# zhnumber 宏包

李清
sobenlee@gmail.com

2012/05/21 v1.4

## 1 简介

zhnumber 宏包用于将阿拉伯数字按照中文格式输出。相比于 CJKnumb，它提供的三个格式转换命令 \zhnumber，\zhdigits 和 \zhnum 都是可以适当展开的，可以正常使用于 PDF 书签和交叉引用。

zhnumber 支持 GBK 和 UTF8 编码，依赖 LaTeX 3 项目的 expl3，xparse 和 l3keys2e 宏包。

目前 \zhnumber 能正确处理的最大整数是 $10^{48}-1$，\zhditits 不受这个大小的限制。

## 2 使用方法

encoding `encoding = ⟨GBK|UTF8⟩`

用于指定编码的宏包选项，可以在调用宏包的时候设定，也可以用 \zhnumsetup 在导言区内设定。对于 XeLaTeX 和 LuaLaTeX，不用指定编码，宏包将自动使用 UTF8 编码。只有 LaTeX 和 pdfLaTeX 需要指定编码，如果没有指定，默认将使用 GBK，并且此时 zhnumber 宏包应该在 CJK 或 CJKutf8 宏包之后载入。

\zhnumber `\zhnumber {⟨number⟩}`

以中文格式输出数字。这里的数字可以是整数、小数和分数。例如

二十亿零一千二百零二万零一百二十
二十亿零一千二百零二万零一百二十
二十亿零一千二百零二万零一百二十
二千零一十二点零二零一二零
二千零一十二点零
零点二零一二
二万零一百二十分之二万零一百二十
二千零一十二分之零
零分之二千零一十二
二百零一又一百二十分之二千零二十

```
\zhnumber{2012020120}\\
\zhnumber{2 012 020 120}\\
\zhnumber{2,012,020,120}\\
\zhnumber{2012.020120}\\
\zhnumber{2012.}\\
\zhnumber{.2012}\\
\zhnumber{20120/20120}\\
\zhnumber{/2012}\\
\zhnumber{2012/}\\
\zhnumber{201;2020/120}
```

\zhdigits `\zhdigits {⟨number⟩}`

将阿拉伯数字转换为中文数字串。缺省状态下，\zhdigits 将 0 映射为〇，如果需要将其映射为零，可以使用 \zhdigits*。例如

二〇一二〇二〇一二〇
二零一二零二零一二零

```
\zhdigits{2012020120}\\
\zhdigits*{2012020120}
```

`\zhnum` `\zhnum {⟨counter⟩}`

与 `\roman` 等类似，用于将 LaTeX 计数器的值转换为中文数字。例如

二

```
\zhnum{section}
```

`\zhnumsetup` `\zhnumsetup {⟨key1⟩=⟨var1⟩, ⟨key2⟩=⟨var2⟩, ...}`

用于在导言区或文档中，设置中文数字的输出格式。目前可以设置的 ⟨*key*⟩ 如下介绍。

`style` `style = {⟨Simplified⟩|⟨Traditional⟩|⟨Normal⟩|⟨Financial⟩|⟨Ancient⟩}`

意义分别为

`Simplified` 以简体中文输出数字；
`Traditional` 以繁体中文输出数字；
`Normal` 以小写形式输出中文数字；
`Financial` 以大写形式输出中文数字；
`Ancient` 以廿输出 20，以卅输出 30，以卌输出 40，以皕输出 200。

可以设置 `style` 为其中一个，也可以是前三个与后两个的适当组合，默认是简体小写。例如

陆萬貳仟零壹拾貳點叁
廿一

```
\zhnumsetup{style={Traditional,Financial}}
\zhnumber{62012.3}\\
\zhnumsetup{style=Ancient}
\zhnumber{21}
```

`null` `null = ⟨true|***false***⟩`

缺省状态下，除了 `\zhdigits` 外，其它的格式转换命令，将 0 映射成零，如果需要将 0 映射成〇，可以使用这个选项。

zhnumber 提供下列选项来控制阿拉伯数字的中文映射。

```
- -0 0 1 2 3 4 5 6 7 8 9 10 20 30 40 200 dot and parts
E2 E3 E4 E8 E12 E16 E20 E24 E28 E32 E36 E40 E44
```

其中 `-` 设置负，`-0` 设置〇，`dot` 设置小数的点，`and` 和 `parts` 分别设置分数的“又”和“分之”，而 `E`$n$ 设置 $10^n$。例如

```
\zhnumsetup{2={两}}
```

