

ltoutenc.dtx

Johannes Braams David Carlisle Alan Jeffrey
Frank Mittelbach Chris Rowley Rainer Schöpf

2004/02/22

1 Font encodings

This section of the kernel contains commands for declaring encoding-specific commands, such as accents. It also contains the code for some of the encoding files, including `omlenc.def`, `omsenc.def`, `t1enc.def` and `ot1enc.def` files, which define the OLM, OMS, T1 and OT1 encodings, and the `fontenc` package for selecting encodings.

The `fontenc` package has options for encodings, of which the last option is the default encoding. For example, to use the OT2, OT3 and T1 encodings, with T1 as the default, you say:

```
\usepackage[OT2,OT3,T1]{fontenc}
```

The standard kernel set-up loads font encoding files and selects an encoding as follows.

```
\input {omlenc.def}  
\input {t1enc.def}  
\input {ot1enc.def}  
\input {omsenc.def}  
\fontencoding{OT1}
```

Note that the files in the standard `inputenc` package depend on this behaviour of the kernel.

The syntax for declaring encoding-specific commands is:

```
\DeclareTextCommand{<command>}{<encoding>}  
[<number>][<default>]{<commands>}
```

This command is like `\newcommand`, except that it defines a command which is specific to one encoding. The resulting command is always robust, even if its definition is fragile. For example, the definition of `\l` in the OT1 encoding is:

```
\DeclareTextCommand{\l}{OT1}{\@xxxii l}
```

`\DeclareTextCommand` takes the same optional arguments as `\newcommand`.

```
\ProvideTextCommand{\langle command \rangle}{\langle encoding \rangle}
[\langle number \rangle] [\langle default \rangle] {\langle commands \rangle}
```

This acts like `\DeclareTextCommand`, but does nothing if the command is already defined.

```
\DeclareTextSymbol{\langle command \rangle}{\langle encoding \rangle}{\langle slot \rangle}
```

This command defines a text symbol, with a particular slot in that encoding. The commands:

```
\DeclareTextSymbol{\ss}{OT1}{25}
\DeclareTextCommand{\ss}{OT1}{\char25 }
```

have the same effect, but the `\DeclareTextSymbol` is faster.

```
\DeclareTextAccent{\langle command \rangle}{\langle encoding \rangle}{\langle slot \rangle}
```

This command declares a text accent. The commands:

```
\DeclareTextAccent{\"}{OT1}{127}
\DeclareTextCommand{\"}{OT1}{\add@accent {127}}
```

have the same effect.

```
\DeclareTextComposite{\langle command \rangle}
{\langle encoding \rangle}{\langle argument \rangle}{\langle slot \rangle}
```

This command declares a composite letter, for example in the T1 encoding `\'a` is slot 225, which is declared by:

```
\DeclareTextComposite{\'}{T1}{a}{225}
```

The *command* will normally have been declared with `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

`\DeclareTextComposite` is the most common example of using the more general declaration `\DeclareTextCompositeCommand`, which can define a composite to be an arbitrary piece of text.

```
\DeclareTextCompositeCommand{\langle command \rangle}
{\langle encoding \rangle}{\langle argument \rangle}{\langle text \rangle}
```

For example, in the OT1 encoding \AA has a hand-crafted definition this is declared as follows

```
\DeclareTextCompositeCommand{\r}{OT1}{A}
{\leavevmode\setbox\z@\hbox{!}\dimen@ \ht\z@\advance\dimen@-1ex%
\rlap{\raise.67\dimen@\hbox{\char23}}A}
```

The *command* will normally have been declared with `\DeclareTextAccent`, or as a one-argument `\DeclareTextCommand`.

The commands defined using the above declarations can be used in two ways. Normally they are used by just calling the command in the appropriate encoding, for example `\ss`. However, sometimes you may wish to use a command in an encoding where it is not defined. If the command has no arguments, then you can use it in another encoding by calling `\UseTextSymbol`:

`\UseTextSymbol{<encoding>}{<command>}`

v1.9e1997/08/05 Corrected order of arguments in `\UseTextSymbol` example. For example, `\UseTextSymbol{OT1}{\ss}` has the same effect as:

`{\fontencoding{OT1}\selectfont\ss}`

If the command has one argument then you can use it in another encoding by calling `\UseTextAccent`:

`\UseTextAccent{<encoding>}{<command>}{<text>}`

For example, if the current encoding is OT2 then `\UseTextAccent{OT1}{\'}{a}` has the same effect as:

`{\fontencoding{OT1}\selectfont\'}{\fontencoding{OT2}\selectfont a}`

You can also declare a default definition for a text command, which will be used if the current encoding has no appropriate definition. Such use will also set the definition for this command in the current encoding to equal this default definition; this makes subsequent uses of the command much faster.

`\DeclareTextCommandDefault{<command>}{<definition>}`

For example, the default definition of the command `\textonequarter` (which produces the fraction $\frac{1}{4}$) could be built using math mode:

`\DeclareTextCommandDefault{\textonequarter}{\ensuremath {\frac{1}{4}}}`

There is a matching `\Provide` command which will not override an existing default definition:

`\ProvideTextCommandDefault{<command>}{<definition>}`

The most common use for these commands is to use symbols from other encodings, so there are some optimizations provided:

`\DeclareTextSymbolDefault{<command>}{<encoding>}`
`\DeclareTextAccentDefault{<command>}{<encoding>}`

are short for:

`\DeclareTextCommandDefault{<command>}`
`\UseTextSymbol{<encoding>}{<command>}`
`\DeclareTextCommandDefault[1]{<command>}`
`\UseTextAccent{<encoding>}{<command>}{#1}`

For example, to make OT1 the default encoding for `\ss` and `\'` you say:

`\DeclareTextSymbolDefault{\ss}{OT1}`
`\DeclareTextAccentDefault{\'}{OT1}`

Note that you can use these commands on any zero- or one-argument commands declared with `\DeclareText*` or `\ProvideText*`, not just those defined using `\DeclareTextSymbol` or `\DeclareTextAccent`.

1.1 Removing encoding-specific commands

In some cases encoding definitions are given to provide some limited support since nothing better is available, for example, the definition for `\textdollar` in `OT1` is a hack since `$` and `£` actually share the same slot in this encoding. Thus if such a glyph becomes available in a different encoding (e.g., `TS1`) one would like to get rid of the flacky one and make the default definition point to the new encoding. In such a case defining

```
\DeclareTextSymbol{\textdollar}{TS1}{36}
\DeclareTextSymbolDefault{\textdollar}{TS1}
```

is not enough since if typesetting in `OT1` \LaTeX will still find the encoding specific-definition for `OT1` and therefore ignore the new default. Therefore to ensure that in this case the `TS1` version is used we have to remove the `OT1` declaration:

```
\UndeclareTextCommand{\textdollar}{OT1}
```

Since the `$` sign is a proper glyph in the `T1` encoding there is no point removing its definition and forcing \LaTeX to pick up the `TS1` version if typesetting in this encoding. However, assume you want to use the variant dollar sign, i.e., `⸀` for your dollars. In that case you have to get rid of the `T1` declaration as well, e.g., the following would do that for you:

```
\UndeclareTextCommand{\textdollar}{OT1}
\UndeclareTextCommand{\textdollar}{T1}
\DeclareTextCommandDefault{\textdollar}
{\UseTextSymbol{TS1}\textdollaroldstyle}
```

1.2 The order of declarations

If an encoding-specific command is defined for more than one encoding, then it will execute fastest in the encoding in which it was defined last since its top-level definition will be set up to execute in that encoding without any overhead.

For this reason the file `fonttext.ltx` currently first loads the definitions for the `T1` encoding and then those for the `OT1` encoding so that typesetting in `OT1` is optimized since that is (still) the default. However, when `T1` is explicitly requested (via `\usepackage[T1]{fontenc}`) the top-level definitions are automatically changed to favour `T1` since its declarations are reloaded in the process.

For the same reason default declarations should never come last since they are implemented as a special encoding themselves (with the name `?`). Specifying them last would simply mean to make those encoding-specific commands equally inefficient in all encodings. Therefore the `textcomp` package, for example, first sets up all defaults to point to `TS1` and then declares the commands in the `TS1` encoding.

1.3 Docstrip modules

This `.dtx` file is be used to generate several related files containing font encoding definitions. The mutually exclusive docstrip options are listed here.

T1	generates <code>t1enc.def</code> for the Cork encoding.
TS1	generates <code>ts1enc.def</code> for the Text Companion encoding.
TS1sty	generates <code>textcomp.sty</code> , package that sets up use of the Text Companion encoding.
OT1	generates <code>ot1enc.def</code> for Knuth's CM encoding.
OMS	generates <code>omsenc.def</code> for Knuth's math symbol encoding.
OML	generates <code>omlenc.def</code> for Knuth's math letters encoding.
OT4	generates <code>ot4enc.def</code> for the Polish extension to the OT1 encoding, created by B. Jackowski and M. Ryćko for use with the Polish version of Computer Modern and Computer Concrete.
package	generates <code>fontenc.sty</code> for selecting encodings.
2ekernel	for the kernel commands.
autoload	for the 'autoload' kernel commands.
autoerr	for the <code>autoerr.sty</code> error message autoload file.

1.4 Definitions for the kernel

1.4.1 Declaration commands

This section contains definitions for commands such as accents which depend on the current encoding. These commands will usually be kept in `.def` files, for example `ot1enc.def` contains the definitions for the OT1 encoding.

```
1 (*2ekernel | autoload)
2 \message{font encodings,}

Far too many macros in one block here!
```

```
\DeclareTextCommand If you say:
\ProvideTextCommand
\DeclareTextSymbol      \DeclareTextCommand{\foo}{T1}...
  \@dec@text@cmd then \foo is defined to be \T1-cmd \foo \T1\foo, where \T1\foo is one control
  \chardef@text@cmd sequence, not two! We then call \newcommand to define \T1\foo.
  \@changed@cmd
  \@changed@x
\TextSymbolUnavailable 3 \def\DeclareTextCommand{%
  \inmathwarn          4   \@dec@text@cmd\newcommand}
                    5 \def\ProvideTextCommand{%
                    6   \@dec@text@cmd\providecommand}
                    7 \def\@dec@text@cmd#1#2#3{%
                    8   \expandafter\def\expandafter#2%
                    9   \expandafter{%
                   10     \csname#3-cmd\expandafter\endcsname
                   11     \expandafter#2%
                   12     \csname#3\string#2\endcsname
                   13   }%
                   14   \let\@ifdefinable\@rc@ifdefinable
                   15   \expandafter#1\csname#3\string#2\endcsname}
```

This command was introduced to fix a major bug in `\@dec@text@cmd` without changing that command itself. This was thought to be necessary because it is defined in more than one package. (Perhaps the more serious bug is to put complex low-level commands like this in packages?)

The problem it solves is that whereas both `\newcommand` and `\providecommand` (used just above) both handle the resetting of `\@ifdefinable` (following its disabling in `\@dec@text@cmd`), the primitive `\chardef` neither needs the disabling, nor does the resetting.

```

16 \def\chardef@text@cmd{%
17   \let\@ifdefinable\@ifdefinable
18   \chardef
19 }
20 \def\DeclareTextSymbol#1#2#3{%
21   \@dec@text@cmd\chardef@text@cmd#1{#2}#3\relax
22 }

```

The declarations are only available before `\begin{document}`.

```

23 \@onlypreamble\DeclareTextCommand
24 \@onlypreamble\DeclareTextSymbol

```

The sneaky bit in all this is what `\T1-cmd \foo \T1\foo` does. There are five possibilities, depending on the current values of `\protect`, `\cf@encoding` and `\ifmmode`:

- If `\protect` is `\@typeset@protect` and `\cf@encoding` is `T1`, then we execute `\T1\foo`. This should be the normal behaviour, and is optimized for speed.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) `OT1`, and `\OT1\foo` is defined, then we execute `\OT1\foo`.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) `OT1`, we're in text mode, and `\OT1\foo` is undefined, then we define `\OT1\foo` to be the default value of `\foo`, and execute `\OT1\foo`.
- If `\protect` is `\@typeset@protect`, `\cf@encoding` is (say) `OT1`, we're in math mode, and `\OT1\foo` is undefined, then we execute the default value of `\foo`. (This is necessary so that things like `X_\copyright` work properly.)
- If `\protect` is not `\@typeset@protect` then we execute `\noexpand\foo`. For example, if we are writing to a file, then this results in `\foo` being written. If we are in a `\mark`, then `\foo` will be put in the mark—since `\foo` is robust, it will then survive all the things which may happen to it whilst it's a `\mark`.

So after all that, we will either execute the appropriate definition of `\foo` for the current encoding, or we will execute `\noexpand\foo`.

The default value of `\foo` is `\?\foo` if it is defined, and an error message otherwise.

When the encoding is changed from T1 to OT1, \T1-cmd is defined to be \@changed@cmd and \OT1-cmd is defined to be \@current@cmd. This means that the test for what the current encoding is can be performed quickly.

```

25 \def\@current@cmd#1{%
26   \ifx\protect\@typeset@protect
27     \@inmathwarn#1%
28   \else
29     \noexpand#1\expandafter\@gobble
30   \fi}
31 \def\@changed@cmd#1#2{%
32   \ifx\protect\@typeset@protect
33     \@inmathwarn#1%
34     \expandafter\ifx\csname\cf@encoding\string#1\endcsname\relax
35     \expandafter\ifx\csname ?\string#1\endcsname\relax
36     \expandafter\def\csname ?\string#1\endcsname{%
37       \TextSymbolUnavailable#1%
38     }%
39   \fi
40   \global\expandafter\let
41     \csname\cf@encoding\string#1\endcsname
42     \csname ?\string#1\endcsname
43   \fi
44   \csname\cf@encoding\string#1%
45     \expandafter\endcsname
46   \else
47     \noexpand#1%
48   \fi}
49 </2ekernel | autoload>
50 <*2ekernel | autoerr>
51 \gdef\TextSymbolUnavailable#1{%
52   \@latex@error{%
53     Command \protect#1 unavailable in encoding \cf@encoding%
54   }\@eha}
55 </2ekernel | autoerr>
56 <autoload>\gdef\TextSymbolUnavailable{\@autoerr\TextSymbolUnavailable}
57 <*2ekernel | autoload>

```

The command \@inmathwarn produces a warning message if we are currently in math mode. Note that since this command is used inside text commands, it can't call \relax before the \ifmmode. This means that it is possible for the warning to fail to be issued at the beginning of a row of an halign whose template enters math mode. This is probably a bad feature, but there's not much that can be done about it, since adding a \relax would break ligatures and kerning between text symbols.

