

zhnumber 宏包

李清

sobenlee@gmail.com

2019/04/07 v2.7*

第 1 节 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, 它提供的四个格式转换命令 \zhnumber, \zhdigits、\zhnum 和 \zhdig 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 LATEX3 项目的 expl3, xparse 和 l3keys2e 宏包。

第 2 节 使用方法

encoding = {GBK|Big5|UTF8}

Updated: 2014-09-09

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、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。

\zhnumber ☆ \zhnumber {{number}}

Updated: 2014-09-12

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 ☆ \zhdigits {{number}}
\zhigits * {{number}}

Updated: 2014-09-09

将阿拉伯数字转换为中文数字串。缺省状态下, \zhigits 将 0 映射为〇, 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 \zhigits{2012020120}\  
2 \zhigits*[2012020120]
```

* ctex-kit rev. 311c61d.

\zhnum ☆ \zhnum {<counter>}
 Updated: 2016-05-01

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

二

1 \zhnum{section}

\zhdig ☆ \zhdig {<counter>}
 New: 2016-05-01

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

二

1 \zhdig{section}

\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>}
 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

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

2012 年 5 月 21 日星期一

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

2019 年 4 月 7 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}

\zhcurrttime ☆

输出当前的时间。例如

New: 2012-05-25

23 时 41 分

1 \zhcurrttime

\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 己

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

\zh dizhi ☆

\zh dizhi {<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 巳

1 \zh dizhi{1} \zh dizhi{2} \zh dizhi{3}
 2 \zh dizhi{4} \zh dizhi{5} \zh dizhi{12}

\zh ganzhi ☆

\zh ganzhi {<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

1 \zh ganzhi{1} \zh ganzhi{2} \zh ganzhi{3} \`
 2 \zh ganzhi{4} \zh ganzhi{5} \zh ganzhi{60}

丁 卯 戌 辰 癸 亥

\zhganzhinian ☆

New: 2015-05-20

\zhganzhinian {\year}

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 己亥

```

1 \zhganzhinian{1898} \zhganzhinian{-246} \\
2 \zhganzhinian{-2697} \zhganzhinian{\year}

```

\zhnumExtendScaleMap

New: 2012-05-25

\zhnumExtendScaleMap [*character*] {\iota*character*₁, *character*₂, ..., *character*_n}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 $10^{4(i+11)}$ 。若给出可选项 *character*, 则当数字大于 $10^{4(n+12)} - 1$ 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。以粗体表示选项的默认值。

time

time = **Arabic** | Chinese

New: 2012-05-25

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

二〇一九年四月七日二十三时四十一分

```

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

```

arabicsep

arabicsep = {\sep}

New: 2016-05-01

设置日期和时间的数字格式为阿拉伯数字时, 阿拉伯数字与汉字的间隔内容。默认为一个空格。

style

Updated: 2012-05-25

style = **Simplified** | Traditional | Normal | Financial | Ancient

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, 以卅输出 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

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 ○, 可以使用这个选项。

ganzhi-cyclic

New: 2015-05-20

ganzhi-cyclic = {true|false}

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 \zh dizhi{13} \zh dizhi{24} \zh dizhi{1211}
5 \zh dizhi{-1} \zh dizhi{-2} \zh dizhi{-8199} \\
6 \zh ganzhi{61} \zh ganzhi{72} \zh ganzhi{2158} \\
7 \zh ganzhi{-1} \zh ganzhi{-2} \zh ganzhi{-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。

reset

reset

Updated: 2014-09-12

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)。

activechar

activechar = {true|false}

New: 2014-09-09

在 L^AT_EX 或者 pdfL^AT_EX 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, zhnumber 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 encoding 或者 reset 选项才会有效果。

\zhnumber

\zhdigits

\zhnum

\zhdig

Updated: 2016-05-01

```

\zhnumber  [(options)] {[number}]
\zhdigits * [(options)] {[number}]
\zhnum   [(options)] {[counter}]
\zhdig    [(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。

第3节 zhnumber 宏包代码实现

```

1 <*package>
2 <@=@=zhnum>
3 \msg_new:nnn { zhnumber } { 13-too-old }
4 {
```

```

5   Support~package~'expl3'~too~old. \\\\
6   Please~update~an~up~to~date~version~of~the~bundles\\\\
7   '13kernel'~and~'13packages'\\\\
8   using~your~TeX~package~manager~or~from~CTAN.
9 }
10 \c_if_package_later { expl3 } { 2019/03/05 } { }
11 { \msg_error:nn { zhnumber } { 13-too-old } }
12 \RequirePackage { xparse , 13keys2e }

```

\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 }

```

\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 }

```

\zhnum_number:n 先判断输入的是小数还是分数。

```

\__zhnum_number:www
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 }

```

__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 }

```

__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 }

```

\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 }
```

\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 }
```

\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 }
```

\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         { \exp_args:Nc \zhnum_int:n { 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_int
101         { \zhnum_parse_number:f { \int_eval:n {#1} } }
102     {
103         \int_compare:nNnTF {#1} < \c_zero_int
104             {
105                 \c__zhnum_minus_tl
106                 \zhnum_parse_number:f { \int_eval:n { - #1 } }
107             }
108             { \c__zhnum_zero_tl }
109     }
110 }
```

\@zhnum 用于支持 \pagelabel{zhnum}。

```
111 \cs_new_nopar:Npn \@zhnum { \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  \int_value:w
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_int } { } ;
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 }

```

__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 { + 1 } #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 }

```

__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_int
158  { \c_zhnum_zero_tl }
159  {
160  \int_compare:nNnF {#1} = \c_one_int
161  { \c_zhnum_minus_tl }
162  \zhnum_parse_number:nn {#2} {#3}
163 }
164 }

```

\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 9 < 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 }
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} { 4 } } {#1} }
177 \cs_new:Npn \__zhnum_parse_number:nnn #1#2
178 {
179     \int_compare:nNnTF {#2} < 2
180     { \zhnum_digit_map:n }
181     {
182         \int_compare:nNnTF {#1} = \c_zero_int
183         { \zhnum_split_number:fn { \int_eval:n { #2 / 4 - 1 } } }
184         { \__zhnum_split_number_aux:nnn {#1} {#2} }
185     }
186 }
187 \cs_generate_variant:Nn \zhnum_parse_number:n { f }

```

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     \int_value:w \int_div_truncate:nn {#2} { 4 }
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} }

```

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 }

```

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:NNnNNNNw #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_int
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_int
217   }
218   { \zhnum_split_number:NNfNNNNw \c_false_bool \c_true_bool }
219   { \zhnum_split_number:NNfNNNNw \c_true_bool \c_false_bool }
220 { \int_eval:n { #3 - 1 } }
221 }
222 \cs_generate_variant:Nn \zhnum_split_number:NNnNNNNw { NNF }

```

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_int
```

```

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_int
229 { \int_compare:nNnF { #1 * (#3#4) } = \c_zero_int { \c__zhnum_zero_tl } }
230 {
231     \bool_lazy_and:nnTF
232     { \l__zhnum_ancient_bool }
233     { \int_compare_p:nNn {#2} = 2 }
234     { \zhnum_digit_map:n { #2 00 } }
235     { \zhnum_digit_map:n {#2} \c__zhnum_hundred_tl }
236 }
237 \int_compare:nNnTF {#3} = \c_zero_int
238 { \int_compare:nNnF { #2 * #4 } = \c_zero_int { \c__zhnum_zero_tl } }
239 {
240     \bool_lazy_all:nF
241     {
242         { \int_compare_p:nNn {#3} = \c_one_int }
243         { \int_compare_p:nNn {#1#2} = \c_zero_int }
244         {#6}
245         {#5}
246     }
247 {
248     \bool_lazy_and:nnTF
249     { \l__zhnum_ancient_bool }
250     { \int_compare_p:n { 1 < #3 < 5 } }
251     { \zhnum_digit_map:n { #3 0 } \use_none:n }
252     { \zhnum_digit_map:n {#3} }
253 }
254 \c__zhnum_ten_tl
255 }
256 \int_compare:nNnF {#4} = \c_zero_int { \zhnum_digit_map:n {#4} }
257 }

```

\zhdig 用于将 L^AT_EX 计数器按中文数字串输出。

```

258 \DeclareExpandableDocumentCommand \zhdig { +o +m }
259 {
260     \IfNoValueTF {#1}
261     { \zhnum_digits_counter:n }
262     { \zhdigwithoptions {#1} }
263 {#2}
264 }
265 \NewDocumentCommand \zhdigwithoptions { +m +m }
266 {
267     \group_begin:
268     \keys_set:nn { zhnum / options } {#1}
269     \zhnum_digits_counter:n #1 {#2}
270     \group_end:
271 }
272 \cs_new:Npn \zhnum_digits_counter:n #1
273 {
274     \int_if_exist:cTF { c@#1 }
275     { \zhnum_digits_null:v { c@#1 } }
276     { \__zhnum_counter_error:n {#1} }
277 }

```

\@zhdig 用于支持 \pagenumbering{zhdig}。

```
278 \cs_new_nopar:Npn \@zhdig #1 { \zhnum_digits_null:f { \int_eval:n {#1} } }
```

\zhdigwithoptions 将输入的数字输出为中文数字串输出。

