# The xišercian package

Jonathan P. Spratte∗

2021-12-20 V1.2

## Contents

<table>
<thead>
<tr>
<th>1</th>
<th>Documentation</th>
<th>1</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.1</td>
<td>Introduction</td>
<td>1</td>
</tr>
<tr>
<td>1.2</td>
<td>Macros</td>
<td>1</td>
</tr>
<tr>
<td>1.3</td>
<td>Options</td>
<td>1</td>
</tr>
<tr>
<td>1.4</td>
<td>Bold Cišercian Numerals</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>Implementation</td>
<td>1</td>
</tr>
<tr>
<td>2.1</td>
<td>Variables</td>
<td>1</td>
</tr>
<tr>
<td>2.2</td>
<td>Options</td>
<td>1</td>
</tr>
<tr>
<td>2.3</td>
<td>User macros</td>
<td>1</td>
</tr>
<tr>
<td>2.4</td>
<td>Parsing</td>
<td>1</td>
</tr>
<tr>
<td>2.4.1</td>
<td>Small Auxiliaries</td>
<td>1</td>
</tr>
<tr>
<td>2.4.2</td>
<td>Input Parsing</td>
<td>1</td>
</tr>
<tr>
<td>2.5</td>
<td>Drawing the Digits</td>
<td>1</td>
</tr>
<tr>
<td>2.6</td>
<td>Rebuild the Glyphs</td>
<td>1</td>
</tr>
</tbody>
</table>

## Index

1

## 1 Documentation

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 The name is chosen to be xišercian instead of cišercian because I’ve learned that a (currently) unpublished package by that name already exists and I don’t want to be an evil name-stealer.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

Ciślęrcian numerals are a system to denote the numbers from 1 to 9999 with a single glyph. They use a slēm (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 (†), upper left tens (ฑ), lower right hundreds (✞), and lower left thousands (‡). 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 Ciślęrcian numerals horizontally instead. This package also supports that, in which case the numerals are rotated by 90 degrees counterclockwise, see figure 2.

Moreover some glyphs have a different shape in some manuscripts. The shapes in figures 1 and 2 are the most used ones nowadays (most likely because those are the ones Wikipedia shows vector graphics of). This package offers some (not all) alternative forms. Most notably some historic documents have swapped shapes of digits. To not further the confusion this package doesn't support swapping digit shapes.

All glyphs have the same bounding box, are drawn using the pgf package, and cached inside \TeXX 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 Ciślęrcian 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 Ciślęrcian numerals as page numbers you can use:

\pagenumbering{cistercian}
or to use them as footnotes:
\renewcommand{\footnote}{\cistercian{footnotes}}

The package tries to play nice on hyperref if it is loaded, but bookmarks containing material that should've been a Cistercian numeral will have a leading CISTER in front of the numeric value to give unique strings compared to \arabic (to my knowledge there are no Unicode points for Cistercian numerals).

Negative numbers just keep a leading -, and numbers with an absolute number greater than 9999 are displayed with multiple Cistercian digits. For instance −12345678 is displayed as \text{-}F31. And in the case that four consecutive digits are zeros they get displayed as that: \text{-}10000.

Full disclaimer: I’m neither a historian nor have I any authority on the matter. I’m just a bloke who was fascinated by this numeral system and wanted to provide \LaTeX{} support for it. Most of my knowledge about this system from Wikipedia and similar sources.

### 1.2 Macros

\cistercian{⟨counter⟩}

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

\cisterciannum{⟨integer⟩}

Prints the ⟨integer⟩ as a Cistercian numeral. The ⟨integer⟩ has to be provided as a string containing only digits and optionally a single leading - (spaces are ignored). This can handle integers of arbitrary size.

\cisterciannumE{⟨integer expression⟩}

Evaluates the ⟨integer expression⟩ and prints the result as a Cistercian numeral.

\cisterciansetup{⟨key=value,...⟩}

Can be used to locally change the options after the package was loaded. The glyphs of the Cistercian 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 Cistercian numeral in the current scope will redraw the glyphs. The lazy variant could result in the glyphs being redrawn multiple times (if Cistercian numerals are used in different nested groups), while the normal variant might result in the glyphs being redrawn too eagerly.

