Providing some UTF-8 support via `inputenc`

David Carlisle Frank Mittelbach Chris Rowley*

v1.3c 2026-06-01 printed October 30, 2026

This file is maintained by the LaTeX Project team.
Bug reports can be opened (category `latex`) at
https://latex-project.org/bugs.html.

Contents

1 Introduction **2**
1.1 Background and general stuff . . . 2
1.2 More specific stuff . . . 2
1.3 Notes . . . 3
1.4 Basic operation of the code . . . 3

2 Coding **4**
2.1 Housekeeping . . . 4
2.2 Parsing UTF-8 input . . . 4
2.3 Mapping Unicode codes to LaTeX internal forms . . . 9
2.4 Loading Unicode mappings at begin document . . . 14

3 Mapping characters — based on font (glyph) encodings **15**
3.1 About the table itself . . . 15
3.2 The mapping table . . . 16
3.3 Notes . . . 28
3.4 Mappings for OT1 glyphs . . . 28
3.5 Mappings for OMS glyphs . . . 29
3.6 Mappings for TS1 glyphs . . . 29
3.7 Mappings for `latex.ltx` glyphs . . . 29
3.8 Old `utf8.def` file as a temp fix for pTeX and friends . . . 30

4 A test document **35**

*Borrowing heavily from tables by Sebastian Rahtz; some table and code cleanup by Javier Bezos

1 Introduction

1.1 Background and general stuff

For many reasons what this package provides is a long way from any type of 'Unicode compliance'.

In stark contrast to 8-bit character sets, with 16 or more bits it can easily be very inefficient to support the full range.[1] Moreover, useful support of character input by a typesetting system overwhelmingly means finding an acceptable visual representation of a sequence of characters and this, for LaTeX, means having available a suitably encoded 8-bit font.

Unfortunately it is not possible to predict exactly what valid UTF-8 octet sequences will appear in a particular file so it is best to make all the unsupported but valid sequences produce a reasonably clear and noticeable error message.

There are two directions from which to approach the question of what to load. One is to specify the ranges of Unicode characters that will result in some sensible typesetting; this requires the provider to ensure that suitable fonts are loaded and that these input characters generate the correct typesetting via the encodings of those fonts. The other is to inspect the font encodings to be used and use these to define which input Unicode characters should be supported.

For Western European languages, at least, going in either direction leads to many straightforward decisions and a few that are more subjective. In both cases some of the specifications are TeX specific whilst most are independent of the particular typesetting software in use.

As we have argued elsewhere, LaTeX needs to refer to characters via 'seven-bit-text' names and, so far, these have been chosen by reference to historical sources such as Plain TeX or Adobe encoding descriptions. It is unclear whether this ad hoc naming structure should simply be extended or whether it would be useful to supplement it with standardized internal Unicode character names such as one or more of the following:[2]

```
\ltxutwochar <4 hex digits>

\ltxuchar {<hex digits>}
  B H U R R R

\ltxueightchartwo   <2 utf8 octets as 8-bit char tokens>
\ltxueightcharthree <3 utf8 octets ...>
\ltxueightcharfour  <4 utf8 octets ...>
```

1.2 More specific stuff

In addition to setting up the mechanism for reading UTF-8 characters and specifying the LaTeX-level support available, this package contains support for some default historically expected TeX-related characters and some example 'Unicode definition files' for standard font encodings.

[1] In fact, LaTeX's current 8-bit support does not go so far as to make all 8-bit characters into valid input.

[2] Burkhard und Holger Mittelbach spielen mit mir! Sie haben etwas hier geschrieben.

1.3 Notes

This package does not support Unicode combining characters as TEX is not really equipped to make this possible.

No attempt is made to be useful beyond Latin, and maybe Cyrillic, for European languages (as of now).

1.4 Basic operation of the code

The `inputenc` package makes the upper 8-bit characters active and assigns to all of them an error message. It then waits for the input encoding files to change this set-up. Similarly, whenever `\inputencoding` is encountered in a document, first the upper 8-bit characters are set back to produce an error and then the definitions for the new input encoding are loaded, changing some of the previous settings.

The 8-bit input encodings currently supported by `inputenc` all use declarations such as `\DeclareInputText` and the like to map an 8-bit number to some LATEX internal form, e.g. to `\"a`.

The situation when supporting UTF-8 as the input encoding is different, however. Here we only have to set up the actions of those 8-bit numbers that can be the first octet in a UTF-8 representation of a Unicode character. But we cannot simply set this to some internal LATEX form since the Unicode character consists of more than one octet; instead we have to define this starting octet to parse the right number of further octets that together form the UTF-8 representation of some Unicode character.

Therefore when switching to `utf8` within the `inputenc` framework the characters with numbers (hex) from `"C2` to `"DF` are defined to parse for a second octet following, the characters from `"E0` to `"EF` are defined to parse for two more octets and finally the characters from `"F0` to `"F4` are defined to parse for three additional octets. These additional octets are always in the range `"80` to `"BF`.

Thus, when such a character is encountered in the document (so long as expansion is not prohibited) a defined number of additional octets (8-bit characters) are read and from them a unique control sequence name is immediately constructed.

This control sequence is either defined (good) or undefined (likely); in the latter case the user gets an error message saying that this UTF-8 sequence (or, better, Unicode character) is not supported.

If the control sequence is set up to do something useful then it will expand to a LATEX internal form: e.g. for the utf8 sequence of two octets `"C3 "A4` we get `\"a` as the internal form which then, depending on the font encoding, eventually resolves to the single glyph 'latin-a-umlaut' or to the composite glyph 'latin-a with an umlaut accent'.

These mappings from (UTF-8 encoded) Unicode characters to LATEX internal forms are made indirectly. The code below provides a declaration `\DeclareUnicodeCharacter` which maps Unicode numbers (as hexadecimal) to LATEX internal forms.

This mapping needs to be set up only once so it is done at `\begin{document}` by looking at the list of font encodings that are loaded by the document and providing mappings related to those font encodings whenever these are available. Thus at most only those Unicode characters that can be represented by the glyphs available in these encodings will be defined.

Technically this is done by loading one file per encoding, if available, that is supposed to provide the necessary mapping information.

2 Coding

2.1 Housekeeping

The usual introductory bits and pieces:

```
⟨utf8⟩\ProvidesFile{utf8.def}
⟨test⟩\ProvidesFile{utf8-test.tex}
⟨+lcy⟩ \ProvidesFile{lcyenc.dfu}
⟨+ly1⟩ \ProvidesFile{ly1enc.dfu}
⟨+oms⟩ \ProvidesFile{omsenc.dfu}
⟨+ot1⟩ \ProvidesFile{ot1enc.dfu}
⟨+ot2⟩ \ProvidesFile{ot2enc.dfu}
⟨+t1⟩  \ProvidesFile{t1enc.dfu}
⟨+t2a⟩ \ProvidesFile{t2aenc.dfu}
⟨+t2b⟩ \ProvidesFile{t2benc.dfu}
⟨+t2c⟩ \ProvidesFile{t2cenc.dfu}
⟨+ts1⟩ \ProvidesFile{ts1enc.dfu}
⟨+x2⟩  \ProvidesFile{x2enc.dfu}
⟨+all⟩ \ProvidesFile{utf8enc.dfu}
⟨-utf8-2018⟩   [2026-06-01 v1.3c UTF-8 support]
⟨*utf8⟩
```

This is a temporary fix for the e-pTEX / e-upTEX engines that do not yet have a `\ifincsname` primitive. Once this is available the extra file will be dropped.

```
\ifx\ifincsname\@undefined  % old e-pTeX or e-upTeX engines
  \input utf8-2018.def
  \expandafter\@firstofone
\else
  \expandafter\@gobble
\fi
  \endinput
\makeatletter
```

We restore the `\catcode` of space (which is set to ignore in `inputenc`) while reading `.def` files. Otherwise we would need to explicitly use `\space` all over the place in error and log messages.

```
\catcode`\ \saved@space@catcode
```

2.2 Parsing UTF-8 input

A UTF-8 char (that is not actually a 7-bit char, i.e. a single octet) is parsed as follows: each starting octet is an active TEX character token; each of these is defined below to be a macro with one to three arguments nominally (depending on the starting octet). It calls one of `\UTFviii@two@octets`, `\UTFviii@three@octets`, or `\UTFviii@four@octets` which then actually picks up the remaining octets as the argument(s).

- When typesetting we pick up the necessary number of additional octets, check if they form a command that LaTeX knows about (via `\csname u8:\string #1\string #2...\endcsname`) and if so use that for typesetting. `\string` is needed as the octets may (all?) be active and we want the literal values in the name.
- If the UTF-8 character is going to be part of a label, then it is essentially becoming part of some csname and with the test `\ifincsname` we can find this out. If so, we render the whole sequence off octets harmless by using `\string` too when the starting octet executes (`\UTF@...@octets@string`).
- Another possible case is that `\protect` has *not* the meaning of `\typeset@protect`. In that case we may do a `\write` or we may do a `\protected@edef` or ... In all such cases we want to keep the sequence of octets unchanged, but we can't use `\string` this time, since at least in the case of `\protect@edef` the result may later be typeset after all (in fact that is quite likely) and so at that point the starting octet needs to be an active character again (the others could be stringified). So for this case we use `\noexpand` ((`\UTF@...s@octets@noexpand`).

`\UTFviii@two@octets` Putting that all together the code for a start octet of a two byte sequence would then look like this:

```
\long\def\UTFviii@two@octets{%
  \ifincsname
    \expandafter \UTF@two@octets@string
  \else
    \ifx \protect\@typeset@protect \else
      \expandafter\expandafter\expandafter \UTF@two@octets@noexpand
    \fi
  \fi
  \UTFviii@two@octets@combine
}
```

`\ifcsname` is tested first because that can be true even if we are otherwise doing typesetting. If this is the case we use `\string` on the whole octet sequence. `\UTF@two@octets@string` not only does this but also gets rid of the command `\UTFviii@two@octets@combine` in the input stream by picking it up as a first argument and dropping it.

If this is not the case and we are doing typesetting (i.e., `\protect` is `\typeset@protect`), then we execute `\UTFviii@two@octets@combine` which picks up all octets and typesets the character (or generates an error if it doesn't know how to typeset it).

However, if we are not doing typesetting, then we execute the command `\UTFviii@two@octets@noexpand` which works like `\UTF@two@octets@string` but uses `\noexpand` instead of `\string`. This way the sequence is temporary rendered harmless, e.g., would display as is or stays put inside a `\protected@edef`. But if the result is later reused the starting octet is still active and so will be able to construct the UTF-8 character again.

`\UTFviii@three@octets` `\UTFviii@four@octets` The definitions for the other starting octets are the same except that they pick up more octets after them.

```
\long\def\UTFviii@three@octets{%
```

```
  \ifincsname
    \expandafter \UTF@three@octets@string
  \else
    \ifx \protect\@typeset@protect \else
      \expandafter\expandafter\expandafter \UTF@three@octets@noexpand
    \fi
  \fi
  \UTFviii@three@octets@combine
}
\long\def\UTFviii@four@octets{%
  \ifincsname
    \expandafter \UTF@four@octets@string
  \else
    \ifx \protect\@typeset@protect \else
      \expandafter\expandafter\expandafter \UTF@four@octets@noexpand
    \fi
  \fi
  \UTFviii@four@octets@combine
}
```

`\UTFviii@two@octets@noexpand` `\UTFviii@three@octets@noexpand` `\UTFviii@four@octets@noexpand` These temporarily prevent the active chars from expanding.

```
\long\def\UTF@two@octets@noexpand#1#2#3{\unexpanded{#2#3}}
\long\def\UTF@three@octets@noexpand#1#2#3#4{\unexpanded{#2#3#4}}
\long\def\UTF@four@octets@noexpand#1#2#3#4#5{\unexpanded{#2#3#4#5}}
```

`\UTFviii@two@octets@string` `\UTFviii@three@octets@string` `\UTFviii@four@octets@string` And the same with `\string` for use in `\csname` constructions.

```
\long\def\UTF@two@octets@string#1#2#3{\detokenize{#2#3}}
\long\def\UTF@three@octets@string#1#2#3#4{\detokenize{#2#3#4}}
\long\def\UTF@four@octets@string#1#2#3#4#5{\detokenize{#2#3#4#5}}
```

`\UTFviii@two@octets@combine` `\UTFviii@three@octets@combine` `\UTFviii@four@octets@combine` From the arguments a control sequence with a name of the form `u8:#1#2...` is constructed where the `#i` ($i > 1$) are the arguments and `#1` is the starting octet (as a TEX active character token). Since some or even all of these characters are active we need to use `\string` when building the `\csname`.

