

fmtcount.sty v1.1: Displaying the Values of L^AT_EX Counters

Nicola L.C. Talbot

26th May 2007

Contents

1	Introduction	2
2	Installation	2
3	Available Commands	2
4	Package Options	4
5	Multilingual Support	4
6	Configuration File <code>fmtcount.cfg</code>	5
7	LaTeX2HTML style	5
8	Acknowledgements	6
9	Troubleshooting	6
10	Contact Details	6
11	The Code	6
11.1	<code>fc-british.def</code>	6
11.2	<code>fc-english.def</code>	6
11.3	<code>fc-french.def</code>	16
11.4	<code>fc-german.def</code>	36
11.5	<code>fc-portuges.def</code>	45
11.6	<code>fc-spanish.def</code>	58
11.7	<code>fc-UKenglish.def</code>	72
11.8	<code>fc-USEnglish.def</code>	73
11.9	<code>fmtcount.sty</code>	73
11.9.1	Multilanguage Definitions	86

1 Introduction

The `fmtcount` package provides commands to display the values of \LaTeX counters in a variety of formats. It also provides equivalent commands for actual numbers rather than counter names. Limited multilingual support is available.

2 Installation

This package is distributed with the files `fmtcount.dtx` and `fmtcount.ins`. To extract the code do:

```
latex fmtcount.ins
```

This will create the files `fmtcount.sty` and `fmtcount.perl`, along with several `.def` files. Place `fmtcount.sty` and the `.def` files somewhere where \LaTeX will find them (e.g. `texmf/tex/latex/fmtcount/`) and place `fmtcount.perl` somewhere where \LaTeX2HTML will find it (e.g. `latex2html/styles`). Remember to refresh the \TeX database (using `texhash` under Linux, for other operating systems check the manual.)

3 Available Commands

The commands can be divided into two categories: those that take the name of a counter as the argument, and those that take a number as the argument.

`\ordinal` The macro `\ordinal{<counter>}` will print the value of a \LaTeX counter
`\fmtord` `<counter>` as an ordinal, where the macro `\fmtord{<text>}` is used to format the
st,nd,rd,th bit. By default the ordinal is formatted as a superscript, if the package
option `level` is used, it is level with the text. For example, if the current section
is 3, then `\ordinal{section}` will produce the output: 3rd.

Note: the `memoir` class also defines a command called `\ordinal` which takes a number as an argument instead of a counter. In order to overcome this incompatibility, if you want to use the `fmtcount` package with the `memoir` class you should use `\FCordinal` to access `fmtcount`'s version of `\ordinal`, and use `\ordinal` to use `memoir`'s version of that command.

`\ordinalnum` The macro `\ordinalnum` is like `\ordinal` but takes an actual number rather
than a counter as the argument. For example: `\ordinalnum{3}` will produce: 3rd.

`\numberstring` The macro `\numberstring{<counter>}` will print the value of `<counter>`
`\Numberstring` as text. E.g. `\numberstring{section}` will produce: three. The macro
`\Numberstring{<counter>}` does the same as `\numberstring`, but with initial let-
ters in uppercase. For example, `\Numberstring{section}` will produce: Three.
`\NUMBERstring` The macro `\NUMBERstring{<counter>}` does the same as `\numberstring`, but con-
verted to upper case. Note that `\MakeUppercase{\NUMBERstring{<counter>}}`
doesn't work, due to the way that `\MakeUppercase` expands its argument¹.

`\numberstringnum` The macros `\numberstringnum`, `\Numberstringnum` and `\NUMBERstringnum`
`\Numberstringnum` work like `\numberstring`, `\Numberstring` and `\NUMBERstring`, respectively, but
`\NUMBERstringnum` take an actual number rather than a counter as the argument. For example:
`\Numberstringnum{105}` will produce: One Hundred and Five.

`\ordinalstring` The macro `\ordinalstring{<counter>}` will print the value of `<counter>`

¹See all the various postings to `comp.text.tex` about `\MakeUppercase`

as a textual ordinal. E.g. `\ordinalstring{section}` will produce: third.

`\Ordinalstring` The macro `\Ordinalstring{<counter>}` does the same as `\ordinalstring`, but with initial letters in uppercase. For example, `\Ordinalstring{section}` will produce: Third. The macro `\ORDINALstring{<counter>}` does the same as `\ordinalstring`, but with all words in upper case (see previous note about `\MakeUppercase`).

`\ordinalstringnum` The macros `\ordinalstringnum`, `\Ordinalstringnum` and `\ORDINALstringnum` work like `\ordinalstring`, `\Ordinalstring` and `\ORDINALstring`, respectively, but take an actual number rather than a counter as the argument. For example, `\ordinalstringnum{3}` will produce: third.

As from version 1.09, textual representations can be stored for later use. This overcomes the problems encountered when you attempt to use one of the above commands in `\edef`.

Each of the following commands takes a label as the first argument, the other arguments are as the analogous commands above. These commands do not display anything, but store the textual representation. This can later be retrieved using `\FMCuse{<label>}`. Note: with `\storeordinal` and `\storeordinalnum`, the only bit that doesn't get expanded is `\fmtord`. So, for example, `\storeordinalnum{mylabel}{3}` will be stored as `3\relax \fmtord{rd}`.

`\storeordinal` `\storeordinal{<label>}{<counter>}[<gender>]`
`\storeordinalstring` `\storeordinalstring{<label>}{<counter>}[<gender>]`
`\storeOrdinalstring` `\storeOrdinalstring{<label>}{<counter>}[<gender>]`
`\storeORDINALstring` `\storeORDINALstring{<label>}{<counter>}[<gender>]`
`\storenumberstring` `\storenumberstring{<label>}{<counter>}[<gender>]`
`\storeNumberstring` `\storeNumberstring{<label>}{<counter>}[<gender>]`
`\storeNUMBERstring` `\storeNUMBERstring{<label>}{<counter>}[<gender>]`
`\storeordinalnum` `\storeordinalnum{<label>}{<number>}[<gender>]`
`\storeordinalstringnum` `\storeordinalstring{<label>}{<number>}[<gender>]`
`\storeOrdinalstringnum` `\storeOrdinalstringnum{<label>}{<number>}[<gender>]`
`\storeORDINALstringnum` `\storeORDINALstringnum{<label>}{<number>}[<gender>]`
`\storenumberstringnum` `\storenumberstring{<label>}{<number>}[<gender>]`
`\storeNumberstringnum` `\storeNumberstring{<label>}{<number>}[<gender>]`
`\storeNUMBERstringnum` `\storeNUMBERstring{<label>}{<number>}[<gender>]`

`\binary` The macro `\binary{<counter>}` will print the value of `<counter>` as a binary number. E.g. `\binary{section}` will produce: 11. The declaration `\padzeroes[<n>]` will ensure numbers are written to `<n>` digits, padding with zeroes if necessary. E.g. `\padzeroes[8]\binary{section}` will produce: 00000011. The default value for `<n>` is 17.

`\binarynum` The macro `\binarynum` is like `\binary` but takes an actual number rather than a counter as the argument. For example: `\binarynum{5}` will produce: 101.

`\octal` The macro `\octal{<counter>}` will print the value of `<counter>` as an octal number. For example, if you have a counter called, say `mycounter`, and you set the value to 125, then `\octal{mycounter}` will produce: 177. Again, the number will be padded with zeroes if necessary, depending on whether `\padzeroes` has been used.

`\octalnum` The macro `\octalnum` is like `\octal` but takes an actual number rather than a counter as the argument. For example: `\octalnum{125}` will produce: 177.

`\hexadecimal` The macro `\hexadecimal{<counter>}` will print the value of `<counter>` as a hexadecimal number. Going back to the previous example, `\hexadecimal{mycounter}`

<code>\Hexadecimal</code>	will produce: 7d. Again, the number will be padded with zeroes if necessary, depending on whether <code>\padzeroes</code> has been used. <code>\Hexadecimal{<counter>}</code> does the same thing, but uses uppercase characters, e.g. <code>\Hexadecimal{mycounter}</code> will produce: 7D.
<code>\hexadecimalnum</code>	The macros <code>\hexadecimalnum</code> and <code>\Hexadecimalnum</code> are like <code>\hexadecimal</code> and <code>\Hexadecimal</code> but take an actual number rather than a counter as the argument. For example: <code>\hexadecimalnum{125}</code> will produce: 7d, and <code>\Hexadecimalnum{125}</code> will produce: 7D.
<code>\Hexadecimalnum</code>	
<code>\decimal</code>	The macro <code>\decimal{<counter>}</code> is similar to <code>\arabic</code> but the number can be padded with zeroes depending on whether <code>\padzeroes</code> has been used. For example: <code>\padzeroes[8]\decimal{section}</code> will produce: 00000005.
<code>\decimalnum</code>	The macro <code>\decimalnum</code> is like <code>\decimal</code> but takes an actual number rather than a counter as the argument. For example: <code>\padzeroes[8]\decimalnum{5}</code> will produce: 00000005.
<code>\aaalph</code>	The macro <code>\aaalph{<counter>}</code> will print the value of <code><counter></code> as: a b ... z aa bb ... zz etc. For example, <code>\aaalpha{mycounter}</code> will produce: uuuuu if <code>mycounter</code> is set to 125. <code>\AAAalph{<counter>}</code> does the same thing, but uses uppercase characters, e.g. <code>\AAAalph{mycounter}</code> will produce: UUUUU.
<code>\AAAalph</code>	
<code>\aaalphnum</code>	The macros <code>\aaalphnum</code> and <code>\AAAalphnum</code> are like <code>\aaalph</code> and <code>\AAAalph</code> but take an actual number rather than a counter as the argument. For example: <code>\aaalphnum{125}</code> will produce: uuuuu, and <code>\AAAalphnum{125}</code> will produce: UU-UUU.
<code>\AAAalphnum</code>	
<code>\abalph</code>	The macro <code>\abalph{<counter>}</code> will print the value of <code><counter></code> as: a b ... z aa ab ... az etc. For example, <code>\abalpha{mycounter}</code> will produce: du if <code>mycounter</code> is set to 125. <code>\ABAlph{<counter>}</code> does the same thing, but uses uppercase characters, e.g. <code>\ABAlph{mycounter}</code> will produce: DU.
<code>\ABAlph</code>	
<code>\abalphnum</code>	The macros <code>\abalphnum</code> and <code>\ABAlphnum</code> are like <code>\abalph</code> and <code>\ABAlph</code> but take an actual number rather than a counter as the argument. For example: <code>\abalphnum{125}</code> will produce: du, and <code>\ABAlphnum{125}</code> will produce: DU.
<code>\ABAlphnum</code>	

4 Package Options

The following options can be passed to this package:

raise make ordinal st,nd,rd,th appear as superscript
level make ordinal st,nd,rd,th appear level with rest of text

These can also be set using the command:

`\fmtcountsetoptions` `\fmtcountsetoptions{fmtord=<type>}`
where `<type>` is either `level` or `raise`.

5 Multilingual Support

Version 1.02 of the `fmtcount` package now has limited multilingual support. The following languages are implemented: English, Spanish, Portuguese, French, French (Swiss) and French (Belgian). German support was added in version 1.1².

The package checks to see if the command `\date<language>` is defined³, and will load the code for those languages. The commands `\ordinal`, `\ordinalstring`

²Thanks to K. H. Fricke for supplying the information

³this will be true if you have loaded `babel`

and `\numberstring` (and their variants) will then be formatted in the currently selected language.

If the French language is selected, the French (France) version will be used by default (e.g. *soixante-dix* for 70). To select the Swiss or Belgian variants (e.g. *septente* for 70) use: `\fmtcountsetoptions{french=<dialect>}` where *<dialect>* is either *swiss* or *belgian*. You can also use this command to change the action of `\ordinal`. `\fmtcountsetoptions{abbrv=true}` to produce ordinals of the form 2^e or `\fmtcountsetoptions{abbrv=false}` to produce ordinals of the form 2^{eme} (default).

The `french` and `abbrv` settings only have an effect if the French language has been defined.

The male gender for all languages is used by default, however the feminine or neuter forms can be obtained by passing `f` or `n` as an optional argument to `\ordinal`, `\ordinalnum` etc. For example: `\numberstring{section}[f]`. Note that the optional argument comes *after* the compulsory argument. If a gender is not defined in a given language, the masculine version will be used instead.

Let me know if you find any spelling mistakes (has been known to happen in English, let alone other languages I'm not so familiar with.) If you want to add support for another language, you will need to let me know how to form the numbers and ordinals from 0 to 99999 in that language for each gender.

6 Configuration File `fmtcount.cfg`

You can save your preferred default settings to a file called `fmtcount.cfg`, and place it on the \TeX path. These settings will then be loaded by the `fmtcount` package.

Note that if you are using the `datetime` package, the `datetime.cfg` configuration file will override the `fmtcount.cfg` configuration file. For example, if `datetime.cfg` has the line:

```
\renewcommand{\fmtord}[1]{\textsuperscript{\underline{#1}}}
```

and if `fmtcount.cfg` has the line:

```
\fmtcountsetoptions{fmtord=level}
```

then the former definition of `\fmtord` will take precedence.

7 LaTeX2HTML style

The LaTeX2HTML style file `fmtcount.perl` is provided. The following limitations apply:

- `\padzeroes` only has an effect in the preamble.
- The configuration file `fmtcount.cfg` is currently ignored. (This is because I can't work out the correct code to do this. If you know how to do this, please let me know.) You can however do:

```
\usepackage{fmtcount}
\html{\input{fmtcount.cfg}}
```

This, I agree, is an unpleasant cludge.

8 Acknowledgements

I would like to thank my mother for the French and Portuguese support and my Spanish dictionary for the Spanish support. Thank you to K. H. Fricke for providing me with the German translations.

9 Troubleshooting

There is a FAQ available at: <http://theoval.cmp.uea.ac.uk/~nlct/latex/packages/faq/>.

10 Contact Details

Dr Nicola Talbot
School of Computing Sciences
University of East Anglia
Norwich. NR4 7TJ.
United Kingdom.
<http://theoval.cmp.uea.ac.uk/~nlct/>

11 The Code

11.1 fc-british.def

British definitions

```
1 \ProvidesFile{fc-british}[2007/06/14]
```

Check that fc-english.def has been loaded

```
2 \@ifundefined{@OrdinalMenglish}{\input{fc-english.def}}{}
```

These are all just synonyms for the commands provided by fc-english.def.

```
3 \let\@OrdinalMbritish\@OrdinalMenglish
```

```
4 \let\@OrdinalFbritish\@OrdinalMenglish
```

```
5 \let\@OrdinalNbritish\@OrdinalMenglish
```

```
6 \let\@numberstringMbritish\@numberstringMenglish
```

```
7 \let\@numberstringFbritish\@numberstringMenglish
```

```
8 \let\@numberstringNbritish\@numberstringMenglish
```

```
9 \let\@NumberstringMbritish\@NumberstringMenglish
```

```
10 \let\@NumberstringFbritish\@NumberstringMenglish
```

```
11 \let\@NumberstringNbritish\@NumberstringMenglish
```

```
12 \let\@ordinalstringMbritish\@ordinalstringMenglish
```

```
13 \let\@ordinalstringFbritish\@ordinalstringMenglish
```

```
14 \let\@ordinalstringNbritish\@ordinalstringMenglish
```

```
15 \let\@OrdinalstringMbritish\@OrdinalstringMenglish
```

```
16 \let\@OrdinalstringFbritish\@OrdinalstringMenglish
```

```
17 \let\@OrdinalstringNbritish\@OrdinalstringMenglish
```

11.2 fc-english.def

English definitions

```
18 \ProvidesFile{fc-english}[2007/05/26]
```

Define macro that converts a number or count register (first argument) to an ordinal, and stores the result in the second argument, which should be a control sequence.

```

19 \newcommand*{\@ordinalMenglish}[2]{%
20 \def\@fc@ord{%
21 \@orgargctr=#1\relax
22 \@ordinalctr=#1%
23 \@modulo{\@ordinalctr}{100}%
24 \ifnum\@ordinalctr=11\relax
25   \def\@fc@ord{th}%
26 \else
27   \ifnum\@ordinalctr=12\relax
28     \def\@fc@ord{th}%
29   \else
30     \ifnum\@ordinalctr=13\relax
31       \def\@fc@ord{th}%
32     \else
33       \@modulo{\@ordinalctr}{10}%
34       \ifcase\@ordinalctr
35         \def\@fc@ord{th}%      case 0
36         \or \def\@fc@ord{st}%  case 1
37         \or \def\@fc@ord{nd}%  case 2
38         \or \def\@fc@ord{rd}%  case 3
39       \else
40         \def\@fc@ord{th}%      default case
41       \fi
42     \fi
43   \fi
44 \fi
45 \edef#2{\number#1\relax\noexpand\fmtord{\@fc@ord}}%
46 }
```

There is no gender difference in English, so make feminine and neuter the same as the masculine.

```

47 \let\@ordinalFenglish=\@ordinalMenglish
48 \let\@ordinalNenglish=\@ordinalMenglish
```

Define the macro that prints the value of a \TeX count register as text. To make it easier, break it up into units, teens and tens. First, the units: the argument should be between 0 and 9 inclusive.

```

49 \newcommand*{\@unitstringenglish}[1]{%
50 \ifcase#1\relax
51 zero%
52 \or one%
53 \or two%
54 \or three%
55 \or four%
56 \or five%
57 \or six%
58 \or seven%
59 \or eight%
60 \or nine%
61 \fi
62 }
```

Next the tens, again the argument should be between 0 and 9 inclusive.

```
63 \newcommand*\@@tenstringenglish}[1]{%
64 \ifcase#1\relax
65 \or ten%
66 \or twenty%
67 \or thirty%
68 \or forty%
69 \or fifty%
70 \or sixty%
71 \or seventy%
72 \or eighty%
73 \or ninety%
74 \fi
75 }
```

Finally the teens, again the argument should be between 0 and 9 inclusive.

```
76 \newcommand*\@@teenstringenglish}[1]{%
77 \ifcase#1\relax
78 ten%
79 \or eleven%
80 \or twelve%
81 \or thirteen%
82 \or fourteen%
83 \or fifteen%
84 \or sixteen%
85 \or seventeen%
86 \or eighteen%
87 \or nineteen%
88 \fi
89 }
```

As above, but with the initial letter in uppercase. The units:

```
90 \newcommand*\@@Unitstringenglish}[1]{%
91 \ifcase#1\relax
92 Zero%
93 \or One%
94 \or Two%
95 \or Three%
96 \or Four%
97 \or Five%
98 \or Six%
99 \or Seven%
100 \or Eight%
101 \or Nine%
102 \fi
103 }
```

The tens:

```
104 \newcommand*\@@Tenstringenglish}[1]{%
105 \ifcase#1\relax
106 \or Ten%
107 \or Twenty%
108 \or Thirty%
109 \or Forty%
110 \or Fifty%
```



```

111 \or Sixty%
112 \or Seventy%
113 \or Eighty%
114 \or Ninety%
115 \fi
116 }

```

The teens:

```

117 \newcommand*{\@@Teenstringenglish}[1]{%
118 \ifcase#1\relax
119 Ten%
120 \or Eleven%
121 \or Twelve%
122 \or Thirteen%
123 \or Fourteen%
124 \or Fifteen%
125 \or Sixteen%
126 \or Seventeen%
127 \or Eighteen%
128 \or Nineteen%
129 \fi
130 }

```

This has changed in version 1.09, so that it now stores the result in the second argument, but doesn't display anything. Since it only affects internal macros, it shouldn't affect documents created with older versions. (These internal macros are not meant for use in documents.)

```

131 \newcommand*{\@@numberstringenglish}[2]{%
132 \ifnum#1>99999
133 \PackageError{fmtcount}{Out of range}%
134 {This macro only works for values less than 100000}%
135 \else
136 \ifnum#1<0
137 \PackageError{fmtcount}{Negative numbers not permitted}%
138 {This macro does not work for negative numbers, however
139 you can try typing "minus" first, and then pass the modulus of
140 this number}%
141 \fi
142 \fi
143 \def#2{}%
144 \@strctr=#1\relax \divide\@strctr by 1000\relax
145 \ifnum\@strctr>9
146 % #1 is greater or equal to 10000
147 \divide\@strctr by 10
148 \ifnum\@strctr>1\relax
149 \let\@@fc@numstr#2\relax
150 \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
151 \@strctr=#1 \divide\@strctr by 1000\relax
152 \@modulo{\@strctr}{10}%
153 \ifnum\@strctr>0\relax
154 \let\@@fc@numstr#2\relax
155 \edef#2{\@@fc@numstr-\@unitstring{\@strctr}}%
156 \fi
157 \else
158 \@strctr=#1\relax

```

```

159 \divide\@strctr by 1000\relax
160 \@modulo{\@strctr}{10}%
161 \let\@@fc@numstr#2\relax
162 \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
163 \fi
164 \let\@@fc@numstr#2\relax
165 \edef#2{\@@fc@numstr\ \@thousand}%
166 \else
167 \ifnum\@strctr>0\relax
168 \let\@@fc@numstr#2\relax
169 \edef#2{\@@fc@numstr\@unitstring{\@strctr}\ \@thousand}%
170 \fi
171 \fi
172 \@strctr=#1\relax \@modulo{\@strctr}{1000}%
173 \divide\@strctr by 100
174 \ifnum\@strctr>0\relax
175 \ifnum#1>1000\relax
176 \let\@@fc@numstr#2\relax
177 \edef#2{\@@fc@numstr\ }%
178 \fi
179 \let\@@fc@numstr#2\relax
180 \edef#2{\@@fc@numstr\@unitstring{\@strctr}\ \@hundred}%
181 \fi
182 \@strctr=#1\relax \@modulo{\@strctr}{100}%
183 \ifnum#1>100\relax
184 \ifnum\@strctr>0\relax
185 \let\@@fc@numstr#2\relax
186 \edef#2{\@@fc@numstr\ \@andname\ }%
187 \fi
188 \fi
189 \ifnum\@strctr>19\relax
190 \divide\@strctr by 10\relax
191 \let\@@fc@numstr#2\relax
192 \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
193 \@strctr=#1\relax \@modulo{\@strctr}{10}%
194 \ifnum\@strctr>0\relax
195 \let\@@fc@numstr#2\relax
196 \edef#2{\@@fc@numstr-\@unitstring{\@strctr}}%
197 \fi
198 \else
199 \ifnum\@strctr<10\relax
200 \ifnum\@strctr=0\relax
201 \ifnum#1<100\relax
202 \let\@@fc@numstr#2\relax
203 \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
204 \fi
205 \else
206 \let\@@fc@numstr#2\relax
207 \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
208 \fi
209 \else
210 \@modulo{\@strctr}{10}%
211 \let\@@fc@numstr#2\relax
212 \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%

```

```

213 \fi
214 \fi
215 }

```

All lower case version, the second argument must be a control sequence.

```

216 \DeclareRobustCommand{\@numberstringMenglish}[2]{%
217 \let\@unitstring=\@unitstringenglish
218 \let\@teenstring=\@teenstringenglish
219 \let\@tenstring=\@tenstringenglish
220 \def\@hundred{hundred}\def\@thousand{thousand}%
221 \def\@andname{and}%
222 \@numberstringenglish{#1}{#2}%
223 }

```

There is no gender in English, so make feminine and neuter the same as the masculine.

```

224 \let\@numberstringFenglish=\@numberstringMenglish
225 \let\@numberstringNenglish=\@numberstringMenglish

```

This version makes the first letter of each word an uppercase character (except “and”). The second argument must be a control sequence.

```

226 \newcommand*{\@NumberstringMenglish}[2]{%
227 \let\@unitstring=\@Unitstringenglish
228 \let\@teenstring=\@Teenstringenglish
229 \let\@tenstring=\@Tenstringenglish
230 \def\@hundred{Hundred}\def\@thousand{Thousand}%
231 \def\@andname{and}%
232 \@numberstringenglish{#1}{#2}}

```

There is no gender in English, so make feminine and neuter the same as the masculine.

```

233 \let\@NumberstringFenglish=\@NumberstringMenglish
234 \let\@NumberstringNenglish=\@NumberstringMenglish

```

Define a macro that produces an ordinal as a string. Again, break it up into units, teens and tens. First the units:

```

235 \newcommand*{\@unitthstringenglish}[1]{%
236 \ifcase#1\relax
237 zeroth%
238 \or first%
239 \or second%
240 \or third%
241 \or fourth%
242 \or fifth%
243 \or sixth%
244 \or seventh%
245 \or eighth%
246 \or ninth%
247 \fi
248 }

```

Next the tens:

```

249 \newcommand*{\@tenthstringenglish}[1]{%
250 \ifcase#1\relax
251 \or tenth%
252 \or twentieth%

