

zhnumber 宏包

李清

sobenlee@gmail.com

2022/07/14 v3.0*

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 L^AT_EX3 项目的 expl3, xparse 和 l3keys2e 宏包。

第 2 节 使用方法

`encoding` `encoding = <GBK|Big5|UTF8>`

Updated: 2022-07-10 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 upL^AT_EX、X₃L^AT_EX 和 LuaL^AT_EX, 不用指定编码, 宏包将自动使用 UTF8 编码。只有 L^AT_EX 和 pdfL^AT_EX 需要指定编码, 如果没有指定, 默认也使用 UTF8。

`\zhnumber` * `\zhnumber <{number}>`

Updated: 2022-07-10 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}>`

`\zhdigits` * `\zhdigits * <{number}>`

Updated: 2022-07-10

将阿拉伯数字转换为中文数字串。缺省状态下, \zhdigits 将 0 映射为〇, 如果需要将其映射为零, 可以使用带星号的形式。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}
```

*ctex-kit rev. 5e8c3fe.

<code>\zhnum</code>	*	<code>\zhnum {<counter>}</code>	
Updated: 2022-07-10		<code>\pagenumbering {zhnum}</code>	
		与 <code>\roman</code> 等类似,用于将 L ^A T _E X 计数器的值转换为中文数字。例如	
		二	1 <code>\zhnum{section}</code>
<code>\zhdig</code>	*	<code>\zhdig {<counter>}</code>	
Updated: 2022-07-10		<code>\pagenumbering {zhdig}</code>	
		与 <code>\roman</code> 等类似,用于将 L ^A T _E X 计数器的值转换为中文数字串。例如	
		二	1 <code>\zhdig{section}</code>
<code>\zhweekday</code>	*	<code>\zhweekday {<yyyy/mm/dd>}</code>	
Updated: 2022-07-10		输出日期当天的星期。例如	
		星期日	1 <code>\zhweekday{2012/5/20}</code>
<code>\zhdate</code>	*	<code>\zhdate {<yyyy/mm/dd>}</code>	
Updated: 2022-07-10		<code>\zhdate * {<yyyy/mm/dd>}</code>	
		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 <code>\zhdate{2012/5/21}\</code>
		2012 年 5 月 21 日星期一	2 <code>\zhdate*{2012/5/21}</code>
<code>\zhtoday</code>	*	与 <code>\today</code> 类似,以中文输出当天的日期。例如	
Updated: 2022-07-10		2022 年 7 月 14 日	1 <code>\zhtoday</code>
<code>\zhtime</code>	*	<code>\zhtime {<hh:mm>}</code>	
Updated: 2022-07-10		以中文格式输出时间。例如	
		23 时 56 分	1 <code>\zhtime{23:56}</code>
<code>\zhcurrtime</code>	*	输出当前的时间。例如	
Updated: 2022-07-10		18 时 55 分	1 <code>\zhcurrtime</code>
<code>\zhtiangan</code>	*	<code>\zhtiangan {<number>}</code>	
Updated: 2022-07-10		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 <code>\zhtiangan{1} \zhtiangan{2} \zhtiangan{3}</code> 2 <code>\zhtiangan{4} \zhtiangan{5} \zhtiangan{10}</code>
<code>\zhdizhi</code>	*	<code>\zhdizhi {<number>}</code>	
Updated: 2022-07-10		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 <code>\zhdizhi{1} \zhdizhi{2} \zhdizhi{3}</code> 2 <code>\zhdizhi{4} \zhdizhi{5} \zhdizhi{12}</code>
<code>\zhganzhi</code>	*	<code>\zhganzhi {<number>}</code>	
Updated: 2022-07-10		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 <code>\zhganzhi{1} \zhganzhi{2} \zhganzhi{3} \</code> 2 <code>\zhganzhi{4} \zhganzhi{5} \zhganzhi{60}</code>

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

Updated: 2022-07-10

输出公元纪年 $\langle year \rangle$ 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 壬寅

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

`\zhnumExtendScaleMap` `\zhnumExtendScaleMap [<character>] {<character12n>}`

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`。 $\langle character_i \rangle$ 设置 $10^{4(i+11)}$ 。若给出可选项 $\langle character \rangle$, 则当数字大于 $10^{4(n+12)} - 1$ 时, 统一用 $\langle character \rangle$ 设置输出数字的进位。

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

Updated: 2022-07-10

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 $\langle key \rangle$ 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, $\langle Arabic \rangle$ 为阿拉伯数字, 而 $\langle Chinese \rangle$ 为中文数字。例如

二〇二二年七月十四日十八时五十五分

```
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` `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, 以佰输出 200。

可以设置 `style` 为其中一个, 也可以是前两个与后三个的适当组合, 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, 其他的格式转换命令, 将 0 映射成零, 如果需要将 0 映射成 O, 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，`\tangan` 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 \zhtangan{11} \zhtangan{12} \zhtangan{209}
3 \zhtangan{-1} \zhtangan{-2} \zhtangan{-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 0，`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 ，`F n` 设置数字 n 的大写，`T n` 设置数字 n 的天干，`D n` 设置数字 n 的地支，而 `GZ n` 设置数字 n 的干支。其他的选项同字面意思，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: 2022-07-10

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 `LATEX` 或者 `pdfLATEX` 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      \zhnumber    [{options}] {<number>}
\zhdigits      \zhdigits *  [{options}] {<number>}
\zhnum         \zhnum       [{options}] {<counter>}
\zhdig         \zhdig      [{options}] {<counter>}

```

Updated: 2022-07-10

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   'l3kernel'~and~'l3packages'\\
8   using~your~TeX~package~manager~or~from~CTAN.
9 }
10 \@ifpackagelater { expl3 } { 2019/03/05 } { }
11 { \msg_error:nn { zhnumber } { l3-too-old } }

12 \cs_if_exist:NF \NewDocumentCommand
13 { \RequirePackage { xparse } }

```

`\zhnum_output:n` 受益于 `\tex_expanded:D`, 我们使用如下的结构简单实现 `f` 展开。
`\zhnum_expand_wrap:wn` `\exp_not:e`

```

{
...
\exp_not:o { <tl_var> }
...
\exp_not:o { <tl_var> }
...
\exp_not:o { <tl_var> }
...
}

```

并且可以将 `\zhnumber` 等无风险地使用于诸如 `\edef` 定义等完全展开的场合。从 `TEX Live 2019` 开始, 各个主要引擎都已经支持 `\tex_expanded:D`, 对于较早的版本, `LATEX 3` 也实现了一个模拟。

```

14 \cs_new:Npn \zhnum_output:n #1
15 { \exp_args:Nc \zhnum_exp_not:o { l__zhnum_ #1 _tl } }
16 \cs_new_protected:Npn \zhnum_expand_wrap:wn #1#
17 { \__zhnum_expand_wrap_aux:nn {#1} }
18 \cs_new_protected:Npn \__zhnum_expand_wrap_aux:nn #1#2
19 { #1 { \exp_not:e {#2} } }
20 \cs_new_eq:NN \zhnum_exp_not:e \exp_not:e
21 \cs_new_eq:NN \zhnum_exp_not:o \exp_not:o

```

但这种方式也有一定的缺点, 比如在 `pdfTEX` 引擎下, `hyperref` 包的 `\pdfstringdef` 就需要把汉字首字节的活动字符展开, 并作适当的重定义后才能得到预期的结果。为此, 我们定义一个清除命令, 不使用以上机制。

```

22 \cs_new_protected:Npn \zhnumClearWrapper
23 {
24   \cs_set_eq:NN \zhnum_exp_not:e \use:n
25   \cs_set_eq:NN \zhnum_exp_not:o \use:n
26 }
27 \cs_new_protected:Npn \zhnumResetWrapper
28 {
29   \cs_set_eq:NN \zhnum_exp_not:e \exp_not:e
30   \cs_set_eq:NN \zhnum_exp_not:o \exp_not:o
31 }
32 \hook_gput_code:nnn { package/CJKutf8/after } { zhnumber }
33 { \g@addto@macro \pdfstringdefPreHook { \zhnumClearWrapper } }
34 \hook_gput_code:nnn { package/xCJK2uni/after } { zhnumber }
35 { \g@addto@macro \pdfstringdefPreHook { \zhnumClearWrapper } }

```

`\zhnumber` 用于将输入的数字按照中文格式输出。

```

36 \NewExpandableDocumentCommand \zhnumber { +o +m }
37 {
38   \tl_if_novalue:nTF {#1}
39     { \zhnum_number:e }
40     { \zhnumberwithoptions {#1} }
41   {#2}
42 }

```

`\zhnumberwithoptions` 带选项的用户函数。

```

43 \NewDocumentCommand \zhnumberwithoptions { +m +m }
44 {
45   \group_begin:
46     \zhnum_set:n {#1}
47     \zhnum_number:e {#2}
48   \group_end:
49 }

```

`\zhnum_number:n` 先判断输入的是小数还是分数。
`__zhnum_number:www`

```

50 \zhnum_expand_wrap:wn
51 \cs_new:Npn \zhnum_number:n #1
52 { \__zhnum_number:www #1 . \q_nil . \q_stop }
53 \cs_new:Npn \__zhnum_number:www #1 . #2 . #3 \q_stop
54 {
55   \quark_if_nil:nTF {#2}
56     { \__zhnum_integer_or_fraction:www #1 / \q_nil / \q_stop }
57     { \zhnum_decimal:nn {#1} {#2} }
58 }
59 \cs_generate_variant:Nn \zhnum_number:n { e }

```

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

```

60 \cs_new:Npn \__zhnum_integer_or_fraction:www #1 / #2 / #3 \q_stop
61 {
62   \quark_if_nil:nTF {#2}
63     { \zhnum_integer:n {#1} }
64     { \__zhnum_fraction:www #2 \q_mark #1 ; \q_nil ; \q_stop }
65 }

```

`__zhnum_fraction:www` 对分数进行预处理。

```

66 \cs_new:Npn \__zhnum_fraction:www #1 \q_mark #2 ; #3 ; #4 \q_stop
67 {
68   \quark_if_nil:nTF {#3}
69     {
70       \zhnum_blank_to_zero:n {#1}
71       \zhnum_output:n { parts }
72       \zhnum_blank_to_zero:n {#2}
73     }
74     {
75       \tl_if_blank:nF {#2}
76       {
77         \zhnum_number:n {#2}
78         \zhnum_output:n { and }
79       }
80       \zhnum_blank_to_zero:n {#1}
81       \zhnum_output:n { parts }
82       \zhnum_blank_to_zero:n {#3}
83     }
84 }

```

`\zhnum_decimal:nn` 对小数进行预处理。

```

85 \cs_new:Npn \zhnum_decimal:nn #1#2
86 {
87   \zhnum_blank_to_zero:n {#1}
88   \zhnum_output:n { dot }
89   \tl_if_blank:nTF {#2}
90     { \zhnum_output:n { 0 } }
91     { \zhnum_digits_zero:n {#2} }
92 }

```

`\zhnum_blank_to_zero:n` 输出小数的整数位。

```

93 \cs_new:Npn \zhnum_blank_to_zero:n #1
94 {

```

```

95   \tl_if_blank:nTF {#1}
96     { \zhnum_output:n { 0 } }
97     { \zhnum_number:n {#1} }
98   }

```

`\zhnum` 用于将 L^AT_EX 计数器按中文格式输出。

`\zhnumberwithoptions`

```

99 \NewExpandableDocumentCommand \zhnum { +o +m }
100 {
101   \tl_if_novalue:nTF {#1}
102     { \zhnum_counter:n }
103     { \zhnumwithoptions {#1} }
104   {#2}
105 }
106 \NewDocumentCommand \zhnumwithoptions { +m +m }
107 {
108   \group_begin:
109     \zhnum_set:n {#1}
110     \zhnum_counter:n {#2}
111   \group_end:
112 }