```

279 \DeclareExpandableDocumentCommand \zhdigwithoptions { +s +o +m }
280 {
281     \IfNoValueTF {#2}
282     { \zhnum_digits:Nn #1 }
283     { \zhdigwithoptions {#1} {#2} }
284 {#3}
285 }
```

```

286 \NewDocumentCommand \zhdigitswithoptions { +m +m +m }
287 {
288   \group_begin:
289     \keys_set:nn { zhnum / options } {#2}
290     \zhnum_digits:Nn #1 {#3}
291   \group_end:
292 }

\zhnum_digits_zero:n 快捷方式。
\zhnum_digits_null:n
293 \cs_new_nopar:Npn \zhnum_digits_zero:n
294   { \zhnum_digits:Nn \BooleanTrue }
295 \cs_new_nopar:Npn \zhnum_digits_null:n
296   { \zhnum_digits:Nn \BooleanFalse }
297 \cs_generate_variant:Nn \zhnum_digits_null:n { V , v , f }

\zhnum_digits:Nn 与 \zhnum_integer:n 类似,但不用去掉多余的零。
298 \cs_new:Npn \zhnum_digits:Nn #1#2
299 {
300   \exp_after:wN \__zhnum_read_digits:w
301   \int_value:w
302   \exp_after:wN \__zhnum_read_sign_loop:NN \exp_after:wN #1
303   \exp:w \exp_end_continue_f:w \use:n
304   #2 \exp_stop_f: \q_recursion_tail \q_recursion_stop
305 }
306 \cs_new:Npn \__zhnum_read_sign_loop:NN #1#2
307 {
308   \if:w + \if:w - \exp_not:N #2 + \fi: \exp_not:N #2
309   \exp_after:wN \__zhnum_read_sign_loop:NN \exp_after:wN #1
310   \exp:w \exp_end_continue_f:w \exp_after:wN \use:n
311   \else:
312     1 \exp_after:wN ;
313     \exp_after:wN \__zhnum_read_digits_loop:NN
314     \exp_after:wN #1
315     \exp_after:wN #2
316   \fi:
317 }
318 \cs_new:Npn \__zhnum_read_digits_loop:NN #1#2
319 {
320   \zhnum_if_digit:NTF #2
321   { \__zhnum_output_digits:NN #1#2 }
322   {
323     \quark_if_recursion_tail_stop:N #2
324     \if:w .\exp_not:N #2 \exp_after:wN \c__zhnum_dot_tl \fi:
325   }
326   \exp_after:wN \__zhnum_read_digits_loop:NN \exp_after:wN #1
327   \exp:w \exp_end_continue_f:w \use:n
328 }
329 \cs_new:Npn \__zhnum_read_digits:w #1 ;
330 {
331   \int_compare:nNnF {#1} = \c_one_int
332   { \c__zhnum_minus_tl }
333 }
334 \cs_new:Npn \__zhnum_output_digits:NN #1#2
335 {
336   \cs:w
337     c__zhnum_
338     \if_int_compare:w #2 = \c_zero_int
339       \IfBooleanTF #1 { zero } { null }
340     \else:
341       #2
342     \fi:
343     _tl
344   \cs_end:
345 }

\zhdate 输出中文日期。
346 \DeclareExpandableDocumentCommand \zhdate { +s +m }

```

```

347  {
348    \__zhnum_date:www #2 \q_stop
349    \IfBooleanT #1
350      { \__zhnum_week_day:www #2 \q_stop }
351    }
352 \cs_new:Npn \__zhnum_date:www #1/#2/#3 \q_stop
353  { \__zhnum_date_aux:n {#1} {#2} {#3} }

```

\zhtoday 输出当天日期。

```

354 \cs_new_nopar:Npn \zhtoday
355  { \__zhnum_date_aux:Vnn \tex_year:D \tex_month:D \tex_day:D }

\__zhnum_date_aux:n
356 \cs_new_nopar:Npn \__zhnum_date_aux:n
357  {
358    \bool_if:NTF \l__zhnum_time_bool
359      { \__zhnum_date_aux>NNnnnn \zhnum_digits_null:n \zhnum_int:n { } }
360      { \__zhnum_date_aux:Nnnnn \int_to_arabic:n { \l__zhnum_arabic_sep_tl } }
361    }
362 \cs_new:Npn \__zhnum_date_aux:Nnnnn #1
363  { \__zhnum_date_aux>NNnnnn #1#1 }
364 \cs_new:Npn \__zhnum_date_aux>NNnnnn #1#2#3#4#5#6
365  {
366    #1 {#4} #3 \c__zhnum_year_tl #3
367    #2 {#5} #3 \c__zhnum_month_tl #3
368    #2 {#6} #3 \c__zhnum_day_tl
369  }
370 \cs_generate_variant:Nn \__zhnum_date_aux:n { V }

```

\zhweekday 输出星期

```

371 \cs_new:Npn \zhweekday #1
372  { \__zhnum_week_day:www #1 \q_stop }

```

__zhnum_week_day:www 用 Zeller 公式计算的结果 h 与实际星期的关系是 $d = h + 5 \pmod{7} + 1$ 。

```

373 \cs_new:Npn \__zhnum_week_day:www #1/#2/#3 \q_stop
374  {
375    \if_case:w \zhnum_Zeller:n {#1} {#2} {#3} \exp_stop_f:
376      \c__zhnum_sat_tl
377      \or: \c__zhnum_sun_tl
378      \or: \c__zhnum_mon_tl
379      \or: \c__zhnum_tue_tl
380      \or: \c__zhnum_wed_tl
381      \or: \c__zhnum_thu_tl
382      \or: \c__zhnum_fri_tl
383    \fi:
384  }

```

\zhnum_Zeller:n 用 Zeller 公式¹ 计算星期几。

```

385 \cs_new:Npn \zhnum_Zeller:n #1#2#3
386  {
387    \int_compare:nNnTF
388      { #1 \zhnum_two_digits:n {#2} \zhnum_two_digits:n {#3} } > { 1582 10 04 }
389      { \__zhnum_Zeller_aux:Nnnn \zhnum_Zeller_Gregorian:nnn }
390      { \__zhnum_Zeller_aux:Nnnn \zhnum_Zeller_Julian:nnn }
391    {#1} {#2} {#3}
392  }
393 \cs_new:Npn \__zhnum_Zeller_aux:Nnnn #1#2#3#4
394  {
395    \int_compare:nNnTF {#3} < 3
396      { #1 { #2 - 1 } { #3 + 12 } {#4} }
397      { #1 {#2} {#3} {#4} }
398  }
399 \cs_new:Npn \zhnum_two_digits:n #1
400  {
401    \int_compare:nNnT {#1} < { 10 } { 0 }

```

¹http://en.wikipedia.org/wiki/Zeller%27s_congruence

```
402     \int_eval:n {#1}
403 }
```

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

$$h = \left(q + \left\lfloor \frac{26(m+1)}{10} \right\rfloor + Y + \left\lfloor \frac{Y}{4} \right\rfloor + 6 \left\lfloor \frac{Y}{100} \right\rfloor + \left\lfloor \frac{Y}{400} \right\rfloor \right) \pmod{7}$$

其中 Y 为年, m 为月, q 为日; 若 $m = 1, 2$, 则令 $m += 12$, 同时 $Y--$ 。

```
404 \cs_new:Npn \zhnum_Zeller_Gregorian:nnn #1#2#3
405 {
406     \int_mod:nn
407     {
408         (#3)
409         + \int_div_truncate:nn { 26 * ( #2 + 1 ) } { 10 }
410         + (#1)
411         + \int_div_truncate:nn {#1} { 4 }
412         + 6 * \int_div_truncate:nn {#1} { 100 }
413         + \int_div_truncate:nn {#1} { 400 }
414     }
415     { 7 }
416 }
```

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

$$h = \left(q + \left\lfloor \frac{26(m+1)}{10} \right\rfloor + Y + \left\lfloor \frac{Y}{4} \right\rfloor + 5 \right) \pmod{7}$$

```
417 \cs_new:Npn \zhnum_Zeller_Julian:nnn #1#2#3
418 {
419     \int_mod:nn
420     {
421         (#3)
422         + \int_div_truncate:nn { 26 * ( #2 + 1 ) } { 10 }
423         + (#1)
424         + \int_div_truncate:nn {#1} { 4 }
425         + 5
426     }
427     { 7 }
428 }
```

\zhtime 输出时间。

```
429 \cs_new:Npn \zhtime #1
430 { \__zhnum_time:ww #1 \q_stop }
431 \use:x
432 {
433     \cs_new:Npn \exp_not:N \__zhnum_time:ww ##1 \c_colon_str ##2 \exp_not:N \q_stop
434 }
435 { \__zhnum_time_aux:nn {#1} {#2} }
```

\zhcurrttime 输出当前时间。

```
436 \cs_new_nopar:Npn \zhcurrttime
437 {
438     \__zhnum_time_aux:nn
439     { \int_div_truncate:nn \tex_time:D { 60 } }
440     { \int_mod:nn \tex_time:D { 60 } }
441 }

