

zhnumber 宏包

李清* 黄晨成†

2026/09/27 v3.1‡

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。

宏包最初由李清编写和维护, 并纳入 ctex-kit 项目管理。自 2022 年至 2026 年, 社区已与李清失联逾四年。考虑到这段时间陆续有相关缺陷反馈和功能请求, 与 CTAN 联络后, 社区决定接手维护该宏包。自 2026 年 6 月起, 由黄晨成实际维护。

第 2 节 使用方法

encoding	encoding = $\langle$ GBK Big5 UTF8 $\rangle$
----------	--

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、X_YL^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。

<code>\zhnumber</code>	*	<code>\zhnumber {<number>}</code>
------------------------	---	---

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

*sobenlee@gmail.com

†liamhuang0205@gmail.com

‡ctex-kit rev. f6b3a4f4.

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

`\zhrod` `\zhrod {<number>}`

New: 2026-08-06

将阿拉伯数字转换为算筹数字。算筹是中国古代的记数符号, 用纵横两式的筹码交替表示各位数字: 个位一式、十位另一式、百位又回到第一式, 依此交替。

`\zhrod` 只产出字符, 不切换字体、不调整字距, 因此它是可展开的, 可以写进 `.toc` 一类辅助文件, 也可以放进 PDF 书签。需要压紧字距时用 `\zhrodbox` (见下)。

下面的例子假定当前字体含算筹字形, 做法见下文 `font` 选项。

| = ||| ≡ ||||
 \ | = ○ ≡ ○. ≡ ○ ≡

```
1 \RodFont
2 \zhrod{12345}\\
3 \zhrod{-12030.405}
```

小数部分继续交替: 分位与个位异式, 厘位与个位同式。负数默认在最前排一个 \, 理由见下文 `minus` 选项。

`\zhrod` 只认数字、一个前导正负号和一个小数点。其余字符原样输出, 第二个小数点之后的部分会被丢掉 (`\zhrod{1.2.3}` 只排出 1.2 那部分)——它是数字排印命令, 不做输入校验, 请自己保证传进来的是数字。

参数里不能带分组。 `\zhrod` 要按每一位数字距个位有多远来选纵式还是横式, 而一个花括号分组在计数时算作一位, 于是各位的纵横会整体错开。传进去的必须是不带花括号的数字串: `\zhrod{1{23}4}` 会把花括号原样排出来, 而 `\zhrod{12\ver}` (其中 `\ver` 展开为 `{2026}`) 会得到纵横全部相反的一串筹码, 看上去仍像正常的算筹却是错的。若数字来自宏, 请让它展开为不带分组的数字串 (`\def\ver{2026}` 而不是 `\def\ver{{2026}}`)。

这一点与 `\zhdigits` 不同: 后者逐位取字、不需要知道位置, 所以分组对它无害。

分组里若还含非数字, 后果更严重: `\zhrod{1{ab}4}` 会报多条错误并以非零状态退出 (不只是排出多余字符)。实测 $\TeX$ 错误为四条 (两条 Missing number、一条 Missing = inserted for \ifnum、一条 You can't use \numexpr in horizontal mode), 此外还会有若干条缺字警告, 条数随字体与引擎而变, 因此这里不写死总数。同属上面那条限制, 处置方式一样: 不要给它带分组的参数。

`\zhrodbox` `\zhrodbox {<number>}`

New: 2026-08-06

与 `\zhrod` 输出相同的筹码, 但会切换到 `font` 指定的字体, 并在相邻筹码之间压紧字距。

它不可展开, 因此**不能**写进辅助文件或 PDF 书签——写进去会在下次编译读回时留下命令名而不是筹码。要用在标题一类会写进目录的位置, 请改用 `\zhrod`。

| = ||| ≡ ||||
 | = ||| ≡ ||||

```
1 \zhrodsetup{font=\RodFont}
2 \zhrodbox{12345}\\
3 {\RodFont\zhrod{12345}}
```