```

```

253 \or thirtieth%
254 \or fortieth%
255 \or fiftieth%
256 \or sixtieth%
257 \or seventieth%
258 \or eightieth%
259 \or ninetieth%
260 \fi
261 }
262 % \end{macrocode}
263 % The teens:
264 % \begin{macrocode}
265 \newcommand*{\@@teenthstringenglish}[1]{%
266 \ifcase#1\relax
267 tenth%
268 \or eleventh%
269 \or twelfth%
270 \or thirteenth%
271 \or fourteenth%
272 \or fifteenth%
273 \or sixteenth%
274 \or seventeenth%
275 \or eighteenth%
276 \or nineteenth%
277 \fi
278 }
279 % \end{macrocode}
280 % As before, but with the first letter in upper case. The units:
281 % \begin{macrocode}
282 \newcommand*{\@@Unitthstringenglish}[1]{%
283 \ifcase#1\relax
284 Zeroth%
285 \or First%
286 \or Second%
287 \or Third%
288 \or Fourth%
289 \or Fifth%
290 \or Sixth%
291 \or Seventh%
292 \or Eighth%
293 \or Ninth%
294 \fi
295 }

The tens:
296 \newcommand*{\@@Tenthstringenglish}[1]{%
297 \ifcase#1\relax
298 \or Tenth%
299 \or Twentieth%
300 \or Thirtieth%
301 \or Fortieth%
302 \or Fiftieth%
303 \or Sixtieth%
304 \or Seventieth%
305 \or Eightieth%

```

```

306 \or Ninetieth%
307 \fi
308 }

```

The teens:

```

309 \newcommand*{\@@Teenthstringenglish}[1]{%
310 \ifcase#1\relax
311 Tenth%
312 \or Eleventh%
313 \or Twelfth%
314 \or Thirteenth%
315 \or Fourteenth%
316 \or Fifteenth%
317 \or Sixteenth%
318 \or Seventeenth%
319 \or Eighteenth%
320 \or Nineteenth%
321 \fi
322 }

```

Again, as from version 1.09, this has been changed to take two arguments, where the second argument is a control sequence. The resulting text is stored in the control sequence, and nothing is displayed.

```

323 \newcommand*{\@@ordinalstringenglish}[2]{%
324 \@strctr=#1\relax
325 \ifnum#1>99999
326 \PackageError{fmtcount}{Out of range}%
327 {This macro only works for values less than 100000 (value given: \number\@strctr)}%
328 \else
329 \ifnum#1<0
330 \PackageError{fmtcount}{Negative numbers not permitted}%
331 {This macro does not work for negative numbers, however
332 you can try typing "minus" first, and then pass the modulus of
333 this number}%
334 \fi
335 \def#2{}%
336 \fi
337 \@strctr=#1\relax \divide\@strctr by 1000\relax
338 \ifnum\@strctr>9\relax
339 % #1 is greater or equal to 10000
340 \divide\@strctr by 10
341 \ifnum\@strctr>1\relax
342 \let\@@fc@ordstr#2\relax
343 \edef#2{\@@fc@ordstr\@tenstring{\@strctr}}%
344 \@strctr=#1\relax
345 \divide\@strctr by 1000\relax
346 \@modulo{\@strctr}{10}%
347 \ifnum\@strctr>0\relax
348 \let\@@fc@ordstr#2\relax
349 \edef#2{\@@fc@ordstr-\@unitstring{\@strctr}}%
350 \fi
351 \else
352 \@strctr=#1\relax \divide\@strctr by 1000\relax
353 \@modulo{\@strctr}{10}%
354 \let\@@fc@ordstr#2\relax

```

```

355 \edef#2{\@@fc@ordstr\@teenstring{\@strctr}}%
356 \fi
357 \@strctr=#1\relax \@modulo{\@strctr}{1000}%
358 \ifnum\@strctr=0\relax
359 \let\@@fc@ordstr#2\relax
360 \edef#2{\@@fc@ordstr\ \@thousandth}%
361 \else
362 \let\@@fc@ordstr#2\relax
363 \edef#2{\@@fc@ordstr\ \@thousand}%
364 \fi
365 \else
366 \ifnum\@strctr>0\relax
367 \let\@@fc@ordstr#2\relax
368 \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
369 \@strctr=#1\relax \@modulo{\@strctr}{1000}%
370 \let\@@fc@ordstr#2\relax
371 \ifnum\@strctr=0\relax
372 \edef#2{\@@fc@ordstr\ \@thousandth}%
373 \else
374 \edef#2{\@@fc@ordstr\ \@thousand}%
375 \fi
376 \fi
377 \fi
378 \@strctr=#1\relax \@modulo{\@strctr}{1000}%
379 \divide\@strctr by 100
380 \ifnum\@strctr>0\relax
381 \ifnum#1>1000\relax
382 \let\@@fc@ordstr#2\relax
383 \edef#2{\@@fc@ordstr\ }%
384 \fi
385 \let\@@fc@ordstr#2\relax
386 \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
387 \@strctr=#1\relax \@modulo{\@strctr}{100}%
388 \let\@@fc@ordstr#2\relax
389 \ifnum\@strctr=0\relax
390 \edef#2{\@@fc@ordstr\ \@hundredth}%
391 \else
392 \edef#2{\@@fc@ordstr\ \@hundred}%
393 \fi
394 \fi
395 \@strctr=#1\relax \@modulo{\@strctr}{100}%
396 \ifnum#1>100\relax
397 \ifnum\@strctr>0\relax
398 \let\@@fc@ordstr#2\relax
399 \edef#2{\@@fc@ordstr\ \@andname\ }%
400 \fi
401 \fi
402 \ifnum\@strctr>19\relax
403 \@tmpstrctr=\@strctr
404 \divide\@strctr by 10\relax
405 \@modulo{\@tmpstrctr}{10}%
406 \let\@@fc@ordstr#2\relax
407 \ifnum\@tmpstrctr=0\relax
408 \edef#2{\@@fc@ordstr\@tenthstring{\@strctr}}%

```

```

409 \else
410   \edef#2{\@fc@ordstr\@tenstring{\@strctr}}%
411 \fi
412 \@strctr=#1\relax \@modulo{\@strctr}{10}%
413 \ifnum\@strctr>0\relax
414   \let\@fc@ordstr#2\relax
415   \edef#2{\@fc@ordstr-\@unitthstring{\@strctr}}%
416 \fi
417 \else
418   \ifnum\@strctr<10\relax
419     \ifnum\@strctr=0\relax
420       \ifnum#1<100\relax
421         \let\@fc@ordstr#2\relax
422         \edef#2{\@fc@ordstr\@unitthstring{\@strctr}}%
423       \fi
424     \else
425       \let\@fc@ordstr#2\relax
426       \edef#2{\@fc@ordstr\@unitthstring{\@strctr}}%
427     \fi
428   \else
429     \@modulo{\@strctr}{10}%
430     \let\@fc@ordstr#2\relax
431     \edef#2{\@fc@ordstr\@teenthstring{\@strctr}}%
432   \fi
433 \fi
434 }

```

All lower case version. Again, the second argument must be a control sequence in which the resulting text is stored.

```

435 \DeclareRobustCommand{\@ordinalstringMenglish}[2]{%
436 \let\@unitthstring=\@unitthstringenglish
437 \let\@teenthstring=\@teenthstringenglish
438 \let\@tenthstring=\@tenthstringenglish
439 \let\@unitstring=\@unitstringenglish
440 \let\@teenstring=\@teenstringenglish
441 \let\@tenstring=\@tenstringenglish
442 \def\@andname{and}%
443 \def\@hundred{hundred}\def\@thousand{thousand}%
444 \def\@hundredth{hundredth}\def\@thousandth{thousandth}%
445 \@ordinalstringenglish{#1}{#2}}

```

No gender in English, so make feminine and neuter same as masculine:

```

446 \let\@ordinalstringFenglish=\@ordinalstringMenglish
447 \let\@ordinalstringNenglish=\@ordinalstringMenglish

```

First letter of each word in upper case:

```

448 \DeclareRobustCommand{\@OrdinalstringMenglish}[2]{%
449 \let\@unitthstring=\@Unitthstringenglish
450 \let\@teenthstring=\@Teenthstringenglish
451 \let\@tenthstring=\@Tenthstringenglish
452 \let\@unitstring=\@Unitstringenglish
453 \let\@teenstring=\@Teenstringenglish
454 \let\@tenstring=\@Tenstringenglish
455 \def\@andname{and}%
456 \def\@hundred{Hundred}\def\@thousand{Thousand}%

```

```

457 \def\@hundredth{Hundredth}\def\@thousandth{Thousandth}%
458 \@@ordinalstringenglish{#1}{#2}}

```

No gender in English, so make feminine and neuter same as masculine:

```

459 \let\@OrdinalstringFenglish=\@OrdinalstringMenglish
460 \let\@OrdinalstringNenglish=\@OrdinalstringMenglish

```

11.3 fc-french.def

French definitions

```

461 \ProvidesFile{fc-french.def}[2007/05/26]

```

Define macro that converts a number or count register (first argument) to an ordinal, and store the result in the second argument, which must be a control sequence. Masculine:

```

462 \newcommand*{\@ordinalMfrench}[2]{%
463 \iffmtord@abbrv
464 \edef#2{\number#1\relax\noexpand\fmtord{e}}}%
465 \else
466 \ifnum#1=1\relax
467 \edef#2{\number#1\relax\noexpand\fmtord{er}}}%
468 \else
469 \edef#2{\number#1\relax\noexpand\fmtord{eme}}}%
470 \fi
471 \fi}

```

Feminine:

```

472 \newcommand*{\@ordinalFfrench}[2]{%
473 \iffmtord@abbrv
474 \edef#2{\number#1\relax\noexpand\fmtord{e}}}%
475 \else
476 \ifnum#1=1\relax
477 \edef#2{\number#1\relax\noexpand\fmtord{ere}}}%
478 \else
479 \edef#2{\number#1\relax\noexpand\fmtord{eme}}}%
480 \fi
481 \fi}

```

Make neuter same as masculine:

```

482 \let\@ordinalNfrench\@ordinalMfrench

```

Textual representation of a number. To make it easier break it into units, tens and teens. First the units:

```

483 \newcommand*{\@unitstringfrench}[1]{%
484 \ifcase#1\relax
485 zero%
486 \or un%
487 \or deux%
488 \or trois%
489 \or quatre%
490 \or cinq%
491 \or six%
492 \or sept%
493 \or huit%
494 \or neuf%

```



```

495 \fi
496 }

```

Feminine only changes for 1:

```

497 \newcommand*{\@@unitstringfrench}[1]{%
498 \ifnum#1=1\relax
499 une%
500 \else\@@unitstringfrench{#1}%
501 \fi
502 }

```

Tens (this includes the Belgian and Swiss variants, special cases employed lower down.)

```

503 \newcommand*{\@@tenstringfrench}[1]{%
504 \ifcase#1\relax
505 \or dix%
506 \or vingt%
507 \or trente%
508 \or quarante%
509 \or cinquante%
510 \or soixante%
511 \or septante%
512 \or huitante%
513 \or nonante%
514 \or cent%
515 \fi
516 }

```

Teens:

```

517 \newcommand*{\@@teenstringfrench}[1]{%
518 \ifcase#1\relax
519 dix%
520 \or onze%
521 \or douze%
522 \or treize%
523 \or quatorze%
524 \or quinze%
525 \or seize%
526 \or dix-sept%
527 \or dix-huit%
528 \or dix-neuf%
529 \fi
530 }

```

Seventies are a special case, depending on dialect:

```

531 \newcommand*{\@@seventiesfrench}[1]{%
532 \@@tenstring{6}%
533 \ifnum#1=1\relax
534 \ \@andname\
535 \else
536 -%
537 \fi
538 \@@teenstring{#1}%
539 }

```

Eighties are a special case, depending on dialect:

```

540 \newcommand*{\@@eightiesfrench}[1]{%
541 \@unitstring{4}-\@tenstring{2}%
542 \ifnum#1>0
543 -\@unitstring{#1}%
544 \else
545 s%
546 \fi
547 }

```

Nineties are a special case, depending on dialect:

```

548 \newcommand*{\@@ninetiesfrench}[1]{%
549 \@unitstring{4}-\@tenstring{2}-\@teenstring{#1}%
550 }

```

Swiss seventies:

```

551 \newcommand*{\@@seventiesfrenchswiss}[1]{%
552 \@tenstring{7}%
553 \ifnum#1=1\ \@andname\ \fi
554 \ifnum#1>1-\fi
555 \ifnum#1>0\@unitstring{#1}\fi
556 }

```

Swiss eighties:

```

557 \newcommand*{\@@eightiesfrenchswiss}[1]{%
558 \@tenstring{8}%
559 \ifnum#1=1\ \@andname\ \fi
560 \ifnum#1>1-\fi
561 \ifnum#1>0\@unitstring{#1}\fi
562 }

```

Swiss nineties:

```

563 \newcommand*{\@@ninetiesfrenchswiss}[1]{%
564 \@tenstring{9}%
565 \ifnum#1=1\ \@andname\ \fi
566 \ifnum#1>1-\fi
567 \ifnum#1>0\@unitstring{#1}\fi
568 }

```

Units with initial letter in upper case:

```

569 \newcommand*{\@@Unitstringfrench}[1]{%
570 \ifcase#1\relax
571 Zero%
572 \or Un%
573 \or Deux%
574 \or Trois%
575 \or Quatre%
576 \or Cinq%
577 \or Six%
578 \or Sept%
579 \or Huit%
580 \or Neuf%
581 \fi
582 }

```

As above, but feminine:

```

583 \newcommand*{\@@UnitstringFfrench}[1]{%

```

```

584 \ifnum#1=1\relax
585 Une%
586 \else \@@Unitstringfrench{#1}%
587 \fi
588 }

```

Tens, with initial letter in upper case (includes Swiss and Belgian variants):

```

589 \newcommand*\@@Tenstringfrench}[1]{%
590 \ifcase#1\relax
591 \or Dix%
592 \or Vingt%
593 \or Trente%
594 \or Quarante%
595 \or Cinquante%
596 \or Soixante%
597 \or Septente%
598 \or Huitante%
599 \or Nonente%
600 \or Cent%
601 \fi
602 }

```

Teens, with initial letter in upper case:

```

603 \newcommand*\@@Teenstringfrench}[1]{%
604 \ifcase#1\relax
605 Dix%
606 \or Onze%
607 \or Douze%
608 \or Treize%
609 \or Quatorze%
610 \or Quinze%
611 \or Seize%
612 \or Dix-Sept%
613 \or Dix-Huit%
614 \or Dix-Neuf%
615 \fi
616 }

```

This has changed in version 1.09, so that it now stores the result in the second argument, but doesn't display anything. Since it only affects internal macros, it shouldn't affect documents created with older versions. (These internal macros are not defined for use in documents.) Firstly, the Swiss version:

```

617 \DeclareRobustCommand{\@numberstringMfrenchswiss}[2]{%
618 \let\@unitstring=\@@unitstringfrench
619 \let\@teenstring=\@@teenstringfrench
620 \let\@tenstring=\@@tenstringfrench
621 \let\@seventies=\@@seventiesfrenchswiss
622 \let\@eighties=\@@eightiesfrenchswiss
623 \let\@nineties=\@@ninetiesfrenchswiss
624 \def\@hundred{cent}\def\@thousand{mille}%
625 \def\@andname{et}%
626 \@numberstringfrench{#1}{#2}}

```

Same as above, but for French as spoken in France:

```

627 \DeclareRobustCommand{\@numberstringMfrenchfrance}[2]{%

```

```

628 \let\@unitstring=\@unitstringfrench
629 \let\@teenstring=\@teenstringfrench
630 \let\@tenstring=\@tenstringfrench
631 \let\@seventies=\@seventiesfrench
632 \let\@eighties=\@eightiesfrench
633 \let\@nineties=\@ninetiesfrench
634 \def\@hundred{cent}\def\@thousand{mille}%
635 \def\@andname{et}%
636 \@numberstringfrench{#1}{#2}}

Same as above, but for Belgian dialect:
637 \DeclareRobustCommand{\@numberstringMfrenchbelgian}[2]{%
638 \let\@unitstring=\@unitstringfrench
639 \let\@teenstring=\@teenstringfrench
640 \let\@tenstring=\@tenstringfrench
641 \let\@seventies=\@seventiesfrenchswiss
642 \let\@eighties=\@eightiesfrench
643 \let\@nineties=\@ninetiesfrench
644 \def\@hundred{cent}\def\@thousand{mille}%
645 \def\@andname{et}%
646 \@numberstringfrench{#1}{#2}}

Set default dialect:
647 \let\@numberstringMfrench=\@numberstringMfrenchfrance

As above, but for feminine version. Swiss:
648 \DeclareRobustCommand{\@numberstringFfrenchswiss}[2]{%
649 \let\@unitstring=\@unitstringFfrench
650 \let\@teenstring=\@teenstringfrench
651 \let\@tenstring=\@tenstringfrench
652 \let\@seventies=\@seventiesfrenchswiss
653 \let\@eighties=\@eightiesfrenchswiss
654 \let\@nineties=\@ninetiesfrenchswiss
655 \def\@hundred{cent}\def\@thousand{mille}%
656 \def\@andname{et}%
657 \@numberstringfrench{#1}{#2}}

French:
658 \DeclareRobustCommand{\@numberstringFfrenchfrance}[2]{%
659 \let\@unitstring=\@unitstringFfrench
660 \let\@teenstring=\@teenstringfrench
661 \let\@tenstring=\@tenstringfrench
662 \let\@seventies=\@seventiesfrench
663 \let\@eighties=\@eightiesfrench
664 \let\@nineties=\@ninetiesfrench
665 \def\@hundred{cent}\def\@thousand{mille}%
666 \def\@andname{et}%
667 \@numberstringfrench{#1}{#2}}

Belgian:
668 \DeclareRobustCommand{\@numberstringFfrenchbelgian}[2]{%
669 \let\@unitstring=\@unitstringFfrench
670 \let\@teenstring=\@teenstringfrench
671 \let\@tenstring=\@tenstringfrench
672 \let\@seventies=\@seventiesfrenchswiss
673 \let\@eighties=\@eightiesfrench

```

```

674 \let\@nineties=\@ninetiesfrench
675 \def\@hundred{cent}\def\@thousand{mille}%
676 \def\@andname{et}%
677 \@@numberstringfrench{#1}{#2}}

Set default dialect:
678 \let\@numberstringFfrench=\@numberstringFfrenchfrance

Make neuter same as masculine:
679 \let\@ordinalstringNfrench\@ordinalstringMfrench

As above, but with initial letter in upper case. Swiss (masculine):
680 \DeclareRobustCommand{\@NumberstringMfrenchswiss}[2]{%
681 \let\@unitstring=\@Unitstringfrench
682 \let\@teenstring=\@Teenstringfrench
683 \let\@tenstring=\@Tenstringfrench
684 \let\@seventies=\@seventiesfrenchswiss
685 \let\@eighties=\@eightiesfrenchswiss
686 \let\@nineties=\@ninetiesfrenchswiss
687 \def\@hundred{Cent}\def\@thousand{Mille}%
688 \def\@andname{et}%
689 \@@numberstringfrench{#1}{#2}}

French:
690 \DeclareRobustCommand{\@NumberstringMfrenchfrance}[2]{%
691 \let\@unitstring=\@Unitstringfrench
692 \let\@teenstring=\@Teenstringfrench
693 \let\@tenstring=\@Tenstringfrench
694 \let\@seventies=\@seventiesfrench
695 \let\@eighties=\@eightiesfrench
696 \let\@nineties=\@ninetiesfrench
697 \def\@hundred{Cent}\def\@thousand{Mille}%
698 \def\@andname{et}%
699 \@@numberstringfrench{#1}{#2}}

Belgian:
700 \DeclareRobustCommand{\@NumberstringMfrenchbelgian}[2]{%
701 \let\@unitstring=\@Unitstringfrench
702 \let\@teenstring=\@Teenstringfrench
703 \let\@tenstring=\@Tenstringfrench
704 \let\@seventies=\@seventiesfrenchswiss
705 \let\@eighties=\@eightiesfrench
706 \let\@nineties=\@ninetiesfrench
707 \def\@hundred{Cent}\def\@thousand{Mille}%
708 \def\@andname{et}%
709 \@@numberstringfrench{#1}{#2}}

Set default dialect:
710 \let\@NumberstringMfrench=\@NumberstringMfrenchfrance

As above, but feminine. Swiss:
711 \DeclareRobustCommand{\@NumberstringFfrenchswiss}[2]{%
712 \let\@unitstring=\@UnitstringFfrench
713 \let\@teenstring=\@Teenstringfrench
714 \let\@tenstring=\@Tenstringfrench
715 \let\@seventies=\@seventiesfrenchswiss
716 \let\@eighties=\@eightiesfrenchswiss

```

```

717 \let\@nineties=\@ninetiesfrenchswiss
718 \def\@hundred{Cent}\def\@thousand{Mille}%
719 \def\@andname{et}%
720 \@@numberstringfrench{#1}{#2}}

French (feminine):
721 \DeclareRobustCommand{\@NumberstringFfrenchfrance}[2]{%
722 \let\@unitstring=\@UnitstringFfrench
723 \let\@teenstring=\@Teenstringfrench
724 \let\@tenstring=\@Tenstringfrench
725 \let\@seventies=\@seventiesfrench
726 \let\@eighties=\@eightiesfrench
727 \let\@nineties=\@ninetiesfrench
728 \def\@hundred{Cent}\def\@thousand{Mille}%
729 \def\@andname{et}%
730 \@@numberstringfrench{#1}{#2}}

Belgian (feminine):
731 \DeclareRobustCommand{\@NumberstringFfrenchbelgian}[2]{%
732 \let\@unitstring=\@UnitstringFfrench
733 \let\@teenstring=\@Teenstringfrench
734 \let\@tenstring=\@Tenstringfrench
735 \let\@seventies=\@seventiesfrenchswiss
736 \let\@eighties=\@eightiesfrench
737 \let\@nineties=\@ninetiesfrench
738 \def\@hundred{Cent}\def\@thousand{Mille}%
739 \def\@andname{et}%
740 \@@numberstringfrench{#1}{#2}}

Set default dialect:
741 \let\@NumberstringFfrench=\@NumberstringFfrenchfrance

Make neuter same as masculine:
742 \let\@NumberstringNfrench\@NumberstringMfrench

Again, as from version 1.09, this has been changed to take two arguments, where
the second argument is a control sequence, and nothing is displayed. Store textual
representation of an ordinal in the given control sequence. Swiss dialect (mascu-
line):
743 \DeclareRobustCommand{\@OrdinalstringMfrenchswiss}[2]{%
744 \ifnum#1=1\relax
745 \def#2{premier}%
746 \else
747 \let\@unitthstring=\@unitthstringfrench
748 \let\@unitstring=\@unitstringfrench
749 \let\@teenthstring=\@teenthstringfrench
750 \let\@teenstring=\@teenstringfrench
751 \let\@tenthstring=\@tenthstringfrench
752 \let\@tenstring=\@tenstringfrench
753 \let\@seventieths=\@seventiethsfrenchswiss
754 \let\@eightieths=\@eightiethsfrenchswiss
755 \let\@ninetieths=\@ninetiethsfrenchswiss
756 \let\@seventies=\@seventiesfrenchswiss
757 \let\@eighties=\@eightiesfrenchswiss
758 \let\@nineties=\@ninetiesfrenchswiss
759 \def\@hundredth{centi\'eme}\def\@hundred{cent}%

```

```

760 \def\@thousandth{mili\‘eme}\def\@thousand{mille}%
761 \def\@andname{et}%
762 \@@ordinalstringfrench{#1}{#2}%
763 \fi}

French (masculine):
764 \DeclareRobustCommand{\@ordinalstringMfrenchfrance}[2]{%
765 \ifnum#1=1\relax
766 \def#2{premier}%
767 \else
768 \let\@unitthstring=\@unitthstringfrench
769 \let\@unitstring=\@unitstringfrench
770 \let\@teenthstring=\@teenthstringfrench
771 \let\@teenstring=\@teenstringfrench
772 \let\@tenthstring=\@tenthstringfrench
773 \let\@tenstring=\@tenstringfrench
774 \let\@seventieths=\@seventiethsfrench
775 \let\@eightieths=\@eightiethsfrench
776 \let\@ninetieths=\@ninetiethsfrench
777 \let\@seventies=\@seventiesfrench
778 \let\@eighties=\@eightiesfrench
779 \let\@nineties=\@ninetiesfrench
780 \let\@teenstring=\@teenstringfrench
781 \def\@hundredth{centi\‘eme}\def\@hundred{cent}%
782 \def\@thousandth{mili\‘eme}\def\@thousand{mille}%
783 \def\@andname{et}%
784 \@@ordinalstringfrench{#1}{#2}%
785 \fi}

Belgian dialect (masculine):
786 \DeclareRobustCommand{\@ordinalstringMfrenchbelgian}[2]{%
787 \ifnum#1=1\relax
788 \def#2{premier}%
789 \else
790 \let\@unitthstring=\@unitthstringfrench
791 \let\@unitstring=\@unitstringfrench
792 \let\@teenthstring=\@teenthstringfrench
793 \let\@teenstring=\@teenstringfrench
794 \let\@tenthstring=\@tenthstringfrench
795 \let\@tenstring=\@tenstringfrench
796 \let\@seventieths=\@seventiethsfrenchswiss
797 \let\@eightieths=\@eightiethsfrench
798 \let\@ninetieths=\@ninetiethsfrenchswiss
799 \let\@seventies=\@seventiesfrench
800 \let\@eighties=\@eightiesfrench
801 \let\@nineties=\@ninetiesfrench
802 \let\@teenstring=\@teenstringfrench
803 \def\@hundredth{centi\‘eme}\def\@hundred{cent}%
804 \def\@thousandth{mili\‘eme}\def\@thousand{mille}%
805 \def\@andname{et}%
806 \@@ordinalstringfrench{#1}{#2}%
807 \fi}

Set up default dialect:
808 \let\@ordinalstringMfrench=\@ordinalstringMfrenchfrance

