fmtcount.sty v1.1: Displaying the Values of LaTeX Counters

Nicola L.C. Talbot

26th May 2007

Contents

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` `\fmtord`

The macro `\ordinal{`⟨*counter*⟩`}` will print the value of a LaTeX counter ⟨*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 incompatiblity, 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` `\Numberstring`

The macro `\numberstring{`⟨*counter*⟩`}` will print the value of ⟨*counter*⟩ as text. E.g. `\numberstring{section}` will produce: three. The macro `\Numberstring{`⟨*counter*⟩`}` does the same as `\numberstring`, but with initial letters in uppercase. For example, `\Numberstring{section}` will produce: Three.

`\NUMBERstring`

The macro `\NUMBERstring{`⟨*counter*⟩`}` does the same as `\numberstring`, but converted to upper case. Note that `\MakeUppercase{\NUMBERstring{`⟨*counter*⟩`}}` doesn't work, due to the way that `\MakeUppercase` expands its argument[1].

`\numberstringnum` `\Numberstringnum` `\NUMBERstringnum`

The macros `\numberstringnum`, `\Numberstringnum` and `\NUMBERstringnum` work like `\numberstring`, `\Numberstring` and `\NUMBERstring`, respectively, but 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*⟩

[1] See 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

`\ORDINALstring` 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`
`\Ordinalstringnum` work like `\ordinalstring`, `\Ordinalstring` and `\ORDINALstring`, respectively,
`\ORDINALstringnum` 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

`\FMCuse` 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` `\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}`

will produce: 7d. Again, the number will be padded with zeroes if necessary, depending on whether `\padzeroes` has been used. `\Hexadecimal{`⟨*counter*⟩`}` does the same thing, but uses uppercase characters, e.g. `\Hexadecimal{mycounter}` will produce: 7D.

`\hexadecimalnum` `\Hexadecimalnum` The macros `\hexadecimalnum` and `\Hexadecimalnum` are like `\hexadecimal` and `\Hexadecimal` but take an actual number rather than a counter as the argument. For example: `\hexadecimalnum{125}` will produce: 7d, and `\Hexadecimalnum{125}` will produce: 7D.

`\decimal` The macro `\decimal{`⟨*counter*⟩`}` is similar to `\arabic` but the number can be padded with zeroes depending on whether `\padzeroes` has been used. For example: `\padzeroes[8]\decimal{section}` will produce: 00000005.

`\decimalnum` The macro `\decimalnum` is like `\decimal` but takes an actual number rather than a counter as the argument. For example: `\padzeroes[8]\decimalnum{5}` will produce: 00000005.

`\aaalph` The macro `\aaalph{`⟨*counter*⟩`}` will print the value of ⟨*counter*⟩ as: a b ... z aa bb ... zz etc. For example, `\aaalpha{mycounter}` will produce: uuuuu if `mycounter` is set to 125. `\AAAlph{`⟨*counter*⟩`}` does the same thing, but uses uppercase characters, e.g. `\AAAlph{mycounter}` will produce: UUUUU.

`\aaalphnum` `\AAAlphnum` The macros `\aaalphnum` and `\AAAlphnum` are like `\aaalph` and `\AAAlph` but take an actual number rather than a counter as the argument. For example: `\aaalphnum{125}` will produce: uuuuu, and `\AAAlphnum{125}` will produce: UUUUU.

`\abalph` The macro `\abalph{`⟨*counter*⟩`}` will print the value of ⟨*counter*⟩ as: a b ... z aa ab ... az etc. For example, `\abalpha{mycounter}` will produce: du if `mycounter` is set to 125. `\ABAlph{`⟨*counter*⟩`}` does the same thing, but uses uppercase characters, e.g. `\ABAlph{mycounter}` will produce: DU.

`\abalphnum` `\ABAlphnum` The macros `\abalphnum` and `\ABAlphnum` are like `\abalph` and `\ABAlph` but take an actual number rather than a counter as the argument. For example: `\abalphnum{125}` will produce: du, and `\ABAlphnum{125}` will produce: DU.

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 1.1[2].

The package checks to see if the command `\date`⟨*language*⟩ is defined[3], and will load the code for those languages. The commands `\ordinal`, `\ordinalstring`

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

[3] this 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. soxiante-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

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

Check that fc-english.def has been loaded

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

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

```
\let\@ordinalMbritish\@ordinalMenglish
\let\@ordinalFbritish\@ordinalMenglish
\let\@ordinalNbritish\@ordinalMenglish
\let\@numberstringMbritish\@numberstringMenglish
\let\@numberstringFbritish\@numberstringMenglish
\let\@numberstringNbritish\@numberstringMenglish
\let\@NumberstringMbritish\@NumberstringMenglish
\let\@NumberstringFbritish\@NumberstringMenglish
\let\@NumberstringNbritish\@NumberstringMenglish
\let\@ordinalstringMbritish\@ordinalstringMenglish
\let\@ordinalstringFbritish\@ordinalstringMenglish
\let\@ordinalstringNbritish\@ordinalstringMenglish
\let\@OrdinalstringMbritish\@OrdinalstringMenglish
\let\@OrdinalstringFbritish\@OrdinalstringMenglish
\let\@OrdinalstringNbritish\@OrdinalstringMenglish
```

11.2 fc-english.def

English definitions

```
\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.

```
\newcommand*{\@ordinalMenglish}[2]{%
\def\@fc@ord{}%
\@orgargctr=#1\relax
\@ordinalctr=#1%
\@modulo{\@ordinalctr}{100}%
\ifnum\@ordinalctr=11\relax
  \def\@fc@ord{th}%
\else
  \ifnum\@ordinalctr=12\relax
    \def\@fc@ord{th}%
  \else
    \ifnum\@ordinalctr=13\relax
      \def\@fc@ord{th}%
    \else
      \@modulo{\@ordinalctr}{10}%
      \ifcase\@ordinalctr
        \def\@fc@ord{th}%      case 0
        \or \def\@fc@ord{st}%  case 1
        \or \def\@fc@ord{nd}%  case 2
        \or \def\@fc@ord{rd}%  case 3
      \else
        \def\@fc@ord{th}%      default case
      \fi
    \fi
  \fi
\fi
\edef#2{\number#1\relax\noexpand\fmtord{\@fc@ord}}%
}
```

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

```
\let\@ordinalFenglish=\@ordinalMenglish
\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.

```
\newcommand*{\@@unitstringenglish}[1]{%
\ifcase#1\relax
zero%
\or one%
\or two%
\or three%
\or four%
\or five%
\or six%
\or seven%
\or eight%
\or nine%
\fi
}
```

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

```
\newcommand*{\@@tenstringenglish}[1]{%
\ifcase#1\relax
\or ten%
\or twenty%
\or thirty%
\or forty%
\or fifty%
\or sixty%
\or seventy%
\or eighty%
\or ninety%
\fi
}
```

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

```
\newcommand*{\@@teenstringenglish}[1]{%
\ifcase#1\relax
ten%
\or eleven%
\or twelve%
\or thirteen%
\or fourteen%
\or fifteen%
\or sixteen%
\or seventeen%
\or eighteen%
\or nineteen%
\fi
}
```

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

```
\newcommand*{\@@Unitstringenglish}[1]{%
\ifcase#1\relax
Zero%
\or One%
\or Two%
\or Three%
\or Four%
\or Five%
\or Six%
\or Seven%
\or Eight%
\or Nine%
\fi
}
```

The tens:

```
\newcommand*{\@@Tenstringenglish}[1]{%
\ifcase#1\relax
\or Ten%
\or Twenty%
\or Thirty%
\or Forty%
\or Fifty%
```

```
\or Sixty%
\or Seventy%
\or Eighty%
\or Ninety%
\fi
}
```

The teens:

```
\newcommand*{\@@Teenstringenglish}[1]{%
\ifcase#1\relax
Ten%
\or Eleven%
\or Twelve%
\or Thirteen%
\or Fourteen%
\or Fifteen%
\or Sixteen%
\or Seventeen%
\or Eighteen%
\or Nineteen%
\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.)

```
\newcommand*{\@@numberstringenglish}[2]{%
\ifnum#1>99999
\PackageError{fmtcount}{Out of range}%
{This macro only works for values less than 100000}%
\else
\ifnum#1<0
\PackageError{fmtcount}{Negative numbers not permitted}%
{This macro does not work for negative numbers, however
you can try typing "minus" first, and then pass the modulus of
this number}%
\fi
\fi
\def#2{}%
\@strctr=#1\relax \divide\@strctr by 1000\relax
\ifnum\@strctr>9
% #1 is greater or equal to 10000
  \divide\@strctr by 10
  \ifnum\@strctr>1\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
    \@strctr=#1 \divide\@strctr by 1000\relax
    \@modulo{\@strctr}{10}%
    \ifnum\@strctr>0\relax
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr-\@unitstring{\@strctr}}%
    \fi
  \else
    \@strctr=#1\relax
```

```
    \divide\@strctr by 1000\relax
    \@modulo{\@strctr}{10}%
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
  \fi
  \let\@@fc@numstr#2\relax
  \edef#2{\@@fc@numstr\ \@thousand}%
\else
  \ifnum\@strctr>0\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@unitstring{\@strctr}\ \@thousand}%
  \fi
\fi
\@strctr=#1\relax \@modulo{\@strctr}{1000}%
\divide\@strctr by 100
\ifnum\@strctr>0\relax
   \ifnum#1>1000\relax
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\ }%
   \fi
   \let\@@fc@numstr#2\relax
   \edef#2{\@@fc@numstr\@unitstring{\@strctr}\ \@hundred}%
\fi
\@strctr=#1\relax \@modulo{\@strctr}{100}%
\ifnum#1>100\relax
  \ifnum\@strctr>0\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\ \@andname\ }%
  \fi
\fi
\ifnum\@strctr>19\relax
  \divide\@strctr by 10\relax
  \let\@@fc@numstr#2\relax
  \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
  \@strctr=#1\relax \@modulo{\@strctr}{10}%
  \ifnum\@strctr>0\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr-\@unitstring{\@strctr}}%
  \fi
\else
  \ifnum\@strctr<10\relax
    \ifnum\@strctr=0\relax
       \ifnum#1<100\relax
          \let\@@fc@numstr#2\relax
          \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
       \fi
    \else
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
    \fi
  \else
    \@modulo{\@strctr}{10}%
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
```

```
  \fi
\fi
}
```

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

```
\DeclareRobustCommand{\@numberstringMenglish}[2]{%
\let\@unitstring=\@@unitstringenglish
\let\@teenstring=\@@teenstringenglish
\let\@tenstring=\@@tenstringenglish
\def\@hundred{hundred}\def\@thousand{thousand}%
\def\@andname{and}%
\@@numberstringenglish{#1}{#2}%
}
```

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

```
\let\@numberstringFenglish=\@numberstringMenglish
\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.

```
\newcommand*{\@NumberstringMenglish}[2]{%
\let\@unitstring=\@@Unitstringenglish
\let\@teenstring=\@@Teenstringenglish
\let\@tenstring=\@@Tenstringenglish
\def\@hundred{Hundred}\def\@thousand{Thousand}%
\def\@andname{and}%
\@@numberstringenglish{#1}{#2}}
```

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

```
\let\@NumberstringFenglish=\@NumberstringMenglish
\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:

```
\newcommand*{\@@unitthstringenglish}[1]{%
\ifcase#1\relax
zeroth%
\or first%
\or second%
\or third%
\or fourth%
\or fifth%
\or sixth%
\or seventh%
\or eighth%
\or ninth%
\fi
}
```

Next the tens:

```
\newcommand*{\@@tenthstringenglish}[1]{%
\ifcase#1\relax
\or tenth%
\or twentieth%
```

```
\or thirtieth%
\or fortieth%
\or fiftieth%
\or sixtieth%
\or seventieth%
\or eightieth%
\or ninetieth%
\fi
}
%   \end{macrocode}
% The teens:
%   \begin{macrocode}
\newcommand*{\@@teenthstringenglish}[1]{%
\ifcase#1\relax
tenth%
\or eleventh%
\or twelfth%
\or thirteenth%
\or fourteenth%
\or fifteenth%
\or sixteenth%
\or seventeenth%
\or eighteenth%
\or nineteenth%
\fi
}
%   \end{macrocode}
% As before, but with the first letter in upper case. The units:
%   \begin{macrocode}
\newcommand*{\@@Unitthstringenglish}[1]{%
\ifcase#1\relax
Zeroth%
\or First%
\or Second%
\or Third%
\or Fourth%
\or Fifth%
\or Sixth%
\or Seventh%
\or Eighth%
\or Ninth%
\fi
}
```

The tens:

```
\newcommand*{\@@Tenthstringenglish}[1]{%
\ifcase#1\relax
\or Tenth%
\or Twentieth%
\or Thirtieth%
\or Fortieth%
\or Fiftieth%
\or Sixtieth%
\or Seventieth%
\or Eightieth%
```

```
\or Ninetieth%
\fi
}
```

The teens:

```
\newcommand*{\@@Teenthstringenglish}[1]{%
\ifcase#1\relax
Tenth%
\or Eleventh%
\or Twelfth%
\or Thirteenth%
\or Fourteenth%
\or Fifteenth%
\or Sixteenth%
\or Seventeenth%
\or Eighteenth%
\or Nineteenth%
\fi
}
```

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.

```
\newcommand*{\@@ordinalstringenglish}[2]{%
\@strctr=#1\relax
\ifnum#1>99999
\PackageError{fmtcount}{Out of range}%
{This macro only works for values less than 100000 (value given: \number\@strctr)}%
\else
\ifnum#1<0
\PackageError{fmtcount}{Negative numbers not permitted}%
{This macro does not work for negative numbers, however
you can try typing "minus" first, and then pass the modulus of
this number}%
\fi
\def#2{}%
\fi
\@strctr=#1\relax \divide\@strctr by 1000\relax
\ifnum\@strctr>9\relax
% #1 is greater or equal to 10000
  \divide\@strctr by 10
  \ifnum\@strctr>1\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\@tenstring{\@strctr}}%
    \@strctr=#1\relax
    \divide\@strctr by 1000\relax
    \@modulo{\@strctr}{10}%
    \ifnum\@strctr>0\relax
      \let\@@fc@ordstr#2\relax
      \edef#2{\@@fc@ordstr-\@unitstring{\@strctr}}%
    \fi
  \else
    \@strctr=#1\relax \divide\@strctr by 1000\relax
    \@modulo{\@strctr}{10}%
    \let\@@fc@ordstr#2\relax
```

```
    \edef#2{\@@fc@ordstr\@teenstring{\@strctr}}%
  \fi
  \@strctr=#1\relax \@modulo{\@strctr}{1000}%
  \ifnum\@strctr=0\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\ \@thousandth}%
  \else
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\ \@thousand}%
  \fi
\else
  \ifnum\@strctr>0\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
    \@strctr=#1\relax \@modulo{\@strctr}{1000}%
    \let\@@fc@ordstr#2\relax
    \ifnum\@strctr=0\relax
      \edef#2{\@@fc@ordstr\ \@thousandth}%
    \else
      \edef#2{\@@fc@ordstr\ \@thousand}%
    \fi
  \fi
\fi
\@strctr=#1\relax \@modulo{\@strctr}{1000}%
\divide\@strctr by 100
\ifnum\@strctr>0\relax
  \ifnum#1>1000\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\ }%
  \fi
  \let\@@fc@ordstr#2\relax
  \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
  \@strctr=#1\relax \@modulo{\@strctr}{100}%
  \let\@@fc@ordstr#2\relax
  \ifnum\@strctr=0\relax
    \edef#2{\@@fc@ordstr\ \@hundredth}%
  \else
    \edef#2{\@@fc@ordstr\ \@hundred}%
  \fi
\fi
\@strctr=#1\relax \@modulo{\@strctr}{100}%
\ifnum#1>100\relax
  \ifnum\@strctr>0\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\ \@andname\ }%
  \fi
\fi
\ifnum\@strctr>19\relax
  \@tmpstrctr=\@strctr
  \divide\@strctr by 10\relax
  \@modulo{\@tmpstrctr}{10}%
  \let\@@fc@ordstr#2\relax
  \ifnum\@tmpstrctr=0\relax
    \edef#2{\@@fc@ordstr\@tenthstring{\@strctr}}%
```

```
  \else
    \edef#2{\@@fc@ordstr\@tenstring{\@strctr}}%
  \fi
  \@strctr=#1\relax \@modulo{\@strctr}{10}%
  \ifnum\@strctr>0\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr-\@unitthstring{\@strctr}}%
  \fi
\else
  \ifnum\@strctr<10\relax
    \ifnum\@strctr=0\relax
      \ifnum#1<100\relax
        \let\@@fc@ordstr#2\relax
        \edef#2{\@@fc@ordstr\@unitthstring{\@strctr}}%
      \fi
    \else
      \let\@@fc@ordstr#2\relax
      \edef#2{\@@fc@ordstr\@unitthstring{\@strctr}}%
    \fi
  \else
    \@modulo{\@strctr}{10}%
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\@teenthstring{\@strctr}}%
  \fi
\fi
}
```

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

```
\DeclareRobustCommand{\@ordinalstringMenglish}[2]{%
\let\@unitthstring=\@@unitthstringenglish
\let\@teenthstring=\@@teenthstringenglish
\let\@tenthstring=\@@tenthstringenglish
\let\@unitstring=\@@unitstringenglish
\let\@teenstring=\@@teenstringenglish
\let\@tenstring=\@@tenstringenglish
\def\@andname{and}%
\def\@hundred{hundred}\def\@thousand{thousand}%
\def\@hundredth{hundredth}\def\@thousandth{thousandth}%
\@@ordinalstringenglish{#1}{#2}}
```

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