The `\csname` thus constructed can of course be undefined but to avoid producing an unhelpful low-level undefined command error we pass it to `\UTFviii@defined` which is responsible for producing a more sensible error message (not yet done!!). If, however, it is defined we simply execute the thing (which should then expand to an encoding specific internal LATEX form).

```
\long\def\UTFviii@two@octets@combine#1#2{\expandafter
    \UTFviii@defined\csname u8:\string#1\string#2\endcsname}
\long\def\UTFviii@three@octets@combine#1#2#3{\expandafter
    \UTFviii@defined\csname u8:\string#1\string#2\string#3\endcsname}
\long\def\UTFviii@four@octets@combine#1#2#3#4{\expandafter
    \UTFviii@defined\csname u8:\string#1\string#2\string#3\string#4\endcsname}
```

`\UTFviii@defined` This tests whether its argument is different from `\relax`: it either calls for a sensible error message (not done), or it gets the `\fi` out of the way (in case the command has arguments) and executes it.

```
\def\UTFviii@defined#1{%
  \ifx#1\relax
```

Test if the sequence is invalid UTF-8 or valid UTF-8 but without a LaTeX definition.

```
    \if\relax\expandafter\UTFviii@checkseq\string#1\relax\relax
```

The endline character has a special definition within the inputenc package (it is gobbling spaces). For this reason we can't produce multiline strings without some precaution.

```
     \UTFviii@undefined@err{#1}%
    \else
      \@latex@error{Invalid UTF-8 byte sequence (\expandafter
                     \@gobblefour\string#1)}%
                    \UTFviii@invalid@help
    \fi
  \else\expandafter
    #1%
  \fi
}
```

\UTFviii@invalid@err
\UTFviii@invalid@help

```
\def\UTFviii@invalid@err#1{%
 \@latex@error{Invalid UTF-8 byte "\UTFviii@hexnumber{`#1}}%
              \UTFviii@invalid@help}
\def\UTFviii@invalid@help{%
   The document does not appear to be in UTF-8 encoding.\MessageBreak
   Try adding \noexpand\UseRawInputEncoding as the first line of the file\MessageBreak
   or specify an encoding such as \noexpand\usepackage[latin1]{inputenc}\MessageBreak
   in the document preamble.\MessageBreak
   Alternatively, save the file in UTF-8 using your editor or another tool}
```

\UTFviii@undefined@err

```
\def\UTFviii@undefined@err#1{%
  \@latex@error{Unicode character \expandafter
                          \UTFviii@splitcsname\string#1\relax
                          \MessageBreak
                          not set up for use with LaTeX}%
                         {You may provide a definition with\MessageBreak
                          \noexpand\DeclareUnicodeCharacter}%
   }
```

\UTFviii@checkseq
\UTFviii@check@continue

Check that the csname consists of a valid UTF-8 sequence.

```
\def\UTFviii@checkseq#1:#2#3{%
 \ifnum`#2<"80 %
   \ifx\relax#3\else1\fi
 \else
   \ifnum`#2<"C0 %
     1 %
   \else
     \expandafter\expandafter\expandafter\UTFviii@check@continue
     \expandafter\expandafter\expandafter#3%
   \fi
 \fi}
```

```
\def\UTFviii@check@continue#1{%
  \ifx\relax#1%
  \else
  \ifnum`#1<"80 1\else\ifnum`#1>"BF 1\fi\fi
  \expandafter\UTFviii@check@continue
  \fi
}
```

`\UTFviii@loop` This bit of code derived from `xmltex` defines the active character corresponding to starting octets to call `\UTFviii@two@octets` etc as appropriate. The starting octet itself is passed directly as the first argument, the others are picked up later en route.

The `\UTFviii@loop` loops through the numbers starting at `\count@` and ending at `\@tempcnta` − 1, each time executing the code in `\UTFviii@tmp`.

Store current settings so can restore after the loops without using a group (gh/762).

```
\edef\reserved@a{%
\catcode`\noexpand\~=\the\catcode`\~\relax
\catcode`\noexpand\"=\the\catcode`\"\relax
\uccode`\noexpand\~=\the\uccode`\~\relax
\count@=\the\count@\relax
\@tempcnta=\the\@tempcnta\relax
\let\noexpand\reserved@a\relax}
```

```
\catcode`\~13
\catcode`\"12
```

```
\def\UTFviii@loop{%
  \uccode`\~\count@
  \uppercase\expandafter{\UTFviii@tmp}%
  \advance\count@\@ne
  \ifnum\count@<\@tempcnta
  \expandafter\UTFviii@loop
  \fi}
```

Handle the single byte control characters. C0 controls are valid UTF-8 but defined to give the "Character not defined error" They may be defined with `\DeclareUnicodeCharacter`.

```
    \def\UTFviii@tmp{\protected\edef~{\noexpand\UTFviii@undefined@err{:\string~}}}
% 0 ^^@ null
    \count@"1
    \@tempcnta9
% 9 ^^I tab
% 10 ^^J nl
\UTFviii@loop
    \count@11
    \@tempcnta12
\UTFviii@loop
% 12 ^^L
% 13 ^^M
    \count@14
    \@tempcnta32
\UTFviii@loop
```

Bytes with leading bits `10` are not valid UTF-8 starting bytes

```
    \count@"80
    \@tempcnta"C2
    \def\UTFviii@tmp{\protected\edef~{\noexpand\UTFviii@invalid@err\string~}}
\UTFviii@loop
```

Setting up 2-byte UTF-8: The starting bytes is passed as an active character so that it can be reprocessed later!

```
    \count@"C2
    \@tempcnta"E0
    \def\UTFviii@tmp{\protected\edef~{\noexpand\UTFviii@two@octets\noexpand~}}
\UTFviii@loop
```

Setting up 3-byte UTF-8:

```
    \count@"E0
    \@tempcnta"F0
    \def\UTFviii@tmp{\protected\edef~{\noexpand\UTFviii@three@octets\noexpand~}}
\UTFviii@loop
```

Setting up 4-byte UTF-8:

```
    \count@"F0
    \@tempcnta"F5
    \def\UTFviii@tmp{\protected\edef~{\noexpand\UTFviii@four@octets\noexpand~}}
\UTFviii@loop
```

Bytes above F4 are not valid UTF-8 starting bytes as they would encode numbers beyond the Unicode range

```
    \count@"F5
    \@tempcnta"100
    \def\UTFviii@tmp{\protected\edef~{\noexpand\UTFviii@invalid@err\string~}}
\UTFviii@loop
```

Restore values after the loops.

```
\reserved@a
```

For this case we must disable the warning generated by `inputenc` if it doesn't see any new `\DeclareInputText` commands.

```
\@inpenc@test
```

If this file (`utf8.def`) is not being read while setting up `inputenc`, i.e. in the preamble, but when `\inputencoding` is called somewhere within the document, we do not need to input the specific Unicode mappings again. We therefore stop reading the file at this point.

```
\ifx\@begindocumenthook\@undefined
  \makeatother
```

The `\fi` must be on the same line as `\endinput` or else it will never be seen!

```
  \endinput \fi
```

2.3 Mapping Unicode codes to LaTeX internal forms

`\DeclareUnicodeCharacter` The `\DeclareUnicodeCharacter` declaration defines a mapping from a Unicode character code point to a LaTeX internal form. The first argument is the Unicode number as hexadecimal digits and the second is the actual LaTeX internal form.

We start by making sure that some characters have the right `\catcode` when they are used in the definitions below.

```
\begingroup
\catcode`\"=12
\catcode`\<=12
\catcode`\.=12
\catcode`\,=12
\catcode`\;=12
\catcode`\!=12
\catcode`\~=13
\gdef\DeclareUnicodeCharacter#1#2{%
  \count@"#1\relax
  \wlog{ \space\space defining Unicode char U+#1 (decimal \the\count@)}%
  \begingroup
```

Next we do the parsing of the number stored in `\count@` and assign the result to `\UTFviii@tmp`. Actually all this could be done in-line, the macro `\parse@XML@charref` is only there to extend this code to parsing Unicode numbers in other contexts one day (perhaps).

```
    \parse@XML@charref
```

Here is an example of what is happening, for the pair `"C2 "A3` (which is the utf8 representation for the character £). After `\parse@XML@charref` we have, stored in `\UTFviii@tmp`, a single command with two character tokens as arguments:

> [t_{C2} and t_{A3} are the characters corresponding to these two octets]
> `\UTFviii@two@octets` $t_{\mathrm{C2}}t_{\mathrm{A3}}$

what we actually need to produce is a definition of the form

> `\def\u8:`$t_{\mathrm{C2}}t_{\mathrm{A3}}$ `{`*LaTeX internal form*`}`.

So here we temporarily redefine the prefix commands `\UTFviii@two@octets`, etc. to generate the csname that we wish to define¿ the `\string`s are added in case these tokens are still active.

```
    \def\UTFviii@two@octets##1##2{\csname u8:##1\string##2\endcsname}%
    \def\UTFviii@three@octets##1##2##3{\csname u8:##1%
                                       \string##2\string##3\endcsname}%
    \def\UTFviii@four@octets##1##2##3##4{\csname u8:##1%
                                \string##2\string##3\string##4\endcsname}%
```

Now we simply:-) need to use the right number of `\expandafter`s to finally construct the definition: expanding `\UTFviii@tmp` once to get its contents, a second time to replace the prefix command by its `\csname` expansion, and a third time to turn the expansion into a csname after which the `\gdef` finally gets applied. We add an irrelevant `\IeC` and braces around the definition, in order to avoid any space after the command being gobbled up when the text is written out to an auxiliary file (see `inputenc` for further details

```
    \expandafter\expandafter\expandafter
    \expandafter\expandafter\expandafter
    \expandafter
     \gdef\UTFviii@tmp{\IeC{#2}}%
   \endgroup
}
```

`\parse@XML@charref` This macro parses a Unicode number (decimal) and returns its UTF-8 representation as a sequence of non-active TEX character tokens. In the original code it

had two arguments delimited by ; here, however, we supply the Unicode number implicitly.

```
\gdef\parse@XML@charref{%
```

We need to keep a few things local, mainly the \uccode's that are set up below. However, the group originally used here is actually unnecessary since we call this macro only within another group; but it will be important to restore the group if this macro gets used for other purposes.

```
%  \begingroup
```

The original code from xmltex supported the convention that a Unicode slot number could be given either as a decimal or as a hexadecimal (by starting with x). We do not do this so this code is also removed. This could be reactivated if one wants to support document commands that accept Unicode numbers (but then the first case needs to be changed from an error message back to something more useful again).

```
%  \uppercase{\count@\if x\noexpand#1"\else#1\fi#2}\relax
```

As \count@ already contains the right value we make \parse@XML@charref work without arguments. In the case single byte UTF-8 sequences, only allow definition if the character os already active. The definition of \UTFviii@tmp looks slightly strange but is designed for the sequence of \expandafter in \DeclareUnicodeCharacter.

```
\ifnum\count@<"80\relax
  \ifnum\catcode\count@=13
    \uccode`\~=\count@\uppercase{\def\UTFviii@tmp{\@empty\@empty~}}%
  \else
    \@latex@error{Cannot define non-active Unicode char value < 0080}%
                  \@eha
    \def\UTFviii@tmp{\UTFviii@tmp}%
  \fi
```

The code below is derived from xmltex and generates the UTF-8 byte sequence for the number in \count@.

The reverse operation (just used in error messages) has now been added as \decode@UTFviii.

```
\else\ifnum\count@<"800\relax
   \parse@UTFviii@a,%
   \parse@UTFviii@b C\UTFviii@two@octets.,%
\else\ifnum\count@<"10000\relax
   \parse@UTFviii@a;%
   \parse@UTFviii@a,%
   \parse@UTFviii@b E\UTFviii@three@octets.{,;}%
 \else
```

Test added here for out of range values, the 4-octet definitions are still set up so that \DeclareUnicodeCharacter does something sensible if the user scrolls past this error.

```
    \ifnum\count@>"10FFFF\relax
         \@latex@error
              {\UTFviii@hexnumber\count@\space too large for Unicode}%
              {Values between 0 and 10FFFF are permitted}%
     \fi
```

```
      \parse@UTFviii@a;%
      \parse@UTFviii@a,%
      \parse@UTFviii@a!%
      \parse@UTFviii@b F\UTFviii@four@octets.{!,;}%
    \fi
    \fi
  \fi
%  \endgroup
}
```

`\parse@UTFviii@a` ... so somebody else can document this part :-)

```
\gdef\parse@UTFviii@a#1{%
    \@tempcnta\count@
    \divide\count@ 64
    \@tempcntb\count@
    \multiply\count@ 64
    \advance\@tempcnta-\count@
    \advance\@tempcnta 128
    \uccode`#1\@tempcnta
    \count@\@tempcntb}
```

`\parse@UTFviii@b` ... same here

```
\gdef\parse@UTFviii@b#1#2#3#4{%
    \advance\count@ "#10\relax
    \uccode`#3\count@
    \uppercase{\gdef\UTFviii@tmp{#2#3#4}}}
```

`\decode@UTFviii` In the reverse direction, take a sequence of octets(bytes) representing a character in UTF-8 and construct the Unicode number. The sequence is terminated by `\relax`.