A more efficient solution would be to make \@inmathwarn and \@inmatherr equal to \@empty and \relax by default, and to have \everymath reset them to their usual definitions. This is left for future investigation (for example it may break some third party code).

```

58 \def\@inmathwarn#1{%
59   \ifmmode
60     \@latex@warning{Command \protect#1 invalid in math mode}%
61   \fi}

```

`\DeclareTextCommandDefault` These define commands with encoding ?.

`\ProvideTextCommandDefault` Note that `\DeclareTextCommandDefault` can only be used in the preamble, but that the `\Provide` version is allowed in inputenc .def files, so is allowed anywhere.

```

62 \def\DeclareTextCommandDefault#1{%
63   \DeclareTextCommand#1?}
64 \def\ProvideTextCommandDefault#1{%
65   \ProvideTextCommand#1?}
66 \@onlypreamble\DeclareTextCommandDefault
67 %\@onlypreamble\ProvideTextCommandDefault

```

They require `\?–cmd` to be initialized as `\@changed@cmd`.

```
68 \expandafter\let\csname?–cmd\endcsname\@changed@cmd
```

`\DeclareTextAccent` This is just a disguise for defining a TeX `\accent` command.

```

69 \def\DeclareTextAccent#1#2#3{%
70   \DeclareTextCommand#1{#2}{\add@accent{#3}}
71 \@onlypreamble\DeclareTextAccent

```

`\add@accent` To save space this code is shared between all text accents that are set using the `\accent` primitive. The argument is pre-set in a box so that any font loading that is needed is already done within the box. This is needed because font-loading involves grouping and that would prevent the accent mechanism from working so that the accent would not be positioned over the argument. Declarations that change the font should be allowed (only low-level ones are at present) inside the argument of an accent command, but not size changes, as they involve `\setbox` operations which also inhibit the mechanism of the `\accent` primitive.

Note that the whole process is within a group. For a detailed discussion of this reimplementaion and its deficiencies, see pr/3160. v1.9z2000/01/30Macro reimplemented (pr/3160)

```
72 \def\add@accent#1#2{\hmode@bgroup
```

Turn off the group in `\UseTextSymbol` in case this is used inside the argument of `\add@accent`.

```
73   \let\hmode@start@before@group\@firstofone
74   \setbox\@tempboxa\hbox{#2%
```

When presetting the argument in a box we record its `\spacefactor` for later use after the accent got typeset. This way something like `\‘A` gets the spacefactor of A (i.e., 999) rather than the default value of 1000.

```
75     \global\mathchardef\accent@spacefactor\spacefactor}%
76   \accent#1 #2\egroup\spacefactor\accent@spacefactor}

```

Default definition for `\accent@spacefactor` prevents a horrible death of the above macro inside an unprotected `\edef`.

```
77 \let\accent@spacefactor\relax
```

```
\hmode@bgroup
```

```
78 \def\hmode@bgroup{\leavevmode\bgroup}
```

```
\DeclareTextCompositeCommand Another amusing game to play with \expandafter, \csname, and \string. When
\DeclareTextComposite you say \DeclareTextCompositeCommand{\foo}{T1}{a}{bar}, we look to see if
\@text@composite the expansion of \T1\foo begins with \@text@composite, and if it doesn't, we
\@text@composite@x redefine \T1\foo to be:
\@strip@args #1 -> \@text@composite \T1\foo #1\@empty \@text@composite {...}
```

where ... is the previous definition of `\T1\foo`. Finally, we define `\\T1\foo-a` to expand to `bar`.

```
79 \def\DeclareTextCompositeCommand#1#2#3#4{%
80   \expandafter\let\expandafter\reserved@a\csname#2\string#1\endcsname
81   \expandafter\expandafter\expandafter\ifx
82   \expandafter\@car\reserved@a\relax\relax\@nil \@text@composite \else
83     \edef\reserved@b##1{%
84       \def\expandafter\noexpand
85       \csname#2\string#1\endcsname###1{%
86         \noexpand\@text@composite
87         \expandafter\noexpand\csname#2\string#1\endcsname
88         ###1\noexpand\@empty\noexpand\@text@composite
89         {##1}}}%
90   \expandafter\reserved@b\expandafter{\reserved@a{##1}}%
91   \fi
92   \expandafter\def\csname\expandafter\string\csname
93     #2\endcsname\string#1-\string#3\endcsname{#4}}
94 \@onlypreamble\DeclareTextCompositeCommand
```

This all works because:

```
\@text@composite \T1\foo A\@empty \@text@composite {...}
```

expands to `\\T1\foo-A` if `\\T1\foo-A` has been defined, and `{...}` otherwise.

Note that `\@text@composite` grabs the first token of the argument and puts just that in the `csname`. This is so that `\'\{\textit{e}\}` will work—it checks whether `\\T1\'-\textit` is defined (which presumably it isn't) and so expands to `\accent 1 \textit{e}`.

This trick won't always work, for example `\'\{\itshape e}\}` will expand to (with spaces added for clarity):

```
\csname \string \T1\' - \string {\itshape e} \@empty \endcsname
```

which will die pretty horribly. Unfortunately there's not much can be done about this if we're going to use `\csname` lookups as a fast way of accessing composites.

This has an unfortunate ‘misfeature’ though, which is that in the T1 encoding, `\’{aa}` produces á. This is not the expected behaviour, and should perhaps be fixed if the fix doesn’t affect performance too badly.

Finally, it’s worth noting that the `\@empty` is used in `\@text@composite` so that accents will work even when the argument is empty. If you say `\’{}` then this looks up `\T1\’-\@empty`, which ought to be `\relax`, and so all is well. If we didn’t include the `\@empty`, then `\’{}` would expand to:

```
\csname \string \T1\’ - \string \endcsname
```

so the `\endcsname` would be `\string’ed` and the whole of the rest of the document would be put inside the `\csname`. This would not be good.

```
95 \def\@text@composite#1#2#3\@text@composite{%
96   \expandafter\@text@composite@x
97     \csname\string#1-\string#2\endcsname}
```

Originally the `\@text@composite@x` macro had two arguments and if `#1` was not `\relax` it was executed, otherwise `#2` was executed. All this happened within the `\ifx` code so that neither `#1` nor `#2` could have picked up any additional arguments from the input stream. This has now been changed using the typical `\@firstoftwo` / `\@secondoftwo` coding. This way the final expansion will happen without any `\else` or `\fi` intervening in the case that we need to get a further token from the input stream.

```
98 \def\@text@composite@x#1{%
99   \ifx#1\relax
100     \expandafter\@secondoftwo
101   \else
102     \expandafter\@firstoftwo
103   \fi
104   #1}
```

The command `\DeclareTextComposite` uses `\DeclareTextCompositeCommand` to declare a command which expands out to a single glyph.

```
105 \catcode\z@=11\relax

106 \def\DeclareTextComposite#1#2#3#4{%
107   \def\reserved@a{\DeclareTextCompositeCommand#1{#2}{#3}}%
108   \bgroup
109     \lccode\z@#4%
110     \lowercase{%
111   \egroup
112     \reserved@a ^^@}}

113 \catcode\z@=15\relax

114 \@onlypreamble\DeclareTextComposite
```

<code>\UseTextAccent</code>	These fragile commands access glyphs from different encodings. They use grotty
<code>\UseTextSymbol</code>	low-level calls to the font selection scheme for speed, and in order to make sure
<code>\@use@text@encoding</code>	that <code>\UseTextSymbol</code> doesn’t do anything which you’re not allowed to do between
	an <code>\accent</code> and its glyph.

For a detailed discussion of this reimplementaion and its deficiencies, see pr/3160. v1.9z2000/01/30Macro reimplemented (pr/3160)

```

115 \def\UseTextAccent#1#2#3{%
116   \hmode@start@before@group
117   {%
    Turn off the group in \UseTextSymbol in case this is used inside the arguments
    of \UseTextAccent.
118     \let\hmode@start@before@group\@firstofone
119     \let\@curr@enc\cf@encoding
120     \@use@text@encoding{#1}%
121     #2{\@use@text@encoding\@curr@enc#3}%
122     }}

123 \def\UseTextSymbol#1#2{%
124   \hmode@start@before@group
125   {%
126     \def\@wrong@font@char{\MessageBreak
127       for \noexpand\symbol'\string#2'}%
128     \@use@text@encoding{#1}%
129     #2%
130   }%
131   }

132 \def\@use@text@encoding#1{%
133   \edef\f@encoding{#1}%
134   \xdef\font@name{%
135     \csname\curr@fontshape/\f@size\endcsname}%
136   \pickup@font
137   \font@name
138   \@@enc@update}

```

`\hmode@start@before@group` The `\hmode@start@before@group` starts hmode and should be immediately followed by an explicit `{...}`. Its purpose is to ensure that hmode is started before this group is opened. Inside `\add@accent` and `\UseTextAccent` it is redefined to remove this group so that it doesn't conflict with the `\accent` primitive.

For a detailed discussion see pr/3160.

```
139 \let\hmode@start@before@group\leavevmode
```

`\DeclareTextSymbolDefault` Some syntactic sugar. Again, these should probably be optimized for speed.

```

\DeclareTextAccentDefault 140 \def\DeclareTextSymbolDefault#1#2{%
141   \DeclareTextCommandDefault#1{\UseTextSymbol{#2}#1}}
142 \def\DeclareTextAccentDefault#1#2{%
143   \DeclareTextCommandDefault#1{\UseTextAccent{#2}#1}}
144 \@onlypreamble\DeclareTextSymbolDefault
145 \@onlypreamble\DeclareTextAccentDefault

```

`\UndeclareTextCommand` This command safely removes and encoding specific declaration for a given encoding. It is helpful if one intends to use the default definition always and therefore wants to get rid of a declaration for some specific encoding.

```
146 \def\UndeclareTextCommand#1#2{%
```

If there is no declaration for the current encoding do nothing. (This makes a hash table entry but without eTeX we can't do anything about that).

```
147 \expandafter\ifx\csname#2\string#1\endcsname\relax
148 \else
```

Else: throw away that declaration.

```
149 \global\expandafter\let\csname#2\string#1\endcsname
150 \undefined
```

But this is unfortunately not enough, we have to take a look at the top-level definition of the encoding specific command which for a command `\foo` would look similar to `\T1-cmd \foo \T1\foo` (three tokens).

Of course, instead of T1 one could see a different encoding name; which one depends the encoding for which `\foo` was declared last.

Now assume we have just removed the declaration for `\foo` in T1 and the top-level of `\foo` expands to the above. Then we better change that pretty fast otherwise we do get an “undefined csname error” when we try to typeset `\foo` within T1 instead of getting the default definition for `\foo`. And what is the best way to change that top-level definition? Well, the only “encoding” we know for sure will still be around is the default encoding denoted by ?.

Thus in case the last token of the top-level expansion is now undefined we change the declaration to look like `\?-cmd \foo \?\foo` which is done by the following (readable?) code:

```
151 \expandafter\expandafter\expandafter
152 \ifx\expandafter\@thirdofthree#1\undefined
153 \expandafter\gdef\expandafter#1\expandafter
154 {\csname ?-cmd\expandafter\endcsname\expandafter
155 #1\csname?\string#1\endcsname}%
156 \fi
157 \fi
158 }
159 \onlypreamble\UndeclareTextCommand
```

1.4.2 Hyphenation

`\patterns` We redefine `\patterns` and `\hyphenation` to allow the use of commands declared with `\DeclareText*` to be used inside them.

```
\@patterns 160 %\let\@patterns\patterns
\hyphenation \@hyphenation 161 %\let\@hyphenation\hyphenation
162 %\def\patterns{%
163 % \bgroup
164 % \let\protect\@empty
165 % \let\@typeset\protect\@empty
166 % \let\@changed@x\@changed@x@mouth
167 % \afterassignment\egroup
168 % \@patterns
169 %}
```

```

170 %\def\hyphenation{%
171 %   \bgroup
172 %     \let\protect\@empty
173 %     \let\@typeset@protect\@empty
174 %     \let\@changed@x\@changed@x@mouth
175 %   \afterassignment\egroup
176 %   \@hyphenation
177 %}

```

1.4.3 Miscellania

`\a` The `\a` command is used to access the accent commands even when they have been redefined (for example by the `tabbing` environment). Its internal name is `\@tabacckludge`.

The `\string` within the `\csname` guards against something like `'` being active at the point of use.

```

178 \def\@tabacckludge#1{\expandafter\@changed@cmd
179                               \csname\string#1\endcsname\relax}
180 \let\a=\@tabacckludge

```

1.4.4 Default encodings

We define the default encodings for most commands to be either OT1, OML or OMS. These defaults are in the kernel and therefore fonts with these encodings must be available unless these defaults are redefined elsewhere. Recall that the standard kernel loads the encoding files for these encodings, and also that for the T1 encoding.

The naming conventions in the kernel are not what we would use if we were starting from scratch... Those defined by DEK (like `\ae` and `\ss`) or by the T_EX Users Group Technical Working Group on multi-lingual typesetting (like `\th` and `\ng`) have short names. Those which were added to the kernel in 1993 and early 1994 are named after their Adobe glyph names (like `\guillemotleft` and `\quotedblbase`). Unfortunately, this naming scheme won't work for all glyphs, since some names (like `\space`) are already used, and some (like `\endash`) are very likely to be defined by users. So we're now using the naming scheme of `\text` followed by the Adobe name, (like `\textendash` and `\textsterling`). Except that some glyphs don't have Adobe names, so we're using the names used by fontinst for those (like `\textcompwordmark`). Sigh.