```
\let\@ordinalstringFenglish=\@ordinalstringMenglish
\let\@ordinalstringNenglish=\@ordinalstringMenglish
```

First letter of each word in upper case:

```
\DeclareRobustCommand{\@OrdinalstringMenglish}[2]{%
\let\@unitthstring=\@@Unitthstringenglish
\let\@teenthstring=\@@Teenthstringenglish
\let\@tenthstring=\@@Tenthstringenglish
\let\@unitstring=\@@Unitstringenglish
\let\@teenstring=\@@Teenstringenglish
\let\@tenstring=\@@Tenstringenglish
\def\@andname{and}%
\def\@hundred{Hundred}\def\@thousand{Thousand}%
```

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

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

```
\let\@OrdinalstringFenglish=\@OrdinalstringMenglish
\let\@OrdinalstringNenglish=\@OrdinalstringMenglish
```

11.3 fc-french.def

French definitions

```
\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:

```
\newcommand*{\@ordinalMfrench}[2]{%
\iffmtord@abbrv
  \edef#2{\number#1\relax\noexpand\fmtord{e}}%
\else
  \ifnum#1=1\relax
    \edef#2{\number#1\relax\noexpand\fmtord{er}}%
  \else
    \edef#2{\number#1\relax\noexpand\fmtord{eme}}%
  \fi
\fi}
```

Feminine:

```
\newcommand*{\@ordinalFfrench}[2]{%
\iffmtord@abbrv
  \edef#2{\number#1\relax\noexpand\fmtord{e}}%
\else
  \ifnum#1=1\relax
     \edef#2{\number#1\relax\noexpand\fmtord{ere}}%
  \else
     \edef#2{\number#1\relax\noexpand\fmtord{eme}}%
  \fi
\fi}
```

Make neuter same as masculine:

```
\let\@ordinalNfrench\@ordinalMfrench
```

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

```
\newcommand*{\@@unitstringfrench}[1]{%
\ifcase#1\relax
zero%
\or un%
\or deux%
\or trois%
\or quatre%
\or cinq%
\or six%
\or sept%
\or huit%
\or neuf%
```

```
\fi
}
```

Feminine only changes for 1:

```
\newcommand*{\@@unitstringFfrench}[1]{%
\ifnum#1=1\relax
une%
\else\@@unitstringfrench{#1}%
\fi
}
```

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

```
\newcommand*{\@@tenstringfrench}[1]{%
\ifcase#1\relax
\or dix%
\or vingt%
\or trente%
\or quarante%
\or cinquante%
\or soixante%
\or septente%
\or huitante%
\or nonente%
\or cent%
\fi
}
```

Teens:

```
\newcommand*{\@@teenstringfrench}[1]{%
\ifcase#1\relax
dix%
\or onze%
\or douze%
\or treize%
\or quatorze%
\or quinze%
\or seize%
\or dix-sept%
\or dix-huit%
\or dix-neuf%
\fi
}
```

Seventies are a special case, depending on dialect:

```
\newcommand*{\@@seventiesfrench}[1]{%
\@tenstring{6}%
\ifnum#1=1\relax
\ \@andname\
\else
-%
\fi
\@teenstring{#1}%
}
```

Eighties are a special case, depending on dialect:

```
\newcommand*{\@@eightiesfrench}[1]{%
\@unitstring{4}-\@tenstring{2}%
\ifnum#1>0
-\@unitstring{#1}%
\else
s%
\fi
}
```

Nineties are a special case, depending on dialect:

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

Swiss seventies:

```
\newcommand*{\@@seventiesfrenchswiss}[1]{%
\@tenstring{7}%
\ifnum#1=1\ \@andname\ \fi
\ifnum#1>1-\fi
\ifnum#1>0\@unitstring{#1}\fi
}
```

Swiss eighties:

```
\newcommand*{\@@eightiesfrenchswiss}[1]{%
\@tenstring{8}%
\ifnum#1=1\ \@andname\ \fi
\ifnum#1>1-\fi
\ifnum#1>0\@unitstring{#1}\fi
}
```

Swiss nineties:

```
\newcommand*{\@@ninetiesfrenchswiss}[1]{%
\@tenstring{9}%
\ifnum#1=1\ \@andname\ \fi
\ifnum#1>1-\fi
\ifnum#1>0\@unitstring{#1}\fi
}
```

Units with initial letter in upper case:

```
\newcommand*{\@@Unitstringfrench}[1]{%
\ifcase#1\relax
Zero%
\or Un%
\or Deux%
\or Trois%
\or Quatre%
\or Cinq%
\or Six%
\or Sept%
\or Huit%
\or Neuf%
\fi
}
```

As above, but feminine:

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

```
\ifnum#1=1\relax
Une%
\else \@@Unitstringfrench{#1}%
\fi
}
```

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

```
\newcommand*{\@@Tenstringfrench}[1]{%
\ifcase#1\relax
\or Dix%
\or Vingt%
\or Trente%
\or Quarante%
\or Cinquante%
\or Soixante%
\or Septente%
\or Huitante%
\or Nonente%
\or Cent%
\fi
}
```

Teens, with initial letter in upper case:

```
\newcommand*{\@@Teenstringfrench}[1]{%
\ifcase#1\relax
Dix%
\or Onze%
\or Douze%
\or Treize%
\or Quatorze%
\or Quinze%
\or Seize%
\or Dix-Sept%
\or Dix-Huit%
\or Dix-Neuf%
\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 defined for use in documents.) Firstly, the Swiss version:

```
\DeclareRobustCommand{\@numberstringMfrenchswiss}[2]{%
\let\@unitstring=\@@unitstringfrench
\let\@teenstring=\@@teenstringfrench
\let\@tenstring=\@@tenstringfrench
\let\@seventies=\@@seventiesfrenchswiss
\let\@eighties=\@@eightiesfrenchswiss
\let\@nineties=\@@ninetiesfrenchswiss
\def\@hundred{cent}\def\@thousand{mille}%
\def\@andname{et}%
\@@numberstringfrench{#1}{#2}}
```

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

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

```
\let\@unitstring=\@@unitstringfrench
\let\@teenstring=\@@teenstringfrench
\let\@tenstring=\@@tenstringfrench
\let\@seventies=\@@seventiesfrench
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\def\@hundred{cent}\def\@thousand{mille}%
\def\@andname{et}%
\@@numberstringfrench{#1}{#2}}
```

Same as above, but for Belgian dialect:

```
\DeclareRobustCommand{\@numberstringMfrenchbelgian}[2]{%
\let\@unitstring=\@@unitstringfrench
\let\@teenstring=\@@teenstringfrench
\let\@tenstring=\@@tenstringfrench
\let\@seventies=\@@seventiesfrenchswiss
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\def\@hundred{cent}\def\@thousand{mille}%
\def\@andname{et}%
\@@numberstringfrench{#1}{#2}}
```

Set default dialect:

```
\let\@numberstringMfrench=\@numberstringMfrenchfrance
```

As above, but for feminine version. Swiss:

```
\DeclareRobustCommand{\@numberstringFfrenchswiss}[2]{%
\let\@unitstring=\@@unitstringFfrench
\let\@teenstring=\@@teenstringfrench
\let\@tenstring=\@@tenstringfrench
\let\@seventies=\@@seventiesfrenchswiss
\let\@eighties=\@@eightiesfrenchswiss
\let\@nineties=\@@ninetiesfrenchswiss
\def\@hundred{cent}\def\@thousand{mille}%
\def\@andname{et}%
\@@numberstringfrench{#1}{#2}}
```

French:

```
\DeclareRobustCommand{\@numberstringFfrenchfrance}[2]{%
\let\@unitstring=\@@unitstringFfrench
\let\@teenstring=\@@teenstringfrench
\let\@tenstring=\@@tenstringfrench
\let\@seventies=\@@seventiesfrench
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\def\@hundred{cent}\def\@thousand{mille}%
\def\@andname{et}%
\@@numberstringfrench{#1}{#2}}
```

Belgian:

```
\DeclareRobustCommand{\@numberstringFfrenchbelgian}[2]{%
\let\@unitstring=\@@unitstringFfrench
\let\@teenstring=\@@teenstringfrench
\let\@tenstring=\@@tenstringfrench
\let\@seventies=\@@seventiesfrenchswiss
\let\@eighties=\@@eightiesfrench
```

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

Set default dialect:

```
\let\@numberstringFfrench=\@numberstringFfrenchfrance
```

Make neuter same as masculine:

```
\let\@ordinalstringNfrench\@ordinalstringMfrench
```

As above, but with initial letter in upper case. Swiss (masculine):

```
\DeclareRobustCommand{\@NumberstringMfrenchswiss}[2]{%
\let\@unitstring=\@@Unitstringfrench
\let\@teenstring=\@@Teenstringfrench
\let\@tenstring=\@@Tenstringfrench
\let\@seventies=\@@seventiesfrenchswiss
\let\@eighties=\@@eightiesfrenchswiss
\let\@nineties=\@@ninetiesfrenchswiss
\def\@hundred{Cent}\def\@thousand{Mille}%
\def\@andname{et}%
\@@numberstringfrench{#1}{#2}}
```

French:

```
\DeclareRobustCommand{\@NumberstringMfrenchfrance}[2]{%
\let\@unitstring=\@@Unitstringfrench
\let\@teenstring=\@@Teenstringfrench
\let\@tenstring=\@@Tenstringfrench
\let\@seventies=\@@seventiesfrench
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\def\@hundred{Cent}\def\@thousand{Mille}%
\def\@andname{et}%
\@@numberstringfrench{#1}{#2}}
```

Belgian:

```
\DeclareRobustCommand{\@NumberstringMfrenchbelgian}[2]{%
\let\@unitstring=\@@Unitstringfrench
\let\@teenstring=\@@Teenstringfrench
\let\@tenstring=\@@Tenstringfrench
\let\@seventies=\@@seventiesfrenchswiss
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\def\@hundred{Cent}\def\@thousand{Mille}%
\def\@andname{et}%
\@@numberstringfrench{#1}{#2}}
```

Set default dialect:

```
\let\@NumberstringMfrench=\@NumberstringMfrenchfrance
```

As above, but feminine. Swiss:

```
\DeclareRobustCommand{\@NumberstringFfrenchswiss}[2]{%
\let\@unitstring=\@@UnitstringFfrench
\let\@teenstring=\@@Teenstringfrench
\let\@tenstring=\@@Tenstringfrench
\let\@seventies=\@@seventiesfrenchswiss
\let\@eighties=\@@eightiesfrenchswiss
```

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

French (feminine):

```
\DeclareRobustCommand{\@NumberstringFfrenchfrance}[2]{%
\let\@unitstring=\@@UnitstringFfrench
\let\@teenstring=\@@Teenstringfrench
\let\@tenstring=\@@Tenstringfrench
\let\@seventies=\@@seventiesfrench
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\def\@hundred{Cent}\def\@thousand{Mille}%
\def\@andname{et}%
\@@numberstringfrench{#1}{#2}}
```

Belgian (feminine):

```
\DeclareRobustCommand{\@NumberstringFfrenchbelgian}[2]{%
\let\@unitstring=\@@UnitstringFfrench
\let\@teenstring=\@@Teenstringfrench
\let\@tenstring=\@@Tenstringfrench
\let\@seventies=\@@seventiesfrenchswiss
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\def\@hundred{Cent}\def\@thousand{Mille}%
\def\@andname{et}%
\@@numberstringfrench{#1}{#2}}
```

Set default dialect:

```
\let\@NumberstringFfrench=\@NumberstringFfrenchfrance
```

Make neuter same as masculine:

```
\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 (masculine):

```
\DeclareRobustCommand{\@ordinalstringMfrenchswiss}[2]{%
\ifnum#1=1\relax
\def#2{premier}%
\else
\let\@unitthstring=\@@unitthstringfrench
\let\@unitstring=\@@unitstringfrench
\let\@teenthstring=\@@teenthstringfrench
\let\@teenstring=\@@teenstringfrench
\let\@tenthstring=\@@tenthstringfrench
\let\@tenstring=\@@tenstringfrench
\let\@seventieths=\@@seventiethsfrenchswiss
\let\@eightieths=\@@eightiethsfrenchswiss
\let\@ninetieths=\@@ninetiethsfrenchswiss
\let\@seventies=\@@seventiesfrenchswiss
\let\@eighties=\@@eightiesfrenchswiss
\let\@nineties=\@@ninetiesfrenchswiss
\def\@hundredth{centi\‘eme}\def\@hundred{cent}%
```

```
\def\@thousandth{mili\‘eme}\def\@thousand{mille}%
\def\@andname{et}%
\@@ordinalstringfrench{#1}{#2}%
\fi}
```

French (masculine):

```
\DeclareRobustCommand{\@ordinalstringMfrenchfrance}[2]{%
\ifnum#1=1\relax
\def#2{premier}%
\else
\let\@unitthstring=\@@unitthstringfrench
\let\@unitstring=\@@unitstringfrench
\let\@teenthstring=\@@teenthstringfrench
\let\@teenstring=\@@teenstringfrench
\let\@tenthstring=\@@tenthstringfrench
\let\@tenstring=\@@tenstringfrench
\let\@seventieths=\@@seventiethsfrench
\let\@eightieths=\@@eightiethsfrench
\let\@ninetieths=\@@ninetiethsfrench
\let\@seventies=\@@seventiesfrench
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\let\@teenstring=\@@teenstringfrench
\def\@hundredth{centi\‘eme}\def\@hundred{cent}%
\def\@thousandth{mili\‘eme}\def\@thousand{mille}%
\def\@andname{et}%
\@@ordinalstringfrench{#1}{#2}%
\fi}
```

Belgian dialect (masculine):

```
\DeclareRobustCommand{\@ordinalstringMfrenchbelgian}[2]{%
\ifnum#1=1\relax
\def#2{premier}%
\else
\let\@unitthstring=\@@unitthstringfrench
\let\@unitstring=\@@unitstringfrench
\let\@teenthstring=\@@teenthstringfrench
\let\@teenstring=\@@teenstringfrench
\let\@tenthstring=\@@tenthstringfrench
\let\@tenstring=\@@tenstringfrench
\let\@seventieths=\@@seventiethsfrenchswiss
\let\@eightieths=\@@eightiethsfrench
\let\@ninetieths=\@@ninetiethsfrenchswiss
\let\@seventies=\@@seventiesfrench
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\let\@teenstring=\@@teenstringfrench
\def\@hundredth{centi\‘eme}\def\@hundred{cent}%
\def\@thousandth{mili\‘eme}\def\@thousand{mille}%
\def\@andname{et}%
\@@ordinalstringfrench{#1}{#2}%
\fi}
```

Set up default dialect:

```
\let\@ordinalstringMfrench=\@ordinalstringMfrenchfrance
```

As above, but feminine. Swiss:

```
\DeclareRobustCommand{\@ordinalstringFfrenchswiss}[2]{%
\ifnum#1=1\relax
\def#2{premi\‘ere}%
\else
\let\@unitthstring=\@@unitthstringfrench
\let\@unitstring=\@@unitstringFfrench
\let\@teenthstring=\@@teenthstringfrench
\let\@teenstring=\@@teenstringfrench
\let\@tenthstring=\@@tenthstringfrench
\let\@tenstring=\@@tenstringfrench
\let\@seventieths=\@@seventiethsfrenchswiss
\let\@eightieths=\@@eightiethsfrenchswiss
\let\@ninetieths=\@@ninetiethsfrenchswiss
\let\@seventies=\@@seventiesfrenchswiss
\let\@eighties=\@@eightiesfrenchswiss
\let\@nineties=\@@ninetiesfrenchswiss
\def\@hundredth{centi\‘eme}\def\@hundred{cent}%
\def\@thousandth{mili\‘eme}\def\@thousand{mille}%
\def\@andname{et}%
\@@ordinalstringfrench{#1}{#2}%
\fi}
```

French (feminine):

```
\DeclareRobustCommand{\@ordinalstringFfrenchfrance}[2]{%
\ifnum#1=1\relax
\def#2{premi\‘ere}%
\else
\let\@unitthstring=\@@unitthstringfrench
\let\@unitstring=\@@unitstringFfrench
\let\@teenthstring=\@@teenthstringfrench
\let\@teenstring=\@@teenstringfrench
\let\@tenthstring=\@@tenthstringfrench
\let\@tenstring=\@@tenstringfrench
\let\@seventieths=\@@seventiethsfrench
\let\@eightieths=\@@eightiethsfrench
\let\@ninetieths=\@@ninetiethsfrench
\let\@seventies=\@@seventiesfrench
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\let\@teenstring=\@@teenstringfrench
\def\@hundredth{centi\‘eme}\def\@hundred{cent}%
\def\@thousandth{mili\‘eme}\def\@thousand{mille}%
\def\@andname{et}%
\@@ordinalstringfrench{#1}{#2}%
\fi}
```

Belgian (feminine):

```
\DeclareRobustCommand{\@ordinalstringFfrenchbelgian}[2]{%
\ifnum#1=1\relax
\def#2{premi\‘ere}%
\else
\let\@unitthstring=\@@unitthstringfrench
\let\@unitstring=\@@unitstringFfrench
\let\@teenthstring=\@@teenthstringfrench
```

```
\let\@teenstring=\@@teenstringfrench
\let\@tenthstring=\@@tenthstringfrench
\let\@tenstring=\@@tenstringfrench
\let\@seventieths=\@@seventiethsfrenchswiss
\let\@eightieths=\@@eightiethsfrench
\let\@ninetieths=\@@ninetiethsfrench
\let\@seventies=\@@seventiesfrench
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\let\@teenstring=\@@teenstringfrench
\def\@hundredth{centi\‘eme}\def\@hundred{cent}%
\def\@thousandth{mili\‘eme}\def\@thousand{mille}%
\def\@andname{et}%
\@@ordinalstringfrench{#1}{#2}%
\fi}
```

