The xistercian package

Jonathan P. Spratte∗

2021-07-17 v1.0

Contents

1 Documentation
  1.1 Introduction ...........................................
  1.2 Macros ..............................................
  1.3 Options ............................................
  1.4 Bold Ciśtercian Numerals ..............................

2 Implementation
  2.1 Variables ............................................
  2.2 Options ............................................
  2.3 User macros ........................................
  2.4 Parsing ............................................
    2.4.1 Small Auxiliaries ..........................
    2.4.2 Input Parsing ................................
  2.5 Drawing the Digits ..............................
  2.6 Rebuild the Glyphs ..............................

Index ..............................................

1 Documentation

The xistercian package provides Ciśtercian numerals for use in \LaTeX. The name is chosen to be xistercian instead of ciśtercian because I’ve learned that a (currently) unpublished package by that name already exists and I don’t want to be an evil name-stealer.

∗jspratte@yahoo.de
1.1 Introduction

Cistercian numerals are a system to denote the numbers from 1 to 9999 with a single glyph. They use a slant (₁) that is used as zero by this package, and add the digits to that using small tick marks, the place denoting the digits value. On the upper right means units (₁), upper left tens (₁₁), lower right hundreds (₁₁₁), and lower left thousands (₁₁₁₁). Figure 1 gives an overview over the base glyphs.

In the medieval times it was quite common (according to Wikipedia more common than the vertical style) to use the Cistercian numerals horizontally instead. This package also supports that, in which case the numerals are rotated by 90 degrees counterclockwise, see figure 2.

All glyphs have the same bounding box, are drawn using the pgf package, and cached inside TeX box registers. Therefore the performance is quite well, at least well enough to actually consider using these numerals without a major performance hit. But it still slows down the compilation, especially if the glyphs have to be redrawn often. By default they will be redrawn if the font size changed since the last usage inside the current group scope, but you can specify which font feature changes should cause a redraw.

Since Cistercian numerals quite compactly cover a big range of natural numbers they might be used to compactly keep track of running numbers. For instance, to use Cistercian numerals as page numbers you can use:

\pagenumbering{cistercian}

or to use them as footnotes:

\renewcommand\thefootnote{\cistercian{footnotes}}
The package tries to play nice on hyperref if it is loaded, but bookmarks containing material that should've been a Cïòlercian numeral will have a leading CISTER in front of the numeric value to give unique strings compared to \arabic.

Negative numbers just keep a leading -, and numbers with an absolute number greater than 9999 are displayed with multiple Cïòlercian digits. For instance \(-12345678\) is displayed as -19. And in the case that four consecutive digits are zeros they get displayed as that: 0 is 10000.

1.2 Macros

\cistercian{\langle counter\rangle}

Prints the value of the \TeX\ (\langle counter\rangle) as a Cïòlercian numeral (similar to how \roman prints a (\langle counter\rangle) as a Roman numeral). It can also be used as \pagenumbering {\cistercian} to change the page numbering, etc.

\cisterciannum{\langle integer\rangle}

Prints the (\langle integer\rangle) as a Cïòlercian numeral. The (\langle integer\rangle) has to be provided as a string containing only digits and optionally a single leading - (spaces are ignored). This can handle integers of almost arbitrary size (well, an integer with more than 2147483647 places will be an issue).

\cisterciannumE{\langle integer expression\rangle}

Evaluates the (\langle integer expression\rangle) and prints the result as a Cïòlercian numeral.

\cisterciansetup{\langle key=value,...\rangle}

Can be used to locally change the options after the package was loaded. The glyphs of the Cïòlercian numerals aren't automatically updated when you change the setup.

\cistercianredraw
\cistercianredrawlazy

When this is used the glyphs will be redrawn for the current group. The lazy variant will not immediately redraw the glyphs, but instead ensure that the next usage of a Cïòlercian numeral in the current scope will redraw the glyphs. The lazy variant could result in the glyphs being redrawn multiple times (if Cïòlercian numerals are used in different nested groups), while the normal variant might result in the glyphs being redrawn too eagerly.

\cistercianstyle{\langle name\rangle}{\langle key=value,...\rangle}

With this macro you can define a key called (\langle name\rangle) that'll set the options in the (\langle key=value\rangle) list if used inside \cisterciansetup. Only new names are allowed.

For example, the key horizontal is equivalently defined to

\cistercianstyle{horizontal}{o=h,wd=0.62em,ht=1ex}
Figure 3: Measurements of a glyph. The red rectangle is the bounding box, the dot on the left shows the reference point placed on the surrounding baseline.