In this version, if the sequence is not valid UTF-8 you probably get a low level arithmetic error from `\numexpr` or stray characters at the end. Getting a better error message would be somewhat expensive. As the main use is for reporting characters in messages, this is done just using expansion, so `\numexpr` is used, A stub returning 0 is defined if `\numexpr` is not available.

```
\ifx\numexpr\@undefined
\gdef\decode@UTFviii#1{0}
\else
```

If the input is malformed UTF-8 there may not be enough closing) so add 5 so there are always some remaining then cleanup and remove any remaining ones at the end. This avoids `\numexpr` parse errors while outputting a package error.

```
\gdef\decode@UTFviii#1\relax{%
  \expandafter\UTFviii@cleanup
    \the\numexpr\dec@de@UTFviii#1\relax)))))\@empty}
\gdef\UTFviii@cleanup#1)#2\@empty{#1}
\gdef\dec@de@UTFviii#1{%
\ifx\relax#1%
\else
  \ifnum`#1>"EF
    ((((`#1-"F0)%
```

```
  \else
    \ifnum`#1>"DF
      ((`#1-"E0)%
    \else
      \ifnum`#1>"BF
        ((`#1-"C0)%
      \else
        \ifnum`#1>"7F
        )*64+(`#1-"80)%
        \else
        +`#1 %
        \fi
      \fi
    \fi
  \fi
  \expandafter\dec@de@UTFviii
\fi}
\fi
```

`\UTFviii@hexnumber` Convert a number to a sequence of uppercase hex digits. If `\numexpr` is not available, it returns its argument unchanged.

```
\ifx\numexpr\@undefined
\global\let\UTFviii@hexnumber\@firstofone
\global\UTFviii@hexdigit\hexnumber@
\else
\gdef\UTFviii@hexnumber#1{%
\ifnum#1>15 %
\expandafter\UTFviii@hexnumber\expandafter{\the\numexpr(#1-8)/16\relax}%
\fi
\UTFviii@hexdigit{\numexpr#1\ifnum#1>0-((#1-8)/16)*16\fi\relax}%
}
```

Almost but not quite `\hexnumber@`.

```
\gdef\UTFviii@hexdigit#1{\ifcase\numexpr#1\relax
  0\or1\or2\or3\or4\or5\or6\or7\or8\or9\or
  A\or B\or C\or D\or E\or F\fi}
\fi
```

`\UTFviii@splitcsname` `\UTFviii@hexcodepoint` Split a csname representing a unicode character and return the character and the unicode number in hex.

```
\gdef\UTFviii@hexcodepoint#1{U+%
 \ifnum#1<16 0\fi
 \ifnum#1<256 0\fi
 \ifnum#1<4096 0\fi
 \UTFviii@hexnumber{#1}%
}%
\gdef\UTFviii@splitcsname#1:#2\relax{%
```

Need to pre-expand the argument to ensure cleanup in case of mal-formed UTF-8.

```
#2 (\expandafter\UTFviii@hexcodepoint\expandafter{%
                      \the\numexpr\decode@UTFviii#2\relax})%
}
```

```
\endgroup
\@onlypreamble\DeclareUnicodeCharacter
```

These are preamble only as long as we don't support Unicode charrefs in documents.

```
\@onlypreamble\parse@XML@charref
\@onlypreamble\parse@UTFviii@a
\@onlypreamble\parse@UTFviii@b
```

2.4 Loading Unicode mappings at begin document

The original plan was to set up the UTF-8 support at `\begin{document}`; but then any text characters used in the preamble (as people do even though advised against it) would fail in one way or the other. So the implementation was changed and the Unicode definition files for already defined encodings are loaded here.

We loop through all defined font encodings (stored in `\cdp@list`) and for each load a file *name*`enc.dfu` if it exist. That file is then supposed to contain `\DeclareUnicodeCharacter` declarations.

```
\begingroup
  \def\cdp@elt#1#2#3#4{%
    \wlog{Now handling font encoding #1 ...}%
    \lowercase{%
        \InputIfFileExists{#1enc.dfu}}%
           {\wlog{... processing UTF-8 mapping file for font %
                     encoding #1}%
```

The previous line is written to the log with the newline char being ignored (thus not producing a space). Therefore either everything has to be on a single input line or some special care must be taken. From this point on we ignore spaces again, i.e., while we are reading the `.dfu` file. The `\endgroup` below will restore it again.

```
            \catcode`\ 9\relax}%
          {\wlog{... no UTF-8 mapping file for font encoding #1}}%
  }
  \cdp@list
\endgroup
```

However, we don't know if there are font encodings still to be loaded (either with `fontenc` or directly with `\input` by some package). Font encoding files are loaded only if the corresponding encoding has not been loaded yet, and they always begin with `\DeclareFontEncoding`. We now redefine the internal kernel version of the latter to load the Unicode file if available.

```
\def\DeclareFontEncoding@#1#2#3{%
  \expandafter
  \ifx\csname T@#1\endcsname\relax
    \def\cdp@elt{\noexpand\cdp@elt}%
    \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
                   {\default@family}{\default@series}%
                   {\default@shape}}%
    \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
    \begingroup
      \wlog{Now handling font encoding #1 ...}%
      \lowercase{%
```

```
        \InputIfFileExists{#1enc.dfu}%
           {\wlog{... processing UTF-8 mapping file for font %
                      encoding #1}}%
           {\wlog{... no UTF-8 mapping file for font encoding #1}}%
     \endgroup
  \else
     \@font@info{Redeclaring font encoding #1}%
  \fi
  \global\@namedef{T@#1}{#2}%
  \global\@namedef{M@#1}{\default@M#3}%
  \xdef\LastDeclaredEncoding{#1}%
  }
```
⟨/utf8⟩

3 Mapping characters — based on font (glyph) encodings

This section is a first attempt to provide Unicode definitions for characters whose standard glyphs are currently provided by the standard LaTeX font-encodings `T1`, `OT1`, etc. They are by no means completed and need checking.

For example, one should check the already existing input encodings for glyphs that may in fact be available and required, e.g. `latin4` has a number of glyphs with the `\=` accent. Since the `T1` encoding does not provide such glyphs, these characters are not listed below (yet).

The list below was generated by looking at the current LaTeX font encoding files, e.g., `t1enc.def` and using the work by Sebastian Rahtz (in `ucharacters.sty`) with a few modifications. In combinations such as `\^\i` the preferred form is that and not `\^i`.

This list has been built from several sources, obviously including the Unicode Standard itself. These sources include Passive TeX by Sebastian Rahtz, the `unicode` package by Dominique P. G. Unruh (mainly for Latin encodings) and `text4ht` by Eitan Gurari (for Cyrillic ones).

Note that it strictly follows the Mittelbach principles for input character encodings: thus it offers no support for using utf8 representations of math symbols such as × or ÷ (in math mode).

3.1 About the table itself

In addition to generating individual files, the table below is, at present, a one-one (we think) partial relationship between the (ill-defined) set of LICRs and the Unicode slots "0080 to "FFFF. At present these entries are used only to define a collection of partial mappings from Unicode slots to LICRs; each of these mappings becomes full if we add an exception value ('not defined') to the set of LICRs.

It is probably not essential for the relationship in the full table to be one-one; this raises questions such as: the exact role of LICRs; the formal relationships on the set of LICRs; the (non-mathematical) relationship between LICRs and Unicode (which has its own somewhat fuzzy equivalences); and ultimately what a character is and what a character representation and/or name is.

It is unclear the extent to which entries in this table should resemble the closely related ones in the 8-bit `inputenc` files. The Unicode standard claims that the

first 256 slots 'are' ASCII and Latin-1.

Of course, TEX itself typically does not treat even many perfectly 'normal text' 7-bit slots as text characters, so it is unclear whether LATEX should even attempt to deal in any consistent way with those Unicode slots that are not definitive text characters.

3.2 The mapping table

Note that the first argument must be a hex-digit number greater than `00BF` and at most `10FFFF`.

There are few notes about inconsistencies etc at the end of the table.

```
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{00A0}{\nobreakspace}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{00A1}{\textexclamdown}
⟨all, ts1, ly1⟩\DeclareUnicodeCharacter{00A2}{\textcent}
⟨all, ts1, t1, ot1, ly1⟩\DeclareUnicodeCharacter{00A3}{\textsterling}
⟨all, x2, ts1, t2c, t2b, t2a, ly1, lcy⟩\DeclareUnicodeCharacter{00A4}{\textcurrency}
⟨all, ts1, ly1⟩\DeclareUnicodeCharacter{00A5}{\textyen}
⟨all, ts1, ly1⟩\DeclareUnicodeCharacter{00A6}{\textbrokenbar}
⟨all, x2, ts1, t2c, t2b, t2a, oms, ly1⟩\DeclareUnicodeCharacter{00A7}{\textsection}
⟨all, ts1⟩\DeclareUnicodeCharacter{00A8}{\textasciidieresis}
⟨all, ts1, utf8⟩\DeclareUnicodeCharacter{00A9}{\textcopyright}
⟨all, ts1, ly1, utf8⟩\DeclareUnicodeCharacter{00AA}{\textordfeminine}
⟨*all, x2, t2c, t2b, t2a, t1, ot2, ly1, lcy⟩
%\DeclareUnicodeCharacter{00AB}{\guillemotleft}  % wrong Adobe name
\DeclareUnicodeCharacter{00AB}{\guillemetleft}
⟨/all, x2, t2c, t2b, t2a, t1, ot2, ly1, lcy⟩
⟨all, ts1⟩\DeclareUnicodeCharacter{00AC}{\textlnot}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{00AD}{\-}
⟨all, ts1, ly1, utf8⟩\DeclareUnicodeCharacter{00AE}{\textregistered}
⟨all, ts1⟩\DeclareUnicodeCharacter{00AF}{\textasciimacron}
⟨all, ts1, ly1⟩\DeclareUnicodeCharacter{00B0}{\textdegree}
⟨all, ts1⟩\DeclareUnicodeCharacter{00B1}{\textpm}
⟨all, ts1⟩\DeclareUnicodeCharacter{00B2}{\texttwosuperior}
⟨all, ts1⟩\DeclareUnicodeCharacter{00B3}{\textthreesuperior}
⟨all, ts1⟩\DeclareUnicodeCharacter{00B4}{\textasciiacute}
⟨all, ts1, ly1⟩\DeclareUnicodeCharacter{00B5}{\textmu} % micro sign
⟨all, ts1, oms, ly1⟩\DeclareUnicodeCharacter{00B6}{\textparagraph}
⟨all, oms, ts1, ly1⟩\DeclareUnicodeCharacter{00B7}{\textperiodcentered}
⟨all, ot1⟩\DeclareUnicodeCharacter{00B8}{\c\ }
⟨all, ts1⟩\DeclareUnicodeCharacter{00B9}{\textonesuperior}
⟨all, ts1, ly1, utf8⟩\DeclareUnicodeCharacter{00BA}{\textordmasculine}
⟨*all, x2, t2c, t2b, t2a, t1, ot2, ly1, lcy⟩
%\DeclareUnicodeCharacter{00BB}{\guillemotright} % wrong Adobe name
\DeclareUnicodeCharacter{00BB}{\guillemetright}
⟨/all, x2, t2c, t2b, t2a, t1, ot2, ly1, lcy⟩
⟨all, ts1, ly1⟩\DeclareUnicodeCharacter{00BC}{\textonequarter}
⟨all, ts1, ly1⟩\DeclareUnicodeCharacter{00BD}{\textonehalf}
⟨all, ts1, ly1⟩\DeclareUnicodeCharacter{00BE}{\textthreequarters}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{00BF}{\textquestiondown}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00C0}{\@tabacckludge`A}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00C1}{\@tabacckludge'A}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00C2}{\^A}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00C3}{\~A}
```

```
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00C4}{\"A}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{00C5}{\r A}
⟨all, t1, ot1, ly1, lcy⟩\DeclareUnicodeCharacter{00C6}{\AE}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00C7}{\c C}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00C8}{\@tabacckludge‘E}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00C9}{\@tabacckludge’E}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00CA}{\^E}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00CB}{\"E}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00CC}{\@tabacckludge‘I}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00CD}{\@tabacckludge’I}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00CE}{\^I}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00CF}{\"I}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00D0}{\DH}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00D1}{\~N}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00D2}{\@tabacckludge‘O}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00D3}{\@tabacckludge’O}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00D4}{\^O}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00D5}{\~O}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00D6}{\"O}
⟨all, ts1⟩\DeclareUnicodeCharacter{00D7}{\texttimes}
⟨all, t1, ot1, ly1, lcy⟩\DeclareUnicodeCharacter{00D8}{\O}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00D9}{\@tabacckludge‘U}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00DA}{\@tabacckludge’U}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00DB}{\^U}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00DC}{\"U}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00DD}{\@tabacckludge’Y}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00DE}{\TH}
⟨all, t1, ot1, ly1, lcy⟩\DeclareUnicodeCharacter{00DF}{\ss}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00E0}{\@tabacckludge‘a}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00E1}{\@tabacckludge’a}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00E2}{\^a}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00E3}{\~a}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00E4}{\"a}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00E5}{\r a}
⟨all, t1, ot1, ly1, lcy⟩\DeclareUnicodeCharacter{00E6}{\ae}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00E7}{\c c}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00E8}{\@tabacckludge‘e}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00E9}{\@tabacckludge’e}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00EA}{\^e}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00EB}{\"e}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{00EC}{\@tabacckludge‘\i}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{00ED}{\@tabacckludge’\i}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{00EE}{\^\i}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{00EF}{\"\i}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00F0}{\dh}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00F1}{\~n}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00F2}{\@tabacckludge‘o}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00F3}{\@tabacckludge’o}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00F4}{\^o}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00F5}{\~o}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00F6}{\"o}
⟨all, ts1⟩\DeclareUnicodeCharacter{00F7}{\textdiv}
⟨all, t1, ot1, ly1, lcy⟩\DeclareUnicodeCharacter{00F8}{\o}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00F9}{\@tabacckludge‘u}
```

⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00FA}{\@tabacckludge'u}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00FB}{\^u}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00FC}{\"u}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00FD}{\@tabacckludge'y}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00FE}{\th}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{00FF}{\"y}
⟨all, t1⟩\DeclareUnicodeCharacter{0100}{\@tabacckludge=A}
⟨all, t1⟩\DeclareUnicodeCharacter{0101}{\@tabacckludge=a}
⟨all, t1⟩\DeclareUnicodeCharacter{0102}{\u A}
⟨all, t1⟩\DeclareUnicodeCharacter{0103}{\u a}
⟨all, t1⟩\DeclareUnicodeCharacter{0104}{\k A}
⟨all, t1⟩\DeclareUnicodeCharacter{0105}{\k a}
⟨all, t1⟩\DeclareUnicodeCharacter{0106}{\@tabacckludge'C}
⟨all, t1⟩\DeclareUnicodeCharacter{0107}{\@tabacckludge'c}
⟨all, t1⟩\DeclareUnicodeCharacter{0108}{\^C}
⟨all, t1⟩\DeclareUnicodeCharacter{0109}{\^c}
⟨all, t1⟩\DeclareUnicodeCharacter{010A}{\.C}
⟨all, t1⟩\DeclareUnicodeCharacter{010B}{\.c}
⟨all, t1⟩\DeclareUnicodeCharacter{010C}{\v C}
⟨all, t1⟩\DeclareUnicodeCharacter{010D}{\v c}
⟨all, t1⟩\DeclareUnicodeCharacter{010E}{\v D}
⟨all, t1⟩\DeclareUnicodeCharacter{010F}{\v d}
⟨all, t1⟩\DeclareUnicodeCharacter{0110}{\DJ}
⟨all, t1⟩\DeclareUnicodeCharacter{0111}{\dj}
⟨all, t1⟩\DeclareUnicodeCharacter{0112}{\@tabacckludge=E}
⟨all, t1⟩\DeclareUnicodeCharacter{0113}{\@tabacckludge=e}
⟨all, t1⟩\DeclareUnicodeCharacter{0114}{\u E}
⟨all, t1⟩\DeclareUnicodeCharacter{0115}{\u e}
⟨all, t1⟩\DeclareUnicodeCharacter{0116}{\.E}
⟨all, t1⟩\DeclareUnicodeCharacter{0117}{\.e}
⟨all, t1⟩\DeclareUnicodeCharacter{0118}{\k E}
⟨all, t1⟩\DeclareUnicodeCharacter{0119}{\k e}
⟨all, t1⟩\DeclareUnicodeCharacter{011A}{\v E}
⟨all, t1⟩\DeclareUnicodeCharacter{011B}{\v e}
⟨all, t1⟩\DeclareUnicodeCharacter{011C}{\^G}
⟨all, t1⟩\DeclareUnicodeCharacter{011D}{\^g}
⟨all, t1⟩\DeclareUnicodeCharacter{011E}{\u G}
⟨all, t1⟩\DeclareUnicodeCharacter{011F}{\u g}
⟨all, t1⟩\DeclareUnicodeCharacter{0120}{\.G}
⟨all, t1⟩\DeclareUnicodeCharacter{0121}{\.g}
⟨all, t1⟩\DeclareUnicodeCharacter{0122}{\c G}
⟨all, t1⟩\DeclareUnicodeCharacter{0123}{\c g}
⟨all, t1⟩\DeclareUnicodeCharacter{0124}{\^H}
⟨all, t1⟩\DeclareUnicodeCharacter{0125}{\^h}
⟨all, t1⟩\DeclareUnicodeCharacter{0128}{\~I}
⟨all, t1⟩\DeclareUnicodeCharacter{0129}{\~\i}
⟨all, t1⟩\DeclareUnicodeCharacter{012A}{\@tabacckludge=I}
⟨all, t1⟩\DeclareUnicodeCharacter{012B}{\@tabacckludge=\i}
⟨all, t1⟩\DeclareUnicodeCharacter{012C}{\u I}
⟨all, t1⟩\DeclareUnicodeCharacter{012D}{\u\i}
⟨all, t1⟩\DeclareUnicodeCharacter{012E}{\k I}
⟨all, t1⟩\DeclareUnicodeCharacter{012F}{\k i}
⟨all, t1⟩\DeclareUnicodeCharacter{0130}{\.I}
⟨all, t2c, t2b, t2a, t1, ot2, ot1, ly1, lcy⟩\DeclareUnicodeCharacter{0131}{\i}

```
⟨all, t1⟩\DeclareUnicodeCharacter{0132}{\IJ}
⟨all, t1⟩\DeclareUnicodeCharacter{0133}{\ij}
⟨all, t1⟩\DeclareUnicodeCharacter{0134}{\^J}
⟨all, t1⟩\DeclareUnicodeCharacter{0135}{\^\j}
⟨all, t1⟩\DeclareUnicodeCharacter{0136}{\c K}
⟨all, t1⟩\DeclareUnicodeCharacter{0137}{\c k}
⟨all, t1⟩\DeclareUnicodeCharacter{0139}{\@tabacckludge'L}
⟨all, t1⟩\DeclareUnicodeCharacter{013A}{\@tabacckludge'l}
⟨all, t1⟩\DeclareUnicodeCharacter{013B}{\c L}
⟨all, t1⟩\DeclareUnicodeCharacter{013C}{\c l}
⟨all, t1⟩\DeclareUnicodeCharacter{013D}{\v L}
⟨all, t1⟩\DeclareUnicodeCharacter{013E}{\v l}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{0141}{\L}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{0142}{\l}
⟨all, t1⟩\DeclareUnicodeCharacter{0143}{\@tabacckludge'N}
⟨all, t1⟩\DeclareUnicodeCharacter{0144}{\@tabacckludge'n}
⟨all, t1⟩\DeclareUnicodeCharacter{0145}{\c N}
⟨all, t1⟩\DeclareUnicodeCharacter{0146}{\c n}
⟨all, t1⟩\DeclareUnicodeCharacter{0147}{\v N}
⟨all, t1⟩\DeclareUnicodeCharacter{0148}{\v n}
⟨all, t1⟩\DeclareUnicodeCharacter{014A}{\NG}
⟨all, t1⟩\DeclareUnicodeCharacter{014B}{\ng}
⟨all, t1⟩\DeclareUnicodeCharacter{014C}{\@tabacckludge=O}
⟨all, t1⟩\DeclareUnicodeCharacter{014D}{\@tabacckludge=o}
⟨all, t1⟩\DeclareUnicodeCharacter{014E}{\u O}
⟨all, t1⟩\DeclareUnicodeCharacter{014F}{\u o}
⟨all, t1⟩\DeclareUnicodeCharacter{0150}{\H O}
⟨all, t1⟩\DeclareUnicodeCharacter{0151}{\H o}
⟨all, t1, ot1, ly1, lcy⟩\DeclareUnicodeCharacter{0152}{\OE}
⟨all, t1, ot1, ly1, lcy⟩\DeclareUnicodeCharacter{0153}{\oe}
⟨all, t1⟩\DeclareUnicodeCharacter{0154}{\@tabacckludge'R}
⟨all, t1⟩\DeclareUnicodeCharacter{0155}{\@tabacckludge'r}
⟨all, t1⟩\DeclareUnicodeCharacter{0156}{\c R}
⟨all, t1⟩\DeclareUnicodeCharacter{0157}{\c r}
⟨all, t1⟩\DeclareUnicodeCharacter{0158}{\v R}
⟨all, t1⟩\DeclareUnicodeCharacter{0159}{\v r}
⟨all, t1⟩\DeclareUnicodeCharacter{015A}{\@tabacckludge'S}
⟨all, t1⟩\DeclareUnicodeCharacter{015B}{\@tabacckludge's}
⟨all, t1⟩\DeclareUnicodeCharacter{015C}{\^S}
⟨all, t1⟩\DeclareUnicodeCharacter{015D}{\^s}
⟨all, t1⟩\DeclareUnicodeCharacter{015E}{\c S}
⟨all, t1⟩\DeclareUnicodeCharacter{015F}{\c s}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{0160}{\v S}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{0161}{\v s}
⟨all, t1⟩\DeclareUnicodeCharacter{0162}{\c T}
⟨all, t1⟩\DeclareUnicodeCharacter{0163}{\c t}
⟨all, t1⟩\DeclareUnicodeCharacter{0164}{\v T}
⟨all, t1⟩\DeclareUnicodeCharacter{0165}{\v t}
⟨all, t1⟩\DeclareUnicodeCharacter{0168}{\~U}
⟨all, t1⟩\DeclareUnicodeCharacter{0169}{\~u}
⟨all, t1⟩\DeclareUnicodeCharacter{016A}{\@tabacckludge=U}
⟨all, t1⟩\DeclareUnicodeCharacter{016B}{\@tabacckludge=u}
⟨all, t1⟩\DeclareUnicodeCharacter{016C}{\u U}
⟨all, t1⟩\DeclareUnicodeCharacter{016D}{\u u}
```

```
⟨all, t1⟩\DeclareUnicodeCharacter{016E}{\r U}
⟨all, t1⟩\DeclareUnicodeCharacter{016F}{\r u}
⟨all, t1⟩\DeclareUnicodeCharacter{0170}{\H U}
⟨all, t1⟩\DeclareUnicodeCharacter{0171}{\H u}
⟨all, t1⟩\DeclareUnicodeCharacter{0172}{\k U}
⟨all, t1⟩\DeclareUnicodeCharacter{0173}{\k u}

⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{0174}{\^W}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{0175}{\^w}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{0176}{\^Y}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{0177}{\^y}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{0178}{\"Y}
⟨all, t1⟩\DeclareUnicodeCharacter{0179}{\@tabacckludge'Z}
⟨all, t1⟩\DeclareUnicodeCharacter{017A}{\@tabacckludge'z}
⟨all, t1⟩\DeclareUnicodeCharacter{017B}{\.Z}
⟨all, t1⟩\DeclareUnicodeCharacter{017C}{\.z}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{017D}{\v Z}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{017E}{\v z}
⟨all, ts1, ly1⟩\DeclareUnicodeCharacter{0192}{\textflorin}

⟨all, t1⟩\DeclareUnicodeCharacter{01C4}{D\v Z}
⟨all, t1⟩\DeclareUnicodeCharacter{01C5}{D\v z}
⟨all, t1⟩\DeclareUnicodeCharacter{01C6}{d\v z}
⟨all, t1⟩\DeclareUnicodeCharacter{01C7}{LJ}
⟨all, t1⟩\DeclareUnicodeCharacter{01C8}{Lj}
⟨all, t1⟩\DeclareUnicodeCharacter{01C9}{lj}
⟨all, t1⟩\DeclareUnicodeCharacter{01CA}{NJ}
⟨all, t1⟩\DeclareUnicodeCharacter{01CB}{Nj}
⟨all, t1⟩\DeclareUnicodeCharacter{01CC}{nj}

⟨all, t1⟩\DeclareUnicodeCharacter{01CD}{\v A}
⟨all, t1⟩\DeclareUnicodeCharacter{01CE}{\v a}
⟨all, t1⟩\DeclareUnicodeCharacter{01CF}{\v I}
⟨all, t1⟩\DeclareUnicodeCharacter{01D0}{\v \i}
⟨all, t1⟩\DeclareUnicodeCharacter{01D1}{\v O}
⟨all, t1⟩\DeclareUnicodeCharacter{01D2}{\v o}
⟨all, t1⟩\DeclareUnicodeCharacter{01D3}{\v U}
⟨all, t1⟩\DeclareUnicodeCharacter{01D4}{\v u}
⟨all, t1⟩\DeclareUnicodeCharacter{01E2}{\@tabacckludge=\AE}
⟨all, t1⟩\DeclareUnicodeCharacter{01E3}{\@tabacckludge=\ae}
⟨all, t1⟩\DeclareUnicodeCharacter{01E6}{\v G}
⟨all, t1⟩\DeclareUnicodeCharacter{01E7}{\v g}
⟨all, t1⟩\DeclareUnicodeCharacter{01E8}{\v K}
⟨all, t1⟩\DeclareUnicodeCharacter{01E9}{\v k}
⟨all, t1⟩\DeclareUnicodeCharacter{01EA}{\k O}
⟨all, t1⟩\DeclareUnicodeCharacter{01EB}{\k o}
⟨all, t1⟩\DeclareUnicodeCharacter{01F0}{\v\j}
⟨all, t1⟩\DeclareUnicodeCharacter{01F4}{\@tabacckludge'G}
⟨all, t1⟩\DeclareUnicodeCharacter{01F5}{\@tabacckludge'g}

⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{0218}{\textcommabelow S}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{0219}{\textcommabelow s}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{021A}{\textcommabelow T}
⟨all, t1, ot1, ly1⟩\DeclareUnicodeCharacter{021B}{\textcommabelow t}

⟨all, t1⟩\DeclareUnicodeCharacter{0232}{\@tabacckludge=Y}
⟨all, t1⟩\DeclareUnicodeCharacter{0233}{\@tabacckludge=y}
```

```
593 ⟨all, t2c, t2b, t2a, t1, ot2, ot1, ly1, lcy⟩\DeclareUnicodeCharacter{0237}{\j}
594 ⟨all, ly1, utf8⟩\DeclareUnicodeCharacter{02C6}{\textasciicircum}
595 ⟨all, ts1⟩\DeclareUnicodeCharacter{02C7}{\textasciicaron}
596 ⟨all, ly1, utf8⟩\DeclareUnicodeCharacter{02DC}{\textasciitilde}
597 ⟨all, ts1⟩\DeclareUnicodeCharacter{02D8}{\textasciibreve}
598 ⟨all, t1⟩\DeclareUnicodeCharacter{02D9}{\.{}}
599 ⟨all, t1⟩\DeclareUnicodeCharacter{02DB}{\k{}}
600 ⟨all, ts1⟩\DeclareUnicodeCharacter{02DD}{\textacutedbl}
```