Set up default dialect:

```
\let\@ordinalstringFfrench=\@ordinalstringFfrenchfrance
```

Make neuter same as masculine:

```
\let\@ordinalstringNfrench\@ordinalstringMfrench
```

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

```
\DeclareRobustCommand{\@OrdinalstringMfrenchswiss}[2]{%
\ifnum#1=1\relax
\def#2{Premi\‘ere}%
\else
\let\@unitthstring=\@@Unitthstringfrench
\let\@unitstring=\@@Unitstringfrench
\let\@teenthstring=\@@Teenthstringfrench
\let\@teenstring=\@@Teenstringfrench
\let\@tenthstring=\@@Tenthstringfrench
\let\@tenstring=\@@Tenstringfrench
\let\@seventieths=\@@seventiethsfrenchswiss
\let\@eightieths=\@@eightiethsfrenchswiss
\let\@ninetieths=\@@ninetiethsfrenchswiss
\let\@seventies=\@@seventiesfrenchswiss
\let\@eighties=\@@eightiesfrenchswiss
\let\@nineties=\@@ninetiesfrenchswiss
\def\@hundredth{Centi\‘eme}\def\@hundred{Cent}%
\def\@thousandth{Mili\‘eme}\def\@thousand{Mille}%
\def\@andname{et}%
\@@ordinalstringfrench{#1}{#2}%
\fi}
```

French (masculine):

```
\DeclareRobustCommand{\@OrdinalstringMfrenchfrance}[2]{%
\ifnum#1=1\relax
\def#2{Premi\‘ere}%
\else
\let\@unitthstring=\@@Unitthstringfrench
\let\@unitstring=\@@Unitstringfrench
\let\@teenthstring=\@@Teenthstringfrench
\let\@teenstring=\@@Teenstringfrench
\let\@tenthstring=\@@Tenthstringfrench
\let\@tenstring=\@@Tenstringfrench
```

```
\let\@seventieths=\@@seventiethsfrench
\let\@eightieths=\@@eightiethsfrench
\let\@ninetieths=\@@ninetiethsfrench
\let\@seventies=\@@seventiesfrench
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\let\@teenstring=\@@Teenstringfrench
\def\@hundredth{Centi\‘eme}\def\@hundred{Cent}%
\def\@thousandth{Mili\‘eme}\def\@thousand{Mille}%
\def\@andname{et}%
\@@ordinalstringfrench{#1}{#2}%
\fi}
```

Belgian (masculine):

```
\DeclareRobustCommand{\@OrdinalstringMfrenchbelgian}[2]{%
\ifnum#1=1\relax
\def#2{Premi\‘ere}%
\else
\let\@unitthstring=\@@Unitthstringfrench
\let\@unitstring=\@@Unitstringfrench
\let\@teenthstring=\@@Teenthstringfrench
\let\@teenstring=\@@Teenstringfrench
\let\@tenthstring=\@@Tenthstringfrench
\let\@tenstring=\@@Tenstringfrench
\let\@seventieths=\@@seventiethsfrenchswiss
\let\@eightieths=\@@eightiethsfrench
\let\@ninetieths=\@@ninetiethsfrench
\let\@seventies=\@@seventiesfrench
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\let\@teenstring=\@@Teenstringfrench
\def\@hundredth{Centi\‘eme}\def\@hundred{Cent}%
\def\@thousandth{Mili\‘eme}\def\@thousand{Mille}%
\def\@andname{et}%
\@@ordinalstringfrench{#1}{#2}%
\fi}
```

Set up default dialect:

```
\let\@OrdinalstringMfrench=\@OrdinalstringMfrenchfrance
```

As above, but feminine form. Swiss:

```
\DeclareRobustCommand{\@OrdinalstringFfrenchswiss}[2]{%
\ifnum#1=1\relax
\def#2{Premi\‘ere}%
\else
\let\@unitthstring=\@@Unitthstringfrench
\let\@unitstring=\@@UnitstringFfrench
\let\@teenthstring=\@@Teenthstringfrench
\let\@teenstring=\@@Teenstringfrench
\let\@tenthstring=\@@Tenthstringfrench
\let\@tenstring=\@@Tenstringfrench
\let\@seventieths=\@@seventiethsfrenchswiss
\let\@eightieths=\@@eightiethsfrenchswiss
\let\@ninetieths=\@@ninetiethsfrenchswiss
\let\@seventies=\@@seventiesfrenchswiss
```

```
\let\@eighties=\@@eightiesfrenchswiss
\let\@nineties=\@@ninetiesfrenchswiss
\def\@hundredth{Centi\‘eme}\def\@hundred{Cent}%
\def\@thousandth{Mili\‘eme}\def\@thousand{Mille}%
\def\@andname{et}%
\@@ordinalstringfrench{#1}{#2}%
\fi}
```

French (feminine):

```
\DeclareRobustCommand{\@OrdinalstringFfrenchfrance}[2]{%
\ifnum#1=1\relax
\def#2{Premi\‘ere}%
\else
\let\@unitthstring=\@@Unitthstringfrench
\let\@unitstring=\@@UnitstringFfrench
\let\@teenthstring=\@@Teenthstringfrench
\let\@teenstring=\@@Teenstringfrench
\let\@tenthstring=\@@Tenthstringfrench
\let\@tenstring=\@@Tenstringfrench
\let\@seventieths=\@@seventiethsfrench
\let\@eightieths=\@@eightiethsfrench
\let\@ninetieths=\@@ninetiethsfrench
\let\@seventies=\@@seventiesfrench
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\let\@teenstring=\@@Teenstringfrench
\def\@hundredth{Centi\‘eme}\def\@hundred{Cent}%
\def\@thousandth{Mili\‘eme}\def\@thousand{Mille}%
\def\@andname{et}%
\@@ordinalstringfrench{#1}{#2}%
\fi}
```

Belgian (feminine):

```
\DeclareRobustCommand{\@OrdinalstringFfrenchbelgian}[2]{%
\ifnum#1=1\relax
\def#2{Premi\‘ere}%
\else
\let\@unitthstring=\@@Unitthstringfrench
\let\@unitstring=\@@UnitstringFfrench
\let\@teenthstring=\@@Teenthstringfrench
\let\@teenstring=\@@Teenstringfrench
\let\@tenthstring=\@@Tenthstringfrench
\let\@tenstring=\@@Tenstringfrench
\let\@seventieths=\@@seventiethsfrenchswiss
\let\@eightieths=\@@eightiethsfrench
\let\@ninetieths=\@@ninetiethsfrench
\let\@seventies=\@@seventiesfrench
\let\@eighties=\@@eightiesfrench
\let\@nineties=\@@ninetiesfrench
\let\@teenstring=\@@Teenstringfrench
\def\@hundredth{Centi\‘eme}\def\@hundred{Cent}%
\def\@thousandth{Mili\‘eme}\def\@thousand{Mille}%
\def\@andname{et}%
\@@ordinalstringfrench{#1}{#2}%
\fi}
```

Set up default dialect:

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

Make neuter same as masculine:

```
\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.

```
\newcommand*{\@@unitthstringfrench}[1]{%
\ifcase#1\relax
zero%
\or uni\‘eme%
\or deuxi\‘eme%
\or troisi\‘eme%
\or quatri\‘eme%
\or cinqui\‘eme%
\or sixi\‘eme%
\or septi\‘eme%
\or huiti\‘eme%
\or neuvi\‘eme%
\fi
}
```

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

```
\newcommand*{\@@tenthstringfrench}[1]{%
\ifcase#1\relax
\or dixi\‘eme%
\or vingti\‘eme%
\or trentri\‘eme%
\or quaranti\‘eme%
\or cinquanti\‘eme%
\or soixanti\‘eme%
\or septenti\‘eme%
\or huitanti\‘eme%
\or nonenti\‘eme%
\fi
}
```

Teens:

```
\newcommand*{\@@teenthstringfrench}[1]{%
\ifcase#1\relax
dixi\‘eme%
\or onzi\‘eme%
\or douzi\‘eme%
\or treizi\‘eme%
\or quatorzi\‘eme%
\or quinzi\‘eme%
\or seizi\‘eme%
\or dix-septi\‘eme%
\or dix-huiti\‘eme%
\or dix-neuvi\‘eme%
\fi
}
```

Seventies vary depending on dialect. Swiss:

```
\newcommand*{\@@seventiethsfrenchswiss}[1]{%
\ifcase#1\relax
\@tenthstring{7}%
\or
\@tenstring{7} \@andname\ \@unitthstring{1}%
\else
\@tenstring{7}-\@unitthstring{#1}%
\fi}
```

Eighties vary depending on dialect. Swiss:

```
\newcommand*{\@@eightiethsfrenchswiss}[1]{%
\ifcase#1\relax
\@tenthstring{8}%
\or
\@tenstring{8} \@andname\ \@unitthstring{1}%
\else
\@tenstring{8}-\@unitthstring{#1}%
\fi}
```

Nineties vary depending on dialect. Swiss:

```
\newcommand*{\@@ninetiethsfrenchswiss}[1]{%
\ifcase#1\relax
\@tenthstring{9}%
\or
\@tenstring{9} \@andname\ \@unitthstring{1}%
\else
\@tenstring{9}-\@unitthstring{#1}%
\fi}
```

French (as spoken in France) version:

```
\newcommand*{\@@seventiethsfrench}[1]{%
\ifnum#1=0\relax
\@tenstring{6}%
-%
\else
\@tenstring{6}%
\ \@andname\
\fi
\@teenthstring{#1}%
}
```

Eighties (as spoken in France):

```
\newcommand*{\@@eightiethsfrench}[1]{%
\ifnum#1>0\relax
\@unitstring{4}-\@tenstring{2}%
-\@unitthstring{#1}%
\else
\@unitstring{4}-\@tenthstring{2}%
\fi
}
```

Nineties (as spoken in France):

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

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

```
\newcommand*{\@@Unitthstringfrench}[1]{%
\ifcase#1\relax
Zero%
\or Uni\‘eme%
\or Deuxi\‘eme%
\or Troisi\‘eme%
\or Quatri\‘eme%
\or Cinqui\‘eme%
\or Sixi\‘eme%
\or Septi\‘eme%
\or Huiti\‘eme%
\or Neuvi\‘eme%
\fi
}
```

Tens (includes Belgian and Swiss variants):

```
\newcommand*{\@@Tenthstringfrench}[1]{%
\ifcase#1\relax
\or Dixi\‘eme%
\or Vingti\‘eme%
\or Trentri\‘eme%
\or Quaranti\‘eme%
\or Cinquanti\‘eme%
\or Soixanti\‘eme%
\or Septenti\‘eme%
\or Huitanti\‘eme%
\or Nonenti\‘eme%
\fi
}
```

Teens:

```
\newcommand*{\@@Teenthstringfrench}[1]{%
\ifcase#1\relax
Dixi\‘eme%
\or Onzi\‘eme%
\or Douzi\‘eme%
\or Treizi\‘eme%
\or Quatorzi\‘eme%
\or Quinzi\‘eme%
\or Seizi\‘eme%
\or Dix-Septi\‘eme%
\or Dix-Huiti\‘eme%
\or Dix-Neuvi\‘eme%
\fi
}
```

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

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

```
\PackageError{fmtcount}{Negative numbers not permitted}%
{This macro does not work for negative numbers, however
you can try typing "minus" first, and then pass the modulus of
this number}%
\fi
\fi
\def#2{}%
\@strctr=#1\relax \divide\@strctr by 1000\relax
\ifnum\@strctr>9\relax
% #1 is greater or equal to 10000
  \@tmpstrctr=\@strctr
  \divide\@strctr by 10\relax
  \ifnum\@strctr>1\relax
    \ifthenelse{\(\@strctr>6\)\and\(\@strctr<10\)}{%
      \@modulo{\@tmpstrctr}{10}%
      \ifnum\@strctr<8\relax
        \let\@@fc@numstr#2\relax
        \edef#2{\@@fc@numstr\@seventies{\@tmpstrctr}}%
      \else
        \ifnum\@strctr<9\relax
          \let\@@fc@numstr#2\relax
          \edef#2{\@@fc@numstr\@eighties{\@tmpstrctr}}%
        \else
          \ifnum\@strctr<10\relax
             \let\@@fc@numstr#2\relax
             \edef#2{\@@fc@numstr\@nineties{\@tmpstrctr}}%
          \fi
        \fi
      \fi
    }{%
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
      \@strctr=#1\relax
      \divide\@strctr by 1000\relax
      \@modulo{\@strctr}{10}%
      \ifnum\@strctr>0\relax
        \let\@@fc@numstr#2\relax
        \edef#2{\@@fc@numstr\ \@unitstring{\@strctr}}%
      \fi
    }%
  \else
    \@strctr=#1\relax
    \divide\@strctr by 1000
    \@modulo{\@strctr}{10}%
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
  \fi
  \let\@@fc@numstr#2\relax
  \edef#2{\@@fc@numstr\ \@thousand}%
\else
  \ifnum\@strctr>0\relax
    \ifnum\@strctr>1\relax
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\@unitstring{\@strctr}\ }%
```

```
    \fi
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@thousand}%
  \fi
\fi
\@strctr=#1\relax \@modulo{\@strctr}{1000}%
\divide\@strctr by 100
\ifnum\@strctr>0\relax
  \ifnum#1>1000\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\ }%
  \fi
  \@tmpstrctr=#1\relax
  \@modulo{\@tmpstrctr}{1000}\relax
  \ifnum\@tmpstrctr=100\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@tenstring{10}}%
  \else
    \ifnum\@strctr>1\relax
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\@unitstring{\@strctr}\ }%
    \fi
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@hundred}%
  \fi
\fi
\@strctr=#1\relax \@modulo{\@strctr}{100}%
%\@tmpstrctr=#1\relax
%\divide\@tmpstrctr by 100\relax
\ifnum#1>100\relax
  \ifnum\@strctr>0\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\ }%
  \else
    \ifnum\@tmpstrctr>0\relax
       \let\@@fc@numstr#2\relax
       \edef#2{\@@fc@numstr s}%
    \fi%
  \fi
\fi
\ifnum\@strctr>19\relax
  \@tmpstrctr=\@strctr
  \divide\@strctr by 10\relax
  \ifthenelse{\@strctr>6}{%
    \@modulo{\@tmpstrctr}{10}%
    \ifnum\@strctr<8\relax
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\@seventies{\@tmpstrctr}}%
    \else
      \ifnum\@strctr<9\relax
        \let\@@fc@numstr#2\relax
        \edef#2{\@@fc@numstr\@eighties{\@tmpstrctr}}%
      \else
        \let\@@fc@numstr#2\relax
```

```
        \edef#2{\@@fc@numstr\@nineties{\@tmpstrctr}}%
      \fi
    \fi
  }{%
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
    \@strctr=#1\relax \@modulo{\@strctr}{10}%
    \ifnum\@strctr>0\relax
      \let\@@fc@numstr#2\relax
      \ifnum\@strctr=1\relax
         \edef#2{\@@fc@numstr\ \@andname\ }%
      \else
         \edef#2{\@@fc@numstr-}%
      \fi
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
    \fi
  }%
\else
  \ifnum\@strctr<10\relax
    \ifnum\@strctr=0\relax
      \ifnum#1<100\relax
        \let\@@fc@numstr#2\relax
        \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
      \fi
    \else%(>0,<10)
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
    \fi
  \else%>10
    \@modulo{\@strctr}{10}%
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
  \fi
\fi
}
```

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

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

```
% #1 is greater or equal to 10000
  \@tmpstrctr=\@strctr
  \divide\@strctr by 10\relax
  \ifnum\@strctr>1\relax
    \ifthenelse{\@strctr>6}{%
      \@modulo{\@tmpstrctr}{10}%
      \ifnum\@strctr=7\relax
        \let\@@fc@ordstr#2\relax
        \edef#2{\@@fc@ordstr\@seventies{\@tmpstrctr}}%
      \else
        \ifnum\@strctr=8\relax
          \let\@@fc@ordstr#2\relax
          \edef#2{\@@fc@ordstr\@eighties{\@tmpstrctr}}%
        \else
          \let\@@fc@ordstr#2\relax
          \edef#2{\@@fc@ordstr\@nineties{\@tmpstrctr}}%
        \fi
      \fi
    }{%
      \let\@@fc@ordstr#2\relax
      \edef#2{\@@fc@ordstr\@tenstring{\@strctr}}%
      \@strctr=#1\relax
      \divide\@strctr by 1000\relax
      \@modulo{\@strctr}{10}%
      \ifnum\@strctr=1\relax
         \let\@@fc@ordstr#2\relax
         \edef#2{\@@fc@ordstr\ \@andname}%
      \fi
      \ifnum\@strctr>0\relax
         \let\@@fc@ordstr#2\relax
         \edef#2{\@@fc@ordstr\ \@unitstring{\@strctr}}%
      \fi
    }%
  \else
    \@strctr=#1\relax
    \divide\@strctr by 1000\relax
    \@modulo{\@strctr}{10}%
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\@teenstring{\@strctr}}%
  \fi
  \@strctr=#1\relax \@modulo{\@strctr}{1000}%
  \ifnum\@strctr=0\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\ \@thousandth}%
  \else
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\ \@thousand}%
  \fi
\else
  \ifnum\@strctr>0\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
    \@strctr=#1\relax \@modulo{\@strctr}{1000}%
    \ifnum\@strctr=0\relax
```