```

`\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`

```

113 \cs_new:Npn \zhnum_counter:n #1
114 {
115   \int_if_exist:cTF { c@#1 }
116     { \exp_args:Nc \zhnum_int:n { c@#1 } }
117     { \__zhnum_counter_error:n {#1} }
118 }
119 \cs_new:Npn \__zhnum_counter_error:n #1
120 { \msg_expandable_error:nnn { zhnumber } { not-counter } {#1} }
121 \msg_new:nnn { zhnumber } { not-counter }
122 { `#1' is not a LATEX counter. }
123 \zhnum_expand_wrap:wn
124 \cs_new:Npn \zhnum_int:n #1
125 {
126   \int_compare:nNnTF {#1} > \c_zero_int
127     { \zhnum_parse_number:f { \int_eval:n {#1} } }
128     {
129       \int_compare:nNnTF {#1} < \c_zero_int
130         {
131           \zhnum_output:n { minus }
132           \zhnum_parse_number:f { \int_eval:n { - #1 } }
133         }
134         { \zhnum_output:n { 0 } }
135       }
136 }

```

`\@zhnum` 用于支持 `\pagenumbering{zhnum}`。

```

137 \cs_new: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。

```

138 \cs_new:Npn \zhnum_integer:n #1
139 {
140   \exp_after:wN \__zhnum_read_integer:www
141   \int_value:w
142   \exp_after:wN \__zhnum_read_sign_loop:N
143   \exp:w \exp_end_continue_f:w \use:n
144   #1 \exp_stop_f: \q_recursion_tail \q_recursion_stop
145   \__zhnum_result:nn { \c_zero_int } { } ;
146 }
147 \cs_new:Npn \__zhnum_read_sign_loop:N #1
148 {

```

```

149 \if:w + \if:w - \exp_not:N #1 + \fi: \exp_not:N #1
150 \exp_after:wN \__zhnum_read_sign_loop:N
151 \exp:w \exp_end_continue_f:w \exp_after:wN \use:n
152 \else:
153 1 \exp_after:wN ;
154 \exp:w \exp_end_continue_f:w
155 \exp_after:wN \__zhnum_read_zeros_loop:N
156 \exp_after:wN #1
157 \fi:
158 }
159 \cs_new:Npn \__zhnum_read_zeros_loop:N #1
160 {
161 \if:w 0 \exp_not:N #1
162 \exp_after:wN \__zhnum_read_zeros_loop:N
163 \exp:w \exp_end_continue_f:w \exp_after:wN \use:n
164 \else:
165 \exp_after:wN \__zhnum_read_abs_loop:Nw
166 \exp_after:wN #1
167 \fi:
168 }

```

__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。

```

169 \cs_new:Npn \__zhnum_read_abs_loop:Nw #1#2 \q_recursion_stop
170 {
171 \zhnum_if_digit:NTF #1
172 { \__zhnum_output:nnwnn { + 1 } #1 }
173 { \quark_if_recursion_tail_stop_do:Nn #1 { \__zhnum_loop_end:wnn } }
174 \exp_after:wN \__zhnum_read_abs_loop:Nw
175 \exp:w \exp_end_continue_f:w \use:n #2 \q_recursion_stop
176 }
177 \cs_new:Npn \__zhnum_output:nnwnn #1#2#3 \__zhnum_result:nn #4#5
178 { #3 \__zhnum_result:nn { #4#1 } { #5#2 } }
179 \cs_new:Npn \__zhnum_loop_end:wnn #1 \__zhnum_result:nn #2#3
180 { \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。

```

181 \cs_new:Npn \__zhnum_read_integer:www #1 ; #2 ; #3 ;
182 {
183 \int_compare:nNnTF {#2} = \c_zero_int
184 { \zhnum_output:n { 0 } }
185 {
186 \int_compare:nNnF {#1} = \c_one_int
187 { \zhnum_output:n { minus } }
188 \zhnum_parse_number:nn {#2} {#3}
189 }
190 }

```

\zhnum_if_digit:NTF 判断 #1 是否为数字位。

```

191 \cs_new:Npn \zhnum_if_digit:NTF #1
192 {
193 \if_int_compare:w 9 < 1 \exp_not:N #1 \exp_stop_f:
194 \exp_after:wN \use_i:nn
195 \else:
196 \exp_after:wN \use_ii:nn
197 \fi:
198 }

```

```

\zhnum_parse_number:n 199 \cs_new:Npn \zhnum_parse_number:n #1
\zhnum_parse_number:nn 200 { \exp_args:Nf \zhnum_parse_number:nn { \tl_count:n {#1} } {#1} }
201 \cs_new:Npn \zhnum_parse_number:nn #1
202 { \exp_args:Nf \__zhnum_parse_number:nnn { \int_mod:nn {#1} { 4 } } {#1} }
203 \cs_new:Npn \__zhnum_parse_number:nnn #1#2
204 {

```



```

205 \int_compare:nNnTF {#2} < 2
206 { \zhnum_output:n }
207 {
208   \int_compare:nNnTF {#1} = \c_zero_int
209   { \zhnum_split_number:fn { \int_eval:n { #2 / 4 - 1 } } }
210   { \__zhnum_split_number_aux:nnn {#1} {#2} }
211 }
212 }
213 \cs_generate_variant:Nn \zhnum_parse_number:n { f }

```

__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。

```

214 \cs_new:Npn \__zhnum_split_number_aux:nnn #1#2
215 {
216   \exp_after:wN \__zhnum_split_number_aux:wnn
217   \int_value:w \int_div_truncate:nn {#2} { 4 }
218   \if_case:w #1 \exp_stop_f:
219     \or: \exp_after:wN \use:n
220     \or: \exp_after:wN \use_i_ii:nnn
221     \or: \exp_after:wN \use_i:nnn
222   \fi:
223   { \exp_stop_f: ; 0 } 0 0 ;
224 }
225 \cs_new:Npn \__zhnum_split_number_aux:wnn #1 ; #2 ; #3
226 { \zhnum_split_number:nn {#1} { #2#3 } }

```

\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。

```

227 \cs_new:Npn \zhnum_split_number:nn #1#2
228 {
229   \zhnum_split_number:NNnNNNw \c_true_bool \c_true_bool {#1}
230   #2 \q_recursion_tail \q_nil \q_nil \q_nil \q_recursion_stop
231 }
232 \cs_generate_variant:Nn \zhnum_split_number:nn { f }

```

\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。

```

233 \cs_new:Npn \zhnum_split_number:NNnNNNw #1#2#3#4#5#6#7
234 {
235   \quark_if_recursion_tail_stop:N #4
236   \int_compare:nNnTF { #4#5#6#7 } = \c_zero_int
237   { \use_i:nn }
238   {
239     \bool_if:NF #1 { \zhnum_output:n { 0 } }
240     \zhnum_process_number:NNNNNN #4#5#6#7#1#2
241     \zhnum_scale_map:n {#3}
242     \int_compare:nNnTF {#7} = \c_zero_int
243   }
244   { \zhnum_split_number:NNfNNNw \c_false_bool \c_true_bool }
245   { \zhnum_split_number:NNfNNNw \c_true_bool \c_false_bool }
246   { \int_eval:n { #3 - 1 } }
247 }
248 \cs_generate_variant:Nn \zhnum_split_number:NNnNNNw { NNf }

```

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

```

249 \cs_new:Npn \zhnum_process_number:NNNNNN #1#2#3#4#5#6
250 {
251   \int_compare:nNnTF {#1} = \c_zero_int
252   {
253     \bool_if:NF #6
254     { \zhnum_output:n { 0 } }
255   }
256   {
257     \zhnum_output:n {#1}
258     \zhnum_output:n { 1000 }
259   }
260   \int_compare:nNnTF {#2} = \c_zero_int
261   {

```

```

262     \int_compare:nNnF { #1 * (#3#4) } = \c_zero_int
263     { \zhnum_output:n { 0 } }
264   }
265   {
266     \int_compare:nNnTF {#2} = 2
267     { \zhnum_output:n { 200 } }
268     {
269       \zhnum_output:n {#2}
270       \zhnum_output:n { 100 }
271     }
272   }
273   \int_compare:nNnTF {#3} = \c_zero_int
274   {
275     \int_compare:nNnF { #2 * #4 } = \c_zero_int
276     { \zhnum_output:n { 0 } }
277   }
278   {
279     \bool_lazy_all:nTF
280     {
281       { \int_compare_p:nNn {#3} = \c_one_int }
282       { \int_compare_p:nNn {#1#2} = \c_zero_int }
283       {#6}
284       {#5}
285     }
286     { \zhnum_output:n { 10 } }
287     {
288       \int_compare:nTF { 1 < #3 < 5 }
289       { \zhnum_output:n { #3 0 } }
290       {
291         \zhnum_output:n {#3}
292         \zhnum_output:n { 10 }
293       }
294     }
295   }
296   \int_compare:nNnF {#4} = \c_zero_int
297   { \zhnum_output:n {#4} }
298 }

```

`\zhdig` 用于将 L^AT_EX 计数器按中文数字串输出。

```

299 \NewExpandableDocumentCommand \zhdig { +o +m }
300   {
301     \tl_if_novalue:nTF {#1}
302     { \zhnum_digits_counter:n }
303     { \zhdigwithoptions {#1} }
304     {#2}
305   }
306 \NewDocumentCommand \zhdigwithoptions { +m +m }
307   {
308     \group_begin:
309     \zhnum_set:n {#1}
310     \zhnum_digits_counter:n #1 {#2}
311     \group_end:
312   }
313 \cs_new:Npn \zhnum_digits_counter:n #1
314   {
315     \int_if_exist:cTF { c@#1 }
316     { \zhnum_digits_null:v { c@#1 } }
317     { \__zhnum_counter_error:n {#1} }
318   }

```

`\@zhdig` 用于支持 `\pagenumbering{zhdig}`。

```

319 \cs_new:Npn \@zhdig #1
320   { \zhnum_digits_null:f { \int_eval:n {#1} } }

```

`\zhdigits` 将输入的数字输出为中文数字串输出。

```

\zhdigitswithoptions 321 \NewExpandableDocumentCommand \zhdigits { +s +o +m }

```

```

322 {
323   \tl_if_novalue:nTF {#2}
324     { \zhnum_digits:Ne #1 }
325     { \zhdigitswithoptions {#1} {#2} }
326     {#3}
327 }
328 \NewDocumentCommand \zhdigitswithoptions { +m +m +m }
329 {
330   \group_begin:
331     \zhnum_set:n {#2}
332     \zhnum_digits:Ne #1 {#3}
333   \group_end:
334 }

```

\zhnum_digits_zero:n 快捷方式。

```

\zhnum_digits_null:n
335 \cs_new:Npn \zhnum_digits_zero:n
336   { \zhnum_digits:Nn \c_true_bool }
337 \cs_new:Npn \zhnum_digits_null:n
338   { \zhnum_digits:Nn \c_false_bool }
339 \cs_generate_variant:Nn \zhnum_digits_null:n { v , f }

```

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

```

340 \zhnum_expand_wrap:wn
341 \cs_new:Npn \zhnum_digits:Nn #1#2
342 {
343   \exp_after:wN \__zhnum_read_digits:w
344   \int_value:w
345   \exp_after:wN \__zhnum_read_sign_loop:NN \exp_after:wN #1
346   \exp:w \exp_end_continue_f:w \use:n
347   #2 \exp_stop_f: \q_recursion_tail \q_recursion_stop
348 }
349 \cs_new:Npn \__zhnum_read_sign_loop:NN #1#2
350 {
351   \if:w + \if:w - \exp_not:N #2 + \fi: \exp_not:N #2
352   \exp_after:wN \__zhnum_read_sign_loop:NN \exp_after:wN #1
353   \exp:w \exp_end_continue_f:w \exp_after:wN \use:n
354   \else:
355     1 \exp_after:wN ;
356     \exp_after:wN \__zhnum_read_digits_loop:NN
357     \exp_after:wN #1
358     \exp_after:wN #2
359   \fi:
360 }
361 \cs_new:Npn \__zhnum_read_digits_loop:NN #1#2
362 {
363   \zhnum_if_digit:NTF #2
364     { \__zhnum_output_digits:NN #1#2 }
365     {
366       \quark_if_recursion_tail_stop:N #2
367       \token_if_eq_charcode:NNT #2 .
368       { \zhnum_output:n { dot } }
369     }
370   \exp_after:wN \__zhnum_read_digits_loop:NN \exp_after:wN #1
371   \exp:w \exp_end_continue_f:w \use:n
372 }
373 \cs_new:Npn \__zhnum_read_digits:w #1 ;
374 {
375   \int_compare:nNnF {#1} = \c_one_int
376     { \zhnum_output:n { minus } }
377 }
378 \cs_new:Npn \__zhnum_output_digits:NN #1#2
379 {
380   \zhnum_output:n
381   {
382     \if_int_compare:w #2 = \c_zero_int
383     \bool_if:NTF #1 { zero } { null }