\cistercianstyle{⟨name⟩}{⟨key=value,...⟩}

With this macro you can define a key called ⟨name⟩ that’ll set the options in the ⟨key=value⟩ list if used inside \cisterciansetup. Only new names are allowed. If your style shouldn’t take an argument (so is fixed) use the normal variant. If your style needs to take an argument use the variant with ⟨*⟩.
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.

For example, the key horizontal is equivalently defined to and can be used as

\cistercianstyle{horizontal}{o=h,wd=0.775em,ht=1.13ex}
\cisterciansetup{horizontal}

And if you want to define a style that at the same time changes the stroke widths for vertical and horizontal strokes you could use

\cistercianstyle*{sh+v}{sh={#1},sv={#1}}
\cisterciansetup{sh+v=.12ex}

1.3 Options

The options described here are all usable as package options (except when explicitly stated otherwise). \textit{xistlercian} 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 \texttt{width} and \texttt{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. A \texttt{(dimension expression)} is evaluated (using the setup specified in \texttt{font}) every time the font changed according to the \texttt{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.

<table>
<thead>
<tr>
<th>width</th>
<th>\texttt{width = {\textit{dimension expression}}}</th>
<th>.5em</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{wd}</td>
<td>Set the width of the Ci\textit{st}lercian numeral digits.</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>height</th>
<th>\texttt{height = {\textit{dimension expression}}}</th>
<th>1.55ex</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{ht}</td>
<td>Set the height of the Ci\textit{st}lercian numeral digits.</td>
<td></td>
</tr>
</tbody>
</table>
bound-x
  \( \text{bound-x} = \{\text{float}\} \)
  Specifies the factor the bounding box in \( x \)-direction is bigger than the width. An empty \( \text{float} \) is considered the same as 1.

bound-y
  \( \text{bound-y} = \{\text{float}\} \)
  empty
  Specifies the factor the bounding box in \( y \)-direction is bigger than the height. An empty \( \text{float} \) is considered the same as 1.

baseline
  \( \text{baseline} = \{\text{float}\} \)
  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.

stroke-v
  \( \text{stroke-v} = \{\text{dimension expression}\} \)
  .18ex
  Sets the stroke thickness of vertical strokes.

stroke-h
  \( \text{stroke-h} = \{\text{dimension expression}\} \)
  .1ex
  Sets the stroke thickness of horizontal strokes.

stroke-du
  \( \text{stroke-du} = \{\text{dimension expression}\} \)
  .1ex
  Sets the stroke thickness of diagonal strokes going from lower left to upper right.

stroke-dd
  \( \text{stroke-dd} = \{\text{dimension expression}\} \)
  .18ex
  Sets the stroke thickness of diagonal strokes going from upper left to lower right.

stroke-s
  \( \text{stroke-s} = \{\text{dimension expression}\} \)
  empty
  Sets the stroke thickness of the zero stem. If this is set to an empty value the stroke thickness in vertical mode is that of stroke-v and in horizontal mode that of stroke-h.

strokes
  \( \text{strokes} = \{\text{dimension expression}\} \)
  Sets all stroke thicknesses at once.

fraction
  \( \text{fraction} = \{\text{float}\} \)
  \( \text{fraction} = \{\text{float}\}/\{\text{integer}\} \)
  The Cistercian digits are drawn only in a fraction of the total bounding box of each symbol. The ones are drawn (with vertical Cistercians) in the upper right rectangle. The width of that rectangle is determined by width, and the upper border by 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 \( \text{float} \), or just pass in a divisor as an \( \text{integer} \) (with a leading slash), or both. An empty \( \text{float} \) equals 1.

Consider the following example (result on the right):

\begin{verbatim}
{\cisterciansetup{fr=2/9,redraw}\cisterciannum{9}}\par
{\cisterciansetup{fr=0.6,redraw}\cisterciannum{9}}\par
\end{verbatim}
Figure 4: Alternate forms of the digit 5. The length of the dash=3 variant is too long in the normal setup of this document, as the difference between \(5\) and \(\overline{5}\) is hard to notice. The tight bounding boxes using bx=1, by=1 are drawn red.

<table>
<thead>
<tr>
<th>orientation</th>
<th>orientation = {(choice)}</th>
<th>vertical</th>
</tr>
</thead>
<tbody>
<tr>
<td>horizontal</td>
<td>horizontal</td>
<td></td>
</tr>
<tr>
<td>vertical</td>
<td>vertical</td>
<td></td>
</tr>
<tr>
<td>alternate-5</td>
<td>alternate-5 = {(choice)}</td>
<td>triangle</td>
</tr>
</tbody>
</table>

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. Please ensure that your readers can’t confuse the results with the shape of 6.