Some accents from OT1:

```

181 \DeclareTextAccentDefault{"}{OT1}
182 \DeclareTextAccentDefault{'}{OT1}
183 \DeclareTextAccentDefault{.}{OT1}
184 \DeclareTextAccentDefault{=}{OT1}
185 \DeclareTextAccentDefault{H}{OT1}
186 \DeclareTextAccentDefault{^}{OT1}
187 \DeclareTextAccentDefault{'}{OT1}
188 \DeclareTextAccentDefault{b}{OT1}

```

```

189 \DeclareTextAccentDefault{\c}{OT1}
190 \DeclareTextAccentDefault{\d}{OT1}
191 \DeclareTextAccentDefault{\r}{OT1}
192 \DeclareTextAccentDefault{\u}{OT1}
193 \DeclareTextAccentDefault{\v}{OT1}
194 \DeclareTextAccentDefault{\~}{OT1}

```

Some symbols from OT1:

```

195 %\DeclareTextSymbolDefault{\AA}{OT1}
196 \DeclareTextSymbolDefault{\AE}{OT1}
197 \DeclareTextSymbolDefault{\L}{OT1}
198 \DeclareTextSymbolDefault{\OE}{OT1}
199 \DeclareTextSymbolDefault{\O}{OT1}
200 %\DeclareTextSymbolDefault{\aa}{OT1}
201 \DeclareTextSymbolDefault{\ae}{OT1}
202 \DeclareTextSymbolDefault{\i}{OT1}
203 \DeclareTextSymbolDefault{\j}{OT1}
204 \DeclareTextSymbolDefault{\l}{OT1}
205 \DeclareTextSymbolDefault{\oe}{OT1}
206 \DeclareTextSymbolDefault{\o}{OT1}
207 \DeclareTextSymbolDefault{\ss}{OT1}
208 \DeclareTextSymbolDefault{\textdollar}{OT1}
209 \DeclareTextSymbolDefault{\textemdash}{OT1}
210 \DeclareTextSymbolDefault{\textendash}{OT1}
211 \DeclareTextSymbolDefault{\textexclamdown}{OT1}
212 %\DeclareTextSymbolDefault{\texthyphenchar}{OT1}
213 %\DeclareTextSymbolDefault{\texthyphen}{OT1}
214 \DeclareTextSymbolDefault{\textquestiondown}{OT1}
215 \DeclareTextSymbolDefault{\textquotedblleft}{OT1}
216 \DeclareTextSymbolDefault{\textquotedblright}{OT1}
217 \DeclareTextSymbolDefault{\textquoteleft}{OT1}
218 \DeclareTextSymbolDefault{\textquoteright}{OT1}
219 \DeclareTextSymbolDefault{\textsterling}{OT1}

```

Some symbols from OMS:

```

220 \DeclareTextSymbolDefault{\textasteriskcentered}{OMS}
221 \DeclareTextSymbolDefault{\textbackslash}{OMS}
222 \DeclareTextSymbolDefault{\textbar}{OMS}
223 \DeclareTextSymbolDefault{\textbardbl}{OMS}
224 \DeclareTextSymbolDefault{\textbraceleft}{OMS}
225 \DeclareTextSymbolDefault{\textbraceright}{OMS}
226 \DeclareTextSymbolDefault{\textbullet}{OMS}
227 \DeclareTextSymbolDefault{\textdaggerdbl}{OMS}
228 \DeclareTextSymbolDefault{\textdagger}{OMS}
229 \DeclareTextSymbolDefault{\textparagraph}{OMS}
230 \DeclareTextSymbolDefault{\textperiodcentered}{OMS}
231 \DeclareTextSymbolDefault{\textsection}{OMS}
232 \DeclareTextAccentDefault{\textcircled}{OMS}

```

Some symbols from OML:

```

233 \DeclareTextSymbolDefault{\textless}{OML}

```

```

234 \DeclareTextSymbolDefault{\textgreater}{OML}
235 \DeclareTextAccentDefault{\t}{OML}
    Some defaults we can fake.
    The interface for defining \copyright changed, it used to use \expandafter
    to add braces at the appropriate points.
236 \DeclareTextCommandDefault{\textcopyright}{\textcircled{c}}
237 % \expandafter\def\expandafter
238 %             \copyright\expandafter{\expandafter{\copyright}}

239 \DeclareTextCommandDefault{\textasciicircum}{\~{}}
240 \DeclareTextCommandDefault{\textasciitilde}{\~{}}
241 \DeclareTextCommandDefault{\textcompwordmark}{\leavevmode\kern\z@}
242 \DeclareTextCommandDefault{\textunderscore}{%
243   \leavevmode \kern.06em\vbox{\hrule\@width.3em}}

244 \DeclareTextCommandDefault{\textvisiblespace}{%
245   \mbox{\kern.06em\vrule \@height.3ex}%
246   \vbox{\hrule \@width.3em}%
247   \hbox{\vrule \@height.3ex}}

    Using \fontdimen3 in the next definition is some sort of a kludge (since it
    is the interword stretch) but it makes the ellipsis come out right in mono-spaced
    fonts too (since there it is zero).
248 \DeclareTextCommandDefault{\textellipsis}{%
249   .\kern\fontdimen3\font
250   .\kern\fontdimen3\font
251   .\kern\fontdimen3\font}

252 %\DeclareTextCommandDefault{\textregistered}{\textcircled{\scshape r}}
253 \DeclareTextCommandDefault{\textregistered}{\textcircled{%
254   \check@mathfonts\fontsize\sf@size\z@\math@fontsfalse\selectfont R}}
255 \DeclareTextCommandDefault{\texttrademark}{\textsuperscript{TM}}
256 \DeclareTextCommandDefault{\SS}{SS}

257 \DeclareTextCommandDefault{\textordfeminine}{\textsuperscript{a}}
258 \DeclareTextCommandDefault{\textordmasculine}{\textsuperscript{o}}

```

1.4.5 Math material

Some commands can be used in both text and math mode:

```

259 \DeclareRobustCommand{\$}{\ifmmode\mathdollar\else\textdollar\fi}
260 \DeclareRobustCommand{\{ }{\ifmmode\lbrace\else\textbraceleft\fi}
261 \DeclareRobustCommand{\} }{\ifmmode\rbrace\else\textbraceright\fi}
262 \DeclareRobustCommand{\P}{\ifmmode\mathparagraph\else\textparagraph\fi}
263 \DeclareRobustCommand{\S}{\ifmmode\mathsection\else\textsection\fi}
264 \DeclareRobustCommand{\dag}{\ifmmode{\dagger}\else\textdagger\fi}
265 \DeclareRobustCommand{\ddag}{\ifmmode{\ddagger}\else\textdaggerdbl\fi}

    For historical reasons \copyright needs {} around the definition in maths.
266 \DeclareRobustCommand{\_}{%

```

```

267 \ifmmode\nfss@text{\textunderscore}\else\textunderscore\fi}
268 \DeclareRobustCommand{\copyright}{%
269 \ifmmode{\nfss@text{\textcopyright}}\else\textcopyright\fi}
270 \DeclareRobustCommand{\pounds}{%
271 \ifmmode\mathsterling\else\textsterling\fi}
272 \DeclareRobustCommand{\dots}{%
273 \ifmmode\mathellipsis\else\textellipsis\fi}
274 \let\ldots\dots
275 </2ekernel | autoload>

```

1.5 Definitions for the OT1 encoding

The definitions for the ‘ \TeX text’ (OT1) encoding.

Declare the encoding.

```

276 <*OT1>
277 \DeclareFontEncoding{OT1}{}{}

```

Declare the accents.

```

278 \DeclareTextAccent{"}{OT1}{127}
279 \DeclareTextAccent{'}{OT1}{19}
280 \DeclareTextAccent{.}{OT1}{95}
281 \DeclareTextAccent{=}{OT1}{22}
282 \DeclareTextAccent{^}{OT1}{94}
283 \DeclareTextAccent{'}{OT1}{18}
284 \DeclareTextAccent{~}{OT1}{126}
285 \DeclareTextAccent{H}{OT1}{125}
286 \DeclareTextAccent{u}{OT1}{21}
287 \DeclareTextAccent{v}{OT1}{20}
288 \DeclareTextAccent{r}{OT1}{23}

```

Some accents have to be built by hand: Note that `\oalign` and `\o@lign` must be inside a group.

```

289 \DeclareTextCommand{\b}{OT1}[1]
290 {\hmode@bgroup\o@lign{\relax#1\crrc\hidewidth\sh@ft{29}%
291 \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
292 \DeclareTextCommand{\c}{OT1}[1]
293 {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
294 \else{\oalign{\unhbox\z@\crrc\hidewidth\char24\hidewidth}}\fi}
295 \DeclareTextCommand{\d}{OT1}[1]
296 {\hmode@bgroup
297 \o@lign{\relax#1\crrc\hidewidth\sh@ft{10}.\hidewidth}\egroup}

```

Declare the text symbols.

```

298 \DeclareTextSymbol{\AE}{OT1}{29}
299 \DeclareTextSymbol{\OE}{OT1}{30}
300 \DeclareTextSymbol{\O}{OT1}{31}
301 \DeclareTextSymbol{\ae}{OT1}{26}
302 \DeclareTextSymbol{\i}{OT1}{16}
303 \DeclareTextSymbol{\j}{OT1}{17}

```

```

304 \DeclareTextSymbol{\oe}{OT1}{27}
305 \DeclareTextSymbol{\o}{OT1}{28}
306 \DeclareTextSymbol{\ss}{OT1}{25}
307 \DeclareTextSymbol{\textemdash}{OT1}{124}
308 \DeclareTextSymbol{\textendash}{OT1}{123}

```

Using the ligatures helps with OT1 fonts that have `\textexclamdown` and `\textquestiondown` in unusual positions.

```

309 %\DeclareTextSymbol{\textexclamdown}{OT1}{60}
310 %\DeclareTextSymbol{\textquestiondown}{OT1}{62}
311 \DeclareTextCommand{\textexclamdown}{OT1}{!'}
312 \DeclareTextCommand{\textquestiondown}{OT1}{?'}
313 %\DeclareTextSymbol{\textthyphenchar}{OT1}{'\-}
314 %\DeclareTextSymbol{\textthyphen}{OT1}{'\-}
315 \DeclareTextSymbol{\textquotedblleft}{OT1}{92}
316 \DeclareTextSymbol{\textquotedblright}{OT1}{'\"}
317 \DeclareTextSymbol{\textquoteleft}{OT1}{'\'}
318 \DeclareTextSymbol{\textquoteright}{OT1}{'\'}

```

Some symbols which are faked from others:

```

319 % \DeclareTextCommand{\aa}{OT1}
320 %   {\accent23a}}
321 \DeclareTextCommand{\L}{OT1}
322   {\leavevmode\setbox\z@\hbox{L}\hb@xt@\wd\z@{\hss\@xxxii L}}
323 \DeclareTextCommand{\l}{OT1}
324   {\hmode@bgroup\@xxxii l\egroup}
325 % \DeclareTextCommand{\AA}{OT1}
326 %   {\leavevmode\setbox\z@\hbox{h}\dimen@ht\z@\advance\dimen@-1ex%
327 %     \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

In the OT1 encoding Å has a hand-crafted definition, so we have here the first recorded explicit use of `\DeclareTextCompositeCommand`.

```

328 \DeclareTextCompositeCommand{\r}{OT1}{A}
329   {\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
330     \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

In the OT1 encoding, £ and \$ share a slot.

```

331 \DeclareTextCommand{\textdollar}{OT1}{\hmode@bgroup
332   \ifdim \fontdimen\@ne\font >\z@
333     \slshape
334   \else
335     \upshape
336   \fi
337   \char'\$\egroup}
338 \DeclareTextCommand{\textsterling}{OT1}{\hmode@bgroup
339   \ifdim \fontdimen\@ne\font >\z@
340     \itshape
341   \else
342     \fontshape{ui}\selectfont
343   \fi
344   \char'\$\egroup}

```

Here we are adding some more composite commands to the OT1 encoding. This makes the use of certain accents with `i` compatible with their use with the T1 encoding; this enables them to become true L^AT_EX internal representations. However, it will make these accents work a little less fast since a check will always be made for the existence of a composite.

```

345 \DeclareTextComposite{\.}{OT1}{i}{'\i}
346 \DeclareTextComposite{\.}{OT1}{\i}{'\i}
347 \DeclareTextCompositeCommand{\'}{OT1}{i}{\@tabacckludge'\i}
348 \DeclareTextCompositeCommand{\'}{OT1}{\i}{\@tabacckludge'\i}
349 \DeclareTextCompositeCommand{\^}{OT1}{i}{\^i}
350 \DeclareTextCompositeCommand{\^}{OT1}{\i}{\^i}
351 </OT1>

```

1.6 Definitions for the T1 encoding

The definitions for the ‘Extended T_EX text’ (T1) encoding.