```

```

384         \else:
385         #2
386         \fi:
387     }
388 }
389 \cs_generate_variant:Nn \zhnum_digits:Nn { Ne }

```

`\zhdate` 输出中文日期。

```

390 \NewExpandableDocumentCommand \zhdate { +s +m }
391 {
392     \__zhnum_date:www #2 \q_stop
393     \bool_if:NT #1 { \__zhnum_week_day:www #2 \q_stop }
394 }
395 \cs_new:Npn \__zhnum_date:www #1/#2/#3 \q_stop
396 { \__zhnum_date_aux:nnn {#1} {#2} {#3} }

```

`\zhtoday` 输出当天日期。

```

397 \cs_new:Npn \zhtoday
398 {
399     \__zhnum_date_aux:nnn
400     { \int_value:w \tex_year:D }
401     { \tex_month:D }
402     { \tex_day:D }
403 }

```

```

\__zhnum_date_aux:nnn 404 \cs_new:Npn \__zhnum_date_aux:nnn
405 {
406     \bool_if:NTF \l__zhnum_time_bool
407     { \__zhnum_date_aux:NNnnnn \zhnum_digits_null:n \zhnum_int:n { } }
408     { \__zhnum_date_aux:Nnnnn \int_to_arabic:n { \l__zhnum_arabic_sep_tl } }
409 }
410 \cs_new:Npn \__zhnum_date_aux:Nnnnn #1#2
411 { \__zhnum_date_aux:NNnnnn #1#1 { \zhnum_exp_not:o {#2} } }
412 \zhnum_expand_wrap:wn
413 \cs_new:Npn \__zhnum_date_aux:NNnnnn #1#2#3#4#5#6
414 {
415     #1 {#4} #3 \zhnum_output:n { year } #3
416     #2 {#5} #3 \zhnum_output:n { month } #3
417     #2 {#6} #3 \zhnum_output:n { day }
418 }

```

`\zhweekday` 输出星期

```

419 \cs_new:Npn \zhweekday #1
420 { \__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 $d = h + 5 \pmod{7} + 1$ 。

```

421 \zhnum_expand_wrap:wn
422 \cs_new:Npn \__zhnum_week_day:www #1/#2/#3 \q_stop
423 {
424     \if_case:w \zhnum_Zeller:nnn {#1} {#2} {#3} \exp_stop_f:
425         \zhnum_output:n { sat }
426         \or: \zhnum_output:n { sun }
427         \or: \zhnum_output:n { mon }
428         \or: \zhnum_output:n { tue }
429         \or: \zhnum_output:n { wed }
430         \or: \zhnum_output:n { thu }
431         \or: \zhnum_output:n { fri }
432     \fi:
433 }

```

`\zhnum_Zeller:nnn` 用 Zeller 公式¹ 计算星期几。
`\zhnum_Zeller_aux:Nnnn`
`\zhnum_two_digits:n` 434 \cs_new:Npn \zhnum_Zeller:nnn #1#2#3

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

```

435 {
436   \int_compare:nNnTF
437     { #1 \zhnum_two_digits:n {#2} \zhnum_two_digits:n {#3} } > { 1582 10 04 }
438     { \__zhnum_Zeller_aux:Nnnn \zhnum_Zeller_Gregorian:nnn }
439     { \__zhnum_Zeller_aux:Nnnn \zhnum_Zeller_Julian:nnn }
440     {#1} {#2} {#3}
441   }
442 \cs_new:Npn \__zhnum_Zeller_aux:Nnnn #1#2#3#4
443 {
444   \int_compare:nNnTF {#3} < 3
445     { #1 { #2 - 1 } { #3 + 12 } {#4} }
446     { #1 {#2} {#3} {#4} }
447   }
448 \cs_new:Npn \zhnum_two_digits:n #1
449 {
450   \int_compare:nNnT {#1} < { 10 } { 0 }
451   \int_eval:n {#1}
452 }

```

\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 $Y --$ 。

```

453 \cs_new:Npn \zhnum_Zeller_Gregorian:nnn #1#2#3
454 {
455   \int_mod:nn
456     {
457       (#3)
458       + \int_div_truncate:nn { 26 * ( #2 + 1 ) } { 10 }
459       + (#1)
460       + \int_div_truncate:nn {#1} { 4 }
461       + 6 * \int_div_truncate:nn {#1} { 100 }
462       + \int_div_truncate:nn {#1} { 400 }
463     }
464     { 7 }
465 }

```

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

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

```

466 \cs_new:Npn \zhnum_Zeller_Julian:nnn #1#2#3
467 {
468   \int_mod:nn
469     {
470       (#3)
471       + \int_div_truncate:nn { 26 * ( #2 + 1 ) } { 10 }
472       + (#1)
473       + \int_div_truncate:nn {#1} { 4 }
474       + 5
475     }
476     { 7 }
477 }

```

\zhtime 输出时间。

```

478 \zhnum_expand_wrap:wn
479 \cs_new:Npn \zhtime #1
480 { \__zhnum_time:ww #1 \q_stop }
481 \use:e
482 {
483   \cs_new:Npn \exp_not:N \__zhnum_time:ww
484     #1 \c_colon_str #2 \exp_not:N \q_stop
485 }
486 { \__zhnum_time_aux:nn {#1} {#2} }

```

`\zhcurrtime` 输出当前时间。

```

487 \zhnum_expand_wrap:wn
488 \cs_new:Npn \zhcurrtime
489 {
490   \__zhnum_time_aux:nn
491   { \int_div_truncate:nn \tex_time:D { 60 } }
492   { \int_mod:nn \tex_time:D { 60 } }
493 }

\__zhnum_time_aux:nn 494 \cs_new:Npn \__zhnum_time_aux:nn
\__zhnum_time_aux:Nnnn 495 {
496   \bool_if:NTF \l__zhnum_time_bool
497   { \__zhnum_time_aux:Nnnn \zhnum_int:n { } }
498   { \__zhnum_time_aux:Nnnn \int_to_arabic:n { \l__zhnum_arabic_sep_tl } }
499 }
500 \cs_new:Npn \__zhnum_time_aux:Nnnn #1#2#3#4
501 {
502   #1 {#3} #2 \zhnum_output:n { hour } #2
503   #1 {#4} #2 \zhnum_output:n { minute }
504 }

```

`\zhnum_scale_map:n` 大数系统的映射。

```

\zhnum_scale_map_loop:n 505 \cs_new:Npn \zhnum_scale_map:n #1
506 {
507   \tl_if_exist:cTF { l__zhnum_s #1 _tl }
508   { \zhnum_output:n { s #1 } }
509   { \zhnum_scale_map_hook:n {#1} }
510 }
511 \cs_new:Npn \zhnum_scale_map_loop:n #1
512 { \zhnum_scale_map:n { \int_mod:nn {#1} \l__zhnum_scale_int } }
513 \cs_generate_variant:Nn \zhnum_scale_map:n { f }
514 \int_new:N \l__zhnum_scale_int
515 \int_set:Nn \l__zhnum_scale_int { 11 }
516 \cs_new_eq:NN \zhnum_scale_map_hook:n \zhnum_scale_map_loop:n
517 \tl_new:c { l__zhnum_s 0 _tl }

```

`\zhnumExtendScaleMap` 扩展进位系统。

```

518 \NewDocumentCommand \zhnumExtendScaleMap { > { \TrimSpaces } +o +m }
519 {
520   \int_zero:N \l__zhnum_tmp_int
521   \clist_map_function:nN {#2} \zhnum_set_scale:n
522   \tl_if_novalue:nF {#1}
523   { \cs_set:Npn \zhnum_scale_map_hook:n ##1 {#1} }
524 }

```

```

\zhnum_set_scale:n 525 \cs_new_protected:Npn \zhnum_set_scale:n #1
526 {
527   \int_incr:N \l__zhnum_tmp_int
528   \exp_args:Nc \__zhnum_set_scale:Nn
529   { l__zhnum_s \int_eval:n { \l__zhnum_tmp_int + 11 } _tl }
530   {#1}
531 }
532 \cs_new_protected:Npn \__zhnum_set_scale:Nn #1
533 {
534   \tl_if_exist:NF #1
535   {
536     \tl_new:N #1
537     \int_incr:N \l__zhnum_scale_int
538   }
539   \tl_set:Nn #1
540 }
541 \int_new:N \l__zhnum_tmp_int

```

`\zhnum_ganzhi_normal:nnn` 保证干支的参数为正数。

```

542 \zhnum_expand_wrap:wn

```

```

543 \cs_new:Npn \znum_ganzhi_normal:nnn #1#2#3
544 {
545   \int_compare:nNnF {#1} < \c_one_int
546   {
547     \cs_if_free:cF { l__znum_ #2 _ #1 _t1 }
548     { \znum_output:n { #2 _ #1 } }
549   }
550 }

```

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

_znum_ganzhi_cyclic_mod:nnnn

```

551 \znum_expand_wrap:wn
552 \cs_new:Npn \znum_ganzhi_cyclic:nnn #1#2#3
553 {
554   \int_compare:nNnF {#1} = \c_zero_int
555   {
556     \tl_if_exist:cTF { l__znum_ #2 _ #1 _t1 }
557     { \znum_output:n { #2 _ #1 } }
558     {
559       \_znum_ganzhi_cyclic_mod:fnnn
560       { \int_mod:nn {#1} {#3} } {#1} {#2} {#3}
561     }
562   }
563 }
564 \cs_new:Npn \_znum_ganzhi_cyclic_mod:nnnn #1#2#3#4
565 {
566   \int_compare:nNnTF {#2} > \c_zero_int
567   { \znum_output:n { #3 _ #1 } }
568   {
569     \int_compare:nNnTF {#1} = \c_zero_int
570     { \znum_output:n { #3 _ 1 } }
571     { \znum_output:n { #3 _ \int_eval:n { #1 + #4 + 1 } } }
572   }
573 }
574 \cs_generate_variant:Nn \_znum_ganzhi_cyclic_mod:nnnn { f }

```

\znum_ganzhi:nnn 默认不对超出范围的数字取模。

```

575 \cs_new_eq:NN \znum_ganzhi:nnn \znum_ganzhi_normal:nnn
576 \cs_generate_variant:Nn \znum_ganzhi:nnn { f }

```

\zhtiangan 天干。

```

577 \cs_new:Npn \zhtiangan #1
578 { \znum_ganzhi:fnn { \int_eval:n {#1} } { tiangan } { 10 } }

```

\zhdizhi 地支。

```

579 \cs_new:Npn \zhdizhi #1
580 { \znum_ganzhi:fnn { \int_eval:n {#1} } { dizhi } { 12 } }

```

\zhganzhi 干支。

```

581 \cs_new:Npn \zhganzhi #1
582 { \znum_ganzhi:fnn { \int_eval:n {#1} } { ganzhi } { 60 } }

```

\zhganzhinian 干支纪年。

```

583 \cs_new:Npn \zhganzhinian #1
584 { \znum_ganzhi_nian:f { \int_eval:n {#1} } }

```

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

```

585 \znum_expand_wrap:wn
586 \cs_new:Npn \znum_ganzhi_nian:n #1
587 {
588   \int_compare:nNnTF {#1} > \c_zero_int
589   { \znum_output:n { ganzhi_ \int_mod:nn { #1 + 57 } { 60 } } }
590   {
591     \int_compare:nNnF {#1} = \c_zero_int

```

```

592     {
593     \zhnum_output:n
594     {
595     ganzhi_ \int_eval:n
596     { \int_mod:nn { #1 - 2 } { 60 } + 60 }
597     }
598     }
599 }
600 }
601 \cs_generate_variant:Nn \zhnum_ganzhi_nian:n { f }

```

根据需要设置中文阿拉伯数字。