```

As above, but feminine. Swiss:

```

809 \DeclareRobustCommand{\@ordinalstringFfrenchswiss}[2]{%
810 \ifnum#1=1\relax
811 \def#2{premi\‘ere}%
812 \else
813 \let\@unitthstring=\@unitthstringfrench
814 \let\@unitstring=\@unitstringFfrench
815 \let\@teenthstring=\@teenthstringfrench
816 \let\@teenstring=\@teenstringfrench
817 \let\@tenthstring=\@tenthstringfrench
818 \let\@tenstring=\@tenstringfrench
819 \let\@seventieths=\@seventiethsfrenchswiss
820 \let\@eightieths=\@eightiethsfrenchswiss
821 \let\@ninetieths=\@ninetiethsfrenchswiss
822 \let\@seventies=\@seventiesfrenchswiss
823 \let\@eighties=\@eightiesfrenchswiss
824 \let\@nineties=\@ninetiesfrenchswiss
825 \def\@hundredth{centi\‘eme}\def\@hundred{cent}%
826 \def\@thousandth{mili\‘eme}\def\@thousand{mille}%
827 \def\@andname{et}%
828 \@ordinalstringfrench{#1}{#2}%
829 \fi}

```

French (feminine):

```

830 \DeclareRobustCommand{\@ordinalstringFfrenchfrance}[2]{%
831 \ifnum#1=1\relax
832 \def#2{premi\‘ere}%
833 \else
834 \let\@unitthstring=\@unitthstringfrench
835 \let\@unitstring=\@unitstringFfrench
836 \let\@teenthstring=\@teenthstringfrench
837 \let\@teenstring=\@teenstringfrench
838 \let\@tenthstring=\@tenthstringfrench
839 \let\@tenstring=\@tenstringfrench
840 \let\@seventieths=\@seventiethsfrench
841 \let\@eightieths=\@eightiethsfrench
842 \let\@ninetieths=\@ninetiethsfrench
843 \let\@seventies=\@seventiesfrench
844 \let\@eighties=\@eightiesfrench
845 \let\@nineties=\@ninetiesfrench
846 \let\@teenstring=\@teenstringfrench
847 \def\@hundredth{centi\‘eme}\def\@hundred{cent}%
848 \def\@thousandth{mili\‘eme}\def\@thousand{mille}%
849 \def\@andname{et}%
850 \@ordinalstringfrench{#1}{#2}%
851 \fi}

```

Belgian (feminine):

```

852 \DeclareRobustCommand{\@ordinalstringFfrenchbelgian}[2]{%
853 \ifnum#1=1\relax
854 \def#2{premi\‘ere}%
855 \else
856 \let\@unitthstring=\@unitthstringfrench
857 \let\@unitstring=\@unitstringFfrench
858 \let\@teenthstring=\@teenthstringfrench

```



```

859 \let\@teenstring=\@teenstringfrench
860 \let\@tenthstring=\@tenthstringfrench
861 \let\@tenstring=\@tenstringfrench
862 \let\@seventieths=\@seventiethsfrenchswiss
863 \let\@eightieths=\@eightiethsfrench
864 \let\@ninetieths=\@ninetiethsfrench
865 \let\@seventies=\@seventiesfrench
866 \let\@eighties=\@eightiesfrench
867 \let\@nineties=\@ninetiesfrench
868 \let\@teenstring=\@teenstringfrench
869 \def\@hundredth{centi\‘eme}\def\@hundred{cent}%
870 \def\@thousandth{mili\‘eme}\def\@thousand{mille}%
871 \def\@andname{et}%
872 \@ordinalstringfrench{#1}{#2}%
873 \fi}

Set up default dialect:
874 \let\@ordinalstringFfrench=\@ordinalstringFfrenchfrance

Make neuter same as masculine:
875 \let\@ordinalstringNfrench\@ordinalstringMfrench

As above, but with initial letters in upper case. Swiss (masculine):
876 \DeclareRobustCommand{\@OrdinalstringMfrenchswiss}[2]{%
877 \ifnum#1=1\relax
878 \def#2{Premi\‘ere}%
879 \else
880 \let\@unitthstring=\@Unitthstringfrench
881 \let\@unitstring=\@Unitstringfrench
882 \let\@teenthstring=\@Teenthstringfrench
883 \let\@teenstring=\@Teenstringfrench
884 \let\@tenthstring=\@Tenthstringfrench
885 \let\@tenstring=\@Tenstringfrench
886 \let\@seventieths=\@seventiethsfrenchswiss
887 \let\@eightieths=\@eightiethsfrenchswiss
888 \let\@ninetieths=\@ninetiethsfrenchswiss
889 \let\@seventies=\@seventiesfrenchswiss
890 \let\@eighties=\@eightiesfrenchswiss
891 \let\@nineties=\@ninetiesfrenchswiss
892 \def\@hundredth{Centi\‘eme}\def\@hundred{Cent}%
893 \def\@thousandth{Mili\‘eme}\def\@thousand{Mille}%
894 \def\@andname{et}%
895 \@ordinalstringfrench{#1}{#2}%
896 \fi}

French (masculine):
897 \DeclareRobustCommand{\@OrdinalstringMfrenchfrance}[2]{%
898 \ifnum#1=1\relax
899 \def#2{Premi\‘ere}%
900 \else
901 \let\@unitthstring=\@Unitthstringfrench
902 \let\@unitstring=\@Unitstringfrench
903 \let\@teenthstring=\@Teenthstringfrench
904 \let\@teenstring=\@Teenstringfrench
905 \let\@tenthstring=\@Tenthstringfrench
906 \let\@tenstring=\@Tenstringfrench

```

```

907 \let\@seventieths=\@seventiethsfrench
908 \let\@eightieths=\@eightiethsfrench
909 \let\@ninetieths=\@ninetiethsfrench
910 \let\@seventies=\@seventiesfrench
911 \let\@eighties=\@eightiesfrench
912 \let\@nineties=\@ninetiesfrench
913 \let\@teenstring=\@Teenstringfrench
914 \def\@hundredth{Centi\‘eme}\def\@hundred{Cent}%
915 \def\@thousandth{Mili\‘eme}\def\@thousand{Mille}%
916 \def\@andname{et}%
917 \@@ordinalstringfrench{#1}{#2}%
918 \fi}

Belgian (masculine):
919 \DeclareRobustCommand{\@OrdinalstringMfrenchbelgian}[2]{%
920 \ifnum#1=1\relax
921 \def#2{Premi\‘ere}%
922 \else
923 \let\@unitthstring=\@Unitthstringfrench
924 \let\@unitstring=\@Unitstringfrench
925 \let\@teenthstring=\@Teenthstringfrench
926 \let\@teenstring=\@Teenstringfrench
927 \let\@tenthstring=\@Tenthstringfrench
928 \let\@tenstring=\@Tenstringfrench
929 \let\@seventieths=\@seventiethsfrenchswiss
930 \let\@eightieths=\@eightiethsfrench
931 \let\@ninetieths=\@ninetiethsfrench
932 \let\@seventies=\@seventiesfrench
933 \let\@eighties=\@eightiesfrench
934 \let\@nineties=\@ninetiesfrench
935 \let\@teenstring=\@Teenstringfrench
936 \def\@hundredth{Centi\‘eme}\def\@hundred{Cent}%
937 \def\@thousandth{Mili\‘eme}\def\@thousand{Mille}%
938 \def\@andname{et}%
939 \@@ordinalstringfrench{#1}{#2}%
940 \fi}

Set up default dialect:
941 \let\@OrdinalstringMfrench=\@OrdinalstringMfrenchfrance

As above, but feminine form. Swiss:
942 \DeclareRobustCommand{\@OrdinalstringFfrenchswiss}[2]{%
943 \ifnum#1=1\relax
944 \def#2{Premi\‘ere}%
945 \else
946 \let\@unitthstring=\@Unitthstringfrench
947 \let\@unitstring=\@UnitstringFfrench
948 \let\@teenthstring=\@Teenthstringfrench
949 \let\@teenstring=\@Teenstringfrench
950 \let\@tenthstring=\@Tenthstringfrench
951 \let\@tenstring=\@Tenstringfrench
952 \let\@seventieths=\@seventiethsfrenchswiss
953 \let\@eightieths=\@eightiethsfrenchswiss
954 \let\@ninetieths=\@ninetiethsfrenchswiss
955 \let\@seventies=\@seventiesfrenchswiss

```

```

956 \let\@eighties=\@eightiesfrenchswiss
957 \let\@nineties=\@ninetiesfrenchswiss
958 \def\@hundredth{Centi\`eme}\def\@hundred{Cent}%
959 \def\@thousandth{Mili\`eme}\def\@thousand{Mille}%
960 \def\@andname{et}%
961 \@@ordinalstringfrench{#1}{#2}%
962 \fi}

French (feminine):
963 \DeclareRobustCommand{\@OrdinalstringFfrenchfrance}[2]{%
964 \ifnum#1=1\relax
965 \def#2{Premi\`ere}%
966 \else
967 \let\@unitthstring=\@Unitthstringfrench
968 \let\@unitstring=\@UnitstringFfrench
969 \let\@teenthstring=\@Teenthstringfrench
970 \let\@teenstring=\@Teenstringfrench
971 \let\@tenthstring=\@Tenthstringfrench
972 \let\@tenstring=\@Tenstringfrench
973 \let\@seventieths=\@seventiethsfrench
974 \let\@eightieths=\@eightiethsfrench
975 \let\@ninetieths=\@ninetiethsfrench
976 \let\@seventies=\@seventiesfrench
977 \let\@eighties=\@eightiesfrench
978 \let\@nineties=\@ninetiesfrench
979 \let\@teenstring=\@Teenstringfrench
980 \def\@hundredth{Centi\`eme}\def\@hundred{Cent}%
981 \def\@thousandth{Mili\`eme}\def\@thousand{Mille}%
982 \def\@andname{et}%
983 \@@ordinalstringfrench{#1}{#2}%
984 \fi}

Belgian (feminine):
985 \DeclareRobustCommand{\@OrdinalstringFfrenchbelgian}[2]{%
986 \ifnum#1=1\relax
987 \def#2{Premi\`ere}%
988 \else
989 \let\@unitthstring=\@Unitthstringfrench
990 \let\@unitstring=\@UnitstringFfrench
991 \let\@teenthstring=\@Teenthstringfrench
992 \let\@teenstring=\@Teenstringfrench
993 \let\@tenthstring=\@Tenthstringfrench
994 \let\@tenstring=\@Tenstringfrench
995 \let\@seventieths=\@seventiethsfrenchswiss
996 \let\@eightieths=\@eightiethsfrench
997 \let\@ninetieths=\@ninetiethsfrench
998 \let\@seventies=\@seventiesfrench
999 \let\@eighties=\@eightiesfrench
1000 \let\@nineties=\@ninetiesfrench
1001 \let\@teenstring=\@Teenstringfrench
1002 \def\@hundredth{Centi\`eme}\def\@hundred{Cent}%
1003 \def\@thousandth{Mili\`eme}\def\@thousand{Mille}%
1004 \def\@andname{et}%
1005 \@@ordinalstringfrench{#1}{#2}%
1006 \fi}

```

Set up default dialect:

```
1007 \let\@OrdinalstringFfrench=\@OrdinalstringFfrenchfrance
```

Make neuter same as masculine:

```
1008 \let\@OrdinalstringNfrench\@OrdinalstringMfrench
```

In order to convert numbers into textual ordinals, need to break it up into units, tens and teens. First the units. The argument must be a number or count register between 0 and 9.

```
1009 \newcommand*{\@unitthstringfrench}[1]{%
1010 \ifcase#1\relax
1011 zero%
1012 \or uni\'eme%
1013 \or deuxi\'eme%
1014 \or troisi\'eme%
1015 \or quatri\'eme%
1016 \or cinqui\'eme%
1017 \or sixi\'eme%
1018 \or septi\'eme%
1019 \or huiti\'eme%
1020 \or neuvi\'eme%
1021 \fi
1022 }
```

Tens (includes Swiss and Belgian variants, special cases are dealt with later.)

```
1023 \newcommand*{\@tenthstringfrench}[1]{%
1024 \ifcase#1\relax
1025 \or dixi\'eme%
1026 \or vingti\'eme%
1027 \or trentri\'eme%
1028 \or quaranti\'eme%
1029 \or cinquanti\'eme%
1030 \or soixanti\'eme%
1031 \or septenti\'eme%
1032 \or huitanti\'eme%
1033 \or nonenti\'eme%
1034 \fi
1035 }
```

Teens:

```
1036 \newcommand*{\@teenthstringfrench}[1]{%
1037 \ifcase#1\relax
1038 dixi\'eme%
1039 \or onzi\'eme%
1040 \or douzi\'eme%
1041 \or treizi\'eme%
1042 \or quatorzi\'eme%
1043 \or quinzi\'eme%
1044 \or seizi\'eme%
1045 \or dix-septi\'eme%
1046 \or dix-huiti\'eme%
1047 \or dix-neuvi\'eme%
1048 \fi
1049 }
```

Seventies vary depending on dialect. Swiss:

```
1050 \newcommand*{\@@seventiethsfrenchswiss}[1]{%
1051 \ifcase#1\relax
1052 \@tenthstring{7}%
1053 \or
1054 \@tenstring{7} \@andname\ \@unitthstring{1}%
1055 \else
1056 \@tenstring{7}-\@unitthstring{#1}%
1057 \fi}
```

Eighties vary depending on dialect. Swiss:

```
1058 \newcommand*{\@@eightiethsfrenchswiss}[1]{%
1059 \ifcase#1\relax
1060 \@tenthstring{8}%
1061 \or
1062 \@tenstring{8} \@andname\ \@unitthstring{1}%
1063 \else
1064 \@tenstring{8}-\@unitthstring{#1}%
1065 \fi}
```

Nineties vary depending on dialect. Swiss:

```
1066 \newcommand*{\@@ninetiethsfrenchswiss}[1]{%
1067 \ifcase#1\relax
1068 \@tenthstring{9}%
1069 \or
1070 \@tenstring{9} \@andname\ \@unitthstring{1}%
1071 \else
1072 \@tenstring{9}-\@unitthstring{#1}%
1073 \fi}
```

French (as spoken in France) version:

```
1074 \newcommand*{\@@seventiethsfrench}[1]{%
1075 \ifnum#1=0\relax
1076 \@tenstring{6}%
1077 -%
1078 \else
1079 \@tenstring{6}%
1080 \ \@andname\
1081 \fi
1082 \@teenthstring{#1}%
1083 }
```

Eighties (as spoken in France):

```
1084 \newcommand*{\@@eightiethsfrench}[1]{%
1085 \ifnum#1>0\relax
1086 \@unitstring{4}-\@tenstring{2}%
1087 -\@unitthstring{#1}%
1088 \else
1089 \@unitstring{4}-\@tenthstring{2}%
1090 \fi
1091 }
```

Nineties (as spoken in France):

```
1092 \newcommand*{\@@ninetiethsfrench}[1]{%
1093 \@unitstring{4}-\@tenstring{2}-\@teenthstring{#1}%
1094 }
```

As above, but with initial letter in upper case. Units:

```

1095 \newcommand*{\@@Unitthstringfrench}[1]{%
1096 \ifcase#1\relax
1097 Zero%
1098 \or Uni\‘eme%
1099 \or Deuxi\‘eme%
1100 \or Troisi\‘eme%
1101 \or Quatri\‘eme%
1102 \or Cinqui\‘eme%
1103 \or Sixi\‘eme%
1104 \or Septi\‘eme%
1105 \or Huiti\‘eme%
1106 \or Neuvi\‘eme%
1107 \fi
1108 }

```

Tens (includes Belgian and Swiss variants):

```

1109 \newcommand*{\@@Tenthstringfrench}[1]{%
1110 \ifcase#1\relax
1111 \or Dixi\‘eme%
1112 \or Vingti\‘eme%
1113 \or Trentri\‘eme%
1114 \or Quaranti\‘eme%
1115 \or Cinquanti\‘eme%
1116 \or Soixanti\‘eme%
1117 \or Septenti\‘eme%
1118 \or Huitanti\‘eme%
1119 \or Nonenti\‘eme%
1120 \fi
1121 }

```

Teens:

```

1122 \newcommand*{\@@Teenthstringfrench}[1]{%
1123 \ifcase#1\relax
1124 Dixi\‘eme%
1125 \or Onzi\‘eme%
1126 \or Douzi\‘eme%
1127 \or Treizi\‘eme%
1128 \or Quatorzi\‘eme%
1129 \or Quinzi\‘eme%
1130 \or Seizi\‘eme%
1131 \or Dix-Septi\‘eme%
1132 \or Dix-Huiti\‘eme%
1133 \or Dix-Neuvi\‘eme%
1134 \fi
1135 }

```

Store textual representation of number (first argument) in given control sequence (second argument).

```

1136 \newcommand*{\@@numberstringfrench}[2]{%
1137 \ifnum#1>99999
1138 \PackageError{fmtcount}{Out of range}%
1139 {This macro only works for values less than 100000}%
1140 \else
1141 \ifnum#1<0

```

```

1142 \PackageError{fmtcount}{Negative numbers not permitted}%
1143 {This macro does not work for negative numbers, however
1144 you can try typing "minus" first, and then pass the modulus of
1145 this number}%
1146 \fi
1147 \fi
1148 \def#2{%
1149 \@strctr=#1\relax \divide\@strctr by 1000\relax
1150 \ifnum\@strctr>9\relax
1151 % #1 is greater or equal to 10000
1152 \@tmpstrctr=\@strctr
1153 \divide\@strctr by 10\relax
1154 \ifnum\@strctr>1\relax
1155 \ifthenelse{(\@strctr>6)\and(\@strctr<10)}{%
1156 \@modulo{\@tmpstrctr}{10}%
1157 \ifnum\@strctr<8\relax
1158 \let\@fc@numstr#2\relax
1159 \edef#2{\@fc@numstr\@seventies{\@tmpstrctr}}%
1160 \else
1161 \ifnum\@strctr<9\relax
1162 \let\@fc@numstr#2\relax
1163 \edef#2{\@fc@numstr\@eighties{\@tmpstrctr}}%
1164 \else
1165 \ifnum\@strctr<10\relax
1166 \let\@fc@numstr#2\relax
1167 \edef#2{\@fc@numstr\@nineties{\@tmpstrctr}}%
1168 \fi
1169 \fi
1170 \fi
1171 }{%
1172 \let\@fc@numstr#2\relax
1173 \edef#2{\@fc@numstr\@tenstring{\@strctr}}%
1174 \@strctr=#1\relax
1175 \divide\@strctr by 1000\relax
1176 \@modulo{\@strctr}{10}%
1177 \ifnum\@strctr>0\relax
1178 \let\@fc@numstr#2\relax
1179 \edef#2{\@fc@numstr\@unitstring{\@strctr}}%
1180 \fi
1181 }%
1182 \else
1183 \@strctr=#1\relax
1184 \divide\@strctr by 1000
1185 \@modulo{\@strctr}{10}%
1186 \let\@fc@numstr#2\relax
1187 \edef#2{\@fc@numstr\@teenstring{\@strctr}}%
1188 \fi
1189 \let\@fc@numstr#2\relax
1190 \edef#2{\@fc@numstr\@thousand}%
1191 \else
1192 \ifnum\@strctr>0\relax
1193 \ifnum\@strctr>1\relax
1194 \let\@fc@numstr#2\relax
1195 \edef#2{\@fc@numstr\@unitstring{\@strctr}\ }%

```

```

1196 \fi
1197 \let\@@fc@numstr#2\relax
1198 \edef#2{\@@fc@numstr\@thousand}%
1199 \fi
1200 \fi
1201 \@strctr=#1\relax \@modulo{\@strctr}{1000}%
1202 \divide\@strctr by 100
1203 \ifnum\@strctr>0\relax
1204 \ifnum#1>1000\relax
1205 \let\@@fc@numstr#2\relax
1206 \edef#2{\@@fc@numstr\ }%
1207 \fi
1208 \@tmpstrctr=#1\relax
1209 \@modulo{\@tmpstrctr}{1000}\relax
1210 \ifnum\@tmpstrctr=100\relax
1211 \let\@@fc@numstr#2\relax
1212 \edef#2{\@@fc@numstr\@tenstring{10}}%
1213 \else
1214 \ifnum\@strctr>1\relax
1215 \let\@@fc@numstr#2\relax
1216 \edef#2{\@@fc@numstr\@unitstring{\@strctr}\ }%
1217 \fi
1218 \let\@@fc@numstr#2\relax
1219 \edef#2{\@@fc@numstr\@hundred}%
1220 \fi
1221 \fi
1222 \@strctr=#1\relax \@modulo{\@strctr}{100}%
1223 %\@tmpstrctr=#1\relax
1224 %\divide\@tmpstrctr by 100\relax
1225 \ifnum#1>100\relax
1226 \ifnum\@strctr>0\relax
1227 \let\@@fc@numstr#2\relax
1228 \edef#2{\@@fc@numstr\ }%
1229 \else
1230 \ifnum\@tmpstrctr>0\relax
1231 \let\@@fc@numstr#2\relax
1232 \edef#2{\@@fc@numstr s}%
1233 \fi%
1234 \fi
1235 \fi
1236 \ifnum\@strctr>19\relax
1237 \@tmpstrctr=\@strctr
1238 \divide\@strctr by 10\relax
1239 \ifthenelse{\@strctr>6}{%
1240 \@modulo{\@tmpstrctr}{10}%
1241 \ifnum\@strctr<8\relax
1242 \let\@@fc@numstr#2\relax
1243 \edef#2{\@@fc@numstr\@seventies{\@tmpstrctr}}%
1244 \else
1245 \ifnum\@strctr<9\relax
1246 \let\@@fc@numstr#2\relax
1247 \edef#2{\@@fc@numstr\@eighties{\@tmpstrctr}}%
1248 \else
1249 \let\@@fc@numstr#2\relax

```



```

1250     \edef#2{\@fc@numstr@nineties{\@tmpstrctr}}%
1251     \fi
1252 \fi
1253 }{%
1254     \let\@fc@numstr#2\relax
1255     \edef#2{\@fc@numstr@tenstring{\@strctr}}%
1256     \@strctr=#1\relax \@modulo{\@strctr}{10}%
1257     \ifnum\@strctr>0\relax
1258         \let\@fc@numstr#2\relax
1259         \ifnum\@strctr=1\relax
1260             \edef#2{\@fc@numstr\ \@andname\ }%
1261         \else
1262             \edef#2{\@fc@numstr-}%
1263         \fi
1264         \let\@fc@numstr#2\relax
1265         \edef#2{\@fc@numstr@unitstring{\@strctr}}%
1266     \fi
1267 }%
1268 \else
1269     \ifnum\@strctr<10\relax
1270         \ifnum\@strctr=0\relax
1271             \ifnum#1<100\relax
1272                 \let\@fc@numstr#2\relax
1273                 \edef#2{\@fc@numstr@unitstring{\@strctr}}%
1274             \fi
1275             \else%(>0,<10)
1276                 \let\@fc@numstr#2\relax
1277                 \edef#2{\@fc@numstr@unitstring{\@strctr}}%
1278             \fi
1279             \else%>10
1280                 \@modulo{\@strctr}{10}%
1281                 \let\@fc@numstr#2\relax
1282                 \edef#2{\@fc@numstr@teenstring{\@strctr}}%
1283             \fi
1284 \fi
1285 }

```

Store textual representation of an ordinal (from number specified in first argument) in given control sequence (second argument).

```

1286 \newcommand*{\@ordinalstringfrench}[2]{%
1287 \ifnum#1>99999
1288 \PackageError{fmtcount}{Out of range}%
1289 {This macro only works for values less than 100000}%
1290 \else
1291 \ifnum#1<0
1292 \PackageError{fmtcount}{Negative numbers not permitted}%
1293 {This macro does not work for negative numbers, however
1294 you can try typing "minus" first, and then pass the modulus of
1295 this number}%
1296 \fi
1297 \fi
1298 \def#2{}%
1299 \@strctr=#1\relax \divide\@strctr by 1000\relax
1300 \ifnum\@strctr>9

