

zhnumber 宏包

李清

sobenlee@gmail.com

2016/02/02 v2.3*

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, Big5 和 UTF8 编码, 依赖 L^AT_EX 3 项目的 `expl3`, `xparse` 和 `l3keys2e` 宏包。

2 使用方法

`encoding`
Updated: 2014-09-09

`encoding = <GBK|Big5|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 `upLATEX`, `XYLATEX` 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。

`\zhnumber` ☆
Updated: 2014-09-12

`\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

```
1 \zhnumber{2012020120}\\  
2 \zhnumber{2 012 020 120}\\  
3 \zhnumber{2,012,020,120}\\  
4 \zhnumber{2012.020120}\\  
5 \zhnumber{2012.}\\  
6 \zhnumber{.2012}\\  
7 \zhnumber{20120/20120}\\  
8 \zhnumber{/2012}\\  
9 \zhnumber{2012/}\\  
10 \zhnumber{201;2020/120}
```

`\zhdigits` ☆
Updated: 2014-09-09

`\zhdigits <{number}>`
`\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 ○, 如果需要将其映射为零, 可以使用带星号的形式。例如

二〇一二〇二〇一二〇
二零一二零二零一二零

```
1 \zhdigits{2012020120}\\  
2 \zhdigits*{2012020120}
```

`\zhnum` ☆
Updated: 2014-09-09

`\zhnum <{counter}>`

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

二

```
1 \zhnum{section}
```

*ctex-kit rev. 9686fd1.

`\zhweekday` ☆ `\zhweekday {<yyyy/mm/dd>}`

New: 2012-05-25

输出日期当天的星期。例如
星期日

```
1 \zhweekday{2012/5/20}
```

`\zhdate` ☆ `\zhdate {<yyyy/mm/dd>}`

`\zhdate * {<yyyy/mm/dd>}`

New: 2012-05-25

以中文格式输出日期,其中带 * 的命令还输出星期。例如

2012 年 5 月 21 日
2012 年 5 月 21 日星期一

```
1 \zhdate{2012/5/21}\
2 \zhdate*{2012/5/21}
```

`\zhtoday` ☆ 与 `\today` 类似,以中文输出当天的日期。例如

New: 2012-05-25

2016 年 2 月 2 日

```
1 \zhtoday
```

`\zhtime` ☆ `\zhtime {<hh:mm>}`

New: 2012-05-25

以中文格式输出时间。例如

23 时 56 分

```
1 \zhtime{23:56}
```

`\zhcurrtime` ☆ 输出当前的时间。例如

New: 2012-05-25

17 时 1 分

```
1 \zhcurrtime
```

`\zhtiangan` ☆ `\zhtiangan {<number>}`

New: 2015-05-20

输出对应的天干计数。<number> 的正常范围是 1–10,超出范围的数字将输出空值。例如

甲 乙 丙 丁 戊 癸

```
1 \zhtiangan{1} \zhtiangan{2} \zhtiangan{3}
2 \zhtiangan{4} \zhtiangan{5} \zhtiangan{10}
```

`\zhdizhi` ☆ `\zhdizhi {<number>}`

New: 2015-05-20

输出对应的地支计数。<number> 的正常范围是 1–12,超出范围的数字将输出空值。例如

子 丑 寅 卯 辰 亥

```
1 \zhdizhi{1} \zhdizhi{2} \zhdizhi{3}
2 \zhdizhi{4} \zhdizhi{5} \zhdizhi{12}
```

`\zhganzhi` ☆ `\zhganzhi {<number>}`

New: 2015-05-20

输出对应的干支计数。<number> 的正常范围是 1–60,超出范围的数字将输出空值。例如

甲子 乙丑 丙寅
丁卯 戊辰 癸亥

```
1 \zhganzhi{1} \zhganzhi{2} \zhganzhi{3} \
2 \zhganzhi{4} \zhganzhi{5} \zhganzhi{60}
```

`\zhganzhinian` ☆ `\zhganzhinian {<year>}`

New: 2015-05-20

输出公元纪年 <year> 对应的干支纪年。公元前的年份用负数表示。例如

戊戌 乙卯
甲子 丙申

```
1 \zhganzhinian{1898} \zhganzhinian{-246} \
2 \zhganzhinian{-2697} \zhganzhinian{<year>}
```

`\zhnumExtendScaleMap` `\zhnumExtendScaleMap [<character>] { <character>_1, <character>_2, ..., <character>_n }`

New: 2012-05-25

缺省状态下 `\zhnumber` 能正确中文格式化的最大整数是 $10^{48} - 1$, `\zhdigits` 不受这个大小的限制。可以通过 `\zhnumExtendScaleMap` 来扩展 `\zhnumber`。<character>_i 设置 $10^{4-i \rightarrow 11}$ 。若给出可选项 <character>,则当数字大于 $10^{4-n \rightarrow 12} - 1$ 时,统一用 <character> 设置输出数字的进位。

`\zhnumsetup` `\zhnumsetup {<key1>=<val1>, <key2>=<val2>, ...}`

用于在引言区或文档中,设置中文数字的输出格式。目前可以设置的 *<key>* 如下介绍。以**粗体**表示选项的默认值。

`time` `time = <(Arabic)|Chinese>`

设置日期和时间的数字格式, *<Arabic>* 为阿拉伯数字,而 *<Chinese>* 为中文数字。例如

二〇一六年二月二日十七时一分

```
1 \zhnumsetup{time=Chinese}
2 \zhtoday\zhcurrtime
```

`style` `style = <((Simplified)|Traditional|(Normal)|Financial|Ancient)>`

Updated: 2012-05-25 意义分别为

- Simplified** 以简体中文输出数字(对 Big5 编码无效);
- Traditional** 以繁体中文输出数字(对 Big5 编码无效);
- 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,以𠫪输出 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

陸萬貳仟零壹拾貳點叁
廿一

```
1 \zhnumsetup{style={Traditional,Financial}}
2 \zhnumber{62012.3}\
3 \zhnumsetup{style=Ancient}
4 \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。

`ganzhi-cyclic` `ganzhi-cyclic = <true|false>`

New: 2015-05-20

天干、地支和干支的数字都有一定范围。若参数大于这个范围, `\tiangan` 等将输出空值。可以将本选项设置为 `true`,对超出范围的数字取相应的模。请注意,数字 0 的结果总是为空值。例如

甲乙壬癸壬辛
子亥戌亥戌酉
甲子乙亥辛酉
癸亥壬戌乙卯

```
1 \zhnumsetup{ganzhi-cyclic}
2 \zhtiangan{11} \zhtiangan{12} \zhtiangan{209}
3 \zhtiangan{-1} \zhtiangan{-2} \zhtiangan{-683} \
4 \zhdizhi{13} \zhdizhi{24} \zhdizhi{1211}
5 \zhdizhi{-1} \zhdizhi{-2} \zhdizhi{-8199} \
6 \zhganzhi{61} \zhganzhi{72} \zhganzhi{2158} \
7 \zhganzhi{-1} \zhganzhi{-2} \zhganzhi{-789}
```

`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 40 100 200 1000
E2 E3 E4 E8 E12 E16 E20 E24 E28 E32 E36 E40 E44
F0 F1 F2 F3 F4 F5 F6 F7 F8 F9 F10 F100 F1000 FE2 FE3
T1 T2 T3 T4 T5 T6 T7 T8 T9 T10
D1 D2 D3 D4 D5 D6 D7 D8 D9 D10 D11 D12
GZ1 GZ2 GZ3 GZ4 GZ5 GZ6 GZ7 GZ8 GZ9 GZ10 ... GZ60
dot and parts
year month day hour minute weekday mon tue wed thu fri sat sun
```

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”, `En` 设置 10^n , `Fn` 设置数字 n 的大写, `Tn` 设置数字 n 的天干, `Dn` 设置数字 n 的地支,而 `GZn` 设置数字 n 的干支。其它的选项同字面意思,不再赘述。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, 所以可能会影响到 `style` 选项。如果要恢复 `style` 的功能, 可以使用 `reset` 选项。

<code>reset</code>	<code>reset</code>
Updated: 2014-09-12	用于恢复 <code>zhnumber</code> 对阿拉伯数字的初始化映射。 <code>zhnumber</code> 的中文数字初始化设置见源代码(第 4 节)。

<code>activechar</code>	<code>activechar = <true false></code>
New: 2014-09-09	在 <code>L^AT_EX</code> 或者 <code>pdfL^AT_EX</code> 下面输出汉字, 传统的办法需要将汉字的首字节设置为活动字符, 然后再通过特殊的宏技巧来实现。因此, <code>zhnumber</code> 在载入配置文件的时候, 默认会将汉字的首字节设置为活动字符。禁用本选项将不会改变汉字首字节的类代码。需要在本选项之后, 使用 <code>encoding</code> 或者 <code>reset</code> 选项才会有效果。

<code>\zhnumber</code>	<code>\zhnumber</code> [<code>options</code>] { <code>number</code> }
<code>\zhdigits</code>	<code>\zhdigits</code> * [<code>options</code>] { <code>number</code> }
<code>\zhnum</code>	<code>\zhnum</code> [<code>options</code>] { <code>counter</code> }
Updated: 2014-09-09	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 <code><options></code> 与 <code>\zhnumsetup</code> 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

```

1 <*package>
2 <@@=zhnum>
3 \msg_new:nnn { zhnumber } { l3-too-old }
4 {
5   Support~package~'expl3'~too~old. \\\
6   Please~update~an~up~to~date~version~of~the~bundles\\\
7   'l3kernel'~and~'l3packages'\\\
8   using~your~TeX~package~manager~or~from~CTAN.
9 }
10 \@ifpackagelater { expl3 } { 2015/09/24 } { }
11 { \msg_error:nn { zhnumber } { l3-too-old } }
12 \RequirePackage { xparse , l3keys2e }

```

`\zhnumber` 用于将输入的数字按照中文格式输出。

```

13 \DeclareExpandableDocumentCommand \zhnumber { +o +m }
14 {
15   \IfNoValueTF {#1}
16     { \zhnum_number:f }
17     { \zhnumberwithoptions {#1} }
18   {#2}
19 }

```

(End definition for `\zhnumber`. This function is documented on page 4.)

`\zhnumberwithoptions` 带选项的用户函数。

```

20 \NewDocumentCommand \zhnumberwithoptions { +m +m }
21 {
22   \group_begin:
23     \keys_set:nn { zhnum / options } {#1}
24     \zhnum_number:f {#2}
25   \group_end:
26 }

```

(End definition for `\zhnumberwithoptions`.)

`\zhnum_number:n` 先判断输入的是小数还是分数。

```

27 \cs_new:Npn \zhnum_number:n #1
28 { \__zhnum_number:www #1 . \q_nil . \q_stop }
29 \cs_new:Npn \__zhnum_number:www #1 . #2 . #3 \q_stop
30 {
31   \quark_if_nil:nTF {#2}

```

```

32     { \__zhnum_integer_or_fraction:www #1 / \q_nil / \q_stop }
33     { \zhnum_decimal:nn {#1} {#2} }
34   }
35   \cs_generate_variant:Nn \zhnum_number:n { f }

```

(End definition for \zhnum_number:n.)

__zhnum_integer_or_fraction:www 判断是否输入的是分数。

```

36   \cs_new:Npn \__zhnum_integer_or_fraction:www #1 / #2 / #3 \q_stop
37   {
38     \quark_if_nil:nTF {#2}
39     { \zhnum_integer:n {#1} }
40     { \__zhnum_fraction:www #2 \q_mark #1 ; \q_nil ; \q_stop }
41   }

```

(End definition for __zhnum_integer_or_fraction:www.)