442 \cs_new_nopar:Npn \__zhnum_time_aux:nn
443 {
444     \bool_if:NTF \l__zhnum_time_bool
445     { \__zhnum_time_aux:Nnnn \zhnum_int:n { } }
446     { \__zhnum_time_aux:Nnnn \int_to_arabic:n { \l__zhnum_arabic_sep_tl } }
447 }
448 \cs_new:Npn \__zhnum_time_aux:Nnnn #1#2#3#4
449 {
```

```

450     #1 {#3} #2 \c__zhnum_hour_tl #2
451     #1 {#4} #2 \c__zhnum_minute_tl
452 }

```

\zhnum_digit_map:n 阿拉伯数字与中文数字的映射。

```

453 \cs_new:Npn \zhnum_digit_map:n #1
454     { \use:c { c__zhnum_ #1 _tl } }

```

\zhnum_scale_map:n 大数系统的映射。

```

455 \cs_new:Npn \zhnum_scale_map:n #1
456     {
457         \cs_if_exist_use:cF { c__zhnum_s #1 _tl }
458             { \zhnum_scale_map_hook:n {#1} }
459     }
460 \cs_new:Npn \zhnum_scale_map_loop:n #1
461     { \zhnum_scale_map:n { \int_mod:nn {#1} \l__zhnum_scale_int } }
462 \cs_generate_variant:Nn \zhnum_scale_map:n { f }
463 \int_new:N \l__zhnum_scale_int
464 \int_set:Nn \l__zhnum_scale_int { 11 }
465 \cs_new_eq:NN \zhnum_scale_map_hook:n \zhnum_scale_map_loop:n
466 \tl_const:cn { c__zhnum_s0_tl } { }

```

\zhnumExtendScaleMap 扩展进位系统。

```

467 \NewDocumentCommand \zhnumExtendScaleMap { > { \TrimSpaces } +o +m }
468     {
469         \int_zero:N \l_tmpa_int
470         \clist_map_function:nN {#2} \zhnum_set_scale:n
471         \IfNoValueF {#1}
472             { \cs_set:Npn \zhnum_scale_map_hook:n ##1 {#1} }
473     }

```

\zhnum_set_scale:n

```

474 \cs_new_protected:Npn \zhnum_set_scale:n #1
475     {
476         \int_incr:N \l_tmpa_int
477         \tl_set:Nx \l_tmpa_tl
478             { c__zhnum_s \int_eval:n { \l_tmpa_int + 11 } _tl }
479         \tl_if_exist:cF { \l_tmpa_tl }
480             { \int_incr:N \l__zhnum_scale_int }
481         \tl_set:cn { \l_tmpa_tl } {#1}
482     }

```

\zhnum_ganzhi_normal:nnn 保证干支的参数为正数。

```

483 \cs_new:Npn \zhnum_ganzhi_normal:nnn #1#2#3
484     {
485         \int_compare:nNnF {#1} < \c_one_int
486             { \cs_if_exist_use:c { c__zhnum_ #2 _ #1 _tl } }
487     }

```

\zhnum_ganzhi_cyclic:nnn 对超出范围的数字取模，参数 0 的结果是空值。

```

488 \cs_new:Npn \zhnum_ganzhi_cyclic:nnn #1#2#3
489     {
490         \int_compare:nNnF {#1} = \c_zero_int
491             {
492                 \cs_if_exist_use:cF { c__zhnum_ #2 _ #1 _tl }
493                     {
494                         \__zhnum_ganzhi_cyclic_mod:fnnn
495                             { \int_mod:nn {#1} {#3} } {#1} {#2} {#3}
496                     }
497             }
498     }
499 \cs_new:Npn \__zhnum_ganzhi_cyclic_mod:nnnn #1#2#3#4
500     {
501         \int_compare:nNnTF {#2} > \c_zero_int
502             { \use:c { c__zhnum_ #3 _ #1 _tl } }
503             {

```

```

504     \int_compare:nNnTF {#1} = \c_zero_int
505     { \use:c { c__zhnum_ #3 _ 1 _tl } }
506     { \use:c { c__zhnum_ #3 _ \int_eval:n { #1 + #4 + 1 } _tl } }
507   }
508 }
509 \cs_generate_variant:Nn \__zhnum_ganzhi_cyclic_mod:nnnn { f }
```

\zhnum_ganzhi:nnn 默认不对超出范围的数字取模。

```

510 \cs_new_eq:NN \zhnum_ganzhi:nnn \zhnum_ganzhi_normal:nnn
511 \cs_generate_variant:Nn \zhnum_ganzhi:nnn { f }
```

\zhtiangan 天干。

```

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

\zhdizhi 地支。

```

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

\zhganzhi 干支。

```

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

\zhganzhinian 干支纪年。

```

518 \cs_new:Npn \zhganzhinian #1
519 { \zhnum_ganzhi_nian:f { \int_eval:n {#1} } }
```

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

```

520 \cs_new:Npn \zhnum_ganzhi_nian:n #1
521 {
522     \int_compare:nNnTF {#1} > \c_zero_int
523     { \use:c { c__zhnum_ganzhi_ \int_mod:nn { #1 + 57 } { 60 } _tl } }
524     {
525         \int_compare:nNnF {#1} = \c_zero_int
526         {
527             \use:c
528             {
529                 c__zhnum_ganzhi_
530                 \int_eval:n { \int_mod:nn { #1 - 2 } { 60 } + 60 }
531                 _tl
532             }
533         }
534     }
535 }
536 \cs_generate_variant:Nn \zhnum_ganzhi_nian:n { f }
```

根据需要设置中文阿拉伯数字。

```

537 \group_begin:
538   \tl_set:Nn \l_tmpa_tl
539   {
540     - .tl_set:N = \l__zhnum_minus_tl ,
541     -0 .tl_set:N = \l__zhnum_null_tl ,
542   }
543   \tl_put_right:Nx \l_tmpa_tl
544   {
545     E2 .tl_set:N = \exp_not:c { l__zhnum_ 100 _tl } ,
546     E3 .tl_set:N = \exp_not:c { l__zhnum_ 1000 _tl } ,
547     FE2 .tl_set:N = \exp_not:c { l__zhnum_financial_ 100 _tl } ,
548     FE3 .tl_set:N = \exp_not:c { l__zhnum_financial_ 1000 _tl } ,
549     D11 .tl_set:N = \exp_not:c { l__zhnum_dizhi_ 11 _tl } ,
550     D12 .tl_set:N = \exp_not:c { l__zhnum_dizhi_ 12 _tl } ,
551     E44 .tl_set:N = \exp_not:c { l__zhnum_s11 _tl } ,
552   }
553 \int_step_inline:nn { 10 }
```

```

554 {
555   \tl_put_right:Nx \l_tmpa_tl
556   {
557     #1 .tl_set:N = \exp_not:c { l_zhnum_ #1 _tl } ,
558     F#1 .tl_set:N = \exp_not:c { l_zhnum_financial_ #1 _tl } ,
559     T#1 .tl_set:N = \exp_not:c { l_zhnum_tiangan_ #1 _tl } ,
560     D#1 .tl_set:N = \exp_not:c { l_zhnum_dizhi_ #1 _tl } ,
561     GZ#1 .tl_set:N = \exp_not:c { l_zhnum_ganzhi_ #1 _tl } ,
562     E \int_eval:n { #1 * 4 }
563     .tl_set:N = \exp_not:c { l_zhnum_ s#1 _tl } ,
564   }
565 }
566 \int_step_inline:nnn { 11 } { 60 }
567 {
568   \tl_put_right:Nx \l_tmpa_tl
569   { GZ#1 .tl_set:N = \exp_not:c { l_zhnum_ganzhi_ #1 _tl } , }
570 }
571 \clist_map_inline:nn { 0 , 100 , 1000 }
572 {
573   \tl_put_right:Nx \l_tmpa_tl
574   {
575     #1 .tl_set:N = \exp_not:c { l_zhnum_ #1 _tl } ,
576     F#1 .tl_set:N = \exp_not:c { l_zhnum_financial_ #1 _tl } ,
577   }
578 }
579 \clist_map_inline:nn { 20 , 30 , 40 , 200 }
580 {
581   \tl_put_right:Nx \l_tmpa_tl
582   { #1 .tl_set:N = \exp_not:c { l_zhnum_ #1 _tl } , }
583 }
584 \clist_map_inline:nn
585 {
586   dot , and , parts , year , month , day , weekday , hour , minute
587   mon , tue , wed , thu , fri , sat , sun
588 }
589 {
590   \tl_put_right:Nx \l_tmpa_tl
591   { #1 .tl_set:N = \exp_not:c { l_zhnum_ #1 _tl } , }
592 }
593 \use:x
594 {
595   \group_end:
596   \keys_define:nn { zhnum / options } { \exp_not:o \l_tmpa_tl }
597 }

