

zhnumber 宏包

李清

sobenlee@gmail.com

2014/09/13 v2.1*

1 简介

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

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

2 使用方法

`encoding`
Updated: 2014-09-09

`encoding = <GBK|Big5|UTF8>`

用于指定编码的宏包选项, 可以在调用宏包的时候设定, 也可以用 `\zhnumsetup` 在导言区内设定。对于 `upLATEX`, `XYLATEX` 和 `LuaLATEX`, 不用指定编码, 宏包将自动使用 UTF8 编码。只有 `LATEX` 和 `pdfLATEX` 需要指定编码, 如果没有指定, 默认将使用 GBK。

`\zhnumber` ☆
Updated: 2014-09-12

`\zhnumber <{number}>`

以中文格式输出数字。这里的数字可以是整数、小数和分数。例如

二十亿零一千二百零二万零一百二十
二十亿零一千二百零二万零一百二十
二十亿零一千二百零二万零一百二十
二千零一十二点零二零二零
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
二零零一又一百二十分之二千零二十

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

`\zhdigits` ☆
Updated: 2014-09-09

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

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

二〇一二〇二〇一二〇
二零二零二零二零二零

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

`\zhnum` ☆
Updated: 2014-09-09

`\zhnum <{counter}>`

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

二

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

*ctex-kit rev718.

`\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

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

以中文格式输出日期,其中带 * 的命令还输出星期。例如

2012 年 5 月 21 日
2012 年 5 月 21 日星期一

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

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

New: 2012-05-25

2014 年 9 月 14 日

```
1 \zhtoday
```

`\zhtime` ☆ `\zhtime {<hh:mm>}`

New: 2012-05-25

以中文格式输出时间。例如

23 时 56 分

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

`\zhcurrtime` ☆ 输出当前的时间。例如

New: 2012-05-25

20 时 59 分

```
1 \zhcurrtime
```

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

New: 2012-05-25

缺省状态下 `\zhnumber` 能正确中文格式化的最大整数是 $10^{48} - 1$, `\zhdigits` 不受这个大小的限制。可以通过 `\zhnumExtendScaleMap` 来扩展 `\zhnumber`。 `<character>_i` 设置 $10^{4(i+11)}$ 。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 {<key>_1=<val>_1, <key>_2=<val>_2, ...}`

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。

`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。默认使用阿拉伯数字。例如

二〇一四年九月十四日二十时五十九分

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

`style` `style = {<Simplified>|<Traditional>|<Normal>|<Financial>|<Ancient>}`

Updated: 2012-05-25

意义分别为

- `Simplified` 以简体中文输出数字(对 Big5 编码无效);
- `Traditional` 以繁体中文输出数字(对 Big5 编码无效);
- `Normal` 以小写形式输出中文数字;
- `Financial` 以大写形式输出中文数字;
- `Ancient` 以廿输出 20, 以卅输出 30, 以卌输出 40, 以百输出 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 = <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。

`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
dot and parts
year month day hour minute weekday mon tue wed thu fri sat sun
```

其中 `-` 设置负, `-0` 设置○, `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 , 而 `F n` 设置数字 n 的大写。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 `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>}
```

Updated: 2014-09-09

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

3 zhnumber 宏包代码实现

```
1 (*package)
2 (@@=zhnum)
3 \msg_new:nnn { zhnumber } { l3-too-old }
4 {
5   Support~package~'expl3'~too~old. \\\
6   Please~update~an~up~to~date~version~of~the~bundles\\
7   'l3kernel'~and~'l3packages'\\
8   using~your~TeX~package~manager~or~from~CTAN.
9 }
10 \ifpackagelater { expl3 } { 2014/08/25 } { }
11 { \msg_error:nn { zhnumber } { l3-too-old } }
12 \RequirePackage { xparse , l3keys2e }
13 \DeclareExpandableDocumentCommand \zhnumber { +o +m }
14 {
15   \IfNoValueTF {#1}
16     { \zhnum_number:f }
17     { \zhnumberwithoptions {#1} }
18 {#2}
19 }
```

`\zhnumber` 用于将输入的数字按照中文格式输出。

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

\zhnumberwithoptions 带选项的用户函数。

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

(End definition for \zhnumberwithoptions. This function is documented on page ??.)

\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 }
```

(End definition for \zhnum_number:n.)

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

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

(End definition for _zhnum_integer_or_fraction:www.)

_zhnum_fraction:www 对分数进行预处理。

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

(End definition for _zhnum_fraction:www.)

\zhnum_decimal:nn 对小数进行预处理。

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

(End definition for \zhnum_decimal:nn.)

`\zhnum_blank_to_zero:n` 输出小数的整数位。

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

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

`\zhnum` 用于将 \LaTeX 计数器按中文格式输出。

```
\zhnumberwithoptions 74 \DeclareExpandableDocumentCommand \zhnum { +o +m }
75 {
76   \IfNoValueTF {#1}
77     { \zhnum_counter:n }
78     { \zhnumwithoptions {#1} }
79   {#2}
80 }
81 \NewDocumentCommand \zhnumwithoptions { +m +m }
82 {
83   \group_begin:
84     \keys_set:nn { zhnum / options } {#1}
85     \zhnum_counter:n {#2}
86   \group_end:
87 }
```

(End definition for `\zhnum` and `\zhnumberwithoptions`. These functions are documented on page 3.)

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

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

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

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

```
110 \cs_new:Npn \zhnum_integer:n #1
111 {
112   \exp_after:wN \__zhnum_read_integer:www
113   \tex_number:D
114   \exp_after:wN \__zhnum_read_sign_loop:N
115   \tex_romannumeral:D -`0 \use:n
116   #1 \exp_stop_f: \q_recursion_tail \q_recursion_stop
117   \__zhnum_result:nn { \c_zero } { } ;
118 }
119 \cs_new:Npn \__zhnum_read_sign_loop:N #1
120 {
```

```

121 \if:w + \if:w - \exp_not:N #1 + \fi: \exp_not:N #1
122 \exp_after:wN \__zhnum_read_sign_loop:N
123 \tex_romannumerals:D -`0 \exp_after:wN \use:n
124 \else:
125 1 \exp_after:wN ;
126 \tex_romannumerals:D -`0
127 \exp_after:wN \__zhnum_read_zeros_loop:N
128 \exp_after:wN #1
129 \fi:
130 }
131 \cs_new:Npn \__zhnum_read_zeros_loop:N #1
132 {
133 \if:w 0 \exp_not:N #1
134 \exp_after:wN \__zhnum_read_zeros_loop:N
135 \tex_romannumerals:D -`0 \exp_after:wN \use:n
136 \else:
137 \exp_after:wN \__zhnum_read_abs_loop:Nw
138 \exp_after:wN #1
139 \fi:
140 }

```

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

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

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

我们在这里参考 `__tl_act:NNNnn` 的实现对它进行了改进。

```

141 \cs_new:Npn \__zhnum_read_abs_loop:Nw #1#2 \q_recursion_stop
142 {
143 \zhnum_if_digit:NTF #1
144 { \__zhnum_output:nnwnn { + \c_one } #1 }
145 { \quark_if_recursion_tail_stop_do:Nn #1 { \__zhnum_loop_end:wnn } }
146 \exp_after:wN \__zhnum_read_abs_loop:Nw
147 \tex_romannumerals:D -`0 \use:n #2 \q_recursion_stop
148 }
149 \cs_new:Npn \__zhnum_output:nnwnn #1#2#3 \__zhnum_result:nn #4#5
150 { #3 \__zhnum_result:nn { #4#1 } { #5#2 } }
151 \cs_new:Npn \__zhnum_loop_end:wnn #1 \__zhnum_result:nn #2#3
152 { \int_eval:n {#2} ; #3 }

```

(End definition for `__zhnum_read_abs_loop:Nw`.)

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

```

153 \cs_new:Npn \__zhnum_read_integer:www #1 ; #2 ; #3 ;
154 {
155 \int_compare:nNnTF {#2} = \c_zero
156 { \c__zhnum_zero_tl }
157 {
158 \int_compare:nNnF {#1} = \c_one
159 { \c__zhnum_minus_tl }
160 \zhnum_parse_number:nn {#2} {#3}
161 }
162 }

```

(End definition for `__zhnum_read_integer:www`.)

`\zhnum_if_digit:NTF` 判断 #1 是否为数字位。

```

163 \cs_new:Npn \zhnum_if_digit:NTF #1
164 {
165 \if_int_compare:w \c_nine < 1 \exp_not:N #1 \exp_stop_f:
166 \exp_after:wN \use_i:nn
167 \else:
168 \exp_after:wN \use_ii:nn
169 \fi:
170 }

```

(End definition for `\zhnum_if_digit:NTF`.)