The Cyrillic code points have been recently checked (2007) and extended and corrected by Matthias Noe (a9931078@unet.univie.ac.at) — thanks.

```
601 ⟨*all, x2, t2c, t2b, t2a, ot2, lcy⟩
602 \DeclareUnicodeCharacter{0400}{\@tabacckludge`\CYRE}
603 ⟨/all, x2, t2c, t2b, t2a, ot2, lcy⟩
604 ⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{0401}{\CYRYO}
605 ⟨all, x2, t2a, ot2⟩\DeclareUnicodeCharacter{0402}{\CYRDJE}
606 ⟨*all, x2, t2c, t2b, t2a, ot2, lcy⟩
607 \DeclareUnicodeCharacter{0403}{\@tabacckludge'\CYRG}
608 ⟨/all, x2, t2c, t2b, t2a, ot2, lcy⟩
609 ⟨all, x2, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{0404}{\CYRIE}
610 ⟨all, x2, t2c, t2b, t2a, ot2⟩\DeclareUnicodeCharacter{0405}{\CYRDZE}
611 ⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{0406}{\CYRII}
612 ⟨all, x2, t2a, lcy⟩\DeclareUnicodeCharacter{0407}{\CYRYI}
613 ⟨all, x2, t2c, t2b, t2a, ot2⟩\DeclareUnicodeCharacter{0408}{\CYRJE}
614 ⟨all, x2, t2b, t2a, ot2⟩\DeclareUnicodeCharacter{0409}{\CYRLJE}
615 ⟨all, x2, t2b, t2a, ot2⟩\DeclareUnicodeCharacter{040A}{\CYRNJE}
616 ⟨all, x2, t2a, ot2⟩\DeclareUnicodeCharacter{040B}{\CYRTSHE}
617 ⟨*all, x2, t2c, t2b, t2a, ot2, lcy⟩
618 \DeclareUnicodeCharacter{040C}{\@tabacckludge'\CYRK}
619 \DeclareUnicodeCharacter{040D}{\@tabacckludge`\CYRI}
620 ⟨/all, x2, t2c, t2b, t2a, ot2, lcy⟩
621 ⟨all, x2, t2b, t2a, lcy⟩\DeclareUnicodeCharacter{040E}{\CYRUSHRT}
622 ⟨all, x2, t2c, t2a, ot2⟩\DeclareUnicodeCharacter{040F}{\CYRDZHE}
623 ⟨*all, x2, t2c, t2b, t2a, ot2, lcy⟩
624 \DeclareUnicodeCharacter{0410}{\CYRA}
625 \DeclareUnicodeCharacter{0411}{\CYRB}
626 \DeclareUnicodeCharacter{0412}{\CYRV}
627 \DeclareUnicodeCharacter{0413}{\CYRG}
628 \DeclareUnicodeCharacter{0414}{\CYRD}
629 \DeclareUnicodeCharacter{0415}{\CYRE}
630 \DeclareUnicodeCharacter{0416}{\CYRZH}
631 \DeclareUnicodeCharacter{0417}{\CYRZ}
632 \DeclareUnicodeCharacter{0418}{\CYRI}
633 \DeclareUnicodeCharacter{0419}{\CYRISHRT}
634 \DeclareUnicodeCharacter{041A}{\CYRK}
635 \DeclareUnicodeCharacter{041B}{\CYRL}
636 \DeclareUnicodeCharacter{041C}{\CYRM}
637 \DeclareUnicodeCharacter{041D}{\CYRN}
638 \DeclareUnicodeCharacter{041E}{\CYRO}
639 \DeclareUnicodeCharacter{041F}{\CYRP}
640 \DeclareUnicodeCharacter{0420}{\CYRR}
641 \DeclareUnicodeCharacter{0421}{\CYRS}
642 \DeclareUnicodeCharacter{0422}{\CYRT}
643 \DeclareUnicodeCharacter{0423}{\CYRU}
```

```
\DeclareUnicodeCharacter{0424}{\CYRF}
\DeclareUnicodeCharacter{0425}{\CYRH}
\DeclareUnicodeCharacter{0426}{\CYRC}
\DeclareUnicodeCharacter{0427}{\CYRCH}
\DeclareUnicodeCharacter{0428}{\CYRSH}
\DeclareUnicodeCharacter{0429}{\CYRSHCH}
\DeclareUnicodeCharacter{042A}{\CYRHRDSN}
\DeclareUnicodeCharacter{042B}{\CYRERY}
\DeclareUnicodeCharacter{042C}{\CYRSFTSN}
\DeclareUnicodeCharacter{042D}{\CYREREV}
\DeclareUnicodeCharacter{042E}{\CYRYU}
\DeclareUnicodeCharacter{042F}{\CYRYA}
\DeclareUnicodeCharacter{0430}{\cyra}
\DeclareUnicodeCharacter{0431}{\cyrb}
\DeclareUnicodeCharacter{0432}{\cyrv}
\DeclareUnicodeCharacter{0433}{\cyrg}
\DeclareUnicodeCharacter{0434}{\cyrd}
\DeclareUnicodeCharacter{0435}{\cyre}
\DeclareUnicodeCharacter{0436}{\cyrzh}
\DeclareUnicodeCharacter{0437}{\cyrz}
\DeclareUnicodeCharacter{0438}{\cyri}
\DeclareUnicodeCharacter{0439}{\cyrishrt}
\DeclareUnicodeCharacter{043A}{\cyrk}
\DeclareUnicodeCharacter{043B}{\cyrl}
\DeclareUnicodeCharacter{043C}{\cyrm}
\DeclareUnicodeCharacter{043D}{\cyrn}
\DeclareUnicodeCharacter{043E}{\cyro}
\DeclareUnicodeCharacter{043F}{\cyrp}
\DeclareUnicodeCharacter{0440}{\cyrr}
\DeclareUnicodeCharacter{0441}{\cyrs}
\DeclareUnicodeCharacter{0442}{\cyrt}
\DeclareUnicodeCharacter{0443}{\cyru}
\DeclareUnicodeCharacter{0444}{\cyrf}
\DeclareUnicodeCharacter{0445}{\cyrh}
\DeclareUnicodeCharacter{0446}{\cyrc}
\DeclareUnicodeCharacter{0447}{\cyrch}
\DeclareUnicodeCharacter{0448}{\cyrsh}
\DeclareUnicodeCharacter{0449}{\cyrshch}
\DeclareUnicodeCharacter{044A}{\cyrhrdsn}
\DeclareUnicodeCharacter{044B}{\cyrery}
\DeclareUnicodeCharacter{044C}{\cyrsftsn}
\DeclareUnicodeCharacter{044D}{\cyrerev}
\DeclareUnicodeCharacter{044E}{\cyryu}
\DeclareUnicodeCharacter{044F}{\cyrya}
\DeclareUnicodeCharacter{0450}{\@tabacckludge`\cyre}
\DeclareUnicodeCharacter{0451}{\cyryo}
⟨/all, x2, t2c, t2b, t2a, ot2, lcy⟩
⟨all, x2, t2a, ot2⟩\DeclareUnicodeCharacter{0452}{\cyrdje}
⟨*all, x2, t2c, t2b, t2a, ot2, lcy⟩
\DeclareUnicodeCharacter{0453}{\@tabacckludge'\cyrg}
⟨/all, x2, t2c, t2b, t2a, ot2, lcy⟩
⟨all, x2, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{0454}{\cyrie}
⟨all, x2, t2c, t2b, t2a, ot2⟩\DeclareUnicodeCharacter{0455}{\cyrdze}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{0456}{\cyrii}
```

```
⟨all, x2, t2a, lcy⟩\DeclareUnicodeCharacter{0457}{\cyryi}
⟨all, x2, t2c, t2b, t2a, ot2⟩\DeclareUnicodeCharacter{0458}{\cyrje}
⟨all, x2, t2b, t2a, ot2⟩\DeclareUnicodeCharacter{0459}{\cyrlje}
⟨all, x2, t2b, t2a, ot2⟩\DeclareUnicodeCharacter{045A}{\cyrnje}
⟨all, x2, t2a, ot2⟩\DeclareUnicodeCharacter{045B}{\cyrtshe}
⟨*all, x2, t2c, t2b, t2a, ot2, lcy⟩
\DeclareUnicodeCharacter{045C}{\@tabacckludge'\cyrk}
\DeclareUnicodeCharacter{045D}{\@tabacckludge`\cyri}
⟨/all, x2, t2c, t2b, t2a, ot2, lcy⟩
⟨all, x2, t2b, t2a, lcy⟩\DeclareUnicodeCharacter{045E}{\cyrushrt}
⟨all, x2, t2c, t2a, ot2⟩\DeclareUnicodeCharacter{045F}{\cyrdzhe}
⟨all, x2, ot2⟩\DeclareUnicodeCharacter{0462}{\CYRYAT}
⟨all, x2, ot2⟩\DeclareUnicodeCharacter{0463}{\cyryat}
⟨all, x2⟩\DeclareUnicodeCharacter{046A}{\CYRBYUS}
⟨all, x2⟩\DeclareUnicodeCharacter{046B}{\cyrbyus}
```

The next two declarations are questionable, the encoding definition should probably contain `\CYROTLD` and `\cyrotld`. Or alternatively, if the characters in the X2 encodings are really meant to represent the historical characters in Ux0472 and Ux0473 (they look like them) then they would need to change instead.