上例中两行内容相同, 上行压紧了字距。实测 12345 未压缩时宽 50.0pt, 压缩后 44.4pt——五位筹码之间有四处字距, 每处 -1.4pt。

`\zhrodsetup` `\zhrodsetup {<key-value list>}`

New: 2026-08-06

设置算筹相关选项。算筹的选项独立于 `\zhnumsetup`: 后者管的是「每个数字写成哪个汉字」(字形与编码), 算筹管的是「按位值选哪一式筹码」, 两者不通用。

与 `\zhnumsetup` 一样, 它设置的是局部变量, 因此受 $\TeX$ 分组约束: 写在分组或环境里只在其中有效, 写在导言区或分组外则此后一直有效。上面各例都靠这一点连续切换设置。

units units = `<vertical|horizontal>`

New: 2026-08-06 指定个位用哪一式, 其余各位由交替推出。vertical 是个位用纵式(即传统所说的「一纵十横」, 较常见), horizontal 是个位用横式。

| = ||| ≡ |||| ⊥ ⊥ ≡ |||
 — || ≡ ||| ≡ ⊥ ⊥ ≡ |||

```
1 \RodFont
2 \zhrodsetup{units=vertical}
3 \zhrod{123456789}\
4 \zhrodsetup{units=horizontal}
5 \zhrod{123456789}
```

zero zero = `<fill|omit>`

New: 2026-08-06 算筹本无零的筹码。默认(fill)在零的位置排一个○占位;omit 则跳过零, 靠纵横交替体现空位。

| = ○ ≡ ○
 | = ≡

```
1 \RodFont
2 \zhrodsetup{zero=fill}
3 \zhrod{12030}\
4 \zhrodsetup{zero=omit}
5 \zhrod{12030}
```

zero=omit 会产生歧义, 不要用在需要还原数值的场合。 交替只能区分相邻两位, 不能表示数量级: 省去零之后, 1、100、10000 的筹码写法完全相同(都是单独一个纵式的一)。穷举核对过, 1 到 99999 之间有 990 组数字因此无法区分。

史料中确有省零的写法, 靠上下文和算筹板上的位置定位数量级; 排印这类内容时可以用 omit, 但要自己保证读者能确定位数。

还有一个更极端的情形: 输入本身就是 0 时, omit 不产出任何字符(\zhrod{0} 得到空串, \zhrod{0.0} 只得到一个小数点)。这不是「读者需要自己定位数量级」, 而是整个数字都没有输出。

zerochar zerochar = `<token list>`

New: 2026-08-06 zero=fill 时占位用的字符, 默认是○。

dot dot = `<token list>`

New: 2026-08-06 小数点用的字符, 默认是半角句点。算筹在史料里不写小数点(小数位靠算筹板上的位置区分), 所以这里不沿用 \zhnumsetup 那套把小数点排成「点」字的做法; 若要排成别的形式, 用这个选项。

minus minus = `<slash|overlay>`

New: 2026-08-06 负数的记法。默认(slash)在筹码之前排一个独立的\。

Unicode 建议的记法是 overlay: 用组合字符 U+20E5 叠加在最右一位筹码上。

两种记法各自依赖的字符互不重合, 需要按字体选择。 slash 用的是 U+FF3C(全角反斜线), overlay 用的是 U+20E5。已知限制: 末位是 0 时, overlay 会叠到零的位置上, 而不是叠在一个筹码上——zero=fill 时叠在占位的○上, zero=omit 时叠在末位之前那个筹码上(实测 \zhrod{-10} 在 fill 下得到「筹码 + ○ + U+20E5」)。这是「叠在最右一个字符上」这条规则本身带来的: 末位没有筹码可叠。此时建议改用 slash。zero=omit 或 zerochar 为空时, 整串全是 0 的负数(如 -0、-0.0)没有任何字符可叠, overlay 不输出负号, 结果与相应的正数相同; 只有负号的 - 也不输出组合字符。