__zhnum_fraction:www 对分数进行预处理。

```

42   \cs_new:Npn \__zhnum_fraction:www #1 \q_mark #2 ; #3 ; #4 \q_stop
43   {
44     \quark_if_nil:nTF {#3}
45     {
46       \zhnum_blank_to_zero:n {#1}
47       \c__zhnum_parts_tl
48       \zhnum_blank_to_zero:n {#2}
49     }
50     {
51       \tl_if_blank:nF {#2}
52       {
53         \zhnum_number:n {#2}
54         \c__zhnum_and_tl
55       }
56       \zhnum_blank_to_zero:n {#1}
57       \c__zhnum_parts_tl
58       \zhnum_blank_to_zero:n {#3}
59     }
60   }

```

(End definition for __zhnum_fraction:www.)

\zhnum_decimal:nn 对小数进行预处理。

```

61   \cs_new:Npn \zhnum_decimal:nn #1#2
62   {
63     \zhnum_blank_to_zero:n {#1} \c__zhnum_dot_tl
64     \tl_if_blank:nTF {#2}
65     { \c__zhnum_zero_tl }
66     { \zhnum_digits_zero:n {#2} }
67   }

```

(End definition for \zhnum_decimal:nn.)

\zhnum_blank_to_zero:n 输出小数的整数位。

```

68   \cs_new:Npn \zhnum_blank_to_zero:n #1
69   {
70     \tl_if_blank:nTF {#1}
71     { \c__zhnum_zero_tl }
72     { \zhnum_number:n {#1} }
73   }

```

(End definition for \zhnum_blank_to_zero:n.)

\zhnum 用于将 L^AT_EX 计数器按中文格式输出。

```

\zhnumberwithoptions 74 \DeclareExpandableDocumentCommand \zhnum { +o +m }
75 {
76   \IfNoValueTF {#1}
77   { \zhnum_counter:n }
78   { \zhnumwithoptions {#1} }
79   {#2}
80 }

```

```

81 \NewDocumentCommand \zhnumwithoptions { +m +m }
82 {
83   \group_begin:
84     \keys_set:nn { zhnum / options } {#1}
85     \zhnum_counter:n {#2}
86   \group_end:
87 }

```

(End definition for `\zhnum` and `\zhnumberwithoptions`. These functions are documented on page 4.)

`\zhnum_counter:n` 可以直接通过比较 L^AT_EX 计数器的值来得到符号和绝对值。

```

\zhnum_int:n
88 \cs_new:Npn \zhnum_counter:n #1
89 {
90   \int_if_exist:cTF { c@#1 }
91   { \zhnum_int:c { c@#1 } }
92   { \__zhnum_counter_error:n {#1} }
93 }
94 \cs_new:Npn \__zhnum_counter_error:n #1
95 { \msg_expandable_error:nnn { zhnumber } { not-counter } {#1} }
96 \msg_new:nnn { zhnumber } { not-counter }
97 { `#1'~is~not~a~LaTeX~counter. }
98 \cs_new:Npn \zhnum_int:n #1
99 {
100   \int_compare:nNnTF {#1} > \c_zero
101   { \zhnum_parse_number:f { \int_eval:n {#1} } }
102   {
103     \int_compare:nNnTF {#1} < \c_zero
104     {
105       \c__zhnum_minus_tl
106       \zhnum_parse_number:f { \int_eval:n { - #1 } }
107     }
108     { \c__zhnum_zero_tl }
109   }
110 }
111 \cs_generate_variant:Nn \zhnum_int:n { c }

```

(End definition for `\zhnum_counter:n` and `\zhnum_int:n`.)

`\zhnum_integer:n` 对整数的处理。这个函数基本抄录自 l3bigint 的 `__bingint_read_do:nn`。它可以正确取得符号，去掉多余的零，还可以循环展开数字。但它在遇到非数字的时候就停止了循环，我们可能需要非数字(例如逗号)来作为分隔符号。因此对它略作修改，跳过非数字。

```

112 \cs_new:Npn \zhnum_integer:n #1
113 {
114   \exp_after:wN \__zhnum_read_integer:www
115   \tex_number:D
116   \exp_after:wN \__zhnum_read_sign_loop:N
117   \exp:w \exp_end_continue_f:w \use:n
118   #1 \exp_stop_f: \q_recursion_tail \q_recursion_stop
119   \__zhnum_result:nn { \c_zero } { } ;
120 }
121 \cs_new:Npn \__zhnum_read_sign_loop:N #1
122 {
123   \if:w + \if:w - \exp_not:N #1 + \fi: \exp_not:N #1
124   \exp_after:wN \__zhnum_read_sign_loop:N
125   \exp:w \exp_end_continue_f:w \exp_after:wN \use:n
126   \else:
127     1 \exp_after:wN ;
128     \exp:w \exp_end_continue_f:w
129     \exp_after:wN \__zhnum_read_zeros_loop:N
130     \exp_after:wN #1
131   \fi:
132 }
133 \cs_new:Npn \__zhnum_read_zeros_loop:N #1
134 {
135   \if:w 0 \exp_not:N #1
136   \exp_after:wN \__zhnum_read_zeros_loop:N
137   \exp:w \exp_end_continue_f:w \exp_after:wN \use:n
138   \else:

```

```

139     \exp_after:wN \__zhnum_read_abs_loop:Nw
140     \exp_after:wN #1
141     \fi:
142   }

```

(End definition for \zhnum_integer:n.)

__zhnum_read_abs_loop:Nw 当数字很大时, l3bigint 的实现会造成 TeX 内存溢出:

```
! TeX capacity exceeded, sorry [expansion depth=10000].
```

我们在这里参考 __tl_act:NNNnn 的实现对它进行了改进。

```

143 \cs_new:Npn \__zhnum_read_abs_loop:Nw #1#2 \q_recursion_stop
144 {
145   \zhnum_if_digit:NTF #1
146   { \__zhnum_output:nnwnn { + \c_one } #1 }
147   { \quark_if_recursion_tail_stop_do:Nn #1 { \__zhnum_loop_end:wnn } }
148   \exp_after:wN \__zhnum_read_abs_loop:Nw
149   \exp:w \exp_end_continue_f:w \use:n #2 \q_recursion_stop
150 }
151 \cs_new:Npn \__zhnum_output:nnwnn #1#2#3 \__zhnum_result:nn #4#5
152 { #3 \__zhnum_result:nn { #4#1 } { #5#2 } }
153 \cs_new:Npn \__zhnum_loop_end:wnn #1 \__zhnum_result:nn #2#3
154 { \int_eval:n {#2} ; #3 }

```

(End definition for __zhnum_read_abs_loop:Nw.)

__zhnum_read_integer:www #1 符号, #3 是绝对值, #2 是绝对值的长度。

```

155 \cs_new:Npn \__zhnum_read_integer:www #1 ; #2 ; #3 ;
156 {
157   \int_compare:nNnTF {#2} = \c_zero
158   { \c__zhnum_zero_tl }
159   {
160     \int_compare:nNnF {#1} = \c_one
161     { \c__zhnum_minus_tl }
162     \zhnum_parse_number:nn {#2} {#3}
163   }
164 }

```

(End definition for __zhnum_read_integer:www.)

\zhnum_if_digit:NTF 判断 #1 是否为数字位。

```

165 \cs_new:Npn \zhnum_if_digit:NTF #1
166 {
167   \if_int_compare:w \c_nine < 1 \exp_not:N #1 \exp_stop_f:
168   \exp_after:wN \use_i:nn
169   \else:
170   \exp_after:wN \use_ii:nn
171   \fi:
172 }

```

(End definition for \zhnum_if_digit:NTF.)

\zhnum_parse_number:n

\zhnum_parse_number:nn

```

173 \cs_new:Npn \zhnum_parse_number:n #1
174 { \exp_args:Nf \zhnum_parse_number:nn { \tl_count:n {#1} } {#1} }
175 \cs_new:Npn \zhnum_parse_number:nn #1
176 { \exp_args:Nf \__zhnum_parse_number:nnn { \int_mod:nn {#1} \c_four } {#1} }
177 \cs_new:Npn \__zhnum_parse_number:nnn #1#2
178 {
179   \int_compare:nNnTF {#2} < \c_two
180   { \zhnum_digit_map:n }
181   {
182     \int_compare:nNnTF {#1} = \c_zero
183     { \zhnum_split_number:fn { \int_eval:n { #2 / \c_four - \c_one } } }
184     { \__zhnum_split_number_aux:nnn {#1} {#2} }
185   }
186 }
187 \cs_generate_variant:Nn \zhnum_parse_number:n { f }

```

(End definition for \zhnum_parse_number:n and \zhnum_parse_number:nn.)

_zhnum_split_number_aux:nnn 为了处理的方便,在整数前面补上适当的0,使其位数可以被4整除。

```
188 \cs_new:Npn \_zhnum_split_number_aux:nnn #1#2
189 {
190   \exp_after:wN \_zhnum_split_number_aux:wwn
191   \tex_number:D \int_div_truncate:nn {#2} \c_four
192   \if_case:w #1 \exp_stop_f:
193     \or: \exp_after:wN \use:n
194     \or: \exp_after:wN \use_i_ii:nnn
195     \or: \exp_after:wN \use_i:nnn
196   \fi:
197   { \exp_stop_f: ; 0 } 0 0 ;
198 }
199 \cs_new:Npn \_zhnum_split_number_aux:wwn #1 ; #2 ; #3
200 { \zhnum_split_number:nn {#1} { #2#3 } }
```

(End definition for _zhnum_split_number_aux:nnn.)

\zhnum_split_number:nn 最后加入的\q_recursion_tail是停止递归的标志,而\q_nil用于占位。

```
201 \cs_new:Npn \zhnum_split_number:nn #1#2
202 {
203   \zhnum_split_number:NNnNNNw \c_true_bool \c_true_bool {#1}
204   #2 \q_recursion_tail \q_nil \q_nil \q_nil \q_recursion_stop
205 }
206 \cs_generate_variant:Nn \zhnum_split_number:nn { f }
```

(End definition for \zhnum_split_number:nn.)

zhnum_split_number:NNnNNNw 将输入的整数由高位到低位,以四位为一段进行处理。

```
207 \cs_new:Npn \zhnum_split_number:NNnNNNw #1#2#3#4#5#6#7
208 {
209   \quark_if_recursion_tail_stop:N #4
210   \int_compare:nNnTF { #4#5#6#7 } = \c_zero
211     { \use_i:nn }
212     {
213       \bool_if:NF #1 { \c__zhnum_zero_tl }
214       \zhnum_process_number:NNNNNN #4#5#6#7#1#2
215       \zhnum_scale_map:n {#3}
216       \int_compare:nNnTF {#7} = \c_zero
217     }
218     { \zhnum_split_number:NNfNNNw \c_false_bool \c_true_bool }
219     { \zhnum_split_number:NNfNNNw \c_true_bool \c_false_bool }
220     { \int_eval:n { #3 - \c_one } }
221   }
222 \cs_generate_variant:Nn \zhnum_split_number:NNnNNNw { NNf }
```

(End definition for \zhnum_split_number:NNnNNNw.)

zhnum_process_number:NNNNNN 对四位数字按情况进行处理。

```
223 \cs_new:Npn \zhnum_process_number:NNNNNN #1#2#3#4#5#6
224 {
225   \int_compare:nNnTF {#1} = \c_zero
226     { \bool_if:NF #6 { \c__zhnum_zero_tl } }
227     { \zhnum_digit_map:n {#1} \c__zhnum_thousand_tl }
228   \int_compare:nNnTF {#2} = \c_zero
229     { \int_compare:nNnF { #1 * (#3#4) } = \c_zero { \c__zhnum_zero_tl } }
230     {
231       \bool_if:nTF
232         { \l__zhnum_ancient_bool && \int_compare_p:nNn {#2} = \c_two }
233         { \zhnum_digit_map:n { #2 00 } }
234         { \zhnum_digit_map:n {#2} \c__zhnum_hundred_tl }
235     }
236   \int_compare:nNnTF {#3} = \c_zero
237     { \int_compare:nNnF { #2 * #4 } = \c_zero { \c__zhnum_zero_tl } }
238     {
239       \bool_if:nF
240         {
241           \int_compare_p:nNn {#3} = \c_one &&
```



```

242         \int_compare_p:nNn {#1#2} = \c_zero && #6 && #5
243     }
244     {
245         \bool_if:nTF
246         {
247             \l__zhnum_ancient_bool           &&
248             ( \int_compare_p:nNn {#3} = \c_two ||
249               \int_compare_p:nNn {#3} = \c_three ||
250               \int_compare_p:nNn {#3} = \c_four )
251         }
252         { \zhnum_digit_map:n { #3 0 } \use_none:n }
253         { \zhnum_digit_map:n {#3} }
254     }
255     \c__zhnum_ten_tl
256 }
257 \int_compare:nNnF {#4} = \c_zero { \zhnum_digit_map:n {#4} }
258 }

```

(End definition for `\zhnum_process_number:NNNNNN`.)