可以将 2 映射成两。需要说明的是，zhnumber 将优先使用这里的设置，所以将会影响到 `style` 选项。如果要恢复 `style` 的功能，可以使用 `reset` 选项。

`reset` `reset`

用于恢复 zhnumber 对阿拉伯数字的初始化映射。zhnumber 的中文数字初始化设置见源代码(第 4 节)。

`\zhnumber` `\zhnumber [⟨options⟩] {⟨number⟩}`
`\zhdigits` `\zhdigits [⟨options⟩] {⟨number⟩}`
`\zhnum` `\zhnum [⟨options⟩] {⟨counter⟩}`

如果只改变当前数字的中文输出格式，可以使用带选项的格式转换命令，其中 ⟨*options*⟩ 与 `\zhnumsetup` 的参数相同，如上所介绍。这些带了选项的命令是不可展开的，在某些场合使用时要小心。

# 3 zhnumber 宏包代码实现

```
⟨*package⟩
\msg_new:nnn { zhnumber } { l3-too-old }
  {
    Support~package~'expl3'~too~old. \\\\
    Please~update~an~up~to~date~version~of~the~bundles\\\\
    'l3kernel'~and~'l3packages'\\\\
    using~your~TeX~package~manager~or~from~CTAN.
  }
\@ifpackagelater { expl3 } { 2012/02/19 } { }
  { \msg_error:nn  { zhnumber }  { l3-too-old } }

\RequirePackage{xparse}
\RequirePackage{l3keys2e}
```

\zhnumber 用于将输入的数字按照中文格式输出。

```
\DeclareExpandableDocumentCommand \zhnumber { o m }
  { \zhnum_number_aux:nNn {#1} \zhnum_number:n {#2} }
\cs_new_nopar:Nn \zhnum_number:n { \zhnum_number:w #1 . \q_nil . \q_stop }
```

(*End definition for* \zhnumber. *This function is documented on page* 2.)

\zhnum_number_aux:nNn

```
\cs_new_nopar:Nn \zhnum_number_aux:nNn
  {
    \IfNoValueTF {#1} { #2 {#3} }
      { \group_begin: \zhnumsetup {#1} #2 {#3} \group_end: }
  }
```

(*End definition for* \zhnum_number_aux:nNn.)

\zhnum_number:w 先判断输入的是小数还是分数。

```
\cs_new_nopar:Npn \zhnum_number:w #1.#2.#3 \q_stop
  {
    \quark_if_nil:nTF {#2}
      { \zhnum_integer_or_fraction:w #1 / \q_nil / \q_stop }
      { \zhnum_decimal:nn {#1} {#2} }
  }
```

(*End definition for* \zhnum_number:w.)

\zhnum_integer_or_fraction:w 判断是否输入的是分数。

```
\cs_new_nopar:Npn \zhnum_integer_or_fraction:w #1/#2/#3 \q_stop
  {
    \quark_if_nil:nTF {#2}
      { \zhnum_integer:f {#1} }
      { \zhnum_fraction:w #2 \q_mark #1 ; \q_nil ; \q_stop }
  }
```

(*End definition for* \zhnum_integer_or_fraction:w.)

\zhnum_fraction:w 对分数进行预处理。

```
\cs_new_nopar:Npn \zhnum_fraction:w #1 \q_mark #2;#3;#4 \q_stop
  {
    \quark_if_nil:nTF {#3}
      {
```

```
        \zhnum_blank_to_zero:f {#1} \c_zhnum_over_tl
        \zhnum_blank_to_zero:f {#2}
      }
      {
        \tl_if_blank:fF {#2} { \zhnumber {#2} \c_zhnum_and_tl }
        \zhnum_blank_to_zero:f {#1} \c_zhnum_over_tl
        \zhnum_blank_to_zero:f {#3}
      }
  }
\cs_generate_variant:Nn \tl_if_blank:nF { f }
```

(*End definition for* `\zhnum_fraction:w`.)

`\zhnum_decimal:nn` 对小数进行预处理。

```
\cs_new_nopar:Nn \zhnum_decimal:nn
  {
    \zhnum_blank_to_zero:f {#1} \c_zhnum_dot_tl
    \tl_if_blank:fTF {#2} { \zhnum_digit_map:n \c_zero } { \zhdigits * {#2} }
  }
\cs_generate_variant:Nn \tl_if_blank:nTF { f }
```

(*End definition for* `\zhnum_decimal:nn`.)

