \let \LastMathSize = \FontSize
 \fi }
```

`\LastMathSize` `\LastMathSize` contains the last math size used. It must be initialized to an unused value so that the math fonts are always loaded the first time math mode is entered.

```
\def \LastMathSize {?}
```

1.7.2 Letters

The standard 26 letters are already defined, but we must define the others, like ‘é’ and ‘Å’. These are taken from the ECTI font (number 5 in StarTEX).

```
\def \Df #1{\mathcode"#1 = "05#1 \relax }
\Df{C0}\Df{C1}\Df{C2}\Df{C3}\Df{C4}\Df{C5}\Df{C6}\Df{C7}
\Df{C8}\Df{C9}\Df{CA}\Df{CB}\Df{CC}\Df{CD}\Df{CE}\Df{CF}
\Df{D0}\Df{D1}\Df{D2}\Df{D3}\Df{D4}\Df{D5}\Df{D6}
\Df{D8}\Df{D9}\Df{DA}\Df{DB}\Df{DC}\Df{DD}\Df{DE}\Df{DF}
\Df{E0}\Df{E1}\Df{E2}\Df{E3}\Df{E4}\Df{E5}\Df{E6}\Df{E7}
\Df{E8}\Df{E9}\Df{EA}\Df{EB}\Df{EC}\Df{ED}\Df{EE}\Df{EF}
\Df{F0}\Df{F1}\Df{F2}\Df{F3}\Df{F4}\Df{F5}\Df{F6}
\Df{F8}\Df{F9}\Df{FA}\Df{FB}\Df{FC}\Df{FD}\Df{FE}\Df{FF}
```

1.7.3 Bold letters

`<bolda>`
`<bolducA>`

The bold letters `<bolda>`…`<boldz>` and `<bolducA>`…`<bolducZ>` are intended for use i math mode. They are found on the CMBX font (number 4 in StarTEX).

```
\def \Df #1#2#3{\Cdef{bold#1}{\MSy{\mathchar"04#2}}%
 \Cdef{bolduc#1}{\MSy{\mathchar"04#3}}}
```

```
\Df{a}{61}{41}\Df{b}{62}{42}\Df{c}{63}{43}\Df{d}{64}{44}
\Df{e}{65}{45}\Df{f}{66}{46}\Df{g}{67}{47}\Df{h}{68}{48}
\Df{i}{69}{49}\Df{j}{6A}{4A}\Df{k}{6B}{4B}\Df{l}{6C}{4C}
\Df{m}{6D}{4D}\Df{n}{6E}{4E}\Df{o}{6F}{4F}\Df{p}{70}{50}
\Df{q}{71}{51}\Df{r}{72}{52}\Df{s}{73}{53}\Df{t}{74}{54}
\Df{u}{75}{55}\Df{v}{76}{56}\Df{w}{77}{57}\Df{x}{78}{58}
\Df{y}{79}{59}\Df{z}{7A}{5A}
```

1.7.4 Operators

`\MSy` `\MSy` is used to insert a mathematical symbol. It will expand to `#1`, and insert a math mode environment (`$`...`$`) if required.

```
\def \MSy #1{\ifmmode #1\else $#1$\fi }
```

`\MOp` `\MOp` is also used to insert the mathematical symbol `#2`, but gives an error message if not in a math environment. Parameter `#1` is the StarTEX command name, which is used in the error message.

```
\def \MOp #1#2{\ifmmode \def \MNext {#2}\else
  \Error{Command <#1> is only allowed in math mode;}%
    {command ignored.}\let \MNext = \relax \fi
  \MNext }
```

`\Mdef` `\Mdef` is used to define a StarTEX mathematical symbol command whenever the StarTEX command is identical to the name of the corresponding TEX macro; parameter `#1` is the name of both.

```
\def \Mdef #1{\Cdef{#1}{\MSy{\csname #1\endcsname}}}
```

`\Odef` `\Odef` is similar to `\Mdef` but is for definitions based on `\MOp` rather than `\MSy`.

```
\def \Odef #1{\Cdef{#1}{\MOp{#1}{\csname #1\endcsname}}}
```

These commands are now used to define all the standard unary and binary operators.

```
\Mdef{amalg} \Mdef{bigcirc} \Mdef{bigtriangleup}
\Mdef{bigtriangledown} \Mdef{bullet} \Mdef{cap} \Mdef{circ}
\Mdef{cup} \Mdef{dagger} \Mdef{ddagger} \Mdef{diamond} \Mdef{mp}
\Mdef{odot} \Mdef{ominus} \Mdef{oplus} \Mdef{oslash} \Mdef{otimes}
\Mdef{setminus} \Mdef{sqcap} \Mdef{sqcup} \Mdef{star}
\Mdef{triangleleft} \Mdef{triangleright} \Mdef{uplus} \Mdef{vee}
\Mdef{wedge} \Mdef{wr}
```

1.7.5 Relations

These are the various relations available.

```
\Mdef{approx} \Mdef{asymp} \Mdef{bowtie} \Mdef{cong} \Mdef{doteq}
\Mdef{dashv} \Mdef{equiv} \Mdef{frown} \Mdef{geq} \Mdef{gg} \Mdef{in}
\Mdef{leq} \Mdef{ll} \Mdef{mid} \Mdef{models} \Mdef{neq} \Mdef{ni}
\Mdef{notin} \Mdef{parallel} \Mdef{perp} \Mdef{prec} \Mdef{preceq}
\Mdef{propto} \Mdef{sim} \Mdef{simeq} \Mdef{smile} \Mdef{sqsubseteq}
\Mdef{sqsupseteq} \Mdef{subset} \Mdef{subseteq} \Mdef{supset}
\Mdef{supseteq} \Mdef{succ} \Mdef{succeq} \Mdef{vdash}
```

`<gt>` `<lt>` The two symbols `<gt>` and `<lt>` must be defined; the former is not really necessary, but is included for symmetry reasons. Note also that the definition of `<lt>` is slightly tricky since `<` is an active symbol in math mode.