`\zhdigits` 将输入的数字输出为中文数字串输出。

```

\zhdigitsoptions 259 \DeclareExpandableDocumentCommand \zhdigits { +s +o +m }
260 {
261     \IfNoValueTF {#2}
262     { \zhnum_digits:Nn #1 }
263     { \zhdigitsoptions {#1} {#2} }
264     {#3}
265 }
266 \NewDocumentCommand \zhdigitsoptions { +m +m +m }
267 {
268     \group_begin:
269     \keys_set:nn { zhnum / options } {#2}
270     \zhnum_digits:Nn #1 {#3}
271     \group_end:
272 }

```

(End definition for `\zhdigits` and `\zhdigitsoptions`. These functions are documented on page 4.)

`\zhnum_digits_zero:n` 快捷方式。

```

\zhnum_digits_zero:n 273 \cs_new_nopar:Npn \zhnum_digits_zero:n
\zhnum_digits_null:n 274 { \zhnum_digits:Nn \BooleanTrue }
275 \cs_new_nopar:Npn \zhnum_digits_null:n
276 { \zhnum_digits:Nn \BooleanFalse }
277 \cs_generate_variant:Nn \zhnum_digits_null:n { V }

```

(End definition for `\zhnum_digits_zero:n` and `\zhnum_digits_null:n`.)

`\zhnum_digits:Nn` 与 `\zhnum_integer:n` 类似,但不用去掉多余的零。

```

278 \cs_new:Npn \zhnum_digits:Nn #1#2
279 {
280     \exp_after:wN \__zhnum_read_digits:w
281     \tex_number:D
282     \exp_after:wN \__zhnum_read_sign_loop:NN \exp_after:wN #1
283     \exp:w \exp_end_continue_f:w \use:n
284     #2 \exp_stop_f: \q_recursion_tail \q_recursion_stop
285 }
286 \cs_new:Npn \__zhnum_read_sign_loop:NN #1#2
287 {
288     \if:w + \if:w - \exp_not:N #2 + \fi: \exp_not:N #2
289     \exp_after:wN \__zhnum_read_sign_loop:NN \exp_after:wN #1
290     \exp:w \exp_end_continue_f:w \exp_after:wN \use:n
291     \else:
292     1 \exp_after:wN ;
293     \exp_after:wN \__zhnum_read_digits_loop:NN
294     \exp_after:wN #1
295     \exp_after:wN #2
296     \fi:
297 }
298 \cs_new:Npn \__zhnum_read_digits_loop:NN #1#2

```

```

299 {
300   \zhnum_if_digit:NTF #2
301   { \__zhnum_output_digits:NN #1#2 }
302   {
303     \quark_if_recursion_tail_stop:N #2
304     \if:w .\exp_not:N #2 \exp_after:wN \c__zhnum_dot_tl \fi:
305   }
306   \exp_after:wN \__zhnum_read_digits_loop:NN \exp_after:wN #1
307   \exp:w \exp_end_continue_f:w \use:n
308 }
309 \cs_new:Npn \__zhnum_read_digits:w #1 ;
310 {
311   \int_compare:nNnF {#1} = \c_one
312   { \c__zhnum_minus_tl }
313 }
314 \cs_new:Npn \__zhnum_output_digits:NN #1#2
315 {
316   \cs:w
317   c__zhnum_
318   \if_int_compare:w #2 = \c_zero
319   \IfBooleanTF #1 { zero } { null }
320   \else:
321   #2
322   \fi:
323   _tl
324   \cs_end:
325 }

```

(End definition for `\zhnum_digits:Nn`.)

`\zhdate` 输出中文日期。

```

326 \DeclareExpandableDocumentCommand \zhdate { +s +m }
327 {
328   \__zhnum_date:www #2 \q_stop
329   \IfBooleanT #1
330   { \__zhnum_week_day:www #2 \q_stop }
331 }
332 \cs_new:Npn \__zhnum_date:www #1/#2/#3 \q_stop
333 {
334   \zhnum_check_time:Nn \zhnum_digits_null:n {#1} \c__zhnum_year_tl
335   \zhnum_check_time:Nn \zhnum_int:n {#2} \c__zhnum_month_tl
336   \zhnum_check_time:Nn \zhnum_int:n {#3} \c__zhnum_day_tl
337 }

```

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

`\zhtoday` 输出当天日期。

```

338 \cs_new_nopar:Npn \zhtoday
339 {
340   \zhnum_check_time:Nn \zhnum_digits_null:V \tex_year:D \c__zhnum_year_tl
341   \zhnum_check_time:Nn \zhnum_int:n \tex_month:D \c__zhnum_month_tl
342   \zhnum_check_time:Nn \zhnum_int:n \tex_day:D \c__zhnum_day_tl
343 }

```

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

`\zhnum_check_time:Nn` 判断是用中文数字还是用阿拉伯数组。

```

344 \cs_new:Npn \zhnum_check_time:Nn #1
345 { \bool_if:NTF \l__zhnum_time_bool {#1} { \int_to_arabic:n } }

```

(End definition for `\zhnum_check_time:Nn`.)

`\zhweekday` 输出星期

```

346 \cs_new:Npn \zhweekday #1
347 { \__zhnum_week_day:www #1 \q_stop }

```

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

`_znum_week_day:www` 用 Zeller 公式计算的结果 h 与实际星期的关系是 $d / h \rightarrow 5 \leftarrow \text{mod } 7 \leftarrow \rightarrow 1$ 。

```

348 \cs_new:Npn \_znum_week_day:www #1/#2/#3 \q_stop
349 {
350   \if_case:w \znum_Zeller:nnn {#1} {#2} {#3} \exp_stop_f:
351     \c_znum_sat_tl
352     \or: \c_znum_sun_tl
353     \or: \c_znum_mon_tl
354     \or: \c_znum_tue_tl
355     \or: \c_znum_wed_tl
356     \or: \c_znum_thu_tl
357     \or: \c_znum_fri_tl
358   \fi:
359 }

```

(End definition for `_znum_week_day:www`.)

`\znum_Zeller:nnn` 用 Zeller 公式¹ 计算星期几。

```

\znum_Zeller_aux:Nnnn
\znum_two_digits:n
360 \cs_new:Npn \znum_Zeller:nnn #1#2#3
361 {
362   \int_compare:nNnTF
363     { #1 \znum_two_digits:n {#2} \znum_two_digits:n {#3} } > { 1582 10 04 }
364     { \_znum_Zeller_aux:Nnnn \znum_Zeller_Gregorian:nnn }
365     { \_znum_Zeller_aux:Nnnn \znum_Zeller_Julian:nnn }
366     {#1} {#2} {#3}
367   }
368 \cs_new:Npn \_znum_Zeller_aux:Nnnn #1#2#3#4
369 {
370   \int_compare:nNnTF {#3} < \c_three
371     { #1 { #2 - \c_one } { #3 + \c_twelve } {#4} }
372     { #1 {#2} {#3} {#4} }
373   }
374 \cs_new:Npn \znum_two_digits:n #1
375 {
376   \int_compare:nNnT {#1} < \c_ten { 0 }
377   \int_eval:n {#1}
378 }

```

(End definition for `\znum_Zeller:nnn`, `\znum_Zeller_aux:Nnnn`, and `\znum_two_digits:n`.)

`\znum_Zeller_Gregorian:nnn` 格里历(1582年10月15日及以后)的计算公式

$$h / (q \rightarrow \lfloor \frac{26 \leftarrow m \rightarrow 1 \leftarrow}{10} \rfloor \rightarrow Y \rightarrow \lfloor \frac{Y}{4} \rfloor \rightarrow 6 \lfloor \frac{Y}{100} \rfloor \rightarrow \lfloor \frac{Y}{400} \rfloor) \leftarrow \text{mod } 7 \leftarrow$$

其中 Y 为年, m 为月, q 为日;若 $m / 1 \leftarrow 2$, 则令 $m \rightarrow / 12$, 同时 $Y \leftarrow \leftarrow$ 。

```

379 \cs_new:Npn \znum_Zeller_Gregorian:nnn #1#2#3
380 {
381   \int_mod:nn
382     {
383       (#3)
384       + \int_div_truncate:nn { 26 * ( #2 + \c_one ) } \c_ten
385       + (#1)
386       + \int_div_truncate:nn {#1} \c_four
387       + \c_six * \int_div_truncate:nn {#1} \c_one_hundred
388       + \int_div_truncate:nn {#1} { 400 }
389     }
390     { \c_seven }
391 }

```

(End definition for `\znum_Zeller_Gregorian:nnn`.)

`\znum_Zeller_Julian:nnn` 儒略历(1582年10月4日及以前)的计算公式

$$h / (q \rightarrow \lfloor \frac{26 \leftarrow m \rightarrow 1 \leftarrow}{10} \rfloor \rightarrow Y \rightarrow \lfloor \frac{Y}{4} \rfloor \rightarrow 5) \leftarrow \text{mod } 7 \leftarrow$$

```

392 \cs_new:Npn \znum_Zeller_Julian:nnn #1#2#3
393 {

```

¹http://en.wikipedia.org/wiki/Zeller's_congruence

```

394 \int_mod:nn
395 {
396   (#3)
397   + \int_div_truncate:nn { 26 * ( #2 + \c_one ) } \c_ten
398   + (#1)
399   + \int_div_truncate:nn {#1} \c_four
400   + \c_five
401 }
402 { \c_seven }
403 }

```

(End definition for \zhnum_Zeller_Julian:nnn.)

\zhtime 输出时间。

```

404 \cs_new:Npn \zhtime #1
405 { \__zhnum_time:ww #1 \q_stop }
406 \use:x
407 {
408   \cs_new:Npn \exp_not:N \__zhnum_time:ww ##1 \c_colon_str ##2 \exp_not:N \q_stop
409 }
410 {
411   \zhnum_check_time:Nn \zhnum_int:n {#1} \c__zhnum_hour_tl
412   \zhnum_check_time:Nn \zhnum_int:n {#2} \c__zhnum_minute_tl
413 }

```

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

\zhcurrtime 输出当前时间。

```

414 \cs_new_nopar:Npn \zhcurrtime
415 {
416   \zhnum_check_time:Nn \zhnum_int:n
417   { \int_div_truncate:nn \tex_time:D { 60 } } \c__zhnum_hour_tl
418   \zhnum_check_time:Nn \zhnum_int:n
419   { \int_mod:nn \tex_time:D { 60 } } \c__zhnum_minute_tl
420 }

```

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

\zhnum_digit_map:n 阿拉伯数字与中文数字的映射。

```

421 \cs_new:Npn \zhnum_digit_map:n #1
422 { \use:c { c__zhnum_ #1 _tl } }

```

(End definition for \zhnum_digit_map:n.)