```
      \let\@@fc@ordstr#2\relax
      \edef#2{\@@fc@ordstr\ \@thousandth}%
    \else
      \let\@@fc@ordstr#2\relax
      \edef#2{\@@fc@ordstr\ \@thousand}%
    \fi
  \fi
\fi
\@strctr=#1\relax \@modulo{\@strctr}{1000}%
\divide\@strctr by 100\relax
\ifnum\@strctr>0\relax
  \ifnum#1>1000\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\ }%
  \fi
  \let\@@fc@ordstr#2\relax
  \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
  \@strctr=#1\relax \@modulo{\@strctr}{100}%
  \let\@@fc@ordstr#2\relax
  \ifnum\@strctr=0\relax
    \edef#2{\@@fc@ordstr\ \@hundredth}%
  \else
    \edef#2{\@@fc@ordstr\ \@hundred}%
  \fi
\fi
\@tmpstrctr=\@strctr
\@strctr=#1\relax \@modulo{\@strctr}{100}%
\ifnum#1>100\relax
  \ifnum\@strctr>0\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\ \@andname\ }%
  \fi
\fi
\ifnum\@strctr>19\relax
  \@tmpstrctr=\@strctr
  \divide\@strctr by 10\relax
  \@modulo{\@tmpstrctr}{10}%
  \ifthenelse{\@strctr>6}{%
    \ifnum\@strctr=7\relax
      \let\@@fc@ordstr#2\relax
      \edef#2{\@@fc@ordstr\@seventieths{\@tmpstrctr}}%
    \else
      \ifnum\@strctr=8\relax
        \let\@@fc@ordstr#2\relax
        \edef#2{\@@fc@ordstr\@eightieths{\@tmpstrctr}}%
      \else
        \let\@@fc@ordstr#2\relax
        \edef#2{\@@fc@ordstr\@ninetieths{\@tmpstrctr}}%
      \fi
    \fi
  }{%
    \ifnum\@tmpstrctr=0\relax
      \let\@@fc@ordstr#2\relax
      \edef#2{\@@fc@ordstr\@tenthstring{\@strctr}}%
```

```
    \else
      \let\@@fc@ordstr#2\relax
      \edef#2{\@@fc@ordstr\@tenstring{\@strctr}}%
    \fi
    \@strctr=#1\relax \@modulo{\@strctr}{10}%
    \ifnum\@strctr=1\relax
      \let\@@fc@ordstr#2\relax
      \edef#2{\@@fc@ordstr\ \@andname}%
    \fi
    \ifnum\@strctr>0\relax
      \let\@@fc@ordstr#2\relax
      \edef#2{\@@fc@ordstr\ \@unitthstring{\@strctr}}%
    \fi
  }%
\else
  \ifnum\@strctr<10\relax
    \ifnum\@strctr=0\relax
      \ifnum#1<100\relax
        \let\@@fc@ordstr#2\relax
        \edef#2{\@@fc@ordstr\@unitthstring{\@strctr}}%
      \fi
    \else
      \let\@@fc@ordstr#2\relax
      \edef#2{\@@fc@ordstr\@unitthstring{\@strctr}}%
    \fi
  \else
    \@modulo{\@strctr}{10}%
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\@teenthstring{\@strctr}}%
  \fi
\fi
}
```

11.4 fc-german.def

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

```
\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:

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

Feminine:

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

Neuter:

```
\newcommand{\@ordinalNgerman}[2]{%
\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):

```
\newcommand{\@@unitstringgerman}[1]{%
\ifcase#1%
null%
\or ein%
\or zwei%
\or drei%
\or vier%
\or f\"unf%
\or sechs%
\or sieben%
\or acht%
\or neun%
\fi
}
```

Tens (argument must go from 1 to 10):

```
\newcommand{\@@tenstringgerman}[1]{%
\ifcase#1%
\or zehn%
\or zwanzig%
\or drei{\ss}ig%
\or vierzig%
\or f\"unfzig%
\or sechzig%
\or siebzig%
\or achtzig%
\or neunzig%
\or einhundert%
\fi
}
```

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

```
\providecommand*{\einhundert}{einhundert}
\providecommand*{\eintausend}{eintausend}
```

Teens:

```
\newcommand{\@@teenstringgerman}[1]{%
\ifcase#1%
zehn%
\or elf%
\or zw\"olf%
\or dreizehn%
\or vierzehn%
\or f\"unfzehn%
\or sechzehn%
\or siebzehn%
\or achtzehn%
\or neunzehn%
\fi
}
```

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

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

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

Feminine and neuter forms:

```
\let\@numberstringFgerman=\@numberstringMgerman
\let\@numberstringNgerman=\@numberstringMgerman
```

As above, but initial letters in upper case:

```
\DeclareRobustCommand{\@NumberstringMgerman}[2]{%
\@numberstringMgerman{#1}{\@@num@str}%
\edef#2{\noexpand\MakeUppercase\@@num@str}}
```

Feminine and neuter form:

```
\let\@NumberstringFgerman=\@NumberstringMgerman
\let\@NumberstringNgerman=\@NumberstringMgerman
```

As above, but for ordinals.

```
\DeclareRobustCommand{\@ordinalstringMgerman}[2]{%
\let\@unitthstring=\@@unitthstringMgerman
\let\@teenthstring=\@@teenthstringMgerman
\let\@tenthstring=\@@tenthstringMgerman
\let\@unitstring=\@@unitstringgerman
\let\@teenstring=\@@teenstringgerman
\let\@tenstring=\@@tenstringgerman
\def\@thousandth{tausendster}%
\def\@hundredth{hundertster}%
\@@ordinalstringgerman{#1}{#2}}
```

Feminine form:

```
\DeclareRobustCommand{\@ordinalstringFgerman}[2]{%
\let\@unitthstring=\@@unitthstringFgerman
\let\@teenthstring=\@@teenthstringFgerman
\let\@tenthstring=\@@tenthstringFgerman
\let\@unitstring=\@@unitstringgerman
\let\@teenstring=\@@teenstringgerman
\let\@tenstring=\@@tenstringgerman
\def\@thousandth{tausendste}%
\def\@hundredth{hundertste}%
\@@ordinalstringgerman{#1}{#2}}
```

Neuter form:

```
\DeclareRobustCommand{\@ordinalstringNgerman}[2]{%
\let\@unitthstring=\@@unitthstringNgerman
\let\@teenthstring=\@@teenthstringNgerman
\let\@tenthstring=\@@tenthstringNgerman
\let\@unitstring=\@@unitstringgerman
\let\@teenstring=\@@teenstringgerman
\let\@tenstring=\@@tenstringgerman
\def\@thousandth{tausendstes}%
\def\@hundredth{hunderstes}%
\@@ordinalstringgerman{#1}{#2}}
```

As above, but with initial letters in upper case.

```
\DeclareRobustCommand{\@OrdinalstringMgerman}[2]{%
\@ordinalstringMgerman{#1}{\@@num@str}%
\edef#2{\protect\MakeUppercase\@@num@str}}
```

Feminine form:

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

Neuter form:

```
\DeclareRobustCommand{\@OrdinalstringNgerman}[2]{%
\@ordinalstringNgerman{#1}{\@@num@str}%
\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:

```
\newcommand{\@@unitthstringMgerman}[1]{%
\ifcase#1%
nullter%
\or erster%
\or zweiter%
\or dritter%
\or vierter%
\or f\"unter%
\or sechster%
\or siebter%
\or achter%
\or neunter%
\fi
}
```

Tens:

```
\newcommand{\@@tenthstringMgerman}[1]{%
\ifcase#1%
\or zehnter%
\or zwanzigster%
\or drei{\ss}igster%
\or vierzigster%
\or f\"unfzigster%
\or sechzigster%
\or siebzigster%
\or achtzigster%
\or neunzigster%
\fi
}
```

Teens:

```
\newcommand{\@@teenthstringMgerman}[1]{%
\ifcase#1%
zehnter%
\or elfter%
\or zw\"olfter%
\or dreizehnter%
\or vierzehnter%
\or f\"unfzehnter%
\or sechzehnter%
\or siebzehnter%
\or achtzehnter%
\or neunzehnter%
```

```
\fi
}
```

Units (feminine):

```
\newcommand{\@@unitthstringFgerman}[1]{%
\ifcase#1%
nullte%
\or erste%
\or zweite%
\or dritte%
\or vierte%
\or f\"unfte%
\or sechste%
\or siebte%
\or achte%
\or neunte%
\fi
}
```

Tens (feminine):

```
\newcommand{\@@tenthstringFgerman}[1]{%
\ifcase#1%
\or zehnte%
\or zwanzigste%
\or drei{\ss}igste%
\or vierzigste%
\or f\"unfzigste%
\or sechzigste%
\or siebzigste%
\or achtzigste%
\or neunzigste%
\fi
}
```

Teens (feminine)

```
\newcommand{\@@teenthstringFgerman}[1]{%
\ifcase#1%
zehnte%
\or elfte%
\or zw\"olfte%
\or dreizehnte%
\or vierzehnte%
\or f\"unfzehnte%
\or sechzehnte%
\or siebzehnte%
\or achtzehnte%
\or neunzehnte%
\fi
}
```

Units (neuter):

```
\newcommand{\@@unitthstringNgerman}[1]{%
\ifcase#1%
nulltes%
\or erstes%
\or zweites%
```

```
\or drittes%
\or viertes%
\or f\"unte%
\or sechstes%
\or siebtes%
\or achtes%
\or neuntes%
\fi
}
```

Tens (neuter):

```
\newcommand{\@@tenthstringNgerman}[1]{%
\ifcase#1%
\or zehntes%
\or zwanzigstes%
\or drei{\ss}igstes%
\or vierzigstes%
\or f\"unfzigstes%
\or sechzigstes%
\or siebzigstes%
\or achtzigstes%
\or neunzigstes%
\fi
}
```

Teens (neuter)

```
\newcommand{\@@teenthstringNgerman}[1]{%
\ifcase#1%
zehntes%
\or elftes%
\or zw\"olftes%
\or dreizehntes%
\or vierzehntes%
\or f\"unfzehntes%
\or sechzehntes%
\or siebzehntes%
\or achtzehntes%
\or neunzehntes%
\fi
}
```

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

```
\newcommand{\@@numberunderhundredgerman}[2]{%
\ifnum#1<10\relax
  \ifnum#1>0\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@unitstring{#1}}%
  \fi
\else
  \@tmpstrctr=#1\relax
  \@modulo{\@tmpstrctr}{10}%
  \ifnum#1<20\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@teenstring{\@tmpstrctr}}%
```

```
  \else
    \ifnum\@tmpstrctr=0\relax
    \else
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\@unitstring{\@tmpstrctr}und}%
    \fi
    \@tmpstrctr=#1\relax
    \divide\@tmpstrctr by 10\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@tenstring{\@tmpstrctr}}%
  \fi
\fi
}
```

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

```
\newcommand{\@@numberstringgerman}[2]{%
\ifnum#1>99999\relax
  \PackageError{fmtcount}{Out of range}%
  {This macro only works for values less than 100000}%
\else
  \ifnum#1<0\relax
    \PackageError{fmtcount}{Negative numbers not permitted}%
    {This macro does not work for negative numbers, however
    you can try typing "minus" first, and then pass the modulus of
    this number}%
  \fi
\fi
\def#2{}%
\@strctr=#1\relax \divide\@strctr by 1000\relax
\ifnum\@strctr>1\relax
% #1 is >= 2000, \@strctr now contains the number of thousands
\@@numberunderhundredgerman{\@strctr}{#2}%
  \let\@@fc@numstr#2\relax
  \edef#2{\@@fc@numstr tausend}%
\else
% #1 lies in range [1000,1999]
  \ifnum\@strctr=1\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\eintausend}%
  \fi
\fi
\@strctr=#1\relax
\@modulo{\@strctr}{1000}%
\divide\@strctr by 100\relax
\ifnum\@strctr>1\relax
% now dealing with number in range [200,999]
  \let\@@fc@numstr#2\relax
  \edef#2{\@@fc@numstr\@unitstring{\@strctr}hundert}%
\else
   \ifnum\@strctr=1\relax
% dealing with number in range [100,199]
     \ifnum#1>1000\relax
% if orginal number > 1000, use einhundert
```

```
        \let\@@fc@numstr#2\relax
        \edef#2{\@@fc@numstr einhundert}%
     \else
% otherwise use \einhundert
        \let\@@fc@numstr#2\relax
        \edef#2{\@@fc@numstr\einhundert}%
      \fi
   \fi
\fi
\@strctr=#1\relax
\@modulo{\@strctr}{100}%
\ifnum#1=0\relax
  \def#2{null}%
\else
  \ifnum\@strctr=1\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr eins}%
  \else
    \@@numberunderhundredgerman{\@strctr}{#2}%
  \fi
\fi
}
```

As above, but for ordinals

```
\newcommand{\@@numberunderhundredthgerman}[2]{%
\ifnum#1<10\relax
 \let\@@fc@numstr#2\relax
 \edef#2{\@@fc@numstr\@unitthstring{#1}}%
\else
  \@tmpstrctr=#1\relax
  \@modulo{\@tmpstrctr}{10}%
  \ifnum#1<20\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@teenthstring{\@tmpstrctr}}%
  \else
    \ifnum\@tmpstrctr=0\relax
    \else
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\@unitstring{\@tmpstrctr}und}%
    \fi
    \@tmpstrctr=#1\relax
    \divide\@tmpstrctr by 10\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@tenthstring{\@tmpstrctr}}%
  \fi
\fi
}
```

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

```
    {This macro does not work for negative numbers, however
    you can try typing "minus" first, and then pass the modulus of
    this number}%
  \fi
\fi
\def#2{}%
\@strctr=#1\relax \divide\@strctr by 1000\relax
\ifnum\@strctr>1\relax
% #1 is >= 2000, \@strctr now contains the number of thousands
\@@numberunderhundredgerman{\@strctr}{#2}%
  \let\@@fc@numstr#2\relax
  % is that it, or is there more?
  \@tmpstrctr=#1\relax \@modulo{\@tmpstrctr}{1000}%
  \ifnum\@tmpstrctr=0\relax
    \edef#2{\@@fc@numstr\@thousandth}%
  \else
    \edef#2{\@@fc@numstr tausend}%
  \fi
\else
% #1 lies in range [1000,1999]
  \ifnum\@strctr=1\relax
    \ifnum#1=1000\relax
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\@thousandth}%
    \else
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\eintausend}%
    \fi
  \fi
\fi
\@strctr=#1\relax
\@modulo{\@strctr}{1000}%
\divide\@strctr by 100\relax
\ifnum\@strctr>1\relax
% now dealing with number in range [200,999]
  \let\@@fc@numstr#2\relax
  % is that it, or is there more?
  \@tmpstrctr=#1\relax \@modulo{\@tmpstrctr}{100}%
  \ifnum\@tmpstrctr=0\relax
     \ifnum\@strctr=1\relax
       \edef#2{\@@fc@numstr\@hundredth}%
     \else
       \edef#2{\@@fc@numstr\@unitstring{\@strctr}\@hundredth}%
     \fi
  \else
     \edef#2{\@@fc@numstr\@unitstring{\@strctr}hundert}%
  \fi
\else
   \ifnum\@strctr=1\relax
% dealing with number in range [100,199]
% is that it, or is there more?
     \@tmpstrctr=#1\relax \@modulo{\@tmpstrctr}{100}%
     \ifnum\@tmpstrctr=0\relax
        \let\@@fc@numstr#2\relax
```

```
      \edef#2{\@@fc@numstr\@hundredth}%
    \else
    \ifnum#1>1000\relax
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr einhundert}%
    \else
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\einhundert}%
    \fi
    \fi
  \fi
\fi
\@strctr=#1\relax
\@modulo{\@strctr}{100}%
\ifthenelse{\@strctr=0 \and #1>0}{}{%
\@@numberunderhundredthgerman{\@strctr}{#2}%
}%
}
```

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

```
\let\@ordinalMngerman=\@ordinalMgerman
\let\@ordinalFngerman=\@ordinalFgerman
\let\@ordinalNngerman=\@ordinalNgerman
\let\@numberstringMngerman=\@numberstringMgerman
\let\@numberstringFngerman=\@numberstringFgerman
\let\@numberstringNngerman=\@numberstringNgerman
\let\@NumberstringMngerman=\@NumberstringMgerman
\let\@NumberstringFngerman=\@NumberstringFgerman
\let\@NumberstringNngerman=\@NumberstringNgerman
\let\@ordinalstringMngerman=\@ordinalstringMgerman
\let\@ordinalstringFngerman=\@ordinalstringFgerman
\let\@ordinalstringNngerman=\@ordinalstringNgerman
\let\@OrdinalstringMngerman=\@OrdinalstringMgerman
\let\@OrdinalstringFngerman=\@OrdinalstringFgerman
\let\@OrdinalstringNngerman=\@OrdinalstringNgerman
```

11.5 fc-portuges.def

Portuguse definitions

```
\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:

```
\newcommand*{\@ordinalMportuges}[2]{%
\ifnum#1=0\relax
  \edef#2{\number#1}%
\else
  \edef#2{\number#1\relax\noexpand\fmtord{o}}%
\fi}
```

Feminine:

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

```
\ifnum#1=0\relax
  \edef#2{\number#1}%
\else
  \edef#2{\number#1\relax\noexpand\fmtord{a}}%
\fi}
```

Make neuter same as masculine:

```
\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):

```
\newcommand*{\@@unitstringportuges}[1]{%
\ifcase#1\relax
zero%
\or um%
\or dois%
\or tr\^es%
\or quatro%
\or cinco%
\or seis%
\or sete%
\or oito%
\or nove%
\fi
}
%   \end{macrocode}
% As above, but for feminine:
%   \begin{macrocode}
\newcommand*{\@@unitstringFportuges}[1]{%
\ifcase#1\relax
zero%
\or uma%
\or duas%
\or tr\^es%
\or quatro%
\or cinco%
\or seis%
\or sete%
\or oito%
\or nove%
\fi
}
```