Declare the encoding.

```

352 <*T1>
353 \DeclareFontEncoding{T1}{}{}

```

Declare the accents.

```

354 \DeclareTextAccent{\'}{T1}{0}
355 \DeclareTextAccent{\'}{T1}{1}
356 \DeclareTextAccent{\^}{T1}{2}
357 \DeclareTextAccent{\^}{T1}{3}
358 \DeclareTextAccent{\"}{T1}{4}
359 \DeclareTextAccent{\H}{T1}{5}
360 \DeclareTextAccent{\R}{T1}{6}
361 \DeclareTextAccent{\V}{T1}{7}
362 \DeclareTextAccent{\U}{T1}{8}
363 \DeclareTextAccent{\=} {T1}{9}
364 \DeclareTextAccent{\.}{T1}{10}

```

Some accents have to be built by hand. Note that `\oalign` and `\o@lign` must be inside a group.

```

365 \DeclareTextCommand{\b}{T1}[1]
366   {\hmode@bgroup\o@lign{\relax#1\crrc\hidewidth\sh@ft{29}%
367     \vbox to.2ex{\hbox{\char9}\vss}\hidewidth}\egroup}
368 \DeclareTextCommand{\c}{T1}[1]
369   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent11 #1%
370     \else{\oalign{\unhbox\z@\crrc
371       \hidewidth\char11\hidewidth}}\fi}
372 \DeclareTextCommand{\d}{T1}[1]
373   {\hmode@bgroup
374     \o@lign{\relax#1\crrc\hidewidth\sh@ft{10}.\hidewidth}\egroup}
375 \DeclareTextCommand{\k}{T1}[1]
376   {\hmode@bgroup\oalign{\null#1\crrc\hidewidth\char12}\egroup}
377 \DeclareTextCommand{\textogonekcentered}{T1}[1]
378   {\hmode@bgroup\oalign{\null#1\crrc\hidewidth\char12\hidewidth}\egroup}

```

Some symbols are constructed.

Slot 24 contains a small circle intended for construction of these two glyphs.

```
379 \DeclareTextCommand{\textperthousand}{T1}
380   {\%\char 24 }           % space or 'relax as delimiter?
381 \DeclareTextCommand{\textpertenthousand}{T1}
382   {\%\char 24\char 24 } % space or 'relax as delimiter?
```

Declare the text symbols.

```
383 %\DeclareTextSymbol{\AA}{T1}{197}
384 \DeclareTextSymbol{\AE}{T1}{198}
385 \DeclareTextSymbol{\DH}{T1}{208}
386 \DeclareTextSymbol{\DJ}{T1}{208}
387 \DeclareTextSymbol{\L}{T1}{138}
388 \DeclareTextSymbol{\NG}{T1}{141}
389 \DeclareTextSymbol{\OE}{T1}{215}
390 \DeclareTextSymbol{\O}{T1}{216}
391 \DeclareTextSymbol{\SS}{T1}{223}
392 \DeclareTextSymbol{\TH}{T1}{222}
393 %\DeclareTextSymbol{\aa}{T1}{229}
394 \DeclareTextSymbol{\ae}{T1}{230}
395 \DeclareTextSymbol{\dh}{T1}{240}
396 \DeclareTextSymbol{\dj}{T1}{158}
397 \DeclareTextSymbol{\guillemotleft}{T1}{19}
398 \DeclareTextSymbol{\guillemotright}{T1}{20}
399 \DeclareTextSymbol{\guilsinglleft}{T1}{14}
400 \DeclareTextSymbol{\guilsinglright}{T1}{15}
401 \DeclareTextSymbol{\i}{T1}{25}
402 \DeclareTextSymbol{\j}{T1}{26}
403 \DeclareTextSymbol{\l}{T1}{170}
404 \DeclareTextSymbol{\ng}{T1}{173}
405 \DeclareTextSymbol{\oe}{T1}{247}
406 \DeclareTextSymbol{\o}{T1}{248}
407 \DeclareTextSymbol{\quotedblbase}{T1}{18}
408 \DeclareTextSymbol{\quotesinglbase}{T1}{13}
409 \DeclareTextSymbol{\ss}{T1}{255}
410 \DeclareTextSymbol{\textasciicircum}{T1}{'\` }
411 \DeclareTextSymbol{\textasciitilde}{T1}{'\~ }
412 \DeclareTextSymbol{\textbackslash}{T1}{'\ \ }
413 \DeclareTextSymbol{\textbar}{T1}{'| }
414 \DeclareTextSymbol{\textbraceleft}{T1}{'\{ }
415 \DeclareTextSymbol{\textbraceright}{T1}{'\} }
416 \DeclareTextSymbol{\textcompwordmark}{T1}{23}
417 \DeclareTextSymbol{\textdollar}{T1}{'\$ }
418 \DeclareTextSymbol{\textemdash}{T1}{22}
419 \DeclareTextSymbol{\textendash}{T1}{21}
420 \DeclareTextSymbol{\textexclamdown}{T1}{189}
421 \DeclareTextSymbol{\textgreater}{T1}{'\> }
422 %\DeclareTextSymbol{\textthyphenchar}{T1}{127}
423 %\DeclareTextSymbol{\textthyphen}{T1}{'\- }
```

```

424 \DeclareTextSymbol{\textless}{T1}{'\<}
425 \DeclareTextSymbol{\textquestiondown}{T1}{190}
426 \DeclareTextSymbol{\textquotedblleft}{T1}{16}
427 \DeclareTextSymbol{\textquotedblright}{T1}{17}
428 \DeclareTextSymbol{\textquotedbl}{T1}{'" }
429 \DeclareTextSymbol{\textquoteleft}{T1}{'\'}
430 \DeclareTextSymbol{\textquoteright}{T1}{'\'}
431 \DeclareTextSymbol{\textsection}{T1}{159}
432 \DeclareTextSymbol{\textsterling}{T1}{191}
433 \DeclareTextSymbol{\textunderscore}{T1}{95}
434 \DeclareTextSymbol{\textvisiblespace}{T1}{32}
435 \DeclareTextSymbol{\th}{T1}{254}

```

Declare the composites.

```

436 \DeclareTextComposite{\.}{T1}{i}{'\i}
437 \DeclareTextComposite{\.}{T1}{\i}{'\i}
"80 = 128
438 \DeclareTextComposite{\u}{T1}{A}{128}
439 \DeclareTextComposite{\k}{T1}{A}{129}
440 \DeclareTextComposite{\'}{T1}{C}{130}
441 \DeclareTextComposite{\v}{T1}{C}{131}
442 \DeclareTextComposite{\v}{T1}{D}{132}
443 \DeclareTextComposite{\v}{T1}{E}{133}
444 \DeclareTextComposite{\k}{T1}{E}{134}
445 \DeclareTextComposite{\u}{T1}{G}{135}
"88 = 136
446 \DeclareTextComposite{\'}{T1}{L}{136}
447 \DeclareTextComposite{\v}{T1}{L}{137}
448 \DeclareTextComposite{\'}{T1}{N}{139}
449 \DeclareTextComposite{\v}{T1}{N}{140}
450 \DeclareTextComposite{\H}{T1}{O}{142}
451 \DeclareTextComposite{\'}{T1}{R}{143}
"90 = 144
452 \DeclareTextComposite{\v}{T1}{R}{144}
453 \DeclareTextComposite{\'}{T1}{S}{145}
454 \DeclareTextComposite{\v}{T1}{S}{146}
455 \DeclareTextComposite{\c}{T1}{S}{147}
456 \DeclareTextComposite{\v}{T1}{T}{148}
457 \DeclareTextComposite{\c}{T1}{T}{149}
458 \DeclareTextComposite{\H}{T1}{U}{150}
459 \DeclareTextComposite{\r}{T1}{U}{151}
"98 = 152
460 \DeclareTextComposite{\"}{T1}{Y}{152}
461 \DeclareTextComposite{\'}{T1}{Z}{153}
462 \DeclareTextComposite{\v}{T1}{Z}{154}
463 \DeclareTextComposite{\.}{T1}{Z}{155}
464 \DeclareTextComposite{\.}{T1}{I}{157}

```

```

”A0 = 160
465 \DeclareTextComposite{\u}{T1}{a}{160}
466 \DeclareTextComposite{\k}{T1}{a}{161}
467 \DeclareTextComposite{\'}{T1}{c}{162}
468 \DeclareTextComposite{\v}{T1}{c}{163}
469 \DeclareTextComposite{\v}{T1}{d}{164}
470 \DeclareTextComposite{\v}{T1}{e}{165}
471 \DeclareTextComposite{\k}{T1}{e}{166}
472 \DeclareTextComposite{\u}{T1}{g}{167}
”A8 = 168
473 \DeclareTextComposite{\'}{T1}{l}{168}
474 \DeclareTextComposite{\v}{T1}{l}{169}
475 \DeclareTextComposite{\'}{T1}{n}{171}
476 \DeclareTextComposite{\v}{T1}{n}{172}
477 \DeclareTextComposite{\H}{T1}{o}{174}
478 \DeclareTextComposite{\'}{T1}{r}{175}
”B0 = 176
479 \DeclareTextComposite{\v}{T1}{r}{176}
480 \DeclareTextComposite{\'}{T1}{s}{177}
481 \DeclareTextComposite{\v}{T1}{s}{178}
482 \DeclareTextComposite{\c}{T1}{s}{179}
483 \DeclareTextComposite{\v}{T1}{t}{180}
484 \DeclareTextComposite{\c}{T1}{t}{181}
485 \DeclareTextComposite{\H}{T1}{u}{182}
486 \DeclareTextComposite{\r}{T1}{u}{183}
”B8 = 184
487 \DeclareTextComposite{\"}{T1}{y}{184}
488 \DeclareTextComposite{\'}{T1}{z}{185}
489 \DeclareTextComposite{\v}{T1}{z}{186}
490 \DeclareTextComposite{\.}{T1}{z}{187}
”C0 = 192
491 \DeclareTextComposite{\'}{T1}{A}{192}
492 \DeclareTextComposite{\'}{T1}{A}{193}
493 \DeclareTextComposite{\^}{T1}{A}{194}
494 \DeclareTextComposite{\~}{T1}{A}{195}
495 \DeclareTextComposite{\"}{T1}{A}{196}
496 \DeclareTextComposite{\r}{T1}{A}{197}
497 \DeclareTextComposite{\c}{T1}{C}{199}
”C8 = 200
498 \DeclareTextComposite{\'}{T1}{E}{200}
499 \DeclareTextComposite{\'}{T1}{E}{201}
500 \DeclareTextComposite{\^}{T1}{E}{202}
501 \DeclareTextComposite{\"}{T1}{E}{203}
502 \DeclareTextComposite{\'}{T1}{I}{204}
503 \DeclareTextComposite{\'}{T1}{I}{205}
504 \DeclareTextComposite{\^}{T1}{I}{206}
505 \DeclareTextComposite{\"}{T1}{I}{207}

```

"D0 = 208

```

506 \DeclareTextComposite{\~}{T1}{N}{209}
507 \DeclareTextComposite{\'}{T1}{O}{210}
508 \DeclareTextComposite{\'}{T1}{O}{211}
509 \DeclareTextComposite{\^}{T1}{O}{212}
510 \DeclareTextComposite{\~}{T1}{O}{213}
511 \DeclareTextComposite{\"}{T1}{O}{214}

```

"D8 = 216

```

512 \DeclareTextComposite{\'}{T1}{U}{217}
513 \DeclareTextComposite{\'}{T1}{U}{218}
514 \DeclareTextComposite{\^}{T1}{U}{219}
515 \DeclareTextComposite{\~}{T1}{U}{220}
516 \DeclareTextComposite{\'}{T1}{Y}{221}

```

"E0 = 224

```

517 \DeclareTextComposite{\'}{T1}{a}{224}
518 \DeclareTextComposite{\'}{T1}{a}{225}
519 \DeclareTextComposite{\^}{T1}{a}{226}
520 \DeclareTextComposite{\~}{T1}{a}{227}
521 \DeclareTextComposite{\"}{T1}{a}{228}
522 \DeclareTextComposite{\r}{T1}{a}{229}
523 \DeclareTextComposite{\c}{T1}{c}{231}

```

"E8 = 232

```

524 \DeclareTextComposite{\'}{T1}{e}{232}
525 \DeclareTextComposite{\~}{T1}{e}{233}
526 \DeclareTextComposite{\^}{T1}{e}{234}
527 \DeclareTextComposite{\"}{T1}{e}{235}
528 \DeclareTextComposite{\'}{T1}{i}{236}
529 \DeclareTextComposite{\'}{T1}{\i}{236}
530 \DeclareTextComposite{\~}{T1}{i}{237}
531 \DeclareTextComposite{\'}{T1}{\i}{237}
532 \DeclareTextComposite{\^}{T1}{i}{238}
533 \DeclareTextComposite{\~}{T1}{\i}{238}
534 \DeclareTextComposite{\"}{T1}{i}{239}
535 \DeclareTextComposite{\~}{T1}{\i}{239}

```

"F0 = 240

```

536 \DeclareTextComposite{\~}{T1}{n}{241}
537 \DeclareTextComposite{\'}{T1}{o}{242}
538 \DeclareTextComposite{\'}{T1}{o}{243}
539 \DeclareTextComposite{\^}{T1}{o}{244}
540 \DeclareTextComposite{\~}{T1}{o}{245}
541 \DeclareTextComposite{\"}{T1}{o}{246}

```

"F8 = 248

```

542 \DeclareTextComposite{\'}{T1}{u}{249}
543 \DeclareTextComposite{\'}{T1}{u}{250}
544 \DeclareTextComposite{\^}{T1}{u}{251}
545 \DeclareTextComposite{\~}{T1}{u}{252}
546 \DeclareTextComposite{\'}{T1}{y}{253}

```

```

547 \DeclareTextCompositeCommand{\k}{T1}{o}{\textogonekcentered{o}}
548 \DeclareTextCompositeCommand{\k}{T1}{0}{\textogonekcentered{0}}
549 </T1>

```

1.7 Definitions for the OMS encoding

The definitions for the ‘ \TeX math symbol’ (OMS) encoding. Even though this is meant to be a math font, it includes some of the standard \LaTeX text symbols.