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

The usage might look like any of the following:

\[\text{\textbackslash cisterciansetup}\]
\{
  5=triangle, 5={triangle}, 5=dot, 5={dot}, 5=dot=.1ex, 5={dot=.1ex}, 5=dash, 5={dash}, 5=dash=2, 5={dash=2}
\}
There are alternative forms of the digit 6. While today the usual form is stroke, there are also historic documents showing the 6 as two dots, which you can achieve using the dot choice, and to support squares or short dashes instead of round dots you can use the dash choice.

For dot you can also specify the radius as a dimension. The default is .09ex. And for dash you can also specify the length in multiples of the used stroke thickness, an empty value is considered the same as 1. The default is empty. The stroke has always the full length and doesn’t accept an additional specification.

A comparison of the alternate forms is shown in figure 5.

Another digit having alternative forms is the 9. For this two (choice)s are available, rectangle and triangle. Neither of the two accept additional specifications.

A comparison of the alternate forms is shown in figure 5.

Set the font which xislercian uses locally to evaluate the dimensions given to width, height, and stroke (only font switches work here, you can’t use stuff like \texttt, instead use \ttfamily).
redraw = \{\langle feature-list \rangle\}
redraw += \{\langle feature-list \rangle\}
redraw!

Selects 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 \langle feature-list \rangle are encoding, family, series, shape, size, and color.

If you see that a Cistercian numeral doesn't have the correct colour of the surrounding text you need to add color to the font features (xi\ltercian uses some hack to get the boxed glyphs to respect the surrounding colour, but that hack might fail in certain circumstances).

If you use the += variant of this option the \langle feature-list \rangle 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.

debug

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 Cistercian 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 bfseries and back to mdseries:

\AddToHook{bfseries}
{%
  \cisterciansetup{sv=.27ex,sdd=.27ex,sh=.15ex,sdu=.15ex,wd=.625em}%
  \cistercianredrawlazy
%
\AddToHook{mdseries}
{%
  \cisterciansetup{sv=.18ex,sdd=.18ex,sh=.1ex,sdu=.1ex,wd=.5em}%
  \cistercianredrawlazy
%
This results in \cisterciannum{2} different \enquote{series}.
\par
bfseries
This results in \cisterciannum{2} different \enquote{series}.

  This results in \dagger different “series”.
This results in \dagger different “series”.

Of course, instead of doing \cistercianredrawlazy in the above code, we could as well do the following equivalent thing and let xi\ltercian determine when to redraw:
\AddToHook{bfseries}
\cisterciansetup{sv=.27ex,sdd=.27ex,sh=.15ex,sdu=.15ex,wd=.625em}}
\AddToHook{mdseries}
\cisterciansetup{sv=.18ex,sdd=.18ex,sh=.1ex,sdu=.1ex,wd=.5em}}
\cisterciansetup{redraw+=series}
2 Implementation

First we say who we are and load the required packages

\ProvidesPackage{xistercian}[2021-12-20 v1.2 Cistercian numerals in LaTeX]
\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@ts}{}
\newcommand*{\xister@x}{.5em}
\newcommand*{\xister@y}{1.55ex}
\newcommand*{\xister@bx}{1.2}
\newcommand*{\xister@by}{}
\newcommand*{\xister@bf}{}
\newcommand*{\xister@font}{\normalfont}
\newcommand*{\xister@five@dot}{.09ex}
\newcommand*{\xister@five@dash}{}
\newcommand*{\xister@six@dot}{.09ex}
\newcommand*{\xister@six@dash}{}
\newcommand*{\xister@share@div}{/6}
\newcommand*{\xister@share@mul}{}
\newif{\xister@dbg@}
\newif{\xister@five@triangle@}
\xister@five@triangle@true
\newif{\xister@five@dot@}
\newdimen{\xister@X}
\newdimen{\xister@Y}
\newdimen{\xister@Th}
\newdimen{\xister@Tv}
\newdimen{\xister@tdu}
\newdimen{\xister@tdd}
\newdimen{\xister@Ts}
\newdimen{\xister@TMPA}
\newdimen{\xister@TMPB}
\newif{\xister@nine@triangle@}
\newdimen{\xister@Th}
\newdimen{\xister@Tv}
\newdimen{\xister@tdu}
\newdimen{\xister@tdd}
\newdimen{\xister@Ts}
\newif{\xister@five@triangle@}
\xister@five@triangle@true
\newif{\xister@five@dot@}
\newif{\xister@six@dot@}
\newif{\xister@six@stroke@}
\xister@six@stroke@true
\newdimen{\xister@X}
\newdimen{\xister@Y}
\newdimen{\xister@Th}
\newdimen{\xister@Tv}
\newdimen{\xister@tdu}
\newdimen{\xister@tdd}
\newdimen{\xister@Ts}
\newdimen{\xister@TMPA}
\newdimen{\xister@TMPB}
\newdimen{\xister@Five@dot}
\newdimen{\xister@Six@dot}