\zhnum_scale_map:n 大数系统的映射。

```

\zhnum_scale_map_loop:n 423 \cs_new:Npn \zhnum_scale_map:n #1
424 {
425   \cs_if_exist_use:cF { c__zhnum_s #1 _tl }
426   { \zhnum_scale_map_hook:n {#1} }
427 }
428 \cs_new:Npn \zhnum_scale_map_loop:n #1
429 { \zhnum_scale_map:n { \int_mod:nn {#1} \l__zhnum_scale_int } }
430 \cs_generate_variant:Nn \zhnum_scale_map:n { f }
431 \int_new:N \l__zhnum_scale_int
432 \int_set_eq:NN \l__zhnum_scale_int \c_eleven
433 \cs_new_eq:NN \zhnum_scale_map_hook:n \zhnum_scale_map_loop:n
434 \tl_const:cn { c__zhnum_s0_tl } { }

```

(End definition for \zhnum_scale_map:n and \zhnum_scale_map_loop:n.)

\zhnumExtendScaleMap 扩展进位系统。

```

435 \NewDocumentCommand \zhnumExtendScaleMap { > { \TrimSpaces } +o +m }
436 {
437   \int_zero:N \l_tmpa_int
438   \clist_map_function:nN {#2} \zhnum_set_scale:n
439   \IfNoValueF {#1}
440   { \cs_set:Npn \zhnum_scale_map_hook:n ##1 {#1} }
441 }

```

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

\zhnum_set_scale:n

```
442 \cs_new_protected:Npn \zhnum_set_scale:n #1
443 {
444   \int_incr:N \l_tmpa_int
445   \tl_set:Nx \l_tmpa_tl
446     { c_zhnum_s \int_eval:n { \l_tmpa_int + \c_eleven } _tl }
447   \tl_if_exist:cF { \l_tmpa_tl }
448     { \int_incr:N \l__zhnum_scale_int }
449   \tl_set:cn { \l_tmpa_tl } {#1}
450 }
```

(End definition for \zhnum_set_scale:n.)

\zhnum_ganzhi_normal:nnn

保证干支的参数为正数。

```
451 \cs_new:Npn \zhnum_ganzhi_normal:nnn #1#2#3
452 {
453   \int_compare:nNnF {#1} < \c_one
454     { \cs_if_exist_use:c { c_zhnum_ #2 _ #1 _tl } }
455 }
```

(End definition for \zhnum_ganzhi_normal:nnn.)

\zhnum_ganzhi_cyclic:nnn

对超出范围的数字取模, 参数 0 的结果是空值。

__zhnum_ganzhi_cyclic_mod:nnnn

```
456 \cs_new:Npn \zhnum_ganzhi_cyclic:nnn #1#2#3
457 {
458   \int_compare:nNnF {#1} = \c_zero
459     {
460       \cs_if_exist_use:cF { c__zhnum_ #2 _ #1 _tl }
461         {
462           \__zhnum_ganzhi_cyclic_mod:fnnn
463             { \int_mod:nn {#1} {#3} } {#1} {#2} {#3}
464         }
465       }
466 }
467 \cs_new:Npn \__zhnum_ganzhi_cyclic_mod:nnnn #1#2#3#4
468 {
469   \int_compare:nNnTF {#2} > \c_zero
470     { \use:c { c__zhnum_ #3 _ #1 _tl } }
471     {
472       \int_compare:nNnTF {#1} = \c_zero
473         { \use:c { c__zhnum_ #3 _ 1 _tl } }
474         { \use:c { c__zhnum_ #3 _ \int_eval:n { #1 + #4 + 1 } _tl } }
475       }
476 }
477 \cs_generate_variant:Nn \__zhnum_ganzhi_cyclic_mod:nnnn { f }
```

(End definition for \zhnum_ganzhi_cyclic:nnn.)

\zhnum_ganzhi:nnn

默认不对超出范围的数字取模。

```
478 \cs_new_eq:NN \zhnum_ganzhi:nnn \zhnum_ganzhi_normal:nnn
479 \cs_generate_variant:Nn \zhnum_ganzhi:nnn { f }
```

(End definition for \zhnum_ganzhi:nnn.)

\zhtiangan 天干。

```
480 \cs_new:Npn \zhtiangan #1
481 { \zhnum_ganzhi:fnn { \int_eval:n {#1} } { tiangan } { 10 } }
```

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

\zhdizhi 地支。

```
482 \cs_new:Npn \zhdizhi #1
483 { \zhnum_ganzhi:fnn { \int_eval:n {#1} } { dizhi } { 12 } }
```

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

`\zhganzhi` 干支。

```
484 \cs_new:Npn \zhganzhi #1
485 { \zhnum_ganzhi:fnn { \int_eval:n {#1} } { ganzhi } { 60 } }
```

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

`\zhganzhinian` 干支纪年。

```
486 \cs_new:Npn \zhganzhinian #1
487 { \zhnum_ganzhi_nian:f { \int_eval:n {#1} } }
```

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

`\zhnum_ganzhi_nian:n` 干支纪年。公元元年是 `\zhganzhi{58}`。

```
488 \cs_new:Npn \zhnum_ganzhi_nian:n #1
489 {
490   \int_compare:nNnTF {#1} > \c_zero
491   { \use:c { c__zhnum_ganzhi_ \int_mod:nn { #1 + 57 } { 60 } _tl } }
492   {
493     \int_compare:nNnF {#1} = \c_zero
494     {
495       \use:c
496       {
497         c__zhnum_ganzhi_
498         \int_eval:n { \int_mod:nn { #1 - 2 } { 60 } + 60 }
499         _tl
500       }
501     }
502   }
503 }
504 \cs_generate_variant:Nn \zhnum_ganzhi_nian:n { f }
```

(End definition for `\zhnum_ganzhi_nian:n`)

根据需要设置中文阿拉伯数字。

```
505 \group_begin:
506 \tl_set:Nn \l_tmpa_tl
507 {
508   - .tl_set:N = \l__zhnum_minus_tl ,
509   -0 .tl_set:N = \l__zhnum_null_tl ,
510 }
511 \tl_put_right:Nx \l_tmpa_tl
512 {
513   E2 .tl_set:N = \exp_not:c { l__zhnum_ 100 _tl } ,
514   E3 .tl_set:N = \exp_not:c { l__zhnum_ 1000 _tl } ,
515   FE2 .tl_set:N = \exp_not:c { l__zhnum_financial_ 100 _tl } ,
516   FE3 .tl_set:N = \exp_not:c { l__zhnum_financial_ 1000 _tl } ,
517   D11 .tl_set:N = \exp_not:c { l__zhnum_dizhi_ 11 _tl } ,
518   D12 .tl_set:N = \exp_not:c { l__zhnum_dizhi_ 12 _tl } ,
519   E44 .tl_set:N = \exp_not:c { l__zhnum_ s11 _tl } ,
520 }
521 \int_step_inline:nnnn { 1 } { 1 } { 10 }
522 {
523   \tl_put_right:Nx \l_tmpa_tl
524   {
525     #1 .tl_set:N = \exp_not:c { l__zhnum_ #1 _tl } ,
526     F#1 .tl_set:N = \exp_not:c { l__zhnum_financial_ #1 _tl } ,
527     T#1 .tl_set:N = \exp_not:c { l__zhnum_tiangang_ #1 _tl } ,
528     D#1 .tl_set:N = \exp_not:c { l__zhnum_dizhi_ #1 _tl } ,
529     GZ#1 .tl_set:N = \exp_not:c { l__zhnum_ganzhi_ #1 _tl } ,
530     E \int_eval:n { #1 * 4 }
531     .tl_set:N = \exp_not:c { l__zhnum_ s#1 _tl } ,
532   }
533 }
534 \int_step_inline:nnnn { 11 } { 1 } { 60 }
535 {
536   \tl_put_right:Nx \l_tmpa_tl
537   { GZ#1 .tl_set:N = \exp_not:c { l__zhnum_ganzhi_ #1 _tl } , }
538 }
539 \clist_map_inline:nn { 0 , 100 , 1000 }
```

```

540     {
541       \tl_put_right:Nx \l_tmpa_tl
542       {
543         #1 .tl_set:N = \exp_not:c { l__zhnum_ #1 _tl } ,
544         F#1 .tl_set:N = \exp_not:c { l__zhnum_financial_ #1 _tl } ,
545       }
546     }
547   \clist_map_inline:nn { 20 , 30 , 40 , 200 }
548   {
549     \tl_put_right:Nx \l_tmpa_tl
550     { #1 .tl_set:N = \exp_not:c { l__zhnum_ #1 _tl } , }
551   }
552   \clist_map_inline:nn
553   {
554     dot , and , parts , year , month , day , weekday , hour , minute
555     mon , tue , wed , thu , fri , sat , sun
556   }
557   {
558     \tl_put_right:Nx \l_tmpa_tl
559     { #1 .tl_set:N = \exp_not:c { l__zhnum_ #1 _tl } , }
560   }
561   \use:x
562   {
563     \group_end:
564     \keys_define:nn { zhnum / options } { \exp_not:o \l_tmpa_tl }
565   }

```

将配置文件中的中文数字保存到 prop 变量中。

```

\zhnum_set_digits_map:nn 将配置文件中的中文数字保存到 prop 变量中。
\zhnum_set_digits_map:nn 566 \cs_new_protected:Npn \zhnum_set_digits_map:nn #1#2
\zhnum_set_financial_map:nn 567 { \prop_put:Nnn \l__zhnum_cfg_map_prop {#1} {#2} }
\zhnum_set_financial_map:nn 568 \cs_new_protected:Npn \zhnum_set_digits_map:nn #1#2#3
\zhnum_set_tiangang_map:nn 569 {
\zhnum_set_dizhi_map:nn 570   \prop_put_if_new:Nnn \l__zhnum_cfg_map_prop {#1} {#3}
\l__zhnum_cfg_map_prop 571   \prop_put:Nnn \l__zhnum_cfg_map_var_prop {#1_#2} {#3}
572 }
\l__zhnum_cfg_map_var_prop 573 \cs_new_protected:Npn \zhnum_set_financial_map:nn #1#2
\l__zhnum_cfg_map_finan_prop 574 { \prop_put:Nnn \l__zhnum_cfg_map_finan_prop {#1} {#2} }
\l__zhnum_cfg_map_ganzhi_prop 575 \cs_new_protected:Npn \zhnum_set_financial_map:nn #1#2#3
576 {
577   \prop_put_if_new:Nnn \l__zhnum_cfg_map_finan_prop {#1} {#3}
578   \prop_put:Nnn \l__zhnum_cfg_map_var_prop { financial_#1_#2 } {#3}
579 }
580 \cs_new_protected:Npn \zhnum_set_tiangang_map:nn #1#2
581 { \prop_put:Nnn \l__zhnum_cfg_map_ganzhi_prop { tiangan_#1 } {#2} }
582 \cs_new_protected:Npn \zhnum_set_dizhi_map:nn #1#2
583 { \prop_put:Nnn \l__zhnum_cfg_map_ganzhi_prop { dizhi_#1 } {#2} }
584 \prop_new:N \l__zhnum_cfg_map_prop
585 \prop_new:N \l__zhnum_cfg_map_var_prop
586 \prop_new:N \l__zhnum_cfg_map_finan_prop
587 \prop_new:N \l__zhnum_cfg_map_ganzhi_prop

```

(End definition for \zhnum_set_digits_map:nn and others.)