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

```
\newcommand*{\@@tenstringportuges}[1]{%
\ifcase#1\relax
\or dez%
\or vinte%
\or trinta%
\or quarenta%
\or cinq\"uenta%
\or sessenta%
\or setenta%
\or oitenta%
\or noventa%
```

```
\or cem%
\fi
}
```

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

```
\newcommand*{\@@teenstringportuges}[1]{%
\ifcase#1\relax
dez%
\or onze%
\or doze%
\or treze%
\or quatorze%
\or quinze%
\or dezesseis%
\or dezessete%
\or dezoito%
\or dezenove%
\fi
}
```

Hundreds:

```
\newcommand*{\@@hundredstringportuges}[1]{%
\ifcase#1\relax
\or cento%
\or duzentos%
\or trezentos%
\or quatrocentos%
\or quinhentos%
\or seiscentos%
\or setecentos%
\or oitocentos%
\or novecentos%
\fi}
```

Hundreds (feminine):

```
\newcommand*{\@@hundredstringFportuges}[1]{%
\ifcase#1\relax
\or cento%
\or duzentas%
\or trezentas%
\or quatrocentas%
\or quinhentas%
\or seiscentas%
\or setecentas%
\or oitocentas%
\or novecentas%
\fi}
```

Units (initial letter in upper case):

```
\newcommand*{\@@Unitstringportuges}[1]{%
\ifcase#1\relax
Zero%
\or Um%
\or Dois%
\or Tr\^es%
\or Quatro%
```

```
\or Cinco%
\or Seis%
\or Sete%
\or Oito%
\or Nove%
\fi
}
```

As above, but feminine:

```
\newcommand*{\@@UnitstringFportuges}[1]{%
\ifcase#1\relax
Zera%
\or Uma%
\or Duas%
\or Tr\^es%
\or Quatro%
\or Cinco%
\or Seis%
\or Sete%
\or Oito%
\or Nove%
\fi
}
```

Tens (with initial letter in upper case):

```
\newcommand*{\@@Tenstringportuges}[1]{%
\ifcase#1\relax
\or Dez%
\or Vinte%
\or Trinta%
\or Quarenta%
\or Cinq\"uenta%
\or Sessenta%
\or Setenta%
\or Oitenta%
\or Noventa%
\or Cem%
\fi
}
```

Teens (with initial letter in upper case):

```
\newcommand*{\@@Teenstringportuges}[1]{%
\ifcase#1\relax
Dez%
\or Onze%
\or Doze%
\or Treze%
\or Quatorze%
\or Quinze%
\or Dezesseis%
\or Dezessete%
\or Dezoito%
\or Dezenove%
\fi
}
```

Hundreds (with initial letter in upper case):

```
\newcommand*{\@@Hundredstringportuges}[1]{%
\ifcase#1\relax
\or Cento%
\or Duzentos%
\or Trezentos%
\or Quatrocentos%
\or Quinhentos%
\or Seiscentos%
\or Setecentos%
\or Oitocentos%
\or Novecentos%
\fi}
```

As above, but feminine:

```
\newcommand*{\@@HundredstringFportuges}[1]{%
\ifcase#1\relax
\or Cento%
\or Duzentas%
\or Trezentas%
\or Quatrocentas%
\or Quinhentas%
\or Seiscentas%
\or Setecentas%
\or Oitocentas%
\or Novecentas%
\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.)

```
\DeclareRobustCommand{\@numberstringMportuges}[2]{%
\let\@unitstring=\@@unitstringportuges
\let\@teenstring=\@@teenstringportuges
\let\@tenstring=\@@tenstringportuges
\let\@hundredstring=\@@hundredstringportuges
\def\@hundred{cem}\def\@thousand{mil}%
\def\@andname{e}%
\@@numberstringportuges{#1}{#2}}
```

As above, but feminine form:

```
\DeclareRobustCommand{\@numberstringFportuges}[2]{%
\let\@unitstring=\@@unitstringFportuges
\let\@teenstring=\@@teenstringportuges
\let\@tenstring=\@@tenstringportuges
\let\@hundredstring=\@@hundredstringFportuges
\def\@hundred{cem}\def\@thousand{mil}%
\def\@andname{e}%
\@@numberstringportuges{#1}{#2}}
```

Make neuter same as masculine:

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

As above, but initial letters in upper case:

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

```
\let\@unitstring=\@@Unitstringportuges
\let\@teenstring=\@@Teenstringportuges
\let\@tenstring=\@@Tenstringportuges
\let\@hundredstring=\@@Hundredstringportuges
\def\@hundred{Cem}\def\@thousand{Mil}%
\def\@andname{e}%
\@@numberstringportuges{#1}{#2}}
```

As above, but feminine form:

```
\DeclareRobustCommand{\@NumberstringFportuges}[2]{%
\let\@unitstring=\@@UnitstringFportuges
\let\@teenstring=\@@Teenstringportuges
\let\@tenstring=\@@Tenstringportuges
\let\@hundredstring=\@@HundredstringFportuges
\def\@hundred{Cem}\def\@thousand{Mil}%
\def\@andname{e}%
\@@numberstringportuges{#1}{#2}}
```

Make neuter same as masculine:

```
\let\@NumberstringNportuges\@NumberstringMportuges
```

As above, but for ordinals.

```
\DeclareRobustCommand{\@ordinalstringMportuges}[2]{%
\let\@unitthstring=\@@unitthstringportuges
\let\@unitstring=\@@unitstringportuges
\let\@teenthstring=\@@teenthstringportuges
\let\@tenthstring=\@@tenthstringportuges
\let\@hundredthstring=\@@hundredthstringportuges
\def\@thousandth{mil\'esimo}%
\@@ordinalstringportuges{#1}{#2}}
```

Feminine form:

```
\DeclareRobustCommand{\@ordinalstringFportuges}[2]{%
\let\@unitthstring=\@@unitthstringFportuges
\let\@unitstring=\@@unitstringFportuges
\let\@teenthstring=\@@teenthstringportuges
\let\@tenthstring=\@@tenthstringFportuges
\let\@hundredthstring=\@@hundredthstringFportuges
\def\@thousandth{mil\'esima}%
\@@ordinalstringportuges{#1}{#2}}
```

Make neuter same as masculine:

```
\let\@ordinalstringNportuges\@ordinalstringMportuges
```

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

```
\DeclareRobustCommand{\@OrdinalstringMportuges}[2]{%
\let\@unitthstring=\@@Unitthstringportuges
\let\@unitstring=\@@Unitstringportuges
\let\@teenthstring=\@@teenthstringportuges
\let\@tenthstring=\@@Tenthstringportuges
\let\@hundredthstring=\@@Hundredthstringportuges
\def\@thousandth{Mil\'esimo}%
\@@ordinalstringportuges{#1}{#2}}
```

Feminine form:

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

```
\let\@unitstring=\@@UnitstringFportuges
\let\@teenthstring=\@@teenthstringportuges
\let\@tenthstring=\@@TenthstringFportuges
\let\@hundredthstring=\@@HundredthstringFportuges
\def\@thousandth{Mil\'esima}%
\@@ordinalstringportuges{#1}{#2}}
```

Make neuter same as masculine:

```
\let\@OrdinalstringNportuges\@OrdinalstringMportuges
```

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

```
\newcommand*{\@@unitthstringportuges}[1]{%
\ifcase#1\relax
zero%
\or primeiro%
\or segundo%
\or terceiro%
\or quarto%
\or quinto%
\or sexto%
\or s\'etimo%
\or oitavo%
\or nono%
\fi
}
```

Tens:

```
\newcommand*{\@@tenthstringportuges}[1]{%
\ifcase#1\relax
\or d\'ecimo%
\or vig\'esimo%
\or trig\'esimo%
\or quadrag\'esimo%
\or q\"uinquag\'esimo%
\or sexag\'esimo%
\or setuag\'esimo%
\or octog\'esimo%
\or nonag\'esimo%
\fi
}
```

Teens:

```
\newcommand*{\@@teenthstringportuges}[1]{%
\@tenthstring{1}%
\ifnum#1>0\relax
-\@unitthstring{#1}%
\fi}
```

Hundreds:

```
\newcommand*{\@@hundredthstringportuges}[1]{%
\ifcase#1\relax
\or cent\'esimo%
\or ducent\'esimo%
\or trecent\'esimo%
\or quadringent\'esimo%
\or q\"uingent\'esimo%
```

```
\or seiscent\'esimo%
\or setingent\'esimo%
\or octingent\'esimo%
\or nongent\'esimo%
\fi}
```

Units (feminine):

```
\newcommand*{\@@unitthstringFportuges}[1]{%
\ifcase#1\relax
zero%
\or primeira%
\or segunda%
\or terceira%
\or quarta%
\or quinta%
\or sexta%
\or s\'etima%
\or oitava%
\or nona%
\fi
}
```

Tens (feminine):

```
\newcommand*{\@@tenthstringFportuges}[1]{%
\ifcase#1\relax
\or d\'ecima%
\or vig\'esima%
\or trig\'esima%
\or quadrag\'esima%
\or q\"uinquag\'esima%
\or sexag\'esima%
\or setuag\'esima%
\or octog\'esima%
\or nonag\'esima%
\fi
}
```

Hundreds (feminine):

```
\newcommand*{\@@hundredthstringFportuges}[1]{%
\ifcase#1\relax
\or cent\'esima%
\or ducent\'esima%
\or trecent\'esima%
\or quadringent\'esima%
\or q\"uingent\'esima%
\or seiscent\'esima%
\or setingent\'esima%
\or octingent\'esima%
\or nongent\'esima%
\fi}
```

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

```
\newcommand*{\@@Unitthstringportuges}[1]{%
\ifcase#1\relax
Zero%
\or Primeiro%
```

```
\or Segundo%
\or Terceiro%
\or Quarto%
\or Quinto%
\or Sexto%
\or S\'etimo%
\or Oitavo%
\or Nono%
\fi
}
```

Tens:

```
\newcommand*{\@@Tenthstringportuges}[1]{%
\ifcase#1\relax
\or D\'ecimo%
\or Vig\'esimo%
\or Trig\'esimo%
\or Quadrag\'esimo%
\or Q\"uinquag\'esimo%
\or Sexag\'esimo%
\or Setuag\'esimo%
\or Octog\'esimo%
\or Nonag\'esimo%
\fi
}
```

Hundreds:

```
\newcommand*{\@@Hundredthstringportuges}[1]{%
\ifcase#1\relax
\or Cent\'esimo%
\or Ducent\'esimo%
\or Trecent\'esimo%
\or Quadringent\'esimo%
\or Q\"uingent\'esimo%
\or Seiscent\'esimo%
\or Setingent\'esimo%
\or Octingent\'esimo%
\or Nongent\'esimo%
\fi}
```

As above, but feminine. Units:

```
\newcommand*{\@@UnitthstringFportuges}[1]{%
\ifcase#1\relax
Zera%
\or Primeira%
\or Segunda%
\or Terceira%
\or Quarta%
\or Quinta%
\or Sexta%
\or S\'etima%
\or Oitava%
\or Nona%
\fi
}
```

Tens (feminine);

```
\newcommand*{\@@TenthstringFportuges}[1]{%
\ifcase#1\relax
\or D\'ecima%
\or Vig\'esima%
\or Trig\'esima%
\or Quadrag\'esima%
\or Q\"uinquag\'esima%
\or Sexag\'esima%
\or Setuag\'esima%
\or Octog\'esima%
\or Nonag\'esima%
\fi
}
```

Hundreds (feminine):

```
\newcommand*{\@@HundredthstringFportuges}[1]{%
\ifcase#1\relax
\or Cent\'esima%
\or Ducent\'esima%
\or Trecent\'esima%
\or Quadringent\'esima%
\or Q\"uingent\'esima%
\or Seiscent\'esima%
\or Setingent\'esima%
\or Octingent\'esima%
\or Nongent\'esima%
\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.)

```
\newcommand*{\@@numberstringportuges}[2]{%
\ifnum#1>99999
\PackageError{fmtcount}{Out of range}%
{This macro only works for values less than 100000}%
\else
\ifnum#1<0
\PackageError{fmtcount}{Negative numbers not permitted}%
{This macro does not work for negative numbers, however
you can try typing "minus" first, and then pass the modulus of
this number}%
\fi
\fi
\def#2{}%
\@strctr=#1\relax \divide\@strctr by 1000\relax
\ifnum\@strctr>9
% #1 is greater or equal to 10000
  \divide\@strctr by 10
  \ifnum\@strctr>1\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
    \@strctr=#1 \divide\@strctr by 1000\relax
```

```
    \@modulo{\@strctr}{10}%
    \ifnum\@strctr>0
      \ifnum\@strctr=1\relax
        \let\@@fc@numstr#2\relax
        \edef#2{\@@fc@numstr\ \@andname}%
      \fi
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\ \@unitstring{\@strctr}}%
    \fi
  \else
    \@strctr=#1\relax
    \divide\@strctr by 1000\relax
    \@modulo{\@strctr}{10}%
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
  \fi
  \let\@@fc@numstr#2\relax
  \edef#2{\@@fc@numstr\ \@thousand}%
\else
  \ifnum\@strctr>0\relax
    \ifnum\@strctr>1\relax
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\@unitstring{\@strctr}\ }%
    \fi
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@thousand}%
  \fi
\fi
\@strctr=#1\relax \@modulo{\@strctr}{1000}%
\divide\@strctr by 100\relax
\ifnum\@strctr>0\relax
  \ifnum#1>1000 \relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\ }%
  \fi
  \@tmpstrctr=#1\relax
  \@modulo{\@tmpstrctr}{1000}%
  \let\@@fc@numstr#2\relax
  \ifnum\@tmpstrctr=100\relax
    \edef#2{\@@fc@numstr\@tenstring{10}}%
  \else
    \edef#2{\@@fc@numstr\@hundredstring{\@strctr}}%
  \fi%
\fi
\@strctr=#1\relax \@modulo{\@strctr}{100}%
\ifnum#1>100\relax
  \ifnum\@strctr>0\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\ \@andname\ }%
  \fi
\fi
\ifnum\@strctr>19\relax
  \divide\@strctr by 10\relax
  \let\@@fc@numstr#2\relax
```

```
  \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
  \@strctr=#1\relax \@modulo{\@strctr}{10}%
  \ifnum\@strctr>0
    \ifnum\@strctr=1\relax
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\ \@andname}%
    \else
      \ifnum#1>100\relax
        \let\@@fc@numstr#2\relax
        \edef#2{\@@fc@numstr\ \@andname}%
      \fi
    \fi
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\ \@unitstring{\@strctr}}%
  \fi
\else
  \ifnum\@strctr<10\relax
    \ifnum\@strctr=0\relax
      \ifnum#1<100\relax
        \let\@@fc@numstr#2\relax
        \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
      \fi
    \else%(>0,<10)
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
    \fi
  \else%>10
    \@modulo{\@strctr}{10}%
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
  \fi
\fi
}
```

As above, but for ordinals.

```
\newcommand*{\@@ordinalstringportuges}[2]{%
\@strctr=#1\relax
\ifnum#1>99999
\PackageError{fmtcount}{Out of range}%
{This macro only works for values less than 100000}%
\else
\ifnum#1<0
\PackageError{fmtcount}{Negative numbers not permitted}%
{This macro does not work for negative numbers, however
you can try typing "minus" first, and then pass the modulus of
this number}%
\else
\def#2{}%
\ifnum\@strctr>999\relax
  \divide\@strctr by 1000\relax
  \ifnum\@strctr>1\relax
    \ifnum\@strctr>9\relax
      \@tmpstrctr=\@strctr
      \ifnum\@strctr<20
        \@modulo{\@tmpstrctr}{10}%
```

```
        \let\@@fc@ordstr#2\relax
        \edef#2{\@@fc@ordstr\@teenthstring{\@tmpstrctr}}%
      \else
        \divide\@tmpstrctr by 10\relax
        \let\@@fc@ordstr#2\relax
        \edef#2{\@@fc@ordstr\@tenthstring{\@tmpstrctr}}%
        \@tmpstrctr=\@strctr
        \@modulo{\@tmpstrctr}{10}%
        \ifnum\@tmpstrctr>0\relax
          \let\@@fc@ordstr#2\relax
          \edef#2{\@@fc@ordstr\@unitthstring{\@tmpstrctr}}%
        \fi
      \fi
    \else
      \let\@@fc@ordstr#2\relax
      \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
    \fi
  \fi
  \let\@@fc@ordstr#2\relax
  \edef#2{\@@fc@ordstr\@thousandth}%
\fi
\@strctr=#1\relax
\@modulo{\@strctr}{1000}%
\ifnum\@strctr>99\relax
  \@tmpstrctr=\@strctr
  \divide\@tmpstrctr by 100\relax
  \ifnum#1>1000\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr-}%
  \fi
  \let\@@fc@ordstr#2\relax
  \edef#2{\@@fc@ordstr\@hundredthstring{\@tmpstrctr}}%
\fi
\@modulo{\@strctr}{100}%
\ifnum#1>99\relax
  \ifnum\@strctr>0\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr-}%
  \fi
\fi
\ifnum\@strctr>9\relax
  \@tmpstrctr=\@strctr
  \divide\@tmpstrctr by 10\relax
  \let\@@fc@ordstr#2\relax
  \edef#2{\@@fc@ordstr\@tenthstring{\@tmpstrctr}}%
  \@tmpstrctr=\@strctr
  \@modulo{\@tmpstrctr}{10}%
  \ifnum\@tmpstrctr>0\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr-\@unitthstring{\@tmpstrctr}}%
  \fi
\else
  \ifnum\@strctr=0\relax
    \ifnum#1=0\relax
```

```
      \let\@@fc@ordstr#2\relax
      \edef#2{\@@fc@ordstr\@unitstring{0}}%
    \fi
  \else
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\@unitthstring{\@strctr}}%
  \fi
\fi
\fi
\fi
}
```