However, their looks are probably a font designers decision and the next two mappings are wrong or rather the names in OT2 should change for consistency.

On the other hand the names `\CYROTLD` are somewhat questionable as the Unicode standard only describes "Cyrillic barred O" while `TLD` refers to a tilde (which is more less what the "Cyrillic FITA looks according to the Unicode book).

```
⟨all, ot2⟩\DeclareUnicodeCharacter{0472}{\CYRFITA}
⟨all, ot2⟩\DeclareUnicodeCharacter{0473}{\cyrfita}

⟨all, x2, ot2⟩\DeclareUnicodeCharacter{0474}{\CYRIZH}
⟨all, x2, ot2⟩\DeclareUnicodeCharacter{0475}{\cyrizh}
```

While the double grave accent seems to exist in X2, T2A, T2B and T2C encoding, the letter izhitsa exists only in X2 and OT2. Therefore, izhitsa with double grave seems to be possible only using X2.

```
⟨all, x2⟩\DeclareUnicodeCharacter{0476}{\C\CYRIZH}
⟨all, x2⟩\DeclareUnicodeCharacter{0477}{\C\cyrizh}

⟨all, t2c⟩\DeclareUnicodeCharacter{048C}{\CYRSEMISFTSN}
⟨all, t2c⟩\DeclareUnicodeCharacter{048D}{\cyrsemisftsn}
⟨all, t2c⟩\DeclareUnicodeCharacter{048E}{\CYRRTICK}
⟨all, t2c⟩\DeclareUnicodeCharacter{048F}{\cyrrtick}
⟨all, x2, t2a, lcy⟩\DeclareUnicodeCharacter{0490}{\CYRGUP}
⟨all, x2, t2a, lcy⟩\DeclareUnicodeCharacter{0491}{\cyrgup}
⟨all, x2, t2b, t2a⟩\DeclareUnicodeCharacter{0492}{\CYRGHCRS}
⟨all, x2, t2b, t2a⟩\DeclareUnicodeCharacter{0493}{\cyrghcrs}
⟨all, x2, t2c, t2b⟩\DeclareUnicodeCharacter{0494}{\CYRGHK}
⟨all, x2, t2c, t2b⟩\DeclareUnicodeCharacter{0495}{\cyrghk}
⟨all, x2, t2b, t2a⟩\DeclareUnicodeCharacter{0496}{\CYRZHDSC}
⟨all, x2, t2b, t2a⟩\DeclareUnicodeCharacter{0497}{\cyrzhdsc}
⟨all, x2, t2a⟩\DeclareUnicodeCharacter{0498}{\CYRZDSC}
⟨all, x2, t2a⟩\DeclareUnicodeCharacter{0499}{\cyrzdsc}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{049A}{\CYRKDSC}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{049B}{\cyrkdsc}
⟨all, x2, t2a⟩\DeclareUnicodeCharacter{049C}{\CYRKVCRS}
⟨all, x2, t2a⟩\DeclareUnicodeCharacter{049D}{\cyrkvcrs}
```

```
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{049E}{\CYRKHCRS}
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{049F}{\cyrkhcrs}
⟨all, x2, t2a⟩\DeclareUnicodeCharacter{04A0}{\CYRKBEAK}
⟨all, x2, t2a⟩\DeclareUnicodeCharacter{04A1}{\cyrkbeak}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04A2}{\CYRNDSC}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04A3}{\cyrndsc}
⟨all, x2, t2b, t2a⟩\DeclareUnicodeCharacter{04A4}{\CYRNG}
⟨all, x2, t2b, t2a⟩\DeclareUnicodeCharacter{04A5}{\cyrng}
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{04A6}{\CYRPHK}
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{04A7}{\cyrphk}
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{04A8}{\CYRABHHA}
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{04A9}{\cyrabhha}
⟨all, x2, t2a⟩\DeclareUnicodeCharacter{04AA}{\CYRSDSC}
⟨all, x2, t2a⟩\DeclareUnicodeCharacter{04AB}{\cyrsdsc}
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{04AC}{\CYRTDSC}
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{04AD}{\cyrtdsc}
⟨all, x2, t2b, t2a⟩\DeclareUnicodeCharacter{04AE}{\CYRY}
⟨all, x2, t2b, t2a⟩\DeclareUnicodeCharacter{04AF}{\cyry}
⟨all, x2, t2a⟩\DeclareUnicodeCharacter{04B0}{\CYRYHCRS}
⟨all, x2, t2a⟩\DeclareUnicodeCharacter{04B1}{\cyryhcrs}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04B2}{\CYRHDSC}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04B3}{\cyrhdsc}
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{04B4}{\CYRTETSE}
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{04B5}{\cyrtetse}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04B6}{\CYRCHRDSC}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04B7}{\cyrchrdsc}
⟨all, x2, t2a⟩\DeclareUnicodeCharacter{04B8}{\CYRCHVCRS}
⟨all, x2, t2a⟩\DeclareUnicodeCharacter{04B9}{\cyrchvcrs}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04BA}{\CYRSHHA}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04BB}{\cyrshha}
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{04BC}{\CYRABHCH}
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{04BD}{\cyrabhch}
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{04BE}{\CYRABHCHDSC}
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{04BF}{\cyrabhchdsc}
```

The character `\CYRpalochka` is not defined by OT2 and LCY. However it is looking identical to `\CYRII` and the Unicode standard explicitly refers to that (and to Latin I). So perhaps those encodings could get an alias? On the other hand, why are there two distinct slots in the T2 encodings even though they are so pressed for space? Perhaps they don't always look alike.

```
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04C0}{\CYRpalochka}

⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04C1}{\U\CYRZH}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04C2}{\U\cyrzh}
⟨all, x2, t2b⟩\DeclareUnicodeCharacter{04C3}{\CYRKHK}
⟨all, x2, t2b⟩\DeclareUnicodeCharacter{04C4}{\cyrkhk}
```

According to the Unicode standard Ux04C5 should be an L with "tail" not with descender (which also exists as Ux04A2) but it looks as if the char names do not make this distinction. Should they?

```
⟨all, x2, t2c, t2b⟩\DeclareUnicodeCharacter{04C5}{\CYRLDSC}
⟨all, x2, t2c, t2b⟩\DeclareUnicodeCharacter{04C6}{\cyrldsc}

⟨all, x2, t2c, t2b⟩\DeclareUnicodeCharacter{04C7}{\CYRNHK}
⟨all, x2, t2c, t2b⟩\DeclareUnicodeCharacter{04C8}{\cyrnhk}
```

```
⟨all, x2, t2b⟩\DeclareUnicodeCharacter{04CB}{\CYRCHLDSC}
⟨all, x2, t2b⟩\DeclareUnicodeCharacter{04CC}{\cyrchldsc}
```

According to the Unicode standard Ux04CD should be an M with "tail" not with descender. However this time there is no M with descender in the Unicode standard.

```
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{04CD}{\CYRMDSC}
⟨all, x2, t2c⟩\DeclareUnicodeCharacter{04CE}{\cyrmdsc}

⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04D0}{\U\CYRA}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04D1}{\U\cyra}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04D2}{\"\CYRA}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04D3}{\"\cyra}
⟨all, x2, t2a⟩\DeclareUnicodeCharacter{04D4}{\CYRAE}
⟨all, x2, t2a⟩\DeclareUnicodeCharacter{04D5}{\cyrae}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04D6}{\U\CYRE}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04D7}{\U\cyre}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04D8}{\CYRSCHWA}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04D9}{\cyrschwa}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04DA}{\"\CYRSCHWA}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04DB}{\"\cyrschwa}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04DC}{\"\CYRZH}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04DD}{\"\cyrzh}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04DE}{\"\CYRZ}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04DF}{\"\cyrz}
⟨all, x2, t2c, t2b⟩\DeclareUnicodeCharacter{04E0}{\CYRABHDZE}
⟨all, x2, t2c, t2b⟩\DeclareUnicodeCharacter{04E1}{\cyrabhdze}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04E2}{\@tabacckludge=\CYRI}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04E3}{\@tabacckludge=\cyri}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04E4}{\"\CYRI}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04E5}{\"\cyri}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04E6}{\"\CYRO}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04E7}{\"\cyro}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04E8}{\CYROTLD}
⟨all, x2, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{04E9}{\cyrotld}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04EC}{\"\CYREREV}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04ED}{\"\cyrerev}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04EE}{\@tabacckludge=\CYRU}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04EF}{\@tabacckludge=\cyru}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04F0}{\"\CYRU}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04F1}{\"\cyru}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04F2}{\H\CYRU}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04F3}{\H\cyru}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04F4}{\"\CYRCH}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04F5}{\"\cyrch}
⟨all, x2, t2b⟩\DeclareUnicodeCharacter{04F6}{\CYRGDSC}
⟨all, x2, t2b⟩\DeclareUnicodeCharacter{04F7}{\cyrgdsc}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04F8}{\"\CYRERY}
⟨all, x2, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{04F9}{\"\cyrery}
⟨all, t2b⟩\DeclareUnicodeCharacter{04FA}{\CYRGDSCHCRS}
⟨all, t2b⟩\DeclareUnicodeCharacter{04FB}{\cyrgdschcrs}
⟨all, x2, t2b⟩\DeclareUnicodeCharacter{04FC}{\CYRHHK}
⟨all, x2, t2b⟩\DeclareUnicodeCharacter{04FD}{\cyrhhk}
⟨all, t2b⟩\DeclareUnicodeCharacter{04FE}{\CYRHHCRS}
⟨all, t2b⟩\DeclareUnicodeCharacter{04FF}{\cyrhhcrs}
```

```
⟨all, ts1⟩\DeclareUnicodeCharacter{0E3F}{\textbaht}
⟨all, t1⟩\DeclareUnicodeCharacter{1E02}{\.B}
⟨all, t1⟩\DeclareUnicodeCharacter{1E03}{\.b}
⟨all, t1⟩\DeclareUnicodeCharacter{1E0D}{\d d}
⟨all, t1⟩\DeclareUnicodeCharacter{1E1E}{\.F}
⟨all, t1⟩\DeclareUnicodeCharacter{1E1F}{\.f}
⟨all, t1⟩\DeclareUnicodeCharacter{1E25}{\d h}
⟨all, t1⟩\DeclareUnicodeCharacter{1E30}{\@tabacckludge'K}
⟨all, t1⟩\DeclareUnicodeCharacter{1E31}{\@tabacckludge'k}
⟨all, t1⟩\DeclareUnicodeCharacter{1E37}{\d l}
⟨all, t1⟩\DeclareUnicodeCharacter{1E8E}{\.Y}
⟨all, t1⟩\DeclareUnicodeCharacter{1E8F}{\.y}
⟨all, t1⟩\DeclareUnicodeCharacter{1E43}{\d m}
⟨all, t1⟩\DeclareUnicodeCharacter{1E45}{\.n}
⟨all, t1⟩\DeclareUnicodeCharacter{1E47}{\d n}
⟨all, t1⟩\DeclareUnicodeCharacter{1E5B}{\d r}
⟨all, t1⟩\DeclareUnicodeCharacter{1E63}{\d s}
⟨all, t1⟩\DeclareUnicodeCharacter{1E6D}{\d t}
⟨all, t1⟩\DeclareUnicodeCharacter{1E90}{\^Z}
⟨all, t1⟩\DeclareUnicodeCharacter{1E91}{\^z}
⟨all, t1⟩\DeclareUnicodeCharacter{1E9E}{\SS}
⟨all, t1⟩\DeclareUnicodeCharacter{1EF2}{\@tabacckludge`Y}
⟨all, t1⟩\DeclareUnicodeCharacter{1EF3}{\@tabacckludge`y}
⟨all, x2, t2c, t2b, t2a, t1, utf8⟩\DeclareUnicodeCharacter{200C}{\textcompwordmark}
⟨all, t1⟩\DeclareUnicodeCharacter{2010}{-}
⟨all, t1⟩\DeclareUnicodeCharacter{2011}{\mbox{-}}
```