We also need a bunch of box registers (37 of them to be precise). For the loop we borrow an \ExplSyntaxOn function.

\ExplSyntaxOn
\int_step_inline:nn {9} \\
{ \expandafter\newsavebox\csname xister@digitbox@#1\endcsname \\
\expandafter\newsavebox\csname xister@digitbox@#10\endcsname }

\ExplSyntaxOff

These are initially just \gobble, but the debug option might change things.

\let\xister@dbg@dim\@gobble
\let\xister@dbg@div\@gobble
\let\xister@dbg@float\@gobble

(End definition for \xister@dbg@dim and \xister@dbg@float.)

2.2 Options

Some macros are required to parse options.

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

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

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

(End definition for \xister@fraction.)

\xister@build First we set the length registers to the correct size, then 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.
\protected\def\xister@build@vertical
{%
  \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@build@horizontal
\xister@build@v
\xister@build@vertical
\xister@drawdigits@vertical{000}{-}{-}\
\xister@Tdd=\xister@Tdu\
\xister@Tdu=\xister@TMPA\
\xister@X=-\xister@X\
\xister@drawdigits@vertical{00}{-}{}\
\xister@Y=-\xister@Y\
\def\xister@sgn{-}\
\xister@drawdigits@vertical{0}{-}{}\
\let\pgfsetcolor\xister@pgfsetcolor\
}
\protected\def\xister@build@horizontal\
{\
\let\xister@pgfsetcolor\pgfsetcolor\
\let\pgfsetcolor\@gobble\
\xister@setlengths\
\def\xister@sgn{-}\
\xister@drawzero@horizontal\
\xister@drawdigits@horizontal{}{}{}\
\xister@X=-\xister@X\
\xister@Y=-\xister@Y\
\def\xister@sgn{+}\
\xister@drawdigits@horizontal{000}{-}{-}\
\xister@TMPA=\xister@Tdd\
\xister@Tdd=\xister@Tdu\
\xister@Tdu=\xister@TMPA\
\xister@X=-\xister@X\
\xister@drawdigits@horizontal{0}{-}{}\
\xister@Y=-\xister@Y\
\def\xister@sgn{-}\
\xister@drawdigits@horizontal{00}{-}{}\
\let\pgfsetcolor\xister@pgfsetcolor\
}
\let\xister@build\xister@build@vertical\
\let\xister@build@v\xister@build@vertical\
\let\xister@build@h\xister@build@horizontal
(End definition for \xister@build and others.)

\xister@fontfeatures\xister@add@fontfeature
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.
\protected\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.

\ExplSyntaxOn
\AddToHook{begindocument/before}{
    \exp_args:No \str_if_in:nFN \xister@fontfeatures { color }
    \xister@add@fontfeature{color}
}
\ExplSyntaxOff

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

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.

\def\xister@tmp#1#2#3{
    \protected\long\ekvdef{xister}{#1}{#3}
    \ekvletkv{xister}{#2}{xister}{#1}
}
\xister@tmp{wd}{width}{\edef\xister@x{\unexpanded{#1}}\xister@dbg@dim{#1}}
\xister@tmp{ht}{height}{\edef\xister@y{\unexpanded{#1}}\xister@dbg@dim{#1}}
\xister@tmp{bx}{bound-x}{\edef\xister@bx{\unexpanded{#1}}\xister@dbg@float{#1}}
\xister@tmp{by}{bound-y}{\edef\xister@by{\unexpanded{#1}}\xister@dbg@float{#1}}
\xister@tmp{bs}{baseline}{\edef\xister@bs{\unexpanded{#1}}\xister@dbg@float{#1}}
\xister@tmp{sv}{stroke-v}{\edef\xister@tv{\unexpanded{#1}}\xister@dbg@dim{#1}}
\xister@tmp{sh}{stroke-h}{\edef\xister@th{\unexpanded{#1}}\xister@dbg@dim{#1}}
\xister@tmp{sd}{stroke-du}