`\zhnum_blank_to_zero:n` 输出小数的整数位。

```
\cs_new_nopar:Nn \zhnum_blank_to_zero:n
  { \tl_if_blank:nTF {#1} { \zhnum_digit_map:n \c_zero } { \zhnumber {#1} } }
\cs_generate_variant:Nn \zhnum_blank_to_zero:n { f }
```

(*End definition for* `\zhnum_blank_to_zero:n`.)

`\zhnum` 用于将 LaTeX 计数器按中文格式输出。

```
\DeclareExpandableDocumentCommand \zhnum { o m }
  { \zhnum_number_aux:nNn {#1} \zhnum_counter:n {#2} }
\cs_new_nopar:Nn \zhnum_counter:n
  {
    \exp_args:Nc \token_if_int_register:NTF { c@#1 }
      { \zhnum_integer:v { c@#1 } }
      { \@nocounterr {#1} }
  }
```

(*End definition for* `\zhnum`. *This function is documented on page* 2.)

`\zhnum_integer:n` 对整数的处理。

```
\cs_new_nopar:Nn \zhnum_integer:n
  { \zhnum_integer_aux:f { \zhnum_erase_separator:n {#1} } }
\cs_new_nopar:Nn \zhnum_integer_aux:n
  {
    \int_compare:nNnT { \int_get_sign:n {#1} \c_one } < \c_zero
      { \zhnum_digit_map:n \c_minus_one }
    \zhnum_parse_number:f { \zhnum_trim_zeros:f { \int_get_digits:n {#1} } }
  }
\cs_generate_variant:Nn \zhnum_integer:n { f , v }
\cs_generate_variant:Nn \zhnum_integer_aux:n { f }
```

(*End definition for* `\zhnum_integer:n`.)

\zhnum_erase_separator:n
\zhnum_trim_zeros:n

去掉分隔符和多余的 0。

```
\cs_new_nopar:Nn \zhnum_erase_separator:n
  { \cs_to_str:c { \tl_map_function:nN {#1} \zhnum_erase_separator_aux:N } }
\cs_new_nopar:Nn \zhnum_erase_separator_aux:N
  { \str_if_eq:xxF {#1} \c_zhnum_separator_tl {#1} }
\cs_new_nopar:Nn \zhnum_trim_zeros:n
  {
    \tl_if_empty:nTF {#1} \c_zero
      {
        \int_compare:nNnTF { \tl_head:n {#1} } = \c_zero
          { \zhnum_trim_zeros:o { \use_none:n #1 } } {#1}
      }
  }
\cs_generate_variant:Nn \zhnum_trim_zeros:n { f , o }
\cs_generate_variant:Nn \cs_to_str:N        { c }
```

(*End definition for* \zhnum_erase_separator:n *and* \zhnum_trim_zeros:n.)

\zhnum_parse_number:n
\zhnum_parse_number:nn
\zhnum_parse_number:nnn

```
\cs_new_nopar:Nn \zhnum_parse_number:n
  { \zhnum_parse_number:nf {#1} { \tl_length:n {#1} } }
\cs_new_nopar:Nn \zhnum_parse_number:nn
  {
    \int_compare:nNnTF {#2} < \c_five
      {
        \int_compare:nNnTF {#1} = \c_zero
          { \zhnum_digit_map:n \c_zero }
          { \zhnum_process_number:NNn \c_true_bool \c_true_bool {#1} }
      }
      {
        \int_compare:nNnTF { \int_mod:nn {#2} \c_four } = \c_zero
          {
            \zhnum_split_number:nNNnfn {#1} \c_true_bool \c_true_bool { \c_zero }
              { \int_div_truncate:nn { #2 - \c_one } \c_four }
              { \c_zero }
          }
          {
            \zhnum_parse_number:nnf {#1} {#2}
              {
                \use:c
                  {
                    zhnum_use_
                    \int_to_roman:n { \int_mod:nn {#2} \c_four }
                    _delimit_by_q_stop:w
                  }
                #1 \q_stop
              }
          }
      }
  }
\cs_new_nopar:Nn \zhnum_parse_number:nnn
  {
    \zhnum_process_number:NNn \c_true_bool \c_true_bool {#3}
    \zhnum_scale_map:n { \int_div_truncate:nn { #2 - \c_one } \c_four }
```

```
    \int_compare:nNnTF { \int_mod:nn {#3} \c_ten } = \c_zero
      { \zhnum_split_number:nNNffn {#1} \c_false_bool \c_true_bool }
      { \zhnum_split_number:nNNffn {#1} \c_true_bool \c_false_bool }
    { \int_mod:nn {#2} \c_four }
    { \int_eval:n { \int_div_truncate:nn { #2 - \c_one } \c_four - \c_one } }
    { \c_zero }
  }
\cs_generate_variant:Nn \zhnum_parse_number:n   {   f }
\cs_generate_variant:Nn \zhnum_parse_number:nn  {  nf }
\cs_generate_variant:Nn \zhnum_parse_number:nnn { nnf }
\cs_new_nopar:Npn \zhnum_use_i_delimit_by_q_stop:w   #1#2     \q_stop {#1}
\cs_new_nopar:Npn \zhnum_use_ii_delimit_by_q_stop:w  #1#2#3   \q_stop {#1#2}
\cs_new_nopar:Npn \zhnum_use_iii_delimit_by_q_stop:w #1#2#3#4 \q_stop {#1#2#3}
```