U+2012 should be the width of a digit, endash is OK in many fonts including cm.

```
⟨all, t1⟩\DeclareUnicodeCharacter{2012}{\textendash}
⟨*all, x2, t2c, t2b, t2a, t1, ot2, ot1, ly1, lcy⟩
\DeclareUnicodeCharacter{2013}{\textendash}
\DeclareUnicodeCharacter{2014}{\textemdash}
```

U+2015 is Horizontal bar

```
⟨all, t1⟩\DeclareUnicodeCharacter{2015}{\textemdash}
⟨/all, x2, t2c, t2b, t2a, t1, ot2, ot1, ly1, lcy⟩
⟨all, ts1⟩\DeclareUnicodeCharacter{2016}{\textbardbl}
⟨*all, x2, t2c, t2b, t2a, t1, ot2, ot1, lcy⟩
\DeclareUnicodeCharacter{2018}{\textquoteleft}
\DeclareUnicodeCharacter{2019}{\textquoteright}
⟨/all, x2, t2c, t2b, t2a, t1, ot2, ot1, lcy⟩
⟨all, t1⟩\DeclareUnicodeCharacter{201A}{\quotesinglbase}
⟨*all, x2, t2c, t2b, t2a, t1, ot2, ot1, ly1, lcy⟩
\DeclareUnicodeCharacter{201C}{\textquotedblleft}
\DeclareUnicodeCharacter{201D}{\textquotedblright}
⟨/all, x2, t2c, t2b, t2a, t1, ot2, ot1, ly1, lcy⟩
⟨all, x2, t2c, t2b, t2a, t1, lcy⟩\DeclareUnicodeCharacter{201E}{\quotedblbase}
⟨all, ts1, oms, ly1⟩\DeclareUnicodeCharacter{2020}{\textdagger}
⟨all, ts1, oms, ly1⟩\DeclareUnicodeCharacter{2021}{\textdaggerdbl}
⟨all, ts1, oms, ly1⟩\DeclareUnicodeCharacter{2022}{\textbullet}
⟨all, ly1, utf8⟩\DeclareUnicodeCharacter{2026}{\textellipsis}
⟨*all, x2, ts1, t2c, t2b, t2a, t1, ly1⟩
\DeclareUnicodeCharacter{2030}{\textperthousand}
⟨/all, x2, ts1, t2c, t2b, t2a, t1, ly1⟩
```

```
⟨*all, x2, ts1, t2c, t2b, t2a, t1⟩
\DeclareUnicodeCharacter{2031}{\textpertenthousand}
⟨/all, x2, ts1, t2c, t2b, t2a, t1⟩
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{2039}{\guilsinglleft}
⟨all, t1, ly1⟩\DeclareUnicodeCharacter{203A}{\guilsinglright}
⟨all, ts1⟩\DeclareUnicodeCharacter{203B}{\textreferencemark}
⟨all, ts1⟩\DeclareUnicodeCharacter{203D}{\textinterrobang}
⟨all, ts1⟩\DeclareUnicodeCharacter{2044}{\textfractionsolidus}
⟨all, ts1⟩\DeclareUnicodeCharacter{204E}{\textasteriskcentered}
⟨all, ts1⟩\DeclareUnicodeCharacter{2052}{\textdiscount}
⟨all, ts1⟩\DeclareUnicodeCharacter{20A1}{\textcolonmonetary}
⟨all, ts1⟩\DeclareUnicodeCharacter{20A4}{\textlira}
⟨all, ts1⟩\DeclareUnicodeCharacter{20A6}{\textnaira}
⟨all, ts1⟩\DeclareUnicodeCharacter{20A9}{\textwon}
⟨all, ts1⟩\DeclareUnicodeCharacter{20AB}{\textdong}
⟨all, ts1⟩\DeclareUnicodeCharacter{20AC}{\texteuro}
⟨all, ts1⟩\DeclareUnicodeCharacter{20B1}{\textpeso}
⟨all, ts1⟩\DeclareUnicodeCharacter{2103}{\textcelsius}
⟨all, x2, ts1, t2c, t2b, t2a, ot2, lcy⟩\DeclareUnicodeCharacter{2116}{\textnumero}
⟨all, ts1⟩\DeclareUnicodeCharacter{2117}{\textcircledP}
⟨all, ts1⟩\DeclareUnicodeCharacter{211E}{\textrecipe}
⟨all, ts1⟩\DeclareUnicodeCharacter{2120}{\textservicemark}
⟨all, ts1, ly1, utf8⟩\DeclareUnicodeCharacter{2122}{\texttrademark}
⟨all, ts1⟩\DeclareUnicodeCharacter{2126}{\textohm}
⟨all, ts1⟩\DeclareUnicodeCharacter{2127}{\textmho}
⟨all, ts1⟩\DeclareUnicodeCharacter{212E}{\textestimated}
⟨all, ts1⟩\DeclareUnicodeCharacter{2190}{\textleftarrow}
⟨all, ts1⟩\DeclareUnicodeCharacter{2191}{\textuparrow}
⟨all, ts1⟩\DeclareUnicodeCharacter{2192}{\textrightarrow}
⟨all, ts1⟩\DeclareUnicodeCharacter{2193}{\textdownarrow}

⟨all, x2, ts1, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{2329}{\textlangle}
⟨all, x2, ts1, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{3008}{\textlangle}
⟨all, x2, ts1, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{232A}{\textrangle}
⟨all, x2, ts1, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{3009}{\textrangle}
⟨all, ts1⟩\DeclareUnicodeCharacter{2422}{\textblank}
⟨all, x2, t2c, t2b, t2a, t1, utf8⟩\DeclareUnicodeCharacter{2423}{\textvisiblespace}
⟨all, ts1⟩\DeclareUnicodeCharacter{25E6}{\textopenbullet}
⟨all, ts1⟩\DeclareUnicodeCharacter{25EF}{\textbigcircle}
⟨all, ts1⟩\DeclareUnicodeCharacter{266A}{\textmusicalnote}

⟨all, x2, ts1, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{27E8}{\textlangle}
⟨all, x2, ts1, t2c, t2b, t2a⟩\DeclareUnicodeCharacter{27E9}{\textrangle}
⟨all, t1⟩\DeclareUnicodeCharacter{1E20}{\@tabacckludge=G}
⟨all, t1⟩\DeclareUnicodeCharacter{1E21}{\@tabacckludge=g}
```

When doing cut-and-paste from other documents f-ligatures might show up as Unicode characters. We translate them back to individual characters so that they get accepted. If supported by the font (which is normally the case) they are then reconstructed as ligatures so they come out as desired. Otherwise they will come out as individual characters which is fine too.

```
⟨all, t1, ot1, ly1, t2a, t2b, t2c⟩\DeclareUnicodeCharacter{FB00}{ff}   % ff
⟨all, t1, ot1, ly1, t2a, t2b, t2c⟩\DeclareUnicodeCharacter{FB01}{fi}   % fi
⟨all, t1, ot1, ly1, t2a, t2b, t2c⟩\DeclareUnicodeCharacter{FB02}{fl}   % fl
⟨all, t1, ot1, ly1, t2a, t2b, t2c⟩\DeclareUnicodeCharacter{FB03}{ffi}  % ffi
```

```
927 ⟨all, t1, ot1, ly1, t2a, t2b, t2c⟩\DeclareUnicodeCharacter{FB04}{ffl}  % ffl
928 ⟨all, t1, ot1, ly1, t2a, t2b, t2c⟩\DeclareUnicodeCharacter{FB05}{st}   % st  -- this is the long s (not
929 ⟨all, t1, ot1, ly1, t2a, t2b, t2c⟩\DeclareUnicodeCharacter{FB06}{st}   % st

930 ⟨all, ts1, utf8⟩\DeclareUnicodeCharacter{FEFF}{\ifhmode\nobreak\fi}
```

3.3 Notes

The following inputs are inconsistent with the 8-bit inputenc files since they will always only produce the 'text character'. This is an area where inputenc is notoriously confused.

```
%<all,ts1,t1,ot1,ly1>\DeclareUnicodeCharacter{00A3}{\textsterling}
%<*all,x2,ts1,t2c,t2b,t2a,oms,ly1>
\DeclareUnicodeCharacter{00A7}{\textsection}
%</all,x2,ts1,t2c,t2b,t2a,oms,ly1>
%<all,ts1,utf8>\DeclareUnicodeCharacter{00A9}{\textcopyright}
%<all,ts1>\DeclareUnicodeCharacter{00B1}{\textpm}
%<all,ts1,oms,ly1>\DeclareUnicodeCharacter{00B6}{\textparagraph}
%<all,ts1,oms,ly1>\DeclareUnicodeCharacter{2020}{\textdagger}
%<all,ts1,oms,ly1>\DeclareUnicodeCharacter{2021}{\textdaggerdbl}
%<all,ly1,utf8>\DeclareUnicodeCharacter{2026}{\textellipsis}
```

The following definitions are in an encoding file but have no direct equivalent in Unicode, or they simply do not make sense in that context (or we have not yet found anything or ...:-). For example, the non-combining accent characters are certainly available somewhere but these are not equivalent to a TEX accent command.

```
\DeclareTextSymbol{\j}{OT1}{17}
\DeclareTextSymbol{\SS}{T1}{223}
\DeclareTextSymbol{\textcompwordmark}{T1}{23}

\DeclareTextAccent{\"}{OT1}{127}
\DeclareTextAccent{\'}{OT1}{19}
\DeclareTextAccent{\.}{OT1}{95}
\DeclareTextAccent{\=}{OT1}{22}
\DeclareTextAccent{\H}{OT1}{125}
\DeclareTextAccent{\^}{OT1}{94}
\DeclareTextAccent{\`}{OT1}{18}
\DeclareTextAccent{\r}{OT1}{23}
\DeclareTextAccent{\u}{OT1}{21}
\DeclareTextAccent{\v}{OT1}{20}
\DeclareTextAccent{\~}{OT1}{126}
\DeclareTextCommand{\b}{OT1}[1]
\DeclareTextCommand{\c}{OT1}[1]
\DeclareTextCommand{\d}{OT1}[1]
\DeclareTextCommand{\k}{T1}[1]
```

3.4 Mappings for OT1 glyphs

This is even more incomplete as again it covers only the single glyphs from `OT1` plus some that have been explicitly defined for this encoding. Everything that is

provided in T1, and that could be provided as composite glyphs via OT1, could and probably should be set up as well. Which leaves the many things that are not provided in T1 but can be provided in OT1 (and in T1) by composite glyphs.

Stuff not mapped (note that \j (*j*) is not equivalent to any Unicode character):

```
\DeclareTextSymbol{\j}{OT1}{17}
\DeclareTextAccent{\"}{OT1}{127}
\DeclareTextAccent{\'}{OT1}{19}
\DeclareTextAccent{\.}{OT1}{95}
\DeclareTextAccent{\=}{OT1}{22}
\DeclareTextAccent{\^}{OT1}{94}
\DeclareTextAccent{\`}{OT1}{18}
\DeclareTextAccent{\~}{OT1}{126}
\DeclareTextAccent{\H}{OT1}{125}
\DeclareTextAccent{\u}{OT1}{21}
\DeclareTextAccent{\v}{OT1}{20}
\DeclareTextAccent{\r}{OT1}{23}
\DeclareTextCommand{\b}{OT1}[1]
\DeclareTextCommand{\c}{OT1}[1]
\DeclareTextCommand{\d}{OT1}[1]
```

3.5 Mappings for OMS glyphs

Characters like \textbackslash are not mapped as they are (primarily) only in the lower 127 and the code here only sets up mappings for UTF-8 characters that are at least 2 octets long.

```
\DeclareTextSymbol{\textbackslash}{OMS}{110}        % "6E
\DeclareTextSymbol{\textbar}{OMS}{106}              % "6A
\DeclareTextSymbol{\textbraceleft}{OMS}{102}        % "66
\DeclareTextSymbol{\textbraceright}{OMS}{103}       % "67
```

But the following (and some others) might actually lurk in Unicode somewhere...

```
\DeclareTextSymbol{\textasteriskcentered}{OMS}{3}   % "03
\DeclareTextCommand{\textcircled}{OMS}
```

3.6 Mappings for TS1 glyphs

Exercise for somebody else.

3.7 Mappings for latex.ltx glyphs

There is also a collection of characters already set up in the kernel, one way or the other. Since these do not clearly relate to any particular font encoding they are mapped when the utf8 support is first set up.

Also there are a number of \providecommands in the various input encoding files which may or may not go into this part.