把 T_EX Live 里含全部算筹字形的字体逐一检查(共 19 款), 结果是两者恰好互补: lxgw-fonts 的 5 款只有 U+FF3C, juliamono 的 14 款只有 U+20E5, 没有一款同时具备。所以用 lxgw-fonts 时要用 slash(默认), 用 juliamono 时要改成 overlay; 用错那一个会得到缺字警告, 且负号占了宽度却什么都不显示。

font font = **

New: 2026-08-06 `\zhrodbox` 使用的字体切换命令, 默认为空(即用当前字体)。

`zhnumber` 本身不加载任何字体, 也不指定用哪一款——含算筹字形的字体随时间变化, 写死一个名字会过期。请自己声明一个字体族再交给这个选项:

```
\newfontfamily\RodFont{LXGWenKaiGBLite-Regular.ttf}
\zhrodsetup{font=\RodFont}
```

截至 2026 年 8 月, $\text{\TeX}$ Live 的 `lxgw-fonts` 里有五款字体覆盖算筹的全部码位 (U+1D360 至 U+1D371): `LXGWenKaiGBLite-Regular.ttf`, `LXGWNeoZhiSong.ttf`, `LXGWNeoXiHei.ttf` 及后两者的 Screen 变体。用它们的好处是筹码与正文汉字可以出自同一款字体。

注意同一字体包里的臻楷(`LXGWZhenKaiGB-Regular.ttf`)与朱雀仿宋(`LXGWZhuqueFangsong-Regular.ttf`)没有算筹字形。臻楷是独立的字形, 不是文楷的粗体版本, 但 `fontset=lxgw` 把它配作 `zhkai` 的 `BoldFont`; 因此在那套字体配置下给算筹加粗会丢字。

另外, `xeCJK` 不把算筹码位当作汉字 (`\XeTeXcharclass` 为 0, 而 $\bigcirc$ 与 一一 为 1), 所以在 `ctexart` 一类文档里靠 `\CJKfamily` 切换取不到筹码字形, 必须像上面那样显式声明字体族。这也意味着筹码与相邻汉字之间会插入 `\CJKecglue` 的间距——那是 `glue`, 因而是可断点, 所以 `\zhrodbox` 的字距虽然用 `\kern`, 整串筹码在 `xeCJK` 文档里仍可能断行 (`zero=fill` 时每个 $\bigcirc$ 两侧各有一处, 实测确实会断; `zero=omit` 时没有 $\bigcirc$, 也就没有这些断点)。要整串一定不断, 把它放进 `\mbox`。

kern kern = *<dimension expression>*

New: 2026-08-06 `\zhrodbox` 在相邻筹码之间插入的字距, 默认 `- .14em`。

这个默认值是照排版效果试出来的临时值, 没有按史料核定过; 若你有可靠的样张, 请自行调整。

引擎要求。算筹只能在 $\text{\XeTeX}$ 与 $\text{\LuaTeX}$ 下使用。 $\text{\pdfTeX}$ 与 $\text{\upTeX}$ 是 8-bit 引擎, 无法表示算筹所在的码位, 在这两个引擎下调用 `\zhrod` 或 `\zhrodbox` 会报错而不会退化成别的写法——算筹没有 8-bit 的替代记法, 静默退化只会让人以为自己排出了算筹。

\zhnum * `\zhnum {<counter>}`
`\pagenumbering {zhnum}`

Updated: 2022-07-10

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

二

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

\zhdig * `\zhdig {<counter>}`
`\pagenumbering {zhdig}`

Updated: 2022-07-10

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

二

```
1 \zhdig{section}
```

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

Updated: 2022-07-10

输出日期当天的星期。例如

星期日

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

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

Updated: 2026-07-09

以中文格式输出日期, 其中带 * 的命令还输出星期。例如

2026 年 7 月 9 日

2026 年 7 月 9 日星期四

2026 年, 2026 年 7 月

```
1 \zhdate {2026/7/9}\
2 \zhdate*{2026/7/9}\
3 \zhdate {2026}, \zhdate {2026/7}
```