```

\zhnum_parse_number:n
\zhnum_parse_number:nn 171 \cs_new:Npn \zhnum_parse_number:n #1
172 { \exp_args:Nf \zhnum_parse_number:nn { \tl_count:n {#1} } {#1} }
173 \cs_new:Npn \zhnum_parse_number:nn #1
174 { \exp_args:Nf \__zhnum_parse_number:nnn { \int_mod:nn {#1} \c_four } {#1} }
175 \cs_new:Npn \__zhnum_parse_number:nnn #1#2
176 {
177   \int_compare:nNnTF {#2} < \c_two
178   { \zhnum_digit_map:n }
179   {
180     \int_compare:nNnTF {#1} = \c_zero
181     { \zhnum_split_number:fn { \int_eval:n { #2 / \c_four - \c_one } } }
182     { \__zhnum_split_number_aux:nnn {#1} {#2} }
183   }
184 }
185 \cs_generate_variant:Nn \zhnum_parse_number:n { f }

```

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

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

```

186 \cs_new:Npn \__zhnum_split_number_aux:nnn #1#2
187 {
188   \exp_after:wN \__zhnum_split_number_aux:wnn
189   \tex_number:D \int_div_truncate:nn {#2} \c_four
190   \if_case:w #1 \exp_stop_f:
191     \or: \exp_after:wN \use:n
192     \or: \exp_after:wN \use_i_ii:nnn
193     \or: \exp_after:wN \use_i:nnn
194   \fi:
195   { \exp_stop_f: ; 0 } 0 0 ;
196 }
197 \cs_new:Npn \__zhnum_split_number_aux:wnn #1 ; #2 ; #3
198 { \zhnum_split_number:nn {#1} { #2#3 } }

```

(End definition for __zhnum_split_number_aux:nnn.)

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

```

199 \cs_new:Npn \zhnum_split_number:nn #1#2
200 {
201   \zhnum_split_number:NNnNNNw \c_true_bool \c_true_bool {#1}
202   #2 \q_recursion_tail \q_nil \q_nil \q_nil \q_recursion_stop
203 }
204 \cs_generate_variant:Nn \zhnum_split_number:nn { f }

```

(End definition for \zhnum_split_number:nn.)

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

```

205 \cs_new:Npn \zhnum_split_number:NNnNNNw #1#2#3#4#5#6#7#8 \q_recursion_stop
206 {
207   \quark_if_recursion_tail_stop:N #4
208   \int_compare:nNnTF { #4#5#6#7 } = \c_zero
209   { \use_i:nn }
210   {
211     \bool_if:NF #1 { \c_zhnum_zero_tl }
212     \zhnum_process_number:NNNNNN #4#5#6#7#1#2
213     \zhnum_scale_map:n {#3}
214     \int_compare:nNnTF {#7} = \c_zero
215     }
216     { \zhnum_split_number:NNfNNNw \c_false_bool \c_true_bool }
217     { \zhnum_split_number:NNfNNNw \c_true_bool \c_false_bool }
218   { \int_eval:n { #3 - \c_one } } #8 \q_recursion_stop
219 }
220 \cs_generate_variant:Nn \zhnum_split_number:NNnNNNw { NNf }

```

(End definition for \zhnum_split_number:NNnNNNw.)

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

```
221 \cs_new:Npn \zhnum_process_number:NNNNNN #1#2#3#4#5#6
222 {
223   \int_compare:nNnTF {#1} = \c_zero
224     { \bool_if:NF #6 { \c__zhnum_zero_tl } }
225     { \zhnum_digit_map:n {#1} \c__zhnum_thousand_tl }
226   \int_compare:nNnTF {#2} = \c_zero
227     { \int_compare:nNnF { #1 * (#3#4) } = \c_zero { \c__zhnum_zero_tl } }
228     {
229       \bool_if:nTF
230         { \l__zhnum_ancient_bool && \int_compare_p:nNn {#2} = \c_two }
231         { \zhnum_digit_map:n { #2 00 } }
232         { \zhnum_digit_map:n {#2} \c__zhnum_hundred_tl }
233     }
234   \int_compare:nNnTF {#3} = \c_zero
235     { \int_compare:nNnF { #2 * #4 } = \c_zero { \c__zhnum_zero_tl } }
236     {
237       \bool_if:nF
238         {
239           \int_compare_p:nNn {#3} = \c_one &&
240           \int_compare_p:nNn {#1#2} = \c_zero && #6 && #5
241         }
242         {
243           \bool_if:nTF
244             {
245               \l__zhnum_ancient_bool &&
246               ( \int_compare_p:nNn {#3} = \c_two ||
247                 \int_compare_p:nNn {#3} = \c_three ||
248                 \int_compare_p:nNn {#3} = \c_four )
249             }
250             { \zhnum_digit_map:n { #3 0 } \use_none:n }
251             { \zhnum_digit_map:n {#3} }
252         }
253       \c__zhnum_ten_tl
254     }
255     \int_compare:nNnF {#4} = \c_zero { \zhnum_digit_map:n {#4} }
256   }
```

(End definition for \zhnum_process_number:NNNNNN.)

\zhdigits 将输入的数字输出为中文数字串输出。

```
\zhdigitswithoptions 257 \DeclareExpandableDocumentCommand \zhdigits { +s +o +m }
258 {
259   \IfNoValueTF {#2}
260     { \zhnum_digits:Nn #1 }
261     { \zhdigitswithoptions {#1} {#2} }
262   {#3}
263 }
264 \NewDocumentCommand \zhdigitswithoptions { +m +m +m }
265 {
266   \group_begin:
267     \keys_set:nn { zhnum / options } {#2}
268     \zhnum_digits:Nn #1 {#3}
269   \group_end:
270 }
```

(End definition for \zhdigits and \zhdigitswithoptions. These functions are documented on page 3.)

\zhnum_digits_zero:n 快捷方式。

```
\zhnum_digits_null:n 271 \cs_new_nopar:Npn \zhnum_digits_zero:n
272   { \zhnum_digits:Nn \BooleanTrue }
273 \cs_new_nopar:Npn \zhnum_digits_null:n
274   { \zhnum_digits:Nn \BooleanFalse }
275 \cs_generate_variant:Nn \zhnum_digits_null:n { V }
```

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

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

```
276 \cs_new:Npn \zhnum_digits:Nn #1#2
277 {
278   \exp_after:wN \__zhnum_read_digits:w
279   \tex_number:D
280   \exp_after:wN \__zhnum_read_sign_loop:NN \exp_after:wN #1
281   \tex_romannumerals:D -`0 \use:n
282   #2 \exp_stop_f: \q_recursion_tail \q_recursion_stop
283 }
284 \cs_new:Npn \__zhnum_read_sign_loop:NN #1#2
285 {
286   \if:w + \if:w - \exp_not:N #2 + \fi: \exp_not:N #2
287   \exp_after:wN \__zhnum_read_sign_loop:NN \exp_after:wN #1
288   \tex_romannumerals:D -`0 \exp_after:wN \use:n
289   \else:
290     1 \exp_after:wN ;
291     \exp_after:wN \__zhnum_read_digits_loop:NN
292     \exp_after:wN #1
293     \exp_after:wN #2
294   \fi:
295 }
296 \cs_new:Npn \__zhnum_read_digits_loop:NN #1#2
297 {
298   \zhnum_if_digit:NTF #2
299   { \__zhnum_output_digits:NN #1#2 }
300   {
301     \quark_if_recursion_tail_stop:N #2
302     \if:w . \exp_not:N #2 \exp_after:wN \c__zhnum_dot_tl \fi:
303   }
304   \exp_after:wN \__zhnum_read_digits_loop:NN \exp_after:wN #1
305   \tex_romannumerals:D -`0 \use:n
306 }
307 \cs_new:Npn \__zhnum_read_digits:w #1 ;
308 {
309   \int_compare:nNnF {#1} = \c_one
310   { \c__zhnum_minus_tl }
311 }
312 \cs_new:Npn \__zhnum_output_digits:NN #1#2
313 {
314   \cs:w
315     c__zhnum_
316     \if_int_compare:w #2 = \c_zero
317     \IfBooleanTF #1 { zero } { null }
318     \else:
319     #2
320     \fi:
321     _tl
322   \cs_end:
323 }
```

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

`\zhdate` 输出中文日期。

```
324 \DeclareExpandableDocumentCommand \zhdate { +s +m }
325 {
326   \__zhnum_date:www #2 \q_stop
327   \IfBooleanT #1
328   { \__zhnum_week_day:www #2 \q_stop }
329 }
330 \cs_new:Npn \__zhnum_date:www #1/#2/#3 \q_stop
331 {
332   \zhnum_check_time:Nn \zhnum_digits_null:n {#1} \c__zhnum_year_tl
333   \zhnum_check_time:Nn \zhnum_int:n {#2} \c__zhnum_month_tl
334   \zhnum_check_time:Nn \zhnum_int:n {#3} \c__zhnum_day_tl
335 }
```

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

`\zhtoday` 输出当天日期。