Declare the encoding.

```

550 <*OMS>
551 \DeclareFontEncoding{OMS}{}{}

Declare the symbols.

552 % \changes{v1.99}{2004/02/02}{Added \cs{textbigcircle}}
553 % Note that slot 13 has in places been named |\Orb|: please root
554 % out and destroy this impolity wherever you find it!
555 % \begin{macrocode}
556 \DeclareTextSymbol{\textasteriskcentered}{OMS}{3} % "03
557 \DeclareTextSymbol{\textbackslash}{OMS}{110} % "6E
558 \DeclareTextSymbol{\textbar}{OMS}{106} % "6A
559 \DeclareTextSymbol{\textbardbl}{OMS}{107} % "6B
560 \DeclareTextSymbol{\textbraceleft}{OMS}{102} % "66
561 \DeclareTextSymbol{\textbraceright}{OMS}{103} % "67
562 \DeclareTextSymbol{\textbullet}{OMS}{15} % "0F
563 \DeclareTextSymbol{\textdaggerdbl}{OMS}{122} % "7A
564 \DeclareTextSymbol{\textdagger}{OMS}{121} % "79
565 \DeclareTextSymbol{\textparagraph}{OMS}{123} % "7B
566 \DeclareTextSymbol{\textperiodcentered}{OMS}{1} % "01
567 \DeclareTextSymbol{\textsection}{OMS}{120} % "78
568 \DeclareTextSymbol{\textbigcircle}{OMS}{13} % "0D
569 \DeclareTextCommand{\textcircled}{OMS}[1]{\hmode@bgroup
570 \oalign{%
571 \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
572 \char 13 % "0D
573 }%
574 \egroup}
575 </OMS>

```

1.8 Definitions for the OML encoding

The definitions for the ‘ \TeX math italic’ (OML) encoding. Even though this is meant to be a math font, it includes some of the standard \LaTeX text symbols.

Declare the encoding.

```

576 <*OML>
577 \DeclareFontEncoding{OML}{}{}

Declare the symbols.

578 \DeclareTextSymbol{\textless}{OML}{'\<}

```

```

579 \DeclareTextSymbol{\textgreater}{OML}{\>}
580 \DeclareTextAccent{\t}{OML}{127} % "7F
581 \end{OML}

```

1.9 Definitions for the OT4 encoding

These definitions are for the Polish extension to the ‘ \TeX text’ (OT1) encoding. This encoding was created by B. Jackowski and M. Ryćko for use with the Polish version of Computer Modern and Computer Concrete. In positions 0–127 it is identical to OT1 but it contains some additional characters in the upper half. The \LaTeX support was developed by Mariusz Olko.

The PL fonts that use it are available as follows:

Metafont sources <ftp://ftp.gust.org.pl/TeX/language/polish/pl-mf.zip>;

Font files <ftp://ftp.gust.org.pl/TeX/language/polish/pl-tfm.zip>.

Declare the encoding.

```

582 (*OT4)
583 \DeclareFontEncoding{OT4}{}{}
584 \DeclareFontSubstitution{OT4}{cmr}{m}{n}

```

Declare the accents.

```

585 \DeclareTextAccent{"}{OT4}{127}
586 \DeclareTextAccent{'}{OT4}{19}
587 \DeclareTextAccent{.}{OT4}{95}
588 \DeclareTextAccent{=}{OT4}{22}
589 \DeclareTextAccent{^}{OT4}{94}
590 \DeclareTextAccent{'}{OT4}{18}
591 \DeclareTextAccent{~}{OT4}{126}
592 \DeclareTextAccent{H}{OT4}{125}
593 \DeclareTextAccent{u}{OT4}{21}
594 \DeclareTextAccent{v}{OT4}{20}
595 \DeclareTextAccent{r}{OT4}{23}

```

The ogonek accent is available only under a e A & E. But we have to provide some definition for $\backslash k$. Some other accents have to be built by hand as in OT1:

```

596 \DeclareTextCommand{\k}{OT4}[1]{%
597   \TextSymbolUnavailable{\k{#1}}#1}
598 \DeclareTextCommand{\b}{OT4}[1]
599   {\hmode@bgroup\o@lign{\relax#1\crrc\hidewidth\sh@ft{29}%
600    \vbox to.2ex{\hbox{\char22}\vss}\hidewidth}\egroup}
601 \DeclareTextCommand{\c}{OT4}[1]
602   {\leavevmode\setbox\z@\hbox{#1}\ifdim\ht\z@=1ex\accent24 #1%
603    \else{\oalign{\unhbox\z@\crrc\hidewidth\char24\hidewidth}}\fi}
604 \DeclareTextCommand{\d}{OT4}[1]
605   {\hmode@bgroup
606    \o@lign{\relax#1\crrc\hidewidth\sh@ft{10}.\hidewidth}\egroup}

```

Declare the text symbols.

```

607 \DeclareTextSymbol{\AE}{OT4}{29}
608 \DeclareTextSymbol{\OE}{OT4}{30}
609 \DeclareTextSymbol{\0}{OT4}{31}

```

```

610 \DeclareTextSymbol{\L}{OT4}{138}
611 \DeclareTextSymbol{\ae}{OT4}{26}
612 \DeclareTextSymbol{\guillemotleft}{OT4}{174}
613 \DeclareTextSymbol{\guillemotright}{OT4}{175}
614 \DeclareTextSymbol{\i}{OT4}{16}
615 \DeclareTextSymbol{\j}{OT4}{17}
616 \DeclareTextSymbol{\l}{OT4}{170}
617 \DeclareTextSymbol{\o}{OT4}{28}
618 \DeclareTextSymbol{\oe}{OT4}{27}
619 \DeclareTextSymbol{\quotedblbase}{OT4}{255}
620 \DeclareTextSymbol{\ss}{OT4}{25}
621 \DeclareTextSymbol{\textendash}{OT4}{124}
622 \DeclareTextSymbol{\textendash}{OT4}{123}
623 \DeclareTextSymbol{\textexclamdown}{OT4}{60}
624 %\DeclareTextSymbol{\textthyphenchar}{OT4}{'\-}
625 %\DeclareTextSymbol{\textthyphen}{OT4}{'\-}
626 \DeclareTextSymbol{\textquestiondown}{OT4}{62}
627 \DeclareTextSymbol{\textquotedblleft}{OT4}{92}
628 \DeclareTextSymbol{\textquotedblright}{OT4}{'\'}
629 \DeclareTextSymbol{\textquotelleft}{OT4}{'\'}
630 \DeclareTextSymbol{\textquoteright}{OT4}{'\'}

```

Definition for Å as in OT1:

```

631 \DeclareTextCompositeCommand{\r}{OT4}{A}
632   {\leavevmode\setbox\z@\hbox{!}\dimen@ht\z@\advance\dimen@-1ex%
633    \rlap{\raise.67\dimen@\hbox{\char23}}A}

```

In the OT4 encoding, £ and \$ share a slot.

```

634 \DeclareTextCommand{\textdollar}{OT4}{\hmode@bgroup
635   \ifdim \fontdimen\@ne\font >\z@
636     \slshape
637   \else
638     \upshape
639   \fi
640   \char'\$\egroup}
641 \DeclareTextCommand{\textsterling}{OT4}{\hmode@bgroup
642   \ifdim \fontdimen\@ne\font >\z@
643     \itshape
644   \else
645     \fontshape{ui}\selectfont
646   \fi
647   \char'\$\egroup}

```

Declare the composites.

```

648 \DeclareTextComposite{\k}{OT4}{A}{129}
649 \DeclareTextComposite{\'}{OT4}{C}{130}
650 \DeclareTextComposite{\k}{OT4}{E}{134}
651 \DeclareTextComposite{\'}{OT4}{N}{139}
652 \DeclareTextComposite{\'}{OT4}{S}{145}
653 \DeclareTextComposite{\'}{OT4}{Z}{153}
654 \DeclareTextComposite{\.}{OT4}{Z}{155}

```

```

655 \DeclareTextComposite{\k}{OT4}{a}{161}
656 \DeclareTextComposite{\'}{OT4}{c}{162}
657 \DeclareTextComposite{\k}{OT4}{e}{166}
658 \DeclareTextComposite{\'}{OT4}{n}{171}
659 \DeclareTextComposite{\'}{OT4}{s}{177}
660 \DeclareTextComposite{\'}{OT4}{z}{185}
661 \DeclareTextComposite{\.}{OT4}{z}{187}
662 \DeclareTextComposite{\'}{OT4}{0}{211}
663 \DeclareTextComposite{\'}{OT4}{o}{243}
664 (/OT4)

```

1.10 Definitions for the TS1 encoding

```

665 (*TS1)
666 \DeclareFontEncoding{TS1}{}{}
667 \DeclareFontSubstitution{TS1}{cmr}{m}{n}

```

Some accents have to be built by hand. Note that `\ooalign` and `\o@lign` must be inside a group.

```

668 \DeclareTextCommand{\capitalcedilla}{TS1}[1]
669   {\hmode@bgroup
670    \ooalign{\null#1\cr\hidewidth\char11\hidewidth}\egroup}
671 \DeclareTextCommand{\capitalogonek}{TS1}[1]
672   {\hmode@bgroup
673    \ooalign{\null#1\cr\hidewidth\char12\hidewidth}\egroup}

```

Accents for capital letters.

These commands can be used by the end user either directly or through definitions of the type

```
\DeclareTextCompositeCommand{\'}{T1}{X}{\capitalacute X}
```

None of the latter definitions are provided by default, since they are probably rarely used.

”00 = 0

```

674 \DeclareTextAccent{\capitalgrave}{TS1}{0}
675 \DeclareTextAccent{\capitalacute}{TS1}{1}
676 \DeclareTextAccent{\capitalcircumflex}{TS1}{2}
677 \DeclareTextAccent{\capitaltilde}{TS1}{3}
678 \DeclareTextAccent{\capitaldieresis}{TS1}{4}
679 \DeclareTextAccent{\capitalhungarumlaut}{TS1}{5}
680 \DeclareTextAccent{\capitalring}{TS1}{6}
681 \DeclareTextAccent{\capitalcaron}{TS1}{7}

```

”08 = 8

```

682 \DeclareTextAccent{\capitalbreve}{TS1}{8}
683 \DeclareTextAccent{\capitalmacron}{TS1}{9}
684 \DeclareTextAccent{\capitaldotaccent}{TS1}{10}

```

Tie accents.

The tie accent was borrowed from the `cmmi` font. The `tc` fonts now provide four tie accents, the first two are done in the classical way with assymmetric glyphs hanging out of their boxes; the new ties are centered in their boxes like all other accents. They need a name: please tell us if you know what to call them.

” =

```
685 \DeclareTextAccent{\t}{TS1}{26}
686 \DeclareTextAccent{\capitaltie}{TS1}{27}
687 \DeclareTextAccent{\newtie}{TS1}{28}
688 \DeclareTextAccent{\capitalnewtie}{TS1}{29}
```

Compound word marks.

The text companion fonts contain two compound word marks of different heights, one has `cap_height`, the other `asc_height`.

```
689 \DeclareTextSymbol{\textcapitalcompwordmark}{TS1}{23}
690 \DeclareTextSymbol{\textascendercompwordmark}{TS1}{31}
```

The text companion symbols.

```
691 \DeclareTextSymbol{\textquotestraightbase}{TS1}{13}
”10 = 16
692 \DeclareTextSymbol{\textquotestraightdblbase}{TS1}{18}
693 \DeclareTextSymbol{\texttwelveudash}{TS1}{21}
694 \DeclareTextSymbol{\textthreequartersemdash}{TS1}{22}
”18 = 24
695 \DeclareTextSymbol{\textleftarrow}{TS1}{24}
696 \DeclareTextSymbol{\textrightarrow}{TS1}{25}
”20 = 32
697 \DeclareTextSymbol{\textblank}{TS1}{32}
698 \DeclareTextSymbol{\textdollar}{TS1}{36}
699 \DeclareTextSymbol{\textquotesingle}{TS1}{39}
”28 = 40
700 \DeclareTextSymbol{\textasteriskcentered}{TS1}{42}
```

Note that `'054` is a comma and `'056` is a full stop: these make numbers using `oldstyle` digits easier to input.

```
701 \DeclareTextSymbol{\textdblhyphen}{TS1}{45}
702 \DeclareTextSymbol{\textfractionssolidus}{TS1}{47}
```

Oldstyle digits.