1.3 Options

The options described here are all usable as package options (except when explicitly slated otherwise). xislercian doesn’t parse the global options provided to the documentclass. Additionally you can change the options (except for the debug option) using \cisterciansetup.

For the options width and height, and probably also the {strokes}, you should use dimensions using \texttt{ex} or \texttt{em} to get sizes depending on the current font size. For all three of those options the \texttt{⟨dimension expression⟩} is evaluated (using the setup specified in font) every time the font changed according to the redraw option’s feature list (by default only when the size changed) since the last time they were drawn.

Initial values if present are printed on the right. A small graphic explaining most of the size related options is shown in figure 3.

\begin{verbatim}
width   width = ⟨⟨dimension expression⟩⟩ .4em
wd      Set the width of the Ci\'slercian numeral digits.

height  height = ⟨⟨dimension expression⟩⟩ 1.55ex
ht      Set the height of the Ci\'slercian numeral digits.

bound-x bound-x = ⟨⟨float⟩⟩ 1.2
bx      Specifies the factor the bounding box in x-direction is bigger than the width. An empty ⟨⟨float⟩⟩ is considered the same as 1.

bound-y bound-y = ⟨⟨float⟩⟩ empty
by      Specifies the factor the bounding box in y-direction is bigger than the height. An empty ⟨⟨float⟩⟩ is considered the same as 1.
\end{verbatim}
baseline = \{\langle float\rangle\} \hspace{1em} empty

Sets the baseline of the symbols in multiples of the negative height. 1 (or empty) results in the symbol starting on the surrounding baseline, bigger values shift the symbols upwards, smaller values downwards.

\texttt{stroke-v = \{\texttt{\langle dimension expression\rangle}\}} \hspace{1em} .18\text{ex}

Sets the stroke thickness of vertical strokes.

\texttt{stroke-h = \{\texttt{\langle dimension expression\rangle}\}} \hspace{1em} .1\text{ex}

Sets the stroke thickness of horizontal strokes.

\texttt{stroke-du = \{\texttt{\langle dimension expression\rangle}\}} \hspace{1em} .1\text{ex}

Sets the stroke thickness of diagonal strokes going from lower left to upper right.

\texttt{stroke-dd = \{\texttt{\langle dimension expression\rangle}\}} \hspace{1em} .18\text{ex}

Sets the stroke thickness of diagonal strokes going from upper left to lower right.

\texttt{stroke = \{\texttt{\langle dimension expression\rangle}\}}

Sets all stroke thicknesses at once.

\texttt{fraction = \{\langle float\rangle\}} \hspace{1em} /6

The Čišlercian digits are drawn only in a fraction of the total bounding box of each symbol. The ones are drawn (with vertical Čišlercians) in the upper right rectangle. The width of that rectangle is determined by \texttt{width}, and the upper border by \texttt{height}, and with this option you can set the lower border as a fraction of the height. You can either just give a factor as a \texttt{\langle float\rangle}, or just pass in a divisor as an \texttt{\langle integer\rangle}, or both. An empty \texttt{\langle float\rangle} equals 1.

Consider the following example (result on the right):

\begin{verbatim}
\{\cisterciannum{9}\}
\{\cisterciannum{9}\}
\end{verbatim}

\texttt{fraction = \{\langle float\rangle\}/\{\langle integer\rangle\}}

\texttt{\langle choice\rangle}

Sets the orientation of the Čišlercian numerals. Choices are horizontal (or \texttt{h} for short) or vertical (or \texttt{v} for short). You'll most likely want to change the \texttt{width} and \texttt{height} of the symbols if you decide to change this.

\texttt{horizontal}

Same as setting \texttt{orientation = horizontal, width = 0.62em, height = 1ex}.

\texttt{vertical}

Same as setting \texttt{orientation = vertical, width = 0.4em, height = 1.55ex}. 

\end{verbatim}
There are alternative forms of the digit 5, while the most often used one nowadays seems to be the triangle form, this package also supports two other variants. Those are called dot and dash. While if you choose triangle you can’t give an additional value, if your choice is dot or dash you can customise those using a second equals sign and some value. If you don’t customise them they use their respective initial value (or last value if you changed the value at some point in time).

For dot you can also specify the radius as a dimension. The default is .09ex.

For dash you can specify the length of the dash in multiples of the used stroke thickness (which might differ for the horizontal and vertical symbols), an empty value is considered the same as 1. The default is empty.

A comparison of the three alternate forms is shown in figure 4.