将 prop 表转化到单独的 tl 变量。

```

\zhnum_parse_config: 将 prop 表转化到单独的 tl 变量。
\zhnum_check_simp:nn 588 \cs_new_protected_nopar:Npn \zhnum_parse_config:
\zhnum_check_financial:nn 589 {
\zhnum_set_zero: 590   \prop_map_function:NN \l__zhnum_cfg_map_prop \zhnum_check_simp:nn
\zhnum_set_week_day: 591   \prop_map_function:NN \l__zhnum_cfg_map_ganzhi_prop \zhnum_assgin_ganzhi:nn
592   \zhnum_set_zero:
593   \zhnum_set_week_day:
594   \bool_if:NF \l__zhnum_reset_bool
595   {
596     \zhnum_assgin_const:
597     \bool_set_true:N \l__zhnum_reset_bool
598   }
599 }
600 \cs_new_protected:Npn \zhnum_check_simp:nn #1#2
601 {
602   \__zhnum_check_simp_aux:nn {#2} {#1}

```

```

603     \prop_get:NnNT \l__zhnum_cfg_map_finan_prop {#1} \l_tmpa_tl
604     { \exp_args:No \__zhnum_check_simp_aux:nn { \l_tmpa_tl } { financial_ #1 } }
605   }
606 \cs_new_protected:Npn \__zhnum_check_simp_aux:nn #1#2
607   {
608     \prop_get:NnNTF \l__zhnum_cfg_map_var_prop { #2 _trad } \l_tmpa_tl
609     {
610       \prop_get:NnNF \l__zhnum_cfg_map_var_prop { #2 _simp } \l_tmpb_tl
611       { \tl_set:Nn \l_tmpb_tl {#1} }
612       \tl_set:cx { l__zhnum_ #2 _tl }
613       {
614         \exp_not:n { \bool_if:NTF \l__zhnum_simp_bool }
615         { \exp_not:o \l_tmpb_tl } { \exp_not:o \l_tmpa_tl }
616       }
617     }
618     { \tl_set:cn { l__zhnum_ #2 _tl } {#1} }
619   }
620 \cs_new_protected_nopar:Npn \zhnum_assgin_const:
621   {
622     \prop_map_function:NN \l__zhnum_cfg_map_prop \zhnum_check_financial:nn
623     \zhnum_set_alias:
624   }
625 \cs_new_protected:Npn \zhnum_check_financial:nn #1#2
626   {
627     \prop_get:NnNTF \l__zhnum_cfg_map_finan_prop {#1} \l_tmpa_tl
628     {
629       \zhnum_assgin_const_tl:cx { c__zhnum_ #1 _tl }
630       {
631         \exp_not:n { \bool_if:NTF \l__zhnum_normal_bool }
632         { \exp_not:c { l__zhnum_ #1 _tl } }
633         { \exp_not:c { l__zhnum_financial_ #1 _tl } }
634       }
635     }
636     {
637       \zhnum_assgin_const_tl:cx
638       { c__zhnum_ #1 _tl } { \exp_not:c { l__zhnum_ #1 _tl } }
639     }
640   }
641 \cs_new_protected_nopar:Npn \zhnum_set_zero:
642   {
643     \tl_set:cx { l__zhnum_0_tl }
644     {
645       \exp_not:n { \bool_if:NTF \l__zhnum_null_bool }
646       { \exp_not:o \l__zhnum_null_tl } { \exp_not:v { l__zhnum_0_tl } }
647     }
648   }
649 \cs_new_protected_nopar:Npn \zhnum_set_week_day:
650   {
651     \tl_set:Nx \l__zhnum_mon_tl
652     { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_1_tl } }
653     \tl_set:Nx \l__zhnum_tue_tl
654     { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_2_tl } }
655     \tl_set:Nx \l__zhnum_wed_tl
656     { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_3_tl } }
657     \tl_set:Nx \l__zhnum_thu_tl
658     { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_4_tl } }
659     \tl_set:Nx \l__zhnum_fri_tl
660     { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_5_tl } }
661     \tl_set:Nx \l__zhnum_sat_tl
662     { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_6_tl } }
663     \tl_set:Nx \l__zhnum_sun_tl
664     { \exp_not:N \c__zhnum_weekday_tl \exp_not:o \l__zhnum_day_tl }
665   }
666 \clist_map_inline:nn { mon , tue , wed , thu , fri , sat , sun }
667   { \tl_const:cx { c__zhnum_ #1 _tl } { \exp_not:c { l__zhnum_ #1 _tl } } }
668 \cs_new_protected:Npn \zhnum_assgin_ganzhi:nn #1#2
669   { \tl_set:cn { l__zhnum_ #1 _tl } {#2} }
670 \cs_new:Npn \zhnum_zero_mod:nn #1#2
671   { \exp_args:Nf \__zhnum_zero_mod_aux:nn { \int_mod:nn {#1} {#2} } {#2} }

```



```

672 \cs_new:Npn \__zhnum_zero_mod_aux:nn #1#2
673 { \int_compare:nNnTF {#1} = \c_zero {#2} {#1} }
674 \int_step_inline:nnnn { 1 } { 1 } { 60 }
675 {
676   \tl_const:cx { c__zhnum_ganzhi_ #1 _tl } { \exp_not:c { l__zhnum_ganzhi_ #1 _tl } }
677   \tl_set:cx { l__zhnum_ganzhi_ #1 _tl }
678   {
679     \exp_not:c { l__zhnum_tiangang_ \zhnum_zero_mod:nn {#1} { 10 } _tl }
680     \exp_not:c { l__zhnum_dizhi_ \zhnum_zero_mod:nn {#1} { 12 } _tl }
681   }
682 }
683 \cs_new_eq:cc { c__zhnum_ganzhi_ 0 _tl } { c__zhnum_ganzhi_ 60 _tl }
684 \cs_new_eq:NN \zhnum_assgin_const_tl:cx \tl_const:cx
685 \AtEndOfPackage
686 {
687   \prop_map_inline:Nn \l__zhnum_cfg_map_ganzhi_prop
688   { \tl_const:cx { c__zhnum_ #1 _tl } { \exp_not:c { l__zhnum_ #1 _tl } } }
689   \cs_new_eq:cc { c__zhnum_tiangang_ 0 _tl } { c__zhnum_tiangang_ 10 _tl }
690   \cs_new_eq:cc { c__zhnum_dizhi_ 0 _tl } { c__zhnum_dizhi_ 12 _tl }
691   \cs_set_eq:NN \zhnum_assgin_const_tl:cx \tl_set:cx
692 }

```

(End definition for \zhnum_parse_config: and others.)

\zhnum_set_alias: 一些易于使用的别名。

```

693 \cs_new_eq:NN \zhnum_set_alias:NN \cs_new_eq:NN
694 \cs_new_protected_nopar:Npx \zhnum_set_alias:
695 {
696   \zhnum_set_alias:NN \exp_not:N \c__zhnum_zero_tl
697   \exp_not:c { c__zhnum_ 0 _tl }
698   \zhnum_set_alias:NN \exp_not:N \c__zhnum_ten_tl
699   \exp_not:c { c__zhnum_ 10 _tl }
700   \zhnum_set_alias:NN \exp_not:N \c__zhnum_hundred_tl
701   \exp_not:c { c__zhnum_ 100 _tl }
702   \zhnum_set_alias:NN \exp_not:N \c__zhnum_thousand_tl
703   \exp_not:c { c__zhnum_ 1000 _tl }
704 }
705 \AtEndOfPackage
706 { \cs_set_eq:NN \zhnum_set_alias:NN \tl_set_eq:NN }

```

(End definition for \zhnum_set_alias:.)

\zhnum_load_cfg:n 根据选定编码载入配置文件。

```

707 \cs_new_protected:Npn \zhnum_load_cfg:n #1
708 {
709   \zhnum_set_cfg_name:Nn \l__zhnum_cfg_str {#1}
710   \str_if_eq:NNF \l__zhnum_cfg_str \l__zhnum_last_cfg_str
711   { \zhnum_update_cfg:n {#1} }
712   \zhnum_parse_config:
713 }
714 \cs_generate_variant:Nn \zhnum_load_cfg:n { o }
715 \cs_new_protected:Npn \zhnum_update_cfg:n #1
716 {
717   \prop_if_exist:cTF { g__zhnum_cfg_ \l__zhnum_cfg_str _prop }
718   { \str_set_eq:NN \l__zhnum_last_cfg_str \l__zhnum_cfg_str }
719   { \zhnum_input_cfg:n {#1} }
720   \__zhnum_update_cfg_prop:N \prop_set_eq:Nc
721 }
722 \cs_new_protected:Npn \zhnum_input_cfg:n #1
723 {
724   \file_if_exist_input:nTF { zhnumber - #1 .cfg }
725   {
726     \bool_set_false:N \l__zhnum_reset_bool
727     \__zhnum_update_cfg_prop:N \__zhnum_prop_initial:Nn
728     \group_begin:
729     \zhnum_set_catcode:
730   }
731   {
732     \msg_error:nxx { zhnumber } { file-not-found } {#1}

```

```

733     \use_none:nnn
734   }
735   \__zhnum_update_cfg_prop:N \__zhnum_prop_gset_eq:Nn
736   \group_end:
737 }
738 \cs_new_protected:Npn \__zhnum_update_cfg_prop:N #1
739 {
740   #1 \l__zhnum_cfg_map_prop      { g__zhnum_cfg_ \l__zhnum_cfg_str _prop }
741   #1 \l__zhnum_cfg_map_var_prop  { g__zhnum_cfg_var_ \l__zhnum_cfg_str _prop }
742   #1 \l__zhnum_cfg_map_finan_prop { g__zhnum_cfg_finan_ \l__zhnum_cfg_str _prop }
743   #1 \l__zhnum_cfg_map_ganzhi_prop { g__zhnum_cfg_ganzhi_ \l__zhnum_cfg_str _prop }
744 }
745 \cs_new_protected:Npn \__zhnum_prop_initial:Nn #1#2
746 {
747   \prop_clear:N #1
748   \prop_new:c {#2}
749 }
750 \cs_new_protected:Npn \__zhnum_prop_gset_eq:Nn #1#2
751 { \prop_gset_eq:cN {#2} #1 }
752 \str_new:N \l__zhnum_cfg_str
753 \str_new:N \l__zhnum_last_cfg_str
754 \bool_new:N \l__zhnum_reset_bool
755 \msg_new:nnnn { zhnumber } { file-not-found }
756 { File~`#1'~not~found. }
757 {
758   The~requested~file~could~not~be~found~in~the~current~directory,~
759   in~the~TeX~search~path~or~in~the~LaTeX~search~path.
760 }

```

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

`\zhnum_if_unicode_engine_p:` 使用 `upTeX` 的时候,也不必将汉字的首字符设置为活动字符。判断 `~~~~0021` 是否为单个记号的办法对 `upTeX` 不适用。
`\zhnum_if_unicode_engine:TF`

```

761 \bool_if:nTF
762 {
763   \sys_if_engine_xetex_p: ||
764   \sys_if_engine_luatex_p: ||
765   \sys_if_engine_uptex_p:
766 }
767 {
768   \cs_new_eq:NN \zhnum_if_unicode_engine_p: \c_true_bool
769   \cs_new_eq:NN \zhnum_if_unicode_engine:TF \use_i:nn
770 }
771 {
772   \cs_new_eq:NN \zhnum_if_unicode_engine_p: \c_false_bool
773   \cs_new_eq:NN \zhnum_if_unicode_engine:TF \use_ii:nn
774 }

```

(End definition for `\zhnum_if_unicode_engine:TF`.)