```

将配置文件中的中文数字保存到 prop 变量中。

```

598 \cs_new_protected:Npn \zhnum_set_digits_map:nn #1#2
599   { \prop_put:Nnn \l_zhnum_cfg_map_prop {#1} {#2} }
600 \cs_new_protected:Npn \zhnum_set_digits_map:nnn #1#2#3
601   {
602     \prop_put_if_new:Nnn \l_zhnum_cfg_map_prop {#1} {#3}
603     \prop_put:Nnn \l_zhnum_cfg_map_var_prop {#1#2} {#3}
604   }
605 \cs_new_protected:Npn \zhnum_set_financial_map:nn #1#2
606   { \prop_put:Nnn \l_zhnum_cfg_map_finan_prop {#1} {#2} }
607 \cs_new_protected:Npn \zhnum_set_financial_map:nnn #1#2#3
608   {
609     \prop_put_if_new:Nnn \l_zhnum_cfg_map_finan_prop {#1} {#3}
610     \prop_put:Nnn \l_zhnum_cfg_map_var_prop { financial_#1#2 } {#3}
611   }
612 \cs_new_protected:Npn \zhnum_set_tiangan_map:nn #1#2
613   { \prop_put:Nnn \l_zhnum_cfg_map_ganzhi_prop { tiangan_#1 } {#2} }
614 \cs_new_protected:Npn \zhnum_set_dizhi_map:nn #1#2
615   { \prop_put:Nnn \l_zhnum_cfg_map_ganzhi_prop { dizhi_#1 } {#2} }
616 \prop_new:N \l_zhnum_cfg_map_prop
617 \prop_new:N \l_zhnum_cfg_map_var_prop
618 \prop_new:N \l_zhnum_cfg_map_finan_prop

```

619 \prop_new:N \l_zhnum_cfg_map_ganzhi_prop

将 prop 表转化到单独的 tl 变量。

```

620 \cs_new_protected_nopar:Npn \zhnum_parse_config:
621 {
622     \prop_map_function:NN \l_zhnum_cfg_map_prop \zhnum_check_simp:nn
623     \prop_map_function:NN \l_zhnum_cfg_map_ganzhi_prop \zhnum_assgin_ganzhi:nn
624     \zhnum_set_zero:
625     \zhnum_set_week_day:
626     \bool_if:NF \l_zhnum_reset_bool
627     {
628         \zhnum_assgin_const:
629         \bool_set_true:N \l_zhnum_reset_bool
630     }
631 }
632 \cs_new_protected:Npn \zhnum_check_simp:nn #1#2
633 {
634     \__zhnum_check_simp_aux:nn {#2} {#1}
635     \prop_get:NnNT \l_zhnum_cfg_map_finan_prop {#1} \l_tmpa_tl
636     { \exp_args:No \__zhnum_check_simp_aux:nn { \l_tmpa_tl } { financial_ #1 } }
637 }
638 \cs_new_protected:Npn \__zhnum_check_simp_aux:nn #1#2
639 {
640     \prop_get:NnNTF \l_zhnum_cfg_map_var_prop { #2 _trad } \l_tmpa_tl
641     {
642         \prop_get:NnNF \l_zhnum_cfg_map_var_prop { #2 _simp } \l_tmpb_tl
643         { \tl_set:Nn \l_tmpb_tl {#1} }
644         \tl_set:cx { l_zhnum_ #2 _tl }
645         {
646             \exp_not:n { \bool_if:NTF \l_zhnum_simp_bool }
647             { \exp_not:o \l_tmpb_tl } { \exp_not:o \l_tmpa_tl }
648         }
649     }
650     { \tl_set:cn { l_zhnum_ #2 _tl } {#1} }
651 }
652 \cs_new_protected_nopar:Npn \zhnum_assgin_const:
653 {
654     \prop_map_function:NN \l_zhnum_cfg_map_prop \zhnum_check_financial:nn
655     \zhnum_set_alias:
656 }
657 \cs_new_protected:Npn \zhnum_check_financial:nn #1#2
658 {
659     \prop_get:NnNTF \l_zhnum_cfg_map_finan_prop {#1} \l_tmpa_tl
660     {
661         \zhnum_assgin_const_tl:cx { c_zhnum_ #1 _tl }
662         {
663             \exp_not:n { \bool_if:NTF \l_zhnum_normal_bool }
664             { \exp_not:c { l_zhnum_ #1 _tl } }
665             { \exp_not:c { l_zhnum_financial_ #1 _tl } }
666         }
667     }
668     {
669         \zhnum_assgin_const_tl:cx
670         { c_zhnum_ #1 _tl } { \exp_not:c { l_zhnum_ #1 _tl } }
671     }
672 }
673 \cs_new_protected_nopar:Npn \zhnum_set_zero:
674 {
675     \tl_set:cx { l_zhnum_0_t1 }
676     {
677         \exp_not:n { \bool_if:NTF \l_zhnum_null_bool }
678         { \exp_not:o \l_zhnum_null_t1 } { \exp_not:v { l_zhnum_0_t1 } }
679     }
680 }
681 \cs_new_protected_nopar:Npn \zhnum_set_week_day:
682 {
683     \tl_set:Nx \l_zhnum_mon_t1

```

```

684      { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_1_tl } }
685      \tl_set:Nx \l__zhnum_tue_tl
686      { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_2_tl } }
687      \tl_set:Nx \l__zhnum_wed_tl
688      { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_3_tl } }
689      \tl_set:Nx \l__zhnum_thu_tl
690      { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_4_tl } }
691      \tl_set:Nx \l__zhnum_fri_tl
692      { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_5_tl } }
693      \tl_set:Nx \l__zhnum_sat_tl
694      { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_6_tl } }
695      \tl_set:Nx \l__zhnum_sun_tl
696      { \exp_not:N \c__zhnum_weekday_tl \exp_not:o \l__zhnum_day_tl }
697  }
698 \clist_map_inline:nn { mon , tue , wed , thu , fri , sat , sun }
699  { \tl_const:cx { c__zhnum_ #1 _tl } { \exp_not:c { l__zhnum_ #1 _tl } } }
700 \cs_new_protected:Npn \zhnum_assgin_ganzhi:nn #1#2
701  { \tl_set:cn { l__zhnum_ #1 _tl } {#2} }
702 \cs_new:Npn \zhnum_zero_mod:nn #1#2
703  { \exp_args:Nf \zhnum_zero_mod_aux:nn { \int_mod:nn {#1} {#2} } {#2} }
704 \cs_new:Npn \zhnum_zero_mod_aux:nn #1#2
705  { \int_compare:nNnTF {#1} = \c_zero_int {#2} {#1} }
706 \int_step_inline:nn { 60 }
707  {
708      \tl_const:cx { c__zhnum_ganzhi_ #1 _tl } { \exp_not:c { l__zhnum_ganzhi_ #1 _tl } }
709      \tl_set:cx { l__zhnum_ganzhi_ #1 _tl }
710      {
711          \exp_not:c { l__zhnum_tiangan_ \zhnum_zero_mod:nn {#1} { 10 } _tl }
712          \exp_not:c { l__zhnum_dizhi_ \zhnum_zero_mod:nn {#1} { 12 } _tl }
713      }
714  }
715 \cs_new_eq:cc { c__zhnum_ganzhi_ 0 _tl } { c__zhnum_ganzhi_ 60 _tl }
716 \cs_new_eq:NN \zhnum_assgin_const_tl:cx \tl_const:cx
717 \AtEndOfPackage
718  {
719      \prop_map_inline:Nn \l__zhnum_cfg_map_ganzhi_prop
720      { \tl_const:cx { c__zhnum_ #1 _tl } { \exp_not:c { l__zhnum_ #1 _tl } } }
721      \cs_new_eq:cc { c__zhnum_tiangan_ 0 _tl } { c__zhnum_tiangan_ 10 _tl }
722      \cs_new_eq:cc { c__zhnum_dizhi_ 0 _tl } { c__zhnum_dizhi_ 12 _tl }
723      \cs_set_eq:NN \zhnum_assgin_const_tl:cx \tl_set:cx
724  }

```

\zhnum_set_alias: 一些易于使用的别名。

```

725 \cs_new_eq:NN \zhnum_set_alias:NN \cs_new_eq:NN
726 \cs_new_protected_nopar:Npx \zhnum_set_alias:
727  {
728      \zhnum_set_alias:NN \exp_not:N \c__zhnum_zero_tl
729          \exp_not:c { c__zhnum_ 0 _tl }
730      \zhnum_set_alias:NN \exp_not:N \c__zhnum_ten_tl
731          \exp_not:c { c__zhnum_ 10 _tl }
732      \zhnum_set_alias:NN \exp_not:N \c__zhnum_hundred_tl
733          \exp_not:c { c__zhnum_ 100 _tl }
734      \zhnum_set_alias:NN \exp_not:N \c__zhnum_thousand_tl
735          \exp_not:c { c__zhnum_ 1000 _tl }
736  }
737 \AtEndOfPackage
738  { \cs_set_eq:NN \zhnum_set_alias:NN \tl_set_eq:NN }

```

\zhnum_load_cfg:n 根据选定编码载入配置文件。

```

739 \cs_new_protected:Npn \zhnum_load_cfg:n #1
740  {
741      \zhnum_set_cfg_name:Nn \l__zhnum_cfg_str {#1}
742      \str_if_eq:NNF \l__zhnum_cfg_str \l__zhnum_last_cfg_str
743          { \zhnum_update_cfg:n {#1} }
744      \zhnum_parse_config:
745  }