11.6 fc-spanish.def

Spanish definitions

```
\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:

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

Feminine:

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

Make neuter same as masculine:

```
\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):

```
\newcommand{\@@unitstringspanish}[1]{%
\ifcase#1\relax
cero%
\or uno%
\or dos%
\or tres%
\or cuatro%
\or cinco%
\or seis%
\or siete%
\or ocho%
\or nueve%
\fi
}
```

Feminine:

```
\newcommand{\@@unitstringFspanish}[1]{%
\ifcase#1\relax
cera%
\or una%
\or dos%
\or tres%
```

```
\or cuatro%
\or cinco%
\or seis%
\or siete%
\or ocho%
\or nueve%
\fi
}
```

Tens (argument must go from 1 to 10):

```
\newcommand{\@@tenstringspanish}[1]{%
\ifcase#1\relax
\or diez%
\or viente%
\or treinta%
\or cuarenta%
\or cincuenta%
\or sesenta%
\or setenta%
\or ochenta%
\or noventa%
\or cien%
\fi
}
```

Teens:

```
\newcommand{\@@teenstringspanish}[1]{%
\ifcase#1\relax
diez%
\or once%
\or doce%
\or trece%
\or catorce%
\or quince%
\or diecis\'eis%
\or diecisiete%
\or dieciocho%
\or diecinueve%
\fi
}
```

Twenties:

```
\newcommand{\@@twentystringspanish}[1]{%
\ifcase#1\relax
veinte%
\or veintiuno%
\or veintid\'os%
\or veintitr\'es%
\or veinticuatro%
\or veinticinco%
\or veintis\'eis%
\or veintisiete%
\or veintiocho%
\or veintinueve%
\fi}
```

Feminine form:

```
\newcommand{\@@twentystringFspanish}[1]{%
\ifcase#1\relax
veinte%
\or veintiuna%
\or veintid\'os%
\or veintitr\'es%
\or veinticuatro%
\or veinticinco%
\or veintis\'eis%
\or veintisiete%
\or veintiocho%
\or veintinueve%
\fi}
```

Hundreds:

```
\newcommand{\@@hundredstringspanish}[1]{%
\ifcase#1\relax
\or ciento%
\or doscientos%
\or trescientos%
\or cuatrocientos%
\or quinientos%
\or seiscientos%
\or setecientos%
\or ochocientos%
\or novecientos%
\fi}
```

Feminine form:

```
\newcommand{\@@hundredstringFspanish}[1]{%
\ifcase#1\relax
\or cienta%
\or doscientas%
\or trescientas%
\or cuatrocientas%
\or quinientas%
\or seiscientas%
\or setecientas%
\or ochocientas%
\or novecientas%
\fi}
```

As above, but with initial letter uppercase:

```
\newcommand{\@@Unitstringspanish}[1]{%
\ifcase#1\relax
Cero%
\or Uno%
\or Dos%
\or Tres%
\or Cuatro%
\or Cinco%
\or Seis%
\or Siete%
\or Ocho%
```

```
\or Nueve%
\fi
}
```

Feminine form:

```
\newcommand{\@@UnitstringFspanish}[1]{%
\ifcase#1\relax
Cera%
\or Una%
\or Dos%
\or Tres%
\or Cuatro%
\or Cinco%
\or Seis%
\or Siete%
\or Ocho%
\or Nueve%
\fi
}
```

Tens:

```
\newcommand{\@@Tenstringspanish}[1]{%
\ifcase#1\relax
\or Diez%
\or Viente%
\or Treinta%
\or Cuarenta%
\or Cincuenta%
\or Sesenta%
\or Setenta%
\or Ochenta%
\or Noventa%
\or Cien%
\fi
}
```

Teens:

```
\newcommand{\@@Teenstringspanish}[1]{%
\ifcase#1\relax
Diez%
\or Once%
\or Doce%
\or Trece%
\or Catorce%
\or Quince%
\or Diecis\'eis%
\or Diecisiete%
\or Dieciocho%
\or Diecinueve%
\fi
}
```

Twenties:

```
\newcommand{\@@Twentystringspanish}[1]{%
\ifcase#1\relax
Veinte%
```

```
\or Veintiuno%
\or Veintid\'os%
\or Veintitr\'es%
\or Veinticuatro%
\or Veinticinco%
\or Veintis\'eis%
\or Veintisiete%
\or Veintiocho%
\or Veintinueve%
\fi}
```

Feminine form:

```
\newcommand{\@@TwentystringFspanish}[1]{%
\ifcase#1\relax
Veinte%
\or Veintiuna%
\or Veintid\'os%
\or Veintitr\'es%
\or Veinticuatro%
\or Veinticinco%
\or Veintis\'eis%
\or Veintisiete%
\or Veintiocho%
\or Veintinueve%
\fi}
```

Hundreds:

```
\newcommand{\@@Hundredstringspanish}[1]{%
\ifcase#1\relax
\or Ciento%
\or Doscientos%
\or Trescientos%
\or Cuatrocientos%
\or Quinientos%
\or Seiscientos%
\or Setecientos%
\or Ochocientos%
\or Novecientos%
\fi}
```

Feminine form:

```
\newcommand{\@@HundredstringFspanish}[1]{%
\ifcase#1\relax
\or Cienta%
\or Doscientas%
\or Trescientas%
\or Cuatrocientas%
\or Quinientas%
\or Seiscientas%
\or Setecientas%
\or Ochocientas%
\or Novecientas%
\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.)

```
\DeclareRobustCommand{\@numberstringMspanish}[2]{%
\let\@unitstring=\@@unitstringspanish
\let\@teenstring=\@@teenstringspanish
\let\@tenstring=\@@tenstringspanish
\let\@twentystring=\@@twentystringspanish
\let\@hundredstring=\@@hundredstringspanish
\def\@hundred{cien}\def\@thousand{mil}%
\def\@andname{y}%
\@@numberstringspanish{#1}{#2}}
```

Feminine form:

```
\DeclareRobustCommand{\@numberstringFspanish}[2]{%
\let\@unitstring=\@@unitstringFspanish
\let\@teenstring=\@@teenstringspanish
\let\@tenstring=\@@tenstringspanish
\let\@twentystring=\@@twentystringFspanish
\let\@hundredstring=\@@hundredstringFspanish
\def\@hundred{cien}\def\@thousand{mil}%
\def\@andname{y}%
\@@numberstringspanish{#1}{#2}}
```

Make neuter same as masculine:

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

As above, but initial letters in upper case:

```
\DeclareRobustCommand{\@NumberstringMspanish}[2]{%
\let\@unitstring=\@@Unitstringspanish
\let\@teenstring=\@@Teenstringspanish
\let\@tenstring=\@@Tenstringspanish
\let\@twentystring=\@@Twentystringspanish
\let\@hundredstring=\@@Hundredstringspanish
\def\@andname{y}%
\def\@hundred{Cien}\def\@thousand{Mil}%
\@@numberstringspanish{#1}{#2}}
```

Feminine form:

```
\DeclareRobustCommand{\@NumberstringFspanish}[2]{%
\let\@unitstring=\@@UnitstringFspanish
\let\@teenstring=\@@Teenstringspanish
\let\@tenstring=\@@Tenstringspanish
\let\@twentystring=\@@TwentystringFspanish
\let\@hundredstring=\@@HundredstringFspanish
\def\@andname{y}%
\def\@hundred{Cien}\def\@thousand{Mil}%
\@@numberstringspanish{#1}{#2}}
```

Make neuter same as masculine:

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

As above, but for ordinals.

```
\DeclareRobustCommand{\@ordinalstringMspanish}[2]{%
\let\@unitthstring=\@@unitthstringspanish
\let\@unitstring=\@@unitstringspanish
\let\@teenthstring=\@@teenthstringspanish
```

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

Feminine form:

```
\DeclareRobustCommand{\@ordinalstringFspanish}[2]{%
\let\@unitthstring=\@@unitthstringFspanish
\let\@unitstring=\@@unitstringFspanish
\let\@teenthstring=\@@teenthstringFspanish
\let\@tenthstring=\@@tenthstringFspanish
\let\@hundredthstring=\@@hundredthstringFspanish
\def\@thousandth{mil\'esima}%
\@@ordinalstringspanish{#1}{#2}}
```

Make neuter same as masculine:

```
\let\@ordinalstringNspanish\@ordinalstringMspanish
```

As above, but with initial letters in upper case.

```
\DeclareRobustCommand{\@OrdinalstringMspanish}[2]{%
\let\@unitthstring=\@@Unitthstringspanish
\let\@unitstring=\@@Unitstringspanish
\let\@teenthstring=\@@Teenthstringspanish
\let\@tenthstring=\@@Tenthstringspanish
\let\@hundredthstring=\@@Hundredthstringspanish
\def\@thousandth{Mil\'esimo}%
\@@ordinalstringspanish{#1}{#2}}
```

Feminine form:

```
\DeclareRobustCommand{\@OrdinalstringFspanish}[2]{%
\let\@unitthstring=\@@UnitthstringFspanish
\let\@unitstring=\@@UnitstringFspanish
\let\@teenthstring=\@@TeenthstringFspanish
\let\@tenthstring=\@@TenthstringFspanish
\let\@hundredthstring=\@@HundredthstringFspanish
\def\@thousandth{Mil\'esima}%
\@@ordinalstringspanish{#1}{#2}}
```

Make neuter same as masculine:

```
\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:

```
\newcommand{\@@unitthstringspanish}[1]{%
\ifcase#1\relax
cero%
\or primero%
\or segundo%
\or tercero%
\or cuarto%
\or quinto%
\or sexto%
\or s\'eptimo%
\or octavo%
\or noveno%
\fi
}
```

Tens:

```
\newcommand{\@@tenthstringspanish}[1]{%
\ifcase#1\relax
\or d\'ecimo%
\or vig\'esimo%
\or trig\'esimo%
\or cuadrag\'esimo%
\or quincuag\'esimo%
\or sexag\'esimo%
\or septuag\'esimo%
\or octog\'esimo%
\or nonag\'esimo%
\fi
}
```

Teens:

```
\newcommand{\@@teenthstringspanish}[1]{%
\ifcase#1\relax
d\'ecimo%
\or und\'ecimo%
\or duod\'ecimo%
\or decimotercero%
\or decimocuarto%
\or decimoquinto%
\or decimosexto%
\or decimos\'eptimo%
\or decimoctavo%
\or decimonoveno%
\fi
}
```

Hundreds:

```
\newcommand{\@@hundredthstringspanish}[1]{%
\ifcase#1\relax
\or cent\'esimo%
\or ducent\'esimo%
\or tricent\'esimo%
\or cuadringent\'esimo%
\or quingent\'esimo%
\or sexcent\'esimo%
\or septing\'esimo%
\or octingent\'esimo%
\or noningent\'esimo%
\fi}
```

Units (feminine):

```
\newcommand{\@@unitthstringFspanish}[1]{%
\ifcase#1\relax
cera%
\or primera%
\or segunda%
\or tercera%
\or cuarta%
\or quinta%
\or sexta%
```

```
\or s\'eptima%
\or octava%
\or novena%
\fi
}
```

Tens (feminine):

```
\newcommand{\@@tenthstringFspanish}[1]{%
\ifcase#1\relax
\or d\'ecima%
\or vig\'esima%
\or trig\'esima%
\or cuadrag\'esima%
\or quincuag\'esima%
\or sexag\'esima%
\or septuag\'esima%
\or octog\'esima%
\or nonag\'esima%
\fi
}
```

Teens (feminine)

```
\newcommand{\@@teenthstringFspanish}[1]{%
\ifcase#1\relax
d\'ecima%
\or und\'ecima%
\or duod\'ecima%
\or decimotercera%
\or decimocuarta%
\or decimoquinta%
\or decimosexta%
\or decimos\'eptima%
\or decimoctava%
\or decimonovena%
\fi
}
```

Hundreds (feminine)

```
\newcommand{\@@hundredthstringFspanish}[1]{%
\ifcase#1\relax
\or cent\'esima%
\or ducent\'esima%
\or tricent\'esima%
\or cuadringent\'esima%
\or quingent\'esima%
\or sexcent\'esima%
\or septing\'esima%
\or octingent\'esima%
\or noningent\'esima%
\fi}
```

As above, but with initial letters in upper case

```
\newcommand{\@@Unitthstringspanish}[1]{%
\ifcase#1\relax
Cero%
\or Primero%
```

```
\or Segundo%
\or Tercero%
\or Cuarto%
\or Quinto%
\or Sexto%
\or S\'eptimo%
\or Octavo%
\or Noveno%
\fi
}
```

Tens:

```
\newcommand{\@@Tenthstringspanish}[1]{%
\ifcase#1\relax
\or D\'ecimo%
\or Vig\'esimo%
\or Trig\'esimo%
\or Cuadrag\'esimo%
\or Quincuag\'esimo%
\or Sexag\'esimo%
\or Septuag\'esimo%
\or Octog\'esimo%
\or Nonag\'esimo%
\fi
}
```

Teens:

```
\newcommand{\@@Teenthstringspanish}[1]{%
\ifcase#1\relax
D\'ecimo%
\or Und\'ecimo%
\or Duod\'ecimo%
\or Decimotercero%
\or Decimocuarto%
\or Decimoquinto%
\or Decimosexto%
\or Decimos\'eptimo%
\or Decimoctavo%
\or Decimonoveno%
\fi
}
```

Hundreds

```
\newcommand{\@@Hundredthstringspanish}[1]{%
\ifcase#1\relax
\or Cent\'esimo%
\or Ducent\'esimo%
\or Tricent\'esimo%
\or Cuadringent\'esimo%
\or Quingent\'esimo%
\or Sexcent\'esimo%
\or Septing\'esimo%
\or Octingent\'esimo%
\or Noningent\'esimo%
\fi}
```

As above, but feminine.

```
\newcommand{\@@UnitthstringFspanish}[1]{%
\ifcase#1\relax
Cera%
\or Primera%
\or Segunda%
\or Tercera%
\or Cuarta%
\or Quinta%
\or Sexta%
\or S\'eptima%
\or Octava%
\or Novena%
\fi
}
```

Tens (feminine)

```
\newcommand{\@@TenthstringFspanish}[1]{%
\ifcase#1\relax
\or D\'ecima%
\or Vig\'esima%
\or Trig\'esima%
\or Cuadrag\'esima%
\or Quincuag\'esima%
\or Sexag\'esima%
\or Septuag\'esima%
\or Octog\'esima%
\or Nonag\'esima%
\fi
}
```

Teens (feminine):

```
\newcommand{\@@TeenthstringFspanish}[1]{%
\ifcase#1\relax
D\'ecima%
\or Und\'ecima%
\or Duod\'ecima%
\or Decimotercera%
\or Decimocuarta%
\or Decimoquinta%
\or Decimosexta%
\or Decimos\'eptima%
\or Decimoctava%
\or Decimonovena%
\fi
}
```

Hundreds (feminine):

```
\newcommand{\@@HundredthstringFspanish}[1]{%
\ifcase#1\relax
\or Cent\'esima%
\or Ducent\'esima%
\or Tricent\'esima%
\or Cuadringent\'esima%
\or Quingent\'esima%
```

```
\or Sexcent\'esima%
\or Septing\'esima%
\or Octingent\'esima%
\or Noningent\'esima%
\fi}

```

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.)

```
\newcommand{\@@numberstringspanish}[2]{%
\ifnum#1>99999
\PackageError{fmtcount}{Out of range}%
{This macro only works for values less than 100000}%
\else
\ifnum#1<0
\PackageError{fmtcount}{Negative numbers not permitted}%
{This macro does not work for negative numbers, however
you can try typing "minus" first, and then pass the modulus of
this number}%
\fi
\fi
\def#2{}%
\@strctr=#1\relax \divide\@strctr by 1000\relax
\ifnum\@strctr>9
% #1 is greater or equal to 10000
  \divide\@strctr by 10
  \ifnum\@strctr>1
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
    \@strctr=#1 \divide\@strctr by 1000\relax
    \@modulo{\@strctr}{10}%
    \ifnum\@strctr>0\relax
       \let\@@fc@numstr#2\relax
       \edef#2{\@@fc@numstr\ \@andname\ \@unitstring{\@strctr}}%
    \fi
  \else
    \@strctr=#1\relax
    \divide\@strctr by 1000\relax
    \@modulo{\@strctr}{10}%
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
  \fi
  \let\@@fc@numstr#2\relax
  \edef#2{\@@fc@numstr\ \@thousand}%
\else
  \ifnum\@strctr>0\relax
    \ifnum\@strctr>1\relax
       \let\@@fc@numstr#2\relax
       \edef#2{\@@fc@numstr\@unitstring{\@strctr}\ }%
    \fi
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@thousand}%
```

```
  \fi
\fi
\@strctr=#1\relax \@modulo{\@strctr}{1000}%
\divide\@strctr by 100\relax
\ifnum\@strctr>0\relax
  \ifnum#1>1000\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\ }%
  \fi
  \@tmpstrctr=#1\relax
  \@modulo{\@tmpstrctr}{1000}%
  \ifnum\@tmpstrctr=100\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@tenstring{10}}%
  \else
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\@hundredstring{\@strctr}}%
  \fi
\fi
\@strctr=#1\relax \@modulo{\@strctr}{100}%
\ifnum#1>100\relax
  \ifnum\@strctr>0\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\ \@andname\ }%
  \fi
\fi
\ifnum\@strctr>29\relax
  \divide\@strctr by 10\relax
  \let\@@fc@numstr#2\relax
  \edef#2{\@@fc@numstr\@tenstring{\@strctr}}%
  \@strctr=#1\relax \@modulo{\@strctr}{10}%
  \ifnum\@strctr>0\relax
    \let\@@fc@numstr#2\relax
    \edef#2{\@@fc@numstr\ \@andname\ \@unitstring{\@strctr}}%
  \fi
\else
  \ifnum\@strctr<10\relax
    \ifnum\@strctr=0\relax
      \ifnum#1<100\relax
        \let\@@fc@numstr#2\relax
        \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
      \fi
    \else
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\@unitstring{\@strctr}}%
    \fi
  \else
    \ifnum\@strctr>19\relax
      \@modulo{\@strctr}{10}%
      \let\@@fc@numstr#2\relax
      \edef#2{\@@fc@numstr\@twentystring{\@strctr}}%
    \else
      \@modulo{\@strctr}{10}%
      \let\@@fc@numstr#2\relax
```

```
      \edef#2{\@@fc@numstr\@teenstring{\@strctr}}%
    \fi
  \fi
\fi
}
```