```
 \Cdef{gt}{>}
 \Cdef{lt}{\ifmmode \mathchar"313C \else <\fi}
```

1.7.6 Delimitiers

In addition to (), [], {} and |, the following delimiters are defined:

```
\Mdef{langle} \Mdef{lceil} \Mdef{lfloor} \Mdef{rangle}
\Mdef{rceil} \Mdef{rfloor}
```

1.7.7 Arrows

All the standard arrows are included.

```
\Mdef{leftarrow} \Mdef{rightarrow} \Mdef{leftrightarrow}
\Mdef{longleftarrow} \Mdef{longrightarrow} \Mdef{longleftrightarrow}
\Mdef{uparrow} \Mdef{downarrow}
\Mdef{updownarrow} \Mdef{nearrow} \Mdef{nwarrow} \Mdef{searrow}
\Mdef{swarrow} \Mdef{mapsto} \Mdef{longmapsto} \Mdef{hookleftarrow}
\Mdef{hookrightarrow} \Mdef{leftharpoonup} \Mdef{rightharpoonup}
\Mdef{leftharpoondown} \Mdef{rightharpoondown} \Mdef{rightleftharpoons}
```

In TEX, some arrows use uppercase letters to indicate double arrows. I use the prefix `double` instead.

```
\Cdef{doubleleftarrow}{\MSy{\Leftarrow}}
\Cdef{doublerightarrow}{\MSy{\Rightarrow}}
\Cdef{doubleleftrightarrow}{\MSy{\Leftrightarrow}}
\Cdef{doublelongleftarrow}{\MSy{\Longleftarrow}}
\Cdef{doublelongrightarrow}{\MSy{\Longrightarrow}}
\Cdef{doublelongleftrightarrow}{\MSy{\Longleftrightarrow}}
\Cdef{doubleuparrow}{\MSy{\Uparrow}}
\Cdef{doubledownarrow}{\MSy{\Downarrow}}
\Cdef{doubleupdownarrow}{\MSy{\Updownarrow}}
```

1.7.8 Big symbols

These are the big symbols.

```
\Odef{bigcap} \Odef{bigcup} \Odef{bigodot} \Odef{bigoplus}
\Odef{bigotimes} \Odef{bigsqcup} \Odef{biguplus} \Odef{bigvee}
\Odef{bigwedge} \Odef{int} \Odef{oint} \Odef{prod} \Odef{sum}
```

1.7.9 Function names

Some names are defined as commands to ensure that they will be printed in the correct font.

```
\Mdef{arccos} \Mdef{arcsin} \Mdef{arctan} \Mdef{arg} \Mdef{cos}
\Mdef{cosh} \Mdef{cot} \Mdef{coth} \Mdef{csc} \Mdef{deg} \Mdef{det}
\Mdef{dim} \Mdef{exp} \Mdef{gcd} \Mdef{hom} \Mdef{inf} \Mdef{ker}
\Mdef{lg} \Mdef{lim} \Mdef{liminf} \Mdef{limsup} \Mdef{ln}
\Mdef{log} \Mdef{max} \Mdef{min} \Mdef{sec} \Mdef{sin} \Mdef{sinh}
\Cdef{sup-op}{\MSy{\sup}} \Mdef{tan} \Mdef{tanh}
```

`<pr>` must be given a special definition as its name in plain TeX has an uppercase letter.

```
\Cdef{pr}{\MSy{\Pr}}
```

1.7.10 Greek letters

The lowercase Greek letters can be found just using the plain TeX name.

```
\Mdef{alpha} \Mdef{beta} \Mdef{gamma} \Mdef{delta} \Mdef{epsilon}
\Mdef{varepsilon} \Mdef{zeta} \Mdef{eta} \Mdef{theta} \Mdef{vartheta}
\Mdef{iota} \Mdef{kappa} \Mdef{lambda} \Mdef{mu} \Mdef{nu} \Mdef{xi}
\Cdef {omicron}{\MSy{o}} \Mdef{pi} \Mdef{varpi} \Mdef{rho} \Mdef{varrho}
\Mdef{sigma} \Mdef{varsigma} \Mdef{tau} \Mdef{upsilon} \Mdef{phi}
\Mdef{varphi} \Mdef{chi} \Mdef{psi} \Mdef{omega}
```

The uppercase Greek letters can be found in the CMR font, which is font no 0 in StarTeX. Those that also exist in the Latin alphabet, can be found in their normal position, while the special Greek letters are placed at the beginning of the font table.

```
\def \Df #1#2{\Cdef{#1}{\MSy{\mathchar"#2}}}
\Df{ucalpha}{0041} \Df{ucbeta}{0042}    \Df{ucgamma}{0000}
\Df{ucdelta}{0001} \Df{ucepsilon}{0045} \Df{uczeta}{005A}
\Df{uceta}{0048}   \Df{uctheta}{0002}   \Df{uciota}{0049}
\Df{uckappa}{004B} \Df{uclambda}{0003}  \Df{ucmu}{004D}
\Df{ucnu}{004E}    \Df{ucxi}{0004}      \Df{ucomicron}{004F}
\Df{ucpi}{0005}    \Df{ucrho}{0050}     \Df{ucsigma}{0006}
\Df{uctau}{0054}   \Df{ucupsilon}{0007} \Df{ucphi}{0008}
\Df{ucchi}{0058}   \Df{ucpsi}{0009}     \Df{ucomega}{000a}
```

1.7.11 Blackboard set letters

The socalled "blackboard bold" letters $\mathbb{C}$, $\mathbb{N}$, $\mathbb{R}$ and $\mathbb{Z}$ are used to denote the standard sets.

```
\def \Df #1#2{\Cdef{#1}{\MSy{\mathchar"06#2}}}
\Df{cset}{43} \Df{nset}{4E} \Df{rset}{52} \Df{zset}{5A}
```

1.7.12 Calligraphic letters

`cala` The calligraphic letters can be found in font 2.