```

```

746 \cs_generate_variant:Nn \zhnum_load_cfg:n { o }
747 \cs_new_protected:Npn \zhnum_update_cfg:n #1
748 {
749     \prop_if_exist:cTF { g__zhnum_cfg_ \l__zhnum_cfg_str _prop }
750         { \str_set_eq:NN \l__zhnum_last_cfg_str \l__zhnum_cfg_str }
751         { \zhnum_input_cfg:n {#1} }
752     \l__zhnum_update_cfg_prop:N \prop_set_eq:Nc
753 }
754 \cs_new_protected:Npn \zhnum_input_cfg:n #1
755 {
756     \file_if_exist:nTF { zhnumber - #1 .cfg }
757     {
758         \bool_set_false:N \l__zhnum_reset_bool
759         \__zhnum_update_cfg_prop:N \__zhnum_prop_initial:Nn
760         \group_begin:
761             \zhnum_set_catcode:
762             \file_input:n { zhnumber - #1 .cfg }
763             \__zhnum_update_cfg_prop:N \__zhnum_prop_gset_eq:Nn
764         \group_end:
765     }
766     { \msg_error:nnx { zhnumber } { file-not-found } {#1} }
767 }
768 \cs_new_protected:Npn \__zhnum_update_cfg_prop:N #1
769 {
770     #1 \l__zhnum_cfg_map_prop { g__zhnum_cfg_ \l__zhnum_cfg_str _prop }
771     #1 \l__zhnum_cfg_map_var_prop { g__zhnum_cfg_var_ \l__zhnum_cfg_str _prop }
772     #1 \l__zhnum_cfg_map_finan_prop { g__zhnum_cfg_finan_ \l__zhnum_cfg_str _prop }
773     #1 \l__zhnum_cfg_map_ganzhi_prop { g__zhnum_cfg_ganzhi_ \l__zhnum_cfg_str _prop }
774 }
775 \cs_new_protected:Npn \__zhnum_prop_initial:Nn #1#2
776 {
777     \prop_clear:N #1
778     \prop_new:c {#2}
779 }
780 \cs_new_protected:Npn \__zhnum_prop_gset_eq:Nn #1#2
781 { \prop_gset_eq:cN {#2} #1 }
782 \str_new:N \l__zhnum_cfg_str
783 \str_new:N \l__zhnum_last_cfg_str
784 \bool_new:N \l__zhnum_reset_bool
785 \msg_new:nnnn { zhnumber } { file-not-found }
786 { File`~#1'~not~found. }
787 {
788     The~requested~file~could~not~be~found~in~the~current~directory,~
789     in~the~TeX~search~path~or~in~the~LaTeX~search~path.
790 }

```

\zhnum_if_unicode_engine_p:
\zhnum_if_unicode_engine:TF 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。

```

791 \bool_lazy_any:nTF
792 {
793     { \sys_if_engine_xetex_p: }
794     { \sys_if_engine_luatex_p: }
795     { \sys_if_engine_uptex_p: }
796 }
797 {
798     \cs_new_eq:NN \zhnum_if_unicode_engine_p: \c_true_bool
799     \cs_new_eq:NN \zhnum_if_unicode_engine:TF \use_i:nn
800 }
801 {
802     \cs_new_eq:NN \zhnum_if_unicode_engine_p: \c_false_bool
803     \cs_new_eq:NN \zhnum_if_unicode_engine:TF \use_ii:nn
804 }

```

\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。

```

805 \if_predicate:w \zhnum_if_unicode_engine_p:
806     \cs_new_eq:NN \zhnum_set_catcode: \prg_do_nothing:

```

```

807 \cs_new_protected:Npn \zhnum_set_cfg_name:Nn #1#2
808 {
809     \str_set:Nx \l__zhnum_encoding_str {#2}
810     \str_set_eq:NN #1 \l__zhnum_encoding_str
811 }
812 \cs_new_eq:NN \zhnum_reset_config: \zhnum_parse_config:
813 \else:
814 \cs_new_protected_nopar:Npn \zhnum_set_catcode:
815     { \bool_if:NT \l__zhnum_active_char_bool { \zhnum_set_active: } }
816 \cs_new_protected_nopar:Npn \zhnum_set_active:
817 {
818     \str_case:onTF { \l__zhnum_encoding_str }
819     {
820         { gbk } { \int_set:Nn \l__zhnum_byte_min_int { "81" } }
821         { big5 } { \int_set:Nn \l__zhnum_byte_min_int { "A1" } }
822     }
823     { \int_set:Nn \l__zhnum_byte_max_int { "FE" } }
824     {
825         \int_set:Nn \l__zhnum_byte_min_int { "E0" }
826         \int_set:Nn \l__zhnum_byte_max_int { "EF" }
827     }
828     \int_step_function:nnN
829     { \l__zhnum_byte_min_int }
830     { \l__zhnum_byte_max_int } \char_set_catcode_active:n
831 }
832 \int_new:N \l__zhnum_byte_min_int
833 \int_new:N \l__zhnum_byte_max_int
834 \cs_new_protected:Npn \zhnum_set_cfg_name:Nn #1#2
835 {
836     \str_set:Nx \l__zhnum_encoding_str {#2}
837     \str_set:Nx #1
838     {
839         \l__zhnum_encoding_str
840         \bool_if:NT \l__zhnum_active_char_bool { _active }
841     }
842 }
843 \cs_new_protected_nopar:Npn \zhnum_reset_config:
844     { \zhnum_load_cfg:o { \l__zhnum_encoding_str } }
845 \bool_new:N \l__zhnum_active_char_bool
846 \bool_set_true:N \l__zhnum_active_char_bool
847 \fi:

```

encoding 宏包设置选项。

```

style 848 \keys_define:nn { zhnum / options }
null 849 {
reset 850     encoding .choices:nn =
851     { UTF8 , GBK , Big5 }
852     {
853         \str_set:Nx \l__zhnum_encoding_str
854         { \str_fold_case:V \l_keys_choice_tl }
855         \zhnum_load_cfg:o { \l__zhnum_encoding_str }
856     } ,
857     encoding .default:n = { GBK } ,
858     encoding / Bg5 .meta:n = { encoding = Big5 } ,
859     encoding / unknown .code:n =
860     { \msg_error:nnn { zhnumber } { encoding-invalid } {#1} } ,
861     style .multichoice: ,
862     style / Normal .code:n =
863     {
864         \bool_set_false:N \l__zhnum_ancient_bool
865         \bool_set_true:N \l__zhnum_normal_bool
866     } ,
867     style / Financial .code:n =
868     {
869         \bool_set_false:N \l__zhnum_ancient_bool
870         \bool_set_false:N \l__zhnum_normal_bool
871     } ,

```

```

872     style / Ancient      .code:n =
873     {
874         \bool_set_true:N \l__zhnum_ancient_bool
875         \bool_set_true:N \l__zhnum_normal_bool
876     } ,
877     style / Simplified   .code:n = { \bool_set_true:N \l__zhnum_simp_bool } ,
878     style / Traditional   .code:n = { \bool_set_false:N \l__zhnum_simp_bool } ,
879     style           .default:n = { Normal , Simplified } ,
880     null           .bool_set:N = \l__zhnum_null_bool ,
881     time .choice: ,
882     time / Chinese      .code:n = { \bool_set_true:N \l__zhnum_time_bool } ,
883     time / Arabic        .code:n = { \bool_set_false:N \l__zhnum_time_bool } ,
884     time           .default:n = { Arabic } ,
885     reset           .code:n = { \zhnum_reset_config: } ,
886     activechar       .bool_set:N = \l__zhnum_active_char_bool ,
887     ganzhi-cyclic .choice: ,
888     ganzhi-cyclic / true .code:n =
889     { \cs_set_eq:NN \zhnum_ganzhi:nnn \zhnum_ganzhi_cyclic:nnn } ,
890     ganzhi-cyclic / false.code:n =
891     { \cs_set_eq:NN \zhnum_ganzhi:nnn \zhnum_ganzhi_normal:nnn } ,
892     ganzhi-cyclic       .default:n = { true } ,
893     arabicsep        .tl_set:N = \l__zhnum_arabic_sep_tl
894 }
895 \str_new:N \l__zhnum_encoding_str
896 \msg_new:nnnn { zhnumber } { encoding-invalid }
897 { The~encoding``#1``is~invalid. }
898 { Available~encodings~are``UTF8'', ``GBK''and``Big5''. }

```

\zhnumsetup 在文档中设置 zhnumber 的接口。

```

899 \NewDocumentCommand \zhnumsetup { +m }
900 {
901     \keys_set:nn { zhnum / options } {#1}
902     \tex_ignorespaces:D
903 }

```

初始化设置和执行宏包选项。

```

904 \keys_set:nn { zhnum / options } { style , time , arabicsep = { ~ } }
905 \ProcessKeysOptions { zhnum / options }

```

如果没有选定编码，则根据引擎自动设置编码。

```

906 \str_if_empty:NT \l__zhnum_encoding_str
907 {
908     \zhnum_if_unicode_engine:TF
909     { \keys_set:nn { zhnum / options } { encoding = UTF8 } }
910     { \keys_set:nn { zhnum / options } { encoding = GBK } }
911 }
912 
```