```
336 \cs_new_nopar:Npn \zhtoday
337 {
338   \zhnum_check_time:Nn \zhnum_digits_null:V \tex_year:D \c__zhnum_year_tl
339   \zhnum_check_time:Nn \zhnum_int:n \tex_month:D \c__zhnum_month_tl
340   \zhnum_check_time:Nn \zhnum_int:n \tex_day:D \c__zhnum_day_tl
341 }
```

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

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

```
342 \cs_new:Npn \zhnum_check_time:Nn #1
343 { \bool_if:NTF \l__zhnum_time_bool {#1} { \int_to_arabic:n } }
```

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

`\zhweekday` 输出星期

```
344 \cs_new:Npn \zhweekday #1
345 { \__zhnum_week_day:www #1 \q_stop }
```

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

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

```
346 \cs_new:Npn \__zhnum_week_day:www #1/#2/#3 \q_stop
347 {
348   \if_case:w \zhnum_Zeller:nnn {#1} {#2} {#3} \exp_stop_f:
349     \c__zhnum_sat_tl
350     \or: \c__zhnum_sun_tl
351     \or: \c__zhnum_mon_tl
352     \or: \c__zhnum_tue_tl
353     \or: \c__zhnum_wed_tl
354     \or: \c__zhnum_thu_tl
355     \or: \c__zhnum_fri_tl
356   \fi:
357 }
```

(End definition for `__zhnum_week_day:www`.)

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

```
\zhnum_Zeller_aux:Nnnn
\zhnum_two_digits:n
358 \cs_new:Npn \zhnum_Zeller:nnn #1#2#3
359 {
360   \int_compare:nNnTF
361     { #1 \zhnum_two_digits:n {#2} \zhnum_two_digits:n {#3} } > { 1582 10 04 }
362     { \__zhnum_Zeller_aux:Nnnn \zhnum_Zeller_Gregorian:nnn }
363     { \__zhnum_Zeller_aux:Nnnn \zhnum_Zeller_Julian:nnn }
364     {#1} {#2} {#3}
365   }
366 \cs_new:Npn \__zhnum_Zeller_aux:Nnnn #1#2#3#4
367 {
368   \int_compare:nNnTF {#3} < \c_three
369     { #1 { #2 - \c_one } { #3 + \c_twelve } {#4} }
370     { #1 {#2} {#3} {#4} }
371   }
372 \cs_new:Npn \zhnum_two_digits:n #1
373 {
374   \int_compare:nNnT {#1} < \c_ten { 0 }
375   \int_eval:n {#1}
376 }
```

(End definition for `\zhnum_Zeller:nnn`, `\zhnum_Zeller_aux:Nnnn`, and `\zhnum_two_digits:n`.)

`\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 $Y --$ 。

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

```

377 \cs_new:Npn \zhnum_Zeller_Gregorian:nnn #1#2#3
378 {
379   \int_mod:nn
380     {
381       (#3)
382       + \int_div_truncate:nn { 26 * ( #2 + \c_one ) } \c_ten
383       + (#1)
384       + \int_div_truncate:nn {#1} \c_four
385       + \c_six * \int_div_truncate:nn {#1} \c_one_hundred
386       + \int_div_truncate:nn {#1} { 400 }
387     }
388     { \c_seven }
389   }

```

(End definition for \zhnum_Zeller_Gregorian:nnn.)

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

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

```

390 \cs_new:Npn \zhnum_Zeller_Julian:nnn #1#2#3
391 {
392   \int_mod:nn
393     {
394       (#3)
395       + \int_div_truncate:nn { 26 * ( #2 + \c_one ) } \c_ten
396       + (#1)
397       + \int_div_truncate:nn {#1} \c_four
398       + \c_five
399     }
400     { \c_seven }
401   }

```

(End definition for \zhnum_Zeller_Julian:nnn.)

\zhtime 输出时间。

```

402 \cs_new:Npn \zhtime #1
403 { \__zhnum_time:ww #1 \q_stop }
404 \group_begin:
405 \char_set_lccode:nn { `; } { `: }
406 \tl_to_lowercase:n
407 {
408   \group_end:
409   \cs_new:Npn \__zhnum_time:ww #1 ; #2 \q_stop
410   {
411     \zhnum_check_time:Nn \zhnum_int:n {#1} \c__zhnum_hour_tl
412     \zhnum_check_time:Nn \zhnum_int:n {#2} \c__zhnum_minute_tl
413   }
414 }

```

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

\zhcurrtime 输出当前时间。

```

415 \cs_new_nopar:Npn \zhcurrtime
416 {
417   \zhnum_check_time:Nn \zhnum_int:n
418   { \int_div_truncate:nn \tex_time:D { 60 } } \c__zhnum_hour_tl
419   \zhnum_check_time:Nn \zhnum_int:n
420   { \int_mod:nn \tex_time:D { 60 } } \c__zhnum_minute_tl
421 }

```

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

\zhnum_digit_map:n 阿拉伯数字与中文数字的映射。

```

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

```

(End definition for \zhnum_digit_map:n.)

```

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

```

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

\zhnumExtendScaleMap 扩展进位系统。

```

436 \NewDocumentCommand \zhnumExtendScaleMap { > { \TrimSpaces } +o +m }
437 {
438   \int_zero:N \l_tmpa_int
439   \clist_map_function:nN {#2} \zhnum_set_scale:n
440   \IfNoValueF {#1}
441   { \cs_set:Npn \zhnum_scale_map_hook:n ##1 {#1} }
442 }

```

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

\zhnum_set_scale:n

```

443 \cs_new_protected:Npn \zhnum_set_scale:n #1
444 {
445   \int_incr:N \l_tmpa_int
446   \tl_set:Nx \l_tmpa_tl
447   { c__zhnum_s \int_eval:n { \l_tmpa_int + \c_eleven } _tl }
448   \tl_if_exist:cF { \l_tmpa_tl }
449   { \int_incr:N \l__zhnum_scale_int }
450   \tl_set:cn { \l_tmpa_tl } {#1}
451 }

```

(End definition for \zhnum_set_scale:n)

根据需要设置中文阿拉伯数字。

```

452 \group_begin:
453   \tl_set:Nn \l_tmpa_tl
454   {
455     - .tl_set:N = \l__zhnum_minus_tl ,
456     -0 .tl_set:N = \l__zhnum_null_tl ,
457   }
458   \tl_put_right:Nx \l_tmpa_tl
459   {
460     E2 .tl_set:N = \exp_not:c { l__zhnum_ 100 _tl } ,
461     E3 .tl_set:N = \exp_not:c { l__zhnum_ 1000 _tl } ,
462     FE2 .tl_set:N = \exp_not:c { l__zhnum_financial_ 100 _tl } ,
463     FE3 .tl_set:N = \exp_not:c { l__zhnum_financial_ 1000 _tl } ,
464   }
465   \clist_map_inline:nn
466   { 0 , 1 , 2 , 3 , 4 , 5 , 6 , 7 , 8 , 9 , 10 , 100 , 1000 }
467   {
468     \tl_put_right:Nx \l_tmpa_tl
469     {
470       #1 .tl_set:N = \exp_not:c { l__zhnum_ #1 _tl } ,
471       F#1 .tl_set:N = \exp_not:c { l__zhnum_financial_ #1 _tl } ,
472     }
473   }
474   \clist_map_inline:nn
475   { 20 , 30 , 40 , 200 }
476   {
477     \tl_put_right:Nx \l_tmpa_tl
478     { #1 .tl_set:N = \exp_not:c { l__zhnum_ #1 _tl } , }
479   }

```