```
\def \Df #1#2{\Cdef{cal#1}{\MSy{\mathchar"02#2}}}
\Df{a}{41} \Df{b}{42} \Df{c}{43} \Df{d}{44} \Df{e}{45}
\Df{f}{46} \Df{g}{47} \Df{h}{48} \Df{i}{49} \Df{j}{4A}
\Df{k}{4B} \Df{l}{4C} \Df{m}{4D} \Df{n}{4E} \Df{o}{4F}
\Df{p}{50} \Df{q}{51} \Df{r}{52} \Df{s}{53} \Df{t}{54}
\Df{u}{55} \Df{v}{56} \Df{w}{57} \Df{x}{58} \Df{y}{59}
\Df{z}{5A}
```

1.7.13 Other symbols

A few symbols are left.

```
\Mdef{aleph} \Mdef{angle} \Mdef{bot} \Mdef{ell} \Mdef{emptyset}
\Mdef{exists} \Mdef{forall} \Mdef{hbar} \Mdef{nabla} \Mdef{neg}
\Odef{not} \Mdef{partial} \Mdef{surd} \Mdef{top} \Mdef{wp}
\Cdef{infinity}{\MSy{\infty}} \Cdef{im}{\MSy{\Im}} \Cdef{re}{\MSy{\Re}}
```

```
\Cdef{,}{{,}}
\Cdef{:}{\MSy{\vdots}}
\Cdef{:::}{\MSy{\cdots}}
```

`<...>` The definition of `<...>` has a minute space to allow StarTEX to split the line.

```
\Cdef{...}{\MSy{\ldots}\hskip 0.001pt \relax }
```

1.7.14 Subscripts and superscripts

`<sub>` `</sub>` `<sup>` `</sup>` The sub- and superscripts in equations are accessible through `<sub>`...`</sub>` and `<sup>`...`</sup>`.

```
\Cdef {sub}{\MOp{sub}{\NewEnvir{sub}{_\bgroup}{\egroup}}}
\Cdef {/sub}{\MOp{/sub}{\EndEnvir{sub}}}
\Cdef {sup}{\MOp{sup}{\NewEnvir{sup}{^\bgroup}{\egroup}}}
\Cdef {/sup}{\MOp{/sup}{\EndEnvir{sup}}}
```

1.7.15 Fractions

`<frac>` `<over>` `</frac>` Fractions are set with in a `<frac>`...`<over>`...`</frac>` environment.

```
\Cdef {frac}{\MOp{frac}{\NewEnvir{frac}{\bgroup}{\egroup}%
    \NOver = 0\relax}}
\Cdef {over}{\MOp{over}{\FracOver}}
\Cdef {/frac}{\MOp{/frac}{\ifnum \NOver = 0
    \Error{No <over> in the <frac>...<over>...</frac> environment.}{}\fi
    \EndEnvir{frac}}}
\def \FracOver {\ifTextEqual{\CurEnv}{frac}%
    \ifnum \NOver = 0 \over
    \else \Error{Only one <over> may occur in each <frac>...</frac>
        environment.}{}%
    \fi  \advance \NOver by 1
  \else \Error{<over> only allowed in a
    <frac>...<over>...</frac> environment.}{}%
  \fi }
```

To check for legal use of `<over>`, a counter named `\NOver` is used.

```
\newcount \NOver
```

1.7.16 Roots

`<sqrt>` `</sqrt>` The `<sqrt>`...`</sqrt>` environment is just an interface to the `\sqrt` macro in plain TEX.

```
\Cdef {sqrt}{\MOp{sqrt}{\NewEnvir{sqrt}{\sqrt\bgroup}{\egroup}}}
\Cdef {/sqrt}{\MOp{/sqrt}{\EndEnvir{sqrt}}}
```

1.8 PostScript figures

`<psfig>` `</psfig>` StarTEX allows the inclusion of POSTSCRIPT figures through the `<psfig>`...`</psfig>` environment. It checks if the command is allowed in the current state, and, if it is, `\PSfig` is called; if not, `\PSfigError` is called instead.

```
\Cdef {psfig}{\if \State x \let \Next = \PSfig \else
  \let \Next = \PSfigError \fi
  \Next }
```

```
\Cdef {/psfig}{\AddVspace{6pt plus 1pt}
  \ifx \PSfile \relax \else
    \centerline{\epsfbox{\PSfile}}%
    \AddVspace{10pt plus 2pt minus 1pt}
  \fi
  \EndEnvir{psfig}}
```

The actual inclusion of the PostScript file in handled by `\epsfbox` from the file `epsf.tex` which is provided with the DVIPS program.

```
\input epsf.tex
```

`\PSfig` `\PSfig` checks whether the user has remembered to provide the file name; if so, `\PSfetch` is called to check the PostScript file.

```
\def \PSfig{\NewEnvir{psfig}{\topinsert}{\endinsert}
  \let \PSfile = \relax \let \State = p
  \IfNextChar{[}%
    {\PSfetch}%
    {\Error{No file name for PostScript figure;}%
      {the syntax is: <psfig>[file name]caption text</psfig>.}%
     \PScaption}}
```

`\PSfigError` `\PSfigError` gives an error message and starts a dummy environment.

```
\def \PSfigError {\endgraf
  \Error{Calls on <psfig> not allowed inside <psfig> or <table>;}%
     {the command was ignored.}%
  \NewEnvir{psfig}{\begingroup}{\endgroup}}
```

`\PSfetch` `\PSfetch` checks that the PostScript exists, and notes its name in `\PSfile`.

```
\def \PSfetch [#1]{\IfFileExists{#1}%
    {\gdef \PSfile {#1}}%
    {\Error{PostScript file `#1' could not be found.}{}}%
    \PScaption }
```

`\epsfsize` `\epsfsize` is used by `\epsfbox` to scale the PostScript figure. We want figures to fill either 80% of the text width or 40% of the text height. This is done by comparing $\frac{0.8\texttt{\textbackslash hsize}}{\#1}$ and $\frac{0.4\texttt{\textbackslash vsize}}{\#2}$, where `#1` and `#2` are the horizontal and vertical original sizes of the the figure.