(*End definition for* `\zhnum_parse_number:n`, `\zhnum_parse_number:nn`, *and* `\zhnum_parse_number:nnn`.)

`\zhnum_split_number:nNNnnn` 将输入的整数由低位到高位,以四位为一段进行处理。

```
\cs_new_nopar:Nn \zhnum_split_number:nNNnnn
  {
    \exp_args:Nf \zhnum_split_number_aux:nnnnnnn
      { \zhnum_number_item:nn {#1} { \c_one + #4 + #6 * \c_four } }
      {#1} {#2} {#3} {#4} {#5} {#6}
  }
\cs_new_nopar:Nn \zhnum_split_number_aux:nnnnnnn
  {
    \int_compare:nNnTF {#1} = \c_zero { \use_i_ii:nnn }
      {
        \bool_if:NF #3 { \zhnum_digit_map:n \c_zero }
        \zhnum_process_number:NNn {#3} {#4} {#1}
        \zhnum_scale_map:n { #6 - #7 }
        \int_compare:nNnTF { \int_mod:nn {#1} \c_ten } = \c_zero
          { \use_i_ii:nnn } { \zhnum_use_i_iii:nnn }
      }
    { \int_compare:nNnF { #6 - #7 } = \c_zero }
    { {
        \zhnum_split_number:nNNnnf
          {#2} \c_false_bool \c_true_bool {#5} {#6} { \int_eval:n { #7 + \c_one } }
    } }
    { {
        \zhnum_split_number:nNNnnf
          {#2} \c_true_bool \c_false_bool {#5} {#6} { \int_eval:n { #7 + \c_one } }
    } }
  }
\cs_new_nopar:Nn \zhnum_use_i_iii:nnn {#1#3}
\cs_generate_variant:Nn \zhnum_split_number:nNNnnn { nNNnf , nNNff , nNNnnf }
```

(*End definition for* `\zhnum_split_number:nNNnnn`.)

`\zhnum_number_item:nn` 截取整数的其中四位数。

`\zhnum_number_item_aux:nN`

```
\cs_new_nopar:Nn \zhnum_number_item:nn
  {
    \zhnum_number_item_aux:nN {#2} #1
    \q_recursion_tail
    \prg_break_point:n { }
  }
```

```
\cs_new_nopar:Nn \zhnum_number_item_aux:nN
  {
    \quark_if_recursion_tail_break:n {#2}
    \int_compare:nNnTF {#1} = \c_one
      { \zhnum_recursion_break:NNNNw #2 }
      { \zhnum_number_item_aux:fN { \int_eval:n { #1 - \c_one } } }
  }
\cs_generate_variant:Nn \zhnum_number_item_aux:nN { f }
\cs_new_nopar:Npn \zhnum_recursion_break:NNNNw #1#2#3#4#5 \prg_break_point:n #6 {#1#2#3#4}
```

(*End definition for* `\zhnum_number_item:nn` *and* `\zhnum_number_item_aux:nN`.)