931 ⟨*utf8⟩
932 `% This space is intentionally empty ...`
933 ⟨/utf8⟩

3.8 Old `utf8.def` file as a temp fix for pTEX and friends

```
⟨*utf8-2018⟩
\ProvidesFile{utf8.def}
   [2018/10/05 v1.2f UTF-8 support for inputenc]
\makeatletter
\catcode`\ \saved@space@catcode
\long\def\UTFviii@two@octets#1#2{\expandafter
    \UTFviii@defined\csname u8:#1\string#2\endcsname}
\long\def\UTFviii@three@octets#1#2#3{\expandafter
    \UTFviii@defined\csname u8:#1\string#2\string#3\endcsname}
\long\def\UTFviii@four@octets#1#2#3#4{\expandafter
    \UTFviii@defined\csname u8:#1\string#2\string#3\string#4\endcsname}
\def\UTFviii@defined#1{%
  \ifx#1\relax
     \if\relax\expandafter\UTFviii@checkseq\string#1\relax\relax
      \UTFviii@undefined@err{#1}%
     \else
      \PackageError{inputenc}{Invalid UTF-8 byte sequence}%
                             \UTFviii@invalid@help
     \fi
  \else\expandafter
    #1%
  \fi
}
\def\UTFviii@invalid@err#1{%
 \PackageError{inputenc}{Invalid UTF-8 byte "\UTFviii@hexnumber{`#1}}%
                        \UTFviii@invalid@help}
\def\UTFviii@invalid@help{%
   The document does not appear to be in UTF-8 encoding.\MessageBreak
   Try adding \noexpand\UseRawInputEncoding as the first line of the file\MessageBreak
   or specify an encoding such as \noexpand\usepackage[latin1]{inputenc}\MessageBreak
   in the document preamble.\MessageBreak
   Alternatively, save the file in UTF-8 using your editor or another tool}
\def\UTFviii@undefined@err#1{%
  \PackageError{inputenc}{Unicode character \expandafter
                          \UTFviii@splitcsname\string#1\relax
                          \MessageBreak
                          not set up for use with LaTeX}%
                         {You may provide a definition with\MessageBreak
                          \noexpand\DeclareUnicodeCharacter}%
   }
\def\UTFviii@checkseq#1:#2#3{%
 \ifnum`#2<"80 %
   \ifx\relax#3\else1\fi
 \else
   \ifnum`#2<"C0 %
     1 %
   \else
     \expandafter\expandafter\expandafter\UTFviii@check@continue
     \expandafter\expandafter\expandafter#3%
   \fi
  \fi}
\def\UTFviii@check@continue#1{%
```

```
  \ifx\relax#1%
  \else
  \ifnum`#1<"80 1\else\ifnum`#1>"BF 1\fi\fi
  \expandafter\UTFviii@check@continue
  \fi
}
\begingroup
\catcode`\~13
\catcode`\"12
\def\UTFviii@loop{%
  \uccode`\~\count@
  \uppercase\expandafter{\UTFviii@tmp}%
  \advance\count@\@ne
  \ifnum\count@<\@tempcnta
  \expandafter\UTFviii@loop
  \fi}
    \def\UTFviii@tmp{\xdef~{\noexpand\UTFviii@undefined@err{:\string~}}}
    \count@"1
    \@tempcnta9
\UTFviii@loop
    \count@11
    \@tempcnta12
\UTFviii@loop
    \count@14
    \@tempcnta32
\UTFviii@loop
    \count@"80
    \@tempcnta"C2
    \def\UTFviii@tmp{\xdef~{\noexpand\UTFviii@invalid@err\string~}}
\UTFviii@loop
    \count@"C2
    \@tempcnta"E0
    \def\UTFviii@tmp{\xdef~{\noexpand\UTFviii@two@octets\string~}}
\UTFviii@loop
    \count@"E0
    \@tempcnta"F0
    \def\UTFviii@tmp{\xdef~{\noexpand\UTFviii@three@octets\string~}}
\UTFviii@loop
    \count@"F0
    \@tempcnta"F5
    \def\UTFviii@tmp{\xdef~{\noexpand\UTFviii@four@octets\string~}}
\UTFviii@loop
    \count@"F5
    \@tempcnta"100
    \def\UTFviii@tmp{\xdef~{\noexpand\UTFviii@invalid@err\string~}}
\UTFviii@loop
\endgroup
\@inpenc@test
\ifx\@begindocumenthook\@undefined
  \makeatother
  \endinput \fi
\begingroup
\catcode`\"=12
\catcode`\<=12
```

```
\catcode`\.=12
\catcode`\,=12
\catcode`\;=12
\catcode`\!=12
\catcode`\~=13
\gdef\DeclareUnicodeCharacter#1#2{%
  \count@"#1\relax
  \wlog{ \space\space defining Unicode char U+#1 (decimal \the\count@)}%
  \begingroup
    \parse@XML@charref
    \def\UTFviii@two@octets##1##2{\csname u8:##1\string##2\endcsname}%
    \def\UTFviii@three@octets##1##2##3{\csname u8:##1%
                                       \string##2\string##3\endcsname}%
    \def\UTFviii@four@octets##1##2##3##4{\csname u8:##1%
                             \string##2\string##3\string##4\endcsname}%
    \expandafter\expandafter\expandafter
    \expandafter\expandafter\expandafter
    \expandafter
     \gdef\UTFviii@tmp{\IeC{#2}}%
   \endgroup
}
\gdef\parse@XML@charref{%
  \ifnum\count@<"A0\relax
    \ifnum\catcode\count@=13
      \uccode`\~=\count@\uppercase{\def\UTFviii@tmp{\@empty\@empty~}}%
    \else
      \PackageError{inputenc}{Cannot define non-active Unicode char value < 00A0}%
                    \@eha
      \def\UTFviii@tmp{\UTFviii@tmp}%
    \fi
  \else\ifnum\count@<"800\relax
     \parse@UTFviii@a,%
     \parse@UTFviii@b C\UTFviii@two@octets.,%
  \else\ifnum\count@<"10000\relax
     \parse@UTFviii@a;%
     \parse@UTFviii@a,%
     \parse@UTFviii@b E\UTFviii@three@octets.{,;}%
   \else
     \ifnum\count@>"10FFFF\relax
           \PackageError{inputenc}%
                {\UTFviii@hexnumber\count@\space too large for Unicode}%
                {Values between 0 and 10FFFF are permitted}%
      \fi
     \parse@UTFviii@a;%
     \parse@UTFviii@a,%
     \parse@UTFviii@a!%
     \parse@UTFviii@b F\UTFviii@four@octets.{!,;}%
    \fi
    \fi
  \fi
}
\gdef\parse@UTFviii@a#1{%
     \@tempcnta\count@
     \divide\count@ 64
```

```
      \@tempcntb\count@
      \multiply\count@ 64
      \advance\@tempcnta-\count@
      \advance\@tempcnta 128
      \uccode`#1\@tempcnta
      \count@\@tempcntb}
\gdef\parse@UTFviii@b#1#2#3#4{%
      \advance\count@ "#10\relax
      \uccode`#3\count@
      \uppercase{\gdef\UTFviii@tmp{#2#3#4}}}
\ifx\numexpr\@undefined
\gdef\decode@UTFviii#1{0}
\else
\gdef\decode@UTFviii#1\relax{%
  \expandafter\UTFviii@cleanup
    \the\numexpr\dec@de@UTFviii#1\relax)))))\@empty}
\gdef\UTFviii@cleanup#1)#2\@empty{#1}
\gdef\dec@de@UTFviii#1{%
\ifx\relax#1%
\else
  \ifnum`#1>"EF
    ((((`#1-"F0)%
  \else
    \ifnum`#1>"DF
      (((`#1-"E0)%
    \else
      \ifnum`#1>"BF
        ((`#1-"C0)%
      \else
        \ifnum`#1>"7F
        )*64+(`#1-"80)%
        \else
        +`#1 %
        \fi
      \fi
    \fi
  \fi
  \expandafter\dec@de@UTFviii
\fi}
\fi
\ifx\numexpr\@undefined
\global\let\UTFviii@hexnumber\@firstofone
\global\UTFviii@hexdigit\hexnumber@
\else
\gdef\UTFviii@hexnumber#1{%
\ifnum#1>15 %
\expandafter\UTFviii@hexnumber\expandafter{\the\numexpr(#1-8)/16\relax}%
\fi
\UTFviii@hexdigit{\numexpr#1\ifnum#1>0-((#1-8)/16)*16\fi\relax}%
}
\gdef\UTFviii@hexdigit#1{\ifcase\numexpr#1\relax
  0\or1\or2\or3\or4\or5\or6\or7\or8\or9\or
  A\or B\or C\or D\or E\or F\fi}
\fi
```

```
\gdef\UTFviii@hexcodepoint#1{U+%
 \ifnum#1<16 0\fi
 \ifnum#1<256 0\fi
 \ifnum#1<4096 0\fi
 \UTFviii@hexnumber{#1}%
}%
\gdef\UTFviii@splitcsname#1:#2\relax{%
#2 (\expandafter\UTFviii@hexcodepoint\expandafter{%
                     \the\numexpr\decode@UTFviii#2\relax})%
}
\endgroup
\@onlypreamble\DeclareUnicodeCharacter
\@onlypreamble\parse@XML@charref
\@onlypreamble\parse@UTFviii@a
\@onlypreamble\parse@UTFviii@b
\begingroup
  \def\cdp@elt#1#2#3#4{%
    \wlog{Now handling font encoding #1 ...}%
    \lowercase{%
        \InputIfFileExists{#1enc.dfu}}%
           {\wlog{... processing UTF-8 mapping file for font %
                     encoding #1}%
            \catcode`\ 9\relax}%
          {\wlog{... no UTF-8 mapping file for font encoding #1}}%
  }
  \cdp@list
\endgroup
\def\DeclareFontEncoding@#1#2#3{%
  \expandafter
  \ifx\csname T@#1\endcsname\relax
    \def\cdp@elt{\noexpand\cdp@elt}%
    \xdef\cdp@list{\cdp@list\cdp@elt{#1}%
                   {\default@family}{\default@series}%
                   {\default@shape}}%
    \expandafter\let\csname#1-cmd\endcsname\@changed@cmd
    \begingroup
      \wlog{Now handling font encoding #1 ...}%
      \lowercase{%
        \InputIfFileExists{#1enc.dfu}}%
           {\wlog{... processing UTF-8 mapping file for font %
                     encoding #1}}%
           {\wlog{... no UTF-8 mapping file for font encoding #1}}%
    \endgroup
  \else
     \@font@info{Redeclaring font encoding #1}%
  \fi
  \global\@namedef{T@#1}{#2}%
  \global\@namedef{M@#1}{\default@M#3}%
  \xdef\LastDeclaredEncoding{#1}%
  }
\DeclareUnicodeCharacter{00A9}{\textcopyright}
\DeclareUnicodeCharacter{00AA}{\textordfeminine}
\DeclareUnicodeCharacter{00AE}{\textregistered}
\DeclareUnicodeCharacter{00BA}{\textordmasculine}
```

```
\DeclareUnicodeCharacter{02C6}{\textasciicircum}
\DeclareUnicodeCharacter{02DC}{\textasciitilde}
\DeclareUnicodeCharacter{200C}{\textcompwordmark}
\DeclareUnicodeCharacter{2026}{\textellipsis}
\DeclareUnicodeCharacter{2122}{\texttrademark}
\DeclareUnicodeCharacter{2423}{\textvisiblespace}
\DeclareUnicodeCharacter{FEFF}{\ifhmode\nobreak\fi}
\endinput
</utf8-2018>
```

4 A test document

Here is a very small test document which may or may not survive if the current document is transferred from one place to the other.

```
⟨*test⟩
\documentclass{article}

\usepackage[latin1,utf8]{inputenc}
\usepackage[T1]{fontenc}
\usepackage{trace}

\scrollmode  % to run past the error below

\begin{document}

 German umlauts in UTF-8: ^^c3^^a4^^c3^^b6^^c3^^bc  %%% äöü

\inputencoding{latin1}  % switch to latin1

 German umlauts in UTF-8 but read by latin1 (and will produce one
 error since \verb=\textcurrency= is not provided):
 ^^c3^^a4^^c3^^b6^^c3^^bc

\inputencoding{utf8}    % switch back to utf8

 German umlauts in UTF-8: ^^c3^^a4^^c3^^b6^^c3^^bc


Some codes that should produce errors as nothing is set up
for them: ^^c3F ^^e1^^a4^^b6

And some that are not legal utf8 sequences: ^^c3X ^^e1XY

\showoutput
\tracingstats=2
\stop
⟨/test⟩
```