`</package>`

第4节 中文数字配置文件

```

913 <*config>
914 <!*big5>
915 \zhnum_set_digits_map:nnn { minus } { simp } { 负 }
916 \zhnum_set_digits_map:nnn { minus } { trad } { 负 }
917 </!*big5>
918 <!*big5>
919 \zhnum_set_digits_map:nn { minus } { 负 }
920 </!*big5>
921 \zhnum_set_digits_map:nn { 0 } { 零 }
922 <!*big5>
923 \zhnum_set_digits_map:nn { null } { O }
924 </!*big5>
925 <!*big5>
926 \zhnum_set_digits_map:nn { null } { O }

```

```

927 </big5>
928 \zhnum_set_digits_map:nn { 1 } { 一 }
929 \zhnum_set_digits_map:nn { 2 } { 二 }
930 \zhnum_set_digits_map:nn { 3 } { 三 }
931 \zhnum_set_digits_map:nn { 4 } { 四 }
932 \zhnum_set_digits_map:nn { 5 } { 五 }
933 \zhnum_set_digits_map:nn { 6 } { 六 }
934 \zhnum_set_digits_map:nn { 7 } { 七 }
935 \zhnum_set_digits_map:nn { 8 } { 八 }
936 \zhnum_set_digits_map:nn { 9 } { 九 }
937 \zhnum_set_digits_map:nn { 10 } { 十 }
938 \zhnum_set_digits_map:nn { 100 } { 百 }
939 \zhnum_set_digits_map:nn { 1000 } { 千 }
940 \zhnum_set_digits_map:nn { 20 } { 廿 }
941 \zhnum_set_digits_map:nn { 30 } {卅 }
942 \zhnum_set_digits_map:nn { 40 } { 晞 }
943 \zhnum_set_digits_map:nn { 200 } { 皕 }
944 <!big5>
945 \zhnum_set_digits_map:nnn { dot } { simp } { 点 }
946 \zhnum_set_digits_map:nnn { dot } { trad } { 點 }
947 </big5>
948 <*big5>
949 \zhnum_set_digits_map:nn { dot } { 點 }
950 </big5>
951 \zhnum_set_digits_map:nn { and } { 又 }
952 \zhnum_set_digits_map:nn { parts } { 分之 }
953 <!big5>
954 \zhnum_set_digits_map:nnn { s1 } { simp } { 万 }
955 \zhnum_set_digits_map:nnn { s1 } { trad } { 萬 }
956 \zhnum_set_digits_map:nnn { s2 } { simp } { 亿 }
957 \zhnum_set_digits_map:nnn { s2 } { trad } { 億 }
958 </big5>
959 <*big5>
960 \zhnum_set_digits_map:nn { s1 } { 萬 }
961 \zhnum_set_digits_map:nn { s2 } { 億 }
962 </big5>
963 \zhnum_set_digits_map:nn { s3 } { 兆 }
964 \zhnum_set_digits_map:nn { s4 } { 京 }
965 \zhnum_set_digits_map:nn { s5 } { 核 }
966 \zhnum_set_digits_map:nn { s6 } { 稗 }
967 \zhnum_set_digits_map:nn { s7 } { 穢 }
968 <!big5>
969 \zhnum_set_digits_map:nnn { s8 } { simp } { 沟 }
970 \zhnum_set_digits_map:nnn { s8 } { trad } { 溝 }
971 \zhnum_set_digits_map:nnn { s9 } { simp } { 涧 }
972 \zhnum_set_digits_map:nnn { s9 } { trad } { 潤 }
973 </big5>
974 <*big5>
975 \zhnum_set_digits_map:nn { s8 } { 溝 }
976 \zhnum_set_digits_map:nn { s9 } { 潤 }
977 </big5>
978 \zhnum_set_digits_map:nn { s10 } { 正 }
979 <!big5>
980 \zhnum_set_digits_map:nnn { s11 } { simp } { 载 }
981 \zhnum_set_digits_map:nnn { s11 } { trad } { 載 }
982 </big5>
983 <*big5>
984 \zhnum_set_digits_map:nn { s11 } { 載 }
985 </big5>
986 \zhnum_set_digits_map:nn { year } { 年 }
987 \zhnum_set_digits_map:nn { month } { 月 }
988 \zhnum_set_digits_map:nn { day } { 日 }
989 <!big5>
990 \zhnum_set_digits_map:nnn { hour } { simp } { 时 }
991 \zhnum_set_digits_map:nnn { hour } { trad } { 時 }
992 </big5>
993 <*big5>

```

```
994 \zhnum_set_digits_map:nn { hour } { 時 }
995 </big5>
996 \zhnum_set_digits_map:nn { minute } { 分 }
997 \zhnum_set_digits_map:nn { weekday } { 星期 }
998 \zhnum_set_financial_map:nn { null } { 零 }
999 \zhnum_set_financial_map:nn { 0 } { 零 }
1000 \zhnum_set_financial_map:nn { 1 } { 壹 }
1001 \zhnum_set_financial_map:nn { 2 } { 贳 }
1002 <!*big5>
1003 \zhnum_set_financial_map:nnn { 3 } { simp } { 叁 }
1004 \zhnum_set_financial_map:nnn { 3 } { trad } { 參 }
1005 </!*big5>
1006 <!*big5>
1007 \zhnum_set_financial_map:nn { 3 } { 參 }
1008 </big5>
1009 \zhnum_set_financial_map:nn { 4 } { 肆 }
1010 \zhnum_set_financial_map:nn { 5 } { 伍 }
1011 <!*big5>
1012 \zhnum_set_financial_map:nnn { 6 } { simp } { 陆 }
1013 \zhnum_set_financial_map:nnn { 6 } { trad } { 陸 }
1014 </!*big5>
1015 <!*big5>
1016 \zhnum_set_financial_map:nn { 6 } { 陸 }
1017 </big5>
1018 \zhnum_set_financial_map:nn { 7 } { 柒 }
1019 \zhnum_set_financial_map:nn { 8 } { 捌 }
1020 \zhnum_set_financial_map:nn { 9 } { 玖 }
1021 \zhnum_set_financial_map:nn { 10 } { 拾 }
1022 \zhnum_set_financial_map:nn { 100 } { 佰 }
1023 \zhnum_set_financial_map:nn { 1000 } { 仟 }
1024 \zhnum_set_tiangan_map:nn { 1 } { 甲 }
1025 \zhnum_set_tiangan_map:nn { 2 } { 乙 }
1026 \zhnum_set_tiangan_map:nn { 3 } { 丙 }
1027 \zhnum_set_tiangan_map:nn { 4 } { 丁 }
1028 \zhnum_set_tiangan_map:nn { 5 } { 戊 }
1029 \zhnum_set_tiangan_map:nn { 6 } { 己 }
1030 \zhnum_set_tiangan_map:nn { 7 } { 庚 }
1031 \zhnum_set_tiangan_map:nn { 8 } { 辛 }
1032 \zhnum_set_tiangan_map:nn { 9 } { 壬 }
1033 \zhnum_set_tiangan_map:nn { 10 } { 癸 }
1034 \zhnum_set_dizhi_map:nn { 1 } { 子 }
1035 \zhnum_set_dizhi_map:nn { 2 } { 丑 }
1036 \zhnum_set_dizhi_map:nn { 3 } { 寅 }
1037 \zhnum_set_dizhi_map:nn { 4 } { 卯 }
1038 \zhnum_set_dizhi_map:nn { 5 } { 辰 }
1039 \zhnum_set_dizhi_map:nn { 6 } { 巳 }
1040 \zhnum_set_dizhi_map:nn { 7 } { 午 }
1041 \zhnum_set_dizhi_map:nn { 8 } { 未 }
1042 \zhnum_set_dizhi_map:nn { 9 } { 申 }
1043 \zhnum_set_dizhi_map:nn { 10 } { 酉 }
1044 \zhnum_set_dizhi_map:nn { 11 } { 戌 }
1045 \zhnum_set_dizhi_map:nn { 12 } { 亥 }
1046 </config>
```