”30 = 48

```
703 \DeclareTextSymbol{\textzerooldstyle}{TS1}{48}
704 \DeclareTextSymbol{\textoneoldstyle}{TS1}{49}
705 \DeclareTextSymbol{\texttwooldstyle}{TS1}{50}
706 \DeclareTextSymbol{\textthreeoldstyle}{TS1}{51}
707 \DeclareTextSymbol{\textfouroldstyle}{TS1}{52}
708 \DeclareTextSymbol{\textfiveoldstyle}{TS1}{53}
709 \DeclareTextSymbol{\textsixoldstyle}{TS1}{54}
710 \DeclareTextSymbol{\textsevenoldstyle}{TS1}{55}
```

```

"38 = 56
711 \DeclareTextSymbol{\texteightoldstyle}{TS1}{56}
712 \DeclareTextSymbol{\textnineoldstyle}{TS1}{57}
    More text companion symbols.
713 \DeclareTextSymbol{\textlangle}{TS1}{60}
714 \DeclareTextSymbol{\textminus}{TS1}{61}
715 \DeclareTextSymbol{\textrangle}{TS1}{62}
"48 = 72
716 \DeclareTextSymbol{\textmho}{TS1}{77}
    The big circle is here to define the command \textcircled. Formerly it was
    taken from the cmsy font.
717 \DeclareTextSymbol{\textbigcircle}{TS1}{79}
718 \DeclareTextCommand{\textcircled}{TS1}[1]{\hmode@bgroup
719   \oalign{%
720     \hfil \raise .07ex\hbox {\upshape#1}\hfil \crcr
721     \char 79   % '117 = "4F
722   }%
723 \egroup}
    More text companion symbols.
"50 = 80
724 \DeclareTextSymbol{\textohm}{TS1}{87}
"58 = 88
725 \DeclareTextSymbol{\textlbrackdbl}{TS1}{91}
726 \DeclareTextSymbol{\textrbrackdbl}{TS1}{93}
727 \DeclareTextSymbol{\textuparrow}{TS1}{94}
728 \DeclareTextSymbol{\textdownarrow}{TS1}{95}
"60 = 96
729 \DeclareTextSymbol{\textasciigrave}{TS1}{96}
730 \DeclareTextSymbol{\textborn}{TS1}{98}
731 \DeclareTextSymbol{\textdivorced}{TS1}{99}
732 \DeclareTextSymbol{\textdied}{TS1}{100}
"68 = 104
733 \DeclareTextSymbol{\textleaf}{TS1}{108}
734 \DeclareTextSymbol{\textmarried}{TS1}{109}
735 \DeclareTextSymbol{\textmusicalnote}{TS1}{110}
"78 = 120
736 \DeclareTextSymbol{\texttildelow}{TS1}{126}
    This glyph, \textdblhyphenchar is hanging, like the hyphenchar of the ec
    fonts.
737 \DeclareTextSymbol{\textdblhyphenchar}{TS1}{127}
"80 = 128
738 \DeclareTextSymbol{\textasciibreve}{TS1}{128}
739 \DeclareTextSymbol{\textasciicaron}{TS1}{129}

```

This next glyph is *not* the same as `\textquotedbl`.

```
740 \DeclareTextSymbol{\textacutedbl}{TS1}{130}
741 \DeclareTextSymbol{\textgravedbl}{TS1}{131}
742 \DeclareTextSymbol{\textdagger}{TS1}{132}
743 \DeclareTextSymbol{\textdaggerdbl}{TS1}{133}
744 \DeclareTextSymbol{\textbardbl}{TS1}{134}
745 \DeclareTextSymbol{\textperthousand}{TS1}{135}
"88 = 136
746 \DeclareTextSymbol{\textbullet}{TS1}{136}
747 \DeclareTextSymbol{\textcelsius}{TS1}{137}
748 \DeclareTextSymbol{\textdollaroldstyle}{TS1}{138}
749 \DeclareTextSymbol{\textcentoldstyle}{TS1}{139}
750 \DeclareTextSymbol{\textflorin}{TS1}{140}
751 \DeclareTextSymbol{\textcolonmonetary}{TS1}{141}
752 \DeclareTextSymbol{\textwon}{TS1}{142}
753 \DeclareTextSymbol{\textnaira}{TS1}{143}
"90 = 144
754 \DeclareTextSymbol{\textguarani}{TS1}{144}
755 \DeclareTextSymbol{\textpeso}{TS1}{145}
756 \DeclareTextSymbol{\textlira}{TS1}{146}
757 \DeclareTextSymbol{\textrecipe}{TS1}{147}
758 \DeclareTextSymbol{\textinterrobang}{TS1}{148}
759 \DeclareTextSymbol{\textinterrobangdown}{TS1}{149}
760 \DeclareTextSymbol{\textdong}{TS1}{150}
761 \DeclareTextSymbol{\texttrademark}{TS1}{151}
"98 = 152
762 \DeclareTextSymbol{\textpertenthousand}{TS1}{152}
763 \DeclareTextSymbol{\textpilcrow}{TS1}{153}
764 \DeclareTextSymbol{\textbaht}{TS1}{154}
765 \DeclareTextSymbol{\textnumero}{TS1}{155}
This next name may change. For the following sign we know only a german name,
which is abzüglich. The meaning is something like "commercial minus". An ASCII
ersatz is ./ (dot slash dot). The temporary English name is \textdiscount.
766 \DeclareTextSymbol{\textdiscount}{TS1}{156}
767 \DeclareTextSymbol{\textestimated}{TS1}{157}
768 \DeclareTextSymbol{\textopenbullet}{TS1}{158}
769 \DeclareTextSymbol{\textservicemark}{TS1}{159}
"A0 = 160
770 \DeclareTextSymbol{\textlquill}{TS1}{160}
771 \DeclareTextSymbol{\textrquill}{TS1}{161}
772 \DeclareTextSymbol{\textcent}{TS1}{162}
773 \DeclareTextSymbol{\textsterling}{TS1}{163}
774 \DeclareTextSymbol{\textcurrency}{TS1}{164}
775 \DeclareTextSymbol{\textyen}{TS1}{165}
776 \DeclareTextSymbol{\textbrokenbar}{TS1}{166}
777 \DeclareTextSymbol{\textsection}{TS1}{167}
```

```

”A8 = 168
778 \DeclareTextSymbol{\textasciidieresis}{TS1}{168}
779 \DeclareTextSymbol{\textcopyright}{TS1}{169}
780 \DeclareTextSymbol{\textordfeminine}{TS1}{170}
781 \DeclareTextSymbol{\textcopyleft}{TS1}{171}
782 \DeclareTextSymbol{\textlnot}{TS1}{172}

    The meaning of the circled-P is “sound recording copyright”.

783 \DeclareTextSymbol{\textcircledP}{TS1}{173}
784 \DeclareTextSymbol{\textregistered}{TS1}{174}
785 \DeclareTextSymbol{\textasciimacron}{TS1}{175}

”B0 = 176
786 \DeclareTextSymbol{\textdegree}{TS1}{176}
787 \DeclareTextSymbol{\textpm}{TS1}{177}
788 \DeclareTextSymbol{\texttwosuperior}{TS1}{178}
789 \DeclareTextSymbol{\textthreesuperior}{TS1}{179}
790 \DeclareTextSymbol{\textasciiacute}{TS1}{180}
791 \DeclareTextSymbol{\textmu}{TS1}{181} % micro sign
792 \DeclareTextSymbol{\textparagraph}{TS1}{182}
793 \DeclareTextSymbol{\textperiodcentered}{TS1}{183}

”B8 = 184
794 \DeclareTextSymbol{\textreferencemark}{TS1}{184}
795 \DeclareTextSymbol{\textonesuperior}{TS1}{185}
796 \DeclareTextSymbol{\textordmasculine}{TS1}{186}
797 \DeclareTextSymbol{\textsurd}{TS1}{187}
798 \DeclareTextSymbol{\textonequarter}{TS1}{188}
799 \DeclareTextSymbol{\textonehalf}{TS1}{189}
800 \DeclareTextSymbol{\textthreequarters}{TS1}{190}
801 \DeclareTextSymbol{\texteuro}{TS1}{191}

”E0 = 208
802 \DeclareTextSymbol{\texttimes}{TS1}{214}

”F0 = 240
803 \DeclareTextSymbol{\textdiv}{TS1}{246}
804 \{/TS1}

```

2 Package files

This file now also contains some packages that provide access to the more specialised encodings.

2.1 The fontenc package

This package allows authors to specify which encodings they will use. For each encoding F00, the package looks to see if the encoding F00 has already been declared. If it has not, the file `foenc.def` is loaded. The default encoding is set to be F00.

In addition the package at the moment contains extra code to extend the `\@uclclist` (list of upper/lower case pairs) for encodings that involve cyrillic characters. THIS IS A TEMPORARY SOLUTION and will not stay this way forever (or so we hope) but right now we are missing a proper interface for this and didn't wanted to rush it.

805 `*package)`

Here we define a macro that extends the `\@uclclist` if needed and afterwards turns itself in a noop.

```
806 \def\update@uclc@with@cyrillic{%
807 \expandafter\def\expandafter\@uclclist\expandafter
808 {\@uclclist
809 \cyr\CYRA\cyrabhch\CYRABHCH\cyrabhchdsc\CYRABHCHDSC\cyrabhdze
810 \CYRABHDZE\cyrabhha\CYRABHHA\cyrac\CYRAE\cyrb\CYRB\cyrbyus
811 \CYRBYUS\cyrC\CYRC\cyrch\CYRCH\cyrchldsc\CYRCHLDSC\cyrchrds
812 \CYRCHRDS\cyrchvcrs\CYRCHVCRS\cyrd\CYRD\cyrdelta\CYRDELTA
813 \cyrdje\CYRDJE\cyrdze\CYRDZE\cyrdzhe\CYRDZHE\cyre\CYRE\cyreps
814 \CYREPS\cyrerev\CYREREV\cyrery\CYRERY\cyrf\CYRF\cyrfita
815 \CYRFITA\cyrG\CYRG\cyrGdsc\CYRGDSC\cyrGdschcrs\CYRGDSCHCRS
816 \cyrghcrs\CYRGHCRS\cyrghk\CYRGHK\cyrGup\CYRGUP\cyrh\CYRH
817 \cyrhdsc\CYRHDS\cyrhhcrs\CYRHHCRS\cyrhhk\CYRHHK\cyrhrdsn
818 \CYRHRDSN\cyri\CYRI\cyrie\CYRIE\cyrii\CYRII\cyrishrt\CYRISHRT
819 \cyrishrtdsc\CYRISHRTDSC\cyrizh\CYRIZH\cyrje\CYRJE\cyrk\CYRK
820 \cyrkbeak\CYRKBEAK\cyrkds\CYRKDSC\cyrkchcrs\CYRKCHCRS\cyrkhk
821 \CYRKHK\cyrkvcrs\CYRKVCRS\cyrl\CYRL\cyrlldsc\CYRLDSC\cyrlhk
822 \CYRLHK\cyrlje\CYRLJE\cyrM\CYRM\cyrmdsc\CYRMDSC\cyrmhk\CYRMMHK
823 \CYRNJE\cyrnlhk\CYRNLHK\cyro\CYRO\cyrotld\CYROTLD\cyrp\CYRP
824 \cyrphk\CYRPHK\cyrq\CYRQ\cyrr\CYRR\cyrrdsc\CYRRDSC\cyrrhk
825 \CYRRHK\cyrrtick\CYRRTICK\cyrs\CYRS\cyrsacrs\CYRSACRS
826 \cyrschwa\CYRSCHWA\cyrsdsc\CYRSDSC\cyrsemisftsn\CYRSEMISFTSN
827 \cyrsftsn\CYRSFTSN\cyrsch\CYRSH\cyrshch\CYRSHCH\cyrshha\CYRSHHA
828 \cyrt\CYRT\cyrtldsc\CYRTDSC\cyrtetse\CYRTETSE\cyrtshe\CYRTSHE
829 \cyru\CYRU\cyrushrt\CYRUSHRT\cyrv\CYRV\cyrw\CYRW\cyr\CYRY
830 \cyrya\CYRYA\cyryat\CYRYAT\cyryhcrs\CYRYHCRS\cyryi\CYRYI\cyryo
831 \CYRYO\cyryu\CYRYU\cyrz\CYRZ\cyrzds\CYRZDSC\cyrzsh\CYRZSH
832 \cyrzhdsc\CYRZHDSC}%
833 \let\update@uclc@with@cyrillic\relax
834 }
835 }
```

Here we process each option:

```
836 \DeclareOption*{%
837 \let\encodingdefault\CurrentOption
838 \edef\reserved@f{%
839 \lowercase{\def\noexpand\reserved@f{\CurrentOption enc.def}}}%
840 \reserved@f
841 \InputIfFileExists\reserved@f
842 {}{\PackageError{fontenc}%
843 {Encoding file '\reserved@f' not found.%
844 \MessageBreak
```

```

845         You might have misspelt the name of the encoding}%
846     {Necessary code for this encoding was not
847     loaded.\MessageBreak
848     Thus calling the encoding later on will
849     produce further error messages.}}%
850 \let\reserved@f\relax

```

In case the current encoding is one of a list of known cyrillic ones we extend the `\@uclclist`:

```

851 \expandafter\in@\expandafter{\CurrentOption}%
852         {T2A,T2B,T2C,X2,LCY,OT2}%
853 \ifin@

```

But only if it hasn't already been extended. This might happen if there are several calls to fontenc loading one of the above encodings. If we don't do this check the `\@uclclist` gets unnecessarily big, slowing down the processing at runtime.

```

854     \expandafter\in@\expandafter\cyr@\expandafter
855     {\@uclclist}%
856 \ifin@
857 \else
858     \update@uclc@with@cyrillic
859 \fi
860 \fi
861 }

```

```
862 \ProcessOptions*
```

```
863 \fontencoding\encodingdefault\selectfont
```

To save some space we get rid of the macro extending the `\@uclclist` (might have happened already).

```
864 \let\update@uclc@with@cyrillic\relax
```

Finally we pretend that the fontenc package wasn't read in. This allows for using it several times, e.g., in a class file and in the preamble (at the cost of not getting any version info). That kind of hackery shows that using a general purpose package just for loading an encoding is not the right kind of interface for setting up encodings — it will get replaced at some point in the future.

```

865 \global\expandafter\let\csname ver@fontenc.sty\endcsname\relax
866 \global\expandafter\let\csname opt@fontenc.sty\endcsname\relax
867 \global\let\@ifl@ter@\@ifl@ter
868 \def\@ifl@ter#1#2#3#4#5{\global\let\@ifl@ter\@ifl@ter@}
869 \endpackage

```

2.2 The textcomp package

This one is for the TS1 encoding which contains text symbols for use with the T1-encoded text fonts. It therefore first inputs the file `TS1enc.def` and then sets (or resets) the defaults for the symbols it contains. The result of this is that when one of these symbols is accessed and the current encoding does not provide it, the symbol will be supplied by a silent, local change to this encoding.

870 $\langle *TS1sty \rangle$

Since many PostScript fonts only implement a subset of TS1 many commands only produce black blobs of ink. To resolve the resulting problems a number of options have been introduced and some code has been developed to distinguish sub-encodings.

The sub-encodings have a numerical id and are defined as follows for TS1:

#5 those TS1 symbols that are also in the ISO-Adobe character set; without `textcurrency`, which is often misused for the Euro. Older Type1 fonts from the non- \TeX world provide only this subset.

#4 = #5 + `\texteuro`. Most newer fonts provide this.

#3 = #4 + `\textomega`. Can also be described as $TS1 \cap (ISO-Adobe \cup MacRoman)$. (Except for the missing "currency".)