```
\def \epsfsize #1#2{0pt \dimen1 = 0.8\hsize  \dimen2 = 0.4\vsize
  \floatdiv{\dimen2}{#1}{#2}%
  \ifdim \dimen1 < \divres \epsfxsize = 0.8\hsize
  \else \epsfysize = 0.4\vsize \fi }
```

`\PScaption` The formatting of the caption is done by `\PScaption`:

```
\def \PScaption {\global\advance \FigCnt by 1
  \edef \CurDef {\the\FigCnt}
  \leftskip = 0.1\hsize  \rightskip = \leftskip
  \ResetFont \SetSize{S}\Indentfalse
  \FigureName~\FigIm: \ignorespaces }
```

`\PScaption` needs a counter to count the figures:

```
\newcount \FigCnt
```

1.9 Tables

`<table>` `</table>` Tables are created with the `<table>`...`</table>` environment. It checks whether the command is allowed in the current state and calls `\Table` if it is; otherwise it calls `\TableError`.

```
\Cdef {table}{\if \State x\let \Next = \Table \else
  \let \Next = \TableError \fi
  \Next }
\Cdef {/table}{\EndEnvir{table}}
```

The table is temporarily placed in a box so that it may be centered when output.

```
\newbox \TableBox
```

`\Table` `\Table` starts the table environment.

```
\def \Table{\topinsert
  \NewEnvir{table}{\TableCaption}{\LastRow\endinsert}%
  \let \State = t}
```

`\TableCaption` The formatting of the table caption is done by `\TableCaption`:

```
\def \TableCaption {\global\advance \TabCnt by 1
  \edef \CurDef {\the\TabCnt}
  \leftskip = 0.1\hsize  \rightskip = \leftskip
  \ResetFont \SetSize{S}\Indentfalse
  \TableName~\TabIm: \ignorespaces }
```

`\TableCaption` needs a counter to count the tables:

```
\newcount \TabCnt
```

`\TableError` `\TableError` gives an error message and starts a dummy environment.

```
\def \TableError {\endgraf
  \Error{Calls on <table> not allowed inside <psfig>
    or <table>;}{the command was ignored.}%
  \NewEnvir{table}{\begingroup}{\endgroup}}
```

1.9.1 Table rows

`<row>` `\Row` The command `<row>` is the interface to `\Row` which is used to start another table row. Its actions depend on the current state.

```
\Cdef {row}{\Row}
\def \Row {\if \State t\FirstRow\NewRow \else
   \if \State r\EndRow\NewRow   \else
   \Error{The <row> command is only allowed inside a
     <table>.}{The command was ignored.}\fi\fi }
```

`\FirstRow` `\FirstRow` is called prior to the first table row.

```
\def \FirstRow {\endgraf \ResetFont \SelectFont \let \State = r%
  \setbox\TableBox = \hbox\bgroup \vbox\bgroup \offinterlineskip
    \halign\bgroup \vrule ##\strut&&
      \kern 6pt \hfil ##\unskip \hfil \kern 6pt \vrule \cr
      \noalign{\hrule}}
```

`\NewRow` `\NewRow` is used to start another table row.

```
\def \NewRow {&}
```

`\EndRow` The command `\EndRow` is used to terminate the current table row.

```
\def \EndRow {\cr \noalign{\hrule}}
```

`\LastRow` The command `\LastRow` is used to terminate the last table row.

```
\def \LastRow {\if \State r\EndRow\PrintTable\fi}
```

`\PrintTable` `\PrintTable` is the final command used when processing a table. It terminates the table structure and outputs it.

```
\def \PrintTable {\egroup\egroup\egroup
  \AddVspace{6pt plus 2pt minus 1pt}
  \centerline{\box\TableBox}%
  \AddVspace{10pt plus 4pt minus 2pt}}
```

1.9.2 Table columns

`<col>` `<col>` is used to start another table column.

```
\Cdef {col}{\if \State r&\else
  \Error{The <col> command is only allowed after a <row>
    inside a <table>.}{The command was ignored.}\fi}
```

1.10 Cross-references

This section describes the various commands used for cross-referencing.

The basic idea is the same as the one used in LaTeX: an extra file (with extension `.aux`) contains all the labels defined in the document. There are two reasons for this:

- Using a separate file makes it possible to have forward references.
- When processing a document, the page number of each label is not known until the page is actually written to the DVI file. The `\write` command is, consequently, delayed until that stage of the processing, so using a file will always give us the correct page number.

`\ifAuxOpen` The `.aux` file is only created when there are `<label>` commands in the document. The first `<label>` creates the file, and the flag `\ifAuxOpen` indicates whether this has been done.

```
\newif \ifAuxOpen
```

1.10.1 Defining labels

`<label>` `<label>` is used to define another cross-reference label. The value of the label is the current value of `\CurDef` which is set by `<h1>`, `<table>`, etc.

`<label>` first checks the syntax, and then alters the `\catcode` of `<` in case there are any `<`s in the label. Finally, it calls `\NewLabel` to perform the actual definition.

```
\Cdef {label}{\IfNextChar{[}%
    {\begingroup \catcode`\< = 12 \NewLabel}%
    {\Error{No label given;}%
      {the syntax is <label>[your label].}}}
```

`\NewLabel`

`\NewLabel` first reads the old `.aux` file (unless it has already been read) and opens a new `.aux` file (unless that has been done).

```
\def \NewLabel [#1]{\endgroup
  \ifAuxRead \else \ReadAuxFile \fi
  \ifAuxOpen \else
    \immediate\openout \NewAuxFile = \jobname.aux
    \global\AuxOpentrue \fi
```

Then, it checks whether the label has been previously defined, and prints an error message if it has been. Otherwise, it notes the definition and writes the necessary data to the `.aux` file.[5]

```
  \expandafter\ifx \csname L]#1\endcsname \relax
    \expandafter\edef \csname L]#1\endcsname{\the\inputlineno}%
    {\let \the = 0\relax
      \edef \WriteCurDef {\write \NewAuxFile
        {\string\LabelDef
          ]#1]\the\pageno]\CurDef E-o-LabelDef\string\relax}}%
      \WriteCurDef }%
  \else
    \Error{Label ‘#1’ already defined on line \csname L]#1\endcsname;}%
      {this definition is ignored.}%
  \fi }
```

Note that when a label *xxx* is defined, a macro named `\L]`*xxx* is defined, and its definition is the line number. This is used to give a better error message in case there are multiple definitions of that label.

The new `.aux` file must be declared.