```

480 \clist_map_inline:nn
481   { 4 , 8 , 12 , 16 , 20 , 24 , 28 , 32 , 36 , 40 , 44 }
482   {
483     \tl_put_right:Nx \l_tmpa_tl
484     { E#1 .tl_set:N = \exp_not:c { l__zhnum_s \int_eval:n { #1 / 4 } _tl } , }
485   }
486 \clist_map_inline:nn
487   {
488     dot , and , parts , year , month , day , weekday , hour , minute
489     mon , tue , wed , thu , fri , sat , sun
490   }
491   {
492     \tl_put_right:Nx \l_tmpa_tl
493     { #1 .tl_set:N = \exp_not:c { l__zhnum_ #1 _tl } , }
494   }
495 \use:x
496   {
497     \group_end:
498     \keys_define:nn { zhnum / options } { \exp_not:o \l_tmpa_tl }
499   }

```

\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 500 \cs_new_protected:Npn \zhnum_set_digits_map:nn #1#2
\zhnum_set_financial_map:nn 501   { \prop_put:Nnn \l__zhnum_cfg_map_prop {#1} {#2} }
\zhnum_set_financial_map:nnn 502 \cs_new_protected:Npn \zhnum_set_digits_map:nnn #1#2#3
  \l__zhnum_cfg_map_prop 503   {
  \l__zhnum_cfg_map_var_prop 504     \prop_put_if_new:Nnn \l__zhnum_cfg_map_prop {#1} {#3}
  \l__zhnum_cfg_map_finan_prop 505     \prop_put:Nnn \l__zhnum_cfg_map_var_prop {#1_#2} {#3}
  506   }
  507 \cs_new_protected:Npn \zhnum_set_financial_map:nn #1#2
  508   { \prop_put:Nnn \l__zhnum_cfg_map_finan_prop {#1} {#2} }
  509 \cs_new_protected:Npn \zhnum_set_financial_map:nnn #1#2#3
  510   {
  511     \prop_put_if_new:Nnn \l__zhnum_cfg_map_finan_prop {#1} {#3}
  512     \prop_put:Nnn \l__zhnum_cfg_map_var_prop { financial_#1_#2 } {#3}
  513   }
  514 \prop_new:N \l__zhnum_cfg_map_prop
  515 \prop_new:N \l__zhnum_cfg_map_var_prop
  516 \prop_new:N \l__zhnum_cfg_map_finan_prop

```

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

\zhnum_parse_config: 将 prop 表转化到单独的 tl 变量。

```

\zhnum_check_simp:nn 517 \cs_new_protected_nopar:Npn \zhnum_parse_config:
\zhnum_check_financial:nn 518   {
  \zhnum_set_zero: 519     \prop_map_function:NN \l__zhnum_cfg_map_prop \zhnum_check_simp:nn
  \zhnum_set_week_day: 520     \zhnum_set_zero:
  521     \zhnum_set_week_day:
  522     \bool_if:NF \l__zhnum_reset_bool
  523     {
  524       \zhnum_assgin_const:
  525       \bool_set_true:N \l__zhnum_reset_bool
  526     }
  527   }
  528 \cs_new_protected:Npn \zhnum_check_simp:nn #1#2
  529   {
  530     \__zhnum_check_simp_aux:nn {#2} {#1}
  531     \prop_get:NnNT \l__zhnum_cfg_map_finan_prop {#1} \l_tmpa_tl
  532     { \exp_args:No \__zhnum_check_simp_aux:nn { \l_tmpa_tl } { financial_ #1 } }
  533   }
  534 \cs_new_protected:Npn \__zhnum_check_simp_aux:nn #1#2
  535   {
  536     \prop_get:NnNTF \l__zhnum_cfg_map_var_prop { #2 _trad } \l_tmpa_tl
  537     {
  538       \prop_get:NnNF \l__zhnum_cfg_map_var_prop { #2 _simp } \l_tmpb_tl
  539       { \tl_set:Nn \l_tmpb_tl {#1} }
  540       \tl_set:cx { l__zhnum_ #2 _tl }
  541       {
  542         \exp_not:n { \bool_if:NTF \l__zhnum_simp_bool }

```

```

543         { \exp_not:o \l_tmpb_tl } { \exp_not:o \l_tmpa_tl }
544     }
545 }
546 { \tl_set:cn { l__zhnum_ #2 _tl } {#1} }
547 }
548 \cs_new_protected_nopar:Npn \zhnum_assgin_const:
549 {
550     \prop_map_function:NN \l__zhnum_cfg_map_prop \zhnum_check_financial:nn
551     \zhnum_set_alias:
552 }
553 \cs_new_protected:Npn \zhnum_check_financial:nn #1#2
554 {
555     \prop_get:NnNTF \l__zhnum_cfg_map_finan_prop {#1} \l_tmpa_tl
556     {
557         \zhnum_assgin_const_tl:cx { c__zhnum_ #1 _tl }
558         {
559             \exp_not:n { \bool_if:NTF \l__zhnum_normal_bool }
560             { \exp_not:c { l__zhnum_ #1 _tl } }
561             { \exp_not:c { l__zhnum_financial_ #1 _tl } }
562         }
563     }
564     {
565         \zhnum_assgin_const_tl:cx
566         { c__zhnum_ #1 _tl } { \exp_not:c { l__zhnum_ #1 _tl } }
567     }
568 }
569 \cs_new_protected_nopar:Npn \zhnum_set_zero:
570 {
571     \tl_set:cx { l__zhnum_0_tl }
572     {
573         \exp_not:n { \bool_if:NTF \l__zhnum_null_bool }
574         { \exp_not:o \l__zhnum_null_tl } { \exp_not:v { l__zhnum_0_tl } }
575     }
576 }
577 \cs_new_protected_nopar:Npn \zhnum_set_week_day:
578 {
579     \tl_set:Nx \l__zhnum_mon_tl
580     { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_1_tl } }
581     \tl_set:Nx \l__zhnum_tue_tl
582     { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_2_tl } }
583     \tl_set:Nx \l__zhnum_wed_tl
584     { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_3_tl } }
585     \tl_set:Nx \l__zhnum_thu_tl
586     { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_4_tl } }
587     \tl_set:Nx \l__zhnum_fri_tl
588     { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_5_tl } }
589     \tl_set:Nx \l__zhnum_sat_tl
590     { \exp_not:N \c__zhnum_weekday_tl \exp_not:v { l__zhnum_6_tl } }
591     \tl_set:Nx \l__zhnum_sun_tl
592     { \exp_not:N \c__zhnum_weekday_tl \exp_not:o \l__zhnum_day_tl }
593 }
594 \clist_map_inline:nn { mon , tue , wed , thu , fri , sat , sun }
595 { \tl_const:cx { c__zhnum_ #1 _tl } { \exp_not:c { l__zhnum_ #1 _tl } } }
596 \cs_new_eq:NN \zhnum_assgin_const_tl:cx \tl_const:cx
597 \AtEndOfPackage
598 { \cs_set_eq:NN \zhnum_assgin_const_tl:cx \tl_set:cx }

```

(End definition for `\zhnum_parse_config:` and others.)

`\zhnum_set_alias:` 一些易于使用的别名。

```

599 \cs_new_eq:NN \zhnum_set_alias:NN \cs_new_eq:NN
600 \cs_new_protected_nopar:Npx \zhnum_set_alias:
601 {
602     \zhnum_set_alias:NN \exp_not:N \c__zhnum_zero_tl
603     \exp_not:c { c__zhnum_ 0 _tl }
604     \zhnum_set_alias:NN \exp_not:N \c__zhnum_ten_tl
605     \exp_not:c { c__zhnum_ 10 _tl }
606     \zhnum_set_alias:NN \exp_not:N \c__zhnum_hundred_tl
607     \exp_not:c { c__zhnum_ 100 _tl }

```

```

608     \zhnum_set_alias:NN \exp_not:N \c__zhnum_thousand_tl
609         \exp_not:c { c__zhnum_ 1000 _tl }
610 }
611 \AtEndOfPackage
612 { \cs_set_eq:NN \zhnum_set_alias:NN \tl_set_eq:NN }

```

(End definition for \zhnum_set_alias:.)