```

```

1301 % #1 is greater or equal to 10000
1302 \@tmpstrctr=\@strctr
1303 \divide\@strctr by 10\relax
1304 \ifnum\@strctr>1\relax
1305 \ifthenelse{\@strctr>6}{%
1306 \@modulo{\@tmpstrctr}{10}%
1307 \ifnum\@strctr=7\relax
1308 \let\@fc@ordstr#2\relax
1309 \edef#2{\@fc@ordstr\@seventies{\@tmpstrctr}}%
1310 \else
1311 \ifnum\@strctr=8\relax
1312 \let\@fc@ordstr#2\relax
1313 \edef#2{\@fc@ordstr\@eighties{\@tmpstrctr}}%
1314 \else
1315 \let\@fc@ordstr#2\relax
1316 \edef#2{\@fc@ordstr\@nineties{\@tmpstrctr}}%
1317 \fi
1318 \fi
1319 }{%
1320 \let\@fc@ordstr#2\relax
1321 \edef#2{\@fc@ordstr\@tenstring{\@strctr}}%
1322 \@strctr=#1\relax
1323 \divide\@strctr by 1000\relax
1324 \@modulo{\@strctr}{10}%
1325 \ifnum\@strctr=1\relax
1326 \let\@fc@ordstr#2\relax
1327 \edef#2{\@fc@ordstr\ \@andname}%
1328 \fi
1329 \ifnum\@strctr>0\relax
1330 \let\@fc@ordstr#2\relax
1331 \edef#2{\@fc@ordstr\ \@unitstring{\@strctr}}%
1332 \fi
1333 }%
1334 \else
1335 \@strctr=#1\relax
1336 \divide\@strctr by 1000\relax
1337 \@modulo{\@strctr}{10}%
1338 \let\@fc@ordstr#2\relax
1339 \edef#2{\@fc@ordstr\@teenstring{\@strctr}}%
1340 \fi
1341 \@strctr=#1\relax \@modulo{\@strctr}{1000}%
1342 \ifnum\@strctr=0\relax
1343 \let\@fc@ordstr#2\relax
1344 \edef#2{\@fc@ordstr\ \@thousandth}%
1345 \else
1346 \let\@fc@ordstr#2\relax
1347 \edef#2{\@fc@ordstr\ \@thousand}%
1348 \fi
1349 \else
1350 \ifnum\@strctr>0\relax
1351 \let\@fc@ordstr#2\relax
1352 \edef#2{\@fc@ordstr\@unitstring{\@strctr}}%
1353 \@strctr=#1\relax \@modulo{\@strctr}{1000}%
1354 \ifnum\@strctr=0\relax

```

```

1355     \let\@@fc@ordstr#2\relax
1356     \edef#2{\@@fc@ordstr\ \@thousandth}%
1357   \else
1358     \let\@@fc@ordstr#2\relax
1359     \edef#2{\@@fc@ordstr\ \@thousand}%
1360   \fi
1361 \fi
1362 \fi
1363 \@strctr=#1\relax \@modulo{\@strctr}{1000}%
1364 \divide\@strctr by 100\relax
1365 \ifnum\@strctr>0\relax
1366   \ifnum#1>1000\relax
1367     \let\@@fc@ordstr#2\relax
1368     \edef#2{\@@fc@ordstr\ }%
1369   \fi
1370   \let\@@fc@ordstr#2\relax
1371   \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
1372   \@strctr=#1\relax \@modulo{\@strctr}{100}%
1373   \let\@@fc@ordstr#2\relax
1374   \ifnum\@strctr=0\relax
1375     \edef#2{\@@fc@ordstr\ \@hundredth}%
1376   \else
1377     \edef#2{\@@fc@ordstr\ \@hundred}%
1378   \fi
1379 \fi
1380 \@tmpstrctr=\@strctr
1381 \@strctr=#1\relax \@modulo{\@strctr}{100}%
1382 \ifnum#1>100\relax
1383   \ifnum\@strctr>0\relax
1384     \let\@@fc@ordstr#2\relax
1385     \edef#2{\@@fc@ordstr\ \@andname\ }%
1386   \fi
1387 \fi
1388 \ifnum\@strctr>19\relax
1389   \@tmpstrctr=\@strctr
1390   \divide\@strctr by 10\relax
1391   \@modulo{\@tmpstrctr}{10}%
1392   \ifthenelse{\@strctr>6}{%
1393     \ifnum\@strctr=7\relax
1394       \let\@@fc@ordstr#2\relax
1395       \edef#2{\@@fc@ordstr\@seventieths{\@tmpstrctr}}%
1396     \else
1397       \ifnum\@strctr=8\relax
1398         \let\@@fc@ordstr#2\relax
1399         \edef#2{\@@fc@ordstr\@eightieths{\@tmpstrctr}}%
1400       \else
1401         \let\@@fc@ordstr#2\relax
1402         \edef#2{\@@fc@ordstr\@ninetieths{\@tmpstrctr}}%
1403       \fi
1404     \fi
1405   }{%
1406     \ifnum\@tmpstrctr=0\relax
1407       \let\@@fc@ordstr#2\relax
1408       \edef#2{\@@fc@ordstr\@tenthstring{\@strctr}}%

```

```

1409 \else
1410   \let\@@fc@ordstr#2\relax
1411   \edef#2{\@@fc@ordstr\@tenstring{\@strctr}}%
1412 \fi
1413 \@strctr=#1\relax \@modulo{\@strctr}{10}%
1414 \ifnum\@strctr=1\relax
1415   \let\@@fc@ordstr#2\relax
1416   \edef#2{\@@fc@ordstr\ \@andname}%
1417 \fi
1418 \ifnum\@strctr>0\relax
1419   \let\@@fc@ordstr#2\relax
1420   \edef#2{\@@fc@ordstr\ \@unitthstring{\@strctr}}%
1421 \fi
1422 }%
1423 \else
1424   \ifnum\@strctr<10\relax
1425     \ifnum\@strctr=0\relax
1426       \ifnum#1<100\relax
1427         \let\@@fc@ordstr#2\relax
1428         \edef#2{\@@fc@ordstr\@unitthstring{\@strctr}}%
1429       \fi
1430     \else
1431       \let\@@fc@ordstr#2\relax
1432       \edef#2{\@@fc@ordstr\@unitthstring{\@strctr}}%
1433     \fi
1434   \else
1435     \@modulo{\@strctr}{10}%
1436     \let\@@fc@ordstr#2\relax
1437     \edef#2{\@@fc@ordstr\@teenthstring{\@strctr}}%
1438   \fi
1439 \fi
1440 }

```

11.4 fc-german.def

German definitions (thank you to K. H. Fricke for supplying this information)

```
1441 \ProvidesFile{fc-german.def}[2007/06/14]
```

Define macro that converts a number or count register (first argument) to an ordinal, and stores the result in the second argument, which must be a control sequence. Masculine:

```

1442 \newcommand{\@ordinalMgerman}[2]{%
1443 \edef#2{\number#1\relax.}}

```

Feminine:

```

1444 \newcommand{\@ordinalFgerman}[2]{%
1445 \edef#2{\number#1\relax.}}

```

Neuter:

```

1446 \newcommand{\@ordinalNgerman}[2]{%
1447 \edef#2{\number#1\relax.}}

```

Convert a number to text. The easiest way to do this is to break it up into units, tens and teens. Units (argument must be a number from 0 to 9, 1 on its own (eins) is dealt with separately):

```

1448 \newcommand{\@@unitstringgerman}[1]{%
1449 \ifcase#1%
1450 null%
1451 \or ein%
1452 \or zwei%
1453 \or drei%
1454 \or vier%
1455 \or f\"unf%
1456 \or sechs%
1457 \or sieben%
1458 \or acht%
1459 \or neun%
1460 \fi
1461 }

```

Tens (argument must go from 1 to 10):

```

1462 \newcommand{\@@tenstringgerman}[1]{%
1463 \ifcase#1%
1464 \or zehn%
1465 \or zwanzig%
1466 \or drei{\ss}ig%
1467 \or vierzig%
1468 \or f\"unfzig%
1469 \or sechzig%
1470 \or siebzig%
1471 \or achtzig%
1472 \or neunzig%
1473 \or einhundert%
1474 \fi
1475 }

```

\einhundert is set to einhundert by default, user can redefine this command to just hundert if required, similarly for \eintausend.

```

1476 \providecommand*\einhundert{\einhundert}
1477 \providecommand*\eintausend{\eintausend}

```

Teens:

```

1478 \newcommand{\@@teenstringgerman}[1]{%
1479 \ifcase#1%
1480 zehn%
1481 \or elf%
1482 \or zw\"olf%
1483 \or dreizehn%
1484 \or vierzehn%
1485 \or f\"unfzehn%
1486 \or sechzehn%
1487 \or siebzehn%
1488 \or achtzehn%
1489 \or neunzehn%
1490 \fi
1491 }

```

The results are stored in the second argument, but doesn't display anything.

```

1492 \DeclareRobustCommand{\@numberstringMgerman}[2]{%
1493 \let\@unitstring=\@unitstringgerman
1494 \let\@teenstring=\@teenstringgerman

```

```

1495 \let\@tenstring=\@tenstringgerman
1496 \@numberstringgerman{#1}{#2}}

Feminine and neuter forms:
1497 \let\@numberstringFgerman=\@numberstringMgerman
1498 \let\@numberstringNgerman=\@numberstringMgerman

As above, but initial letters in upper case:
1499 \DeclareRobustCommand{\@NumberstringMgerman}[2]{%
1500 \@numberstringMgerman{#1}{\@num@str}%
1501 \edef#2{\noexpand\MakeUppercase\@num@str}}

Feminine and neuter form:
1502 \let\@NumberstringFgerman=\@NumberstringMgerman
1503 \let\@NumberstringNgerman=\@NumberstringMgerman

As above, but for ordinals.
1504 \DeclareRobustCommand{\@OrdinalstringMgerman}[2]{%
1505 \let\@unitthstring=\@unitthstringMgerman
1506 \let\@teenthstring=\@teenthstringMgerman
1507 \let\@tenthstring=\@tenthstringMgerman
1508 \let\@unitstring=\@unitstringgerman
1509 \let\@teenstring=\@teenstringgerman
1510 \let\@tenstring=\@tenstringgerman
1511 \def\@thousandth{tausendster}%
1512 \def\@hundredth{hundertster}%
1513 \@ordinalstringgerman{#1}{#2}}

Feminine form:
1514 \DeclareRobustCommand{\@OrdinalstringFgerman}[2]{%
1515 \let\@unitthstring=\@unitthstringFgerman
1516 \let\@teenthstring=\@teenthstringFgerman
1517 \let\@tenthstring=\@tenthstringFgerman
1518 \let\@unitstring=\@unitstringgerman
1519 \let\@teenstring=\@teenstringgerman
1520 \let\@tenstring=\@tenstringgerman
1521 \def\@thousandth{tausendste}%
1522 \def\@hundredth{hundertste}%
1523 \@ordinalstringgerman{#1}{#2}}

Neuter form:
1524 \DeclareRobustCommand{\@OrdinalstringNgerman}[2]{%
1525 \let\@unitthstring=\@unitthstringNgerman
1526 \let\@teenthstring=\@teenthstringNgerman
1527 \let\@tenthstring=\@tenthstringNgerman
1528 \let\@unitstring=\@unitstringgerman
1529 \let\@teenstring=\@teenstringgerman
1530 \let\@tenstring=\@tenstringgerman
1531 \def\@thousandth{tausendstes}%
1532 \def\@hundredth{hunderstes}%
1533 \@ordinalstringgerman{#1}{#2}}

As above, but with initial letters in upper case.
1534 \DeclareRobustCommand{\@OrdinalstringMgerman}[2]{%
1535 \@ordinalstringMgerman{#1}{\@num@str}%
1536 \edef#2{\protect\MakeUppercase\@num@str}}

```

Feminine form:

```
1537 \DeclareRobustCommand{\@OrdinalstringFgerman}[2]{%
1538 \@OrdinalstringFgerman{#1}{\@num@str}%
1539 \edef#2{\protect\MakeUppercase\@num@str}}
```

Neuter form:

```
1540 \DeclareRobustCommand{\@OrdinalstringNgerman}[2]{%
1541 \@OrdinalstringNgerman{#1}{\@num@str}%
1542 \edef#2{\protect\MakeUppercase\@num@str}}
```

Code for converting numbers into textual ordinals. As before, it is easier to split it into units, tens and teens. Units:

```
1543 \newcommand{\@unitthstringMgerman}[1]{%
1544 \ifcase#1%
1545 nullter%
1546 \or erster%
1547 \or zweiter%
1548 \or dritter%
1549 \or vierter%
1550 \or f\"unter%
1551 \or sechster%
1552 \or siebter%
1553 \or achter%
1554 \or neunter%
1555 \fi
1556 }
```

Tens:

```
1557 \newcommand{\@tenthstringMgerman}[1]{%
1558 \ifcase#1%
1559 \or zehnter%
1560 \or zwanzigster%
1561 \or drei{\ss}zigster%
1562 \or vierzigster%
1563 \or f\"unfzigster%
1564 \or sechzigster%
1565 \or siebzigster%
1566 \or achtzigster%
1567 \or neunzigster%
1568 \fi
1569 }
```

Teens:

```
1570 \newcommand{\@teenthstringMgerman}[1]{%
1571 \ifcase#1%
1572 zehnter%
1573 \or elfter%
1574 \or zw\"olfter%
1575 \or dreizehnter%
1576 \or vierzehnter%
1577 \or f\"unfzehnter%
1578 \or sechzehnter%
1579 \or siebzehnter%
1580 \or achtzehnter%
1581 \or neunzehnter%
```

```

1582 \fi
1583 }

Units (feminine):
1584 \newcommand{\@@unitthstringFgerman}[1]{%
1585 \ifcase#1%
1586 nullte%
1587 \or erste%
1588 \or zweite%
1589 \or dritte%
1590 \or vierte%
1591 \or f\ "unfte%
1592 \or sechste%
1593 \or siebte%
1594 \or achte%
1595 \or neunte%
1596 \fi
1597 }

Tens (feminine):
1598 \newcommand{\@@tenthstringFgerman}[1]{%
1599 \ifcase#1%
1600 \or zehnte%
1601 \or zwanzigste%
1602 \or drei{\ss}igste%
1603 \or vierzigste%
1604 \or f\ "unfzigste%
1605 \or sechzigste%
1606 \or siebzigste%
1607 \or achtzigste%
1608 \or neunzigste%
1609 \fi
1610 }

Teens (feminine)
1611 \newcommand{\@@teenthstringFgerman}[1]{%
1612 \ifcase#1%
1613 zehnte%
1614 \or elfte%
1615 \or zw\ "olfte%
1616 \or dreizehnte%
1617 \or vierzehnte%
1618 \or f\ "unfzehnte%
1619 \or sechzehnte%
1620 \or siebzehnte%
1621 \or achtzehnte%
1622 \or neunzehnte%
1623 \fi
1624 }

Units (neuter):
1625 \newcommand{\@@unitthstringNgerman}[1]{%
1626 \ifcase#1%
1627 nulltes%
1628 \or erstes%
1629 \or zweites%

```



```

1630 \or drittes%
1631 \or viertes%
1632 \or f\"unte%
1633 \or sechstes%
1634 \or siebtes%
1635 \or achtes%
1636 \or neuntes%
1637 \fi
1638 }

```

Tens (neuter):

```

1639 \newcommand{\@@tenthstringNgerman}[1]{%
1640 \ifcase#1%
1641 \or zehntes%
1642 \or zwanzigstes%
1643 \or drei{\ss}igstes%
1644 \or vierzigstes%
1645 \or f\"unfzigstes%
1646 \or sechzigstes%
1647 \or siebzigstes%
1648 \or achtzigstes%
1649 \or neunzigstes%
1650 \fi
1651 }

```

Teens (neuter)

```

1652 \newcommand{\@@teenthstringNgerman}[1]{%
1653 \ifcase#1%
1654 zehntes%
1655 \or elftes%
1656 \or zw\"olftes%
1657 \or dreizehntes%
1658 \or vierzehntes%
1659 \or f\"unfzehntes%
1660 \or sechzehntes%
1661 \or siebzehntes%
1662 \or achtzehntes%
1663 \or neunzehntes%
1664 \fi
1665 }

```

This appends the results to #2 for number #2 (in range 0 to 100.) null and eins are dealt with separately in \@@numberstringgerman.

```

1666 \newcommand{\@@numberunderhundredgerman}[2]{%
1667 \ifnum#1<10\relax
1668 \ifnum#1>0\relax
1669 \let\@@fc@numstr#2\relax
1670 \edef#2{\@@fc@numstr\@unitstring{#1}}%
1671 \fi
1672 \else
1673 \@tmpstrctr=#1\relax
1674 \@modulo{\@tmpstrctr}{10}%
1675 \ifnum#1<20\relax
1676 \let\@@fc@numstr#2\relax
1677 \edef#2{\@@fc@numstr\@teenstring{\@tmpstrctr}}%

```

```

1678 \else
1679 \ifnum \@tmpstrctr=0\relax
1680 \else
1681 \let \@fc@numstr#2\relax
1682 \edef#2{\@fc@numstr\@unitstring{\@tmpstrctr}und}%
1683 \fi
1684 \@tmpstrctr=#1\relax
1685 \divide \@tmpstrctr by 10\relax
1686 \let \@fc@numstr#2\relax
1687 \edef#2{\@fc@numstr\@tenstring{\@tmpstrctr}}%
1688 \fi
1689 \fi
1690 }

```

This stores the results in the second argument (which must be a control sequence), but it doesn't display anything.

```

1691 \newcommand{\@numberstringgerman}[2]{%
1692 \ifnum#1>99999\relax
1693 \PackageError{fmtcount}{Out of range}%
1694 {This macro only works for values less than 100000}%
1695 \else
1696 \ifnum#1<0\relax
1697 \PackageError{fmtcount}{Negative numbers not permitted}%
1698 {This macro does not work for negative numbers, however
1699 you can try typing "minus" first, and then pass the modulus of
1700 this number}%
1701 \fi
1702 \fi
1703 \def#2{%
1704 \@strctr=#1\relax \divide \@strctr by 1000\relax
1705 \ifnum \@strctr>1\relax
1706 % #1 is >= 2000, \@strctr now contains the number of thousands
1707 \@numberunderhundredgerman{\@strctr}{#2}%
1708 \let \@fc@numstr#2\relax
1709 \edef#2{\@fc@numstr tausend}%
1710 \else
1711 % #1 lies in range [1000,1999]
1712 \ifnum \@strctr=1\relax
1713 \let \@fc@numstr#2\relax
1714 \edef#2{\@fc@numstr eintausend}%
1715 \fi
1716 \fi
1717 \@strctr=#1\relax
1718 \@modulo{\@strctr}{1000}%
1719 \divide \@strctr by 100\relax
1720 \ifnum \@strctr>1\relax
1721 % now dealing with number in range [200,999]
1722 \let \@fc@numstr#2\relax
1723 \edef#2{\@fc@numstr\@unitstring{\@strctr}hundert}%
1724 \else
1725 \ifnum \@strctr=1\relax
1726 % dealing with number in range [100,199]
1727 \ifnum#1>1000\relax
1728 % if original number > 1000, use einhundert

```

```

1729         \let\@@fc@numstr#2\relax
1730         \edef#2{\@@fc@numstr einhundert}%
1731     \else
1732 % otherwise use \einhundert
1733         \let\@@fc@numstr#2\relax
1734         \edef#2{\@@fc@numstr\einhundert}%
1735     \fi
1736 \fi
1737 \fi
1738 \@strctr=#1\relax
1739 \@modulo{\@strctr}{100}%
1740 \ifnum#1=0\relax
1741     \def#2{null}%
1742 \else
1743     \ifnum\@strctr=1\relax
1744         \let\@@fc@numstr#2\relax
1745         \edef#2{\@@fc@numstr eins}%
1746     \else
1747         \@numberunderhundredgerman{\@strctr}{#2}%
1748     \fi
1749 \fi
1750 }

    As above, but for ordinals

1751 \newcommand{\@@numberunderhundredthgerman}[2]{%
1752 \ifnum#1<10\relax
1753     \let\@@fc@numstr#2\relax
1754     \edef#2{\@@fc@numstr\@unitthstring{#1}}%
1755 \else
1756     \@tmpstrctr=#1\relax
1757     \@modulo{\@tmpstrctr}{10}%
1758     \ifnum#1<20\relax
1759         \let\@@fc@numstr#2\relax
1760         \edef#2{\@@fc@numstr\@teenthstring{\@tmpstrctr}}%
1761     \else
1762         \ifnum\@tmpstrctr=0\relax
1763             \else
1764                 \let\@@fc@numstr#2\relax
1765                 \edef#2{\@@fc@numstr\@unitstring{\@tmpstrctr}und}%
1766             \fi
1767             \@tmpstrctr=#1\relax
1768             \divide\@tmpstrctr by 10\relax
1769             \let\@@fc@numstr#2\relax
1770             \edef#2{\@@fc@numstr\@tenthstring{\@tmpstrctr}}%
1771         \fi
1772 \fi
1773 }

1774 \newcommand{\@@ordinalstringgerman}[2]{%
1775 \ifnum#1>99999\relax
1776     \PackageError{fmtcount}{Out of range}%
1777     {This macro only works for values less than 100000}%
1778 \else
1779     \ifnum#1<0\relax
1780         \PackageError{fmtcount}{Negative numbers not permitted}%

```

```

1781     {This macro does not work for negative numbers, however
1782     you can try typing "minus" first, and then pass the modulus of
1783     this number}%
1784 \fi
1785 \fi
1786 \def#2{}%
1787 \@strctr=#1\relax \divide\@strctr by 1000\relax
1788 \ifnum\@strctr>1\relax
1789 % #1 is >= 2000, \@strctr now contains the number of thousands
1790 \@numberunderhundredgerman{\@strctr}{#2}%
1791 \let\@fc@numstr#2\relax
1792 % is that it, or is there more?
1793 \@tmpstrctr=#1\relax \@modulo{\@tmpstrctr}{1000}%
1794 \ifnum\@tmpstrctr=0\relax
1795 \edef#2{\@fc@numstr\@thousandth}%
1796 \else
1797 \edef#2{\@fc@numstr tausend}%
1798 \fi
1799 \else
1800 % #1 lies in range [1000,1999]
1801 \ifnum\@strctr=1\relax
1802 \ifnum#1=1000\relax
1803 \let\@fc@numstr#2\relax
1804 \edef#2{\@fc@numstr\@thousandth}%
1805 \else
1806 \let\@fc@numstr#2\relax
1807 \edef#2{\@fc@numstr\eintausend}%
1808 \fi
1809 \fi
1810 \fi
1811 \@strctr=#1\relax
1812 \@modulo{\@strctr}{1000}%
1813 \divide\@strctr by 100\relax
1814 \ifnum\@strctr>1\relax
1815 % now dealing with number in range [200,999]
1816 \let\@fc@numstr#2\relax
1817 % is that it, or is there more?
1818 \@tmpstrctr=#1\relax \@modulo{\@tmpstrctr}{100}%
1819 \ifnum\@tmpstrctr=0\relax
1820 \ifnum\@strctr=1\relax
1821 \edef#2{\@fc@numstr\@hundredth}%
1822 \else
1823 \edef#2{\@fc@numstr\@unitstring{\@strctr}\@hundredth}%
1824 \fi
1825 \else
1826 \edef#2{\@fc@numstr\@unitstring{\@strctr}hundert}%
1827 \fi
1828 \else
1829 \ifnum\@strctr=1\relax
1830 % dealing with number in range [100,199]
1831 % is that it, or is there more?
1832 \@tmpstrctr=#1\relax \@modulo{\@tmpstrctr}{100}%
1833 \ifnum\@tmpstrctr=0\relax
1834 \let\@fc@numstr#2\relax

```

```

1835     \edef#2{\@fc@numstr\@hundredth}%
1836     \else
1837     \ifnum#1>1000\relax
1838     \let\@fc@numstr#2\relax
1839     \edef#2{\@fc@numstr einhundert}%
1840     \else
1841     \let\@fc@numstr#2\relax
1842     \edef#2{\@fc@numstr\@hundredth}%
1843     \fi
1844     \fi
1845     \fi
1846 \fi
1847 \@strctr=#1\relax
1848 \@modulo{\@strctr}{100}%
1849 \ifthenelse{\@strctr=0 \and #1>0}{-}{%
1850 \@numberunderhundredthgerman{\@strctr}{#2}%
1851 }%
1852 }

```

Set `ngerman` to be equivalent to `german`. Is it okay to do this? (I don't know the difference between the two.)

```

1853 \let\@ordinalMngerman=\@ordinalMgerman
1854 \let\@ordinalFngerman=\@ordinalFgerman
1855 \let\@ordinalNngerman=\@ordinalNgerman
1856 \let\@numberstringMngerman=\@numberstringMgerman
1857 \let\@numberstringFngerman=\@numberstringFgerman
1858 \let\@numberstringNngerman=\@numberstringNgerman
1859 \let\@NumberstringMngerman=\@NumberstringMgerman
1860 \let\@NumberstringFngerman=\@NumberstringFgerman
1861 \let\@NumberstringNngerman=\@NumberstringNgerman
1862 \let\@ordinalstringMngerman=\@ordinalstringMgerman
1863 \let\@ordinalstringFngerman=\@ordinalstringFgerman
1864 \let\@ordinalstringNngerman=\@ordinalstringNgerman
1865 \let\@OrdinalstringMngerman=\@OrdinalstringMgerman
1866 \let\@OrdinalstringFngerman=\@OrdinalstringFgerman
1867 \let\@OrdinalstringNngerman=\@OrdinalstringNgerman