`\zhnum_process_number:NNn`
`\zhnum_process_number:NNNNNN`

对四位数字按情况进行处理。

```
\cs_new_nopar:Nn \zhnum_process_number:NNn
  {
    \zhnum_process_number:ffffNN
      { \int_mod:nn {#3} \c_ten }
      { \int_mod:nn { \int_div_truncate:nn {#3} \c_ten } \c_ten }
      { \int_mod:nn { \int_div_truncate:nn {#3} \c_one_hundred } \c_ten }
      { \int_div_truncate:nn {#3} \c_one_thousand }
      {#1} {#2}
  }
\cs_new_nopar:Nn \zhnum_process_number:NNNNNN
  {
    \int_compare:nNnTF {#4} = \c_zero
      { \bool_if:NF #6 { \zhnum_digit_map:n \c_zero } }
      { \zhnum_digit_map:n {#4} \zhnum_digit_map:n \c_one_thousand }
    \int_compare:nNnTF {#3} = \c_zero
      { \int_compare:nNnT { #4 * (#2#1) } > \c_zero { \zhnum_digit_map:n \c_zero } }
      {
        \bool_if:nTF
          { \l_zhnum_ancient_bool && \int_compare_p:nNn {#3} = \c_two }
          { \zhnum_digit_map:n { #3 * \c_one_hundred } }
          { \zhnum_digit_map:n {#3} \zhnum_digit_map:n \c_one_hundred }
      }
    \int_compare:nNnTF {#2} = \c_zero
      { \int_compare:nNnT { #3 * #1 } > \c_zero { \zhnum_digit_map:n \c_zero } }
      {
        \bool_if:nF
          {
            \int_compare_p:nNn {#2}   = \c_one  &&
            \int_compare_p:nNn {#4#3} = \c_zero && #6 && #5
          }
          {
            \bool_if:nTF
              {
                \l_zhnum_ancient_bool                    &&
                ( \int_compare_p:nNn {#2} = \c_two   ||
                  \int_compare_p:nNn {#2} = \c_three ||
                  \int_compare_p:nNn {#2} = \c_four )
              }
              {
                \zhnum_digit_map:n { #2 * \c_ten }
                \use_none:nn
```

```
            }
            { \zhnum_digit_map:n {#2} }
        }
      \zhnum_digit_map:n \c_ten
    }
  \int_compare:nNnF {#1} = \c_zero { \zhnum_digit_map:n {#1} }
 }
\cs_generate_variant:Nn \zhnum_process_number:NNn    { NNf  }
\cs_generate_variant:Nn \zhnum_process_number:NNNNNN { ffff }
```

(*End definition for* `\zhnum_process_number:NNn` *and* `\zhnum_process_number:NNNNNN`.)

`\zhdigits` 将输入的数字输出为中文数字串输出。

```
\DeclareExpandableDocumentCommand \zhdigits { s o m }
 {
  \IfBooleanTF {#1}
    { \zhnum_digits_aux:nnN {#2} {#3} \zhnum_digits_zero_aux:N }
    { \zhnum_digits_aux:nnN {#2} {#3} \zhnum_digits_null_aux:N }
 }
\cs_new_nopar:Nn \zhnum_digits_aux:nnN
 {
  \IfNoValueTF {#1} { \tl_map_function:fN {#2} #3 }
    { \group_begin: \zhnumsetup {#1} \tl_map_function:fN {#2} #3 \group_end: }
 }
\cs_generate_variant:Nn \tl_map_function:nN { f }
```

(*End definition for* `\zhdigits`. *This function is documented on page* 2.)

`\zhnum_digits_null_aux:N` `\zhnum_digits_zero_aux:N` `\zhnum_digits_aux:nn` 将输入的数字输出为中文数字串输出。

```
\cs_new_nopar:Nn \zhnum_digits_null_aux:N { \zhnum_digits_aux:nn \c_zhnum_null_int {#1} }
\cs_new_nopar:Nn \zhnum_digits_zero_aux:N { \zhnum_digits_aux:nn \c_zero {#1} }
\cs_new_nopar:Nn \zhnum_digits_aux:nn
 {
  \str_if_eq:xxF {#2} \c_zhnum_separator_tl
    {
      \str_if_eq:xxTF {#2} . \c_zhnum_dot_tl
        {
          \zhnum_digit_map:n
            {
              \str_if_eq:xxTF {#2} - \c_minus_one
                {
                  \bool_if:nTF
                    {
                      \int_compare_p:nNn {#2} = \c_zero           &&
                      \int_compare_p:nNn {#1} = \c_zhnum_null_int
                    }
                    { \c_zhnum_null_int } {#2}
                }
            }
        }
    }
 }
```

(*End definition for* `\zhnum_digits_null_aux:N`, `\zhnum_digits_zero_aux:N`, *and* `\zhnum_digits_aux:nn`.)