\end{document}

\let\xister@build\xister@build@horizontal
\def\xister@x{.775em}\%
\def\xister@y{1.13ex}\%
}\protected\ekvdefNoVal{xister}{vertical}{%
\let\xister@build\xister@build@vertical
\def\xister@x{.5em}\%
\def\xister@y{1.55ex}\%
}\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.}\%}
}\}

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:
\ekvdefNoVal{xister}{debug}{%
\PackageError{xister}{‘debug’ is only available as a load time option};}}
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
\xister@style

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, or a normal key if their style needs to take a value.

\protected\def\cistercianstyle
{%
\@ifstar{
xister@style\ekvifdefined{\long\ekvdef}\
  \xister@style\ekvifdefinedNoVal\ekvdefNoVal%
}
%
\protected\long\def\xister@style#1#2#3#4%
{%
  #1{xister}{#3}%
  {\PackageError{xistercian}{Key ‘#3’ already defined}{}}%
  {\protected\long\def\xister@style{#2}{#3}{\cisterciansetup{#4}}%}
%
}(End definition for \cistercianstyle and \xister@style. These functions are 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\cisterciannum@pdf\#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
}\protect\def\cistercianredrawlazy{\def\xister@last@font\xister@last@font}

(End definition for \cistercianredraw and \cistercianredrawlazy. These functions are documented on page 1.)

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.)