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

Updated: 2022-07-10

2026 年 9 月 30 日

```
1 \zhtoday
```

`\zhptime` * `\zhptime {<hh:mm>}`

Updated: 2022-07-10

以中文格式输出时间。例如

23 时 56 分

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

`\zhcurrtime` * 输出当前的时间。例如

Updated: 2022-07-10

2 时 22 分

```
1 \zhcurrtime
```

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

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

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

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

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

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
丁卯戊辰亥亥

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

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

Updated: 2022-07-10

输出公元纪年 `<year>` 对应的干支纪年。公元前的年份用负数表示。例如

戊戌乙卯
甲子丙午

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

`\zhnumExtendScaleMap` `\zhnumExtendScaleMap [ <character> ] {<character_1>, <character_2>, ..., <character_n>}`

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

`\zhnumsetup` `\zhnumsetup {<key_1>=<val_1>, <key_2>=<val_2>, ...}`

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 `<key>` 如下介绍。以**粗体**表示选项的默认值。

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

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

二〇二六年九月三十日二时二十二分

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

`arabicsep` `arabicsep = {<sep>}`

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>`

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○, 可以使用这个选项。

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

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

甲乙壬癸壬辛
子亥戌亥戌酉
甲子乙亥辛酉
癸亥壬戌乙卯

```
1 \zhnumsetup{ganzhi-cyclic}
2 \zhtiangan{11} \zhtiangan{12} \zhtiangan{209}
3 \zhtiangan{-1} \zhtiangan{-2} \zhtiangan{-683} \
4 \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○, dot 设置小数的点, and 和 parts 分别设置分数的“又”和“分之”, En 设置 10^n , Fn 设置数字 n 的大写, Tn 设置数字 n 的天干, Dn 设置数字 n 的地支, 而 GZn 设置数字 n 的干支。其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>`

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。

<code>\zhnumber</code>	<code>\zhnumber</code>	<code>[[<options>]]</code>	<code>{<number>}</code>
<code>\zhdigits</code>	<code>\zhdigits</code>	<code>*</code>	<code>[[<options>]]</code>
<code>\zhnum</code>	<code>\zhnum</code>	<code>[[<options>]]</code>	<code>{<counter>}</code>
<code>\zhdig</code>	<code>\zhdig</code>	<code>[[<options>]]</code>	<code>{<counter>}</code>

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。

不可展开并不妨碍把它们用在会写入辅助文件的位置 (例如 `\thesection` 进而进入 `.toc`)。自 v3.1 起, `<counter>` 在写入的那一刻就被取成数值固定下来, 写进辅助文件的形如 `\zhnumberwithoptions{style=Financial}{7}`; 样式则留待读回时套用。因此下面这种按层级混搭样式的写法是可行的:

```
\renewcommand\thesection{\zhnum[style=Normal]{<section>}}
\renewcommand\thesubsection{\thesection.\zhnum[style=Financial]{<subsection>}}
```

v3.0 及以前写进辅助文件的是计数器名, 读回时计数器早已归零或停在别的值上, 目录里的编号会与正文不一致。

仍有一处限制: 辅助文件能容纳不可展开的命令, 而 `hyperref` 的 `\pdfstringdef` (PDF 书签、文档信息一类纯文本字段) 不能——它要求内容完全可展开。所以把带选项的形式放进 `\thesection` 后, 若同时用了 `bookmarksnumbered`, 书签里的编号会退化成 `style=Normal1` 这样的字面文本, 并伴随一条 `Token not allowed in a PDF string` 警告。这不是 v3.1 引入的: 已发布的 v3.0 在同样写法下退化成 `style=Normalsection`, 警告条数相同。要让书签也正确, 请对相应层级改用不带选项的 `\zhnum`/`\zhdig` (先用 `\zhnumsetup` 设定样式), 或在 `\pdfstringdefPreHook` 里自行处理。「不可展开」「不能写进辅助文件」「不能进 `\pdfstringdef`」是三件不同的事, 只有中间那件在 v3.1 里不再成立。

第 3 节 `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 } { 2025/10/09 } { }
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。

带选项的形式仍然不可完全展开(它需要一个分组来局部改样式),但计数器名不应该被原样写进辅助文件。原先 `\zhnum` 把 `<counter>` 直接交给 `\zhnumwithoptions`, 写进 `.toc` 的是 `\zhnumwithoptions{...}{section}`; 该文件在下一次编译的 `\tableofcontents` 处被读回, 此时计数器已经归零或停在别的值上, 于是目录里的编号与正文不一致(#1008 实测: 正文「一／一. 壹」而目录「零／零. 零」)。

修正办法是在 `\zhnum` 这一层先用 `f` 展开把计数器取成数值, 再交给处理数值的 `\zhnumberwithoptions`: 写进辅助文件的成为 `\zhnumberwithoptions{style=Financial}{7}` (第二个参数是写入那一刻的计数器值, 与前面「用法」一节举的例子是同一个), 数值已经固定, 样式则留待读回时套用。不改成完全可展开是有意的——样式靠 `\tl_set_eq:NN` 一类赋值实现 (见 `\zhnum_reset_style:`), 赋值无法在 `\edef` 的展开中生效, 强行这样做只会把「不可展开」换成「静默用错样式」。

```

99 \NewExpandableDocumentCommand \zhnum { +o +m }
100 {
101   \tl_if_novalue:nTF {#1}
102     { \zhnum_counter:n {#2} }
103     { \__zhnum_counter_with_options:nn {#1} {#2} }
104 }

```

`\__zhnum_counter_with_options:nn` 先确认计数器存在 (不存在时给出与无选项形式一致的报错), 再把它展开成数值。 `\int_use:c` 只在计数器存在时才可用, 所以存在性判断必须在展开之前。

```

105 \cs_new:Npn \__zhnum_counter_with_options:nn #1#2
106 {
107   \int_if_exist:cTF { c@#2 }
108   {
109     \exp_args:Nnf \zhnumberwithoptions {#1}
110     { \exp_args:Nc \int_use:N { c@#2 } }
111   }
112   { \__zhnum_counter_error:n {#2} }
113 }

```

`\zhnumwithoptions` 为兼容而保留: v3.0 及以前写进辅助文件的 .toc 里可能还留着这个名字, 删掉会让旧文件读回时报 `undefined control sequence`。它现在只是转发。

```

114 \NewDocumentCommand \zhnumwithoptions { +m +m }
115 {
116   \group_begin:
117   \zhnum_set:n {#1}
118   \zhnum_counter:n {#2}
119   \group_end:
120 }

```

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

```

121 \cs_new:Npn \zhnum_counter:n #1
122 {
123   \int_if_exist:cTF { c@#1 }
124   { \exp_args:Nc \zhnum_int:n { c@#1 } }
125   { \__zhnum_counter_error:n {#1} }
126 }
127 \cs_new:Npn \__zhnum_counter_error:n #1
128 { \msg_expandable_error:nnn { zhnumber } { not-counter } {#1} }
129 \msg_new:nnn { zhnumber } { not-counter }
130 { `#1'~is~not~a~LaTeX~counter. }
131 \zhnum_expand_wrap:wn
132 \cs_new:Npn \zhnum_int:n #1
133 {
134   \int_compare:nNnTF {#1} > \c_zero_int
135   { \zhnum_parse_number:f { \int_eval:n {#1} } }
136   {
137     \int_compare:nNnTF {#1} < \c_zero_int
138     {
139       \zhnum_output:n { minus }
140       \zhnum_parse_number:f { \int_eval:n { - #1 } }
141     }
142     { \zhnum_output:n { 0 } }
143   }
144 }

```

\@zhnum 用于支持\pagenumbering{zhnum}。