The usage might look like any of the following:

```
cisterciansetup
{
  5=triangle,
  5=dot, 5=dot=.1ex, 5={dot=.1ex},
  5=dash, 5=dash=2, 5={dash=2}
}
```

Set the font which xiślercian uses locally to evaluate the dimensions given to width, height, and stroke (only font switches work here, you can’t use \texttt like \texttt, instead use \ttfamily).
Seleőľs which font features to consider when deciding to redraw the glyphs. If one of
the features has changed (or the features to consider) the glyphs will be redrawn. This
is especially handy if you change the font used while evaluating the given dimensions.
The available features in the comma separated ⟨feature-list⟩ are encoding, family,
series, shape, size, and color.

If you see that a Cįšłercian numeral doesn't have the correct colour of the surrounding
text you need to add color to the font features (xįšłercian uses some hack to get the
boxed glyphs to respect the surrounding colour, but that hack might fail in certain cir-
cumstances).

If you use the += variant of this option the ⟨feature-list⟩ is added to the existing
features, else they overwrite any existing configuration.

Without a value the key will issue \cistercianredrawlazy, and if you use the !
variant it'll use \cistercianredraw. This usage is only available after the package
was loaded.

Only usable as a package option. If this is used a bit of debugging information is printed
in the log file and terminal, and some keys try to evaluate the given argument on the
spot instead of lazily. This might help to locate faulty input.

1.4 Bold Cišłercian Numerals

While the package doesn’t directly support bold glyphs, it is possible to utilize the hooks
provided by the NFSS to pick different stroke widths whenever the font is changed to
to bfseries and back to mdseries:

```latex
\AddToHook{bfseries}
  {%
    \cisterciansetup{sv=.27ex,sdd=.27ex,sh=.15ex,sdu=.15ex,wd=.5em}%
    \cistercianredrawlazy
  }
\AddToHook{mdseries}
  {%
    \cisterciansetup{sv=.18ex,sdd=.18ex,sh=.1ex,sdu=.1ex,wd=.4em}%
    \cistercianredrawlazy
  }
\AddToHook{bfseries}
\AddToHook{mdseries}
\AddToHook{bfseries}
\AddToHook{mdseries}
\AddToHook{bfseries}
\AddToHook{mdseries}
\AddToHook{bfseries}
\AddToHook{mdseries}
\AddToHook{bfseries}
\AddToHook{mdseries}
\AddToHook{bfseries}
\AddToHook{mdseries}
\AddToHook{bfseries}
\AddToHook{mdseries}
\AddToHook{bfseries}
\AddToHook{mdseries}
\AddToHook{bfseries}
\AddToHook{mdseries}
\AddToHook{bfseries}
\AddToHook{mdseries}
\AddToHook{bfseries}
\AddToHook{mdseries}
```

This results in \cisterciannum{2} different \enquote{series}.\par
\bfseries
This results in \cisterciannum{2} different \enquote{series}.

This results in 1 different “series”.
**This results in 1 different “series”**.

Of course, instead of doing \cistercianredrawlazy in the above code, we could as
well do the following equivalent thing and let xįšłercian determine when to redraw:
\AddToHook{bfseries}
  {\cisterciansetup{sv=.27ex,sdd=.27ex,sh=.15ex,sdu=.15ex,wd=.5em}}
\AddToHook{mdseries}
  {\cisterciansetup{sv=.18ex,sdd=.18ex,sh=.1ex,sdu=.1ex,wd=.4em}}
\cisterciansetup{redraw+=series}
2 Implementation

First we load the required packages
\RequirePackage{pgf,expkv-opt}

2.1 Variables

There are variables stored as macros to be evaluated later, as well as register types for faster access during the glyph drawing.

\newcommand*\xister@th{.1ex}
\newcommand*\xister@tv{.18ex}
\newcommand*\xister@tdu{.1ex}
\newcommand*\xister@tdd{.18ex}
\newcommand*\xister@x{.4em}
\newcommand*\xister@y{1.55ex}
\newcommand*\xister@bx{1.2}
\newcommand*\xister@by{}
\newcommand*\xister@bs{}
\newcommand*\xister@font{\normalfont}
\newcommand*\xister@five@dot{.09ex}
\newcommand*\xister@five@dash{}
\newcommand*\xister@share@div{/6}
\newcommand*\xister@share@mul{}
\newcommand*\xister@dbg{}
\newcommand*\xister@five@triangle@true
\newcommand*\xister@five@triangle@
\newcommand*\xister@digitbox@0
\newdimen\xister@X
\newdimen\xister@Y
\newdimen\xister@XY@share
\newdimen\xister@Th
\newdimen\xister@Tv
\newdimen\xister@Tdu
\newdimen\xister@Tdd
\newdimen\xister@TMPA
\newdimen\xister@TMPB