```
\newwrite \NewAuxFile
```

We must also define an initial value for `\CurDef` in case the user calls `<label>` prior to the first `<h`*x*`>`.

```
\edef \CurDef {0}
```

1.10.2 Reading the cross-reference label file

When the user first references a label (using a `<ref>` command), the `.aux` file is read to find all the labels that were defined during the previous run. Each label will be represented by to macros:

`R]`*xxx* gives the reference, i.e. “2.1” if the label was defined in Section 2.1 of the document.

`P]`*xxx* gives the page number where the label was placed.

`\ReadAuxFile`

`\ReadAuxFile` reads the `.aux` file produced by the previous run. It redefines the `\catcode` of `<` in case that characters occurs in a label.

```
\def \ReadAuxFile {\IfFileExists{\jobname.aux}%
    {\begingroup
       \def \LabelDef {\begingroup
          \catcode ‘\\ = 12\relax \FetchLabel }%
       \catcode ‘\< = 12 \catcode ‘\\ = 0\relax
```

---

[5] Note the trick of redefining `\the` so that `\pageno` is not expanded until the page is completed. This was found in [2, Exercise 21.10].

```
       \input \jobname.aux
       \global\AuxReadtrue
     \endgroup }%
    {\global\Reruntrue}}
```

`\LabelDef` A StarTeX `.aux` file consists of a sequence of calls on `\LabelDef`. `\LabelDef` just alters the `\catcode` of `\` (in case there are `\`s in the label), and calls `\FetchLabel` to do the actual definition.

`\FetchLabel` `\FetchLabel` has three parameters. The label is parameter `#1`; it is enclosed by a set of `]`s.[6] Parameter `#2` is the referenced page number; it is also terminated by a `]`. The label reference is parameter `#3`; it is terminated by the text "`E-o-LabelDef`".

```
\def \FetchLabel ]#1]#2]#3E-o-LabelDef{%
    \expandafter\gdef \csname R]#1\endcsname {#3}%
    \expandafter\gdef \csname P]#1\endcsname {#2}%
 \endgroup }
```

The flag `\ifAuxRead` tells whether the `.aux` file from the previous run has been read.

```
\newif \ifAuxRead
```

The flag `\ifRerun` is set whenever it is detected that a new label has been defined or the definition of a label has been modified. This implies that it is necessary to run StarTeX again.[7]

```
\newif \ifRerun
```

1.10.3 Checking the `.aux` file

`\CheckAux` At the end of each run, the `.aux` file is checked by `\CheckAux` to see if any label were modified during the run. If so, the flag `\ifRerun` is set.

```
\def \CheckAux {\immediate\closeout \NewAuxFile
 \begingroup
    \def \LabelDef {\begingroup
        \catcode ‘\\ = 12 \relax  \CheckLabel }%
    \catcode ‘\< = 12  \catcode ‘\\ = 0\relax
    \input \jobname.aux
 \endgroup }
```

`\CheckLabel` `\CheckLabel` does the actual checking by comparing the new definition with the old one.

```
\def \CheckLabel ]#1]#2]#3E-o-LabelDef{%
  \ifTextEqual{#2++#3}%
    {\csname P]#1\endcsname++\csname R]#1\endcsname}%
  \else
    \global\Reruntrue
  \fi \endgroup }
```

---

[6] We can be certain that no `]` occurs in the label.

[7] It would be nice to be able to do this automatically, but that is not possible with the present TeX.

1.10.4 Referencing labels

`<ref>`

`<ref>` is used to reference a label. It checks the syntax, and calls `\GiveRef` to produce the actual reference.

```
\Cdef {ref}{\IfNextChar{[}%
  {\begingroup \catcode `\< = 12 \relax \GiveRef }%
  {\Error{No label referenced;}{the syntax is <ref>[your label].}}}
```

`\GiveRef`

`\GiveRef` checks whether the reference has been defined, and, if it is defined, typesets the reference in the format specified by `\RefFormat`.

```
\def \GiveRef [#1]{\ifAuxRead \else \ReadAuxFile \fi
  \expandafter\ifx \csname R]#1\endcsname \relax
    [label #1]%
    \Error{Label '#1' not defined.}{}\global\Reruntrue
  \else
    \RefFormat{\csname R]#1\endcsname}{\csname P]#1\endcsname}%
  \fi \endgroup }
```

`\RefFormat`

The macro `\RefFormat` gives the appearance of the reference. The default is a number followed by “on page x” unless the reference is to the same page.[8]

```
\def \RefFormat #1#2{#1\ifTextEqual{#2}{\the\pageno}\else
    \space on page~#2\fi }
```

1.11 Miscellaneous other commands

1.11.1 Symbols and logos

`<>`
`<->`
`<-->`
`<--->`
`<``>`
`<''>`
`<tex>`
`<latex>`
`<startex>`

Here is the definition of some symbols that the user needs for quality typesetting.

```
{\catcode`\~ = 12 \Cdef {~}{\nobreak\ }}
\Cdef {}{\null}
\Cdef {-}{\-} \Cdef {--}{--} \Cdef {---}{---}
\Cdef {``}{``} \Cdef {''}{''}
\Cdef {tex}{\TeX} \Cdef {latex}{La\TeX} \Cdef {startex}{Star\TeX}
```

1.11.2 Error handling

`\Error`

This macro gives an error message with one or two lines of text. (The second line is omitted if it is empty.)

```
\def \Error #1#2{%
  \message{^^J** StarTeX error detected on line \the\inputlineno:^^J}%
  \message{\space\space\space #1^^J}%
  \ifTextEqual{#2}{}\else \message{\space\space\space #2^^J}\fi}
```

`\Warning`

This macro gives a warning with one or two lines of text. (The second line is omitted if it is empty.)