As above, but for ordinals

```
\newcommand{\@@ordinalstringspanish}[2]{%
\@strctr=#1\relax
\ifnum#1>99999
\PackageError{fmtcount}{Out of range}%
{This macro only works for values less than 100000}%
\else
\ifnum#1<0
\PackageError{fmtcount}{Negative numbers not permitted}%
{This macro does not work for negative numbers, however
you can try typing "minus" first, and then pass the modulus of
this number}%
\else
\def#2{}%
\ifnum\@strctr>999\relax
  \divide\@strctr by 1000\relax
  \ifnum\@strctr>1\relax
    \ifnum\@strctr>9\relax
      \@tmpstrctr=\@strctr
      \ifnum\@strctr<20
        \@modulo{\@tmpstrctr}{10}%
        \let\@@fc@ordstr#2\relax
        \edef#2{\@@fc@ordstr\@teenthstring{\@tmpstrctr}}%
      \else
        \divide\@tmpstrctr by 10\relax
        \let\@@fc@ordstr#2\relax
        \edef#2{\@@fc@ordstr\@tenthstring{\@tmpstrctr}}%
        \@tmpstrctr=\@strctr
        \@modulo{\@tmpstrctr}{10}%
        \ifnum\@tmpstrctr>0\relax
          \let\@@fc@ordstr#2\relax
          \edef#2{\@@fc@ordstr\@unitthstring{\@tmpstrctr}}%
        \fi
      \fi
    \else
       \let\@@fc@ordstr#2\relax
       \edef#2{\@@fc@ordstr\@unitstring{\@strctr}}%
    \fi
  \fi
  \let\@@fc@ordstr#2\relax
  \edef#2{\@@fc@ordstr\@thousandth}%
\fi
\@strctr=#1\relax
\@modulo{\@strctr}{1000}%
\ifnum\@strctr>99\relax
  \@tmpstrctr=\@strctr
  \divide\@tmpstrctr by 100\relax
  \ifnum#1>1000\relax
    \let\@@fc@ordstr#2\relax
```

```
    \edef#2{\@@fc@ordstr\ }%
  \fi
  \let\@@fc@ordstr#2\relax
  \edef#2{\@@fc@ordstr\@hundredthstring{\@tmpstrctr}}%
\fi
\@modulo{\@strctr}{100}%
\ifnum#1>99\relax
  \ifnum\@strctr>0\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\ }%
  \fi
\fi
\ifnum\@strctr>19\relax
  \@tmpstrctr=\@strctr
  \divide\@tmpstrctr by 10\relax
  \let\@@fc@ordstr#2\relax
  \edef#2{\@@fc@ordstr\@tenthstring{\@tmpstrctr}}%
  \@tmpstrctr=\@strctr
  \@modulo{\@tmpstrctr}{10}%
  \ifnum\@tmpstrctr>0\relax
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\ \@unitthstring{\@tmpstrctr}}%
  \fi
\else
  \ifnum\@strctr>9\relax
    \@modulo{\@strctr}{10}%
    \let\@@fc@ordstr#2\relax
    \edef#2{\@@fc@ordstr\@teenthstring{\@strctr}}%
  \else
    \ifnum\@strctr=0\relax
      \ifnum#1=0\relax
        \let\@@fc@ordstr#2\relax
        \edef#2{\@@fc@ordstr\@unitstring{0}}%
      \fi
    \else
      \let\@@fc@ordstr#2\relax
      \edef#2{\@@fc@ordstr\@unitthstring{\@strctr}}%
    \fi
  \fi
\fi
\fi
\fi
}
```

11.7 fc-UKenglish.def

UK English definitions

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

Check that fc-english.def has been loaded

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

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

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

```
\let\@ordinalFUKenglish\@ordinalMenglish
\let\@ordinalNUKenglish\@ordinalMenglish
\let\@numberstringMUKenglish\@numberstringMenglish
\let\@numberstringFUKenglish\@numberstringMenglish
\let\@numberstringNUKenglish\@numberstringMenglish
\let\@NumberstringMUKenglish\@NumberstringMenglish
\let\@NumberstringFUKenglish\@NumberstringMenglish
\let\@NumberstringNUKenglish\@NumberstringMenglish
\let\@ordinalstringMUKenglish\@ordinalstringMenglish
\let\@ordinalstringFUKenglish\@ordinalstringMenglish
\let\@ordinalstringNUKenglish\@ordinalstringMenglish
\let\@OrdinalstringMUKenglish\@OrdinalstringMenglish
\let\@OrdinalstringFUKenglish\@OrdinalstringMenglish
\let\@OrdinalstringNUKenglish\@OrdinalstringMenglish
```

11.8 fc-USenglish.def

US English definitions

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

Check that fc-english.def has been loaded

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

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

```
\let\@ordinalMUSenglish\@ordinalMenglish
\let\@ordinalFUSenglish\@ordinalMenglish
\let\@ordinalNUSenglish\@ordinalMenglish
\let\@numberstringMUSenglish\@numberstringMenglish
\let\@numberstringFUSenglish\@numberstringMenglish
\let\@numberstringNUSenglish\@numberstringMenglish
\let\@NumberstringMUSenglish\@NumberstringMenglish
\let\@NumberstringFUSenglish\@NumberstringMenglish
\let\@NumberstringNUSenglish\@NumberstringMenglish
\let\@ordinalstringMUSenglish\@ordinalstringMenglish
\let\@ordinalstringFUSenglish\@ordinalstringMenglish
\let\@ordinalstringNUSenglish\@ordinalstringMenglish
\let\@OrdinalstringMUSenglish\@OrdinalstringMenglish
\let\@OrdinalstringFUSenglish\@OrdinalstringMenglish
\let\@OrdinalstringNUSenglish\@OrdinalstringMenglish
```

11.9 fmtcount.sty

This section deals with the code for `fmtcount.sty`

```
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{fmtcount}[2007/06/14 v1.1]
\RequirePackage{ifthen}
\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.

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

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

```
\newcount\c@padzeroesN
\c@padzeroesN=1\relax
```

```
\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):

```
\@ifundefined{dateenglish}{}{\input{fc-english.def}}
\@ifundefined{l@UKenglish}{}{\input{fc-UKenglish.def}}
\@ifundefined{l@british}{}{\input{fc-british.def}}
\@ifundefined{l@USenglish}{}{\input{fc-USenglish.def}}
\@ifundefined{datespanish}{}{\input{fc-spanish.def}}
\@ifundefined{dateportuges}{}{\input{fc-portuges.def}}
\@ifundefined{datefrench}{}{\input{fc-french.def}}
\@ifundefined{dategerman}{%
\@ifundefined{datengerman}{}{\input{fc-german.def}}}{%
\input{fc-german.def}}
```

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

```
\def\fmtcount@french{france}
\define@key{fmtcount}{french}[france]{%
\@ifundefined{datefrench}{%
\PackageError{fmtcount}{Language 'french' not defined}{You need
to load babel before loading fmtcount}}{
\ifthenelse{\equal{#1}{france}
         \or\equal{#1}{swiss}
         \or\equal{#1}{belgian}}{%
         \def\fmtcount@french{#1}}{%
\PackageError{fmtcount}{Invalid value '#1' to french key}
{Option 'french' can only take the values 'france',
'belgian' or 'swiss'}}
}}
```

Key to determine how to display the ordinal

```
\define@key{fmtcount}{fmtord}{%
\ifthenelse{\equal{#1}{level}
          \or\equal{#1}{raise}
          \or\equal{#1}{user}}{
          \def\fmtcount@fmtord{#1}}{%
\PackageError{fmtcount}{Invalid value '#1' to fmtord key}
{Option 'fmtord' can only take the values 'level', 'raise'
or 'user'}}}
```

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

```
\newif\iffmtord@abbrv
\fmtord@abbrvfalse
\define@key{fmtcount}{abbrv}[true]{%
\ifthenelse{\equal{#1}{true}\or\equal{#1}{false}}{
          \csname fmtord@abbrv#1\endcsname}{%
\PackageError{fmtcount}{Invalid value '#1' to fmtord key}
{Option 'fmtord' can only take the values 'true' or
'false'}}}
```

Define command to set options.

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

```
\setkeys{fmtcount}{#1}%
\@ifundefined{datefrench}{}{%
\edef\@ordinalstringMfrench{\noexpand
\csname @ordinalstringMfrench\fmtcount@french\noexpand\endcsname}%
\edef\@ordinalstringFfrench{\noexpand
\csname @ordinalstringFfrench\fmtcount@french\noexpand\endcsname}%
\edef\@OrdinalstringMfrench{\noexpand
\csname @OrdinalstringMfrench\fmtcount@french\noexpand\endcsname}%
\edef\@OrdinalstringFfrench{\noexpand
\csname @OrdinalstringFfrench\fmtcount@french\noexpand\endcsname}%
\edef\@numberstringMfrench{\noexpand
\csname @numberstringMfrench\fmtcount@french\noexpand\endcsname}%
\edef\@numberstringFfrench{\noexpand
\csname @numberstringFfrench\fmtcount@french\noexpand\endcsname}%
\edef\@NumberstringMfrench{\noexpand
\csname @NumberstringMfrench\fmtcount@french\noexpand\endcsname}%
\edef\@NumberstringFfrench{\noexpand
\csname @NumberstringFfrench\fmtcount@french\noexpand\endcsname}%
}%
%
\ifthenelse{\equal{\fmtcount@fmtord}{level}}{%
\renewcommand{\fmtord}[1]{##1}}{%
\ifthenelse{\equal{\fmtcount@fmtord}{raise}}{%
\renewcommand{\fmtord}[1]{\textsuperscript{##1}}}{%
}}
}
```

Load confguration 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.

```
\InputIfFileExists{fmtcount.cfg}{%
\typeout{Using configuration file fmtcount.cfg}}{%
\typeout{No configuration file fmtcount.cfg found.}}
```

Declare options

```
\DeclareOption{level}{\def\fmtcount@fmtord{level}%
\def\fmtord#1{#1}}
\DeclareOption{raise}{\def\fmtcount@fmtord{raise}%
\def\fmtord#1{\textsuperscript{#1}}}
```

Process package options

```
\ProcessOptions
```

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

```
\newcount\@DT@modctr
\def\@modulo#1#2{%
\@DT@modctr=#1\relax
\divide \@DT@modctr by #2\relax
\multiply \@DT@modctr by #2\relax
\advance #1 by -\@DT@modctr}
```

The following registers are needed by `\@ordinal` etc

```
\newcount\@ordinalctr
\newcount\@orgargctr
\newcount\@strctr
```

```
\newcount\@tmpstrctr
```

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

```
\newif\if@DT@padzeroes
\newcount\@DT@loopN
\newcount\@DT@X
```

Binary

```
\newcommand{\@binary}[1]{%
\@DT@padzeroestrue
\@DT@loopN=17\relax
\@strctr=\@DT@loopN
\whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by 1}%
\@strctr=65536\relax
\@DT@X=#1\relax
\loop
\@DT@modctr=\@DT@X
\divide\@DT@modctr by \@strctr
\ifthenelse{\boolean{@DT@padzeroes} \and \(\@DT@modctr=0\) \and \(\@DT@loopN>\c@padzeroesN\)}{
\ifnum\@DT@modctr=0\else\@DT@padzeroesfalse\fi
\multiply\@DT@modctr by \@strctr
\advance\@DT@X by -\@DT@modctr
\divide\@strctr by 2\relax
\advance\@DT@loopN by -1\relax
\ifnum\@strctr>1
\repeat
\the\@DT@X}

\let\binarynum=\@binary
```

Octal

```
\newcommand{\@octal}[1]{%
\ifnum#1>32768
\PackageError{fmtcount}{Value of counter too large for \protect\@octal}{Maximum value 32768}
\else
\@DT@padzeroestrue
\@DT@loopN=6\relax
\@strctr=\@DT@loopN
\whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by 1}%
\@strctr=32768\relax
\@DT@X=#1\relax
\loop
\@DT@modctr=\@DT@X
\divide\@DT@modctr by \@strctr
\ifthenelse{\boolean{@DT@padzeroes} \and \(\@DT@modctr=0\) \and \(\@DT@loopN>\c@padzeroesN\)}{
\ifnum\@DT@modctr=0\else\@DT@padzeroesfalse\fi
\multiply\@DT@modctr by \@strctr
\advance\@DT@X by -\@DT@modctr
\divide\@strctr by 8\relax
\advance\@DT@loopN by -1\relax
\ifnum\@strctr>1
\repeat
\the\@DT@X
\fi}
```

```
\let\octalnum=\@octal
```

Lowercase hexadecimal

```
\newcommand{\@@hexadecimal}[1]{\ifcase#10\or1\or2\or3\or4\or5\or6\or7\or8\or9\or a\or b\or c\o

\newcommand{\@hexadecimal}[1]{%
\@DT@padzeroestrue
\@DT@loopN=5\relax
\@strctr=\@DT@loopN
\whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by 1}%
\@strctr=65536\relax
\@DT@X=#1\relax
\loop
\@DT@modctr=\@DT@X
\divide\@DT@modctr by \@strctr
\ifthenelse{\boolean{@DT@padzeroes} \and \(\@DT@modctr=0\) \and \(\@DT@loopN>\c@padzeroesN\)}{
\ifnum\@DT@modctr=0\else\@DT@padzeroesfalse\fi
\multiply\@DT@modctr by \@strctr
\advance\@DT@X by -\@DT@modctr
\divide\@strctr by 16\relax
\advance\@DT@loopN by -1\relax
\ifnum\@strctr>1
\repeat
\@@hexadecimal\@DT@X}

\let\hexadecimalnum=\@hexadecimal
```

Uppercase hexadecimal

```
\newcommand{\@@Hexadecimal}[1]{\ifcase#10\or1\or2\or3\or4\or5\or6\or
7\or8\or9\or A\or B\or C\or D\or E\or F\fi}

\newcommand{\@Hexadecimal}[1]{%
\@DT@padzeroestrue
\@DT@loopN=5\relax
\@strctr=\@DT@loopN
\whiledo{\@strctr<\c@padzeroesN}{0\advance\@strctr by 1}%
\@strctr=65536\relax
\@DT@X=#1\relax
\loop
\@DT@modctr=\@DT@X
\divide\@DT@modctr by \@strctr
\ifthenelse{\boolean{@DT@padzeroes} \and \(\@DT@modctr=0\) \and \(\@DT@loopN>\c@padzeroesN\)}{
\ifnum\@DT@modctr=0\else\@DT@padzeroesfalse\fi
\multiply\@DT@modctr by \@strctr
\advance\@DT@X by -\@DT@modctr
\divide\@strctr by 16\relax
\advance\@DT@loopN by -1\relax
\ifnum\@strctr>1
\repeat
\@@Hexadecimal\@DT@X}

\let\Hexadecimalnum=\@Hexadecimal
```

Uppercase alphabetical representation (a … z aa … zz)

```
\newcommand{\@aaalph}[1]{%
```

```
\@DT@loopN=#1\relax
\advance\@DT@loopN by -1\relax
\divide\@DT@loopN by 26\relax
\@DT@modctr=\@DT@loopN
\multiply\@DT@modctr by 26\relax
\@DT@X=#1\relax
\advance\@DT@X by -1\relax
\advance\@DT@X by -\@DT@modctr
\advance\@DT@loopN by 1\relax
\advance\@DT@X by 1\relax
\loop
\@alph\@DT@X
\advance\@DT@loopN by -1\relax
\ifnum\@DT@loopN>0
\repeat
}

\let\aaalphnum=\@aaalph
```

Uppercase alphabetical representation (a … z aa … zz)

```
\newcommand{\@AAAlph}[1]{%
\@DT@loopN=#1\relax
\advance\@DT@loopN by -1\relax
\divide\@DT@loopN by 26\relax
\@DT@modctr=\@DT@loopN
\multiply\@DT@modctr by 26\relax
\@DT@X=#1\relax
\advance\@DT@X by -1\relax
\advance\@DT@X by -\@DT@modctr
\advance\@DT@loopN by 1\relax
\advance\@DT@X by 1\relax
\loop
\@Alph\@DT@X
\advance\@DT@loopN by -1\relax
\ifnum\@DT@loopN>0
\repeat
}

\let\AAAlphnum=\@AAAlph
```