```
145 \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 l3bigit 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。

```
146 \cs_new:Npn \zhnum_integer:n #1
147 {
148   \exp_after:wN \__zhnum_read_integer:www
149   \int_value:w
150   \exp_after:wN \__zhnum_read_sign_loop:N
151   \exp:w \exp_end_continue_f:w \use:n
152   #1 \exp_stop_f: \q_recursion_tail \q_recursion_stop
153   \__zhnum_result:nn { \c_zero_int } { } ;
154 }
155 \cs_new:Npn \__zhnum_read_sign_loop:N #1
156 {
157   \if:w + \if:w - \exp_not:N #1 + \fi: \exp_not:N #1
158   \exp_after:wN \__zhnum_read_sign_loop:N
159   \exp:w \exp_end_continue_f:w \exp_after:wN \use:n
160   \else:
161     1 \exp_after:wN ;
162     \exp:w \exp_end_continue_f:w
163     \exp_after:wN \__zhnum_read_zeros_loop:N
164     \exp_after:wN #1
165   \fi:
166 }
167 \cs_new:Npn \__zhnum_read_zeros_loop:N #1
168 {
169   \if:w 0 \exp_not:N #1
170   \exp_after:wN \__zhnum_read_zeros_loop:N
171   \exp:w \exp_end_continue_f:w \exp_after:wN \use:n
172   \else:
173     \exp_after:wN \__zhnum_read_abs_loop:Nw
174     \exp_after:wN #1
175   \fi:
176 }
```

__zhnum_read_abs_loop:Nw 当数字很大时, l3bigit 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。

```
177 \cs_new:Npn \__zhnum_read_abs_loop:Nw #1#2 \q_recursion_stop
178 {
179   \zhnum_if_digit:NTF #1
180   { \__zhnum_output:nnwnn { + 1 } #1 }
181   { \quark_if_recursion_tail_stop_do:Nn #1 { \__zhnum_loop_end:wnn } }
182   \exp_after:wN \__zhnum_read_abs_loop:Nw
183   \exp:w \exp_end_continue_f:w \use:n #2 \q_recursion_stop
184 }
185 \cs_new:Npn \__zhnum_output:nnwnn #1#2#3 \__zhnum_result:nn #4#5
186 { #3 \__zhnum_result:nn { #4#1 } { #5#2 } }
187 \cs_new:Npn \__zhnum_loop_end:wnn #1 \__zhnum_result:nn #2#3
188 { \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。

```
189 \cs_new:Npn \__zhnum_read_integer:www #1 ; #2 ; #3 ;
190 {
191   \int_compare:nNnTF {#2} = \c_zero_int
192   { \zhnum_output:n { 0 } }
193   {
194     \int_compare:nNnF {#1} = \c_one_int
195     { \zhnum_output:n { minus } }
196     \zhnum_parse_number:nn {#2} {#3}
```

```

197     }
198 }

```

\zhnum_if_digit:NTF 判断 #1 是否为数字位。

```

199 \cs_new:Npn \zhnum_if_digit:NTF #1
200 {
201     \if_int_compare:w 9 < 1 \exp_not:N #1 \exp_stop_f:
202     \exp_after:wN \use_i:nn
203     \else:
204     \exp_after:wN \use_ii:nn
205     \fi:
206 }