```

11.5 fc-portuges.def

Portuguese definitions

```

1868 \ProvidesFile{fc-portuges.def}[2007/05/26]

```

Define macro that converts a number or count register (first argument) to an ordinal, and stores the result in the second argument, which should be a control sequence. Masculine:

```

1869 \newcommand*{\@ordinalMportuges}[2]{%
1870 \ifnum#1=0\relax
1871     \edef#2{\number#1}%
1872 \else
1873     \edef#2{\number#1\relax\noexpand\fmtord{o}}%
1874 \fi}

```

Feminine:

```

1875 \newcommand*{\@ordinalFportuges}[2]{%

```

```

1876 \ifnum#1=0\relax
1877   \edef#2{\number#1}%
1878 \else
1879   \edef#2{\number#1\relax\noexpand\fmtord{a}}%
1880 \fi}

```

Make neuter same as masculine:

```

1881 \let\@ordinalNportuges\@ordinalMportuges

```

Convert a number to a textual representation. To make it easier, split it up into units, tens, teens and hundreds. Units (argument must be a number from 0 to 9):

```

1882 \newcommand*{\@unitstringportuges}[1]{%
1883 \ifcase#1\relax
1884 zero%
1885 \or um%
1886 \or dois%
1887 \or tr\`es%
1888 \or quatro%
1889 \or cinco%
1890 \or seis%
1891 \or sete%
1892 \or oito%
1893 \or nove%
1894 \fi
1895 }
1896 %   \end{macrocode}
1897 % As above, but for feminine:
1898 %   \begin{macrocode}
1899 \newcommand*{\@unitstringFportuges}[1]{%
1900 \ifcase#1\relax
1901 zero%
1902 \or uma%
1903 \or duas%
1904 \or tr\`es%
1905 \or quatro%
1906 \or cinco%
1907 \or seis%
1908 \or sete%
1909 \or oito%
1910 \or nove%
1911 \fi
1912 }

```

Tens (argument must be a number from 0 to 10):

```

1913 \newcommand*{\@tenstringportuges}[1]{%
1914 \ifcase#1\relax
1915 \or dez%
1916 \or vinte%
1917 \or trinta%
1918 \or quarenta%
1919 \or cinq\"uenta%
1920 \or sessenta%
1921 \or setenta%
1922 \or oitenta%
1923 \or noventa%

```

```

1924 \or cem%
1925 \fi
1926 }

```

Teens (argument must be a number from 0 to 9):

```

1927 \newcommand*{\@@teenstringportuges}[1]{%
1928 \ifcase#1\relax
1929 dez%
1930 \or onze%
1931 \or doze%
1932 \or treze%
1933 \or quatorze%
1934 \or quinze%
1935 \or dezesesseis%
1936 \or dezessete%
1937 \or dezoito%
1938 \or dezenove%
1939 \fi
1940 }

```

Hundreds:

```

1941 \newcommand*{\@@hundredstringportuges}[1]{%
1942 \ifcase#1\relax
1943 \or cento%
1944 \or duzentos%
1945 \or trezentos%
1946 \or quatrocentos%
1947 \or quinhentos%
1948 \or seiscentos%
1949 \or setecentos%
1950 \or oitocentos%
1951 \or novecentos%
1952 \fi}

```

Hundreds (feminine):

```

1953 \newcommand*{\@@hundredstringFportuges}[1]{%
1954 \ifcase#1\relax
1955 \or cento%
1956 \or duzentas%
1957 \or trezentas%
1958 \or quatrocentas%
1959 \or quinhentas%
1960 \or seiscentas%
1961 \or setecentas%
1962 \or oitocentas%
1963 \or novecentas%
1964 \fi}

```

Units (initial letter in upper case):

```

1965 \newcommand*{\@@Unitstringportuges}[1]{%
1966 \ifcase#1\relax
1967 Zero%
1968 \or Um%
1969 \or Dois%
1970 \or Tr\`es%
1971 \or Quatro%

```

```

1972 \or Cinco%
1973 \or Seis%
1974 \or Sete%
1975 \or Oito%
1976 \or Nove%
1977 \fi
1978 }

```

As above, but feminine:

```

1979 \newcommand*{\@@UnitstringFportuges}[1]{%
1980 \ifcase#1\relax
1981 Zera%
1982 \or Uma%
1983 \or Duas%
1984 \or Tr\`es%
1985 \or Quatro%
1986 \or Cinco%
1987 \or Seis%
1988 \or Sete%
1989 \or Oito%
1990 \or Nove%
1991 \fi
1992 }

```

Tens (with initial letter in upper case):

```

1993 \newcommand*{\@@Tenstringportuges}[1]{%
1994 \ifcase#1\relax
1995 \or Dez%
1996 \or Vinte%
1997 \or Trinta%
1998 \or Quarenta%
1999 \or Cinq\"uenta%
2000 \or Sessenta%
2001 \or Setenta%
2002 \or Oitenta%
2003 \or Noventa%
2004 \or Cem%
2005 \fi
2006 }

```

Teens (with initial letter in upper case):

```

2007 \newcommand*{\@@Teenstringportuges}[1]{%
2008 \ifcase#1\relax
2009 Dez%
2010 \or Onze%
2011 \or Doze%
2012 \or Treze%
2013 \or Quatorze%
2014 \or Quinze%
2015 \or Dezesseis%
2016 \or Dezessete%
2017 \or Dezoito%
2018 \or Dezenove%
2019 \fi
2020 }

```


Hundreds (with initial letter in upper case):

```
2021 \newcommand*{\@@Hundredstringportuges}[1]{%
2022 \ifcase#1\relax
2023 \or Cento%
2024 \or Duzentos%
2025 \or Trezentos%
2026 \or Quatrocentos%
2027 \or Quinhentos%
2028 \or Seiscentos%
2029 \or Setecentos%
2030 \or Oitocentos%
2031 \or Novecentos%
2032 \fi}
```

As above, but feminine:

```
2033 \newcommand*{\@@HundredstringFportuges}[1]{%
2034 \ifcase#1\relax
2035 \or Cento%
2036 \or Duzentas%
2037 \or Trezentas%
2038 \or Quatrocentas%
2039 \or Quinhentas%
2040 \or Seiscentas%
2041 \or Setecentas%
2042 \or Oitocentas%
2043 \or Novecentas%
2044 \fi}
```

This has changed in version 1.08, so that it now stores the result in the second argument, but doesn't display anything. Since it only affects internal macros, it shouldn't affect documents created with older versions. (These internal macros are not meant for use in documents.)

```
2045 \DeclareRobustCommand{\@numberstringMportuges}[2]{%
2046 \let\@unitstring=\@unitstringportuges
2047 \let\@teenstring=\@teenstringportuges
2048 \let\@tenstring=\@tenstringportuges
2049 \let\@hundredstring=\@hundredstringportuges
2050 \def\@hundred{cem}\def\@thousand{mil}%
2051 \def\@andname{e}%
2052 \@numberstringportuges{#1}{#2}}
```

As above, but feminine form:

```
2053 \DeclareRobustCommand{\@numberstringFportuges}[2]{%
2054 \let\@unitstring=\@unitstringFportuges
2055 \let\@teenstring=\@teenstringportuges
2056 \let\@tenstring=\@tenstringportuges
2057 \let\@hundredstring=\@hundredstringFportuges
2058 \def\@hundred{cem}\def\@thousand{mil}%
2059 \def\@andname{e}%
2060 \@numberstringportuges{#1}{#2}}
```

Make neuter same as masculine:

```
2061 \let\@numberstringNportuges\@numberstringMportuges
```

As above, but initial letters in upper case:

```
2062 \DeclareRobustCommand{\@NumberstringMportuges}[2]{%
```

```

2063 \let\@unitstring=\@Unitstringportuges
2064 \let\@teenstring=\@Teenstringportuges
2065 \let\@tenstring=\@Tenstringportuges
2066 \let\@hundredstring=\@Hundredstringportuges
2067 \def\@hundred{Cem}\def\@thousand{Mil}%
2068 \def\@andname{e}%
2069 \@numberstringportuges{#1}{#2}}

```

As above, but feminine form:

```

2070 \DeclareRobustCommand{\@NumberstringFportuges}[2]{%
2071 \let\@unitstring=\@UnitstringFportuges
2072 \let\@teenstring=\@Teenstringportuges
2073 \let\@tenstring=\@Tenstringportuges
2074 \let\@hundredstring=\@HundredstringFportuges
2075 \def\@hundred{Cem}\def\@thousand{Mil}%
2076 \def\@andname{e}%
2077 \@numberstringportuges{#1}{#2}}

```

Make neuter same as masculine:

```

2078 \let\@NumberstringNportuges\@NumberstringMportuges

```

As above, but for ordinals.

```

2079 \DeclareRobustCommand{\@OrdinalstringMportuges}[2]{%
2080 \let\@unitthstring=\@Unitthstringportuges
2081 \let\@unitstring=\@Unitstringportuges
2082 \let\@teenthstring=\@Teenthstringportuges
2083 \let\@tenthstring=\@Tenthstringportuges
2084 \let\@hundredthstring=\@Hundredthstringportuges
2085 \def\@thousandth{mil\'esimo}%
2086 \@ordinalstringportuges{#1}{#2}}

```

Feminine form:

```

2087 \DeclareRobustCommand{\@OrdinalstringFportuges}[2]{%
2088 \let\@unitthstring=\@UnitthstringFportuges
2089 \let\@unitstring=\@UnitstringFportuges
2090 \let\@teenthstring=\@Teenthstringportuges
2091 \let\@tenthstring=\@TenthstringFportuges
2092 \let\@hundredthstring=\@HundredthstringFportuges
2093 \def\@thousandth{mil\'esima}%
2094 \@ordinalstringportuges{#1}{#2}}

```

Make neuter same as masculine:

```

2095 \let\@OrdinalstringNportuges\@OrdinalstringMportuges

```

As above, but initial letters in upper case (masculine):

```

2096 \DeclareRobustCommand{\@OrdinalstringMportuges}[2]{%
2097 \let\@unitthstring=\@Unitthstringportuges
2098 \let\@unitstring=\@Unitstringportuges
2099 \let\@teenthstring=\@Teenthstringportuges
2100 \let\@tenthstring=\@Tenthstringportuges
2101 \let\@hundredthstring=\@Hundredthstringportuges
2102 \def\@thousandth{Mil\'esimo}%
2103 \@ordinalstringportuges{#1}{#2}}

```

Feminine form:

```

2104 \DeclareRobustCommand{\@OrdinalstringFportuges}[2]{%
2105 \let\@unitthstring=\@UnitthstringFportuges

```

```

2106 \let\@unitstring=\@UnitstringFportuges
2107 \let\@teenthstring=\@teenthstringportuges
2108 \let\@tenthstring=\@TenthstringFportuges
2109 \let\@hundredthstring=\@HundredthstringFportuges
2110 \def\@thousandth{Mil\'esima}%
2111 \@ordinalstringportuges{#1}{#2}}

```

Make neuter same as masculine:

```

2112 \let\@OrdinalstringNportuges\@OrdinalstringMportuges

```

In order to do the ordinals, split into units, teens, tens and hundreds. Units:

```

2113 \newcommand*{\@unitthstringportuges}[1]{%
2114 \ifcase#1\relax
2115 zero%
2116 \or primeiro%
2117 \or segundo%
2118 \or terceiro%
2119 \or quarto%
2120 \or quinto%
2121 \or sexto%
2122 \or s\'etimo%
2123 \or oitavo%
2124 \or nono%
2125 \fi
2126 }

```

Tens:

```

2127 \newcommand*{\@tenthstringportuges}[1]{%
2128 \ifcase#1\relax
2129 \or d\'ecimo%
2130 \or vig\'esimo%
2131 \or trig\'esimo%
2132 \or quadrag\'esimo%
2133 \or q\'uinquag\'esimo%
2134 \or sexag\'esimo%
2135 \or setuag\'esimo%
2136 \or octog\'esimo%
2137 \or nonag\'esimo%
2138 \fi
2139 }

```

Teens:

```

2140 \newcommand*{\@teenthstringportuges}[1]{%
2141 \@tenthstring{1}%
2142 \ifnum#1>0\relax
2143 -\@unitthstring{#1}%
2144 \fi}

```

Hundreds:

```

2145 \newcommand*{\@hundredthstringportuges}[1]{%
2146 \ifcase#1\relax
2147 \or cent\'esimo%
2148 \or ducent\'esimo%
2149 \or trecent\'esimo%
2150 \or quadringent\'esimo%
2151 \or q\'uingent\'esimo%

```

```

2152 \or seiscient\'esimo%
2153 \or setingent\'esimo%
2154 \or octingent\'esimo%
2155 \or nongent\'esimo%
2156 \fi}

```

Units (feminine):

```

2157 \newcommand*{\@@unitthstringFportuges}[1]{%
2158 \ifcase#1\relax
2159 zero%
2160 \or primeira%
2161 \or segunda%
2162 \or terceira%
2163 \or quarta%
2164 \or quinta%
2165 \or sexta%
2166 \or s\'etima%
2167 \or oitava%
2168 \or nona%
2169 \fi
2170 }

```

Tens (feminine):

```

2171 \newcommand*{\@@tenthstringFportuges}[1]{%
2172 \ifcase#1\relax
2173 \or d\'ecima%
2174 \or vig\'esima%
2175 \or trig\'esima%
2176 \or quadrag\'esima%
2177 \or q\'uinquag\'esima%
2178 \or sexag\'esima%
2179 \or setuag\'esima%
2180 \or octog\'esima%
2181 \or nonag\'esima%
2182 \fi
2183 }

```

Hundreds (feminine):

```

2184 \newcommand*{\@@hundredthstringFportuges}[1]{%
2185 \ifcase#1\relax
2186 \or cent\'esima%
2187 \or ducent\'esima%
2188 \or trecent\'esima%
2189 \or quadringent\'esima%
2190 \or q\'uingent\'esima%
2191 \or seiscient\'esima%
2192 \or setingent\'esima%
2193 \or octingent\'esima%
2194 \or nongent\'esima%
2195 \fi}

```

As above, but with initial letter in upper case. Units:

```

2196 \newcommand*{\@@Unitthstringportuges}[1]{%
2197 \ifcase#1\relax
2198 Zero%
2199 \or Primeiro%

```

```

2200 \or Segundo%
2201 \or Terceiro%
2202 \or Quarto%
2203 \or Quinto%
2204 \or Sexto%
2205 \or S\'etimo%
2206 \or Oitavo%
2207 \or Nono%
2208 \fi
2209 }

```

Tens:

```

2210 \newcommand*{\@@Tenthstringportuges}[1]{%
2211 \ifcase#1\relax
2212 \or D\'ecimo%
2213 \or Vig\'esimo%
2214 \or Trig\'esimo%
2215 \or Quadrag\'esimo%
2216 \or Q\'uinquag\'esimo%
2217 \or Sexag\'esimo%
2218 \or Setuag\'esimo%
2219 \or Octog\'esimo%
2220 \or Nonag\'esimo%
2221 \fi
2222 }

```

Hundreds:

```

2223 \newcommand*{\@@Hundredthstringportuges}[1]{%
2224 \ifcase#1\relax
2225 \or Cent\'esimo%
2226 \or Ducent\'esimo%
2227 \or Trecent\'esimo%
2228 \or Quadringent\'esimo%
2229 \or Q\'uingent\'esimo%
2230 \or Seiscent\'esimo%
2231 \or Setingent\'esimo%
2232 \or Octingent\'esimo%
2233 \or Nongent\'esimo%
2234 \fi}

```

As above, but feminine. Units:

```

2235 \newcommand*{\@@UnitthstringFportuges}[1]{%
2236 \ifcase#1\relax
2237 Zera%
2238 \or Primeira%
2239 \or Segunda%
2240 \or Terceira%
2241 \or Quarta%
2242 \or Quinta%
2243 \or Sexta%
2244 \or S\'etima%
2245 \or Oitava%
2246 \or Nona%
2247 \fi
2248 }

```

Tens (feminine);

```

2249 \newcommand*{\@@TenthstringFportuges}[1]{%
2250 \ifcase#1\relax
2251 \or D\'ecima%
2252 \or Vig\'esima%
2253 \or Trig\'esima%
2254 \or Quadrag\'esima%
2255 \or Q\'uinquag\'esima%
2256 \or Sexag\'esima%
2257 \or Setuag\'esima%
2258 \or Octog\'esima%
2259 \or Nonag\'esima%
2260 \fi
2261 }

```

Hundreds (feminine):

```

2262 \newcommand*{\@@HundredthstringFportuges}[1]{%
2263 \ifcase#1\relax
2264 \or Cent\'esima%
2265 \or Ducent\'esima%
2266 \or Trecent\'esima%
2267 \or Quadringent\'esima%
2268 \or Q\'uingent\'esima%
2269 \or Seiscent\'esima%
2270 \or Setingent\'esima%
2271 \or Octingent\'esima%
2272 \or Nongent\'esima%
2273 \fi}

```

This has changed in version 1.09, so that it now stores the result in the second argument (a control sequence), but it doesn't display anything. Since it only affects internal macros, it shouldn't affect documents created with older versions. (These internal macros are not meant for use in documents.)

```

2274 \newcommand*{\@@numberstringportuges}[2]{%
2275 \ifnum#1>99999
2276 \PackageError{fmtcount}{Out of range}%
2277 {This macro only works for values less than 100000}%
2278 \else
2279 \ifnum#1<0
2280 \PackageError{fmtcount}{Negative numbers not permitted}%
2281 {This macro does not work for negative numbers, however
2282 you can try typing "minus" first, and then pass the modulus of
2283 this number}%
2284 \fi
2285 \fi
2286 \def#2{}%
2287 \@strctr=#1\relax \divide\@strctr by 1000\relax
2288 \ifnum\@strctr>9
2289 % #1 is greater or equal to 10000
2290 \divide\@strctr by 10
2291 \ifnum\@strctr>1\relax
2292 \let\@fc@numstr#2\relax
2293 \edef#2{\@fc@numstr\@tenstring{\@strctr}}%
2294 \@strctr=#1 \divide\@strctr by 1000\relax

```

```

2295 \@modulo{\@strctr}{10}%
2296 \ifnum\@strctr>0
2297 \ifnum\@strctr=1\relax
2298 \let\@@fc@numstr#2\relax
2299 \edef#2{\@@fc@numstr\ \@andname}%
2300 \fi
2301 \let\@@fc@numstr#2\relax
2302 \edef#2{\@@fc@numstr\ \@unitstring{\@strctr}}%
2303 \fi
2304 \else
2305 \@strctr=#1\relax
2306 \divide\@strctr by 1000\relax
2307 \@modulo{\@strctr}{10}%
2308 \let\@@fc@numstr#2\relax
2309 \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
2310 \fi
2311 \let\@@fc@numstr#2\relax
2312 \edef#2{\@@fc@numstr\ \@thousand}%
2313 \else
2314 \ifnum\@strctr>0\relax
2315 \ifnum\@strctr>1\relax
2316 \let\@@fc@numstr#2\relax
2317 \edef#2{\@@fc@numstr\@unitstring{\@strctr}\ }%
2318 \fi
2319 \let\@@fc@numstr#2\relax
2320 \edef#2{\@@fc@numstr\@thousand}%
2321 \fi
2322 \fi
2323 \@strctr=#1\relax \@modulo{\@strctr}{1000}%
2324 \divide\@strctr by 100\relax
2325 \ifnum\@strctr>0\relax
2326 \ifnum#1>1000 \relax
2327 \let\@@fc@numstr#2\relax
2328 \edef#2{\@@fc@numstr\ }%
2329 \fi
2330 \@tmpstrctr=#1\relax
2331 \@modulo{\@tmpstrctr}{1000}%
2332 \let\@@fc@numstr#2\relax
2333 \ifnum\@tmpstrctr=100\relax
2334 \edef#2{\@@fc@numstr\@tenstring{10}}%
2335 \else
2336 \edef#2{\@@fc@numstr\@hundredstring{\@strctr}}%
2337 \fi%
2338 \fi
2339 \@strctr=#1\relax \@modulo{\@strctr}{100}%
2340 \ifnum#1>100\relax
2341 \ifnum\@strctr>0\relax
2342 \let\@@fc@numstr#2\relax
2343 \edef#2{\@@fc@numstr\ \@andname\ }%
2344 \fi
2345 \fi
2346 \ifnum\@strctr>19\relax
2347 \divide\@strctr by 10\relax
2348 \let\@@fc@numstr#2\relax

```

```

2349 \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
2350 \@strctr=#1\relax \@modulo{\@strctr}{10}%
2351 \ifnum\@strctr>0
2352   \ifnum\@strctr=1\relax
2353     \let\@@fc@numstr#2\relax
2354     \edef#2{\@@fc@numstr\ \@andname}%
2355   \else
2356     \ifnum#1>100\relax
2357       \let\@@fc@numstr#2\relax
2358       \edef#2{\@@fc@numstr\ \@andname}%
2359     \fi
2360   \fi
2361   \let\@@fc@numstr#2\relax
2362   \edef#2{\@@fc@numstr\ \@unitstring{\@strctr}}%
2363 \fi
2364 \else
2365   \ifnum\@strctr<10\relax
2366     \ifnum\@strctr=0\relax
2367       \ifnum#1<100\relax
2368         \let\@@fc@numstr#2\relax
2369         \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
2370       \fi
2371     \else%(>0,<10)
2372       \let\@@fc@numstr#2\relax
2373       \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
2374     \fi
2375   \else%>10
2376     \@modulo{\@strctr}{10}%
2377     \let\@@fc@numstr#2\relax
2378     \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
2379   \fi
2380 \fi
2381 }

```

As above, but for ordinals.

```

2382 \newcommand*{\@@ordinalstringportuges}[2]{%
2383 \@strctr=#1\relax
2384 \ifnum#1>99999
2385 \PackageError{fmtcount}{Out of range}%
2386 {This macro only works for values less than 100000}%
2387 \else
2388 \ifnum#1<0
2389 \PackageError{fmtcount}{Negative numbers not permitted}%
2390 {This macro does not work for negative numbers, however
2391 you can try typing "minus" first, and then pass the modulus of
2392 this number}%
2393 \else
2394 \def#2{%
2395 \ifnum\@strctr>999\relax
2396   \divide\@strctr by 1000\relax
2397   \ifnum\@strctr>1\relax
2398     \ifnum\@strctr>9\relax
2399       \@tmpstrctr=\@strctr
2400       \ifnum\@strctr<20
2401         \@modulo{\@tmpstrctr}{10}%

```



```

2402     \let\@@fc@ordstr#2\relax
2403     \edef#2{\@@fc@ordstr\@tenthstring{\@tmpstrctr}}%
2404     \else
2405         \divide\@tmpstrctr by 10\relax
2406         \let\@@fc@ordstr#2\relax
2407         \edef#2{\@@fc@ordstr\@tenthstring{\@tmpstrctr}}%
2408         \@tmpstrctr=\@strctr
2409         \@modulo{\@tmpstrctr}{10}%
2410         \ifnum\@tmpstrctr>0\relax
2411             \let\@@fc@ordstr#2\relax
2412             \edef#2{\@@fc@ordstr\@unitthstring{\@tmpstrctr}}%
2413         \fi
2414     \fi
2415     \else
2416         \let\@@fc@ordstr#2\relax
2417         \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
2418     \fi
2419 \fi
2420 \let\@@fc@ordstr#2\relax
2421 \edef#2{\@@fc@ordstr\@thousandth}%
2422 \fi
2423 \@strctr=#1\relax
2424 \@modulo{\@strctr}{1000}%
2425 \ifnum\@strctr>99\relax
2426     \@tmpstrctr=\@strctr
2427     \divide\@tmpstrctr by 100\relax
2428     \ifnum#1>1000\relax
2429         \let\@@fc@ordstr#2\relax
2430         \edef#2{\@@fc@ordstr-}%
2431     \fi
2432     \let\@@fc@ordstr#2\relax
2433     \edef#2{\@@fc@ordstr\@hundredthstring{\@tmpstrctr}}%
2434 \fi
2435 \@modulo{\@strctr}{100}%
2436 \ifnum#1>99\relax
2437     \ifnum\@strctr>0\relax
2438         \let\@@fc@ordstr#2\relax
2439         \edef#2{\@@fc@ordstr-}%
2440     \fi
2441 \fi
2442 \ifnum\@strctr>9\relax
2443     \@tmpstrctr=\@strctr
2444     \divide\@tmpstrctr by 10\relax
2445     \let\@@fc@ordstr#2\relax
2446     \edef#2{\@@fc@ordstr\@tenthstring{\@tmpstrctr}}%
2447     \@tmpstrctr=\@strctr
2448     \@modulo{\@tmpstrctr}{10}%
2449     \ifnum\@tmpstrctr>0\relax
2450         \let\@@fc@ordstr#2\relax
2451         \edef#2{\@@fc@ordstr-\@unitthstring{\@tmpstrctr}}%
2452     \fi
2453 \else
2454     \ifnum\@strctr=0\relax
2455         \ifnum#1=0\relax