#2 = #3 + `\textestimated` + `textcurrency`. Can also be described as $TS1 \cap Adobe-Western-2$. This may be relevant for OpenType fonts, which usually show the Adobe-Western-2 character set.

#1 = TS1 without `\textcircled` and `\t`. These two glyphs are often not implemented and if their kernel defaults are changed commands like `\copyright` unnecessarily fail.

#0 = full TS1

And here a summary to go in the transcript file:

```
871 \PackageInfo{textcomp}{Sub-encoding information:\MessageBreak
872   \space\space 5 = only ISO-Adobe without \string\textcurrency\MessageBreak
873   \space\space 4 = 5 + \string\texteuro\MessageBreak
874   \space\space 3 = 4 + \string\textohm\MessageBreak
875   \space\space 2 = 3 + \noexpand\textestimated+ \string\textcurrency\MessageBreak
876   \space\space 1 = TS1 - \noexpand\textcircled- \string\t\MessageBreak
877   \space\space 0 = TS1 (full)\MessageBreak
878   Font families with sub-encoding setting implement\MessageBreak
879   only a restricted character set as indicated.\MessageBreak
880   Family '?' is the default used for unknown fonts.\MessageBreak
881   See the documentation for details\@gobble}
```

`\DeclareEncodingSubset` An encoding subset to which a font family belongs is declared by `\DeclareEncodingSubset` that take the major encoding as the first argument (e.g., TS1), the family name as the second argument (e.g., cmr), and the subset encoding id as a third, (e.g., 0 for cmr).

The default encoding subset to use when nothing is known about the current font family is named ?.

```
882 \def\DeclareEncodingSubset#1#2#3{%
883   \@ifundefined{#1:#2}%
884     {\PackageInfo{textcomp}{Setting #2 sub-encoding to #1/#3}}%
885     {\PackageInfo{textcomp}{Changing #2 sub-encoding to #1/#3}}%
```

```
886 \namedef{#1:#2}{#3}}
887 \@onlypreamble\DeclareEncodingSubset
```

The options for the package are the following:

safe for unknown font families enables only symbols that are also in the ISO-Adobe character set; without "currency", which is often misused for the Euro. Older Type1 fonts from the non-TeX world provide only this subset.

euro enables the "safe" symbols plus the `\texteuro` command. Most newer fonts provide this.

full enables all TS1 commands; useful only with fonts like EC or CM bright.

almostfull same as "full", except that `\textcircled` and `\t` are *not* redefined from their defaults to avoid that commands like `\copyright` suddenly no longer work.

force ignore all subset encoding definitions stored in the package itself or in the configuration file and always use the default subset as specified by one of the other options (seldom useful, only dangerous).

`\iftc@forced` Switch used to implement the `force` option

```
888 \newif\iftc@forced \tc@forcedfalse
```

This is implemented by defining the default subset:

```
889 \DeclareOption{full}{\DeclareEncodingSubset{TS1}{?}{0}}
890 \DeclareOption{almostfull}{\DeclareEncodingSubset{TS1}{?}{1}}
891 \DeclareOption{euro}{\DeclareEncodingSubset{TS1}{?}{4}}
892 \DeclareOption{safe}{\DeclareEncodingSubset{TS1}{?}{5}}
```

The default is "almostfull" which means that old documents will work except that `\textcircled` and `\t` will use the kernel defaults (with the advantage that this also works if the current font (as often the case) doesn't implement these glyphs.

The "force" option simply sets the switch to true.

```
893 \DeclareOption{force}{\tc@forcedtrue}
```

The suggestions to user is to use the "safe" option always unless that balks in which case they could switch to "almostfull" but then better check their output manually.

```
894 \def\tc@errorwarn{\PackageError}
895 \DeclareOption{warn}{\gdef\tc@errorwarn#1#2#3{\PackageWarning{#1}{#2}}}
896 \ExecuteOptions{almostfull}
897 \ProcessOptions\relax
```

`\CheckEncodingSubset` The command `\CheckEncodingSubset` will check if the current font family has the right encoding subset to typeset a certain command. It takes five arguments as follows: first argument is either `\UseTextSymbol`, `\UseTextAccent` depending on whether or not the symbol is a text symbol or a text accent.

The second argument is the encoding from which this symbol should be fetched.
 The third argument is either a fake accessor command or an error message.
 the code in that argument (if ever executed) receives two arguments: #2 and #5
 of `\CheckEncodingSubset`.

Argument four is the subset encoding id to test against: if this value is higher
 than the subset id of the current font family then we typeset the symbol, i.e.,
 execute `#1{#2}#5` otherwise it runs `#3#5`, e.g., to produce an error message or
 fake the glyph somehow.

Argument five is the symbol or accent command that is being checked.

For usage examples see definitions below.

```
898 \iftc@forced
```

If the “force” option was given we always use the default for testing against.

```
899 \def\CheckEncodingSubset#1#2#3#4#5{%
900   \ifnum #4>%
901     0\csname #2:\endcsname
902     \relax
903   \expandafter\@firstoftwo
904   \else
905   \expandafter\@secondoftwo
906 \fi
907   {#1{#2}}{#3}%
908   #5%
909 }
```

In normal circumstances the test is a bit more complicated: first check if there
 exists a macro `\langle arg2\rangle:\langle current-family\rangle` and if so use that value to test against,
 otherwise use the default to test against.

```
910 \else
911 \def\CheckEncodingSubset#1#2#3#4#5{%
912   \ifnum #4>%
913     \expandafter\ifx\csname #2:\f@family\endcsname\relax
914     0\csname #2:\endcsname
915     \else
916     \csname #2:\f@family\endcsname
917     \fi
918     \relax
919   \expandafter\@firstoftwo
920   \else
921   \expandafter\@secondoftwo
922 \fi
923   {#1{#2}}{#3}%
924   #5%
925 }
926 \fi
```

```
tc@subst
```

```
927 \def\tc@subst#1{%
928   \tc@errorwarn{textcomp}% % should be latex error if general
```

```

929   {Symbol \string#1 not provided by\MessageBreak
930   font family \f@family\space
931   in TS1 encoding.\MessageBreak Default family used instead}\@eha
932   \bgroup\fontfamily\textcompsubstdefault\selectfont#1\egroup
933 }

```

`\textcompsubstdefault`

```

934 \def\textcompsubstdefault{cmr}

```

`\tc@error` `\tc@error` is going to be used in arg #3 of `\CheckEncodingSubset` when a symbol is not available in a certain font family. It gets pass the encoding it normally lives in (arg one) and the name of the symbol or accent that has a problem.

```

935 % error commands take argument:
936 % #1 symbol to be used
937 \def\tc@error#1{%
938   \PackageError{textcomp}% % should be latex error if general
939   {Accent \string#1 not provided by\MessageBreak
940   font family \f@family\space
941   in TS1 encoding}\@eha
942 }

```

`\tc@fake@euro` `\tc@fake@euro` is an example of a “fake” definition to use in arg #3 of `\CheckEncodingSubset` when a symbol is not available in a certain font family. Here we produce an Euro symbol by combining a “C” with a “=”.

```

943 \def\tc@fake@euro#1{%
944   \leavevmode
945   \PackageInfo{textcomp}{Faking \noexpand#1for font family
946                       \f@family\MessageBreak in TS1 encoding}%
947   \valign{##}\cr
948   \vfil\hbox to 0.07em{\dimen@\f@size\p@
949                       \math@fontsfalse
950                       \fontsize{.7\dimen@}\z@\selectfont=\hss}\vfil\cr%
951   \hbox{C}\crrc
952   }%
953 }

```

`\tc@check@symbol` `\tc@check@symbol` These are two abbreviations that we use below to check symbols and accents in TS1. Only there to save some space, e.g., we can then write

```

\DeclareTextCommandDefault{\textcurrency}{\tc@check@symbol3\textcurrency}

```

to ensure that `\textcurrency` is only typeset if the current font has a TS1 subset id of less than 3. Otherwise `\tc@error` is called telling the user that for this font family `\textcurrency` is not available.

```

954 \def\tc@check@symbol{\CheckEncodingSubset\UseTextSymbol{TS1}\tc@subst}
955 \def\tc@check@accent{\CheckEncodingSubset\UseTextAccent{TS1}\tc@error}

```

We start with the commands that are “safe” and which can be unconditionally set up, first the accents...

```

956 \DeclareTextAccentDefault{\capitalcedilla}{TS1}
957 \DeclareTextAccentDefault{\capitalogonek}{TS1}
958 \DeclareTextAccentDefault{\capitalgrave}{TS1}
959 \DeclareTextAccentDefault{\capitalacute}{TS1}
960 \DeclareTextAccentDefault{\capitalcircumflex}{TS1}
961 \DeclareTextAccentDefault{\capitaltilde}{TS1}
962 \DeclareTextAccentDefault{\capitaldieresis}{TS1}
963 \DeclareTextAccentDefault{\capitalhungarumlaut}{TS1}
964 \DeclareTextAccentDefault{\capitalring}{TS1}
965 \DeclareTextAccentDefault{\capitalcaron}{TS1}
966 \DeclareTextAccentDefault{\capitalbreve}{TS1}
967 \DeclareTextAccentDefault{\capitalmacron}{TS1}
968 \DeclareTextAccentDefault{\capitaldotaccent}{TS1}
... and then the other glyphs.
969 \DeclareTextSymbolDefault{\textcapitalcompwordmark}{TS1}
970 \DeclareTextSymbolDefault{\textascendercompwordmark}{TS1}
971 \DeclareTextSymbolDefault{\textquotestraightbase}{TS1}
972 \DeclareTextSymbolDefault{\textquotestraightdblbase}{TS1}
973 \DeclareTextSymbolDefault{\texttwelveudash}{TS1}
974 \DeclareTextSymbolDefault{\textthreequartersemdash}{TS1}
975 \DeclareTextSymbolDefault{\textdollar}{TS1}
976 \DeclareTextSymbolDefault{\textquotesingle}{TS1}
977 \DeclareTextSymbolDefault{\textasteriskcentered}{TS1}
978 \DeclareTextSymbolDefault{\textfractionsolidus}{TS1}
979 \DeclareTextSymbolDefault{\textminus}{TS1}
980 \DeclareTextSymbolDefault{\textlbrackdbl}{TS1}
981 \DeclareTextSymbolDefault{\textrbrackdbl}{TS1}
982 \DeclareTextSymbolDefault{\textasciigrave}{TS1}
983 \DeclareTextSymbolDefault{\texttildelow}{TS1}
984 \DeclareTextSymbolDefault{\textasciibreve}{TS1}
985 \DeclareTextSymbolDefault{\textasciicaron}{TS1}
986 \DeclareTextSymbolDefault{\textgravedbl}{TS1}
987 \DeclareTextSymbolDefault{\textacutedbl}{TS1}
988 \DeclareTextSymbolDefault{\textdagger}{TS1}
989 \DeclareTextSymbolDefault{\textdaggerdbl}{TS1}
990 \DeclareTextSymbolDefault{\textbardbl}{TS1}
991 \DeclareTextSymbolDefault{\textperthousand}{TS1}
992 \DeclareTextSymbolDefault{\textbullet}{TS1}
993 \DeclareTextSymbolDefault{\textcelsius}{TS1}
994 \DeclareTextSymbolDefault{\textflorin}{TS1}
995 \DeclareTextSymbolDefault{\texttrademark}{TS1}
996 \DeclareTextSymbolDefault{\textcent}{TS1}
997 \DeclareTextSymbolDefault{\textsterling}{TS1}
998 \DeclareTextSymbolDefault{\textyen}{TS1}
999 \DeclareTextSymbolDefault{\textbrokenbar}{TS1}
1000 \DeclareTextSymbolDefault{\textsection}{TS1}
1001 \DeclareTextSymbolDefault{\textasciidieresis}{TS1}
1002 \DeclareTextSymbolDefault{\textcopyright}{TS1}
1003 \DeclareTextSymbolDefault{\textordfeminine}{TS1}

```

```

1004 \DeclareTextSymbolDefault{\textlnot}{TS1}
1005 \DeclareTextSymbolDefault{\textregistered}{TS1}
1006 \DeclareTextSymbolDefault{\textasciimacron}{TS1}
1007 \DeclareTextSymbolDefault{\textdegree}{TS1}
1008 \DeclareTextSymbolDefault{\textpm}{TS1}
1009 \DeclareTextSymbolDefault{\texttwosuperior}{TS1}
1010 \DeclareTextSymbolDefault{\textthreesuperior}{TS1}
1011 \DeclareTextSymbolDefault{\textasciiacute}{TS1}
1012 \DeclareTextSymbolDefault{\textmu}{TS1}
1013 \DeclareTextSymbolDefault{\textparagraph}{TS1}
1014 \DeclareTextSymbolDefault{\textperiodcentered}{TS1}
1015 \DeclareTextSymbolDefault{\textonesuperior}{TS1}
1016 \DeclareTextSymbolDefault{\textordmasculine}{TS1}
1017 \DeclareTextSymbolDefault{\textonequarter}{TS1}
1018 \DeclareTextSymbolDefault{\textonehalf}{TS1}
1019 \DeclareTextSymbolDefault{\textthreequarters}{TS1}
1020 \DeclareTextSymbolDefault{\texttimes}{TS1}
1021 \DeclareTextSymbolDefault{\textdiv}{TS1}

```

The `\texteuro` is only available for subsets with id 4 or less. Otherwise we fake the glyph using `\tc@fake@euro`

```

1022 \DeclareTextCommandDefault{\texteuro}
1023   {\CheckEncodingSubset\UseTextSymbol{TS1}\tc@fake@euro5\texteuro}

```

The `\textohm` is only available for subsets with id 3 or less. Otherwise we produce an error.

```

1024 \DeclareTextCommandDefault{\textohm}{\tc@check@symbol4\textohm}