\zhnum_parse_number:n 207 \cs_new:Npn \zhnum_parse_number:n #1
\zhnum_parse_number:nn 208 { \exp_args:Nf \zhnum_parse_number:nn { \tl_count:n {#1} } {#1} }
209 \cs_new:Npn \zhnum_parse_number:nn #1
210 { \exp_args:Nf \__zhnum_parse_number:nnn { \int_mod:nn {#1} { 4 } } {#1} }
211 \cs_new:Npn \__zhnum_parse_number:nnn #1#2
212 {
213     \int_compare:nNnTF {#2} < 2
214     { \zhnum_output:n }
215     {
216         \int_compare:nNnTF {#1} = \c_zero_int
217         { \zhnum_split_number:fn { \int_eval:n { #2 / 4 - 1 } } }
218         { \__zhnum_split_number_aux:nnn {#1} {#2} }
219     }
220 }
221 \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。

```

222 \cs_new:Npn \__zhnum_split_number_aux:nnn #1#2
223 {
224     \exp_after:wN \__zhnum_split_number_aux:wnn
225     \int_value:w \int_div_truncate:nn {#2} { 4 }
226     \if_case:w #1 \exp_stop_f:
227     \or: \exp_after:wN \use:n
228     \or: \exp_after:wN \use_i_ii:nnn
229     \or: \exp_after:wN \use_i:nnn
230     \fi:
231     { \exp_stop_f: ; 0 } 0 0 ;
232 }
233 \cs_new:Npn \__zhnum_split_number_aux:wnn #1 ; #2 ; #3
234 { \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。

```

235 \cs_new:Npn \zhnum_split_number:nn #1#2
236 {
237     \zhnum_split_number:NNnNNNw \c_true_bool \c_true_bool {#1}
238     #2 \q_recursion_tail \q_nil \q_nil \q_nil \q_recursion_stop
239 }
240 \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。

```

241 \cs_new:Npn \zhnum_split_number:NNnNNNw #1#2#3#4#5#6#7
242 {
243     \quark_if_recursion_tail_stop:N #4
244     \int_compare:nNnTF { #4#5#6#7 } = \c_zero_int
245     { \use_i:nn }
246     {
247         \bool_if:NF #1 { \zhnum_output:n { 0 } }
248         \zhnum_process_number:NNNNN #4#5#6#7#1#2
249         \zhnum_scale_map:n {#3}
250         \int_compare:nNnTF {#7} = \c_zero_int
251     }
252     { \zhnum_split_number:NNfNNNNw \c_false_bool \c_true_bool }

```

```

253     { \zhnum_split_number:NNfNNNNw \c_true_bool \c_false_bool }
254     { \int_eval:n { #3 - 1 } }
255   }
256 \cs_generate_variant:Nn \zhnum_split_number:NNnnNNNw { NNf }

```

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

```

257 \cs_new:Npn \zhnum_process_number:NNNNNN #1#2#3#4#5#6
258 {
259   \int_compare:nNnTF {#1} = \c_zero_int
260   {
261     \bool_if:NF #6
262     { \zhnum_output:n { 0 } }
263   }
264   {
265     \zhnum_output:n {#1}
266     \zhnum_output:n { 1000 }
267   }
268   \int_compare:nNnTF {#2} = \c_zero_int
269   {
270     \int_compare:nNnF { #1 * (#3#4) } = \c_zero_int
271     { \zhnum_output:n { 0 } }
272   }
273   {
274     \int_compare:nNnTF {#2} = 2
275     { \zhnum_output:n { 200 } }
276     {
277       \zhnum_output:n {#2}
278       \zhnum_output:n { 100 }
279     }
280   }
281   \int_compare:nNnTF {#3} = \c_zero_int
282   {
283     \int_compare:nNnF { #2 * #4 } = \c_zero_int
284     { \zhnum_output:n { 0 } }
285   }
286   {
287     \bool_lazy_all:nTF
288     {
289       { \int_compare_p:nNn {#3} = \c_one_int }
290       { \int_compare_p:nNn {#1#2} = \c_zero_int }
291       {#6}
292       {#5}
293     }
294     { \zhnum_output:n { 10 } }
295     {
296       \int_compare:nTF { 1 < #3 < 5 }
297       { \zhnum_output:n { #3 0 } }
298       {
299         \zhnum_output:n {#3}
300         \zhnum_output:n { 10 }
301       }
302     }
303   }
304   \int_compare:nNnF {#4} = \c_zero_int
305   { \zhnum_output:n {#4} }
306 }