```

602 \tl_new:N \l__zhnum_kv_tl
603 \tl_new:N \l__zhnum_tmp_tl
604 \group_begin:
605 \tl_build_begin:N \l__zhnum_kv_tl
606 \int_step_inline:nn { 10 }
607 {
608 \tl_new:c { l__zhnum_ #1 _tl }
609 \tl_build_put_right:Nx \l__zhnum_kv_tl
610 {
611 #1 .tl_set:N = \exp_not:c { l__zhnum_normal_ #1 _tl } ,
612 F#1 .tl_set:N = \exp_not:c { l__zhnum_financial_ #1 _tl } ,
613 E \int_eval:n { #1 * 4 }
614 .tl_set:N = \exp_not:c { l__zhnum_ s#1 _tl } ,
615 }
616 }
617 \clist_map_inline:nn { 0 , 100 , 1000 }
618 {
619 \tl_new:c { l__zhnum_ #1 _tl }
620 \tl_build_put_right:Nx \l__zhnum_kv_tl
621 {
622 #1 .tl_set:N = \exp_not:c { l__zhnum_normal_ #1 _tl } ,
623 F#1 .tl_set:N = \exp_not:c { l__zhnum_financial_ #1 _tl } ,
624 }
625 }
626 \clist_map_inline:nn
627 {
628 20 , 30 , 40 , 200 ,
629 dot , and , parts , year , month , day , hour , minute
630 }
631 {
632 \tl_build_put_right:Nx \l__zhnum_kv_tl
633 { #1 .tl_set:N = \exp_not:c { l__zhnum_ #1 _tl } , }
634 }
635 \tl_build_put_right:Nx \l__zhnum_kv_tl
636 {
637 - .tl_set:N = \exp_not:N \l__zhnum_minus_tl ,
638 -0 .tl_set:N = \exp_not:N \l__zhnum_null_tl ,
639 E2 .tl_set:N = \exp_not:c { l__zhnum_normal_ 100 _tl } ,
640 E3 .tl_set:N = \exp_not:c { l__zhnum_normal_ 1000 _tl } ,
641 FE2 .tl_set:N = \exp_not:c { l__zhnum_financial_ 100 _tl } ,
642 FE3 .tl_set:N = \exp_not:c { l__zhnum_financial_ 1000 _tl } ,
643 E44 .tl_set:N = \exp_not:c { l__zhnum_ s11 _tl }
644 }
645 \tl_build_get:NN \l__zhnum_kv_tl \l__zhnum_tmp_tl
646 \cs_set:Npn \__zhnum_tmp:w #1 . #2 \q_stop
647 { , #1 .groups:n = { user } }
648 \clist_map_inline:Nn \l__zhnum_tmp_tl
649 {
650 \tl_build_put_right:Nx \l__zhnum_kv_tl
651 { \__zhnum_tmp:w #1 \q_stop }
652 }
653 \tl_build_put_right:Nn \l__zhnum_kv_tl
654 {
655 ,
656 weekday .tl_set:N = \l__zhnum_weekday_tl ,

```



```

657     weekday .groups:n = { user , pre , weekday } ,
658   }
659   \clist_map_inline:nn
660   { mon , tue , wed , thu , fri , sat , sun }
661   {
662     \tl_build_put_right:Nx \l__zhnum_kv_tl
663     {
664       #1 .tl_set:N = \exp_not:c { l__zhnum_ #1 _tl } ,
665       #1 .groups:n = { user , pos , day } ,
666     }
667   }
668   \int_step_inline:nn { 10 }
669   {
670     \tl_build_put_right:Nx \l__zhnum_kv_tl
671     {
672       T#1 .tl_set:N = \exp_not:c { l__zhnum_ganzhi_ #1 _tl } ,
673       T#1 .groups:n = { user , pre , tiandi } ,
674     }
675   }
676   \int_step_inline:nn { 12 }
677   {
678     \tl_build_put_right:Nx \l__zhnum_kv_tl
679     {
680       D#1 .tl_set:N = \exp_not:c { l__zhnum_dizhi_ #1 _tl } ,
681       D#1 .groups:n = { user , pre , tiandi } ,
682     }
683   }
684   \int_step_inline:nn { 60 }
685   {
686     \tl_build_put_right:Nx \l__zhnum_kv_tl
687     {
688       GZ#1 .tl_set:N = \exp_not:c { l__zhnum_ganzhi_ #1 _tl } ,
689       GZ#1 .groups:n = { user , pos , ganzhi } ,
690     }
691   }
692   \tl_build_end:N \l__zhnum_kv_tl
693   \exp_args:NNno \group_end:
694   \keys_define:nn
695   { zhnum / options }
696   { \l__zhnum_kv_tl }

```

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:nnn 697 \cs_new_protected:Npn \zhnum_set_digits_map:nn #1
\zhnum_set_financial_map:nn 698 { \prop_put:Nnn \l__zhnum_cfg_map_prop {#1} }
\zhnum_set_financial_map:nnn 699 \cs_new_protected:Npn \zhnum_set_digits_map:nnn #1#2#3
\zhnum_set_tiangang_map:nn 700 {
\zhnum_set_dizhi_map:nn 701   \prop_put_if_new:Nnn \l__zhnum_cfg_map_prop {#1} {#3}
\l__zhnum_cfg_map_prop 702   \prop_put:Nnn \l__zhnum_cfg_map_var_prop {#1_#2} {#3}
\l__zhnum_cfg_map_var_prop 703 }
\l__zhnum_cfg_map_finan_prop 704 \cs_new_protected:Npn \zhnum_set_financial_map:nn #1
\l__zhnum_cfg_map_ganzhi_prop 705 { \prop_put:Nnn \l__zhnum_cfg_map_finan_prop {#1} }
706 \cs_new_protected:Npn \zhnum_set_financial_map:nnn #1#2#3
707 {
708   \prop_put_if_new:Nnn \l__zhnum_cfg_map_finan_prop {#1} {#3}
709   \prop_put:Nnn \l__zhnum_cfg_map_var_prop { financial_#1_#2 } {#3}
710 }
711 \cs_new_protected:Npn \zhnum_set_tiangang_map:nn #1
712 { \prop_put:Nnn \l__zhnum_cfg_map_ganzhi_prop { tiangan_#1 } }
713 \cs_new_protected:Npn \zhnum_set_dizhi_map:nn #1
714 { \prop_put:Nnn \l__zhnum_cfg_map_ganzhi_prop { dizhi_#1 } }
715 \prop_new:N \l__zhnum_cfg_map_prop
716 \prop_new:N \l__zhnum_cfg_map_var_prop
717 \prop_new:N \l__zhnum_cfg_map_finan_prop
718 \prop_new:N \l__zhnum_cfg_map_ganzhi_prop

```

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 719 \cs_new_protected:Npn \zhnum_parse_config:
\zhnum_check_financial:nn
\zhnum_set_week_day:

```

```

720 {
721   \tl_clear_new:N \l__zhnum_reset_tl
722   \tl_clear_new:N \l__zhnum_reset_simp_tl
723   \tl_clear_new:N \l__zhnum_reset_trad_tl
724   \tl_clear_new:N \l__zhnum_set_ancient_tl
725   \tl_clear_new:N \l__zhnum_set_normal_tl
726   \tl_clear_new:N \l__zhnum_reset_ancient_tl
727   \tl_clear_new:N \l__zhnum_reset_normal_tl
728   \tl_clear_new:N \l__zhnum_reset_financial_tl
729   \prop_map_function:NN \l__zhnum_cfg_map_prop \zhnum_check_simp:nn
730   \zhnum_set_ganzhi:
731   \zhnum_reset_all:
732 }
733 \cs_new_protected:Npn \zhnum_check_simp:nn #1#2
734 {
735   \prop_get:NnNTF \l__zhnum_cfg_map_var_prop
736   { #1_ancient } \l__zhnum_ancient_tl
737   { \__zhnum_add_reset_ancient:nN {#1} \l__zhnum_ancient_tl }
738   {
739     \__zhnum_check_simp_aux:nn {#2} {#1}
740     \prop_get:NnNT \l__zhnum_cfg_map_finan_prop {#1} \l__zhnum_tmp_tl
741     {
742       \exp_args:No \__zhnum_check_simp_aux:nn
743       { \l__zhnum_tmp_tl } { financial_ #1 }
744       \__zhnum_add_reset_financial:n {#1}
745     }
746   }
747 }
748 \cs_new_protected:Npn \__zhnum_check_simp_aux:nn #1#2
749 {
750   \prop_get:NnNTF \l__zhnum_cfg_map_var_prop
751   { #2_trad } \l__zhnum_trad_tl
752   {
753     \prop_get:NnNF \l__zhnum_cfg_map_var_prop
754     { #2_simp } \l__zhnum_simp_tl
755     { \tl_set:Nn \l__zhnum_simp_tl {#1} }
756     \__zhnum_add_reset_simp:nNN
757     {#2} \l__zhnum_simp_tl \l__zhnum_trad_tl
758   }
759   { \__zhnum_add_reset:nn {#2} {#1} }
760 }
761 \tl_new:N \l__zhnum_simp_tl
762 \tl_new:N \l__zhnum_trad_tl
763 \tl_new:N \l__zhnum_ancient_tl
764 \cs_new_protected:Npn \__zhnum_add_reset:nn #1#2
765 {
766   \tl_put_right:Nx \l__zhnum_reset_tl
767   {
768     \tl_set:Nn \exp_not:c { l__zhnum_ #1 _tl }
769     { \exp_not:n {#2} }
770   }
771 }
772 \cs_new_protected:Npn \__zhnum_add_reset_simp:nNN #1#2#3
773 {
774   \tl_put_right:Nx \l__zhnum_reset_simp_tl
775   {
776     \tl_set:Nn \exp_not:c { l__zhnum_ #1 _tl }
777     { \exp_not:o {#2} }
778   }
779   \tl_put_right:Nx \l__zhnum_reset_trad_tl
780   {
781     \tl_set:Nn \exp_not:c { l__zhnum_ #1 _tl }
782     { \exp_not:o {#3} }
783   }
784 }
785 \cs_new_protected:Npn \__zhnum_add_reset_financial:n #1
786 {

```

```

787 \tl_put_right:Nx \l__zhnum_set_normal_tl
788 {
789   \tl_set_eq:NN
790   \exp_not:c { l__zhnum_normal_ #1 _tl }
791   \exp_not:c { l__zhnum_ #1 _tl }
792 }
793 \tl_put_right:Nx \l__zhnum_reset_normal_tl
794 {
795   \tl_set_eq:NN
796   \exp_not:c { l__zhnum_ #1 _tl }
797   \exp_not:c { l__zhnum_normal_ #1 _tl }
798 }
799 \tl_put_right:Nx \l__zhnum_reset_financial_tl
800 {
801   \tl_set_eq:NN
802   \exp_not:c { l__zhnum_ #1 _tl }
803   \exp_not:c { l__zhnum_financial_ #1 _tl }
804 }
805 }
806 \cs_new_protected:Npn \__zhnum_add_reset_ancient:nN #1#2
807 {
808   \tl_put_right:Nx \l__zhnum_reset_ancient_tl
809   {
810     \tl_set:Nn \exp_not:c { l__zhnum_ #1 _tl }
811     { \exp_not:o {#2} }
812   }
813   \tl_put_right:Nx \l__zhnum_set_ancient_tl
814   {
815     \tl_concat:NNN
816     \exp_not:c { l__zhnum_ #1 _tl }
817     \exp_not:c { l__zhnum_ \str_head:n {#1} _tl }
818     \exp_not:c { l__zhnum_ 1 \str_tail:n {#1} _tl }
819   }
820 }

\zhnum_set_week_day: 821 \cs_new_protected:Npn \zhnum_set_week_day:
\zhnum_reset_week_day: 822 {
823   \cs_set_protected:Npx \zhnum_reset_week_day:
824   {
825     \__zhnum_set_week_day:nn { mon } { 1 }
826     \__zhnum_set_week_day:nn { tue } { 2 }
827     \__zhnum_set_week_day:nn { wed } { 3 }
828     \__zhnum_set_week_day:nn { thu } { 4 }
829     \__zhnum_set_week_day:nn { fri } { 5 }
830     \__zhnum_set_week_day:nn { sat } { 6 }
831     \__zhnum_set_week_day:nn { sun } { day }
832   }
833 }
834 \cs_new_eq:NN \zhnum_reset_week_day: \prg_do_nothing:
835 \cs_new:Npn \__zhnum_set_week_day:nn #1#2
836 {
837   \tl_set:Nx \exp_not:c { l__zhnum_ #1 _tl }
838   {
839     \exp_not:N \exp_not:o { \exp_not:N \l__zhnum_weekday_tl }
840     \exp_not:N \exp_not:n { \exp_not:v { l__zhnum_ #2 _tl } }
841   }
842 }