```

```

2456     \let\@@fc@ordstr#2\relax
2457     \edef#2{\@@fc@ordstr\@unitstring{0}}%
2458     \fi
2459     \else
2460     \let\@@fc@ordstr#2\relax
2461     \edef#2{\@@fc@ordstr\@unitthstring{\@strctr}}%
2462     \fi
2463 \fi
2464 \fi
2465 \fi
2466 }

```

11.6 fc-spanish.def

Spanish definitions

```
2467 \ProvidesFile{fc-spanish.def}[2007/05/26]
```

Define macro that converts a number or count register (first argument) to an ordinal, and stores the result in the second argument, which must be a control sequence. Masculine:

```

2468 \newcommand{\@ordinalMspanish}[2]{%
2469 \edef#2{\number#1\relax\noexpand\fmtord{o}}}

```

Feminine:

```

2470 \newcommand{\@ordinalFspanish}[2]{%
2471 \edef#2{\number#1\relax\noexpand\fmtord{a}}}

```

Make neuter same as masculine:

```
2472 \let\@ordinalNspanish\@ordinalMspanish
```

Convert a number to text. The easiest way to do this is to break it up into units, tens, teens, twenties and hundreds. Units (argument must be a number from 0 to 9):

```

2473 \newcommand{\@@unitstringspanish}[1]{%
2474 \ifcase#1\relax
2475 cero%
2476 \or uno%
2477 \or dos%
2478 \or tres%
2479 \or cuatro%
2480 \or cinco%
2481 \or seis%
2482 \or siete%
2483 \or ocho%
2484 \or nueve%
2485 \fi
2486 }

```

Feminine:

```

2487 \newcommand{\@@unitstringFspanish}[1]{%
2488 \ifcase#1\relax
2489 cera%
2490 \or una%
2491 \or dos%
2492 \or tres%

```

```

2493 \or cuatro%
2494 \or cinco%
2495 \or seis%
2496 \or siete%
2497 \or ocho%
2498 \or nueve%
2499 \fi
2500 }

```

Tens (argument must go from 1 to 10):

```

2501 \newcommand{\@@tenstringspanish}[1]{%
2502 \ifcase#1\relax
2503 \or diez%
2504 \or veinte%
2505 \or treinta%
2506 \or cuarenta%
2507 \or cincuenta%
2508 \or sesenta%
2509 \or setenta%
2510 \or ochenta%
2511 \or noventa%
2512 \or cien%
2513 \fi
2514 }

```

Teens:

```

2515 \newcommand{\@@teenstringspanish}[1]{%
2516 \ifcase#1\relax
2517 diez%
2518 \or once%
2519 \or doce%
2520 \or trece%
2521 \or catorce%
2522 \or quince%
2523 \or diecis\`eis%
2524 \or diecisiete%
2525 \or dieciocho%
2526 \or diecinueve%
2527 \fi
2528 }

```

Twenties:

```

2529 \newcommand{\@@twentystringspanish}[1]{%
2530 \ifcase#1\relax
2531 veinte%
2532 \or veintiuno%
2533 \or veintid\`os%
2534 \or veintitr\`es%
2535 \or veinticuatro%
2536 \or veinticinco%
2537 \or veintiseis%
2538 \or veintisiete%
2539 \or veintiocho%
2540 \or veintinueve%
2541 \fi}

```

Feminine form:

```
2542 \newcommand{\@@twentystringFspanish}[1]{%
2543 \ifcase#1\relax
2544 veinte%
2545 \or veintiuna%
2546 \or veintid\'os%
2547 \or veintitr\'es%
2548 \or veinticuatro%
2549 \or veinticinco%
2550 \or veintis\'eis%
2551 \or veintisiete%
2552 \or veintiocho%
2553 \or veintinueve%
2554 \fi}
```

Hundreds:

```
2555 \newcommand{\@@hundredstringspanish}[1]{%
2556 \ifcase#1\relax
2557 \or ciento%
2558 \or doscientos%
2559 \or trescientos%
2560 \or cuatrocientos%
2561 \or quinientos%
2562 \or seiscientos%
2563 \or setecientos%
2564 \or ochocientos%
2565 \or novecientos%
2566 \fi}
```

Feminine form:

```
2567 \newcommand{\@@hundredstringFspanish}[1]{%
2568 \ifcase#1\relax
2569 \or cienta%
2570 \or doscientas%
2571 \or trescientas%
2572 \or cuatrocientas%
2573 \or quinientas%
2574 \or seiscientas%
2575 \or setecientas%
2576 \or ochocientas%
2577 \or novecientas%
2578 \fi}
```

As above, but with initial letter uppercase:

```
2579 \newcommand{\@@Unitstringspanish}[1]{%
2580 \ifcase#1\relax
2581 Cero%
2582 \or Uno%
2583 \or Dos%
2584 \or Tres%
2585 \or Cuatro%
2586 \or Cinco%
2587 \or Seis%
2588 \or Siete%
2589 \or Ocho%
```

```

2590 \or Nueve%
2591 \fi
2592 }

```

Feminine form:

```

2593 \newcommand{\@@UnitstringFspanish}[1]{%
2594 \ifcase#1\relax
2595 Cera%
2596 \or Una%
2597 \or Dos%
2598 \or Tres%
2599 \or Cuatro%
2600 \or Cinco%
2601 \or Seis%
2602 \or Siete%
2603 \or Ocho%
2604 \or Nueve%
2605 \fi
2606 }

```

Tens:

```

2607 \newcommand{\@@Tenstringspanish}[1]{%
2608 \ifcase#1\relax
2609 \or Diez%
2610 \or Viente%
2611 \or Treinta%
2612 \or Cuarenta%
2613 \or Cincuenta%
2614 \or Sesenta%
2615 \or Setenta%
2616 \or Ochenta%
2617 \or Noventa%
2618 \or Cien%
2619 \fi
2620 }

```

Teens:

```

2621 \newcommand{\@@Teenstringspanish}[1]{%
2622 \ifcase#1\relax
2623 Diez%
2624 \or Once%
2625 \or Doce%
2626 \or Trece%
2627 \or Catorce%
2628 \or Quince%
2629 \or Diecis\'eis%
2630 \or Diecisiete%
2631 \or Dieciocho%
2632 \or Diecinueve%
2633 \fi
2634 }

```

Twenties:

```

2635 \newcommand{\@@Twentystringspanish}[1]{%
2636 \ifcase#1\relax
2637 Veinte%

```

```

2638 \or Veintiuno%
2639 \or Veintid\'os%
2640 \or Veintitr\'es%
2641 \or Veinticuatro%
2642 \or Veinticinco%
2643 \or Veintis\'eis%
2644 \or Veintisiete%
2645 \or Veintiocho%
2646 \or Veintinueve%
2647 \fi}

```

Feminine form:

```

2648 \newcommand{\@@TwentystringFspanish}[1]{%
2649 \ifcase#1\relax
2650 Veinte%
2651 \or Veintiuna%
2652 \or Veintid\'os%
2653 \or Veintitr\'es%
2654 \or Veinticuatro%
2655 \or Veinticinco%
2656 \or Veintis\'eis%
2657 \or Veintisiete%
2658 \or Veintiocho%
2659 \or Veintinueve%
2660 \fi}

```

Hundreds:

```

2661 \newcommand{\@@Hundredstringspanish}[1]{%
2662 \ifcase#1\relax
2663 \or Ciento%
2664 \or Doscientos%
2665 \or Trescientos%
2666 \or Cuatrocientos%
2667 \or Quinientos%
2668 \or Seiscientos%
2669 \or Setecientos%
2670 \or Ochocientos%
2671 \or Novecientos%
2672 \fi}

```

Feminine form:

```

2673 \newcommand{\@@HundredstringFspanish}[1]{%
2674 \ifcase#1\relax
2675 \or Cienta%
2676 \or Doscientas%
2677 \or Trescientas%
2678 \or Cuatrocientas%
2679 \or Quinientas%
2680 \or Seiscientas%
2681 \or Setecientas%
2682 \or Ochocientas%
2683 \or Novecientas%
2684 \fi}

```

This has changed in version 1.09, so that it now stores the result in the second argument, but doesn't display anything. Since it only affects internal macros, it

shouldn't affect documents created with older versions. (These internal macros are not meant for use in documents.)

```
2685 \DeclareRobustCommand{\@numberstringMspanish}[2]{%
2686 \let\@unitstring=\@@unitstringspanish
2687 \let\@teenstring=\@@teenstringspanish
2688 \let\@tenstring=\@@tenstringspanish
2689 \let\@twentystring=\@@twentystringspanish
2690 \let\@hundredstring=\@@hundredstringspanish
2691 \def\@hundred{cien}\def\@thousand{mil}%
2692 \def\@andname{y}%
2693 \@@numberstringspanish{#1}{#2}}
```

Feminine form:

```
2694 \DeclareRobustCommand{\@numberstringFspanish}[2]{%
2695 \let\@unitstring=\@@unitstringFspanish
2696 \let\@teenstring=\@@teenstringspanish
2697 \let\@tenstring=\@@tenstringspanish
2698 \let\@twentystring=\@@twentystringFspanish
2699 \let\@hundredstring=\@@hundredstringFspanish
2700 \def\@hundred{cien}\def\@thousand{mil}%
2701 \def\@andname{y}%
2702 \@@numberstringspanish{#1}{#2}}
```

Make neuter same as masculine:

```
2703 \let\@numberstringNspanish\@numberstringMspanish
```

As above, but initial letters in upper case:

```
2704 \DeclareRobustCommand{\@NumberstringMspanish}[2]{%
2705 \let\@unitstring=\@@Unitstringspanish
2706 \let\@teenstring=\@@Teenstringspanish
2707 \let\@tenstring=\@@Tenstringspanish
2708 \let\@twentystring=\@@Twentystringspanish
2709 \let\@hundredstring=\@@Hundredstringspanish
2710 \def\@andname{Y}%
2711 \def\@hundred{Cien}\def\@thousand{Mil}%
2712 \@@numberstringspanish{#1}{#2}}
```

Feminine form:

```
2713 \DeclareRobustCommand{\@NumberstringFspanish}[2]{%
2714 \let\@unitstring=\@@UnitstringFspanish
2715 \let\@teenstring=\@@Teenstringspanish
2716 \let\@tenstring=\@@Tenstringspanish
2717 \let\@twentystring=\@@TwentystringFspanish
2718 \let\@hundredstring=\@@HundredstringFspanish
2719 \def\@andname{Y}%
2720 \def\@hundred{Cien}\def\@thousand{Mil}%
2721 \@@numberstringspanish{#1}{#2}}
```

Make neuter same as masculine:

```
2722 \let\@NumberstringNspanish\@NumberstringMspanish
```

As above, but for ordinals.

```
2723 \DeclareRobustCommand{\@OrdinalstringMspanish}[2]{%
2724 \let\@unitthstring=\@@unitthstringspanish
2725 \let\@unitstring=\@@unitstringspanish
2726 \let\@teenthstring=\@@teenthstringspanish
```

```

2727 \let\@tenthstring=\@tenthstringspanish
2728 \let\@hundredthstring=\@hundredthstringspanish
2729 \def\@thousandth{mil\'esimo}%
2730 \@ordinalstringspanish{#1}{#2}}

```

Feminine form:

```

2731 \DeclareRobustCommand{\@ordinalstringFspanish}[2]{%
2732 \let\@unitthstring=\@unitthstringFspanish
2733 \let\@unitstring=\@unitstringFspanish
2734 \let\@teenthstring=\@teenthstringFspanish
2735 \let\@tenthstring=\@tenthstringFspanish
2736 \let\@hundredthstring=\@hundredthstringFspanish
2737 \def\@thousandth{mil\'esima}%
2738 \@ordinalstringspanish{#1}{#2}}

```

Make neuter same as masculine:

```

2739 \let\@ordinalstringNspanish\@ordinalstringMspanish

```

As above, but with initial letters in upper case.

```

2740 \DeclareRobustCommand{\@OrdinalstringMspanish}[2]{%
2741 \let\@unitthstring=\@Unitthstringspanish
2742 \let\@unitstring=\@Unitstringspanish
2743 \let\@teenthstring=\@Teenthstringspanish
2744 \let\@tenthstring=\@Tenthstringspanish
2745 \let\@hundredthstring=\@Hundredthstringspanish
2746 \def\@thousandth{Mil\'esimo}%
2747 \@ordinalstringspanish{#1}{#2}}

```

Feminine form:

```

2748 \DeclareRobustCommand{\@OrdinalstringFspanish}[2]{%
2749 \let\@unitthstring=\@UnitthstringFspanish
2750 \let\@unitstring=\@UnitstringFspanish
2751 \let\@teenthstring=\@TeenthstringFspanish
2752 \let\@tenthstring=\@TenthstringFspanish
2753 \let\@hundredthstring=\@HundredthstringFspanish
2754 \def\@thousandth{Mil\'esima}%
2755 \@ordinalstringspanish{#1}{#2}}

```

Make neuter same as masculine:

```

2756 \let\@OrdinalstringNspanish\@OrdinalstringMspanish

```

Code for convert numbers into textual ordinals. As before, it is easier to split it into units, tens, teens and hundreds. Units:

```

2757 \newcommand{\@unitthstringspanish}[1]{%
2758 \ifcase#1\relax
2759 cero%
2760 \or primero%
2761 \or segundo%
2762 \or tercero%
2763 \or cuarto%
2764 \or quinto%
2765 \or sexto%
2766 \or s\'eptimo%
2767 \or octavo%
2768 \or noveno%
2769 \fi
2770 }

```


Tens:

```
2771 \newcommand{\@@tenthstringspanish}[1]{%
2772 \ifcase#1\relax
2773 \or d\'ecimo%
2774 \or vig\'esimo%
2775 \or trig\'esimo%
2776 \or cuadrag\'esimo%
2777 \or quincuag\'esimo%
2778 \or sexag\'esimo%
2779 \or septuag\'esimo%
2780 \or octog\'esimo%
2781 \or nonag\'esimo%
2782 \fi
2783 }
```

Teens:

```
2784 \newcommand{\@@teenthstringspanish}[1]{%
2785 \ifcase#1\relax
2786 d\'ecimo%
2787 \or und\'ecimo%
2788 \or duod\'ecimo%
2789 \or decimotercero%
2790 \or decimocuarto%
2791 \or decimoquinto%
2792 \or decimosexto%
2793 \or decimos\'eptimo%
2794 \or decimoctavo%
2795 \or decimonoveno%
2796 \fi
2797 }
```

Hundreds:

```
2798 \newcommand{\@@hundredthstringspanish}[1]{%
2799 \ifcase#1\relax
2800 \or cent\'esimo%
2801 \or ducent\'esimo%
2802 \or tricent\'esimo%
2803 \or cuadringent\'esimo%
2804 \or quingent\'esimo%
2805 \or sexcent\'esimo%
2806 \or septing\'esimo%
2807 \or octingent\'esimo%
2808 \or noningent\'esimo%
2809 \fi}
```

Units (feminine):

```
2810 \newcommand{\@@unitthstringFspanish}[1]{%
2811 \ifcase#1\relax
2812 cera%
2813 \or primera%
2814 \or segunda%
2815 \or tercera%
2816 \or cuarta%
2817 \or quinta%
2818 \or sexta%
```

```

2819 \or s\'eptima%
2820 \or octava%
2821 \or novena%
2822 \fi
2823 }

```

Tens (feminine):

```

2824 \newcommand{\@@tenthstringFspanish}[1]{%
2825 \ifcase#1\relax
2826 \or d\'ecima%
2827 \or vig\'esima%
2828 \or trig\'esima%
2829 \or cuadrag\'esima%
2830 \or quincuag\'esima%
2831 \or sexag\'esima%
2832 \or septuag\'esima%
2833 \or octog\'esima%
2834 \or nonag\'esima%
2835 \fi
2836 }

```

Teens (feminine)

```

2837 \newcommand{\@@teenthstringFspanish}[1]{%
2838 \ifcase#1\relax
2839 d\'ecima%
2840 \or und\'ecima%
2841 \or duod\'ecima%
2842 \or decimotercera%
2843 \or decimocuarta%
2844 \or decimoquinta%
2845 \or decimosexta%
2846 \or decimos\'eptima%
2847 \or decimoctava%
2848 \or decimonovena%
2849 \fi
2850 }

```

Hundreds (feminine)

```

2851 \newcommand{\@@hundredthstringFspanish}[1]{%
2852 \ifcase#1\relax
2853 \or cent\'esima%
2854 \or ducent\'esima%
2855 \or tricent\'esima%
2856 \or cuadringent\'esima%
2857 \or quingent\'esima%
2858 \or sexcent\'esima%
2859 \or septing\'esima%
2860 \or octingent\'esima%
2861 \or noningent\'esima%
2862 \fi}

```

As above, but with initial letters in upper case

```

2863 \newcommand{\@@Unitthstringspanish}[1]{%
2864 \ifcase#1\relax
2865 Cero%
2866 \or Primero%

```

2867 \or Segundo%
 2868 \or Tercero%
 2869 \or Cuarto%
 2870 \or Quinto%
 2871 \or Sexto%
 2872 \or S\'eptimo%
 2873 \or Octavo%
 2874 \or Noveno%
 2875 \fi
 2876 }

Tens:

2877 \newcommand{\@@Tenthstringspanish}[1]{%
 2878 \ifcase#1\relax
 2879 \or D\'ecimo%
 2880 \or Vig\'esimo%
 2881 \or Trig\'esimo%
 2882 \or Cuadrag\'esimo%
 2883 \or Quincuag\'esimo%
 2884 \or Sexag\'esimo%
 2885 \or Septuag\'esimo%
 2886 \or Octog\'esimo%
 2887 \or Nonag\'esimo%
 2888 \fi
 2889 }

Teens:

2890 \newcommand{\@@Teenthstringspanish}[1]{%
 2891 \ifcase#1\relax
 2892 D\'ecimo%
 2893 \or Und\'ecimo%
 2894 \or Duod\'ecimo%
 2895 \or Decimotercero%
 2896 \or Decimocuarto%
 2897 \or Decimoquinto%
 2898 \or Decimosexto%
 2899 \or Decimos\'eptimo%
 2900 \or Decimoctavo%
 2901 \or Decimonoveno%
 2902 \fi
 2903 }

Hundreds

2904 \newcommand{\@@Hundredthstringspanish}[1]{%
 2905 \ifcase#1\relax
 2906 \or Cent\'esimo%
 2907 \or Ducent\'esimo%
 2908 \or Tricent\'esimo%
 2909 \or Cuadringent\'esimo%
 2910 \or Quingent\'esimo%
 2911 \or Sexcent\'esimo%
 2912 \or Septing\'esimo%
 2913 \or Octingent\'esimo%
 2914 \or Noningent\'esimo%
 2915 \fi}

As above, but feminine.

```
2916 \newcommand{\@@UnitthstringFspanish}[1]{%
2917 \ifcase#1\relax
2918 Cera%
2919 \or Primera%
2920 \or Segunda%
2921 \or Tercera%
2922 \or Cuarta%
2923 \or Quinta%
2924 \or Sexta%
2925 \or S\'eptima%
2926 \or Octava%
2927 \or Novena%
2928 \fi
2929 }
```

Tens (feminine)

```
2930 \newcommand{\@@TenthstringFspanish}[1]{%
2931 \ifcase#1\relax
2932 \or D\'ecima%
2933 \or Vig\'esima%
2934 \or Trig\'esima%
2935 \or Cuadrag\'esima%
2936 \or Quincuag\'esima%
2937 \or Sexag\'esima%
2938 \or Septuag\'esima%
2939 \or Octog\'esima%
2940 \or Nonag\'esima%
2941 \fi
2942 }
```

Teens (feminine):

```
2943 \newcommand{\@@TeenthstringFspanish}[1]{%
2944 \ifcase#1\relax
2945 D\'ecima%
2946 \or Und\'ecima%
2947 \or Duod\'ecima%
2948 \or Decimotercera%
2949 \or Decimocuarta%
2950 \or Decimoquinta%
2951 \or Decimosexta%
2952 \or Decimos\'eptima%
2953 \or Decimoctava%
2954 \or Decimonovena%
2955 \fi
2956 }
```

Hundreds (feminine):

```
2957 \newcommand{\@@HundredthstringFspanish}[1]{%
2958 \ifcase#1\relax
2959 \or Cent\'esima%
2960 \or Ducent\'esima%
2961 \or Tricent\'esima%
2962 \or Cuadringent\'esima%
2963 \or Quingent\'esima%
```

```

2964 \or Sexcent\'esima%
2965 \or Septing\'esima%
2966 \or Octingent\'esima%
2967 \or Noningent\'esima%
2968 \fi}
2969

```

This has changed in version 1.09, so that it now stores the results in the second argument (which must be a control sequence), but it doesn't display anything. Since it only affects internal macros, it shouldn't affect documnets created with older versions. (These internal macros are not meant for use in documents.)

```

2970 \newcommand{\@@numberstringspanish}[2]{%
2971 \ifnum#1>99999
2972 \PackageError{fmtcount}{Out of range}%
2973 {This macro only works for values less than 100000}%
2974 \else
2975 \ifnum#1<0
2976 \PackageError{fmtcount}{Negative numbers not permitted}%
2977 {This macro does not work for negative numbers, however
2978 you can try typing "minus" first, and then pass the modulus of
2979 this number}%
2980 \fi
2981 \fi
2982 \def#2{%
2983 \@strctr=#1\relax \divide\@strctr by 1000\relax
2984 \ifnum\@strctr>9
2985 % #1 is greater or equal to 10000
2986 \divide\@strctr by 10
2987 \ifnum\@strctr>1
2988 \let\@@fc@numstr#2\relax
2989 \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
2990 \@strctr=#1 \divide\@strctr by 1000\relax
2991 \@modulo{\@strctr}{10}%
2992 \ifnum\@strctr>0\relax
2993 \let\@@fc@numstr#2\relax
2994 \edef#2{\@@fc@numstr\ \@andname\ \@unitstring{\@strctr}}%
2995 \fi
2996 \else
2997 \@strctr=#1\relax
2998 \divide\@strctr by 1000\relax
2999 \@modulo{\@strctr}{10}%
3000 \let\@@fc@numstr#2\relax
3001 \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
3002 \fi
3003 \let\@@fc@numstr#2\relax
3004 \edef#2{\@@fc@numstr\ \@thousand}%
3005 \else
3006 \ifnum\@strctr>0\relax
3007 \ifnum\@strctr>1\relax
3008 \let\@@fc@numstr#2\relax
3009 \edef#2{\@@fc@numstr\@unitstring{\@strctr}\ }%
3010 \fi
3011 \let\@@fc@numstr#2\relax
3012 \edef#2{\@@fc@numstr\@thousand}%

```

```

3013 \fi
3014 \fi
3015 \@strctr=#1\relax \@modulo{\@strctr}{1000}%
3016 \divide\@strctr by 100\relax
3017 \ifnum\@strctr>0\relax
3018 \ifnum#1>1000\relax
3019 \let\@fc@numstr#2\relax
3020 \edef#2{\@fc@numstr\ }%
3021 \fi
3022 \@tmpstrctr=#1\relax
3023 \@modulo{\@tmpstrctr}{1000}%
3024 \ifnum\@tmpstrctr=100\relax
3025 \let\@fc@numstr#2\relax
3026 \edef#2{\@fc@numstr\@tenstring{10}}%
3027 \else
3028 \let\@fc@numstr#2\relax
3029 \edef#2{\@fc@numstr\@hundredstring{\@strctr}}%
3030 \fi
3031 \fi
3032 \@strctr=#1\relax \@modulo{\@strctr}{100}%
3033 \ifnum#1>100\relax
3034 \ifnum\@strctr>0\relax
3035 \let\@fc@numstr#2\relax
3036 \edef#2{\@fc@numstr\ \@andname\ }%
3037 \fi
3038 \fi
3039 \ifnum\@strctr>29\relax
3040 \divide\@strctr by 10\relax
3041 \let\@fc@numstr#2\relax
3042 \edef#2{\@fc@numstr\@tenstring{\@strctr}}%
3043 \@strctr=#1\relax \@modulo{\@strctr}{10}%
3044 \ifnum\@strctr>0\relax
3045 \let\@fc@numstr#2\relax
3046 \edef#2{\@fc@numstr\ \@andname\ \@unitstring{\@strctr}}%
3047 \fi
3048 \else
3049 \ifnum\@strctr<10\relax
3050 \ifnum\@strctr=0\relax
3051 \ifnum#1<100\relax
3052 \let\@fc@numstr#2\relax
3053 \edef#2{\@fc@numstr\@unitstring{\@strctr}}%
3054 \fi
3055 \else
3056 \let\@fc@numstr#2\relax
3057 \edef#2{\@fc@numstr\@unitstring{\@strctr}}%
3058 \fi
3059 \else
3060 \ifnum\@strctr>19\relax
3061 \@modulo{\@strctr}{10}%
3062 \let\@fc@numstr#2\relax
3063 \edef#2{\@fc@numstr\@twentystring{\@strctr}}%
3064 \else
3065 \@modulo{\@strctr}{10}%
3066 \let\@fc@numstr#2\relax