\xister@texorpdf
\let\xister@texorpdf@firstoftwo
\AddToHook{package/hyperref/after}{%
  \ifdefined\texorpdfstring
    \def\xister@texorpdf{\texorpdfstring}
  \fi
}\AddToHook{begindocument/before}{%
  \ifdefined\texorpdfstring
    \def\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@d
    \else\expandafter\xister@d\expandafter#1\fi
\}
Next we check whether the remaining number is empty. If that is the case we use \texttt{\numexpr\relax} to throw a missing number error and produce just a zero \texttt{st}em. Else we do a dry run over the input number to see how many leading zeros we must pad to get blocks of four. The two blocks of \texttt{@}s serve as end markers since each \texttt{step} of both \texttt{xister@c} grabs eight and each of \texttt{xister@symbols} will grab four digits. The \texttt{xister@c} is used to put enough zeros in front of our number such that it consists of blocks of four digits.

```latex
\def\xister@b#1{%\relax
  \if\relax\detokenize{#1}\relax
    \expandafter\xister@gobbletozero\the\numexpr\relax
    \\expandafter\@gobbletwo
    \@firstofone{\xister@c#1@@@@@@@@\relax#1@@@@}%
  \fi
  %\def\xister@c#1#2#3#4#5#6#7#8% {%%
    \xister@gobbletoat #1\xister@d{}% 
    #2\xister@d{000}% 
    #3\xister@d{00}% 
    #4\xister@d0% 
    #5\xister@d{}% 
    #6\xister@d{000}% 
    #7\xister@d{00}% 
    #8\xister@d0% 
    \@xister@c
  }
  \def\xister@c#1#2#3#4#5#6#7#8% {%%
    \xister@gobbletozero\xister@c#1#2#3#4#5#6#7#8% }
  %
  \def\xister@b#1#2\xister@c#3\relax{\xister@symbols#1}
  (End definition for \xister@a and others.)
}
\edef\xister@b#1#2#3#4#5#6#7#8\xister@c#1#2#3#4#5#6#7#8\xister@c#1#2#3#4\relax{\xister@symbols#1}

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

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

\begin{verbatim}
\newcommand\xister@clip@v[1]{%
  \pgfpathrectanglecorners
  {\pgfpoint{\dimexpr\ifdim\xister@X<\z@-\fi\xister@Ts\relax}{#1}}%
  {\pgfpoint{\xister@X}{\xister@Y}}%
  \pgfusepath{clip}%
}\newcommand\xister@clip@h[1]{%
  \pgfpathrectanglecorners
  {\pgfpoint{#1}{\dimexpr\ifdim\xister@Y<\z@-\fi\xister@Ts\relax}}%
  {\pgfpoint{-\xister@X}{\xister@Y}}%
  \pgfusepath{clip}%
}\end{verbatim}

(End definition for \xister@clip@v and \xister@clip@h.)

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.

\begin{verbatim}
\newcommand\xister@drawdigits@vertical[3]{%
  \xister@XY@share=\dimexpr\xister@share@mul\xister@Y\xister@share@div\relax
  \xister@drawdigit{1#1}{%\
    \pgfsetlinewidth{2\xister@Th}\
    \xister@TMPA=\dimexpr\xister@Y\xister@sgn\xister@Th\relax
    \pgfpathmoveto{\pgfpoint{\z@}{\xister@TMPA}}%\
    \pgfpathlineto{\pgfpoint{\xister@X}{\xister@TMPA}}%
  }%
  \xister@drawdigit{2#1}{%\
    \pgfsetlinewidth{2\xister@Th}\
    \xister@TMPA=\dimexpr\xister@XY@share+\#2\xister@Th\relax
    \pgfpathmoveto{\pgfpoint{\z@}{\xister@TMPA}}%\
    \pgfpathlineto{\pgfpoint{\xister@X}{\xister@TMPA}}%
  }%
  \xister@drawdigit{3#1}{%\
    \pgfsetlinewidth{2\xister@Th}\
    \xister@TMPA=\dimexpr\xister@XY@share+\#2\xister@Th\relax
    \pgfpathmoveto{\pgfpoint{\z@}{\xister@TMPA}}%\
    \pgfpathlineto{\pgfpoint{\xister@X}{\xister@TMPA}}%
  }%
  \xister@drawdigit{4#1}{%\
    \pgfsetlinewidth{2\xister@Th}\
    \xister@TMPA=\dimexpr\xister@XY@share+\#2\xister@Th\relax
    \pgfpathmoveto{\pgfpoint{\z@}{\xister@TMPA}}%\
    \pgfpathlineto{\pgfpoint{\xister@X}{\xister@XY@share}}%
  }%
  \xister@drawdigit{5#1}{%\
    \pgfsetlinewidth{2\xister@Th}\
    \xister@TMPA=\dimexpr\xister@XY@share+\#2\xister@Th\relax
    \pgfpathmoveto{\pgfpoint{\z@}{\xister@TMPA}}%\
    \pgfpathlineto{\pgfpoint{\xister@X}{\xister@XY@share}}%
  }%
  \xister@drawdigit{6#1}{%\
    \pgfsetlinewidth{2\xister@Th}\
    \xister@TMPA=\dimexpr\xister@XY@share+\#2\xister@Th\relax
    \pgfpathmoveto{\pgfpoint{\z@}{\xister@TMPA}}%\
    \pgfpathlineto{\pgfpoint{\xister@X}{\xister@XY@share}}%
  }%
  \xister@drawdigit{7#1}{%\
    \pgfsetlinewidth{2\xister@Th}\
    \xister@TMPA=\dimexpr\xister@XY@share+\#2\xister@Th\relax
    \pgfpathmoveto{\pgfpoint{\z@}{\xister@TMPA}}%\
    \pgfpathlineto{\pgfpoint{\xister@X}{\xister@XY@share}}%
  }%
  \xister@drawdigit{8#1}{%\
    \pgfsetlinewidth{2\xister@Th}\
    \xister@TMPA=\dimexpr\xister@XY@share+\#2\xister@Th\relax
    \pgfpathmoveto{\pgfpoint{\z@}{\xister@TMPA}}%\
    \pgfpathlineto{\pgfpoint{\xister@X}{\xister@XY@share}}%
  }%
  \xister@drawdigit{9#1}{%\
    \pgfsetlinewidth{2\xister@Th}\
    \xister@TMPA=\dimexpr\xister@XY@share+\#2\xister@Th\relax
    \pgfpathmoveto{\pgfpoint{\z@}{\xister@TMPA}}%\
    \pgfpathlineto{\pgfpoint{\xister@X}{\xister@XY@share}}%
  }%
}
\end{verbatim}
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}% #2 
\pgfusepath{stroke}% 
\pgfsavenormal %}
Some digits can be build from a few of the other ones. This is faster than drawing everything 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 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@Five@dot=\the\dimexpr\xister@five@dot\relax