```
\def \Warning #1#2{%
  \message{^^J** StarTeX warning:^^J}%
  \message{\space\space\space #1^^J}%
  \ifTextEqual{#2}{}\else \message{\space\space\space #2^^J}\fi}
```

---

[8] The page test used will not always give the correct result; if the reference is very near the top or the bottom of a page, `\pageno` will sometimes be 1 off. This is not a serious problem, as the referenced object in that case will be very close to the reference.

This problem was solved in the `varioref`[3] package in LaTeX, but the solution is somewhat complicated, so I have chosen to stick with the simple solution even if it is not optimal.

We need to modify the `\newlinechar` in order to produce a line feed when we want it. The LF character (ASCII code 10 = `^^J`) seems a good choice.

```
\newlinechar = `^^J
```

1.11.3 Comments

`<comment>`
`</comment>`

The environment `<comment>`...`</comment>` is defined to make it possible for the user to add comments to his or her StarTEX code. It uses `\SkipComment` to fetch the comment text.

```
\Cdef {comment}{\NewEnvir{comment}{\begingroup}{\endgroup}%
  \catcode `\< = 12 \SkipComment }
```

`\SkipComment`

`\SkipComment` works just like `\ReadCode` for a `<code>`...`</code>` environment, except that the code is simply ignored.

```
{\catcode `\< = 12
  \gdef \SkipComment #1</comment>{\EndEnvir{comment}}}
```

1.11.4 Text handling

There is very little text handling in StarTEX, but a few macros are necessary.

`\ifTextEqual`

The macro `\ifTextEqual`[9] tests if the two texts `#1` and `#2` are equal. This is achieved by letting the TEX primitive `\ifx` do the job.

```
\def \ifTextEqual #1#2{\edef \TmpA {#1}\edef \TmpB {#2}%
  \ifx \TmpA\TmpB }
```

`\IfNextChar`

Sometimes we need to check the next input character, for instance to see if it is a `[`. The macro `\IfNextChar` expands into `#2` if the next input character is `#1`; otherwise it expands into `#3`.

The TEX primitive `\futurelet` is used to implement `\IfNextChar`; it was designed just for this. Note that the call on `\IfNextChar` must be at the very end of the macro in which it occurs.

```
\def \IfNextChar #1#2#3{\def \TestChar {#1}%
  \def \AltA {#2}\def \AltB {#3}%
  \futurelet \NextChar \TestNextChar }
\def \TestNextChar {%
  \if \NextChar \TestChar \let\Next=\AltA \else \let\Next=\AltB \fi
  \Next }
```

`\IfNextCharTwo`

A similar macro testing whether the following character is one of two possible alternatives, is also necessary. It takes four parameters: `#1` and `#2` are the two characters to check for, and the result of `\IfNextCharTwo` is `#3` if the next character is either of the two alternatives, or else `#4`.

```
\def \IfNextCharTwo #1#2#3#4{\def \TestChar {#1}\def \TestCharX {#2}%
  \def \AltA {#3}\def \AltB {#4}%
  \futurelet \NextChar \TestNextCharTwo }
\def \TestNextCharTwo {%
```

---

[9] The name `\ifTextEqual` starts with a lowercase `\if`- to show its relationship to the various `\if`... macros of primitive TEX. `\ifTextEqual` is used just like these, which means that it must have a corresponding `\fi`. It may also have an `\else` part. Note, however, that calls on `\ifTextEqual` may not occur inside other `\if`s.

```
  \if \NextChar \TestChar  \let \Next = \AltA \else
  \if \NextChar \TestCharX \let \Next = \AltA \else
    \let \Next = \AltB
  \fi \fi
  \Next }
```

1.11.5 Numerical operations

`\floatdiv` TeX has no floating-point division, so a routine like `\floatdiv` is necessary. It computes $\frac{\#1\times\#2}{\#3}$ and places the result in `\divres`.[10] All parameters must be `\dimen` registers.

```
\def \floatdiv #1#2#3{\divtemp = #1\divide \divtemp by #3%
  \divres = #2\multiply \divres by \divtemp
  \multiply \divtemp by #3%
  \divrem = #1\advance \divrem by -\divtemp
  \divtemp = #2%
  \loop
    \multiply \divrem by 2  \divide \divtemp by 2
  \ifnum \divtemp > 0
    \ifnum \divrem < #3\else
      \advance \divrem by -#3\advance \divres by \divtemp
    \fi
  \repeat }
```

The necessary registers must be declared:

```
\newdimen \divrem   \newdimen \divres   \newdimen \divtemp
```

1.11.6 File operations

`\ifFileExists` To provide better error messages, StarTeX tests whether a file exists before any attempt is made to read it. `\ifFileExists` first checks if file `#1` exists;[11] if it does, the file is closed and `#2` is executed, otherwise `#3` is executed.

```
\def \IfFileExists #1#2#3{\openin\TestFile = #1
  \ifeof\TestFile \def \Next {#3}\else
  \closein\TestFile \def \Next {#2}\fi
  \Next }
```

`\TestFile` We must also define the input file we use for testing.

```
\newread \TestFile
```

1.11.7 Format creation

`<make-new-format>` `<make-new-format>` is used by the system to make a new format; it is not intended to be called by the user.

```
\Cdef {make-new-format}{\dump}
```

`<trace-on>` `<trace-off>` `<trace-on>` is used during debugging to trace the expansion of commands, and `<trace-off>` turns this tracing off. Using tracing requires insight into both the inner mechanisms of StarTeX as well as TeX itself.

---

[10] The algorithm used is taken from the code in `epsf.tex`.

[11] This checking is done by using `\ifeof` which means that empty files will be regarded as non-existant; this should not really be a problem.