代码索引

意大利体的数字表示描述对应索引项的页码；带下划线的数字表示定义对应索引项的代码行号；罗马字体的数字表示使用对应索引项的代码行号。

Symbols	
\\	5, 6, 7
A	
activechar	4
arabicsep	3
\AtEndOfPackage	<u>717, 737</u>
B	
bingint internal commands:	
_bingint_read_do:nn	6
bool commands:	
_bool_if:NTF	<u>213, 226, 358, 444, 626, 646, 663, 677, 815, 840</u>
_bool_lazy_all:nTF	240
_bool_lazy_and:nnTF	<u>231, 248</u>
_bool_lazy_any:nTF	791
_bool_new:N	<u>784, 845</u>
_bool_set_false:N	<u>758, 864, 869, 870, 878, 883</u>
_bool_set_true:N	<u>629, 846, 865, 874, 875, 877, 882</u>
_c_false_bool	<u>218, 219, 802</u>
_c_true_bool	<u>203, 218, 219, 798</u>
\BooleanFalse	296
\BooleanTrue	294
C	
char commands:	
_char_set_catcode_active:n	830
clist commands:	
_clist_map_function:nN	470
_clist_map_inline:nn	<u>571, 579, 584, 698</u>
cs commands:	
_cs:w	336
_cs_end:	344
_cs_generate_variant:Nn	<u>35, 187, 206, 222, 297, 370, 462, 509, 511, 536, 746</u>
_cs_if_exist_use:N	486
_cs_if_exist_use:NTF	<u>457, 492</u>
_cs_new:Npn	<u>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, 272, 298, 306, 318, 329, 334, 352, 362, 364, 371, 373, 385, 393, 399, 404, 417, 429, 433, 448, 453, 455, 460, 483, 488, 499, 512, 514, 516, 518, 520, 702, 704</u>
_cs_new_eq:NN	465,
<u>510, 715, 716, 721, 722, 725, 798, 799, 802, 803, 806, 812</u>	
_cs_new_nopar:Npn	<u>111, 278, 293, 295, 354, 356, 436, 442</u>
_cs_new_protected:Npn	<u>474, 598, 600, 605, 607, 612, 614, 632, 638, 657, 700, 739, 747, 754, 768, 775, 780, 807, 834</u>
_cs_new_protected_nopar:Npn	<u>620, 652, 673, 681, 814, 816, 843</u>
D	
\cs_new_protected_nopar:Npx	726
\cs_set:Npn	472
\cs_set_eq:NN	<u>723, 738, 889, 891</u>
E	
else commands:	
_else:	<u>126, 138, 169, 311, 340, 813</u>
encoding	<u>1, 848</u>
exp commands:	
_exp:w	<u>117, 125, 128, 137, 149, 303, 310, 327</u>
_exp_after:wN	<u>114, 116, 124, 125, 127, 129, 130, 136, 137, 139, 140, 148, 168, 170, 190, 193, 194, 195, 300, 302, 309, 310, 312, 313, 314, 315, 324, 326</u>
_exp_args:Nc	91
_exp_args:Nf	<u>174, 176, 703</u>
_exp_args:No	636
_exp_end_continue_f:w	<u>117, 125, 128, 137, 149, 303, 310, 327</u>
_exp_not:N	<u>123, 135, 167, 308, 324, 433, 545, 546, 547, 548, 549, 550, 551, 557, 558, 559, 560, 561, 563, 569, 575, 576, 582, 591, 664, 665, 670, 684, 686, 688, 690, 692, 694, 696, 699, 708, 711, 712, 720, 728, 729, 730, 731, 732, 733, 734, 735</u>
_exp_not:n	596,
<u>646, 647, 663, 677, 678, 684, 686, 688, 690, 692, 694, 696</u>	
_exp_stop:f:	<u>118, 167, 192, 197, 304, 375</u>
F	
fi commands:	
_fi:	<u>123, 131, 141, 171, 196, 308, 316, 324, 342, 383, 847</u>
file commands:	
_file_if_exist:nTF	756
_file_input:n	762
G	
ganzhi-cyclic	4
group commands:	
_group_begin:	<u>22, 83, 267, 288, 537, 760</u>
_group_end:	<u>25, 86, 270, 291, 595, 764</u>
I	
if commands:	
_if:w	<u>123, 135, 308, 324</u>
_if_case:w	192, 375
_if_int_compare:w	<u>167, 338</u>
_if_predicate:w	805
_IfBooleanT	349
_IfBooleanTF	339
_IfNoValueF	471
_IfNoValueTF	<u>15, 76, 260, 281</u>

int commands:

\int_compare:nNnTF 100, 103,
 157, 160, 179, 182, 210, 216, 225, 228, 229, 237, 238,
 256, 331, 387, 395, 401, 485, 490, 501, 504, 522, 525, 705
 \int_compare_p:n 250
 \int_compare_p:nNn 233, 242, 243
 \int_div_truncate:nn 191, 409, 411, 412, 413, 422, 424, 439
 \int_eval:n 101, 106, 154,
 183, 220, 278, 402, 478, 506, 513, 515, 517, 519, 530, 562
 \int_if_exist:NTF 90, 274
 \int_incr:N 476, 480
 \int_mod:nn ... 176, 406, 419, 440, 461, 495, 523, 530, 703
 \int_new:N 463, 832, 833
 \int_set:Nn 464, 820, 821, 823, 825, 826
 \int_step_function:nnN 828
 \int_step_inline:nn 553, 706
 \int_step_inline:nnn 566
 \int_to_arabic:n 360, 446
 \int_value:w 115, 191, 301
 \int_zero:N 469
 \c_one_int 160, 242, 331, 485
 \l_tmpa_int 469, 476, 478
 \c_zero_int 100, 103, 119, 157, 182, 210, 216, 225, 228,
 229, 237, 238, 243, 256, 338, 490, 501, 504, 522, 525, 705

K

keys commands:

\l_keys_choice_tl 854
 \keys_define:nn 596, 848
 \keys_set:nn 23, 84, 268, 289, 901, 904, 909, 910

M

msg commands:

\msg_error:nn 11
 \msg_error:nnn 766, 860
 \msg_expandable_error:nnn 95
 \msg_new:nnn 3, 96
 \msg_new:nnnn 785, 896

N

\NewDocumentCommand 20, 81, 265, 286, 467, 899
 null 3, 848

O

or commands:

\or: 193, 194, 195, 377, 378, 379, 380, 381, 382

P

prg commands:

\prg_do_nothing: 806
 \ProcessKeysOptions 905

prop commands:

\prop_clear:N 777
 \prop_get:NnNTF 635, 640, 642, 659
 \prop_gset_eq:NN 781
 \prop_if_exist:NTF 749
 \prop_map_function:NN 622, 623, 654

\prop_map_inline:Nn 719
 \prop_new:N 616, 617, 618, 619, 778
 \prop_put:Nnn 599, 603, 606, 610, 613, 615
 \prop_put_if_new:Nnn 602, 609
 \prop_set_eq:NN 752

Q

quark commands:

\q_mark 40, 42
 \q_nil 8, 28, 32, 40, 204
 \quark_if_nil:nTF 31, 38, 44
 \quark_if_recursion_tail_stop:N 209, 323
 \quark_if_recursion_tail_stop_do:Nn 147
 \q_recursion_stop 118, 143, 149, 204, 304
 \q_recursion_tail 8, 118, 204, 304
 \q_stop 28, 29, 32, 36, 40, 42, 348, 350, 352, 372, 373, 430, 433

R

\RequirePackage 12
 reset 4, 848

S

str commands:

\cColonStr 433
 \str_case:nnTF 818
 \str_fold_case:n 854
 \str_if_empty:NTF 906
 \str_if_eq:NNTF 742
 \str_new:N 782, 783, 895
 \str_set:Nn 809, 836, 837, 853
 \str_set_eq:NN 750, 810

style 3, 848

sys commands:

\sys_if_engine_luatex_p: 794
 \sys_if_engine_uptex_p: 795
 \sys_if_engine_xetex_p: 793

T

TeX and L^AT_EX 2 _{ϵ} commands:

\@ifpackagelater 10
 \zhdig 278
 \zhnum 111
 \pagenumbering 2, 2
 \tiangan 4
 \zhdate 2
 \zhdig 1, 2, 4
 \zhdigits 1, 1, 3, 3, 4
 \zhdizhi 2
 \zhganzhi 2
 \zhganzhinian 3
 \zhnum 1, 2, 4
 \zhnumber 1, 1, 3, 4
 \zhnumExtendScaleMap 3
 \zhnumsetup 1, 3, 4
 \zhtiangan 2
 \zhtime 2
 \zhweekday 2

tex commands:

\tex_day:D	355
\tex_ignorespaces:D	902
\tex_month:D	355
\tex_time:D	439, 440
\tex_year:D	355
time	3
tl commands:	
\tl_const:Nn	466, 699, 708, 716, 720
\tl_count:n	174
\tl_if_blank:nTF	51, 64, 70
\tl_if_exist:NTF	479
\tl_put_right:Nn	543, 555, 568, 573, 581, 590
\tl_set:Nn	477, 481, 538, 643, 644, 650, 675, 683, 685, 687, 689, 691, 693, 695, 701, 709, 723
\tl_set_eq:NN	738
\l_tmpa_tl	477, 479, 481, 538, 543, 555, 568, 573, 581, 590, 596, 635, 636, 640, 647, 659
\l_tmpb_tl	642, 643, 647
tl internal commands:	
__tl_act>NNNnn	7
\TrimSpaces	467

U

use commands:

\use:N	454, 502, 505, 506, 523, 527
\use:n	117, 125, 137, 149, 193, 303, 310, 327, 431, 593
\use_i:nn	168, 211, 799
\use_i:nnn	195
\use_i_ii:nnn	194
\use_ii:nn	170, 803
\use_none:n	251

Z

\zhcurrtime	2, 436
\zhdate	2, 346
\zhdig	2, 4, 258
\zhdigits	1, 4, 279
\zhdigithswoptions	279
\zhdigwithoptions	262, 265
\zhdizhi	2, 514
\zhganzhi	2, 516
\zhganzhinian	3, 518
\zhnum	2, 4, 74

zhnum commands:

\zhnum_assgin_const:	628, 652
\zhnum_assgin_const_tl:Nn	661, 669, 716, 723
\zhnum_assgin_ganzhi:nn	623, 700
\zhnum_blank_to_zero:n	46, 48, 56, 58, 63, 68
\zhnum_check_financial:nn	620
\zhnum_check_simp:nn	620
\zhnum_counter:n	77, 85, 88
\zhnum_decimal:nn	33, 61
\zhnum_digit_map:n	180, 227, 234, 235, 251, 252, 256, 453
\zhnum_digits:Nn	282, 290, 294, 296, 298
\zhnum_digits_counter:n	261, 269, 272
\zhnum_digits_null:n	275, 278, 293, 359

\zhnum_digits_zero:n	66, 293
\zhnum_ganzhi:nnn	510, 513, 515, 517, 889, 891
\zhnum_ganzhi_cyclic:nnn	488, 889
\zhnum_ganzhi_nian:n	519, 520
\zhnum_ganzhi_normal:nnn	483, 510, 891
\zhnum_if_digit:NTF	145, 165, 320
\zhnum_if_unicode_engine:TF	791, 908
\zhnum_if_unicode_engine_p:	791, 805
\zhnum_input_cfg:n	751, 754
\zhnum_int:n	88, 111, 359, 445
\zhnum_integer:n	10, 39, 112
\zhnum_load_cfg:n	739, 844, 855
\zhnum_number:n	16, 24, 27, 53, 72
\zhnum_parse_config:	620, 744, 812
\zhnum_parse_number:n	101, 106, 173
\zhnum_parse_number:nn	162, 173
\zhnum_process_number:NNNNNN	214, 223
\zhnum_reset_config:	805, 885
\zhnum_scale_map:n	215, 455
\zhnum_scale_map_hook:n	458, 465, 472
\zhnum_scale_map_loop:n	455
\zhnum_set_active:	815, 816
\zhnum_set_alias:	655, 725
\zhnum_set_alias:NN	725, 728, 730, 732, 734, 738
\zhnum_set_catcode:	761, 805
\zhnum_set_cfg_name:Nn	741, 805
\zhnum_set_digits_map:nn	
.....	598, 919, 921, 923, 926, 928, 929, 930, 931, 932, 933, 934, 935, 936, 937, 938, 939, 940, 941, 942, 943, 949, 951, 952, 960, 961, 963, 964, 965, 966, 967, 975, 976, 978, 984, 986, 987, 988, 994, 996, 997
\zhnum_set_digits_map:nnn	598, 915, 916, 945, 946, 954, 955, 956, 957, 969, 970, 971, 972, 980, 981, 990, 991
\zhnum_set_dizhi_map:nn	598, 1034, 1035, 1036, 1037, 1038, 1039, 1040, 1041, 1042, 1043, 1044, 1045
\zhnum_set_financial_map:nn	
.....	598, 998, 999, 1000, 1001, 1007, 1009, 1010, 1016, 1018, 1019, 1020, 1021, 1022, 1023
\zhnum_set_financial_map:nnn	
.....	598, 1003, 1004, 1012, 1013
\zhnum_set_scale:n	470, 474
\zhnum_set_tiangan_map:nn	598, 1024, 1025, 1026, 1027, 1028, 1029, 1030, 1031, 1032, 1033
\zhnum_set_week_day:	620
\zhnum_set_zero:	620
\zhnum_split_number:nn	183, 200, 201
\zhnum_split_number:NNnNNNNw	203, 207, 218, 219
\zhnum_two_digits:n	385
\zhnum_update_cfg:n	743, 747
\zhnum_Zeller:nnn	375, 385
\zhnum_Zeller_aux:Nnnn	385
\zhnum_Zeller_Gregorian:nnn	389, 404
\zhnum_Zeller_Julian:nnn	390, 417
\zhnum_zero_mod:nn	702, 711, 712

zhnum internal commands:

\l_zhnum_active_char_bool	815, 840, 845, 846, 886
---------------------------------	-------------------------

```

\l_zhnum_ancient_bool ..... 232, 249, 864, 869, 874
\c_zhnum_and_t1 ..... 54
\l_zhnum_arabic_sep_t1 ..... 360, 446, 893
\l_zhnum_byte_max_int ..... 823, 826, 830, 833
\l_zhnum_byte_min_int ..... 820, 821, 825, 829, 832
\l_zhnum_cfg_map_finan_prop ..... 598, 635, 659, 772
\l_zhnum_cfg_map_ganzhi_prop ..... 598, 623, 719, 773
\l_zhnum_cfg_map_prop ..... 598, 622, 654, 770
\l_zhnum_cfg_map_var_prop ..... 598, 640, 642, 771
\l_zhnum_cfg_str ..... 741, 742, 749, 750, 770, 771, 772, 773, 782
\l_zhnum_check_simp_aux:nn ..... 634, 636, 638
\l_zhnum_counter_error:n ..... 92, 94, 276
\l_zhnum_date:www ..... 348, 352
\l_zhnum_date_aux:nnn ..... 353, 355, 356
\l_zhnum_date_aux:Nnnn ..... 360, 362
\l_zhnum_date_aux:NNnnnn ..... 359, 363, 364
\c_zhnum_day_t1 ..... 368
\l_zhnum_day_t1 ..... 696
\c_zhnum_dot_t1 ..... 63, 324
\l_zhnum_encoding_str ..... 809, 810, 818, 836, 839, 844, 853, 855, 895, 906
\l_zhnum_fraction:www ..... 40, 42
\c_zhnum_fri_t1 ..... 382
\l_zhnum_fri_t1 ..... 691
\l_zhnum_ganzhi_cyclic_mod:nnnn ..... 488, 494
\c_zhnum_hour_t1 ..... 450
\c_zhnum_hundred_t1 ..... 235, 732
\l_zhnum_integer_or_fraction:www ..... 32, 36
\l_zhnum_last_cfg_str ..... 742, 750, 783
\l_zhnum_loop_end:wnn ..... 147, 153
\c_zhnum_minus_t1 ..... 105, 161, 332
\l_zhnum_minus_t1 ..... 540
\c_zhnum_minute_t1 ..... 451
\c_zhnum_mon_t1 ..... 378
\l_zhnum_mon_t1 ..... 683
\c_zhnum_month_t1 ..... 367
\l_zhnum_normal_bool ..... 663, 865, 870, 875
\l_zhnum_null_bool ..... 677, 880
\l_zhnum_null_t1 ..... 541, 678
\l_zhnum_number:www ..... 27
\l_zhnum_output:nnwnn ..... 146, 151
\l_zhnum_output_digits>NN ..... 321, 334
\l_zhnum_parse_number:nnn ..... 176, 177
\c_zhnum_parts_t1 ..... 47, 57
\l_zhnum_prop_gset_eq:Nn ..... 763, 780
\l_zhnum_prop_initial:Nn ..... 759, 775
\l_zhnum_read_abs_loop:Nw ..... 139, 143
\l_zhnum_read_digits:w ..... 300, 329
\l_zhnum_read_digits_loop>NN ..... 313, 318, 326
\l_zhnum_read_integer:www ..... 114, 155
\l_zhnum_read_sign_loop:N ..... 116, 121, 124
\l_zhnum_read_sign_loop>NN ..... 302, 306, 309
\l_zhnum_read_zeros_loop:N ..... 129, 133, 136
\l_zhnum_reset_bool ..... 626, 629, 758, 784
\l_zhnum_result:nn ..... 119, 151, 152, 153
\c_zhnum_sat_t1 ..... 376
\l_zhnum_sat_t1 ..... 693
\l_zhnum_scale_int ..... 461, 463, 464, 480
\l_zhnum_simp_bool ..... 646, 877, 878
\l_zhnum_split_number_aux:nnn ..... 184, 188
\l_zhnum_split_number_aux:wwn ..... 190, 199
\c_zhnum_sun_t1 ..... 377
\l_zhnum_sun_t1 ..... 695
\c_zhnum_ten_t1 ..... 254, 730
\c_zhnum_thousand_t1 ..... 227, 734
\c_zhnum_thu_t1 ..... 381
\l_zhnum_thu_t1 ..... 689
\l_zhnum_time:ww ..... 430, 433
\l_zhnum_time_aux:nn ..... 435, 438, 442
\l_zhnum_time_aux:Nnnn ..... 442
\l_zhnum_time_bool ..... 358, 444, 882, 883
\c_zhnum_tue_t1 ..... 379
\l_zhnum_tue_t1 ..... 685
\l_zhnum_update_cfg_prop:N ..... 752, 759, 763, 768
\c_zhnum_wed_t1 ..... 380
\l_zhnum_wed_t1 ..... 687
\l_zhnum_week_day:www ..... 350, 372, 373
\c_zhnum_weekday_t1 ..... 684, 686, 688, 690, 692, 694, 696
\c_zhnum_year_t1 ..... 366
\l_zhnum_Zeller_aux:Nnnn ..... 389, 390, 393
\l_zhnum_zero_mod_aux:nn ..... 703, 704
\c_zhnum_zero_t1 ..... 65, 71, 108, 158, 213, 226, 229, 238, 728
\zhnumber ..... 1, 4, 13
\zhnumberwithoptions ..... 17, 20, 74
\zhnumExtendScaleMap ..... 3, 467
\zhnumsetup ..... 3, 899
\zhnumwithoptions ..... 78, 81
\zhtiangan ..... 2, 512
\zhtime ..... 2, 429
\zhtoday ..... 2, 354
\zhweekday ..... 2, 371

```