\xister@Six@dot=\the\dimexpr\xister@six@dot\relax
\ifx\xister@ts\@empty
\xister@Ts=\xister@Th\else\xister@Tv\fi

(End definition for \xister@setlengths.)

\xister@superimpose Some digits can be build from a few of the other ones. This is faster than drawing everything 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 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@Five@dot=\the\dimexpr\xister@five@dot\relax
\xister@Six@dot=\the\dimexpr\xister@six@dot\relax
\ifx\xister@ts\@empty
\xister@Ts=\xister@Th\else\xister@Tv\fi

(End definition for \xister@setlengths.)
The glyphs have to be rebuilt if the font size or colour changed. The latter is an annoyance 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 \f@size and colour (using \color@. ) was active when we last built the glyphs. If this matches we do nothing, else we rebuild the glyphs.

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

\protected\edef\xister@ensure@current
{%
\unexpanded\edef\xister@tmp{\xister@fontfeatures}\
\unexpanded\unless\ifx\xister@tmp\xister@last@font\xister@dbg@
\unexpanded
{%
\typeout
{{Package xistercian Debug: Font changed. Building glyphs for \xister@tmp.}}
%
}\fi
\unexpanded
{%
\let\xister@last@font\xister@tmp
\xister@build
\xister@dbg@
\unexpanded{\typeout{Package xistercian Debug: Done.}}%
\fi
\unexpanded{\fi}%
}
\AtBeginDocument{\xister@ensure@current}

(End definition for \xister@ensure@current.)
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 .......................... 123, 354, 360
\AtBeginDocument .......................... 790

B
\begin ................................ 711
\begingroup ........................................ 112, 185, 744
\endgroup ........................................ 151

C
\cistercian ........................................ 331
\cisterciannum ....................................... 332, 333, 342
\cisterciannumE ..................................... 341
\cistercianredraw ................................... 312, 343
\cistercianredrawlazy .............................. 313, 343
\cisterciansetup .................................. 318, 329
\cistercianstyle ................................... 319
\copy .................................................. 402, 404, 406, 408, 409, 505, 506, 510, 511, 533, 534, 535, 644, 645, 649, 650, 672, 673, 674
\csnewprotected:Npn .................................. 52
\curname .............................................. 41, 42, 43, 44, 47, 114, 118, 135, 144, 186, 190, 331, 337, 339, 402, 404, 406, 408, 413, 505, 506, 510, 511, 533, 534, 535, 644, 645, 649, 650, 672, 673, 674, 709, 741
\detokenize .......................................... 114, 116, 118, 375
\ekvcsvloop ........................................ 204, 205
\ekvdef ............................................. 154, 208, 215, 237, 244, 322
\ekvdefNoVal ........................................ 222, 223, 228, 251, 252, 257, 266, 267, 271, 277, 283, 312, 313, 316, 323
\ekvifdefined ....................................... 322
\ekvifdefinedNoVal .................................. 323
\ekvifdefinedset .................................... 150
\ekvlet ............................................. 234, 235, 263, 266, 269, 270
\ekvletkv ............................................ 155, 206, 207
\ekvletkvNoVal ...................................... 314, 315
\ekvOProcessLocalOptions .......................... 311
\ekvsetdef .......................................... 233, 262, 268, 318
\else ................................................. 117, 189, 371, 478, 494
\end .................................................. 530, 552, 617, 623, 669, 691, 759, 760
\endgroup ............................................ 719
\endcsname ......................................... 118, 135, 144, 186, 190, 331, 337, 339, 402, 404, 406, 408, 413, 505, 506, 510, 511, 533, 534, 535, 644, 645, 649, 650, 672, 673, 674, 709, 741
\endinput ............................................. 151
\exp commands:
\exp_args:No ........................................ 127
\expafter ............................................ 41, 42, 43, 44
\explSyntaxOff ...................................... 46, 53, 132
\explSyntaxOn ....................................... 38, 51, 122
\fi ..................................................... 119, 171, 191, 358, 364, 371, 379, 417, 424, 494, 537, 564, 566, 633, 676, 703, 705, 759, 762, 779, 787, 788
\hbox .................................................. 709, 741
\if .................................................... 375
\ifdefified .......................................... 356, 362
\ifdim ................................................. 417, 424
\ifx ................................................. 113, 171, 186, 370, 757, 759, 769
\int commands:
\int_step_inline:nn .................................. 39