Lowercase alphabetical representation

```
\newcommand{\@abalph}[1]{%
\ifnum#1>17576
\PackageError{fmtcount}{Value of counter too large for \protect\@abalph}{Maximum value 17576}
\else
\@DT@padzeroestrue
\@strctr=17576\relax
\@DT@X=#1\relax
\advance\@DT@X by -1\relax
\loop
\@DT@modctr=\@DT@X
\divide\@DT@modctr by \@strctr
\ifthenelse{\boolean{@DT@padzeroes} \and \(\@DT@modctr=1\)}{}{\@alph\@DT@modctr}%
\ifnum\@DT@modctr=1\else\@DT@padzeroesfalse\fi
\multiply\@DT@modctr by \@strctr
```

```
\advance\@DT@X by -\@DT@modctr
\divide\@strctr by 26\relax
\ifnum\@strctr>1
\repeat
\advance\@DT@X by 1\relax
\@alph\@DT@X
\fi}

\let\abalphnum=\@abalph
```

Uppercase alphabetical representation

```
\newcommand{\@ABAlph}[1]{%
\ifnum#1>17576
\PackageError{fmtcount}{Value of counter too large for \protect\@ABAlph}{Maximum value 17576}
\else
\@DT@padzeroestrue
\@strctr=17576\relax
\@DT@X=#1\relax
\advance\@DT@X by -1\relax
\loop
\@DT@modctr=\@DT@X
\divide\@DT@modctr by \@strctr
\ifthenelse{\boolean{@DT@padzeroes} \and \(\@DT@modctr=1\)}{}{\@Alph\@DT@modctr}%
\ifnum\@DT@modctr=1\else\@DT@padzeroesfalse\fi
\multiply\@DT@modctr by \@strctr
\advance\@DT@X by -\@DT@modctr
\divide\@strctr by 26\relax
\ifnum\@strctr>1
\repeat
\advance\@DT@X by 1\relax
\@Alph\@DT@X
\fi}

\let\ABAlphnum=\@ABAlph
```

Recursive command to count number of characters in argument. `\@strctr` should be set to zero before calling it.

```
\def\@fmtc@count#1#2\relax{%
\if\relax#1
\else
\advance\@strctr by 1\relax
\@fmtc@count#2\relax
\fi}
```

Internal decimal macro:

```
\newcommand{\@decimal}[1]{%
\@strctr=0\relax
\expandafter\@fmtc@count\number#1\relax
\@DT@loopN=\c@padzeroesN
\advance\@DT@loopN by -\@strctr
\ifnum\@DT@loopN>0\relax
\@strctr=0\relax
\whiledo{\@strctr < \@DT@loopN}{0\advance\@strctr by 1}%
\fi
\number#1\relax
```

```
}

\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.

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

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

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

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

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

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

```
\def\@ordinalnum#1[#2]{{%
\ifthenelse{\equal{#2}{f}}{%
\protect\@ordinalF{#1}{\@fc@ordstr}}{%
\ifthenelse{\equal{#2}{n}}{%
\protect\@ordinalN{#1}{\@fc@ordstr}}{%
\ifthenelse{\equal{#2}{m}}{}{%
\PackageError{fmtcount}{Invalid gender option `#2'}{%
Available options are m, f or n}}%
\protect\@ordinalM{#1}{\@fc@ordstr}}}\@fc@ordstr}}
```

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

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

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

```
\newcommand*{\storeordinalnum}[2]{%
\@ifnextchar[{\@storeordinalnum{#1}{#2}}{%
\@storeordinalnum{#1}{#2}[m]}}
```

Store ordinal according to gender:

```
\def\@storeordinalnum#1#2[#3]{%
\ifthenelse{\equal{#3}{f}}{%
\protect\@ordinalF{#2}{\@fc@ord}}{%
\ifthenelse{\equal{#3}{n}}{%
\protect\@ordinalN{#2}{\@fc@ord}}{%
```

```
\ifthenelse{\equal{#3}{m}}{}{%
\PackageError{fmtcount}{Invalid gender option `#3'}{%
Available options are m or f}}%
\protect\@ordinalM{#2}{\@fc@ord}}}%
\expandafter\let\csname @fcs@#1\endcsname\@fc@ord}
```

Get stored information:

```
\newcommand*{\FMCuse}[1]{\csname @fcs@#1\endcsname}
```

Display ordinal as a string (argument is a counter)

```
\newcommand{\ordinalstring}[1]{%
\expandafter\protect\expandafter\ordinalstringnum{%
\expandafter\the\csname c@#1\endcsname}}
```

Display ordinal as a string (argument is a count register or number.)

```
\newcommand{\ordinalstringnum}[1]{%
\@ifnextchar[{\@ordinal@string{#1}}{\@ordinal@string{#1}[m]}}
```

Display ordinal as a string according to gender.

```
\def\@ordinal@string#1[#2]{{%
\ifthenelse{\equal{#2}{f}}{%
\protect\@ordinalstringF{#1}{\@fc@ordstr}}{%
\ifthenelse{\equal{#2}{n}}{%
\protect\@ordinalstringN{#1}{\@fc@ordstr}}{%
\ifthenelse{\equal{#2}{m}}{}{%
\PackageError{fmtcount}{Invalid gender option `#2' to
\string\ordinalstring}{Available options are m, f or f}}%
\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.

```
\newcommand{\storeordinalstring}[2]{%
\expandafter\protect\expandafter\storeordinalstringnum{#1}{%
\expandafter\the\csname c@#2\endcsname}}
```

Store textual representation of number. First argument is identifying name, second argument is a count register or number.

```
\newcommand{\storeordinalstringnum}[2]{%
\@ifnextchar[{\@store@ordinal@string{#1}{#2}}{%
\@store@ordinal@string{#1}{#2}[m]}}
```

Store textual representation of number according to gender.

```
\def\@store@ordinal@string#1#2[#3]{%
\ifthenelse{\equal{#3}{f}}{%
\protect\@ordinalstringF{#2}{\@fc@ordstr}}{%
\ifthenelse{\equal{#3}{n}}{%
\protect\@ordinalstringN{#2}{\@fc@ordstr}}{%
\ifthenelse{\equal{#3}{m}}{}{%
\PackageError{fmtcount}{Invalid gender option `#3' to
\string\ordinalstring}{Available options are m, f or n}}%
\protect\@ordinalstringM{#2}{\@fc@ordstr}}}%
\expandafter\let\csname @fcs@#1\endcsname\@fc@ordstr}
```

Display ordinal as a string with initial letters in upper case (argument is a counter)

```
\newcommand{\Ordinalstring}[1]{%
\expandafter\protect\expandafter\Ordinalstringnum{%
\expandafter\the\csname c@#1\endcsname}}
```

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

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

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

```
\def\@Ordinal@string#1[#2]{{%
\ifthenelse{\equal{#2}{f}}{%
\protect\@OrdinalstringF{#1}{\@fc@ordstr}}{%
\ifthenelse{\equal{#2}{n}}{%
\protect\@OrdinalstringN{#1}{\@fc@ordstr}}{%
\ifthenelse{\equal{#2}{m}}{}{%
\PackageError{fmtcount}{Invalid gender option `#2'}{%
Available options are m, f or n}}%
\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.

```
\newcommand{\storeOrdinalstring}[2]{%
\expandafter\protect\expandafter\storeOrdinalstringnum{#1}{%
\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.

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

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

```
\def\@store@Ordinal@string#1#2[#3]{%
\ifthenelse{\equal{#3}{f}}{%
\protect\@OrdinalstringF{#2}{\@fc@ordstr}}{%
\ifthenelse{\equal{#3}{n}}{%
\protect\@OrdinalstringN{#2}{\@fc@ordstr}}{%
\ifthenelse{\equal{#3}{m}}{}{%
\PackageError{fmtcount}{Invalid gender option `#3'}{%
Available options are m or f}}%
\protect\@OrdinalstringM{#2}{\@fc@ordstr}}}%
\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.

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

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

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

Gender is specified as an optional argument at the end.

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

```
\protect\@ordinalstringF{#2}{\@fc@ordstr}}{%
\ifthenelse{\equal{#3}{n}}{%
\protect\@ordinalstringN{#2}{\@fc@ordstr}}{%
\ifthenelse{\equal{#3}{m}}{}{%
\PackageError{fmtcount}{Invalid gender option `#3'}{%
Available options are m or f}}%
\protect\@ordinalstringM{#2}{\@fc@ordstr}}}%
\expandafter\edef\csname @fcs@#1\endcsname{%
\noexpand\MakeUppercase{\@fc@ordstr}}}
```

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

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

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

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

Gender is specified as an optional argument at the end.

```
\def\@ORDINAL@string#1[#2]{{%
\ifthenelse{\equal{#2}{f}}{%
\protect\@ordinalstringF{#1}{\@fc@ordstr}}{%
\ifthenelse{\equal{#2}{n}}{%
\protect\@ordinalstringN{#1}{\@fc@ordstr}}{%
\ifthenelse{\equal{#2}{m}}{}{%
\PackageError{fmtcount}{Invalid gender option `#2'}{%
Available options are m, f or n}}%
\protect\@ordinalstringM{#1}{\@fc@ordstr}}}%
\MakeUppercase{\@fc@ordstr}}}
```

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

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

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

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

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

```
\def\@store@number@string#1#2[#3]{%
\ifthenelse{\equal{#3}{f}}{%
\protect\@numberstringF{#2}{\@fc@numstr}}{%
\ifthenelse{\equal{#3}{n}}{%
\protect\@numberstringN{#2}{\@fc@numstr}}{%
\ifthenelse{\equal{#3}{m}}{}{%
\PackageError{fmtcount}{Invalid gender option `#3'}{%
Available options are m, f or n}}%
\protect\@numberstringM{#2}{\@fc@numstr}}}%
\expandafter\let\csname @fcs@#1\endcsname\@fc@numstr}
```

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

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

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

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

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

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

```
\def\@number@string#1[#2]{{%
\ifthenelse{\equal{#2}{f}}{%
\protect\@numberstringF{#1}{\@fc@numstr}}{%
\ifthenelse{\equal{#2}{n}}{%
\protect\@numberstringN{#1}{\@fc@numstr}}{%
\ifthenelse{\equal{#2}{m}}{}{%
\PackageError{fmtcount}{Invalid gender option `#2'}{%
Available options are m, f or n}}%
\protect\@numberstringM{#1}{\@fc@numstr}}}\@fc@numstr}}
```

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

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

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

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

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

```
\def\@store@Number@string#1#2[#3]{%
\ifthenelse{\equal{#3}{f}}{%
\protect\@NumberstringF{#2}{\@fc@numstr}}{%
\ifthenelse{\equal{#3}{n}}{%
\protect\@NumberstringN{#2}{\@fc@numstr}}{%
\ifthenelse{\equal{#3}{m}}{}{%
\PackageError{fmtcount}{Invalid gender option `#3'}{%
Available options are m, f or n}}%
\protect\@NumberstringM{#2}{\@fc@numstr}}}%
\expandafter\let\csname @fcs@#1\endcsname\@fc@numstr}
```

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

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

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

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

Gender is specified as an optional argument at the end.

```
\def\@Number@string#1[#2]{{%
\ifthenelse{\equal{#2}{f}}{%
\protect\@NumberstringF{#1}{\@fc@numstr}}{%
\ifthenelse{\equal{#2}{n}}{%
\protect\@NumberstringN{#1}{\@fc@numstr}}{%
\ifthenelse{\equal{#2}{m}}{}{%
```

```
\PackageError{fmtcount}{Invalid gender option `#2'}{%
Available options are m, f or n}}%
\protect\@NumberstringM{#1}{\@fc@numstr}}}\@fc@numstr}}
```

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

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

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

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

Gender is specified as an optional argument at the end.

```
\def\@store@NUMBER@string#1#2[#3]{%
\ifthenelse{\equal{#3}{f}}{%
\protect\@numberstringF{#2}{\@fc@numstr}}{%
\ifthenelse{\equal{#3}{n}}{%
\protect\@numberstringN{#2}{\@fc@numstr}}{%
\ifthenelse{\equal{#3}{m}}{}{%
\PackageError{fmtcount}{Invalid gender option `#3'}{%
Available options are m or f}}%
\protect\@numberstringM{#2}{\@fc@numstr}}}%
\expandafter\edef\csname @fcs@#1\endcsname{%
\noexpand\MakeUppercase{\@fc@numstr}}}
```

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

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

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

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

Gender is specified as an optional argument at the end.

```
\def\@NUMBER@string#1[#2]{{%
\ifthenelse{\equal{#2}{f}}{%
\protect\@numberstringF{#1}{\@fc@numstr}}{%
\ifthenelse{\equal{#2}{n}}{%
\protect\@numberstringN{#1}{\@fc@numstr}}{%
\ifthenelse{\equal{#2}{m}}{}{%
\PackageError{fmtcount}{Invalid gender option `#2'}{%
Available options are m, f or n}}%
\protect\@numberstringM{#1}{\@fc@numstr}}}%
\MakeUppercase{\@fc@numstr}}}
```

Number representations in other bases. Binary:

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

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

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

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

As before, but upper case.

```
\providecommand{\AAAlph}[1]{%
\expandafter\protect\expandafter\@AAAlph{%
\expandafter\the\csname c@#1\endcsname}}
```

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

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

As above, but upper case.

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

Hexadecimal:

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

As above, but in upper case.

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

Octal:

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

Decimal:

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

11.9.1 Multilinguage Definitions

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

```
\def\@setdef@ultfmtcount{
\@ifundefined{@ordinalMenglish}{\input{fc-english.def}}{}
\def\@ordinalstringM{\@ordinalstringMenglish}
\let\@ordinalstringF=\@ordinalstringMenglish
\let\@ordinalstringN=\@ordinalstringMenglish
\def\@OrdinalstringM{\@OrdinalstringMenglish}
\let\@OrdinalstringF=\@OrdinalstringMenglish
\let\@OrdinalstringN=\@OrdinalstringMenglish
\def\@numberstringM{\@numberstringMenglish}
\let\@numberstringF=\@numberstringMenglish
\let\@numberstringN=\@numberstringMenglish
\def\@NumberstringM{\@NumberstringMenglish}
\let\@NumberstringF=\@NumberstringMenglish
```

```
\let\@NumberstringN=\@NumberstringMenglish
\def\@ordinalM{\@ordinalMenglish}
\let\@ordinalF=\@ordinalM
\let\@ordinalN=\@ordinalM
}
```

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

```
\def\@set@mulitling@fmtcount{%
%
\def\@numberstringM{\@ifundefined{@numberstringM\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@numberstringM}}{%
\csname @numberstringM\languagename\endcsname}}%
%
\def\@numberstringF{\@ifundefined{@numberstringF\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@numberstringF}}{%
\csname @numberstringF\languagename\endcsname}}%
%
\def\@numberstringN{\@ifundefined{@numberstringN\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@numberstringN}}{%
\csname @numberstringN\languagename\endcsname}}%
%
\def\@NumberstringM{\@ifundefined{@NumberstringM\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@NumberstringM}}{%
\csname @NumberstringM\languagename\endcsname}}%
%
\def\@NumberstringF{\@ifundefined{@NumberstringF\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@NumberstringF}}{%
\csname @NumberstringF\languagename\endcsname}}%
%
\def\@NumberstringN{\@ifundefined{@NumberstringN\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@NumberstringN}}{%
\csname @NumberstringN\languagename\endcsname}}%
%
\def\@ordinalM{\@ifundefined{@ordinalM\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@ordinalM}}{%
\csname @ordinalM\languagename\endcsname}}%
%
\def\@ordinalF{\@ifundefined{@ordinalF\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@ordinalF}}{%
```

```
\csname @ordinalF\languagename\endcsname}}%
%
\def\@ordinalN{\@ifundefined{@ordinalN\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@ordinalN}}{%
\csname @ordinalN\languagename\endcsname}}%
%
\def\@ordinalstringM{\@ifundefined{@ordinalstringM\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@ordinalstringM}}{%
\csname @ordinalstringM\languagename\endcsname}}%
%
\def\@ordinalstringF{\@ifundefined{@ordinalstringF\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@ordinalstringF}}{%
\csname @ordinalstringF\languagename\endcsname}}%
%
\def\@ordinalstringN{\@ifundefined{@ordinalstringN\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@ordinalstringN}}{%
\csname @ordinalstringN\languagename\endcsname}}%
%
\def\@OrdinalstringM{\@ifundefined{@OrdinalstringM\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@OrdinalstringM}}{%
\csname @OrdinalstringM\languagename\endcsname}}%
%
\def\@OrdinalstringF{\@ifundefined{@OrdinalstringF\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@OrdinalstringF}}{%
\csname @OrdinalstringF\languagename\endcsname}}%
%
\def\@OrdinalstringN{\@ifundefined{@OrdinalstringN\languagename}{%
\PackageError{fmtcount}{No support for language '\languagename'}{%
The fmtcount package currently does not support language
'\languagename' for command \string\@OrdinalstringN}}{%
\csname @OrdinalstringN\languagename\endcsname}}
}
```

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

```
\@ifpackageloaded{babel}{%
\ifthenelse{\equal{\languagename}{nohyphenation}\or
\equal{languagename}{english}}{\@setdef@ultfmtcount}{%
\@set@mulitling@fmtcount}
}{%
\@ifpackageloaded{ngerman}{%
\@ifundefined{@numberstringMgerman}{%
\input{fc-german.def}}{}\@set@mulitling@fmtcount}{%
\@setdef@ultfmtcount}}
```

Backwards compatibility:

```
\let\@ordinal=\@ordinalM
\let\@ordinalstring=\@ordinalstringM
\let\@Ordinalstring=\@OrdinalstringM
\let\@numberstring=\@numberstringM
\let\@Numberstring=\@NumberstringM
```