`\zhnum_set_catcode:` 设置与恢复配置文件前后的 `catcode`。`pdfLATEX` 需要将汉字的首字节设置为活动字符。

```

\zhnum_set_cfg_name:Nn 775 \if_predicate:w \zhnum_if_unicode_engine_p:
\zhnum_reset_config: 776 \cs_new_eq:NN \zhnum_set_catcode: \prg_do_nothing:
777 \cs_new_protected:Npn \zhnum_set_cfg_name:Nn #1#2
778 {
779   \str_set:Nx \l__zhnum_encoding_str {#2}
780   \str_set_eq:NN #1 \l__zhnum_encoding_str
781 }
782 \cs_new_eq:NN \zhnum_reset_config: \zhnum_parse_config:
783 \else:
784 \cs_new_protected_nopar:Npn \zhnum_set_catcode:
785 { \bool_if:NT \l__zhnum_active_char_bool { \zhnum_set_active: } }
786 \cs_new_protected_nopar:Npn \zhnum_set_active:
787 {
788   \str_case:onTF { \l__zhnum_encoding_str }
789   {
790     { gbk } { \int_set:Nn \l__zhnum_byte_min_int { "81 } }
791     { big5 } { \int_set:Nn \l__zhnum_byte_min_int { "A1 } }
792   }

```

```

793     { \int_set:Nn \l__zhnum_byte_max_int { "FE } }
794     {
795     \int_set:Nn \l__zhnum_byte_min_int { "E0 }
796     \int_set:Nn \l__zhnum_byte_max_int { "EF }
797     }
798     \int_step_function:nnnN
799     { \l__zhnum_byte_min_int } { \c_one }
800     { \l__zhnum_byte_max_int } \char_set_catcode_active:n
801     }
802     \int_new:N \l__zhnum_byte_min_int
803     \int_new:N \l__zhnum_byte_max_int
804     \cs_new_protected:Npn \zhnum_set_cfg_name:Nn #1#2
805     {
806     \str_set:Nx \l__zhnum_encoding_str {#2}
807     \str_set:Nx #1
808     {
809     \l__zhnum_encoding_str
810     \bool_if:NT \l__zhnum_active_char_bool { _active }
811     }
812     }
813     \cs_new_protected_nopar:Npn \zhnum_reset_config:
814     { \zhnum_load_cfg:o { \l__zhnum_encoding_str } }
815     \bool_new:N \l__zhnum_active_char_bool
816     \bool_set_true:N \l__zhnum_active_char_bool
817     \fi:

```

(End definition for \zhnum_set_catcode:, \zhnum_set_cfg_name:Nn, and \zhnum_reset_config:.)

encoding 宏包设置选项。

```

style 818 \keys_define:nn { zhnum / options }
null 819 {
reset 820     encoding .choices:nn =
821     { UTF8 , GBK , Big5 }
822     {
823     \str_set:Nx \l__zhnum_encoding_str
824     { \str_fold_case:V \l_keys_choice_tl }
825     \zhnum_load_cfg:o { \l__zhnum_encoding_str }
826     } ,
827     encoding .default:n = { GBK } ,
828     encoding / Bg5 .meta:n = { encoding = Big5 } ,
829     encoding / unknown .code:n =
830     { \msg_error:nnn { zhnumber } { encoding-invalid } {#1} } ,
831     style .multichoice: ,
832     style / Normal .code:n =
833     {
834     \bool_set_false:N \l__zhnum_ancient_bool
835     \bool_set_true:N \l__zhnum_normal_bool
836     } ,
837     style / Financial .code:n =
838     {
839     \bool_set_false:N \l__zhnum_ancient_bool
840     \bool_set_false:N \l__zhnum_normal_bool
841     } ,
842     style / Ancient .code:n =
843     {
844     \bool_set_true:N \l__zhnum_ancient_bool
845     \bool_set_true:N \l__zhnum_normal_bool
846     } ,
847     style / Simplified .code:n = { \bool_set_true:N \l__zhnum_simp_bool } ,
848     style / Traditional .code:n = { \bool_set_false:N \l__zhnum_simp_bool } ,
849     style .default:n = { Normal , Simplified } ,
850     null .bool_set:N = \l__zhnum_null_bool ,
851     time .choice: ,
852     time / Chinese .code:n = { \bool_set_true:N \l__zhnum_time_bool } ,
853     time / Arabic .code:n = { \bool_set_false:N \l__zhnum_time_bool } ,
854     time .default:n = { Arabic } ,
855     reset .code:n = { \zhnum_reset_config: } ,
856     activechar .bool_set:N = \l__zhnum_active_char_bool ,
857     ganzhi-cyclic .choice: ,

```

```

858     ganzhi-cyclic / true .code:n =
859     { \cs_set_eq:NN \zhnum_ganzhi:nnn \zhnum_ganzhi_cyclic:nnn } ,
860     ganzhi-cyclic / false.code:n =
861     { \cs_set_eq:NN \zhnum_ganzhi:nnn \zhnum_ganzhi_normal:nnn } ,
862     ganzhi-cyclic .default:n = { true } ,
863   }
864   \str_new:N \l__zhnum_encoding_str
865   \msg_new:nnnn { zhnumber } { encoding-invalid }
866   { The~encoding~`#1'~is~invalid. }
867   { Available~encodings~are~`UTF8',~`GBK'~and~`Big5'. }

```

(End definition for encoding and others. These functions are documented on page 1.)

`\zhnumsetup` 在文档中设置 zhnumber 的接口。

```

868   \NewDocumentCommand \zhnumsetup { +m }
869   {
870     \keys_set:nn { zhnum / options } {#1}
871     \tex_ignorespaces:D
872   }

```

(End definition for \zhnumsetup. This function is documented on page 3.)

初始化设置和执行宏包选项。

```

873   \keys_set:nn { zhnum / options } { style , time }
874   \ProcessKeysOptions { zhnum / options }
875   如果没有选定编码,则根据引擎自动设置编码。
876   \str_if_empty:NT \l__zhnum_encoding_str
877   {
878     \zhnum_if_unicode_engine:TF
879     { \keys_set:nn { zhnum / options } { encoding = UTF8 } }
880     { \keys_set:nn { zhnum / options } { encoding = GBK } }
881   }
882   <package>

```

4 中文数字配置文件

```

882   <*config>
883   <!*big5>
884   \zhnum_set_digits_map:nnn { minus } { simp } { 负 }
885   \zhnum_set_digits_map:nnn { minus } { trad } { 負 }
886   <!big5>
887   <*big5>
888   \zhnum_set_digits_map:nn { minus } { 負 }
889   <big5>
890   \zhnum_set_digits_map:nn { 0 } { 零 }
891   <!*big5>
892   \zhnum_set_digits_map:nn { null } { ○ }
893   <!big5>
894   <*big5>
895   \zhnum_set_digits_map:nn { null } { ○ }
896   <big5>
897   \zhnum_set_digits_map:nn { 1 } { 一 }
898   \zhnum_set_digits_map:nn { 2 } { 二 }
899   \zhnum_set_digits_map:nn { 3 } { 三 }
900   \zhnum_set_digits_map:nn { 4 } { 四 }
901   \zhnum_set_digits_map:nn { 5 } { 五 }
902   \zhnum_set_digits_map:nn { 6 } { 六 }
903   \zhnum_set_digits_map:nn { 7 } { 七 }
904   \zhnum_set_digits_map:nn { 8 } { 八 }
905   \zhnum_set_digits_map:nn { 9 } { 九 }
906   \zhnum_set_digits_map:nn { 10 } { 十 }
907   \zhnum_set_digits_map:nn { 100 } { 百 }
908   \zhnum_set_digits_map:nn { 1000 } { 千 }
909   \zhnum_set_digits_map:nn { 20 } { 廿 }
910   \zhnum_set_digits_map:nn { 30 } { 卅 }
911   \zhnum_set_digits_map:nn { 40 } { 卌 }
912   \zhnum_set_digits_map:nn { 200 } { 佰 }

```

913 $*\!big5$
914 \zhnum_set_digits_map:nnn { dot } { simp } { 点 }
915 \zhnum_set_digits_map:nnn { dot } { trad } { 點 }
916 $\!big5$
917 $*\!big5$
918 \zhnum_set_digits_map:nn { dot } { 點 }
919 $\!big5$
920 \zhnum_set_digits_map:nn { and } { 又 }
921 \zhnum_set_digits_map:nn { parts } { 分之 }
922 $*\!big5$
923 \zhnum_set_digits_map:nnn { s1 } { simp } { 万 }
924 \zhnum_set_digits_map:nnn { s1 } { trad } { 萬 }
925 \zhnum_set_digits_map:nnn { s2 } { simp } { 亿 }
926 \zhnum_set_digits_map:nnn { s2 } { trad } { 億 }
927 $\!big5$
928 $*\!big5$
929 \zhnum_set_digits_map:nn { s1 } { 萬 }
930 \zhnum_set_digits_map:nn { s2 } { 億 }
931 $\!big5$
932 \zhnum_set_digits_map:nn { s3 } { 兆 }
933 \zhnum_set_digits_map:nn { s4 } { 京 }
934 \zhnum_set_digits_map:nn { s5 } { 垓 }
935 \zhnum_set_digits_map:nn { s6 } { 秭 }
936 \zhnum_set_digits_map:nn { s7 } { 穰 }
937 $*\!big5$
938 \zhnum_set_digits_map:nnn { s8 } { simp } { 沟 }
939 \zhnum_set_digits_map:nnn { s8 } { trad } { 溝 }
940 \zhnum_set_digits_map:nnn { s9 } { simp } { 涧 }
941 \zhnum_set_digits_map:nnn { s9 } { trad } { 澗 }
942 $\!big5$
943 $*\!big5$
944 \zhnum_set_digits_map:nn { s8 } { 溝 }
945 \zhnum_set_digits_map:nn { s9 } { 澗 }
946 $\!big5$
947 \zhnum_set_digits_map:nn { s10 } { 正 }
948 $*\!big5$
949 \zhnum_set_digits_map:nnn { s11 } { simp } { 载 }
950 \zhnum_set_digits_map:nnn { s11 } { trad } { 載 }
951 $\!big5$
952 $*\!big5$
953 \zhnum_set_digits_map:nn { s11 } { 載 }
954 $\!big5$
955 \zhnum_set_digits_map:nn { year } { 年 }
956 \zhnum_set_digits_map:nn { month } { 月 }
957 \zhnum_set_digits_map:nn { day } { 日 }
958 $*\!big5$
959 \zhnum_set_digits_map:nnn { hour } { simp } { 时 }
960 \zhnum_set_digits_map:nnn { hour } { trad } { 時 }
961 $\!big5$
962 $*\!big5$
963 \zhnum_set_digits_map:nn { hour } { 時 }
964 $\!big5$
965 \zhnum_set_digits_map:nn { minute } { 分 }
966 \zhnum_set_digits_map:nn { weekday } { 星期 }
967 \zhnum_set_financial_map:nn { null } { 零 }
968 \zhnum_set_financial_map:nn { 0 } { 零 }
969 \zhnum_set_financial_map:nn { 1 } { 壹 }
970 \zhnum_set_financial_map:nn { 2 } { 貳 }
971 $*\!big5$
972 \zhnum_set_financial_map:nnn { 3 } { simp } { 叁 }
973 \zhnum_set_financial_map:nnn { 3 } { trad } { 叁 }
974 $\!big5$
975 $*\!big5$
976 \zhnum_set_financial_map:nn { 3 } { 參 }
977 $\!big5$
978 \zhnum_set_financial_map:nn { 4 } { 肆 }
979 \zhnum_set_financial_map:nn { 5 } { 伍 }
980 $*\!big5$
981 \zhnum_set_financial_map:nnn { 6 } { simp } { 陆 }

```

982 \zhnum_set_financial_map:nn { 6 } { trad } { 陸 }
983 <!big5>
984 <*big5>
985 \zhnum_set_financial_map:nn { 6 } { 陸 }
986 <<big5>
987 \zhnum_set_financial_map:nn { 7 } { 柒 }
988 \zhnum_set_financial_map:nn { 8 } { 捌 }
989 \zhnum_set_financial_map:nn { 9 } { 玖 }
990 \zhnum_set_financial_map:nn { 10 } { 拾 }
991 \zhnum_set_financial_map:nn { 100 } { 佰 }
992 \zhnum_set_financial_map:nn { 1000 } { 仟 }
993 \zhnum_set_tiangang_map:nn { 1 } { 甲 }
994 \zhnum_set_tiangang_map:nn { 2 } { 乙 }
995 \zhnum_set_tiangang_map:nn { 3 } { 丙 }
996 \zhnum_set_tiangang_map:nn { 4 } { 丁 }
997 \zhnum_set_tiangang_map:nn { 5 } { 戊 }
998 \zhnum_set_tiangang_map:nn { 6 } { 己 }
999 \zhnum_set_tiangang_map:nn { 7 } { 庚 }
1000 \zhnum_set_tiangang_map:nn { 8 } { 辛 }
1001 \zhnum_set_tiangang_map:nn { 9 } { 壬 }
1002 \zhnum_set_tiangang_map:nn { 10 } { 癸 }
1003 \zhnum_set_dizhi_map:nn { 1 } { 子 }
1004 \zhnum_set_dizhi_map:nn { 2 } { 丑 }
1005 \zhnum_set_dizhi_map:nn { 3 } { 寅 }
1006 \zhnum_set_dizhi_map:nn { 4 } { 卯 }
1007 \zhnum_set_dizhi_map:nn { 5 } { 辰 }
1008 \zhnum_set_dizhi_map:nn { 6 } { 巳 }
1009 \zhnum_set_dizhi_map:nn { 7 } { 午 }
1010 \zhnum_set_dizhi_map:nn { 8 } { 未 }
1011 \zhnum_set_dizhi_map:nn { 9 } { 申 }
1012 \zhnum_set_dizhi_map:nn { 10 } { 酉 }
1013 \zhnum_set_dizhi_map:nn { 11 } { 戌 }
1014 \zhnum_set_dizhi_map:nn { 12 } { 亥 }
1015 <<config>