\zhnum_load_cfg:n 根据选定编码载入配置文件。

```

613 \cs_new_protected:Npn \zhnum_load_cfg:n #1
614 {
615     \zhnum_set_cfg_name:Nn \l__zhnum_cfg_tl {#1}
616     \tl_if_eq:NNF \l__zhnum_cfg_tl \l__zhnum_last_cfg_tl
617         { \zhnum_update_cfg:n {#1} }
618     \zhnum_parse_config:
619 }
620 \cs_generate_variant:Nn \zhnum_load_cfg:n { o }
621 \cs_new_protected:Npn \zhnum_update_cfg:n #1
622 {
623     \prop_if_exist:cTF { g__zhnum_cfg_ \l__zhnum_cfg_tl _prop }
624         { \tl_set_eq:NN \l__zhnum_last_cfg_tl \l__zhnum_cfg_tl }
625         { \zhnum_input_cfg:n {#1} }
626     \__zhnum_update_cfg_prop:N \prop_set_eq:Nc
627 }
628 \cs_new_protected:Npn \zhnum_input_cfg:n #1
629 {
630     \file_if_exist_input:nTF { zhnumber - #1 .cfg }
631     {
632         \bool_set_false:N \l__zhnum_reset_bool
633         \__zhnum_update_cfg_prop:N \__zhnum_prop_initial:Nn
634         \group_begin:
635             \zhnum_set_catcode:
636         }
637     {
638         \msg_error:nxx { zhnumber } { file-not-found } {#1}
639         \use_none:nmn
640     }
641     \__zhnum_update_cfg_prop:N \__zhnum_prop_gset_eq:Nn
642     \group_end:
643 }
644 \cs_new_protected:Npn \__zhnum_update_cfg_prop:N #1
645 {
646     #1 \l__zhnum_cfg_map_prop      { g__zhnum_cfg_ \l__zhnum_cfg_tl _prop }
647     #1 \l__zhnum_cfg_map_var_prop  { g__zhnum_cfg_var_ \l__zhnum_cfg_tl _prop }
648     #1 \l__zhnum_cfg_map_finan_prop { g__zhnum_cfg_finan_ \l__zhnum_cfg_tl _prop }
649 }
650 \cs_new_protected:Npn \__zhnum_prop_initial:Nn #1#2
651 {
652     \prop_clear:N #1
653     \prop_new:c {#2}
654 }
655 \cs_new_protected:Npn \__zhnum_prop_gset_eq:Nn #1#2
656 { \prop_gset_eq:cN {#2} #1 }
657 \tl_new:N \l__zhnum_cfg_tl
658 \tl_new:N \l__zhnum_last_cfg_tl
659 \bool_new:N \l__zhnum_reset_bool
660 \msg_new:nnnn { zhnumber } { file-not-found }
661 { File~`#1'~not~found. }
662 {
663     The~requested~file~could~not~be~found~in~the~current~directory,~
664     in~the~TeX~search~path~or~in~the~LaTeX~search~path.
665 }

```

(End definition for \zhnum_load_cfg:n.)

\zhnum_if_unicode_engine_p: 使用 upTeX 的时候,也不必将汉字的首字符设置为活动字符。判断 ~~~~0021 是否为单个记号的办法对 upTeX 不适用。因此,参考 ifuptex 宏包,通过 \kchar 是否为 primitive 来判断。

```

666 \pdfTeX_if_engine:TF
667 {

```

```

668 \str_if_eq_x:nnTF
669 { \token_to_str:N \kchar }
670 { \token_to_meaning:N \kchar }
671 }
672 { \use_i:nn }
673 {
674 \cs_new_eq:NN \zhnum_if_unicode_engine_p: \c_true_bool
675 \cs_new_eq:NN \zhnum_if_unicode_engine:TF \use_i:nn
676 }
677 {
678 \cs_new_eq:NN \zhnum_if_unicode_engine_p: \c_false_bool
679 \cs_new_eq:NN \zhnum_if_unicode_engine:TF \use_ii:nn
680 }

```

(End definition for \zhnum_if_unicode_engine:TF.)

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

```

\zhnum_set_cfg_name:Nn 681 \if_predicate:w \zhnum_if_unicode_engine_p:
\zhnum_reset_config: 682 \cs_new_eq:NN \zhnum_set_catcode: \prg_do_nothing:
683 \cs_new_protected:Npn \zhnum_set_cfg_name:Nn #1#2
684 {
685 \tl_set:Nx \l__zhnum_encoding_tl {#2}
686 \tl_set:Nx #1 { \tl_to_str:N \l__zhnum_encoding_tl }
687 }
688 \cs_new_eq:NN \zhnum_reset_config: \zhnum_parse_config:
689 \else:
690 \cs_new_protected_nopar:Npn \zhnum_set_catcode:
691 { \bool_if:NT \l__zhnum_active_char_bool { \zhnum_set_active: } }
692 \cs_new_protected_nopar:Npn \zhnum_set_active:
693 {
694 \str_case:onTF { \l__zhnum_encoding_tl }
695 {
696 { gbk } { \int_set:Nn \l__zhnum_byte_min_int { "81 } }
697 { big5 } { \int_set:Nn \l__zhnum_byte_min_int { "A1 } }
698 }
699 { \int_set:Nn \l__zhnum_byte_max_int { "FE } }
700 {
701 \int_set:Nn \l__zhnum_byte_min_int { "EO }
702 \int_set:Nn \l__zhnum_byte_max_int { "EF }
703 }
704 \int_step_function:nnnN
705 { \l__zhnum_byte_min_int } { \c_one }
706 { \l__zhnum_byte_max_int } \char_set_catcode_active:n
707 }
708 \int_new:N \l__zhnum_byte_min_int
709 \int_new:N \l__zhnum_byte_max_int
710 \cs_new_protected:Npn \zhnum_set_cfg_name:Nn #1#2
711 {
712 \tl_set:Nx \l__zhnum_encoding_tl {#2}
713 \tl_set:Nx #1
714 {
715 \tl_to_str:N \l__zhnum_encoding_tl
716 \bool_if:NT \l__zhnum_active_char_bool
717 { \tl_to_str:n { _active } }
718 }
719 }
720 \cs_new_protected_nopar:Npn \zhnum_reset_config:
721 { \zhnum_load_cfg:o { \l__zhnum_encoding_tl } }
722 \bool_new:N \l__zhnum_active_char_bool
723 \bool_set_true:N \l__zhnum_active_char_bool
724 \fi:

```

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

encoding 宏包设置选项。

```

style 725 \keys_define:nn { zhnum / options }
null 726 {
reset 727 encoding .choices:nn =
728 { UTF8 , GBK , Big5 }

```

```

729     {
730       \tl_set:Nx \l__zhnum_encoding_tl
731       { \exp_args:No \tl_expandable_lowercase:n { \l_keys_choice_tl } }
732       \zhnum_load_cfg:o { \l__zhnum_encoding_tl }
733     } ,
734     encoding .default:n = { GBK } ,
735     encoding / Bg5 .meta:n = { encoding = Big5 } ,
736     encoding / unknown .code:n =
737     { \msg_error:nnn { zhnumber } { encoding-invalid } {#1} } ,
738     style .multichoice: ,
739     style / Normal .code:n =
740     {
741       \bool_set_false:N \l__zhnum_ancient_bool
742       \bool_set_true:N \l__zhnum_normal_bool
743     } ,
744     style / Financial .code:n =
745     {
746       \bool_set_false:N \l__zhnum_ancient_bool
747       \bool_set_false:N \l__zhnum_normal_bool
748     } ,
749     style / Ancient .code:n =
750     {
751       \bool_set_true:N \l__zhnum_ancient_bool
752       \bool_set_true:N \l__zhnum_normal_bool
753     } ,
754     style / Simplified .code:n = { \bool_set_true:N \l__zhnum_simp_bool } ,
755     style / Traditional .code:n = { \bool_set_false:N \l__zhnum_simp_bool } ,
756     style .default:n = { Normal , Simplified } ,
757     null .bool_set:N = \l__zhnum_null_bool ,
758     time .choice: ,
759     time / Chinese .code:n = { \bool_set_true:N \l__zhnum_time_bool } ,
760     time / Arabic .code:n = { \bool_set_false:N \l__zhnum_time_bool } ,
761     time .default:n = { Arabic } ,
762     reset .code:n = { \zhnum_reset_config: } ,
763     activechar .bool_set:N = \l__zhnum_active_char_bool ,
764   }
765   \tl_new:N \l__zhnum_encoding_tl
766   \msg_new:nnnn { zhnumber } { encoding-invalid }
767   { The~encoding~`#1'~is~invalid. }
768   { Available~encodings~are~`UTF8',~`GBK'~and~`Big5'. }

```

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

\zhnumsetup 在文档中设置 zhnumber 的接口。

```

769 \NewDocumentCommand \zhnumsetup { +m }
770 {
771   \keys_set:nn { zhnum / options } {#1}
772   \tex_ignorespaces:D
773 }

```

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

初始化设置和执行宏包选项。

```

774 \keys_set:nn { zhnum / options } { style , time }
775 \ProcessKeysOptions { zhnum / options }

```

如果没有选定编码,则根据引擎自动设置编码。

```

776 \tl_if_empty:NT \l__zhnum_encoding_tl
777 {
778   \zhnum_if_unicode_engine:TF
779   { \keys_set:nn { zhnum / options } { encoding = UTF8 } }
780   { \keys_set:nn { zhnum / options } { encoding = GBK } }
781 }
782 </package>