```

```

3067     \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
3068     \fi
3069 \fi
3070 \fi
3071 }

    As above, but for ordinals
3072 \newcommand{\@@ordinalstringspanish}[2]{%
3073 \@strctr=#1\relax
3074 \ifnum#1>99999
3075 \PackageError{fmtcount}{Out of range}%
3076 {This macro only works for values less than 100000}%
3077 \else
3078 \ifnum#1<0
3079 \PackageError{fmtcount}{Negative numbers not permitted}%
3080 {This macro does not work for negative numbers, however
3081 you can try typing "minus" first, and then pass the modulus of
3082 this number}%
3083 \else
3084 \def#2{%
3085 \ifnum\@strctr>999\relax
3086   \divide\@strctr by 1000\relax
3087   \ifnum\@strctr>1\relax
3088     \ifnum\@strctr>9\relax
3089       \@tmpstrctr=\@strctr
3090       \ifnum\@strctr<20
3091         \@modulo{\@tmpstrctr}{10}%
3092         \let\@@fc@ordstr#2\relax
3093         \edef#2{\@@fc@ordstr\@tenthstring{\@tmpstrctr}}%
3094       \else
3095         \divide\@tmpstrctr by 10\relax
3096         \let\@@fc@ordstr#2\relax
3097         \edef#2{\@@fc@ordstr\@tenthstring{\@tmpstrctr}}%
3098         \@tmpstrctr=\@strctr
3099         \@modulo{\@tmpstrctr}{10}%
3100         \ifnum\@tmpstrctr>0\relax
3101           \let\@@fc@ordstr#2\relax
3102           \edef#2{\@@fc@ordstr\@unitthstring{\@tmpstrctr}}%
3103         \fi
3104       \fi
3105     \else
3106       \let\@@fc@ordstr#2\relax
3107       \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
3108     \fi
3109   \fi
3110   \let\@@fc@ordstr#2\relax
3111   \edef#2{\@@fc@ordstr\@thousandth}%
3112 \fi
3113 \@strctr=#1\relax
3114 \@modulo{\@strctr}{1000}%
3115 \ifnum\@strctr>99\relax
3116   \@tmpstrctr=\@strctr
3117   \divide\@tmpstrctr by 100\relax
3118   \ifnum#1>1000\relax
3119     \let\@@fc@ordstr#2\relax

```

```

3120 \edef#2{\@fc@ordstr\ }%
3121 \fi
3122 \let\@fc@ordstr#2\relax
3123 \edef#2{\@fc@ordstr\@hundredthstring{\@tmpstrctr}}%
3124 \fi
3125 \@modulo{\@strctr}{100}%
3126 \ifnum#1>99\relax
3127 \ifnum\@strctr>0\relax
3128 \let\@fc@ordstr#2\relax
3129 \edef#2{\@fc@ordstr\ }%
3130 \fi
3131 \fi
3132 \ifnum\@strctr>19\relax
3133 \@tmpstrctr=\@strctr
3134 \divide\@tmpstrctr by 10\relax
3135 \let\@fc@ordstr#2\relax
3136 \edef#2{\@fc@ordstr\@tenthstring{\@tmpstrctr}}%
3137 \@tmpstrctr=\@strctr
3138 \@modulo{\@tmpstrctr}{10}%
3139 \ifnum\@tmpstrctr>0\relax
3140 \let\@fc@ordstr#2\relax
3141 \edef#2{\@fc@ordstr\ \@unitthstring{\@tmpstrctr}}%
3142 \fi
3143 \else
3144 \ifnum\@strctr>9\relax
3145 \@modulo{\@strctr}{10}%
3146 \let\@fc@ordstr#2\relax
3147 \edef#2{\@fc@ordstr\@teenthstring{\@strctr}}%
3148 \else
3149 \ifnum\@strctr=0\relax
3150 \ifnum#1=0\relax
3151 \let\@fc@ordstr#2\relax
3152 \edef#2{\@fc@ordstr\@unitstring{0}}%
3153 \fi
3154 \else
3155 \let\@fc@ordstr#2\relax
3156 \edef#2{\@fc@ordstr\@unitthstring{\@strctr}}%
3157 \fi
3158 \fi
3159 \fi
3160 \fi
3161 \fi
3162 }

```

11.7 fc-UKenglish.def

UK English definitions

```
3163 \ProvidesFile{fc-UKenglish}[2007/06/14]
```

Check that fc-english.def has been loaded

```
3164 \@ifundefined{ordinalMenglish}{\input{fc-english.def}}{}
```

These are all just synonyms for the commands provided by fc-english.def.

```
3165 \let\@ordinalMUKenglish\@ordinalMenglish
```



```

3166 \let\@ordinalFUKenglish\@ordinalMenglish
3167 \let\@ordinalNUKenglish\@ordinalMenglish
3168 \let\@numberstringMUKenglish\@numberstringMenglish
3169 \let\@numberstringFUKenglish\@numberstringMenglish
3170 \let\@numberstringNUKenglish\@numberstringMenglish
3171 \let\@NumberstringMUKenglish\@NumberstringMenglish
3172 \let\@NumberstringFUKenglish\@NumberstringMenglish
3173 \let\@NumberstringNUKenglish\@NumberstringMenglish
3174 \let\@ordinalstringMUKenglish\@ordinalstringMenglish
3175 \let\@ordinalstringFUKenglish\@ordinalstringMenglish
3176 \let\@ordinalstringNUKenglish\@ordinalstringMenglish
3177 \let\@OrdinalstringMUKenglish\@OrdinalstringMenglish
3178 \let\@OrdinalstringFUKenglish\@OrdinalstringMenglish
3179 \let\@OrdinalstringNUKenglish\@OrdinalstringMenglish

```

11.8 fc-USenglish.def

US English definitions

```
3180 \ProvidesFile{fc-USenglish}[2007/06/14]
```

Check that fc-english.def has been loaded

```
3181 \@ifundefined{ordinalMenglish}{\input{fc-english.def}}{}
```

These are all just synonyms for the commands provided by fc-english.def.

```

3182 \let\@ordinalMUSenglish\@ordinalMenglish
3183 \let\@ordinalFUSenglish\@ordinalMenglish
3184 \let\@ordinalNUSenglish\@ordinalMenglish
3185 \let\@numberstringMUSenglish\@numberstringMenglish
3186 \let\@numberstringFUSenglish\@numberstringMenglish
3187 \let\@numberstringNUSenglish\@numberstringMenglish
3188 \let\@NumberstringMUSenglish\@NumberstringMenglish
3189 \let\@NumberstringFUSenglish\@NumberstringMenglish
3190 \let\@NumberstringNUSenglish\@NumberstringMenglish
3191 \let\@ordinalstringMUSenglish\@ordinalstringMenglish
3192 \let\@ordinalstringFUSenglish\@ordinalstringMenglish
3193 \let\@ordinalstringNUSenglish\@ordinalstringMenglish
3194 \let\@OrdinalstringMUSenglish\@OrdinalstringMenglish
3195 \let\@OrdinalstringFUSenglish\@OrdinalstringMenglish
3196 \let\@OrdinalstringNUSenglish\@OrdinalstringMenglish

```

11.9 fmtcount.sty

This section deals with the code for fmtcount.sty

```

3197 \NeedsTeXFormat{LaTeX2e}
3198 \ProvidesPackage{fmtcount}[2007/06/14 v1.1]
3199 \RequirePackage{ifthen}
3200 \RequirePackage{keyval}

```

These commands need to be defined before the configuration file is loaded.

Define the macro to format the st, nd, rd or th of an ordinal.

```
3201 \providecommand{\fmtord}[1]{\textsuperscript{#1}}
```

Define \padzeroes to specify how many digits should be displayed.

```

3202 \newcount\c@padzeroesN
3203 \c@padzeroesN=1\relax

```

```
3204 \providecommand{\padzeroes}[1][17]{\c@padzeroesN=#1}
```

Load appropriate language definition files (I don't know if there is a standard way of detecting which languages are defined, so I'm just going to check if `\date<language>` is defined):

```
3205 \ifundefined{dateenglish}{\input{fc-english.def}}
3206 \ifundefined{l@UKenglish}{\input{fc-UKenglish.def}}
3207 \ifundefined{l@british}{\input{fc-british.def}}
3208 \ifundefined{l@USenglish}{\input{fc-USenglish.def}}
3209 \ifundefined{datespanish}{\input{fc-spanish.def}}
3210 \ifundefined{dateportuges}{\input{fc-portuges.def}}
3211 \ifundefined{datefrench}{\input{fc-french.def}}
3212 \ifundefined{dategerman}{%
3213 \ifundefined{datengerman}{\input{fc-german.def}}}{%
3214 \input{fc-german.def}}
```

Define keys for use with `\fmtcountsetoptions`. Key to switch French dialects (Does babel store this kind of information?)

```
3215 \def\fmtcount@french{france}
3216 \define@key{fmtcount}{french}[france]{%
3217 \ifundefined{datefrench}{%
3218 \PackageError{fmtcount}{Language 'french' not defined}{You need
3219 to load babel before loading fmtcount}}{
3220 \ifthenelse{\equal{#1}{france}
3221 \or\equal{#1}{swiss}
3222 \or\equal{#1}{belgian}}{%
3223 \def\fmtcount@french{#1}}{%
3224 \PackageError{fmtcount}{Invalid value '#1' to french key}
3225 {Option 'french' can only take the values 'france',
3226 'belgian' or 'swiss'}}
3227 }}
```

Key to determine how to display the ordinal

```
3228 \define@key{fmtcount}{fmtord}{%
3229 \ifthenelse{\equal{#1}{level}
3230 \or\equal{#1}{raise}
3231 \or\equal{#1}{user}}{
3232 \def\fmtcount@fmtord{#1}}{%
3233 \PackageError{fmtcount}{Invalid value '#1' to fmtord key}
3234 {Option 'fmtord' can only take the values 'level', 'raise'
3235 or 'user'}}}
```

Key to determine whether the ordinal should be abbreviated (language dependent, currently only affects French ordinals.)

```
3236 \newif\iffmtord@abbrev
3237 \fmtord@abbrevfalse
3238 \define@key{fmtcount}{abbrev}[true]{%
3239 \ifthenelse{\equal{#1}{true}\or\equal{#1}{false}}{
3240 \csname fmtord@abbrev#1\endcsname}{%
3241 \PackageError{fmtcount}{Invalid value '#1' to fmtord key}
3242 {Option 'fmtord' can only take the values 'true' or
3243 'false'}}}
```

Define command to set options.

```
3244 \newcommand{\fmtcountsetoptions}[1]{%
3245 \def\fmtcount@fmtord{}}
```

```

3246 \setkeys{fmtcount}{#1}%
3247 \@ifundefined{datefrench}{}{%
3248 \edef\@ordinalstringMfrench{\noexpand
3249 \csname @ordinalstringMfrench\fmtcount@french\noexpand\endcsname}%
3250 \edef\@ordinalstringFfrench{\noexpand
3251 \csname @ordinalstringFfrench\fmtcount@french\noexpand\endcsname}%
3252 \edef\@OrdinalstringMfrench{\noexpand
3253 \csname @OrdinalstringMfrench\fmtcount@french\noexpand\endcsname}%
3254 \edef\@OrdinalstringFfrench{\noexpand
3255 \csname @OrdinalstringFfrench\fmtcount@french\noexpand\endcsname}%
3256 \edef\@numberstringMfrench{\noexpand
3257 \csname @numberstringMfrench\fmtcount@french\noexpand\endcsname}%
3258 \edef\@numberstringFfrench{\noexpand
3259 \csname @numberstringFfrench\fmtcount@french\noexpand\endcsname}%
3260 \edef\@NumberstringMfrench{\noexpand
3261 \csname @NumberstringMfrench\fmtcount@french\noexpand\endcsname}%
3262 \edef\@NumberstringFfrench{\noexpand
3263 \csname @NumberstringFfrench\fmtcount@french\noexpand\endcsname}%
3264 }%
3265 %
3266 \ifthenelse{\equal{\fmtcount@fmtord}{level}}{%
3267 \renewcommand{\fmtord}[1]{##1}}{%
3268 \ifthenelse{\equal{\fmtcount@fmtord}{raise}}{%
3269 \renewcommand{\fmtord}[1]{\textsuperscript{##1}}}%
3270 }}
3271 }

```

Load configuration file if it exists. This needs to be done before the package options, to allow the user to override the settings in the configuration file.

```

3272 \InputIfFileExists{fmtcount.cfg}{%
3273 \typeout{Using configuration file fmtcount.cfg}}{%
3274 \typeout{No configuration file fmtcount.cfg found.}}

```

Declare options

```

3275 \DeclareOption{level}{\def\fmtcount@fmtord{level}%
3276 \def\fmtord#1{#1}}
3277 \DeclareOption{raise}{\def\fmtcount@fmtord{raise}%
3278 \def\fmtord#1{\textsuperscript{#1}}}

```

Process package options

```

3279 \ProcessOptions

```

Define macro that performs modulo arithmetic. This is used for the date, time, ordinal and numberstring commands. (The fmtcount package was originally part of the datetime package.)

```

3280 \newcount\@DT@modctr
3281 \def\@modulo#1#2{%
3282 \@DT@modctr=#1\relax
3283 \divide \@DT@modctr by #2\relax
3284 \multiply \@DT@modctr by #2\relax
3285 \advance #1 by -\@DT@modctr}

```

The following registers are needed by \@ordinal etc

```

3286 \newcount\@ordinalctr
3287 \newcount\@orgargctr
3288 \newcount\@strctr

```

3289 \newcount\tmpstrctr

Define commands that display numbers in different bases. Define counters and conditionals needed.

3290 \newif\if@DT@padzeroes

3291 \newcount\@DT@loopN

3292 \newcount\@DT@X

Binary

3293 \newcommand{\@binary}[1]{%

3294 \@DT@padzeroestrue

3295 \@DT@loopN=17\relax

3296 \@strctr=\@DT@loopN

3297 \whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by 1}%

3298 \@strctr=65536\relax

3299 \@DT@X=#1\relax

3300 \loop

3301 \@DT@modctr=\@DT@X

3302 \divide\@DT@modctr by \@strctr

3303 \ifthenelse{\boolean{@DT@padzeroes} \and \(\@DT@modctr=0\) \and \(\@DT@loopN>\c@padzeroesN\)}{-

3304 \ifnum\@DT@modctr=0\else\@DT@padzeroesfalse\fi

3305 \multiply\@DT@modctr by \@strctr

3306 \advance\@DT@X by -\@DT@modctr

3307 \divide\@strctr by 2\relax

3308 \advance\@DT@loopN by -1\relax

3309 \ifnum\@strctr>1

3310 \repeat

3311 \the\@DT@X}

3312

3313 \let\binarynum=\@binary

Octal

3314 \newcommand{\@octal}[1]{%

3315 \ifnum#1>32768

3316 \PackageError{fmtcount}{Value of counter too large for \protect\@octal}{Maximum value 32768}

3317 \else

3318 \@DT@padzeroestrue

3319 \@DT@loopN=6\relax

3320 \@strctr=\@DT@loopN

3321 \whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by 1}%

3322 \@strctr=32768\relax

3323 \@DT@X=#1\relax

3324 \loop

3325 \@DT@modctr=\@DT@X

3326 \divide\@DT@modctr by \@strctr

3327 \ifthenelse{\boolean{@DT@padzeroes} \and \(\@DT@modctr=0\) \and \(\@DT@loopN>\c@padzeroesN\)}{-

3328 \ifnum\@DT@modctr=0\else\@DT@padzeroesfalse\fi

3329 \multiply\@DT@modctr by \@strctr

3330 \advance\@DT@X by -\@DT@modctr

3331 \divide\@strctr by 8\relax

3332 \advance\@DT@loopN by -1\relax

3333 \ifnum\@strctr>1

3334 \repeat

3335 \the\@DT@X

3336 \fi}

```

3337 \let\octalnum=\@octal
      Lowercase hexadecimal
3338 \newcommand{\@@hexadecimal}[1]{\ifcase#10\or1\or2\or3\or4\or5\or6\or7\or8\or9\or a\or b\or c\
3339
3340 \newcommand{\@hexadecimal}[1]{%
3341 \@DT@padzeroestru
3342 \@DT@loopN=5\relax
3343 \@strctr=\@DT@loopN
3344 \whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by 1}%
3345 \@strctr=65536\relax
3346 \@DT@X=#1\relax
3347 \loop
3348 \@DT@modctr=\@DT@X
3349 \divide\@DT@modctr by \@strctr
3350 \ifthenelse{\boolean{\@DT@padzeroes} \and \!(\@DT@modctr=0) \and \!(\@DT@loopN>\c@padzeroesN)}{
3351 \ifnum\@DT@modctr=0\else\@DT@padzeroesfalse\fi
3352 \multiply\@DT@modctr by \@strctr
3353 \advance\@DT@X by -\@DT@modctr
3354 \divide\@strctr by 16\relax
3355 \advance\@DT@loopN by -1\relax
3356 \ifnum\@strctr>1
3357 \repeat
3358 \@hexadecimal\@DT@X}
3359
3360 \let\hexadecimalnum=\@hexadecimal
      Uppercase hexadecimal
3361 \newcommand{\@@Hexadecimal}[1]{\ifcase#10\or1\or2\or3\or4\or5\or6\or
3362 7\or8\or9\or A\or B\or C\or D\or E\or F\fi}
3363
3364 \newcommand{\@Hexadecimal}[1]{%
3365 \@DT@padzeroestru
3366 \@DT@loopN=5\relax
3367 \@strctr=\@DT@loopN
3368 \whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by 1}%
3369 \@strctr=65536\relax
3370 \@DT@X=#1\relax
3371 \loop
3372 \@DT@modctr=\@DT@X
3373 \divide\@DT@modctr by \@strctr
3374 \ifthenelse{\boolean{\@DT@padzeroes} \and \!(\@DT@modctr=0) \and \!(\@DT@loopN>\c@padzeroesN)}{
3375 \ifnum\@DT@modctr=0\else\@DT@padzeroesfalse\fi
3376 \multiply\@DT@modctr by \@strctr
3377 \advance\@DT@X by -\@DT@modctr
3378 \divide\@strctr by 16\relax
3379 \advance\@DT@loopN by -1\relax
3380 \ifnum\@strctr>1
3381 \repeat
3382 \@Hexadecimal\@DT@X}
3383
3384 \let\Hexadecimalnum=\@Hexadecimal
      Uppercase alphabetical representation (a ... z aa ... zz)
3385 \newcommand{\@aaalph}[1]{%

```

```

3386 \@DT@loopN=#1\relax
3387 \advance\@DT@loopN by -1\relax
3388 \divide\@DT@loopN by 26\relax
3389 \@DT@modctr=\@DT@loopN
3390 \multiply\@DT@modctr by 26\relax
3391 \@DT@X=#1\relax
3392 \advance\@DT@X by -1\relax
3393 \advance\@DT@X by -\@DT@modctr
3394 \advance\@DT@loopN by 1\relax
3395 \advance\@DT@X by 1\relax
3396 \loop
3397 \@alph\@DT@X
3398 \advance\@DT@loopN by -1\relax
3399 \ifnum\@DT@loopN>0
3400 \repeat
3401 }
3402
3403 \let\aaalphnum=\@aaalph

```

Uppercase alphabetical representation (a ... z aa ... zz)

```

3404 \newcommand{\@AAAalph}[1]{%
3405 \@DT@loopN=#1\relax
3406 \advance\@DT@loopN by -1\relax
3407 \divide\@DT@loopN by 26\relax
3408 \@DT@modctr=\@DT@loopN
3409 \multiply\@DT@modctr by 26\relax
3410 \@DT@X=#1\relax
3411 \advance\@DT@X by -1\relax
3412 \advance\@DT@X by -\@DT@modctr
3413 \advance\@DT@loopN by 1\relax
3414 \advance\@DT@X by 1\relax
3415 \loop
3416 \@Alph\@DT@X
3417 \advance\@DT@loopN by -1\relax
3418 \ifnum\@DT@loopN>0
3419 \repeat
3420 }
3421
3422 \let\AAAalphnum=\@AAAalph

```

Lowercase alphabetical representation

```

3423 \newcommand{\@abalph}[1]{%
3424 \ifnum#1>17576
3425 \PackageError{fmtcount}{Value of counter too large for \protect\@abalph}{Maximum value 17576}
3426 \else
3427 \@DT@padzeroestruer
3428 \@strctr=17576\relax
3429 \@DT@X=#1\relax
3430 \advance\@DT@X by -1\relax
3431 \loop
3432 \@DT@modctr=\@DT@X
3433 \divide\@DT@modctr by \@strctr
3434 \ifthenelse{\boolean{\@DT@padzeroes} \and \(\@DT@modctr=1\)}{\@alph\@DT@modctr}%
3435 \ifnum\@DT@modctr=1\else\@DT@padzeroesfalse\fi
3436 \multiply\@DT@modctr by \@strctr

```

```

3437 \advance\@DT@X by -\@DT@modctr
3438 \divide\@strctr by 26\relax
3439 \ifnum\@strctr>1
3440 \repeat
3441 \advance\@DT@X by 1\relax
3442 \@alph\@DT@X
3443 \fi}
3444
3445 \let\abalphnum=\@abalph

    Uppercase alphabetical representation
3446 \newcommand{\@ABAlph}[1]{%
3447 \ifnum#1>17576
3448 \PackageError{fmtcount}{Value of counter too large for \protect\@ABAlph}{Maximum value 17576}%
3449 \else
3450 \@DT@padzeroes\true
3451 \@strctr=17576\relax
3452 \@DT@X=#1\relax
3453 \advance\@DT@X by -1\relax
3454 \loop
3455 \@DT@modctr=\@DT@X
3456 \divide\@DT@modctr by \@strctr
3457 \ifthenelse{\boolean{\@DT@padzeroes} \and \(\@DT@modctr=1\)}{\@Alph\@DT@modctr}%
3458 \ifnum\@DT@modctr=1\else\@DT@padzeroes\false\fi
3459 \multiply\@DT@modctr by \@strctr
3460 \advance\@DT@X by -\@DT@modctr
3461 \divide\@strctr by 26\relax
3462 \ifnum\@strctr>1
3463 \repeat
3464 \advance\@DT@X by 1\relax
3465 \@Alph\@DT@X
3466 \fi}
3467
3468 \let\ABAlphnum=\@ABAlph

    Recursive command to count number of characters in argument. \@strctr should
    be set to zero before calling it.
3469 \def\@fmtc@count#1#2\relax{%
3470 \if\relax#1
3471 \else
3472 \advance\@strctr by 1\relax
3473 \@fmtc@count#2\relax
3474 \fi}

    Internal decimal macro:
3475 \newcommand{\@decimal}[1]{%
3476 \@strctr=0\relax
3477 \expandafter\@fmtc@count\number#1\relax
3478 \@DT@loopN=\c@padzeroesN
3479 \advance\@DT@loopN by -\@strctr
3480 \ifnum\@DT@loopN>0\relax
3481 \@strctr=0\relax
3482 \whiledo{\@strctr < \@DT@loopN}{0\advance\@strctr by 1}%
3483 \fi
3484 \number#1\relax

```

```

3485 }
3486
3487 \let\decimalnum=\@decimal

```

This is a bit cumbersome. Previously `\@ordinal` was defined in a similar way to `\abalph` etc. This ensured that the actual value of the counter was written in the new label stuff in the .aux file. However adding in an optional argument to determine the gender for multilingual compatibility messed things up somewhat. This was the only work around I could get to keep the the cross-referencing stuff working, which is why the optional argument comes *after* the compulsory argument, instead of the usual manner of placing it before. Version 1.04 changed `\ordinal` to `\FCordinal` to prevent it clashing with the memoir class.

```

3488 \newcommand{\FCordinal}[1]{%
3489 \expandafter\protect\expandafter\ordinalnum{%
3490 \expandafter\the\csname c@#1\endcsname}}

```

If `\ordinal` isn't defined make `\ordinal` a synonym for `\FCordinal` to maintain compatibility with previous versions.

```

3491 \@ifundefined{ordinal}{\let\ordinal\FCordinal}{%
3492 \PackageWarning{fmtcount}{\string\ordinal
3493 \space already defined use \string\FCordinal \space instead.}}

```

Display ordinal where value is given as a number or count register instead of a counter:

```

3494 \newcommand{\ordinalnum}[1]{\@ifnextchar[{\@ordinalnum{#1}}{%
3495 \@ordinalnum{#1}[m]}}

```

Display ordinal according to gender (neuter added in v1.1):

```

3496 \def\@ordinalnum#1[#2]{%
3497 \ifthenelse{\equal{#2}{f}}{%
3498 \protect\@ordinalF{#1}{\@fc@ordstr}}{%
3499 \ifthenelse{\equal{#2}{n}}{%
3500 \protect\@ordinalN{#1}{\@fc@ordstr}}{%
3501 \ifthenelse{\equal{#2}{m}}{%
3502 \PackageError{fmtcount}{Invalid gender option '#2'}{%
3503 Available options are m, f or n}}%
3504 \protect\@ordinalM{#1}{\@fc@ordstr}}\@fc@ordstr}}

```

Store the ordinal (first argument is identifying name, second argument is a counter.)

```

3505 \newcommand*\storeordinal[2]{%
3506 \expandafter\protect\expandafter\storeordinalnum{#1}{%
3507 \expandafter\the\csname c@#2\endcsname}}