```

5 代码索引

斜体的数字表示对应项说明所在的页码,下划线的数字表示定义所在的代码行号,而直立的数字表示对应项使用时所在的行号。

Symbols	E
\\ 5, 6, 7	eleven commands:
A	\c_eleven 432, 446
activechar 4	else commands:
\AtEndOfPackage 685, 705	\else: 126, 138, 169, 291, 320, 783
B	encoding 1, <u>19</u>
bingint commands:	exp commands:
_bingint_read_do:nn 6	\exp:w 117, 125, 128, 137, 149, 283, 290, 307
bool commands:	\exp_after:wN 114, 116, 124, 125, 127, 129, 130, 136, 137, 139, 140, 148, 168, 170, 190, 193, 194, 195, 280, 282, 289, 290, 292, 293, 294, 295, 304, 306
\bool_if:Nf 213, 226, 594	\exp_args:Nf 174, 176, 671
\bool_if:nf 239	\exp_args:No 604
\bool_if:NT 785, 810	\exp_end_continue_f:w 117, 125, 128, 137, 149, 283, 290, 307
\bool_if:NTF 345, 614, 631, 645	\exp_not:c 513, 514, 515, 516, 517, 518, 519, 525, 526, 527, 528, 529, 531, 537, 543, 544, 550, 559, 632, 633, 638, 667, 676, 679, 680, 688, 697, 699, 701, 703
\bool_if:nTF 231, 245, 761	\exp_not:N 123, 135, 167, 288, 304, 408, 652, 654, 656, 658, 660, 662, 664, 696, 698, 700, 702
\bool_new:N 754, 815	\exp_not:n 614, 631, 645
\bool_set_false:N 726, 834, 839, 840, 848, 853	\exp_not:o 564, 615, 646, 664
\bool_set_true:N 597, 816, 835, 844, 845, 847, 852	\exp_not:v 646, 652, 654, 656, 658, 660, 662
\BooleanFalse 276	\exp_stop_f: 118, 167, 192, 197, 284, 350
\BooleanTrue 274	F
C	false commands:
char commands:	\c_false_bool 218, 219, 772
\char_set_catcode_active:n 800	fi commands:
clist commands:	\fi: ... 123, 131, 141, 171, 196, 288, 296, 304, 322, 358, 817
\clist_map_function:nN 438	file commands:
\clist_map_inline:nn 539, 547, 552, 666	\file_if_exist_input:nTF 724
colon commands:	five commands:
\c_colon_str 408	\c_five 400
cs commands:	four commands:
\cs:w 316	\c_four 176, 183, 191, 250, 386, 399
\cs_end: 324	G
\cs_generate_variant:Nn 35, 111, 187, 206, 222, 277, 430, 477, 479, 504, 714	ganzhi-cyclic 3
\cs_if_exist_use:c 454	group commands:
\cs_if_exist_use:cF 425, 460	\group_begin: 22, 83, 268, 505, 728
\cs_new:Npn 27, 29, 36, 42, 61, 68, 88, 94, 98, 112, 121, 133, 143, 151, 153, 155, 165, 173, 175, 177, 188, 199, 201, 207, 223, 278, 286, 298, 309, 314, 332, 344, 346, 348, 360, 368, 374, 379, 392, 404, 408, 421, 423, 428, 451, 456, 467, 480, 482, 484, 486, 488, 670, 672	\group_end: 25, 86, 271, 563, 736
\cs_new_eq:cc 683, 689, 690	I
\cs_new_eq:NN 433, 478, 684, 693, 768, 769, 772, 773, 776, 782	if commands:
\cs_new_nopar:Npn 273, 275, 338, 414	\if:w 123, 135, 288, 304
\cs_new_protected:Npn 442, 566, 568, 573, 575, 580, 582, 600, 606, 625, 668, 707, 715, 722, 738, 745, 750, 777, 804	\if_case:w 192, 350
\cs_new_protected_nopar:Npn 588, 620, 641, 649, 784, 786, 813	\if_int_compare:w 167, 318
\cs_new_protected_nopar:Npx 694	\if_predicate:w 775
\cs_set:Npn 440	\IfBooleanT 329
\cs_set_eq:NN 691, 706, 859, 861	\IfBooleanTF 319
D	\IfNoValueF 439
\DeclareExpandableDocumentCommand 13, 74, 259, 326	\IfNoValueTF 15, 76, 261
	int commands:
	\int_compare:nNfF . 160, 229, 237, 257, 311, 453, 458, 493
	\int_compare:nNtT 376
	\int_compare:nNtF 100, 103, 157, 179, 182, 210, 216, 225, 228, 236, 362, 370, 469, 472, 490, 673

<code>\int_compare:nNn</code>	232, 241, 242, 248, 249, 250	<code>\prop_set_eq:Nc</code>	720
<code>\int_div_truncate:nn</code>	191, 384, 386, 387, 388, 397, 399, 417	Q	
<code>\int_eval:n</code>	101, 106, 154, 183, 220, 377, 446, 474, 481, 483, 485, 487, 498, 530	quark commands:	
<code>\int_if_exist:cTF</code>	90	<code>\quark_if_nil:nTF</code>	31, 38, 44
<code>\int_incr:N</code>	444, 448	<code>\quark_if_recursion_tail_stop:N</code>	209, 303
<code>\int_mod:nn</code>	176, 381, 394, 419, 429, 463, 491, 498, 671	<code>\quark_if_recursion_tail_stop_do:Nn</code>	147
<code>\int_new:N</code>	431, 802, 803	R	
<code>\int_set:Nn</code>	790, 791, 793, 795, 796	recursion commands:	
<code>\int_set_eq:NN</code>	432	<code>\q_recursion_stop</code>	118, 143, 149, 204, 284
<code>\int_step_function:nnnN</code>	798	<code>\q_recursion_tail</code>	8, 118, 204, 284
<code>\int_step_inline:nnnn</code>	521, 534, 674	<code>\RequirePackage</code>	12
<code>\int_to_arabic:n</code>	345	reset	4, <u>19</u>
<code>\int_zero:N</code>	437	S	
K			
keys commands:		seven commands:	
<code>\l_keys_choice_tl</code>	824	<code>\c_seven</code>	390, 402
<code>\keys_define:nn</code>	564, 818	six commands:	
<code>\keys_set:nn</code>	23, 84, 269, 870, 873, 878, 879	<code>\c_six</code>	387
M			
mark commands:		stop commands:	
<code>\q_mark</code>	40, 42	<code>\q_stop</code> 28, 29, 32, 36, 40, 42, 328, 330, 332, 347, 348, 405, 408	
msg commands:		str commands:	
<code>\msg_error:nn</code>	11	<code>\str_case:onTF</code>	788
<code>\msg_error:nnn</code>	830	<code>\str_fold_case:V</code>	824
<code>\msg_error:nxx</code>	732	<code>\str_if_empty:NT</code>	875
<code>\msg_expandable_error:nnn</code>	95	<code>\str_if_eq:NNF</code>	710
<code>\msg_new:nnn</code>	3, 96	<code>\str_new:N</code>	752, 753, 864
<code>\msg_new:nnnn</code>	755, 865	<code>\str_set:Nx</code>	779, 806, 807, 823
N			
<code>\NewDocumentCommand</code>	20, 81, 266, 435, 868	<code>\str_set_eq:NN</code>	718, 780
nil commands:		style	3, <u>19</u>
<code>\q_nil</code>	8, 28, 32, 40, 204	sys commands:	
nine commands:		<code>\sys_if_engine luatex_p:</code>	764
<code>\c_nine</code>	167	<code>\sys_if_engine uptex_p:</code>	765
null	3, <u>19</u>	<code>\sys_if_engine xetex_p:</code>	763
O			
one commands:		T	
<code>\c_one</code>	146, 160, 183, 220, 241, 311, 371, 384, 397, 453, 799	ten commands:	
<code>\c_one_hundred</code>	387	<code>\c_ten</code>	376, 384, 397
or commands:		\TeX and \LaTeX 2 ϵ commands:	
<code>\or:</code>	193, 194, 195, 352, 353, 354, 355, 356, 357	<code>\@ifpackagelater</code>	10
P			
prg commands:		<code>\tiangan</code>	3
<code>\prg_do_nothing:</code>	776	<code>\zhdate</code>	2, 2
<code>\ProcessKeysOptions</code>	874	<code>\zhdigits</code>	1, 1, 1, 1, 2, 3, 4
prop commands:		<code>\zhdizhi</code>	2
<code>\prop_clear:N</code>	747	<code>\zhganzhi</code>	2
<code>\prop_get:NnNF</code>	610	<code>\zhganzhinian</code>	2
<code>\prop_get:NnNT</code>	603	<code>\zhnum</code>	1, 1, 4
<code>\prop_get:NnNTF</code>	608, 627	<code>\zhnumber</code>	1, 1, 2, 2, 4
<code>\prop_gset_eq:cN</code>	751	<code>\zhnumExtendScaleMap</code>	2, 2
<code>\prop_if_exist:cTF</code>	717	<code>\zhnumsetup</code>	1, 3, 4
<code>\prop_map_function:NN</code>	590, 591, 622	<code>\zhtiangan</code>	2
<code>\prop_map_inline:Nn</code>	687	<code>\zhtime</code>	2
<code>\prop_new:c</code>	748	<code>\zhweekday</code>	2
<code>\prop_new:N</code>	584, 585, 586, 587	tex commands:	
<code>\prop_put:Nnn</code>	567, 571, 574, 578, 581, 583	<code>\tex_day:D</code>	342
<code>\prop_put_if_new:Nnn</code>	570, 577	<code>\tex_ignorespaces:D</code>	871
		<code>\tex_month:D</code>	341
		<code>\tex_number:D</code>	115, 191, 281
		<code>\tex_time:D</code>	417, 419
		<code>\tex_year:D</code>	340
		three commands:	
		<code>\c_three</code>	249, 370
		time	3

tl commands:	
_tl_act:NNNnn	7
_tl_const:cn	434
_tl_const:cx	667, 676, 684, 688
_tl_count:n	174
_tl_if_blank:nF	51
_tl_if_blank:nTF	64, 70
_tl_if_exist:cF	447
_tl_put_right:Nx	511, 523, 536, 541, 549, 558
_tl_set:cn	449, 618, 669
_tl_set:cx	612, 643, 677, 691
_tl_set:Nn	506, 611
_tl_set:Nx	445, 651, 653, 655, 657, 659, 661, 663
_tl_set_eq:NN	706
tmpa commands:	
_l_tmpa_int	437, 444, 446
_l_tmpa_tl	445, 447, 449, 506, 511, 523, 536, 541, 549, 558, 564, 603, 604, 608, 615, 627
tmpb commands:	
_l_tmpb_tl	610, 611, 615
\TrimSpaces	435
true commands:	
_c_true_bool	203, 218, 219, 768
twelve commands:	
_c_twelve	371
two commands:	
_c_two	179, 232, 248