```

4 中文数字配置文件

```
783 <!*config>
784 <!*big5>
785 \zhnum_set_digits_map:nnn { minus } { simp } { 負 }
786 \zhnum_set_digits_map:nnn { minus } { trad } { 負 }
787 </!big5>
788 <!*big5>
789 \zhnum_set_digits_map:nn { minus } { 負 }
790 </big5>
791 \zhnum_set_digits_map:nn { 0 } { 零 }
792 <!*big5>
793 \zhnum_set_digits_map:nn { null } { 〇 }
794 </!big5>
795 <!*big5>
796 \zhnum_set_digits_map:nn { null } { 〇 }
797 </big5>
798 \zhnum_set_digits_map:nn { 1 } { 一 }
799 \zhnum_set_digits_map:nn { 2 } { 二 }
800 \zhnum_set_digits_map:nn { 3 } { 三 }
801 \zhnum_set_digits_map:nn { 4 } { 四 }
802 \zhnum_set_digits_map:nn { 5 } { 五 }
803 \zhnum_set_digits_map:nn { 6 } { 六 }
804 \zhnum_set_digits_map:nn { 7 } { 七 }
805 \zhnum_set_digits_map:nn { 8 } { 八 }
806 \zhnum_set_digits_map:nn { 9 } { 九 }
807 \zhnum_set_digits_map:nn { 10 } { 十 }
808 \zhnum_set_digits_map:nn { 100 } { 百 }
809 \zhnum_set_digits_map:nn { 1000 } { 千 }
810 \zhnum_set_digits_map:nn { 20 } { 廿 }
811 \zhnum_set_digits_map:nn { 30 } { 卅 }
812 \zhnum_set_digits_map:nn { 40 } { 卌 }
813 \zhnum_set_digits_map:nn { 200 } { 貳 }
814 <!*big5>
815 \zhnum_set_digits_map:nnn { dot } { simp } { 点 }
816 \zhnum_set_digits_map:nnn { dot } { trad } { 點 }
817 </!big5>
818 <!*big5>
819 \zhnum_set_digits_map:nn { dot } { 點 }
820 </big5>
821 \zhnum_set_digits_map:nn { and } { 又 }
822 \zhnum_set_digits_map:nn { parts } { 分之 }
823 <!*big5>
824 \zhnum_set_digits_map:nnn { s1 } { simp } { 万 }
825 \zhnum_set_digits_map:nnn { s1 } { trad } { 萬 }
826 \zhnum_set_digits_map:nnn { s2 } { simp } { 亿 }
827 \zhnum_set_digits_map:nnn { s2 } { trad } { 億 }
828 </!big5>
829 <!*big5>
830 \zhnum_set_digits_map:nn { s1 } { 萬 }
831 \zhnum_set_digits_map:nn { s2 } { 億 }
832 </big5>
833 \zhnum_set_digits_map:nn { s3 } { 兆 }
834 \zhnum_set_digits_map:nn { s4 } { 京 }
835 \zhnum_set_digits_map:nn { s5 } { 垓 }
836 \zhnum_set_digits_map:nn { s6 } { 秭 }
837 \zhnum_set_digits_map:nn { s7 } { 穰 }
838 <!*big5>
839 \zhnum_set_digits_map:nnn { s8 } { simp } { 沟 }
840 \zhnum_set_digits_map:nnn { s8 } { trad } { 溝 }
841 \zhnum_set_digits_map:nnn { s9 } { simp } { 澗 }
842 \zhnum_set_digits_map:nnn { s9 } { trad } { 澗 }
843 </!big5>
844 <!*big5>
845 \zhnum_set_digits_map:nn { s8 } { 溝 }
846 \zhnum_set_digits_map:nn { s9 } { 澗 }
847 </big5>
848 \zhnum_set_digits_map:nn { s10 } { 正 }
```

```

849 (*!big5)
850 \zhnum_set_digits_map:nnn { s11 } { simp } { 載 }
851 \zhnum_set_digits_map:nnn { s11 } { trad } { 載 }
852 </!big5>
853 (*big5)
854 \zhnum_set_digits_map:nn { s11 } { 載 }
855 </big5>
856 \zhnum_set_digits_map:nn { year } { 年 }
857 \zhnum_set_digits_map:nn { month } { 月 }
858 \zhnum_set_digits_map:nn { day } { 日 }
859 (*!big5)
860 \zhnum_set_digits_map:nnn { hour } { simp } { 時 }
861 \zhnum_set_digits_map:nnn { hour } { trad } { 時 }
862 </!big5>
863 (*big5)
864 \zhnum_set_digits_map:nn { hour } { 時 }
865 </big5>
866 \zhnum_set_digits_map:nn { minute } { 分 }
867 \zhnum_set_digits_map:nn { weekday } { 星期 }
868 \zhnum_set_financial_map:nn { null } { 零 }
869 \zhnum_set_financial_map:nn { 0 } { 零 }
870 \zhnum_set_financial_map:nn { 1 } { 壹 }
871 \zhnum_set_financial_map:nn { 2 } { 貳 }
872 (*!big5)
873 \zhnum_set_financial_map:nnn { 3 } { simp } { 叁 }
874 \zhnum_set_financial_map:nnn { 3 } { trad } { 參 }
875 </!big5>
876 (*big5)
877 \zhnum_set_financial_map:nn { 3 } { 參 }
878 </big5>
879 \zhnum_set_financial_map:nn { 4 } { 肆 }
880 \zhnum_set_financial_map:nn { 5 } { 伍 }
881 (*!big5)
882 \zhnum_set_financial_map:nnn { 6 } { simp } { 陆 }
883 \zhnum_set_financial_map:nnn { 6 } { trad } { 陸 }
884 </!big5>
885 (*big5)
886 \zhnum_set_financial_map:nn { 6 } { 陸 }
887 </big5>
888 \zhnum_set_financial_map:nn { 7 } { 柒 }
889 \zhnum_set_financial_map:nn { 8 } { 捌 }
890 \zhnum_set_financial_map:nn { 9 } { 玖 }
891 \zhnum_set_financial_map:nn { 10 } { 拾 }
892 \zhnum_set_financial_map:nn { 100 } { 佰 }
893 \zhnum_set_financial_map:nn { 1000 } { 仟 }
894 </config>

```

5 代码索引

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

Symbols	
<code>\:</code>	405
<code>\;</code>	405
<code>\\</code>	5, 6, 7
A	
<code>activechar</code>	3
<code>\AtEndOfPackage</code>	597, 611
B	
bingint commands:	
<code>__bingint_read_do:nn</code>	5
bool commands:	
<code>\bool_if:Nf</code>	211, 224, 522
<code>\bool_if:nf</code>	237
<code>\bool_if:NT</code>	691, 716
<code>\bool_if:NfT</code>	343, 542, 559, 573
<code>\bool_if:nfT</code>	229, 243
<code>\bool_new:N</code>	659, 722
<code>\bool_set_false:N</code>	632, 741, 746, 747, 755, 760
<code>\bool_set_true:N</code>	525, 723, 742, 751, 752, 754, 759
<code>\BooleanFalse</code>	274
<code>\BooleanTrue</code>	272
C	
char commands:	
<code>\char_set_catcode_active:n</code>	706
<code>\char_set_lccode:nn</code>	405
clist commands:	
<code>\clist_map_function:nN</code>	439
<code>\clist_map_inline:nn</code>	465, 474, 480, 486, 594
cs commands:	
<code>\cs:w</code>	314
<code>\cs_end:</code>	322
<code>\cs_generate_variant:Nn</code>	35, 109, 185, 204, 220, 275, 431, 620
<code>\cs_if_exist_use:cF</code>	426
<code>\cs_new:Npn</code>	27, 29, 36, 42, 61, 68, 88, 94, 96, 110, 119, 131, 141, 149, 151, 153, 163, 171, 173, 175, 186, 197, 199, 205, 221, 276, 284, 296, 307, 312, 330, 342, 344, 346, 358, 366, 372, 377, 390, 402, 409, 422, 424, 429
<code>\cs_new_eq:NN</code> .	434, 596, 599, 674, 675, 678, 679, 682, 688
<code>\cs_new_nopar:Npn</code>	271, 273, 336, 415
<code>\cs_new_protected:Npn</code>	443, 500, 502, 507, 509, 528, 534, 553, 613, 621, 628, 644, 650, 655, 683, 710
<code>\cs_new_protected_nopar:Npn</code>	517, 548, 569, 577, 690, 692, 720
<code>\cs_new_protected_nopar:Npx</code>	600
<code>\cs_set:Npn</code>	441
<code>\cs_set_eq:NN</code>	598, 612
D	
<code>\DeclareExpandableDocumentCommand</code>	13, 74, 257, 324
E	
eleven commands:	
<code>\c_eleven</code>	433, 447
else commands:	
<code>\else:</code>	124, 136, 167, 289, 318, 689
encoding	1, 16
exp commands:	
<code>\exp_after:wN</code>	112, 114, 122, 123, 125, 127, 128, 134, 135, 137, 138, 146, 166, 168, 188, 191, 192, 193, 278, 280, 287, 288, 290, 291, 292, 293, 302, 304
<code>\exp_args:Nf</code>	172, 174
<code>\exp_args:No</code>	532, 731
<code>\exp_not:c</code>	460, 461, 462, 463, 470, 471, 478, 484, 493, 560, 561, 566, 595, 603, 605, 607, 609
<code>\exp_not:N</code>	121, 133, 165, 286, 302, 580, 582, 584, 586, 588, 590, 592, 602, 604, 606, 608
<code>\exp_not:n</code>	542, 559, 573
<code>\exp_not:o</code>	498, 543, 574, 592
<code>\exp_not:v</code>	574, 580, 582, 584, 586, 588, 590
<code>\exp_stop_f:</code>	116, 165, 190, 195, 282, 348
F	
false commands:	
<code>\c_false_bool</code>	216, 217, 678
fi commands:	
<code>\fi:</code> ...	121, 129, 139, 169, 194, 286, 294, 302, 320, 356, 724
file commands:	
<code>\file_if_exist_input:nTF</code>	630
five commands:	
<code>\c_five</code>	398
four commands:	
<code>\c_four</code>	174, 181, 189, 248, 384, 397
G	
group commands:	
<code>\group_begin:</code>	22, 83, 266, 404, 452, 634
<code>\group_end:</code>	25, 86, 269, 408, 497, 642
I	
if commands:	
<code>\if:w</code>	121, 133, 286, 302
<code>\if_case:w</code>	190, 348
<code>\if_int_compare:w</code>	165, 316
<code>\if_predicate:w</code>	681
<code>\IfBooleanT</code>	327
<code>\IfBooleanTF</code>	317
<code>\IfNoValueF</code>	440
<code>\IfNoValueTF</code>	15, 76, 259
int commands:	
<code>\int_compare:nNf</code>	158, 227, 235, 255, 309
<code>\int_compare:nNt</code>	374
<code>\int_compare:nNtF</code>	98, 101, 155, 177, 180, 208, 214, 223, 226, 234, 360, 368
<code>\int_compare_p:nNn</code>	230, 239, 240, 246, 247, 248
<code>\int_div_truncate:nn</code>	189, 382, 384, 385, 386, 395, 397, 418
<code>\int_eval:n</code>	99, 104, 152, 181, 218, 375, 447, 484
<code>\int_if_exist:cTF</code>	90
<code>\int_incr:N</code>	445, 449
<code>\int_mod:nn</code>	174, 379, 392, 420, 430
<code>\int_new:N</code>	432, 708, 709
<code>\int_set:Nn</code>	696, 697, 699, 701, 702