`\c_zhnum_null_int` 设置“〇”对应的阿拉伯数字。

```
\int_const:Nn \c_zhnum_null_int { -10 }
```

(*End definition for* `\c_zhnum_null_int`.)

`\zhnum_digit_map:n` 阿拉伯数字与中文数字的映射。

```
\cs_new_nopar:Nn \zhnum_digit_map:n
  {
    \prg_case_int:nnn {#1}
      {
        { \c_minus_one      } { \c_zhnum_minus_tl       }
        { \c_zero           } { \c_zhnum_zero_tl        }
        { \c_one            } { \c_zhnum_one_tl         }
        { \c_two            } { \c_zhnum_two_tl         }
        { \c_three          } { \c_zhnum_three_tl       }
        { \c_four           } { \c_zhnum_four_tl        }
        { \c_five           } { \c_zhnum_five_tl        }
        { \c_six            } { \c_zhnum_six_tl         }
        { \c_seven          } { \c_zhnum_seven_tl       }
        { \c_eight          } { \c_zhnum_eight_tl       }
        { \c_nine           } { \c_zhnum_nine_tl        }
        { \c_ten            } { \c_zhnum_ten_tl         }
        { \c_one_hundred    } { \c_zhnum_hundred_tl     }
        { \c_one_thousand   } { \c_zhnum_thousand_tl    }
        { \c_zhnum_null_int } { \c_zhnum_null_tl        }
        { 20                } { \c_zhnum_twenty_tl      }
        { 30                } { \c_zhnum_thirty_tl      }
        { 40                } { \c_zhnum_forty_tl       }
        { 200               } { \c_zhnum_two_hundred_tl }
      }
      { \prg_do_nothing: }
  }
```

(*End definition for* `\zhnum_digit_map:n`.)

`\zhnum_scale_map:n` 大数系统的映射。

```
\cs_new_nopar:Nn \zhnum_scale_map:n
  {
    \prg_case_int:nnn {#1}
      {
        { \c_zero   } { \c_zhnum_scale_zero_tl   }
        { \c_one    } { \c_zhnum_scale_one_tl    }
        { \c_two    } { \c_zhnum_scale_two_tl    }
        { \c_three  } { \c_zhnum_scale_three_tl  }
        { \c_four   } { \c_zhnum_scale_four_tl   }
        { \c_five   } { \c_zhnum_scale_five_tl   }
        { \c_six    } { \c_zhnum_scale_six_tl    }
        { \c_seven  } { \c_zhnum_scale_seven_tl  }
        { \c_eight  } { \c_zhnum_scale_eight_tl  }
        { \c_nine   } { \c_zhnum_scale_nine_tl   }
        { \c_ten    } { \c_zhnum_scale_ten_tl    }
        { \c_eleven } { \c_zhnum_scale_eleven_tl }
      }
      { \zhnum_scale_map_hook:n {#1} }
  }
```

```
\cs_new_nopar:Nn \zhnum_scale_map_hook:n
  { \zhnum_scale_map:n { \int_mod:nn {#1} \c_eleven } }
```

(*End definition for* `\zhnum_scale_map:n`.)

根据需要设置中文阿拉伯数字。

```
\keys_define:nn { zhnum / options }
  {
    -   .tl_set:N = \c_zhnum_minus_tl        ,
    -0  .tl_set:N = \c_zhnum_null_tl         ,
    0   .tl_set:N = \c_zhnum_zero_tl         ,
    1   .tl_set:N = \c_zhnum_one_tl          ,
    2   .tl_set:N = \c_zhnum_two_tl          ,
    3   .tl_set:N = \c_zhnum_three_tl        ,
    4   .tl_set:N = \c_zhnum_four_tl         ,
    5   .tl_set:N = \c_zhnum_five_tl         ,
    6   .tl_set:N = \c_zhnum_six_tl          ,
    7   .tl_set:N = \c_zhnum_seven_tl        ,
    8   .tl_set:N = \c_zhnum_eight_tl        ,
    9   .tl_set:N = \c_zhnum_nine_tl         ,
    10  .tl_set:N = \c_zhnum_ten_tl          ,
    20  .tl_set:N = \c_zhnum_twenty_tl       ,
    30  .tl_set:N = \c_zhnum_thirty_tl       ,
    40  .tl_set:N = \c_zhnum_forty_tl        ,
    200 .tl_set:N = \c_zhnum_two_hundred_tl  ,
    E2  .tl_set:N = \c_zhnum_hundred_tl      ,
    E3  .tl_set:N = \c_zhnum_thousand_tl     ,
    E4  .tl_set:N = \c_zhnum_scale_one_tl    ,
    E8  .tl_set:N = \c_zhnum_scale_two_tl    ,
    E12 .tl_set:N = \c_zhnum_scale_three_tl  ,
    E16 .tl_set:N = \c_zhnum_scale_four_tl   ,
    E20 .tl_set:N = \c_zhnum_scale_five_tl   ,
    E24 .tl_set:N = \c_zhnum_scale_six_tl    ,
    E28 .tl_set:N = \c_zhnum_scale_seven_tl  ,
    E32 .tl_set:N = \c_zhnum_scale_eight_tl  ,
    E36 .tl_set:N = \c_zhnum_scale_nine_tl   ,
    E40 .tl_set:N = \c_zhnum_scale_ten_tl    ,
    E44 .tl_set:N = \c_zhnum_scale_eleven_tl ,
  }
```