\zhnum_set_ganzhi: 843 \cs_new_protected:Npn \zhnum_set_ganzhi:
\zhnum_reset_ganzhi: 844 {
845   \prop_map_function:NN
846   \l__zhnum_cfg_map_ganzhi_prop
847   \__zhnum_add_reset:nn
848 }
849 \cs_new_protected:Npn \__zhnum_reset_ganzhi:nn #1#2
850 { \tl_set:cn { l__zhnum_ #1 _tl } {#2} }
851 \cs_new:Npn \zhnum_zero_mod:nn #1#2
852 { \exp_args:Nf \__zhnum_zero_mod_aux:nn { \int_mod:nn {#1} {#2} } {#2} }

```

```

853 \cs_new:Npn \__zhnum_zero_mod_aux:nn #1#2
854 { \int_compare:nNnTF {#1} = \c_zero_int {#2} {#1} }
855 \tl_new:c { l__zhnum_dizhi_0_tl }
856 \tl_new:c { l__zhnum_ganzhi_0_tl }
857 \tl_new:c { l__zhnum_tiangang_0_tl }
858 \group_begin:
859 \cs_set:Npn \__zhnum_tmp:w #1
860 {
861   \tl_concat:NNN
862     \exp_not:c { l__zhnum_ganzhi_ #1_tl }
863     \exp_not:c { l__zhnum_tiangang_ \zhnum_zero_mod:nn {#1} { 10 }_tl }
864     \exp_not:c { l__zhnum_dizhi_ \zhnum_zero_mod:nn {#1} { 12 }_tl }
865 }
866 \cs_new_protected:Npx \zhnum_reset_ganzhi:
867 {
868   \tl_set_eq:NN
869     \exp_not:c { l__zhnum_dizhi_0_tl }
870     \exp_not:c { l__zhnum_dizhi_12_tl }
871   \tl_set_eq:NN
872     \exp_not:c { l__zhnum_tiangang_0_tl }
873     \exp_not:c { l__zhnum_tiangang_10_tl }
874   \int_step_function:nN { 60 } \__zhnum_tmp:w
875   \tl_set_eq:NN
876     \exp_not:c { l__zhnum_ganzhi_0_tl }
877     \exp_not:c { l__zhnum_ganzhi_60_tl }
878 }
879 \group_end:
\zhnum_reset_config: 880 \cs_new_protected:Npn \zhnum_reset_config:
  \zhnum_reset_all: 881 { \zhnum_load_cfg:o { \l__zhnum_encoding_str } }
  \zhnum_reset_style: 882 \cs_new_protected:Npn \zhnum_reset_all:
883 {
884   \zhnum_reset_main:
885   \zhnum_reset_simp:
886   \zhnum_set_week_day:
887   \zhnum_reset_week_day:
888   \zhnum_reset_ganzhi:
889   \zhnum_reset_normal:
890 }
891 \cs_new_protected:Npn \zhnum_reset_main:
892 {
893   \tl_use:N \l__zhnum_reset_tl
894   \tl_use:N \l__zhnum_set_normal_tl
895   \tl_concat:NNN
896     \l__zhnum_reset_normal_tl
897     \l__zhnum_reset_normal_tl
898     \c__zhnum_set_zero_tl
899   \tl_concat:NNN
900     \l__zhnum_reset_financial_tl
901     \l__zhnum_reset_financial_tl
902     \c__zhnum_set_zero_tl
903   \tl_concat:NNN
904     \l__zhnum_reset_financial_tl
905     \l__zhnum_reset_financial_tl
906     \l__zhnum_set_ancient_tl
907 }
908 \tl_const:Nx \c__zhnum_set_zero_tl
909 {
910   \tl_set_eq:NN
911     \exp_not:N \l__zhnum_zero_tl
912     \exp_not:c { l__zhnum_0_tl }
913 }
914 \tl_new:N \l__zhnum_zero_tl
915 \cs_new_protected:Npn \zhnum_reset_style:
916 {
917   \zhnum_reset_simp:
918   \zhnum_reset_normal:
919 }

```

```

920 \cs_new_protected:Npn \zhnum_reset_simp:
921 {
922   \bool_if:NTF \l__zhnum_simp_bool
923     { \tl_use:N \l__zhnum_reset_simp_tl }
924     { \tl_use:N \l__zhnum_reset_trad_tl }
925 }
926 \cs_new_protected:Npn \zhnum_reset_normal:
927 {
928   \bool_if:NTF \l__zhnum_normal_bool
929     {
930       \tl_use:N \l__zhnum_reset_normal_tl
931       \__zhnum_reset_ancient:
932       \__zhnum_reset_zero:
933     }
934     { \tl_use:N \l__zhnum_reset_financial_tl }
935 }
936 \cs_new_protected:Npn \__zhnum_reset_ancient:
937 {
938   \bool_if:NTF \l__zhnum_ancient_bool
939     { \tl_use:N \l__zhnum_reset_ancient_tl }
940     { \tl_use:N \l__zhnum_set_ancient_tl }
941 }
942 \cs_new_protected:Npx \__zhnum_reset_zero:
943 {
944   \exp_not:n { \bool_if:NT \l__zhnum_null_bool }
945   {
946     \tl_set_eq:NN
947     \exp_not:c { l__zhnum_0_tl }
948     \exp_not:N \l__zhnum_null_tl
949   }
950 }

```

\zhnum_load_cfg:n 根据选定编码载入配置文件。

```

951 \cs_new_protected:Npn \zhnum_load_cfg:n #1
952 {
953   \zhnum_set_cfg_name:Nn \l__zhnum_cfg_str {#1}
954   \str_if_eq:NNTF \l__zhnum_cfg_str \l__zhnum_last_cfg_str
955     { \zhnum_reset_all: }
956     {
957       \zhnum_update_cfg:n {#1}
958       \zhnum_parse_config:
959     }
960 }
961 \cs_generate_variant:Nn \zhnum_load_cfg:n { o }
962 \cs_new_protected:Npn \zhnum_update_cfg:n #1
963 {
964   \prop_if_exist:cTF { g__zhnum_cfg_ \l__zhnum_cfg_str _prop }
965     { \__zhnum_set_cfg_prop: }
966     { \zhnum_input_cfg:n {#1} }
967 }
968 \cs_new_protected:Npn \__zhnum_set_cfg_prop:
969 {
970   \str_set_eq:NN \l__zhnum_last_cfg_str \l__zhnum_cfg_str
971   \__zhnum_update_cfg_prop:N \prop_set_eq:Nc
972 }
973 \cs_new_protected:Npn \zhnum_input_cfg:n #1
974 {
975   \file_get_full_name:nNTF { zhnumber - #1 .cfg } \l__zhnum_cfg_file_tl
976   {
977     \bool_set_false:N \l__zhnum_reset_bool
978     \__zhnum_update_cfg_prop:N \__zhnum_prop_initial:Nn
979     \group_begin:
980       \zhnum_set_catcode:
981       \exp_args:No \file_input:n { \l__zhnum_cfg_file_tl }
982       \__zhnum_update_cfg_prop:N \__zhnum_prop_gset_eq:Nn
983     \group_end:
984     \__zhnum_set_cfg_prop:

```

```

985     }
986     { \msg_error:nx { zhnumber } { file-not-found } {#1} }
987 }
988 \tl_new:N \l__zhnum_cfg_file_tl
989 \cs_new_protected:Npn \__zhnum_update_cfg_prop:N
990   { \exp_args:No \__zhnum_update_cfg_prop_aux:nN { \l__zhnum_cfg_str } }
991 \cs_new_protected:Npn \__zhnum_update_cfg_prop_aux:nN #1#2
992   {
993     #2 \l__zhnum_cfg_map_prop      { g__zhnum_cfg_      #1 _prop }
994     #2 \l__zhnum_cfg_map_var_prop  { g__zhnum_cfg_var_  #1 _prop }
995     #2 \l__zhnum_cfg_map_finan_prop { g__zhnum_cfg_finan_ #1 _prop }
996     #2 \l__zhnum_cfg_map_ganzhi_prop { g__zhnum_cfg_ganzhi_ #1 _prop }
997   }
998 \cs_new_protected:Npn \__zhnum_prop_initial:Nn #1#2
999   {
1000     \prop_clear:N #1
1001     \prop_new:c {#2}
1002   }
1003 \cs_new_protected:Npn \__zhnum_prop_gset_eq:Nn #1#2
1004   { \prop_gset_eq:cN {#2} #1 }
1005 \str_new:N \l__zhnum_cfg_str
1006 \str_new:N \l__zhnum_last_cfg_str
1007 \bool_new:N \l__zhnum_reset_bool
1008 \msg_new:nnnn { zhnumber } { file-not-found }
1009   { File~`#1'~not~found. }
1010   {
1011     The~requested~file~could~not~be~found~in~the~current~directory,~
1012     in~the~TeX~search~path~or~in~the~LaTeX~search~path.
1013   }

```

`\c__zhnum_unicode_engine_bool` 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。

```

1014 \bool_const:Nn \c__zhnum_unicode_engine_bool
1015   {
1016     \bool_lazy_any_p:n
1017       {
1018         { \sys_if_engine_xetex_p: }
1019         { \sys_if_engine_luatex_p: }
1020         { \sys_if_engine_uptex_p: }
1021       }
1022   }

```

`\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`
`\zhnum_reset_config:`

```

1023 \bool_if:NTF \c__zhnum_unicode_engine_bool
1024   {
1025     \cs_new_eq:NN \zhnum_set_catcode: \prg_do_nothing:
1026     \cs_new_protected:Npn \zhnum_set_cfg_name:Nn #1#2
1027       {
1028         \str_set:Nx \l__zhnum_encoding_str {#2}
1029         \str_set_eq:NN #1 \l__zhnum_encoding_str
1030       }
1031   }
1032   {
1033     \cs_new_protected:Npn \zhnum_set_catcode:
1034       {
1035         \bool_if:NTF \l__zhnum_active_char_bool
1036           { \zhnum_set_active: }
1037           { \zhnum_set_other: }
1038         }
1039     \cs_new_protected:Npx \zhnum_set_active:
1040       {
1041         \int_step_function:nnN
1042           { 128 } { 255 } \char_set_catcode_active:n
1043       }
1044     \cs_new_protected:Npx \zhnum_set_other:
1045       {

```

```

1046     \int_step_function:nnN
1047     { 128 } { 255 } \char_set_catcode_other:n
1048   }
1049   \cs_new_protected:Npn \zhnum_set_cfg_name:Nn #1#2
1050   {
1051     \str_set:Nx \l__zhnum_encoding_str {#2}
1052     \str_set:Nx #1
1053     {
1054       \l__zhnum_encoding_str
1055       \bool_if:NTF \l__zhnum_active_char_bool
1056         { / active }
1057         { / other }
1058     }
1059   }
1060   \bool_new:N \l__zhnum_active_char_bool
1061   \bool_set_true:N \l__zhnum_active_char_bool
1062 }

```

\zhnum_set_encoding:n 设置编码。

```

1063 \cs_new_protected:Npn \zhnum_set_encoding:n #1
1064 {
1065   \str_set:Nx \l__zhnum_encoding_str
1066     { \str_lowercase:n {#1} }
1067   \zhnum_load_cfg:o { \l__zhnum_encoding_str }
1068 }
1069 \str_new:N \l__zhnum_encoding_str

```

encoding 宏包设置选项。