```
\Cdef {trace-on}{\tracingcommands = 1 \tracingmacros = 1
  \tracingrestores = 1 \tracingoutput = 1 \relax }
\Cdef {trace-off}{\tracingcommands = 0 \tracingmacros = 0
  \tracingrestores = 0 \tracingoutput = 0 \relax }
```

1.12 Final initialization

In the final initialization, we need to load the hyphenation patterns defined at this site:

```
\input startex.lan
```

We select the default text font:

```
\ResetFont \SetSize{N}
```

Our very last initialization is to modify the `\catcode`s:

```
\endinput \SpecialCatCodes
⟨/code⟩
```

2 Language definitions

This section describes the file `startex.lan` that defines which hyphenation patterns are known to StarTEX. It should be adapted to the needs of each local installation.

```
⟨*languages⟩
```

`\DefLang` The macro `\DefLang` is used to defined a new language. It has three parameters: `#1` is the internal number of the language, `#2` is the internal name of the language, and `#3` is the name of the file from which to read hyphenation patterns for the language.

```
\def \DefLang #1#2#3{\expandafter\def \csname L-#2\endcsname {#1}%
  \language = #1 \input #3}%
```

Only one language (Norwegian) is predefined here, as this file will have to be adapted to local needs anyhow.

```
\DefLang{1}{norsk}{nohyph2.tex}%
```

`american` (American English) has already been defined as language 0 by plain TEX.

```
\expandafter\def \csname L-american\endcsname {0}%
```

We want American English as our default:

```
\language = 0
⟨/languages⟩
```

3 Sample document styles

Included are also four document styles:

article is used for ordinary articles. As this is the default definition of StarTEX, this style is empty.

a4-article is just an adaption of *article* to fit A4 paper.

ifi-article is the style used for articles at Ifi ("Institutt for informatikk", the Department of Informatics at the Univeristy of Oslo). It uses Concorde as the main text font, Palatino for its math, and A4 paper. (If you want to use this style, you must have access to a Concorde font; the Palatino Math is available at CTAN.)

ifi-artikkel is *ifi-article* adapted to Norwegian text.

Each style file contains the necessary redefinitions.

3.1 Paper size

Every style except *article* uses A4 paper. The margins are set to 2.5 cm.

```
⟨*a4 – article | ifi – article | ifi – artikkel⟩
\hoffset = -1in  \advance \hoffset by 2.5cm  \hsize = 16cm
\voffset = -1in  \advance \voffset by 2.5cm  \vsize = 24.7cm
⟨/a4 – article | ifi – article | ifi – artikkel⟩
```

3.2 Language adaption

The *ifi-artikkel* is for text in Norwegian. We must select the correct language and adjust `\lefthyphenmin` and `\righthyphenmin`.

```
⟨*ifi – artikkel⟩
\language = \csname L-norsk\endcsname
\lefthyphenmin = 1 \righthyphenmin = 2
```

`\AbstractName` `\FigureName` `\TableName` `\TimeSep`

Some names must also be redefined.

```
\def \AbstractName {Sammendrag}%
\def \FigureName   {Figur}%
\def \TableName    {Tabell}%
\def \TimeSep      {.}%
```

`\RefFormat`

The format for references must be translated into Norwegian.

```
\def \RefFormat #1#2{#1\ifTextEqual{#2}{\the\pageno}\else
  \space p\aa{} side~#2\fi }%
```

`<today>` `\Month`

Finally, the date command `<today>` must be redefined.

```
\Cdef {today}{\the\day.\space \Month\space \the\year }%
\def \Month {\ifcase \month \or januar\or februar\or mars\or
  april\or mai\or juni\or juli\or august\or september\or
  oktober\or november\or desember\fi }%
⟨/ifi – artikkel⟩
```

3.3 Text font

The two *ifi-* styles use Concorde[12] as the main body text font; it is the official font of the University of Oslo. Since Concorde is a wider font than ECR, and because it has a larger x-height, a smaller font size is used. The EC Companion font is kept for the special symbols.

```
⟨*ifi – article | ifi – artikkel⟩
```

---

[12] In the Karl Berry font name scheme [1], Concorde is known as *poc-*.

```
\FontDef{R}{M}{U}{N}{pocr8t at 9.8pt}{tcrm1095}%
\FontDef{R}{M}{I}{N}{pocri8t at 9.8pt}{tcti1095}%
\FontDef{R}{B}{U}{N}{pocb8t at 9.8pt}{tcbx1095}%
\FontDef{R}{B}{I}{N}{pocbi8t at 9.8pt}{tcbi1095}%
\FontDef{T}{M}{U}{N}{pcrr8t at 9.8pt}{tctt1095}%
\FontDef{T}{M}{I}{N}{pcrro8t at 9.8pt}{tcit1095}%
\FontDef{T}{B}{U}{N}{pcrb8t at 9.8pt}{tctt1095}%
\FontDef{T}{B}{I}{N}{pcrbo8t at 9.8pt}{tcit1095}%
\FontDef{R}{M}{U}{L}{pocr8t at 12.9pt}{tcrm1440}%
\FontDef{R}{M}{I}{L}{pocri8t at 12.9pt}{tcti1440}%
\FontDef{R}{B}{U}{L}{pocb8t at 12.9pt}{tcbx1440}%
\FontDef{R}{B}{I}{L}{pocbi8t at 12.9pt}{tcbi1440}%
\FontDef{T}{M}{U}{L}{pcrr8t at 12.9pt}{tctt1440}%
\FontDef{T}{M}{I}{L}{pcrro8t at 12.9pt}{tcit1440}%
\FontDef{T}{B}{U}{L}{pcrb8t at 12.9pt}{tctt1440}%
\FontDef{T}{B}{I}{L}{pcrbo8t at 12.9pt}{tcit1440}%
\FontDef{R}{M}{U}{X}{pocr8t at 15.5pt}{tcrm1728}%
\FontDef{R}{M}{I}{X}{pocri8t at 15.5pt}{tcti1728}%
\FontDef{R}{B}{U}{X}{pocb8t at 15.5pt}{tcbx1728}%
\FontDef{R}{B}{I}{X}{pocbi8t at 15.5pt}{tcbi1728}%
\FontDef{T}{M}{U}{X}{pcrr8t at 15.5pt}{tctt1728}%
\FontDef{T}{M}{I}{X}{pcrro8t at 15.5pt}{tcit1728}%
\FontDef{T}{B}{U}{X}{pcrb8t at 15.5pt}{tctt1728}%
\FontDef{T}{B}{I}{X}{pcrro8t at 15.5pt}{tcit1728}%
\FontDef{R}{M}{U}{S}{pocr8t at 8.9pt}{tcrm1000}%
\FontDef{R}{M}{I}{S}{pocri8t at 8.9pt}{tcti1000}%
\FontDef{R}{B}{U}{S}{pocb8t at 8.9pt}{tcbx1000}%
\FontDef{R}{B}{I}{S}{pocbi8t at 8.9pt}{tcbi1000}%
\FontDef{T}{M}{U}{S}{pcrr8t at 8.9pt}{tctt1000}%
\FontDef{T}{M}{I}{S}{pcrro8t at 8.9pt}{tcit1000}%
\FontDef{T}{B}{U}{S}{pcrb8t at 8.9pt}{tctt1000}%
\FontDef{T}{B}{I}{S}{pcrbo8t at 8.9pt}{tcit1000}%
```

The “Math Palatino” font is used for math.