U

use commands:	
_use:c	422, 470, 473, 474, 491, 495
_use:n	117, 125, 137, 149, 193, 283, 290, 307
_use:x	406, 561
_use_i:nn	168, 211, 769
_use_i:nnn	195
_use_i_ii:nnn	194
_use_ii:nn	170, 773
_use_none:n	252
_use_none:nnn	733

Z

zero commands:	
_c_zero	100, 103, 119, 157, 182, 210, 216, 225, 228, 229, 236, 237, 242, 257, 318, 458, 469, 472, 490, 493, 673
\zhcurrttime	2, 12, 414
\zhdate	2, 10, 326
\zhdigits	1, 4, 9, 259
\zhdigitswithoptions	9, 263, 266
\zhdizhi	2, 13, 482
\zhganzhi	2, 14, 484
\zhganzhinian	2, 14, 486
\zhnum	1, 4, 5, 74
zhnum commands:	
_l_zhnum_active_char_bool	785, 810, 815, 816, 856
_l_zhnum_ancient_bool	232, 247, 834, 839, 844
_c_zhnum_and_tl	54
_zhnum_assgin_const:	596, 620
_zhnum_assgin_const_tl:cx	629, 637, 684, 691
_zhnum_assgin_ganzhi:nn	591, 668
_zhnum_blank_to_zero:n	5, 46, 48, 56, 58, 63, 68
_l_zhnum_byte_max_int	793, 796, 800, 803
_l_zhnum_byte_min_int	790, 791, 795, 799, 802

_l_zhnum_cfg_map_finan_prop	15, 574, 577, 586, 603, 627, 742
_l_zhnum_cfg_map_ganzhi_prop	15, 581, 583, 587, 591, 687, 743
_l_zhnum_cfg_map_prop	15, 567, 570, 584, 590, 622, 740
_l_zhnum_cfg_map_var_prop	15, 571, 578, 585, 608, 610, 741
_l_zhnum_cfg_str	709, 710, 717, 718, 740, 741, 742, 743, 752
_zhnum_check_financial:nn	15, 622, 625
_zhnum_check_simp:nn	15, 590, 600
_zhnum_check_simp_aux:nn	602, 604, 606
_zhnum_check_time:Nn	10, 334, 335, 336, 340, 341, 342, 344, 411, 412, 416, 418
_zhnum_counter:n	6, 77, 85, 88
_zhnum_counter_error:n	92, 94
_zhnum_date:www	328, 332
_c_zhnum_day_tl	336, 342
_l_zhnum_day_tl	664
_zhnum_decimal:nn	5, 33, 61
_zhnum_digit_map:n	12, 180, 227, 233, 234, 252, 253, 257, 421
_zhnum_digits:Nn	9, 262, 270, 274, 276, 278
_zhnum_digits_null:n	9, 275, 277, 334
_zhnum_digits_null:V	340
_zhnum_digits_zero:n	9, 66, 273
_c_zhnum_dot_tl	63, 304
_l_zhnum_encoding_str	779, 780, 788, 806, 809, 814, 823, 825, 864, 875
_zhnum_fraction:www	5, 40, 42
_c_zhnum_fri_tl	357
_l_zhnum_fri_tl	659
_zhnum_ganzhi:fnn	481, 483, 485
_zhnum_ganzhi:nnn	13, 478, 479, 859, 861
_zhnum_ganzhi_cyclic:nnn	13, 456, 859
_zhnum_ganzhi_cyclic_mod:fnnn	462
_zhnum_ganzhi_cyclic_mod:nnnn	13, 467, 477
_zhnum_ganzhi_nian:f	487
_zhnum_ganzhi_nian:n	14, 488, 504
_zhnum_ganzhi_normal:nnn	13, 451, 478, 861
_c_zhnum_hour_tl	411, 417
_c_zhnum_hundred_tl	234, 700
_zhnum_if_digit:NTF	7, 145, 165, 300
_zhnum_if_unicode_engine:TF	18, 769, 773, 877
_zhnum_if_unicode_engine_p:	18, 768, 772, 775
_zhnum_input_cfg:n	719, 722
_zhnum_int:c	91
_zhnum_int:n	6, 98, 111, 335, 336, 341, 342, 411, 412, 416, 418
_zhnum_integer:n	6, 9, 39, 112
_zhnum_integer_or_fraction:www	5, 32, 36
_l_zhnum_last_cfg_str	710, 718, 753
_zhnum_load_cfg:n	17, 707, 714
_zhnum_load_cfg:o	814, 825
_zhnum_loop_end:wnn	147, 153
_c_zhnum_minus_tl	105, 161, 312
_l_zhnum_minus_tl	508
_c_zhnum_minute_tl	412, 419
_c_zhnum_mon_tl	353
_l_zhnum_mon_tl	651
_c_zhnum_month_tl	335, 341
_l_zhnum_normal_bool	631, 835, 840, 845
_l_zhnum_null_bool	645, 850
_l_zhnum_null_tl	509, 646

\zhnum_number:f	16, 24	\zhnum_set_scale:n	<u>13</u> , 438, 442
\zhnum_number:n	<u>4</u> , 27, 35, 53, 72	\zhnum_set_tiangang_map:nn	<u>15</u> , 580, 993, 994, 995, 996, 997, 998, 999, 1000, 1001, 1002
_zhnum_number:www	<u>4</u> , 28, 29	\zhnum_set_week_day:	<u>15</u> , 593, 649
_zhnum_output:nnwnn	146, 151	\zhnum_set_zero:	<u>15</u> , 592, 641
_zhnum_output_digits:NN	301, 314	\l_zhnum_simp_bool	614, 847, 848
\zhnum_parse_config:	<u>15</u> , 588, 712, 782	\zhnum_split_number:fn	183
\zhnum_parse_number:f	101, 106	\zhnum_split_number:nn	<u>8</u> , 200, 201, 206
\zhnum_parse_number:n	<u>7</u> , 173, 187	\zhnum_split_number:NNfNNNNw	218, 219
\zhnum_parse_number:nn	<u>7</u> , 162, 174, 175	\zhnum_split_number:NNnNNNNw	<u>8</u> , 203, 207, 222
_zhnum_parse_number:nnn	176, 177	_zhnum_split_number_aux:nnn	<u>8</u> , 184, 188
\c_zhnum_parts_tl	47, 57	_zhnum_split_number_aux:wnn	190, 199
\zhnum_process_number:NNNNNN	<u>8</u> , 214, 223	\c_zhnum_sun_tl	352
_zhnum_prop_gset_eq:Nn	735, 750	\l_zhnum_sun_tl	663
_zhnum_prop_initial:Nn	727, 745	\c_zhnum_ten_tl	255, 698
_zhnum_read_abs_loop:Nw	<u>7</u> , 139, 143, 148	\c_zhnum_thousand_tl	227, 702
_zhnum_read_digits:w	280, 309	\c_zhnum_thu_tl	356
_zhnum_read_digits_loop:NN	293, 298, 306	\l_zhnum_thu_tl	657
_zhnum_read_integer:www	<u>7</u> , 114, 155	_zhnum_time:ww	405, 408
_zhnum_read_sign_loop:N	116, 121, 124	\l_zhnum_time_bool	345, 852, 853
_zhnum_read_sign_loop:NN	282, 286, 289	\c_zhnum_tue_tl	354
_zhnum_read_zeros_loop:N	129, 133, 136	\l_zhnum_tue_tl	653
\l_zhnum_reset_bool	594, 597, 726, 754	\zhnum_two_digits:n	<u>11</u> , 363, 374
\zhnum_reset_config:	<u>18</u> , 782, 813, 855	\zhnum_update_cfg:n	711, 715
_zhnum_result:nn	119, 151, 152, 153	_zhnum_update_cfg_prop:N	720, 727, 735, 738
\c_zhnum_sat_tl	351	\c_zhnum_wed_tl	355
\l_zhnum_sat_tl	661	\l_zhnum_wed_tl	655
\l_zhnum_scale_int	429, 431, 432, 448	_zhnum_week_day:www	<u>11</u> , 330, 347, 348
\zhnum_scale_map:n	<u>12</u> , 215, 423, 429, 430	\c_zhnum_weekday_tl	652, 654, 656, 658, 660, 662, 664
\zhnum_scale_map_hook:n	426, 433, 440	\c_zhnum_year_tl	334, 340
\zhnum_scale_map_loop:n	<u>12</u> , 428, 433	\zhnum_Zeller:nnn	<u>11</u> , 350, 360
\zhnum_set_active:	785, 786	_zhnum_Zeller_aux:Nnnn	364, 365, 368
\zhnum_set_alias:	<u>17</u> , 623, 694	\zhnum_Zeller_aux:Nnnn	<u>11</u>
\zhnum_set_alias:NN	693, 696, 698, 700, 702, 706	\zhnum_Zeller_Gregorian:nnn	<u>11</u> , 364, 379
\zhnum_set_catcode:	<u>18</u> , 729, 776, 784	\zhnum_Zeller_Julian:nnn	<u>11</u> , 365, 392
\zhnum_set_cfg_name:Nn	<u>18</u> , 709, 777, 804	\zhnum_zero_mod:nn	670, 679, 680
\zhnum_set_digits_map:nn		_zhnum_zero_mod_aux:nn	671, 672
	<u>15</u> , 566, 888, 890, 892, 895, 897, 898,	\c_zhnum_zero_tl	65, 71, 108, 158, 213, 226, 229, 237, 696
	899, 900, 901, 902, 903, 904, 905, 906, 907, 908, 909,	\zhnumber	1, 4, <u>4</u> , 13
	910, 911, 912, 918, 920, 921, 929, 930, 932, 933, 934,	\zhnumberwithoptions	<u>4</u> , <u>5</u> , 17, 20
	935, 936, 944, 945, 947, 953, 955, 956, 957, 963, 965, 966	\zhnumExtendScaleMap	2, <u>12</u> , 435
\zhnum_set_digits_map:nnn	<u>15</u> , 568, 884, 885, 914, 915,	\zhnumsetup	3, <u>20</u> , 868
	923, 924, 925, 926, 938, 939, 940, 941, 949, 950, 959, 960	\zhnumwithoptions	78, 81
\zhnum_set_dizhi_map:nn	<u>15</u> , 582, 1003, 1004,	\zhtiangang	2, <u>13</u> , 480
	1005, 1006, 1007, 1008, 1009, 1010, 1011, 1012, 1013, 1014	\zhtime	2, <u>12</u> , 404
\zhnum_set_financial_map:nn	<u>15</u> , 573, 967, 968,	\zhtoday	2, <u>10</u> , 338
	969, 970, 976, 978, 979, 985, 987, 988, 989, 990, 991, 992	\zhweekday	2, <u>10</u> , 346
\zhnum_set_financial_map:nnn	<u>15</u> , 575, 972, 973, 981, 982		