`\zhnum_load_cfg:` 根据选定编码载入配置文件。

```
\cs_new:Nn \zhnum_load_cfg:
  {
    \cs_if_exist:NT \CJK@makeActive
      {
        \int_compare:nNnTF { \char_value_catcode:n {"080} } = \active
          { \bool_set_false:N \l_zhnum_set_CJK_active_bool }
          { \bool_set_true:N \l_zhnum_set_CJK_active_bool \CJK@makeActive }
      }
    \file_input:n { zhnumber - \bool_if:NTF \g_zhnum_gbk_bool { gbk } { utf8 } .cfg }
   \bool_if:nT { \cs_if_exist_p:N \CJK@makeInactive && \l_zhnum_set_CJK_active_bool }
      { \CJK@makeInactive }
  }
```

(*End definition for* `\zhnum_load_cfg:`.)

encoding 宏包设置选项。

style
null
reset

```
\keys_define:nn { zhnum / options }
  {
    encoding .choice: ,
   encoding / UTF8    .code:n = { \bool_gset_false:N \g_zhnum_gbk_bool \zhnum_load_cfg: } ,
   encoding / GBK     .code:n = { \bool_gset_true:N  \g_zhnum_gbk_bool \zhnum_load_cfg: } ,
    encoding          .default:n = { GBK } ,
    style .multichoice: ,
    style / Normal      .code:n =
      {
        \bool_set_false:N \l_zhnum_ancient_bool
        \bool_set_true:N  \l_zhnum_normal_bool
      } ,
    style / Financial   .code:n =
      {
        \bool_set_false:N \l_zhnum_ancient_bool
        \bool_set_false:N \l_zhnum_normal_bool
      } ,
    style / Ancient     .code:n =
      {
        \bool_set_true:N \l_zhnum_ancient_bool
        \bool_set_true:N \l_zhnum_normal_bool
      } ,
    style / Simplified  .code:n = { \bool_set_true:N  \l_zhnum_simp_bool } ,
    style / Traditional .code:n = { \bool_set_false:N \l_zhnum_simp_bool } ,
    style            .default:n = { Normal , Simplified } ,
    null            .bool_set:N = \l_zhnum_null_bool ,
    reset               .code:n = \zhnum_load_cfg: ,
    dot               .tl_set:N = \c_zhnum_dot_tl  ,
    and               .tl_set:N = \c_zhnum_and_tl  ,
    parts             .tl_set:N = \c_zhnum_over_tl ,
    separator         .tl_set:N = \c_zhnum_separator_tl ,
  }
```

(*End definition for* encoding *and others. These functions are documented on page 2.*)

\zhnumsetup 在文档中设置 zhnumber 的接口。

```
\NewDocumentCommand \zhnumsetup { m }
  {
    \keys_set:nn { zhnum / options } {#1}
    \tex_ignorespaces:D
  }
```

(*End definition for* \zhnumsetup. *This function is documented on page 2.*)

初始化设置和执行宏包选项。

```
\keys_set:nn { zhnum / options } { style , null = false , separator = {,} }
\ProcessKeysOptions { zhnum / options }
```

如果没有选定编码,则根据引擎自动设置编码。

```
\ExplSyntaxOn
\bool_if_exist:NF \g_zhnum_gbk_bool
  {
    \exp_args:Nnx \keys_set:nn { zhnum / options }
      {
        encoding =
```

```
          {
            \bool_if:nTF { \xetex_if_engine_p: || \luatex_if_engine_p: }
              { UTF8 } { GBK }
          }
      }
  }
\ExplSyntaxOff
```
⟨/package⟩