We also need a bunch of box registers (37 of them to be precise). For the loop we borrow an expl3 function.
\ExplSyntaxOn
\int_step_inline:nn {9} { }
\ExplSyntaxOff
\xister@dbg@dim
\xister@dbg@float

These are initially just \texttt{\textbackslash gobble}, but the debug option might change things.
\let\xister@dbg@dim\texttt{\textbackslash gobble}
\let\xister@dbg@div\texttt{\textbackslash gobble}
\let\xister@dbg@float\texttt{\textbackslash gobble}

(End definition for \texttt{\textbackslash xister@dbg@dim and \textbackslash xister@dbg@float.})
2.2 Options

Some macros are required to parse options.

\texttt{\xister@if@slash}

We need to test whether a slash is inside an argument to parse some user option. Since this isn't needed inside an inner loop we don't create an optimised version and only borrow a generic expl3 function.

\texttt{\ExplSyntaxOn}
\texttt{\cs_new_protected:Npn \xister@if@slash #1 { \tl_if_in:nnTF {#1} { / } { } }}
\texttt{\ExplSyntaxOff}

(End definition for \xister@if@slash.)

\texttt{\xister@fraction}

\texttt{\protected\long\def\xister@fraction#1/#2\xister@stop}
\texttt{\{\}
\edef\xister@share@mul{\unexpanded{#1}}%
\edef\xister@share@div{\unexpanded{#2}}%
\xister@dbg@float{#1}%
\xister@dbg@div{#2}%
\}

(End definition for \xister@fraction.)

\texttt{\xister@build}

\texttt{\xister@build@h}
\texttt{\xister@build@v}
\texttt{\xister@build@horizontal}
\texttt{\xister@build@vertical}

First we set the length registers to the correct size, than we draw the glyphs batch wise. We have to flip the x-direction for the tenths and thousands, and the y-direction for the hundreds and thousands. The macro \xister@sgn is used to set the baseline in the correct direction. We locally disable\pgfsetcolor to allow building colour-unsafe boxes. This only works as long as luacolor (or something equivalent) isn't used, so consider this a crude hack.

\texttt{\protected\def\xister@build@vertical}
\texttt{\{\}
\let\xister@pgfsetcolor\pgfsetcolor
\let\pgfsetcolor\@gobble
\xister@setlengths
\def\xister@sgn{-}\%
\xister@drawzero@vertical
\xister@drawdigits@vertical{}{}{}%
\xister@X=-\xister@X
\xister@Y=-\xister@Y
\def\xister@sgn{+}\%
\xister@drawdigits@vertical{000}{-}{-}%
\xister@TMPA=\xister@Tdd
\xister@Tdd=\xister@Tdu
\xister@Tdu=\xister@TMPA
\xister@X=-\xister@X
\xister@drawdigits@vertical{00}{-}{-}%
\xister@X=-\xister@X
\xister@Y=-\xister@Y
\def\xister@sgn{-}\%
\xister@drawdigits@vertical{0}{-}{-}%
\let\pgfsetcolor\xister@pgfsetcolor
\}

\protected\def\xister@build@horizontal

1
This macro should store the different font features which should be kept track of. It will be redefined programmatically but the default is quite easy, just track the font size.

```latex
\long\def\xister@add@fontfeature#1{\begingroup\expandafter\expandafter\expandafter\endgroup\expandafter\ifx\csname xister@add@fontfeature@\detokenize{#1}\endcsname\relax\PackageError{xistercian}{Unsupported font feature \detokenize{#1}}{}\else\csname xister@add@fontfeature@\detokenize{#1}\endcsname\fi}

\def\xister@fontfeatures{\unexpanded\expandafter{\f@size}/}
```

We know that the colour hack breaks if luacolor is loaded, so if that is found to be the case we add the color feature to the list of things we need to track. But we only want to add it if it’s not yet in the list, so we run a check we borrow from expl3.

```latex
\ExplSyntaxOn
\AddToHook{begindocument/before}{\ifpackageloaded{luacolor}{\\exp_args:No \str_if_in:nnF \xister@fontfeatures { color } \\{ \xister@add@fontfeature{color} \} \}}
\ExplSyntaxOff
```
\ExplSyntaxOff

These macros all just add a specific macro to the list of things contained in \xister@fontfeatures, which determines what to consider deciding to redraw.