<code>\int_set_eq:NN</code>	433		
<code>\int_step_function:nnnN</code>	704		
<code>\int_to_arabic:n</code>	343		
<code>\int_zero:N</code>	438		
		K	
<code>\kchar</code>	669, 670		
keys commands:			
<code>\l_keys_choice_tl</code>	731		
<code>\keys_define:nn</code>	498, 725		
<code>\keys_set:nn</code>	23, 84, 267, 771, 774, 779, 780		
		M	
mark commands:			
<code>\q_mark</code>	40, 42		
msg commands:			
<code>\msg_error:nn</code>	11		
<code>\msg_error:nnn</code>	737		
<code>\msg_error:nnx</code>	638		
<code>_msg_expandable_error:n</code>	95		
<code>\msg_new:nnn</code>	3		
<code>\msg_new:nnnn</code>	660, 766		
		N	
<code>\NewDocumentCommand</code>	20, 81, 264, 436, 769		
nil commands:			
<code>\q_nil</code>	7, 28, 32, 40, 202		
nine commands:			
<code>\c_nine</code>	165		
null	3, 16		
		O	
one commands:			
<code>\c_one</code>	144, 158, 181, 218, 239, 309, 369, 382, 395, 705		
<code>\c_one_hundred</code>	385		
or commands:			
<code>\or:</code>	191, 192, 193, 350, 351, 352, 353, 354, 355		
		P	
pdftex commands:			
<code>\pdftex_if_engine:TF</code>	666		
prg commands:			
<code>\prg_do_nothing:</code>	682		
<code>\ProcessKeysOptions</code>	775		
prop commands:			
<code>\prop_clear:N</code>	652		
<code>\prop_get:NnNF</code>	538		
<code>\prop_get:NnNT</code>	531		
<code>\prop_get:NnNTF</code>	536, 555		
<code>\prop_gset_eq:cN</code>	656		
<code>\prop_if_exist:cTF</code>	623		
<code>\prop_map_function:NN</code>	519, 550		
<code>\prop_new:c</code>	653		
<code>\prop_new:N</code>	514, 515, 516		
<code>\prop_put:Nnn</code>	501, 505, 508, 512		
<code>\prop_put_if_new:Nnn</code>	504, 511		
<code>\prop_set_eq:Nc</code>	626		
		Q	
quark commands:			
<code>\quark_if_nil:nTF</code>	31, 38, 44		
<code>\quark_if_recursion_tail_stop:N</code>	207, 301		
<code>\quark_if_recursion_tail_stop_do:Nn</code>	145		
		R	
recursion commands:			
<code>\q_recursion_stop</code>	116, 141, 147, 202, 205, 218, 282		
<code>\q_recursion_tail</code>	7, 116, 202, 282		
<code>\RequirePackage</code>	12		
reset	3, 16		
		S	
seven commands:			
<code>\c_seven</code>	388, 400		
six commands:			
<code>\c_six</code>	385		
stop commands:			
<code>\q_stop</code> 28, 29, 32, 36, 40, 42, 326, 328, 330, 345, 346, 403, 409			
str commands:			
<code>\str_case:onTF</code>	694		
<code>\str_if_eq_x:nnTF</code>	668		
style	2, 16		
		T	
ten commands:			
<code>\c_ten</code>	374, 382, 395		
\TeX and $\LaTeX 2_\epsilon$ commands:			
<code>\ifpackagelater</code>	10		
<code>\kchar</code>	15		
<code>\zhdate</code>	2, 2		
<code>\zhdigits</code>	1, 1, 1, 1, 2, 3, 3		
<code>\zhnum</code>	1, 1, 3		
<code>\zhnumber</code>	1, 1, 2, 2, 3		
<code>\zhnumExtendScaleMap</code>	2, 2		
<code>\zhnumsetup</code>	1, 2, 3		
<code>\zhtime</code>	2		
<code>\zhweekday</code>	2		
tex commands:			
<code>\tex_day:D</code>	340		
<code>\tex_ignorespaces:D</code>	772		
<code>\tex_month:D</code>	339		
<code>\tex_number:D</code>	113, 189, 279		
<code>\tex_romannumeral:D</code> 115, 123, 126, 135, 147, 281, 288, 305			
<code>\tex_time:D</code>	418, 420		
<code>\tex_year:D</code>	338		
three commands:			
<code>\c_three</code>	247, 368		
time	2		
tl commands:			
<code>_tl_act:NNNnn</code>	6		
<code>\tl_const:cn</code>	435		
<code>\tl_const:cx</code>	595, 596		
<code>\tl_count:n</code>	172		
<code>\tl_expandable_lowercase:n</code>	731		
<code>\tl_if_blank:nF</code>	51		
<code>\tl_if_blank:nTF</code>	64, 70		
<code>\tl_if_empty:NT</code>	776		
<code>\tl_if_eq:NNF</code>	616		
<code>\tl_if_exist:cF</code>	448		
<code>\tl_new:N</code>	657, 658, 765		
<code>\tl_put_right:Nx</code>	458, 468, 477, 483, 492		
<code>\tl_set:cn</code>	450, 546		
<code>\tl_set:cx</code>	540, 571, 598		
<code>\tl_set:Nn</code>	453, 539		
<code>\tl_set:Nx</code>	446,		
579, 581, 583, 585, 587, 589, 591, 685, 686, 712, 713, 730			
<code>\tl_set_eq:NN</code>	612, 624		

\tl_to_lowercase:n	406
\tl_to_str:N	686, 715
\tl_to_str:n	717
tmpa commands:	
\l_tmpa_int	438, 445, 447
\l_tmpa_tl	446, 448, 450, 453, 458, 468, 477, 483, 492, 498, 531, 532, 536, 543, 555
tmpb commands:	
\l_tmpb_tl	538, 539, 543
token commands:	
\token_to_meaning:N	670
\token_to_str:N	669
\TrimSpaces	436
true commands:	
\c_true_bool	201, 216, 217, 674
twelve commands:	
\c_twelve	369
two commands:	
\c_two	177, 230, 246

U

use commands:	
\use:c	423
\use:n	115, 123, 135, 147, 191, 281, 288, 305
\use:x	495
\use_i:nn	166, 209, 672, 675
\use_i:nnn	193
\use_i_ii:nnn	192
\use_ii:nn	168, 679
\use_none:n	250
\use_none:nnn	639