```

style 1070 \keys_define:nn { zhnum / options }
null 1071 {
reset 1072   encoding .choices:nn =
1073     { UTF8 , GBK , Big5 }
1074     { \exp_args:No \zhnum_set_encoding:n { \l_keys_choice_tl } } ,
1075   encoding .default:n = { GBK } ,
1076   encoding / Bg5 .meta:n = { encoding = Big5 } ,
1077   encoding / unknown .code:n =
1078     { \msg_error:nnn { zhnumber } { encoding-invalid } {#1} } ,
1079   style .multichoice: ,
1080   style / Normal .code:n =
1081     {
1082       \bool_set_false:N \l__zhnum_ancient_bool
1083       \bool_set_true:N \l__zhnum_normal_bool
1084     } ,
1085   style / Financial .code:n =
1086     {
1087       \bool_set_false:N \l__zhnum_ancient_bool
1088       \bool_set_false:N \l__zhnum_normal_bool
1089     } ,
1090   style / Ancient .code:n =
1091     {
1092       \bool_set_true:N \l__zhnum_ancient_bool
1093       \bool_set_true:N \l__zhnum_normal_bool
1094     } ,
1095   style / Simplified .code:n = { \bool_set_true:N \l__zhnum_simp_bool } ,
1096   style / Traditional .code:n = { \bool_set_false:N \l__zhnum_simp_bool } ,
1097   style .default:n = { Normal , Simplified } ,
1098   style .groups:n = { style } ,
1099   null .bool_set:N = \l__zhnum_null_bool ,
1100   null .groups:n = { style } ,
1101   time .choice: ,
1102   time / Chinese .code:n = { \bool_set_true:N \l__zhnum_time_bool } ,
1103   time / Arabic .code:n = { \bool_set_false:N \l__zhnum_time_bool } ,
1104   time .default:n = { Arabic } ,
1105   reset .bool_set:N = \l__zhnum_reset_bool ,
1106   activechar .bool_set:N = \l__zhnum_active_char_bool ,
1107   ganzhi-cyclic .choice: ,

```

```

1108 ganzhi-cyclic / true .code:n =
1109   { \cs_set_eq:NN \zhnum_ganzhi:nnn \zhnum_ganzhi_cyclic:nnn } ,
1110 ganzhi-cyclic / false.code:n =
1111   { \cs_set_eq:NN \zhnum_ganzhi:nnn \zhnum_ganzhi_normal:nnn } ,
1112 ganzhi-cyclic .default:n = { true } ,
1113 arabicsep .tl_set:N = \l__zhnum_arabic_sep_tl
1114 }
1115 \bool_new:N \l__zhnum_simp_bool
1116 \bool_new:N \l__zhnum_normal_bool
1117 \bool_new:N \l__zhnum_ancient_bool
1118 \msg_new:nnnn { zhnumber } { encoding-invalid }
1119 { The~encoding~`#1'~is~invalid. }
1120 { Available~encodings~are~`UTF8',~`GBK'~and~`Big5'. }

```

`\zhnumsetup` 在文档中设置 `zhnumber` 的接口。

```

1121 \NewDocumentCommand \zhnumsetup { +m }
1122 {
1123   \zhnum_set:n {#1}
1124   \tex_ignorespaces:D
1125 }
1126 \cs_new_protected:Npn \zhnum_set:n #1
1127 {
1128   \bool_set_false:N \l__zhnum_reset_bool
1129   \keys_set_filter:nnnN { zhnum / options }
1130     { style , user } {#1} \l__zhnum_kv_tl
1131   \tl_if_empty:NF \l__zhnum_kv_tl
1132     { \__zhnum_set_style: }
1133   \bool_if:NT \l__zhnum_reset_bool
1134     { \zhnum_reset_config: }
1135 }
1136 \cs_new_protected:Npn \__zhnum_set_style:
1137 {
1138   \keys_set_filter:nnoN { zhnum / options }
1139     { user } { \l__zhnum_kv_tl } \l__zhnum_tmp_tl
1140   \tl_if_empty:NTF \l__zhnum_tmp_tl
1141     { \zhnum_reset_style: }
1142     {
1143       \tl_if_eq:NNF \l__zhnum_tmp_tl \l__zhnum_kv_tl
1144         { \zhnum_reset_style: }
1145       \bool_if:NF \l__zhnum_reset_bool
1146         { \__zhnum_set_user: }
1147     }
1148 }
1149 \cs_new_protected:Npn \__zhnum_set_user:
1150 {
1151   \keys_set_filter:nnoN { zhnum / options }
1152     { pre , pos } { \l__zhnum_tmp_tl } \l__zhnum_kv_tl
1153   \tl_if_eq:NNF \l__zhnum_kv_tl \l__zhnum_tmp_tl
1154     { \zhnum_reset_normal: }
1155   \tl_if_empty:NF \l__zhnum_kv_tl
1156     { \__zhnum_set_pre_pos: }
1157 }
1158 \cs_new_protected:Npn \__zhnum_set_pre_pos:
1159 {
1160   \keys_set_filter:nnoN { zhnum / options }
1161     { pos } { \l__zhnum_kv_tl } \l__zhnum_tmp_tl
1162   \tl_if_eq:NNF \l__zhnum_tmp_tl \l__zhnum_kv_tl
1163     {
1164       \zhnum_reset_week_day:
1165       \zhnum_reset_ganzhi:
1166     }
1167   \tl_if_empty:NF \l__zhnum_kv_tl
1168     { \keys_set:no { zhnum / options } { \l__zhnum_kv_tl } }
1169 }
1170 \cs_generate_variant:Nn \keys_set_filter:nnnN { nno }

```


初始化设置和执行宏包选项。

```

1171 \keys_set:nn { zhnum / options }
1172 { style , time , arabicsep = { ~ } }
1173 \cs_if_exist:NTF \ProcessKeyOptions
1174 { \ProcessKeyOptions [ zhnum / options ] }
1175 {
1176   \RequirePackage { l3keys2e }
1177   \ProcessKeysOptions { zhnum / options }
1178 }

```

如果没有选定编码,则选用 UTF8。

```

1179 \str_if_empty:NTF \l__zhnum_encoding_str
1180 { \zhnum_set_encoding:n { UTF8 } }
1181 { \zhnum_reset_style: }
1182 </package>

```

第 4 节 中文数字配置文件

```

1183 <*config>
1184 <!*big5>
1185 \zhnum_set_digits_map:nnn { minus } { simp } { 负 }
1186 \zhnum_set_digits_map:nnn { minus } { trad } { 負 }
1187 </!big5>
1188 <*big5>
1189 \zhnum_set_digits_map:nn { minus } { 負 }
1190 </big5>
1191 \zhnum_set_digits_map:nn { 0 } { 零 }
1192 <!*big5>
1193 \zhnum_set_digits_map:nn { null } { 〇 }
1194 </!big5>
1195 <*big5>
1196 \zhnum_set_digits_map:nn { null } { 〇 }
1197 </big5>
1198 \zhnum_set_digits_map:nn { 1 } { 一 }
1199 \zhnum_set_digits_map:nn { 2 } { 二 }
1200 \zhnum_set_digits_map:nn { 3 } { 三 }
1201 \zhnum_set_digits_map:nn { 4 } { 四 }
1202 \zhnum_set_digits_map:nn { 5 } { 五 }
1203 \zhnum_set_digits_map:nn { 6 } { 六 }
1204 \zhnum_set_digits_map:nn { 7 } { 七 }
1205 \zhnum_set_digits_map:nn { 8 } { 八 }
1206 \zhnum_set_digits_map:nn { 9 } { 九 }
1207 \zhnum_set_digits_map:nn { 10 } { 十 }
1208 \zhnum_set_digits_map:nn { 100 } { 百 }
1209 \zhnum_set_digits_map:nn { 1000 } { 千 }
1210 \zhnum_set_digits_map:nnn { 20 } { ancient } { 廿 }
1211 \zhnum_set_digits_map:nnn { 30 } { ancient } { 卅 }
1212 \zhnum_set_digits_map:nnn { 40 } { ancient } { 卌 }
1213 \zhnum_set_digits_map:nnn { 200 } { ancient } { 貳 }
1214 <!*big5>
1215 \zhnum_set_digits_map:nnn { dot } { simp } { 点 }
1216 \zhnum_set_digits_map:nnn { dot } { trad } { 點 }
1217 </!big5>
1218 <*big5>
1219 \zhnum_set_digits_map:nn { dot } { 點 }
1220 </big5>
1221 \zhnum_set_digits_map:nn { and } { 又 }
1222 \zhnum_set_digits_map:nn { parts } { 分之 }
1223 <!*big5>
1224 \zhnum_set_digits_map:nnn { s1 } { simp } { 万 }
1225 \zhnum_set_digits_map:nnn { s1 } { trad } { 萬 }
1226 \zhnum_set_digits_map:nnn { s2 } { simp } { 亿 }
1227 \zhnum_set_digits_map:nnn { s2 } { trad } { 億 }
1228 </!big5>

```

```

1229 <*big5>
1230 \zhnum_set_digits_map:nn { s1 } { 萬 }
1231 \zhnum_set_digits_map:nn { s2 } { 億 }
1232 </big5>
1233 \zhnum_set_digits_map:nn { s3 } { 兆 }
1234 \zhnum_set_digits_map:nn { s4 } { 京 }
1235 \zhnum_set_digits_map:nn { s5 } { 垓 }
1236 \zhnum_set_digits_map:nn { s6 } { 秭 }
1237 \zhnum_set_digits_map:nn { s7 } { 穰 }
1238 <*!big5>
1239 \zhnum_set_digits_map:nnn { s8 } { simp } { 沟 }
1240 \zhnum_set_digits_map:nnn { s8 } { trad } { 溝 }
1241 \zhnum_set_digits_map:nnn { s9 } { simp } { 涧 }
1242 \zhnum_set_digits_map:nnn { s9 } { trad } { 澗 }
1243 </!big5>
1244 <*big5>
1245 \zhnum_set_digits_map:nn { s8 } { 溝 }
1246 \zhnum_set_digits_map:nn { s9 } { 澗 }
1247 </big5>
1248 \zhnum_set_digits_map:nn { s10 } { 正 }
1249 <*!big5>
1250 \zhnum_set_digits_map:nnn { s11 } { simp } { 载 }
1251 \zhnum_set_digits_map:nnn { s11 } { trad } { 載 }
1252 </!big5>
1253 <*big5>
1254 \zhnum_set_digits_map:nn { s11 } { 載 }
1255 </big5>
1256 \zhnum_set_digits_map:nn { year } { 年 }
1257 \zhnum_set_digits_map:nn { month } { 月 }
1258 \zhnum_set_digits_map:nn { day } { 日 }
1259 <*!big5>
1260 \zhnum_set_digits_map:nnn { hour } { simp } { 时 }
1261 \zhnum_set_digits_map:nnn { hour } { trad } { 時 }
1262 </!big5>
1263 <*big5>
1264 \zhnum_set_digits_map:nn { hour } { 時 }
1265 </big5>
1266 \zhnum_set_digits_map:nn { minute } { 分 }
1267 \zhnum_set_digits_map:nn { weekday } { 星期 }
1268 \zhnum_set_financial_map:nn { null } { 零 }
1269 \zhnum_set_financial_map:nn { 0 } { 零 }
1270 \zhnum_set_financial_map:nn { 1 } { 壹 }
1271 <*!big5>
1272 \zhnum_set_financial_map:nnn { 2 } { simp } { 贰 }
1273 \zhnum_set_financial_map:nnn { 2 } { trad } { 貳 }
1274 \zhnum_set_financial_map:nnn { 3 } { simp } { 叁 }
1275 \zhnum_set_financial_map:nnn { 3 } { trad } { 參 }
1276 </!big5>
1277 <*big5>
1278 \zhnum_set_financial_map:nn { 2 } { 貳 }
1279 \zhnum_set_financial_map:nn { 3 } { 參 }
1280 </big5>
1281 \zhnum_set_financial_map:nn { 4 } { 肆 }
1282 \zhnum_set_financial_map:nn { 5 } { 伍 }
1283 <*!big5>
1284 \zhnum_set_financial_map:nnn { 6 } { simp } { 陆 }
1285 \zhnum_set_financial_map:nnn { 6 } { trad } { 陸 }
1286 </!big5>
1287 <*big5>
1288 \zhnum_set_financial_map:nn { 6 } { 陸 }
1289 </big5>
1290 \zhnum_set_financial_map:nn { 7 } { 柒 }
1291 \zhnum_set_financial_map:nn { 8 } { 捌 }
1292 \zhnum_set_financial_map:nn { 9 } { 玖 }
1293 \zhnum_set_financial_map:nn { 10 } { 拾 }
1294 \zhnum_set_financial_map:nn { 100 } { 佰 }
1295 \zhnum_set_financial_map:nn { 1000 } { 仟 }

```