Now we define the keys

\def\xister@tmp#1#2#3{%\par
  \protected\long\ekvdef{xister}{#1}{#3}\par
  \ekvletkv{xister}{#2}{xister}{#1}\par
%}

\xister@tmp{wd}{width}{\edef\xister@w{\unexpanded{#1}}\xister@dbg{\dim{#1}}}\par
\xister@tmp{ht}{height}{\edef\xister@h{\unexpanded{#1}}\xister@dbg{\dim{#1}}}\par
\xister@tmp{sv}{stroke-v}{\edef\xister@tv{\unexpanded{#1}}\xister@dbg{\dim{#1}}}\par
\xister@tmp{sh}{stroke-h}{\edef\xister@th{\unexpanded{#1}}\xister@dbg{\dim{#1}}}\par
\xister@tmp{sdu}{stroke-du}{\edef\xister@tdu{\unexpanded{#1}}\xister@dbg{\dim{#1}}}\par
\xister@tmp{sdd}{stroke-dd}{\edef\xister@tdd{\unexpanded{#1}}\xister@dbg{\dim{#1}}}\par
\xister@tmp{bx}{bound-x}{\edef\xister@bx{\unexpanded{#1}}\xister@dbg{\float{#1}}}\par
\xister@tmp{by}{bound-y}{\edef\xister@by{\unexpanded{#1}}\xister@dbg{\float{#1}}}\par
\xister@tmp{bs}{baseline}{\edef\xister@bs{\unexpanded{#1}}\xister@dbg{\float{#1}}}\par
\xister@tmp{s}{strokes}{\edef\xister@tv{\unexpanded{#1}}\let\xister@th\xister@tv\let\xister@tdu\xister@tv\let\xister@tdd\xister@tv\xister@dbg{\dim{#1}}}\par

\}

(End definition for \xister@fontfeatures and \xister@add@fontfeature.)

(End definition for \xister@add@fontfeature@encoding and others.)
\let\xister@five@set\xister@undefined
\protected\ekvdefNoVal{xister}{horizontal}
  {%
    \let\xister@build\xister@build@horizontal
    \def\xister@x{.62\text{em}}
    \def\xister@y{1\text{ex}}
  }
\protected\ekvdefNoVal{xister}{vertical}
  {%
    \let\xister@build\xister@build@vertical
    \def\xister@x{.4\text{em}}
    \def\xister@y{1.55\text{ex}}
  }

The debug option redefines the debug macros to the correct definition.

\protected\ekvdefNoVal{xister}{debug}
  {%
    \xister@dbg@true
    \protected\long\def\xister@dbg@dim##1\%
      {%
        \typeout Package xistercian Debug: Trying to use dimension '{\unexpanded{##1}}'.
      }%
    \xister@TMPA=\dimexpr##1\relax
    \typeout{Package xistercian Debug: Done.}%
  }
\protected\long\def\xister@dbg@float##1\%
  {%
    \typeout Package xistercian Debug: Trying to use float '{\unexpanded{##1}}'.
    \xister@TMPA={##1}\z@
    \typeout{Package xistercian Debug: Done.}%
  }
\protected\long\def\xister@dbg@div##1\%
  {%
    \typeout Package xistercian Debug: Trying to use divisor '{\unexpanded{##1}}'.
    \xister@TMPA=\dimexpr\z@/#1\relax
    \typeout{Package xistercian Debug: Done.}%
  }

and parse the package options (and since the used names width, height, and debug are quite frequent we don't look at the global options).

\ekvoProcessLocalOptions{xister}

After the package was loaded we add the two redraw variants without a value.
\protected\ekvdefNoVal{xister}{r!}{\cistercianredraw}
\protected\ekvdefNoVal{xister}{r} {\cistercianredrawlazy}
\ekvletkvNoVal{xister}{redraw} {xister}{r}
\ekvletkvNoVal{xister}{redraw!}{xister}{r!}

The debug option is only available as a package option, we let it throw an error:
2.3 User macros

\cisterciansetup

Just as a simple way to pick options later on.

\protected\ekvsetdef\cisterciansetup{xister}

(End definition for \cisterciansetup. This function is documented on page 1.)

\cistercianstyle

We allow users to define a style they can call with a single key name inside \cisterciansetup. To achieve that we just define a NoVal key that’ll call a nested \cisterciansetup.

\protected\long\def\cistercianstyle#1#2\{
\ekvifdefinedNoVal{xister}{#1}\{
\PackageError{xistercian}{Key ‘#1’ already defined}{}
\}
\protected\ekvdefNoVal{xister}{#1}{\cisterciansetup{#2}}
\}

(End definition for \cistercianstyle. This function is documented on page 1.)