Z

zero commands:	
\c_zero	98, 101, 117, 155, 180, 208, 214, 223, 226, 227, 234, 235, 240, 255, 316
\zhcurrtime	2, 11, 415
\zhdate	2, 9, 324
\zhdigits	1, 3, 8, 257
\zhdigitsoptions	8, 261, 264
\zhnum	1, 3, 5, 74
zhnum commands:	
\l_zhnum_active_char_bool	691, 716, 722, 723, 763
\l_zhnum_ancient_bool	230, 245, 741, 746, 751
\c_zhnum_and_tl	54
\zhnum_assgin_const:	524, 548
\zhnum_assgin_const_tl:cx	557, 565, 596, 598
\zhnum_blank_to_zero:n	5, 46, 48, 56, 58, 63, 68
\l_zhnum_byte_max_int	699, 702, 706, 709
\l_zhnum_byte_min_int	696, 697, 701, 705, 708
\l_zhnum_cfg_map_finan_prop	13, 508, 511, 516, 531, 555, 648
\l_zhnum_cfg_map_prop	13, 501, 504, 514, 519, 550, 646
\l_zhnum_cfg_map_var_prop	13, 505, 512, 515, 536, 538, 647
\l_zhnum_cfg_tl	615, 616, 623, 624, 646, 647, 648, 657
\zhnum_check_financial:nn	13, 550, 553
\zhnum_check_simp:nn	13, 519, 528
_zhnum_check_simp_aux:nn	530, 532, 534
\zhnum_check_time:Nn	10, 332, 333, 334, 338, 339, 340, 342, 411, 412, 417, 419
\zhnum_counter:n	5, 77, 85, 88
_zhnum_counter_error:n	92, 94

_zhnum_date:www	326, 330
\c_zhnum_day_tl	334, 340
\l_zhnum_day_tl	592
\zhnum_decimal:nn	4, 33, 61
\zhnum_digit_map:n	11, 178, 225, 231, 232, 250, 251, 255, 422
\zhnum_digits:Nn	9, 260, 268, 272, 274, 276
\zhnum_digits_null:n	8, 273, 275, 332
\zhnum_digits_null:V	338
\zhnum_digits_zero:n	8, 66, 271
\c_zhnum_dot_tl	63, 302
\l_zhnum_encoding_tl	685, 686, 694, 712, 715, 721, 730, 732, 765, 776
_zhnum_fraction:www	4, 40, 42
\c_zhnum_fri_tl	355
\l_zhnum_fri_tl	587
\c_zhnum_hour_tl	411, 418
\c_zhnum_hundred_tl	232, 606
\zhnum_if_digit:NTF	6, 143, 163, 298
\zhnum_if_unicode_engine:TF	15, 675, 679, 778
\zhnum_if_unicode_engine_p:	15, 674, 678, 681
\zhnum_input_cfg:n	625, 628
\zhnum_int:c	91
\zhnum_int:n	5, 96, 109, 333, 334, 339, 340, 411, 412, 417, 419
\zhnum_integer:n	5, 9, 39, 110
_zhnum_integer_or_fraction:www	4, 32, 36
\l_zhnum_last_cfg_tl	616, 624, 658
\zhnum_load_cfg:n	15, 613, 620
\zhnum_load_cfg:o	721, 732
_zhnum_loop_end:wnn	145, 151
\c_zhnum_minus_tl	103, 159, 310
\l_zhnum_minus_tl	455
\c_zhnum_minute_tl	412, 420
\c_zhnum_mon_tl	351
\l_zhnum_mon_tl	579
\c_zhnum_month_tl	333, 339
\l_zhnum_normal_bool	559, 742, 747, 752
\l_zhnum_null_bool	573, 757
\l_zhnum_null_tl	456, 574
\zhnum_number:f	16, 24
\zhnum_number:n	4, 27, 35, 53, 72
_zhnum_number:www	4, 28, 29
_zhnum_output:nwnn	144, 149
_zhnum_output_digits:NN	299, 312
\zhnum_parse_config:	13, 517, 618, 688
\zhnum_parse_number:f	99, 104
\zhnum_parse_number:n	7, 171, 185
\zhnum_parse_number:nn	7, 160, 172, 173
_zhnum_parse_number:nnn	174, 175
\c_zhnum_parts_tl	47, 57
\zhnum_process_number:NNNNNN	8, 212, 221
_zhnum_prop_gset_eq:Nn	641, 655
_zhnum_prop_initial:Nn	633, 650
_zhnum_read_abs_loop:Nw	6, 137, 141, 146
_zhnum_read_digits:w	278, 307
_zhnum_read_digits_loop:NN	291, 296, 304
_zhnum_read_integer:www	6, 112, 153
_zhnum_read_sign_loop:N	114, 119, 122
_zhnum_read_sign_loop:NN	280, 284, 287
_zhnum_read_zeros_loop:N	127, 131, 134
\l_zhnum_reset_bool	522, 525, 632, 659
\zhnum_reset_config:	16, 688, 720, 762

_zhnum_result:nn	117, 149, 150, 151	\c_zhnum_sun_tl	350
\c_zhnum_sat_tl	349	\l_zhnum_sun_tl	591
\l_zhnum_sat_tl	589	\c_zhnum_ten_tl	253, 604
\l_zhnum_scale_int	430, 432, 433, 449	\c_zhnum_thousand_tl	225, 608
\zhnum_scale_map:n	12, 213, 424, 430, 431	\c_zhnum_thu_tl	354
\zhnum_scale_map_hook:n	427, 434, 441	\l_zhnum_thu_tl	585
\zhnum_scale_map_loop:n	12, 429, 434	_zhnum_time:ww	403, 409
\zhnum_set_active:	691, 692	\l_zhnum_time_bool	343, 759, 760
\zhnum_set_alias:	14, 551, 600	\c_zhnum_tue_tl	352
\zhnum_set_alias:NN	599, 602, 604, 606, 608, 612	\l_zhnum_tue_tl	581
\zhnum_set_catcode:	16, 635, 682, 690	\zhnum_two_digits:n	10, 361, 372
\zhnum_set_cfg_name:Nn	16, 615, 683, 710	\zhnum_update_cfg:n	617, 621
\zhnum_set_digits_map:nn		_zhnum_update_cfg_prop:N	626, 633, 641, 644
.....	13, 500, 789, 791, 793, 796, 798, 799,	\c_zhnum_wed_tl	353
	800, 801, 802, 803, 804, 805, 806, 807, 808, 809, 810,	\l_zhnum_wed_tl	583
	811, 812, 813, 819, 821, 822, 830, 831, 833, 834, 835,	_zhnum_week_day:www	10, 328, 345, 346
	836, 837, 845, 846, 848, 854, 856, 857, 858, 864, 866, 867	\c_zhnum_weekday_tl	580, 582, 584, 586, 588, 590, 592
\zhnum_set_digits_map:nnn	13, 502, 785, 786, 815, 816,	\c_zhnum_year_tl	332, 338
	824, 825, 826, 827, 839, 840, 841, 842, 850, 851, 860, 861	\zhnum_Zeller:nnn	10, 348, 358
\zhnum_set_financial_map:nn	13, 507, 868, 869,	_zhnum_Zeller_aux:Nnnn	362, 363, 366
	870, 871, 877, 879, 880, 886, 888, 889, 890, 891, 892, 893	\zhnum_Zeller_aux:Nnnn	10
\zhnum_set_financial_map:nnn	13, 509, 873, 874, 882, 883	\zhnum_Zeller_Gregorian:nnn	10, 362, 377
\zhnum_set_scale:n	12, 439, 443	\zhnum_Zeller_Julian:nnn	11, 363, 390
\zhnum_set_week_day:	13, 521, 577	\c_zhnum_zero_tl	65, 71, 106, 156, 211, 224, 227, 235, 602
\zhnum_set_zero:	13, 520, 569	\zhnumber	1, 3, 3, 13
\l_zhnum_simp_bool	542, 754, 755	\zhnumberwithoptions	4, 5, 17, 20
\zhnum_split_number:fn	181	\zhnumExtendScaleMap	2, 12, 436
\zhnum_split_number:nn	7, 198, 199, 204	\zhnumsetup	2, 17, 769
\zhnum_split_number:NNfNnNw	216, 217	\zhnumwithoptions	78, 81
\zhnum_split_number:NNnNnNw	7, 201, 205, 220	\zhptime	2, 11, 402
_zhnum_split_number_aux:nnn	7, 182, 186	\zhtoday	2, 10, 336
_zhnum_split_number_aux:wn	188, 197	\zhweekday	2, 10, 344