```
\def \XIIpt  { at 12pt}
\def \Xpt    { at 10pt}
\def \IXpt   { at 9pt}
\def \VIIIpt { at 8pt}
\def \VIIpt  { at 7pt}
\def \VIpt   { at 6pt}
\def \Vpt    { at 5pt}
\MathFontDef{N}{0}{zppler7t\XIpt}{zppler7t\VIIIpt}{zppler7t\VIpt}
\MathFontDef{N}{1}{zppler7m\XIpt}{zppler7m\VIIIpt}{zppler7m\VIpt}
\MathFontDef{N}{2}{zppler7y\XIpt}{zppler7y\VIIIpt}{zppler7y\VIpt}
\MathFontDef{N}{3}{zppler7v\XIpt}{zppler7v\XIpt}{zppler7v\XIpt}
\MathFontDef{N}{4}{zppleb7t\XIpt}{zppleb7t\VIIIpt}{zppleb7t\VIpt}
\MathFontDef{N}{5}{pplri8t\XIpt}{pplri8t\VIIIpt}{pplri8t\VIpt}
\MathFontDef{L}{0}{zppler7t\XIVpt}{zppler7t\Xpt}{zppler7t\VIIIpt}
\MathFontDef{L}{1}{zppler7m\XIVpt}{zppler7m\Xpt}{zppler7m\VIIIpt}
\MathFontDef{L}{2}{zppler7y\XIVpt}{zppler7y\Xpt}{zppler7y\VIIIpt}
\MathFontDef{L}{3}{zppler7v\XIVpt}{zppler7v\XIVpt}{zppler7v\XIVpt}
\MathFontDef{L}{4}{zppleb7t\XIVpt}{zppleb7t\Xpt}{zppleb7t\VIIIpt}
\MathFontDef{L}{5}{pplri8t\XIVpt}{pplri8t\Xpt}{pplri8t\VIIIpt}
\MathFontDef{X}{0}{zppler7t\XVIIpt}{zppler7t\XIIpt}{zppler7t\IXpt}
\MathFontDef{X}{1}{zppler7m\XVIIpt}{zppler7m\XIIpt}{zppler7m\IXpt}
```

```
\MathFontDef{X}{2}{zppler7y\XVIIpt}{zppler7y\XIIpt}{zppler7y\IXpt}
\MathFontDef{X}{3}{zppler7v\XVIIpt}{zppler7v\XVIIpt}{zppler7v\XVIIpt}
\MathFontDef{X}{4}{zppleb7t\XVIIpt}{zppleb7t\XIIpt}{zppleb7t\IXpt}
\MathFontDef{X}{5}{pplri8t\XVIIpt}{pplri8t\XIIpt}{pplri8t\IXpt}
\MathFontDef{S}{0}{zppler7t\Xpt}{zppler7t\VIIpt}{zppler7t\Vpt}
\MathFontDef{S}{1}{zppler7m\Xpt}{zppler7m\VIIpt}{zppler7m\Vpt}
\MathFontDef{S}{2}{zppler7y\Xpt}{zppler7y\VIIpt}{zppler7y\Vpt}
\MathFontDef{S}{3}{zppler7v\Xpt}{zppler7v\Xpt}{zppler7v\Xpt}
\MathFontDef{S}{4}{zppleb7t\Xpt}{zppleb7t\VIIpt}{zppleb7t\Vpt}
\MathFontDef{S}{5}{pplri8t\Xpt}{pplri8t\VIIpt}{pplri8t\Vpt}
```

We must select the newly selected standard font.

```
\SelectFont
⟨/ifi – article | ifi – artikkel⟩
```

References

[1] Karl Berry. Fontname. Available from CTAN in info/fontname, July 1998.

[2] Donald E. Knuth. *The TeXbook*. 1991.

[3] Frank Mittelbach. The `varioref` package. Part of the LaTeX 2ε distribution., May 1995.

[4] Philip Taylor. ε-TeX: a 100%-compatible successor to TeX. In *EuroTeX'95*, pages 359–370, September 1995.

Index

Numbers written in italic refer to the page where the corresponding entry is described, the ones underlined to the code line of the definition, the rest to the code lines where the entry is used.

Change History

1.01

General: Added forgotten operator `<propto>`. 22

Name of `\sup` operator changed to `<sup-op>` because of name clash with the `<sup>` command used for indices. (Detected by Marco Paggi, aampaggi@input.lognet.it.) 23

1.02

General: Changed from ‘dc’ to ‘ec’ fonts. 8

1.03

General: Added the various set names. (Suggested by Christian Sommerfeldt Øien, christso@ifi.uio.no.) 24

1.04

General: Changed math font to “Palatino Math” for the Ifi document styles. 36

`cala`: Fixed a bug in the `<calA>` and related commands. 24

`<bolducA>`: Added `<bolda>`, `<bolducA>`, and related commands. 21