\cistercian\@cistercian

The macro is equivalent to other \LaTeX{} counter formatting macros, so we let it build the counter name and forward that to the canonically named auxiliary macro \@cistercian.

\newcommand\cistercian[1]{\expandafter\@cistercian\csname c@#1\endcsname}
The internal just gets the current value of a counter and forwards it.

\newcommand\@cistercian[1]{\expandafter\cisterciannum\expandafter{\the#1}}

(End definition for \cistercian and \@cistercian. These functions are documented on page 1.)

\cisterciannum\cisterciannum@pdf

\edef\cisterciannum
\{\unexpanded{\xister@texorpdf}\unexpanded{\expandafter{\csname cisterciannum \endcsname\cisterciannum@pdf}}\}

\protected\expandafter\def\csname cisterciannum \endcsname#1{\xister@a#1@}
\def\cisterciannum@pdf#1{CISTER#1}

(End definition for \cisterciannum and \cisterciannum@pdf. These functions are documented on page 1.)

\cisterciannumE

\newcommand\cisterciannumE[1]{\expandafter\cisterciannum\expandafter{\the\numexpr#1\relax}}

(End definition for \cisterciannumE. This function is documented on page 1.)

\cistercianredraw\cistercianredrawlazy

\protected\def\cistercianredraw
{\def\xister@last@font{\xister@last@font}\xister@ensure@current}
\protected\def\cistercianredrawlazy{\def\xister@last@font{\xister@last@font}}
2.4 Parsing

2.4.1 Small Auxiliaries

These are just small functions gobbling some tokens.

\def\xister@gobbletoat#1@{}
\def\xister@gobbletozero#10{}
\def\xister@gobblecopy0\rlap#1{}
\def\xister@done#1\xister@symbols@{}

(End definition for \xister@gobbletoat and others.)

\def\xister@count#1{\xister@gobbletoat#1\xister@count@done @+1\xister@count}
\def\xister@count@done @+1\xister@count{\relax @}

(End definition for \xister@count and \xister@count@done.)

\def\xister@texorpdf\let\xister@texorpdf@firstoftwo
\AddToHook{package/after/hyperref}{%\ifdef\texorpdfstring
\let\xister@texorpdf\texorpdfstring
\fi}
\AddToHook{begindocument/before}{%\ifdef\texorpdfstring
\let\xister@texorpdf\texorpdfstring
\fi}

(End definition for \xister@texorpdf.)

2.4.2 Input Parsing

First, we have to make sure that we’re in horizontal mode, else the \rlaps used to insert the glyphs will create havoc. We also need to ensure that the boxes contain the correct glyphs according to the current size and colour. After that we check whether we have to step over a minus sign and go to the next step.

\def\xister@a#1%{\leavevmode\xister@ensure@current\ifx#1-\expandafter\xister@b\else\expandafter\xister@b\expandafter#1\fi}

\def\xister@b#1%{\leavevmode\xister@ensure@current\ifx#1-%\expandafter\xister@c\else\expandafter\xister@c\expandafter#1\fi}

\def\xister@c#1%{\leavevmode\xister@ensure@current\ifx#1-%\expandafter\xister@\else\expandafter\xister\expandafter#1\fi}
Next we count the number of decimal digits in the number we want to print, take that
modulo 4 (that's what's $4*((#1-2)/4)$ is doing, thanks to Heiko Oberdiek and the code
in expl3), and use the correct symbol creating macro. The @@ serves as the end marker
since each step (except the first) will grab four digits.

\def\xister@b#1@% 319
  {\expandafter\xister@c\the\numexpr\xister@count#1@#1@@}
\def\xister@c#1@% 320
  {\csname xister@symbols@\romannumeral\numexpr#1-4*\numexpr#1-2\relax\endcsname}

(End definition for \xister@a, \xister@b, and \xister@c.)

\xister@symbols@
\xister@symbols@i
\xister@symbols@ii
\xister@symbols@iii

When this loop is done all arguments will be @, but we only check the first here. We
generate the symbols by overlaying up to five boxes, one for each decade plus the zero
stem. Each digit might be skipped if it is 0. Next grab the next four digits.