```

Store ordinal (first argument is identifying name, second argument is a number or count register.)

```

3508 \newcommand*\storeordinalnum[2]{%
3509 \@ifnextchar[{\@storeordinalnum{#1}{#2}}{%
3510 \@storeordinalnum{#1}{#2}[m]}}

```

Store ordinal according to gender:

```

3511 \def\@storeordinalnum#1#2[#3]{%
3512 \ifthenelse{\equal{#3}{f}}{%
3513 \protect\@ordinalF{#2}{\@fc@ord}}{%
3514 \ifthenelse{\equal{#3}{n}}{%
3515 \protect\@ordinalN{#2}{\@fc@ord}}{%

```



```

3516 \ifthenelse{\equal{#3}{m}}{\%
3517 \PackageError{fmtcount}{Invalid gender option ‘#3’}{%
3518 Available options are m or f}}%
3519 \protect\ordinalM{#2}{\@fc@ord}}}%
3520 \expandafter\let\csname @fcs@#1\endcsname\@fc@ord}

  Get stored information:
3521 \newcommand*{\FMCuse}[1]{\csname @fcs@#1\endcsname}

  Display ordinal as a string (argument is a counter)
3522 \newcommand{\ordinalstring}[1]{%
3523 \expandafter\protect\expandafter\ordinalstringnum{%
3524 \expandafter\the\csname c@#1\endcsname}}

  Display ordinal as a string (argument is a count register or number.)
3525 \newcommand{\ordinalstringnum}[1]{%
3526 \@ifnextchar[{\@ordinal@string{#1}}{\@ordinal@string{#1}[m]}}

  Display ordinal as a string according to gender.
3527 \def\@ordinal@string#1[#2]{%
3528 \ifthenelse{\equal{#2}{f}}{%
3529 \protect\ordinalstringF{#1}{\@fc@ordstr}}{%
3530 \ifthenelse{\equal{#2}{n}}{%
3531 \protect\ordinalstringN{#1}{\@fc@ordstr}}{%
3532 \ifthenelse{\equal{#2}{m}}{\%
3533 \PackageError{fmtcount}{Invalid gender option ‘#2’ to
3534 \string\ordinalstring}{Available options are m, f or f}}%
3535 \protect\ordinalstringM{#1}{\@fc@ordstr}}{\@fc@ordstr}}

  Store textual representation of number. First argument is identifying name, second
  argument is the counter set to the required number.
3536 \newcommand{\storeordinalstring}[2]{%
3537 \expandafter\protect\expandafter\storeordinalstringnum{#1}{%
3538 \expandafter\the\csname c@#2\endcsname}}

  Store textual representation of number. First argument is identifying name, second
  argument is a count register or number.
3539 \newcommand{\storeordinalstringnum}[2]{%
3540 \@ifnextchar[{\@store@ordinal@string{#1}{#2}}{%
3541 \@store@ordinal@string{#1}{#2}[m]}}

  Store textual representation of number according to gender.
3542 \def\@store@ordinal@string#1#2[#3]{%
3543 \ifthenelse{\equal{#3}{f}}{%
3544 \protect\ordinalstringF{#2}{\@fc@ordstr}}{%
3545 \ifthenelse{\equal{#3}{n}}{%
3546 \protect\ordinalstringN{#2}{\@fc@ordstr}}{%
3547 \ifthenelse{\equal{#3}{m}}{\%
3548 \PackageError{fmtcount}{Invalid gender option ‘#3’ to
3549 \string\ordinalstring}{Available options are m, f or n}}%
3550 \protect\ordinalstringM{#2}{\@fc@ordstr}}}%
3551 \expandafter\let\csname @fcs@#1\endcsname\@fc@ordstr}

  Display ordinal as a string with initial letters in upper case (argument is a counter)
3552 \newcommand{\Ordinalstring}[1]{%
3553 \expandafter\protect\expandafter\Ordinalstringnum{%
3554 \expandafter\the\csname c@#1\endcsname}}

```

Display ordinal as a string with initial letters in upper case (argument is a number or count register)

```
3555 \newcommand{\Ordinalstringnum}[1]{%
3556 \@ifnextchar[{\@Ordinal@string{#1}}{\@Ordinal@string{#1}[m]}}
```

Display ordinal as a string with initial letters in upper case according to gender

```
3557 \def\@Ordinal@string#1[#2]{%
3558 \ifthenelse{\equal{#2}{f}}{%
3559 \protect\@OrdinalstringF{#1}{\@fc@ordstr}}{%
3560 \ifthenelse{\equal{#2}{n}}{%
3561 \protect\@OrdinalstringN{#1}{\@fc@ordstr}}{%
3562 \ifthenelse{\equal{#2}{m}}{%
3563 \PackageError{fmtcount}{Invalid gender option ‘#2’}{%
3564 Available options are m, f or n}}%
3565 \protect\@OrdinalstringM{#1}{\@fc@ordstr}}{\@fc@ordstr}}
```

Store textual representation of number, with initial letters in upper case. First argument is identifying name, second argument is the counter set to the required number.

```
3566 \newcommand{\storeOrdinalstring}[2]{%
3567 \expandafter\protect\expandafter\storeOrdinalstringnum{#1}{%
3568 \expandafter\the\csname c@#2\endcsname}}
```

Store textual representation of number, with initial letters in upper case. First argument is identifying name, second argument is a count register or number.

```
3569 \newcommand{\storeOrdinalstringnum}[2]{%
3570 \@ifnextchar[{\@store@Ordinal@string{#1}{#2}}{%
3571 \@store@Ordinal@string{#1}{#2}[m]}}
```

Store textual representation of number according to gender, with initial letters in upper case.

```
3572 \def\@store@Ordinal@string#1#2[#3]{%
3573 \ifthenelse{\equal{#3}{f}}{%
3574 \protect\@OrdinalstringF{#2}{\@fc@ordstr}}{%
3575 \ifthenelse{\equal{#3}{n}}{%
3576 \protect\@OrdinalstringN{#2}{\@fc@ordstr}}{%
3577 \ifthenelse{\equal{#3}{m}}{%
3578 \PackageError{fmtcount}{Invalid gender option ‘#3’}{%
3579 Available options are m or f}}%
3580 \protect\@OrdinalstringM{#2}{\@fc@ordstr}}%
3581 \expandafter\let\csname @fcs@#1\endcsname\@fc@ordstr}
```

Store upper case textual representation of ordinal. The first argument is identifying name, the second argument is a counter.

```
3582 \newcommand{\storeORDINALstring}[2]{%
3583 \expandafter\protect\expandafter\storeORDINALstringnum{#1}{%
3584 \expandafter\the\csname c@#2\endcsname}}
```

As above, but the second argument is a count register or a number.

```
3585 \newcommand{\storeORDINALstringnum}[2]{%
3586 \@ifnextchar[{\@store@ORDINAL@string{#1}{#2}}{%
3587 \@store@ORDINAL@string{#1}{#2}[m]}}
```

Gender is specified as an optional argument at the end.

```
3588 \def\@store@ORDINAL@string#1#2[#3]{%
3589 \ifthenelse{\equal{#3}{f}}{%
```

```

3590 \protect\@ordinalstringF{#2}{\@fc@ordstr}}{%
3591 \ifthenelse{\equal{#3}{n}}{%
3592 \protect\@ordinalstringN{#2}{\@fc@ordstr}}{%
3593 \ifthenelse{\equal{#3}{m}}{}}{%
3594 \PackageError{fmtcount}{Invalid gender option ‘#3’}{%
3595 Available options are m or f}}%
3596 \protect\@ordinalstringM{#2}{\@fc@ordstr}}}%
3597 \expandafter\edef\csname @fcs@#1\endcsname{%
3598 \noexpand\MakeUppercase{\@fc@ordstr}}

```

Display upper case textual representation of an ordinal. The argument must be a counter.

```

3599 \newcommand{\ORDINALstring}[1]{%
3600 \expandafter\protect\expandafter\ORDINALstringnum{%
3601 \expandafter\the\csname c@#1\endcsname}}

```

As above, but the argument is a count register or a number.

```

3602 \newcommand{\ORDINALstringnum}[1]{%
3603 \@ifnextchar[{\@ORDINAL@string{#1}}{\@ORDINAL@string{#1}[m]}}

```

Gender is specified as an optional argument at the end.

```

3604 \def\@ORDINAL@string#1[#2]{%
3605 \ifthenelse{\equal{#2}{f}}{%
3606 \protect\@ordinalstringF{#1}{\@fc@ordstr}}{%
3607 \ifthenelse{\equal{#2}{n}}{%
3608 \protect\@ordinalstringN{#1}{\@fc@ordstr}}{%
3609 \ifthenelse{\equal{#2}{m}}{}}{%
3610 \PackageError{fmtcount}{Invalid gender option ‘#2’}{%
3611 Available options are m, f or n}}%
3612 \protect\@ordinalstringM{#1}{\@fc@ordstr}}}%
3613 \MakeUppercase{\@fc@ordstr}}

```

Convert number to textual representation, and store. First argument is the identifying name, second argument is a counter containing the number.

```

3614 \newcommand{\storenumberstring}[2]{%
3615 \expandafter\protect\expandafter\storenumberstringnum{#1}{%
3616 \expandafter\the\csname c@#2\endcsname}}

```

As above, but second argument is a number or count register.

```

3617 \newcommand{\storenumberstringnum}[2]{%
3618 \@ifnextchar[{\@store@number@string{#1}{#2}}{%
3619 \@store@number@string{#1}{#2}[m]}}

```

Gender is given as optional argument, *at the end*.

```

3620 \def\@store@number@string#1#2[#3]{%
3621 \ifthenelse{\equal{#3}{f}}{%
3622 \protect\@numberstringF{#2}{\@fc@numstr}}{%
3623 \ifthenelse{\equal{#3}{n}}{%
3624 \protect\@numberstringN{#2}{\@fc@numstr}}{%
3625 \ifthenelse{\equal{#3}{m}}{}}{%
3626 \PackageError{fmtcount}{Invalid gender option ‘#3’}{%
3627 Available options are m, f or n}}%
3628 \protect\@numberstringM{#2}{\@fc@numstr}}}%
3629 \expandafter\let\csname @fcs@#1\endcsname\@fc@numstr}

```

Display textual representation of a number. The argument must be a counter.

```

3630 \newcommand{\numberstring}[1]{%

```

```

3631 \expandafter\protect\expandafter\numberstringnum{%
3632 \expandafter\the\csname c@#1\endcsname}}

```

As above, but the argument is a count register or a number.

```

3633 \newcommand{\numberstringnum}[1]{%
3634 \@ifnextchar[{\@number@string{#1}}{\@number@string{#1}[m]}}

```

Gender is specified as an optional argument *at the end*.

```

3635 \def\@number@string#1[#2]{%
3636 \ifthenelse{\equal{#2}{f}}{%
3637 \protect\@numberstringF{#1}{\@fc@numstr}}{%
3638 \ifthenelse{\equal{#2}{n}}{%
3639 \protect\@numberstringN{#1}{\@fc@numstr}}{%
3640 \ifthenelse{\equal{#2}{m}}{%
3641 \PackageError{fmtcount}{Invalid gender option ‘#2’}{%
3642 Available options are m, f or n}}%
3643 \protect\@numberstringM{#1}{\@fc@numstr}}{\@fc@numstr}}

```

Store textual representation of number. First argument is identifying name, second argument is a counter.

```

3644 \newcommand{\storeNumberstring}[2]{%
3645 \expandafter\protect\expandafter\storeNumberstringnum{#1}{%
3646 \expandafter\the\csname c@#2\endcsname}}

```

As above, but second argument is a count register or number.

```

3647 \newcommand{\storeNumberstringnum}[2]{%
3648 \@ifnextchar[{\@store@Number@string{#1}{#2}}{%
3649 \@store@Number@string{#1}{#2}[m]}}

```

Gender is specified as an optional argument *at the end*:

```

3650 \def\@store@Number@string#1#2[#3]{%
3651 \ifthenelse{\equal{#3}{f}}{%
3652 \protect\@NumberstringF{#2}{\@fc@numstr}}{%
3653 \ifthenelse{\equal{#3}{n}}{%
3654 \protect\@NumberstringN{#2}{\@fc@numstr}}{%
3655 \ifthenelse{\equal{#3}{m}}{%
3656 \PackageError{fmtcount}{Invalid gender option ‘#3’}{%
3657 Available options are m, f or n}}%
3658 \protect\@NumberstringM{#2}{\@fc@numstr}}}%
3659 \expandafter\let\csname @fcs@#1\endcsname\@fc@numstr}

```

Display textual representation of number. The argument must be a counter.

```

3660 \newcommand{\Numberstring}[1]{%
3661 \expandafter\protect\expandafter\Numberstringnum{%
3662 \expandafter\the\csname c@#1\endcsname}}

```

As above, but the argument is a count register or number.

```

3663 \newcommand{\Numberstringnum}[1]{%
3664 \@ifnextchar[{\@Number@string{#1}}{\@Number@string{#1}[m]}}

```

Gender is specified as an optional argument at the end.

```

3665 \def\@Number@string#1[#2]{%
3666 \ifthenelse{\equal{#2}{f}}{%
3667 \protect\@NumberstringF{#1}{\@fc@numstr}}{%
3668 \ifthenelse{\equal{#2}{n}}{%
3669 \protect\@NumberstringN{#1}{\@fc@numstr}}{%
3670 \ifthenelse{\equal{#2}{m}}{%

```

```

3671 \PackageError{fmtcount}{Invalid gender option ‘#2’}{%
3672 Available options are m, f or n}}%
3673 \protect\@NumberstringM{#1}{\@fc@numstr}}\@fc@numstr}}

Store upper case textual representation of number. The first argument is identi-
fying name, the second argument is a counter.

3674 \newcommand{\storeNUMBERstring}[2]{%
3675 \expandafter\protect\expandafter\storeNUMBERstringnum{#1}{%
3676 \expandafter\the\csname c@#2\endcsname}}

As above, but the second argument is a count register or a number.

3677 \newcommand{\storeNUMBERstringnum}[2]{%
3678 \@ifnextchar[{\@store@NUMBER@string{#1}{#2}}{%
3679 \@store@NUMBER@string{#1}{#2}[m]}}

Gender is specified as an optional argument at the end.

3680 \def\@store@NUMBER@string#1#2[#3]{%
3681 \ifthenelse{\equal{#3}{f}}{%
3682 \protect\@numberstringF{#2}{\@fc@numstr}}{%
3683 \ifthenelse{\equal{#3}{n}}{%
3684 \protect\@numberstringN{#2}{\@fc@numstr}}{%
3685 \ifthenelse{\equal{#3}{m}}{%
3686 \PackageError{fmtcount}{Invalid gender option ‘#3’}{%
3687 Available options are m or f}}%
3688 \protect\@numberstringM{#2}{\@fc@numstr}}}%
3689 \expandafter\edef\csname @fcs@#1\endcsname{%
3690 \noexpand\MakeUppercase{\@fc@numstr}}}

Display upper case textual representation of a number. The argument must be a
counter.

3691 \newcommand{\NUMBERstring}[1]{%
3692 \expandafter\protect\expandafter\NUMBERstringnum{%
3693 \expandafter\the\csname c@#1\endcsname}}

As above, but the argument is a count register or a number.

3694 \newcommand{\NUMBERstringnum}[1]{%
3695 \@ifnextchar[{\@NUMBER@string{#1}}{\@NUMBER@string{#1}[m]}}

Gender is specified as an optional argument at the end.

3696 \def\@NUMBER@string#1#2[#3]{%
3697 \ifthenelse{\equal{#3}{f}}{%
3698 \protect\@numberstringF{#1}{\@fc@numstr}}{%
3699 \ifthenelse{\equal{#3}{n}}{%
3700 \protect\@numberstringN{#1}{\@fc@numstr}}{%
3701 \ifthenelse{\equal{#3}{m}}{%
3702 \PackageError{fmtcount}{Invalid gender option ‘#2’}{%
3703 Available options are m, f or n}}%
3704 \protect\@numberstringM{#1}{\@fc@numstr}}}%
3705 \MakeUppercase{\@fc@numstr}}}

Number representations in other bases. Binary:

3706 \providecommand{\binary}[1]{%
3707 \expandafter\protect\expandafter\@binary{%
3708 \expandafter\the\csname c@#1\endcsname}}

Like \alph, but goes beyond 26. (a ... z aa ... zz ...)

3709 \providecommand{\aaalph}[1]{%

```

```

3710 \expandafter\protect\expandafter\@aaalph{%
3711 \expandafter\the\csname c@#1\endcsname}}

As before, but upper case.

3712 \providecommand{\AAAAlph}[1]{%
3713 \expandafter\protect\expandafter\@AAAAlph{%
3714 \expandafter\the\csname c@#1\endcsname}}

Like \alph, but goes beyond 26. (a ... z ab ... az ...)

3715 \providecommand{\abalph}[1]{%
3716 \expandafter\protect\expandafter\@abalph{%
3717 \expandafter\the\csname c@#1\endcsname}}

As above, but upper case.

3718 \providecommand{\ABAlph}[1]{%
3719 \expandafter\protect\expandafter\@ABAlph{%
3720 \expandafter\the\csname c@#1\endcsname}}

Hexadecimal:

3721 \providecommand{\hexadecimal}[1]{%
3722 \expandafter\protect\expandafter\@hexadecimal{%
3723 \expandafter\the\csname c@#1\endcsname}}

As above, but in upper case.

3724 \providecommand{\Hexadecimal}[1]{%
3725 \expandafter\protect\expandafter\@Hexadecimal{%
3726 \expandafter\the\csname c@#1\endcsname}}

Octal:

3727 \providecommand{\octal}[1]{%
3728 \expandafter\protect\expandafter\@octal{%
3729 \expandafter\the\csname c@#1\endcsname}}

Decimal:

3730 \providecommand{\decimal}[1]{%
3731 \expandafter\protect\expandafter\@decimal{%
3732 \expandafter\the\csname c@#1\endcsname}}

```

11.9.1 Multilingual Definitions

If multilingual support is provided, make `\@numberstring` etc use the correct language (if defined). Otherwise use English definitions. `"setdef@ulftfmtcount"` sets the macros to use English.

```

3733 \def\@setdef@ulftfmtcount{
3734 \ifundefined{@ordinalMenglish}{\input{fc-english.def}}{}
3735 \def\@ordinalstringM{\@ordinalstringMenglish}
3736 \let\@ordinalstringF=\@ordinalstringMenglish
3737 \let\@ordinalstringN=\@ordinalstringMenglish
3738 \def\@OrdinalstringM{\@OrdinalstringMenglish}
3739 \let\@OrdinalstringF=\@OrdinalstringMenglish
3740 \let\@OrdinalstringN=\@OrdinalstringMenglish
3741 \def\@numberstringM{\@numberstringMenglish}
3742 \let\@numberstringF=\@numberstringMenglish
3743 \let\@numberstringN=\@numberstringMenglish
3744 \def\@NumberstringM{\@NumberstringMenglish}
3745 \let\@NumberstringF=\@NumberstringMenglish

```

```

3746 \let\@NumberstringN=\@NumberstringMenglish
3747 \def\@ordinalM{\@ordinalMenglish}
3748 \let\@ordinalF=\@ordinalM
3749 \let\@ordinalN=\@ordinalM
3750 }

```

Define a command to set macros to use "language":

```

3751 \def\@set@multiling@fmtcount{%
3752 %
3753 \def\@numberstringM{\@ifundefined{@numberstringM\language}{%
3754 \PackageError{fmtcount}{No support for language '\language'}{%
3755 The fmtcount package currently does not support language
3756 '\language' for command \string\@numberstringM}}{%
3757 \csname @numberstringM\language\endcsname}}%
3758 %
3759 \def\@numberstringF{\@ifundefined{@numberstringF\language}{%
3760 \PackageError{fmtcount}{No support for language '\language'}{%
3761 The fmtcount package currently does not support language
3762 '\language' for command \string\@numberstringF}}{%
3763 \csname @numberstringF\language\endcsname}}%
3764 %
3765 \def\@numberstringN{\@ifundefined{@numberstringN\language}{%
3766 \PackageError{fmtcount}{No support for language '\language'}{%
3767 The fmtcount package currently does not support language
3768 '\language' for command \string\@numberstringN}}{%
3769 \csname @numberstringN\language\endcsname}}%
3770 %
3771 \def\@NumberstringM{\@ifundefined{@NumberstringM\language}{%
3772 \PackageError{fmtcount}{No support for language '\language'}{%
3773 The fmtcount package currently does not support language
3774 '\language' for command \string\@NumberstringM}}{%
3775 \csname @NumberstringM\language\endcsname}}%
3776 %
3777 \def\@NumberstringF{\@ifundefined{@NumberstringF\language}{%
3778 \PackageError{fmtcount}{No support for language '\language'}{%
3779 The fmtcount package currently does not support language
3780 '\language' for command \string\@NumberstringF}}{%
3781 \csname @NumberstringF\language\endcsname}}%
3782 %
3783 \def\@NumberstringN{\@ifundefined{@NumberstringN\language}{%
3784 \PackageError{fmtcount}{No support for language '\language'}{%
3785 The fmtcount package currently does not support language
3786 '\language' for command \string\@NumberstringN}}{%
3787 \csname @NumberstringN\language\endcsname}}%
3788 %
3789 \def\@ordinalM{\@ifundefined{@ordinalM\language}{%
3790 \PackageError{fmtcount}{No support for language '\language'}{%
3791 The fmtcount package currently does not support language
3792 '\language' for command \string\@ordinalM}}{%
3793 \csname @ordinalM\language\endcsname}}%
3794 %
3795 \def\@ordinalF{\@ifundefined{@ordinalF\language}{%
3796 \PackageError{fmtcount}{No support for language '\language'}{%
3797 The fmtcount package currently does not support language
3798 '\language' for command \string\@ordinalF}}{%

```

```

3799 \csname @ordinalF\language\endcsname}}%
3800 %
3801 \def\@ordinalN{\@ifundefined{@ordinalN\language}{%
3802 \PackageError{fmtcount}{No support for language '\language'}{%
3803 The fmtcount package currently does not support language
3804 '\language' for command \string\@ordinalN}}{%
3805 \csname @ordinalN\language\endcsname}}%
3806 %
3807 \def\@ordinalstringM{\@ifundefined{@ordinalstringM\language}{%
3808 \PackageError{fmtcount}{No support for language '\language'}{%
3809 The fmtcount package currently does not support language
3810 '\language' for command \string\@ordinalstringM}}{%
3811 \csname @ordinalstringM\language\endcsname}}%
3812 %
3813 \def\@ordinalstringF{\@ifundefined{@ordinalstringF\language}{%
3814 \PackageError{fmtcount}{No support for language '\language'}{%
3815 The fmtcount package currently does not support language
3816 '\language' for command \string\@ordinalstringF}}{%
3817 \csname @ordinalstringF\language\endcsname}}%
3818 %
3819 \def\@ordinalstringN{\@ifundefined{@ordinalstringN\language}{%
3820 \PackageError{fmtcount}{No support for language '\language'}{%
3821 The fmtcount package currently does not support language
3822 '\language' for command \string\@ordinalstringN}}{%
3823 \csname @ordinalstringN\language\endcsname}}%
3824 %
3825 \def\@OrdinalstringM{\@ifundefined{@OrdinalstringM\language}{%
3826 \PackageError{fmtcount}{No support for language '\language'}{%
3827 The fmtcount package currently does not support language
3828 '\language' for command \string\@OrdinalstringM}}{%
3829 \csname @OrdinalstringM\language\endcsname}}%
3830 %
3831 \def\@OrdinalstringF{\@ifundefined{@OrdinalstringF\language}{%
3832 \PackageError{fmtcount}{No support for language '\language'}{%
3833 The fmtcount package currently does not support language
3834 '\language' for command \string\@OrdinalstringF}}{%
3835 \csname @OrdinalstringF\language\endcsname}}%
3836 %
3837 \def\@OrdinalstringN{\@ifundefined{@OrdinalstringN\language}{%
3838 \PackageError{fmtcount}{No support for language '\language'}{%
3839 The fmtcount package currently does not support language
3840 '\language' for command \string\@OrdinalstringN}}{%
3841 \csname @OrdinalstringN\language\endcsname}}
3842 }

```

Check to see if babel or ngerman packages have been loaded.

```

3843 \@ifpackageloaded{babel}{%
3844 \ifthenelse{equal{\language}{nohyphenation}}\or
3845 \equal{\language}{english}}{\@setdef@ultfmtcount}{%
3846 \@set@multiling@fmtcount}
3847 }{%
3848 \@ifpackageloaded{ngerman}{%
3849 \@ifundefined{numberstringMgerman}{%
3850 \input{fc-german.def}}{\@set@multiling@fmtcount}{%
3851 \@setdef@ultfmtcount}}

```


Backwards compatibility:

```
3852 \let\@ordinal=\@ordinalM
3853 \let\@ordinalstring=\@ordinalstringM
3854 \let\@Ordinalstring=\@OrdinalstringM
3855 \let\@numberstring=\@numberstringM
3856 \let\@Numberstring=\@NumberstringM
```