# 4 中文数字设置

⟨*config-gbk | config-utf8⟩

```
\tl_set:Nn \c_zhnum_minus_tl    { \bool_if:NTF \l_zhnum_simp_bool { 负 } { 負 } }
\tl_set:Nn \c_zhnum_zero_tl   { \bool_if:nTF \l_zhnum_null_bool \c_zhnum_null_tl { 零 } }
\tl_set:Nn \c_zhnum_null_tl     { \bool_if:NTF \l_zhnum_normal_bool { 〇 } { 零 } }
\tl_set:Nn \c_zhnum_one_tl      { \bool_if:NTF \l_zhnum_normal_bool { 一 } { 壹 } }
\tl_set:Nn \c_zhnum_two_tl    { \bool_if:NTF \l_zhnum_normal_bool { 二 } { \bool_if:NTF \l_zhnum_simp_bo
\tl_set:Nn \c_zhnum_three_tl   { \bool_if:NTF \l_zhnum_normal_bool { 三 } { \bool_if:NTF \l_zhnum_simp_b
\tl_set:Nn \c_zhnum_four_tl     { \bool_if:NTF \l_zhnum_normal_bool { 四 } { 肆 } }
\tl_set:Nn \c_zhnum_five_tl     { \bool_if:NTF \l_zhnum_normal_bool { 五 } { 伍 } }
\tl_set:Nn \c_zhnum_six_tl    { \bool_if:NTF \l_zhnum_normal_bool { 六 } { \bool_if:NTF \l_zhnum_simp_bo
\tl_set:Nn \c_zhnum_seven_tl    { \bool_if:NTF \l_zhnum_normal_bool { 七 } { 柒 } }
\tl_set:Nn \c_zhnum_eight_tl    { \bool_if:NTF \l_zhnum_normal_bool { 八 } { 捌 } }
\tl_set:Nn \c_zhnum_nine_tl     { \bool_if:NTF \l_zhnum_normal_bool { 九 } { 玖 } }
\tl_set:Nn \c_zhnum_ten_tl      { \bool_if:NTF \l_zhnum_normal_bool { 十 } { 拾 } }
\tl_set:Nn \c_zhnum_hundred_tl  { \bool_if:NTF \l_zhnum_normal_bool { 百 } { 佰 } }
\tl_set:Nn \c_zhnum_thousand_tl { \bool_if:NTF \l_zhnum_normal_bool { 千 } { 仟 } }
\tl_set:Nn \c_zhnum_twenty_tl      { 廿 }
\tl_set:Nn \c_zhnum_thirty_tl      { 卅 }
\tl_set:Nn \c_zhnum_forty_tl       { 卌 }
\tl_set:Nn \c_zhnum_two_hundred_tl { 皕 }
\tl_set:Nn \c_zhnum_dot_tl { \bool_if:NTF \l_zhnum_simp_bool { 点 } { 點 } }
\tl_set:Nn \c_zhnum_and_tl  { 又 }
\tl_set:Nn \c_zhnum_over_tl { 分之 }
\tl_set:Nn \c_zhnum_scale_zero_tl   { }
\tl_set:Nn \c_zhnum_scale_one_tl    { \bool_if:NTF \l_zhnum_simp_bool { 万 } { 萬 } }
\tl_set:Nn \c_zhnum_scale_two_tl    { \bool_if:NTF \l_zhnum_simp_bool { 亿 } { 億 } }
\tl_set:Nn \c_zhnum_scale_three_tl  { 兆 }
\tl_set:Nn \c_zhnum_scale_four_tl   { 京 }
\tl_set:Nn \c_zhnum_scale_five_tl   { 垓 }
\tl_set:Nn \c_zhnum_scale_six_tl    { 秭 }
\tl_set:Nn \c_zhnum_scale_seven_tl  { 穰 }
\tl_set:Nn \c_zhnum_scale_eight_tl  { \bool_if:NTF \l_zhnum_simp_bool { 沟 } { 溝 } }
\tl_set:Nn \c_zhnum_scale_nine_tl   { \bool_if:NTF \l_zhnum_simp_bool { 涧 } { 澗 } }
\tl_set:Nn \c_zhnum_scale_ten_tl    { 正 }
\tl_set:Nn \c_zhnum_scale_eleven_tl { \bool_if:NTF \l_zhnum_simp_bool { 载 } { 載 } }
\ExplSyntaxOff
```
⟨/config-gbk | config-utf8⟩