\def\xister@symbols@#1% 323
  {\protect\def\xister@symbols@##1##2##3##4% 324
    {\xister@gobbletoat##1\xister@done @% 325
      \xister@gobbletozero##1\xister@gobblecopy0% 326
      \rlap{\copy\csname xister@digitbox@##1000\endcsname}%% 327
      \xister@gobbletozero##2\xister@gobblecopy0% 328
      \rlap{\copy\csname xister@digitbox@##200\endcsname}%% 329
      \xister@gobbletozero##3\xister@gobblecopy0% 330
      \rlap{\copy\csname xister@digitbox@##30\endcsname}%% 331
      \xister@gobbletozero##4\xister@gobblecopy0% 332
      \rlap{\copy\csname xister@digitbox@##4\endcsname}%% 333
      \copy#1% 334
      \xister@symbols@}}% 335
\expandafter\xister@symbols@\csname xister@digitbox@0\endcsname

If there are less than four decimal digits inside the first Ci\textl{}erician numeral we provide
enough zeros to fill up the remainder.

\def\xister@symbols@i{\xister@symbols@0000} 341
\def\xister@symbols@ii{\xister@symbols@000} 342
\def\xister@symbols@iii{\xister@symbols@0} 343

(End definition for \xister@symbols@ and others.)

\section{Drawing the Digits}

\xister@clip@v
\xister@clip@h

Some of the digit glyphs need a bit of clipping to look nice. Let's define a small auxiliary
to select a clipping region.

\newcommand*=\xister@clip@v[1]
{\% 344
  \pgfpathrectanglecorners 345
  {\pgfpoint{dimexpr\ifdim\xister@X<\z@\fi\xister@Tv-relax}{#1}}% 346
  {\pgfpoint\xister@X\xister@Y}% 347
  \pgfusepath{clip}% 348
}
\newcommand*=\xister@clip@h[1]
{\% 349
The digit glyphs. Since each numeral is build from the zero stem and some appendix to it this only specifies the form of the appendices. The zero stem is defined a bit down the road. #1 will determine the decade, and #2 will be either – or empty.

\newcommand{xister@drawdigits@vertical}[3]{%
\xister@XY@share=\dimexpr\xister@share@mul\xister@Y\xister@share@div\relax
\xister@drawdigit{1#1}%
\xister@drawdigit{2#1}%
\xister@drawdigit{3#1}%
\xister@drawdigit{4#1}%
\xister@drawdigit{5#1}%
}%

\newcommand{xister@drawdigits@horizontal}{%
\xister@drawdigit{1}1
\xister@drawdigit{2}1
\xister@drawdigit{3}1
\xister@drawdigit{4}1
\xister@drawdigit{5}1
}%

(End definition for \xister@clip@v and \xister@clip@h.)
Each digit has the same bounding box and baseline which is specified here. #1 will be
the name, #2 the digit specific path.
\newcommand\xister@drawdigit[2]{%
  \expandafter\setbox\csname xister@digitbox@#1\endcsname=\hbox
  \begin{pgfpicture}
    \pgfpathrectanglecorners
    {\pgfpoint{-\xister@bx\xister@X}{-\xister@by\xister@Y}}{\pgfpoint{\xister@bx\xister@X}{\xister@by\xister@Y}}
    \pgfusepath{use as bounding box}
    \pgfusepath{stroke}
    \pgfsetbaseline{\xister@sgn\xister@bs\xister@Y}
  \end{pgfpicture}
}

(End definition for \xister@drawdigits@vertical and \xister@drawdigits@horizontal.)
\xister@drawzero@horizontal
\xister@drawzero@vertical
\newcommand*\xister@drawzero@vertical
Some digits can be build from a few of the other ones. This is faster than drawing every-thing again, and this macro eases the process a bit.

```
\newcommand*{\xister@superimpose}[2] {\expandafter\setbox\csname xister@digitbox@#1\endcsname=\hbox{#2}}
```

(End definition for \xister@superimpose.)

### 2.6 Rebuild the Glyphs

Lengths are set depending on the current font size and the sizes of \xister@font. To keep the font changes local they are kept inside a group and a chain of \expandafter\s is used to evaluate the user specified sizes while the \xister@font is still active.

```
\protected\def\xister@setlengths {
\begingroup
\xister@font
\edef\xister@tmp {
% 
\endgroup
\xister@X=\the\dimexpr.5\dimexpr\xister@x\relax\relax
\xister@Y=\the\dimexpr.5\dimexpr\xister@y\relax\relax
\xister@Th=\the\dimexpr.5\dimexpr\xister@th\relax\relax
\xister@Tv=\the\dimexpr.5\dimexpr\xister@tv\relax\relax
\xister@Tdu=\the\dimexpr.5\dimexpr\xister@tdu\relax\relax
\xister@Tdd=\the\dimexpr.5\dimexpr\xister@tdd\relax\relax
\xister@tmp}
```

(End definition for \xister@setlengths.)