```
1296 \zhnum_set_tiangang_map:nn { 1 } { 甲 }
1297 \zhnum_set_tiangang_map:nn { 2 } { 乙 }
1298 \zhnum_set_tiangang_map:nn { 3 } { 丙 }
1299 \zhnum_set_tiangang_map:nn { 4 } { 丁 }
1300 \zhnum_set_tiangang_map:nn { 5 } { 戊 }
1301 \zhnum_set_tiangang_map:nn { 6 } { 己 }
1302 \zhnum_set_tiangang_map:nn { 7 } { 庚 }
1303 \zhnum_set_tiangang_map:nn { 8 } { 辛 }
1304 \zhnum_set_tiangang_map:nn { 9 } { 壬 }
1305 \zhnum_set_tiangang_map:nn { 10 } { 癸 }
1306 \zhnum_set_dizhi_map:nn { 1 } { 子 }
1307 \zhnum_set_dizhi_map:nn { 2 } { 丑 }
1308 \zhnum_set_dizhi_map:nn { 3 } { 寅 }
1309 \zhnum_set_dizhi_map:nn { 4 } { 卯 }
1310 \zhnum_set_dizhi_map:nn { 5 } { 辰 }
1311 \zhnum_set_dizhi_map:nn { 6 } { 巳 }
1312 \zhnum_set_dizhi_map:nn { 7 } { 午 }
1313 \zhnum_set_dizhi_map:nn { 8 } { 未 }
1314 \zhnum_set_dizhi_map:nn { 9 } { 申 }
1315 \zhnum_set_dizhi_map:nn { 10 } { 酉 }
1316 \zhnum_set_dizhi_map:nn { 11 } { 戌 }
1317 \zhnum_set_dizhi_map:nn { 12 } { 亥 }
1318 </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	E
\\ 5, 6, 7	else commands:
A	\else: 152, 164, 195, 354, 384
activechar 4	encoding 1, <u>1070</u>
arabicsep 3	exp commands:
B	\exp:w 143, 151, 154, 163, 175, 346, 353, 371
bingint internal commands:	\exp_after:wN 140, 142, 150, 151,
_bingint_read_do:nn 7	153, 155, 156, 162, 163, 165, 166, 174, 194, 196, 216,
bool commands:	219, 220, 221, 343, 345, 352, 353, 355, 356, 357, 358, 370
\bool_const:Nn 1014	\exp_args:Nc 15, 116, 528
\bool_if:NTF 239, 253, 383, 393,	\exp_args:Nf 200, 202, 852
406, 496, 922, 928, 938, 944, 1023, 1035, 1055, 1133, 1145	\exp_args:NNno 693
\bool_lazy_all:nTF 279	\exp_args:No 742, 981, 990, 1074
\bool_lazy_any_p:n 1016	\exp_end_continue_f:w 143, 151, 154, 163, 175, 346, 353, 371
\bool_new:N 1007, 1060, 1115, 1116, 1117	\exp_not:N 149, 161, 193, 351, 483, 484, 611,
\bool_set_false:N 977, 1082, 1087, 1088, 1096, 1103, 1128	612, 614, 622, 623, 633, 637, 638, 639, 640, 641, 642,
\bool_set_true:N ... 1061, 1083, 1092, 1093, 1095, 1102	643, 664, 672, 680, 688, 768, 776, 781, 790, 791, 796,
\c_false_bool 244, 245, 338	797, 802, 803, 810, 816, 817, 818, 837, 839, 840, 862,
\c_true_bool 229, 244, 245, 336	863, 864, 869, 870, 872, 873, 876, 877, 911, 912, 947, 948
C	\exp_not:n 19, 20, 21, 29, 30, 769, 777, 782, 811, 839, 840, 944
char commands:	\exp_stop_f: 144, 193, 218, 223, 347, 424
\char_set_catcode_active:n 1042	F
\char_set_catcode_other:n 1047	fi commands:
clist commands:	\fi: 149, 157, 167, 197, 222, 351, 359, 386, 432
\clist_map_function:nN 521	file commands:
\clist_map_inline:Nn 648	\file_get_full_name:nNTF 975
\clist_map_inline:nn 617, 626, 659	\file_input:n 981
cs commands:	G
\cs_generate_variant:Nn 59,	ganzhi-cyclic 4
213, 232, 248, 339, 389, 513, 574, 576, 601, 961, 1170	group commands:
\cs_if_exist:NTF 12, 1173	\group_begin: 45, 108, 308, 330, 604, 858, 979
\cs_if_free:NTF 547	\group_end: 48, 111, 311, 333, 693, 879, 983
\cs_new:Npn ... 14, 51, 53, 60, 66, 85, 93, 113, 119, 124,	H
137, 138, 147, 159, 169, 177, 179, 181, 191, 199, 201,	hook commands:
203, 214, 225, 227, 233, 249, 313, 319, 335, 337, 341,	\hook_gput_code:nnn 32, 34
349, 361, 373, 378, 395, 397, 404, 410, 413, 419, 422,	I
434, 442, 448, 453, 466, 479, 483, 488, 494, 500, 505,	if commands:
511, 543, 552, 564, 577, 579, 581, 583, 586, 835, 851, 853	\if:w 149, 161, 351
\cs_new_eq:NN 20, 21, 516, 575, 834, 1025	\if_case:w 218, 424
\cs_new_protected:Npn 16,	\if_int_compare:w 193, 382
18, 22, 27, 525, 532, 697, 699, 704, 706, 711, 713, 719,	int commands:
733, 748, 764, 772, 785, 806, 821, 843, 849, 880, 882,	\int_compare:nNnTF 126, 129, 183,
891, 915, 920, 926, 936, 951, 962, 968, 973, 989, 991,	186, 205, 208, 236, 242, 251, 260, 262, 266, 273, 275,
998, 1003, 1026, 1033, 1049, 1063, 1126, 1136, 1149, 1158	296, 375, 436, 444, 450, 545, 554, 566, 569, 588, 591, 854
\cs_new_protected:Npx 866, 942, 1039, 1044	\int_compare:nTF 288
\cs_set:Npn 523, 646, 859	\int_compare_p:nNn 281, 282
\cs_set_eq:NN 24, 25, 29, 30, 1109, 1111	\int_div_truncate:nn 217, 458, 460, 461, 462, 471, 473, 491
\cs_set_protected:Npx 823	

`\int_eval:n` 127, 132, 180, 209, 246, 320, 451, 529, 571, 578, 580, 582, 584, 595, 613

`\int_if_exist:NTF` 115, 315

`\int_incr:N` 527, 537

`\int_mod:nn` ... 202, 455, 468, 492, 512, 560, 589, 596, 852

`\int_new:N` 514, 541

`\int_set:Nn` 515

`\int_step_function:nN` 874

`\int_step_function:nnN` 1041, 1046

`\int_step_inline:nn` 606, 668, 676, 684

`\int_to_arabic:n` 408, 498

`\int_value:w` 141, 217, 344, 400

`\int_zero:N` 520

`\c_one_int` 186, 281, 375, 545

`\c_zero_int` 126, 129, 145, 183, 208, 236, 242, 251, 260, 262, 273, 275, 282, 296, 382, 554, 566, 569, 588, 591, 854

K

keys commands:

`\l_keys_choice_tl` 1074

`\keys_define:nn` 694, 1070

`\keys_set:nn` 1168, 1171

`\keys_set_filter:nnnN` ... 1129, 1138, 1151, 1160, 1170

M

msg commands:

`\msg_error:nn` 11

`\msg_error:nnn` 986, 1078

`\msg_expandable_error:nnn` 120

`\msg_new:nnn` 3, 121

`\msg_new:nnnn` 1008, 1118

N

`\NewDocumentCommand` 12, 43, 106, 306, 328, 518, 1121

`\NewExpandableDocumentCommand` 36, 99, 299, 321, 390

`null` 3, 1070

O

or commands:

`\or:` 219, 220, 221, 426, 427, 428, 429, 430, 431

P

`\pdfstringdefPreHook` 33, 35

prg commands:

`\prg_do_nothing:` 834, 1025

`\ProcessKeyOptions` 1173, 1174

`\ProcessKeysOptions` 1177

prop commands:

`\prop_clear:N` 1000

`\prop_get:NnNTF` 735, 740, 750, 753

`\prop_gset_eq:NN` 1004

`\prop_if_exist:NTF` 964

`\prop_map_function:NN` 729, 845

`\prop_new:N` 715, 716, 717, 718, 1001

`\prop_put:Nnn` 698, 702, 705, 709, 712, 714

`\prop_put_if_new:Nnn` 701, 708

`\prop_set_eq:NN` 971

Q

quark commands:

`\q_mark` 64, 66

`\q_nil` 9, 52, 56, 64, 230

`\quark_if_nil:nTF` 55, 62, 68

`\quark_if_recursion_tail_stop:N` 235, 366

`\quark_if_recursion_tail_stop_do:Nn` 173

`\q_recursion_stop` 144, 169, 175, 230, 347

`\q_recursion_tail` 9, 144, 230, 347

`\q_stop` 52, 53, 56, 60, 64, 66, 392, 393, 395, 420, 422, 480, 484, 646, 651

R

`\RequirePackage` 13, 1176

`reset` 4, 1070

S

str commands:

`\c_colon_str` 484

`\str_head:n` 817

`\str_if_empty:NTF` 1179

`\str_if_eq:NNTF` 954

`\str_lowercase:n` 1066

`\str_new:N` 1005, 1006, 1069

`\str_set:Nn` 1028, 1051, 1052, 1065

`\str_set_eq:NN` 970, 1029

`\str_tail:n` 818

style 3, 1070

sys commands:

`\sys_if_engine luatex_p:` 1019

`\sys_if_engine uptex_p:` 1020

`\sys_if_engine xetex_p:` 1018

T

TeX and L^AT_EX₂_ε commands:

`\@ifpackagelater` 10

`\@zhdig` 319

`\@zhnum` 137

`\edef` 5

`\g@addto@macro` 33, 35

`\pagenumbering` 2

`\pdfstringdef` 5

`\tex_expanded:D` 5

`\tiangan` 4

`\zhdate` 2

`\zhdig` 1, 2, 4

`\zhdigits` 1, 3, 4

`\zhdizhi` 2

`\zhganzhi` 2

`\zhganzhinian` 3

`\zhnum` 1, 2, 4

`\zhnumber` 1, 3–5

`\zhnumExtendScaleMap` 3

`\zhnumsetup` 1, 3, 4

`\zhtiangan` 2

`\zhtime` 2

`\zhweekday` 2

- tex commands:
- \tex_day:D 402
 - \tex_ignorespaces:D 1124
 - \tex_month:D 401
 - \tex_time:D 491, 492
 - \tex_year:D 400
- time 3
- tl commands:
- \tl_build_begin:N 605
 - \tl_build_end:N 692
 - \tl_build_get:NN 645
 - \tl_build_put_right:Nn 609, 620, 632, 635, 650, 653, 662, 670, 678, 686
 - \tl_clear_new:N ... 721, 722, 723, 724, 725, 726, 727, 728
 - \tl_concat:NNN 815, 861, 895, 899, 903
 - \tl_const:Nn 908
 - \tl_count:n 200
 - \tl_if_blank:nTF 75, 89, 95
 - \tl_if_empty:nTF 1131, 1140, 1155, 1167
 - \tl_if_eq:NNTF 1143, 1153, 1162
 - \tl_if_exist:nTF 507, 534, 556
 - \tl_if_novalue:nTF 38, 101, 301, 323, 522
 - \tl_new:N 517, 536, 602, 603, 608, 619, 761, 762, 763, 855, 856, 857, 914, 988
 - \tl_put_right:Nn .. 766, 774, 779, 787, 793, 799, 808, 813
 - \tl_set:Nn 539, 755, 768, 776, 781, 810, 837, 850
 - \tl_set_eq:NN 789, 795, 801, 868, 871, 875, 910, 946
 - \tl_use:N 893, 894, 923, 924, 930, 934, 939, 940
- tl internal commands:
- _tl_act:NNNnn 8
- token commands:
- \token_if_eq_charcode:NNTF 367
 - \TrimSpaces 518
- U**
- use commands:
- \use:n .. 24, 25, 143, 151, 163, 175, 219, 346, 353, 371, 481
 - \use_i:nn 194, 237
 - \use_i:nnn 221
 - \use_i_ii:nnn 220
 - \use_ii:nn 196
- Z**
- \zhcurrtime 2, 487
 - \zhdate 2, 390
 - \zhdig 2, 4, 299
 - \zhdigits 1, 4, 321
 - \zhdigitsoptions 321
 - \zhdigitsoptions 303, 306
 - \zhdizhi 2, 579
 - \zhganzhi 2, 581
 - \zhganzhinian 3, 583
 - \zhnum 2, 4, 99
- zhnum commands:
- \zhnum_blank_to_zero:n 70, 72, 80, 82, 87, 93, 93
 - \zhnum_check_financial:nn 719
 - \zhnum_check_simp:nn 719, 729, 733
 - \zhnum_counter:n 102, 110, 113, 113
 - \zhnum_decimal:nn 57, 85, 85
 - \zhnum_digits:Nn 324, 332, 336, 338, 340, 341, 389
 - \zhnum_digits_counter:n 302, 310, 313
 - \zhnum_digits_null:n 316, 320, 335, 337, 339, 407
 - \zhnum_digits_zero:n 91, 335, 335
 - \zhnum_exp_not:n 15, 20, 21, 24, 25, 29, 30, 411
 - \zhnum_expand_wrap:wn 14, 16, 50, 123, 340, 412, 421, 478, 487, 542, 551, 585
 - \zhnum_ganzhi:nnn 575, 575, 576, 578, 580, 582, 1109, 1111
 - \zhnum_ganzhi_cyclic:nnn 551, 552, 1109
 - \zhnum_ganzhi_nian:n 584, 585, 586, 601
 - \zhnum_ganzhi_normal:nnn 542, 543, 575, 1111
 - \zhnum_if_digit:NNTF 171, 191, 191, 363
 - \zhnum_input_cfg:n 966, 973
 - \zhnum_int:n 113, 116, 124, 137, 407, 497
 - \zhnum_integer:n 11, 63, 138, 138
 - \zhnum_load_cfg:n 881, 951, 951, 961, 1067
 - \zhnum_number:n 39, 47, 50, 51, 59, 77, 97
 - \zhnum_output:n 14, 14, 71, 78, 81, 88, 90, 96, 131, 134, 184, 187, 206, 239, 254, 257, 258, 263, 267, 269, 270, 276, 286, 289, 291, 292, 297, 368, 376, 380, 415, 416, 417, 425, 426, 427, 428, 429, 430, 431, 502, 503, 508, 548, 557, 567, 570, 571, 589, 593
 - \zhnum_parse_config: 719, 719, 958
 - \zhnum_parse_number:n 127, 132, 199, 199, 213
 - \zhnum_parse_number:nn 188, 199, 200, 201
 - \zhnum_process_number:NNNNNN 240, 249, 249
 - \zhnum_reset_all: 731, 880, 882, 955
 - \zhnum_reset_config: 880, 880, 1023, 1134
 - \zhnum_reset_ganzhi: 843, 866, 888, 1165
 - \zhnum_reset_main: 884, 891
 - \zhnum_reset_normal: 889, 918, 926, 1154
 - \zhnum_reset_simp: 885, 917, 920
 - \zhnum_reset_style: 880, 915, 1141, 1144, 1181
 - \zhnum_reset_week_day: 821, 823, 834, 887, 1164
 - \zhnum_scale_map:n 241, 505, 505, 512, 513
 - \zhnum_scale_map_hook:n 509, 516, 523
 - \zhnum_scale_map_loop:n 505, 511, 516
 - \zhnum_set:n 46, 109, 309, 331, 1123, 1126
 - \zhnum_set_active: 1036, 1039
 - \zhnum_set_catcode: 980, 1023, 1025, 1033
 - \zhnum_set_cfg_name:Nn 953, 1023, 1026, 1049
 - \zhnum_set_digits_map:nn 697, 697, 1189, 1191, 1193, 1196, 1198, 1199, 1200, 1201, 1202, 1203, 1204, 1205, 1206, 1207, 1208, 1209, 1219, 1221, 1222, 1230, 1231, 1233, 1234, 1235, 1236, 1237, 1245, 1246, 1248, 1254, 1256, 1257, 1258, 1264, 1266, 1267
 - \zhnum_set_digits_map:nnn 697, 699, 1185, 1186, 1210, 1211, 1212, 1213, 1215, 1216, 1224, 1225, 1226, 1227, 1239, 1240, 1241, 1242, 1250, 1251, 1260, 1261
 - \zhnum_set_dizhi_map:nn ... 697, 713, 1306, 1307, 1308, 1309, 1310, 1311, 1312, 1313, 1314, 1315, 1316, 1317
 - \zhnum_set_encoding:n 1063, 1063, 1074, 1180

\zhnum_set_financial_map:nn	697, 704, 1268, 1269, 1270, 1278, 1279, 1281, 1282, 1288, 1290, 1291, 1292, 1293, 1294, 1295
\zhnum_set_financial_map:nnn	697, 706, 1272, 1273, 1274, 1275, 1284, 1285
\zhnum_set_ganzhi:	730, 843, 843
\zhnum_set_other:	1037, 1044
\zhnum_set_scale:n	521, 525, 525
\zhnum_set_tiangang_map:nn	697, 711, 1296, 1297, 1298, 1299, 1300, 1301, 1302, 1303, 1304, 1305
\zhnum_set_week_day:	719, 821, 821, 886
\zhnum_split_number:nn	209, 226, 227, 227, 232
\zhnum_split_number:NNnNNNw	229, 233, 233, 244, 245, 248
\zhnum_two_digits:n	434, 437, 448
\zhnum_update_cfg:n	957, 962
\zhnum_Zeller:nnn	424, 434, 434
\zhnum_Zeller_aux:Nnnn	434
\zhnum_Zeller_Gregorian:nnn	438, 453, 453
\zhnum_Zeller_Julian:nnn	439, 466, 466
\zhnum_zero_mod:nn	851, 863, 864
zhnum internal commands:	
\l_zhnum_active_char_bool	1035, 1055, 1060, 1061, 1106
_zhnum_add_reset:nn	759, 764, 847
_zhnum_add_reset_ancient:nN	737, 806
_zhnum_add_reset_financial:n	744, 785
_zhnum_add_reset_simp:nNN	756, 772
\l_zhnum_ancient_bool	938, 1082, 1087, 1092, 1117
\l_zhnum_ancient_tl	736, 737, 763
\l_zhnum_arabic_sep_tl	408, 498, 1113
\l_zhnum_cfg_file_tl	975, 981, 988
\l_zhnum_cfg_map_finan_prop	697, 740, 995
\l_zhnum_cfg_map_ganzhi_prop	697, 846, 996
\l_zhnum_cfg_map_prop	697, 729, 993
\l_zhnum_cfg_map_var_prop	697, 735, 750, 753, 994
\l_zhnum_cfg_str	953, 954, 964, 970, 990, 1005
_zhnum_check_simp_aux:nn	739, 742, 748
_zhnum_counter_error:n	117, 119, 317
_zhnum_date:www	392, 395
_zhnum_date_aux:nnn	396, 399, 404, 404
_zhnum_date_aux:Nnnnn	408, 410
_zhnum_date_aux:NNnnnn	407, 411, 413
\l_zhnum_encoding_str	881, 1028, 1029, 1051, 1054, 1065, 1067, 1069, 1179
_zhnum_expand_wrap_aux:nn	17, 18
_zhnum_fraction:www	64, 66, 66
_zhnum_ganzhi_cyclic_mod:nnnn	551, 559, 564, 574
_zhnum_integer_or_fraction:www	56, 60, 60
\l_zhnum_kv_tl	602, 605, 609, 620, 632, 635, 645, 650, 653, 662, 670, 678, 686, 692, 696, 1130, 1131, 1139, 1143, 1152, 1153, 1155, 1161, 1162, 1167, 1168
\l_zhnum_last_cfg_str	954, 970, 1006
_zhnum_loop_end:wnn	173, 179
\l_zhnum_minus_tl	637
\l_zhnum_normal_bool	928, 1083, 1088, 1093, 1116
\l_zhnum_null_bool	944, 1099
\l_zhnum_null_tl	638, 948
_zhnum_number:www	50, 52, 53
_zhnum_output:nnwnn	172, 177
_zhnum_output_digits:NN	364, 378
_zhnum_parse_number:nnn	202, 203
_zhnum_prop_gset_eq:Nn	982, 1003
_zhnum_prop_initial:Nn	978, 998
_zhnum_read_abs_loop:Nw	165, 169, 169, 174
_zhnum_read_digits:w	343, 373
_zhnum_read_digits_loop:NN	356, 361, 370
_zhnum_read_integer:www	140, 181, 181
_zhnum_read_sign_loop:N	142, 147, 150
_zhnum_read_sign_loop:NN	345, 349, 352
_zhnum_read_zeros_loop:N	155, 159, 162
_zhnum_reset_ancient:	931, 936
\l_zhnum_reset_ancient_tl	726, 808, 939
\l_zhnum_reset_bool	977, 1007, 1105, 1128, 1133, 1145
\l_zhnum_reset_financial_tl	728, 799, 900, 901, 904, 905, 934
_zhnum_reset_ganzhi:nn	849
\l_zhnum_reset_normal_tl	727, 793, 896, 897, 930
\l_zhnum_reset_simp_tl	722, 774, 923
\l_zhnum_reset_tl	721, 766, 893
\l_zhnum_reset_trad_tl	723, 779, 924
_zhnum_reset_zero:	932, 942
_zhnum_result:nn	145, 177, 178, 179
\l_zhnum_scale_int	512, 514, 515, 537
\l_zhnum_set_ancient_tl	724, 813, 906, 940
_zhnum_set_cfg_prop:	965, 968, 984
\l_zhnum_set_normal_tl	725, 787, 894
_zhnum_set_pre_pos:	1156, 1158
_zhnum_set_scale:Nn	528, 532
_zhnum_set_style:	1132, 1136
_zhnum_set_user:	1146, 1149
_zhnum_set_week_day:nn	825, 826, 827, 828, 829, 830, 831, 835
\c_zhnum_set_zero_tl	898, 902, 908
\l_zhnum_simp_bool	922, 1095, 1096, 1115
\l_zhnum_simp_tl	754, 755, 757, 761
_zhnum_split_number_aux:nnn	210, 214, 214
_zhnum_split_number_aux:wnn	216, 225
_zhnum_time:ww	480, 483
_zhnum_time_aux:nn	486, 490, 494, 494
_zhnum_time_aux:Nnnn	494, 497, 498, 500
\l_zhnum_time_bool	406, 496, 1102, 1103
_zhnum_tmp:w	646, 651, 859, 874
\l_zhnum_tmp_int	520, 527, 529, 541
\l_zhnum_tmp_tl	603, 645, 648, 740, 743, 1139, 1140, 1143, 1152, 1153, 1161, 1162
\l_zhnum_trad_tl	751, 757, 762
\c_zhnum_unicode_engine_bool	1014, 1023
_zhnum_update_cfg_prop:N	971, 978, 982, 989
_zhnum_update_cfg_prop_aux:nN	990, 991
_zhnum_week_day:www	393, 420, 421, 422
\l_zhnum_weekday_tl	656, 839
_zhnum_Zeller_aux:Nnnn	438, 439, 442

_zhnum_zero_mod_aux:nn	852, 853	\zhnumsetup	3, <u>1121</u>
\l_zhnum_zero_tl	911, 914	\zhnumwithoptions	103, 106
\zhnumber	1, 4, <u>36</u>	\zhtiangan	2, <u>577</u>
\zhnumberwithoptions	40, <u>43</u> , <u>99</u>	\zhtime	2, <u>478</u>
\zhnumClearWrapper	22, 33, 35	\zhtoday	2, <u>397</u>
\zhnumExtendScaleMap	3, <u>518</u>	\zhweekday	2, <u>419</u>
\zhnumResetWrapper	27		