```

The `\textestimated` and `\textcurrency` are only provided for fonts with subset encoding with id 2 or less.

```

1025 \DeclareTextCommandDefault{\textestimated}{\tc@check@symbol3\textestimated}
1026 \DeclareTextCommandDefault{\textcurrency}{\tc@check@symbol3\textcurrency}

```

Nearly all of the remaining glyphs are provided only with fonts with id 1 or 0, i.e., are essentially complete.

```

1027 \DeclareTextCommandDefault{\capitaltie}{\tc@check@accent2\capitaltie}
1028 \DeclareTextCommandDefault{\newtie}{\tc@check@accent2\newtie}
1029 \DeclareTextCommandDefault{\capitalnewtie}{\tc@check@accent2\capitalnewtie}
1030 \DeclareTextCommandDefault{\textleftarrow}{\tc@check@symbol2\textleftarrow}
1031 \DeclareTextCommandDefault{\textrightarrow}{\tc@check@symbol2\textrightarrow}
1032 \DeclareTextCommandDefault{\textblank}{\tc@check@symbol2\textblank}
1033 \DeclareTextCommandDefault{\textdblhyphen}{\tc@check@symbol2\textdblhyphen}
1034 \DeclareTextCommandDefault{\textzerooldstyle}{\tc@check@symbol2\textzerooldstyle}
1035 \DeclareTextCommandDefault{\textoneoldstyle}{\tc@check@symbol2\textoneoldstyle}
1036 \DeclareTextCommandDefault{\texttwooldstyle}{\tc@check@symbol2\texttwooldstyle}
1037 \DeclareTextCommandDefault{\textthreeoldstyle}{\tc@check@symbol2\textthreeoldstyle}
1038 \DeclareTextCommandDefault{\textfouroldstyle}{\tc@check@symbol2\textfouroldstyle}
1039 \DeclareTextCommandDefault{\textfiveoldstyle}{\tc@check@symbol2\textfiveoldstyle}
1040 \DeclareTextCommandDefault{\textsixoldstyle}{\tc@check@symbol2\textsixoldstyle}
1041 \DeclareTextCommandDefault{\textsevenoldstyle}{\tc@check@symbol2\textsevenoldstyle}
1042 \DeclareTextCommandDefault{\texteightoldstyle}{\tc@check@symbol2\texteightoldstyle}

```

```

1043 \DeclareTextCommandDefault{\textnineoldstyle}{\tc@check@symbol2\textnineoldstyle}
1044 \DeclareTextCommandDefault{\textlangle}{\tc@check@symbol2\textlangle}
1045 \DeclareTextCommandDefault{\textrangle}{\tc@check@symbol2\textrangle}
1046 \DeclareTextCommandDefault{\textmho}{\tc@check@symbol2\textmho}
1047 \DeclareTextCommandDefault{\textbigcircle}{\tc@check@symbol2\textbigcircle}
1048 \DeclareTextCommandDefault{\textuparrow}{\tc@check@symbol2\textuparrow}
1049 \DeclareTextCommandDefault{\textdownarrow}{\tc@check@symbol2\textdownarrow}
1050 \DeclareTextCommandDefault{\textborn}{\tc@check@symbol2\textborn}
1051 \DeclareTextCommandDefault{\textdivorced}{\tc@check@symbol2\textdivorced}
1052 \DeclareTextCommandDefault{\textdied}{\tc@check@symbol2\textdied}
1053 \DeclareTextCommandDefault{\textleaf}{\tc@check@symbol2\textleaf}
1054 \DeclareTextCommandDefault{\textmarried}{\tc@check@symbol2\textmarried}
1055 \DeclareTextCommandDefault{\textmusicalnote}{\tc@check@symbol2\textmusicalnote}
1056 \DeclareTextCommandDefault{\textdblhyphenchar}{\tc@check@symbol2\textdblhyphenchar}
1057 \DeclareTextCommandDefault{\textdollaroldstyle}{\tc@check@symbol2\textdollaroldstyle}
1058 \DeclareTextCommandDefault{\textcentoldstyle}{\tc@check@symbol2\textcentoldstyle}
1059 \DeclareTextCommandDefault{\textcolonmonetary}{\tc@check@symbol2\textcolonmonetary}
1060 \DeclareTextCommandDefault{\textwon}{\tc@check@symbol2\textwon}
1061 \DeclareTextCommandDefault{\textnaira}{\tc@check@symbol2\textnaira}
1062 \DeclareTextCommandDefault{\textguarani}{\tc@check@symbol2\textguarani}
1063 \DeclareTextCommandDefault{\textpeso}{\tc@check@symbol2\textpeso}
1064 \DeclareTextCommandDefault{\textlira}{\tc@check@symbol2\textlira}
1065 \DeclareTextCommandDefault{\textrecipe}{\tc@check@symbol2\textrecipe}
1066 \DeclareTextCommandDefault{\textinterrobang}{\tc@check@symbol2\textinterrobang}
1067 \DeclareTextCommandDefault{\textinterrobangdown}{\tc@check@symbol2\textinterrobangdown}
1068 \DeclareTextCommandDefault{\textdong}{\tc@check@symbol2\textdong}
1069 \DeclareTextCommandDefault{\textpertenthousand}{\tc@check@symbol2\textpertenthousand}
1070 \DeclareTextCommandDefault{\textpilcrow}{\tc@check@symbol2\textpilcrow}
1071 \DeclareTextCommandDefault{\textbaht}{\tc@check@symbol2\textbaht}
1072 \DeclareTextCommandDefault{\textnumero}{\tc@check@symbol2\textnumero}
1073 \DeclareTextCommandDefault{\textdiscount}{\tc@check@symbol2\textdiscount}
1074 \DeclareTextCommandDefault{\textopenbullet}{\tc@check@symbol2\textopenbullet}
1075 \DeclareTextCommandDefault{\textservicemark}{\tc@check@symbol2\textservicemark}
1076 \DeclareTextCommandDefault{\textlquill}{\tc@check@symbol2\textlquill}
1077 \DeclareTextCommandDefault{\textrquill}{\tc@check@symbol2\texttrquill}
1078 \DeclareTextCommandDefault{\textcopyleft}{\tc@check@symbol2\textcopyleft}
1079 \DeclareTextCommandDefault{\textcircledP}{\tc@check@symbol2\textcircledP}
1080 \DeclareTextCommandDefault{\textreferencemark}{\tc@check@symbol2\textreferencemark}
1081 \DeclareTextCommandDefault{\textsurd}{\tc@check@symbol2\textsurd}

```

The `\textcircled` and `\t` are handled specially, unless the current font has a subset id of 0 (i.e. full TS1) we pick the symbols up from the the math font encodings, i.e., the third argument to `\CheckEncodingSubset` uses `\UseTextAccent` to get them from there.

```

1082 \DeclareTextCommandDefault{\textcircled}
1083   {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OMS}}1\textcircled}
1084 \DeclareTextCommandDefault{\t}
1085   {\CheckEncodingSubset\UseTextAccent{TS1}{\UseTextAccent{OML}}1\t}

```

Finally input the encoding-specific definitions for TS1 thus making the top-level definitions optimised for this encoding (and not for the default encoding, see

section 1.2).

```
1086 \input{ts1enc.def}
```

Now having the new glyphs available we also want to make sure that they are used. For most cases this will automatically happen but for some glyphs there are inferior definitions already known to L^AT_EX which will prevent the usage of the TS1 versions (see section 1.1 above). So we better get rid of them:

```
1087 \UndeclareTextCommand{\textsterling}{OT1}
```

```
1088 \UndeclareTextCommand{\textdollar} {OT1}
```

Similar declarations should probably be made for other encodings like OT4 if they are in use.

```
1089 %\UndeclareTextCommand{\textsterling}{OT4}
```

```
1090 %\UndeclareTextCommand{\textdollar} {OT4}
```

From the T1 encoding there are two candidates for removal: %₀₀ and %₀₀₀ since these are both constructed from % followed by a tiny ‘o’ rather than being a single glyph. The problem with this approach is that in PostScript fonts this small zero is usually not available resulting in %₀ rather than %₀₀ while the real glyph (at least for `\textperthousand`) is available in the PostScript version of TS1. So for the moment we compromise by removing the T1 declaration for `\textperthousand` but keeping the one for `\textpertenthousand`. This will have the effect that with Computer Modern fonts everything will come out (although %₀₀ and %₀₀₀ are not taken from the same physical font) and with PostScript fonts %₀₀ will come out correctly while %₀₀₀ will most likely look like %₀ — which is probably an improvement over just getting a single ‘■’ to indicate a completely missing glyph, which would happen if we also ‘undeclared’ `\textpertenthousand`.

```
1091 \UndeclareTextCommand{\textperthousand}{T1}
```

```
1092 %\UndeclareTextCommand{\textpertenthousand}{T1}
```

2.2.1 Supporting oldstyle digits

```
1093 \DeclareRobustCommand\oldstylenums[1]{%
```

```
1094 \begingroup
```

```
1095 \ifmmode
```

```
1096 \mathgroup\symletters #1%
```

```
1097 \else
```

```
1098 \CheckEncodingSubset\@use@text@encoding{TS1}%
```

```
1099 {\PackageWarning{textcomp}%
```

```
1100 {Oldstyle digits unavailable for
```

```
1101 family \f@family.\MessageBreak
```

```
1102 Lining digits used instead}}%
```

```
1103 \tw@{#1}%
```

```
1104 \fi
```

```
1105 \endgroup
```

```
1106 }
```

2.2.2 Subset encoding defaults

For many font families commonly used in the T_EX world we provide the subset encoding data here. Users can add additional font families in the file `textcomp.cfg`

if they own other fonts.

However, if the option “forced” was given then all subset encoding specifications are ignored, so there is no point in setting any of them up:

```
1107 \iftc@forced \else
```

Computer modern based fonts (e.g., CM, CM-Bright, Concrete):

```
1108 \DeclareEncodingSubset{TS1}{cmr}      {0}
1109 \DeclareEncodingSubset{TS1}{cmss}     {0}
1110 \DeclareEncodingSubset{TS1}{cmtt}     {0}
1111 \DeclareEncodingSubset{TS1}{cmvtt}    {0}
1112 \DeclareEncodingSubset{TS1}{cmbr}     {0}
1113 \DeclareEncodingSubset{TS1}{cmtl}     {0}
1114 \DeclareEncodingSubset{TS1}{ccr}      {0}
```

PSNFSS fonts:

```
1115 \DeclareEncodingSubset{TS1}{ptm}     {4}
1116 \DeclareEncodingSubset{TS1}{pcr}     {4}
1117 \DeclareEncodingSubset{TS1}{phv}     {4}
1118 \DeclareEncodingSubset{TS1}{ppl}     {3}
1119 \DeclareEncodingSubset{TS1}{pag}     {4}
1120 \DeclareEncodingSubset{TS1}{pbk}     {4}
1121 \DeclareEncodingSubset{TS1}{pnc}     {4}
1122 \DeclareEncodingSubset{TS1}{pzc}     {4}
1123 \DeclareEncodingSubset{TS1}{bch}     {4}
1124 \DeclareEncodingSubset{TS1}{put}     {5}
```

Other CTAN fonts (probably not complete):

```
1125 \DeclareEncodingSubset{TS1}{uag}     {5}
1126 \DeclareEncodingSubset{TS1}{ugq}     {5}
1127 \DeclareEncodingSubset{TS1}{ul8}     {4}
1128 \DeclareEncodingSubset{TS1}{ul9}     {4} % (LuxiSans, one day)
1129 \DeclareEncodingSubset{TS1}{augie}   {5}
1130 \DeclareEncodingSubset{TS1}{dayrom}   {3}
1131 \DeclareEncodingSubset{TS1}{dayroms} {3}
1132 \DeclareEncodingSubset{TS1}{pxr}     {0}
1133 \DeclareEncodingSubset{TS1}{pxss}    {0}
1134 \DeclareEncodingSubset{TS1}{pxtt}    {0}
1135 \DeclareEncodingSubset{TS1}{txr}     {0}
1136 \DeclareEncodingSubset{TS1}{txss}    {0}
1137 \DeclareEncodingSubset{TS1}{txtt}    {0}
```

Fourier-GUTenberg:

```
1138 \DeclareEncodingSubset{TS1}{futs}    {4}
1139 \DeclareEncodingSubset{TS1}{futx}    {4}
1140 \DeclareEncodingSubset{TS1}{futj}    {4}
```

Y&Y’s Lucida Bright

```
1141 \DeclareEncodingSubset{TS1}{hlh}     {3}
1142 \DeclareEncodingSubset{TS1}{hls}     {3}
1143 \DeclareEncodingSubset{TS1}{hlst}    {3}
```

The remaining settings for Lucida are conservative: the following fonts contain the `\textohm` character but not the `\texteuro`, i.e., belong to neither subset 4 nor subset 3. If you want to use the `\textohm` with these fonts copy these definition to `textcomp.cfg` and change the subset to 3. However in that case make sure that you do not use the `\texteuro`.

```
1144 \DeclareEncodingSubset{TS1}{hlct} {5}
1145 \DeclareEncodingSubset{TS1}{hlx} {5}
1146 \DeclareEncodingSubset{TS1}{hlce} {5}
1147 \DeclareEncodingSubset{TS1}{hlcn} {5}
1148 \DeclareEncodingSubset{TS1}{hlcw} {5}
1149 \DeclareEncodingSubset{TS1}{hlcfl} {5}
```

Other commercial families...

```
1150 \DeclareEncodingSubset{TS1}{pplx} {3}
1151 \DeclareEncodingSubset{TS1}{pplj} {3}
1152 \DeclareEncodingSubset{TS1}{ptmx} {4}
1153 \DeclareEncodingSubset{TS1}{ptmj} {4}
```

If the file `textcomp.cfg` exists it will be loaded at this point. This allows to define further subset encodings for font families not covered by default.

```
1154 \InputIfFileExists{textcomp.cfg}
1155 {\PackageInfo{textcomp}{Local configuration file used}}{}

1156 \fi
1157 </TS1sty>
```