The glyphs have to be rebuilt if the font size or colour changed. The latter is an annoy-ance because we can build boxes in a way that they change colour to the surrounding colour, but PGF is too intelligent here and does issue the corresponding \specials to
save the current colour for the box. Because of that we test which \texttt{\textbackslash f@size} and colour (using \texttt{\textbackslash color@.}) was active when we last built the glyphs. If this matches we do nothing, else we rebuild the glyphs.

The \texttt{\edef} is just used to remove the debug code if it isn’t necessary.

\begin{verbatim}
\protected\edef\xister@ensure@current
{%
\unexpanded{\edef\xister@tmp{\xister@fontfeatures}}%
\unexpanded{\unless\ifx\xister@tmp\xister@last@font}%
\ifx\xister@tmp\xister@last@font%
\typeout{%
  Package xistercian Debug: Font changed. Building glyphs for \xister@tmp.
}%
\fi
\unexpanded{%
\let\xister@last@font\xister@tmp
\xister@build
}%
\if\xister@tmp\xister@last@font%
\unexpanded{\typeout{Package xistercian Debug: Done.}}%
\fi
\unexpanded{\fi}%
\AtBeginDocument{\xister@ensure@current}
(End definition for \xister@ensure@current.)
\end{verbatim}
Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

A
\AddToHook ........................................ 113, 300, 306
\AtBeginDocument .................................... 616

B
\begin ................................................. 545
\begingroup ........................................... 102, 169, 578

C
\cistercian .............................................. 275
\cisterciannum ........................................ 276, 277, 286
\cisterciannumE ....................................... 285
\cistercianredraw ...................................... 262, 287
\cistercianredrawlazy .................................. 263, 287
\cisterciansetup ....................................... 268, 273
\cistercianstyle ........................................ 269
\copy ..................................................... 329, 331, 333, 335, 336, 435, 436, 440, 441, 445, 446, 447, 526, 527, 531, 532, 536, 537, 538
\csname ................................................. 42
\cs_new_protected:Npn .................................. 42
\def ...................................................... 44, 51, 56, 61, 70, 74, 79, 84, 93, 100, 111, 123, 125, 142, 224, 225, 230, 231, 236, 246, 253, 269, 283, 284, 287, 289, 292, 293, 294, 295, 296, 297, 298, 312, 319, 321, 323, 325, 341, 342, 343, 576
\detokenize ............................................. 104, 106, 108

D
\edef ..................................................... 46, 48, 127, 147, 148, 149, 150, 152, 154, 155, 156, 157, 160, 166, 183, 196, 203, 277, 580, 592, 594
\ekvcsvloop ........................................... 188, 189
\ekvdef ............................................... 144, 192, 199

\ekvdefNoVal .......................................... 206, 207, 212, 221, 227, 233, 262, 263, 266, 273
\ekvfdefinedNoVal .................................... 271
\ekvfdefinedset ...................................... 140
\ekvlet ............................................... 218, 219
\ekvletkv ............................................. 145, 190, 191
\ekvletkvNoVal ....................................... 264, 265
\ekoProcessLocalOptions .............................. 261
\evksetdef .......................................... 217, 268
\else .................................................. 107, 173, 317, 408, 414, 499, 505
\end ................................................... 533
\endgroup ............................................. 102, 169, 582
\endinput ............................................. 141
\exp commands:
\exp_args:No .......................................... 117
\expandafter ......................................... 31, 32, 33, 34, 37, 102, 103, 111, 125, 129, 130, 134, 169, 170, 174, 275, 276, 280, 283, 286, 316, 317, 320, 340, 543, 575
\ExplSyntaxOff ...................................... 36, 43, 122
\ExplSyntaxOn ........................................ 28, 41, 112
\fi ..................................................... 109, 175, 304, 310, 317, 347, 354, 424, 515, 605, 613, 614
\hbox .................................................. 543, 575
\ifdefined ........................................... 302, 308
\ifdim ............................................... 347, 354
\ifx .................................................. 103, 170, 316, 595
\int commands:
\int_step_inline:nn .................................. 29
\leavevmode .......................................... 314
\let ................................................... 38, 39, 40, 53, 54, 72, 76, 77, 95, 97, 98, 99, 161, 162, 163, 174, 182, 188, 220, 223, 229, 299, 303, 309, 608
\long .................................................. 44, 100, 144, 192, 199, 236, 246, 253, 269