```

\zhdig 用于将 L^AT_EX 计数器按中文数字串输出。

```

307 \NewExpandableDocumentCommand \zhdig { +o +m }
308 {
309   \tl_if_novalue:nTF {#1}
310   { \zhnum_digits_counter:n {#2} }
311   { \__zhnum_digits_counter_with_options:nn {#1} {#2} }
312 }

```

__zhnum_digits_counter_with_options:nn 与 \zhnum 的对应处理一致, 先固定数值再套样式, 理由见那里。

星号参数必须写成 `{\BooleanFalse}` 而不是 `\c_false_bool`: 这一整串会被写进辅助文件, 再在下次编译时重新切分成记号(tokenise), 而那时 `_` 不是 letter(catcode 8), `\c_false_bool` 的名字会在写出时就断成 `\c _false_bool`, 读回即出错。具体报什么错取决于写法, 实测两种: 不带花括号时(初版的`\exp_args:Nnne ... \c_false_bool`)断开的 `_` 被当成键名, 报 The key ``zhnum/options/_'` is unknown 之类的一串错误; 带花括号时报 Missing \$ inserted。两者都是「名字在写出那一刻断开」的后果。`\zhdigits` 的无计数器版本本来就用 `{\BooleanFalse}` (它是 `expl3` 为此提供的、名字里没有 `_` 的记号), 这里对齐它。

这一点是盲审查出来的: 初版用 `\c_false_bool` 时第一遍编译正常, 第二遍读回 .toc 才出错——等于把 #1008 那类「跨编译才暴露」的失败换到了另一个命令上。没有为它单独记 `\changes`: 这条路径本身是 v3.1 新增的(已发布的 v3.0 里 `\zhdig` 的带选项形式根本不可用, 见下面 `\zhdigwithoptions` 的笔误), 所以它不构成对已发布行为的修正, 并入核心修复那一条即可。

```

313 \cs_new:Npn \__zhnum_digits_counter_with_options:nn #1#2
314 {
315   \int_if_exist:cTF { c@#2 }
316   {
317     \exp_args:Nnnf \zhdigitswithoptions { \BooleanFalse }
318     {#1} { \exp_args:Nc \int_use:N { c@#2 } }
319   }
320   { \__zhnum_counter_error:n {#2} }
321 }

```

`\zhdigwithoptions` 为兼容旧辅助文件而保留, 只是转发。

原先这里写的是 `\zhnum_digits_counter:n #1 {#2}`, 多传了一个 #1 (选项本身)。`\zhnum_digits_counter:n` 只取一个参数, 于是选项被当成计数器名、真正的计数器名溢出到后面。实测 `\zhdig[style=Financial]{section}` 报 Use of `????` doesn't match its definition, 并把 `tle=Financialsection` 印到页面上。

```

322 \NewDocumentCommand \zhdigwithoptions { +m +m }
323 {
324   \group_begin:
325   \zhnum_set:n {#1}
326   \zhnum_digits_counter:n {#2}
327   \group_end:
328 }
329 \cs_new:Npn \zhnum_digits_counter:n #1
330 {
331   \int_if_exist:cTF { c@#1 }
332   { \zhnum_digits_null:v { c@#1 } }
333   { \__zhnum_counter_error:n {#1} }
334 }

```

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

```

335 \cs_new:Npn \@zhdig #1
336 { \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
337 \NewExpandableDocumentCommand \zhdigits { +s +o +m }
338 {
339   \tl_if_novalue:nTF {#2}
340   { \zhnum_digits:Ne #1 }
341   { \zhdigitswithoptions {#1} {#2} }
342   {#3}
343 }
344 \NewDocumentCommand \zhdigitswithoptions { +m +m +m }
345 {
346   \group_begin:
347   \zhnum_set:n {#2}
348   \zhnum_digits:Ne #1 {#3}
349   \group_end:
350 }

```

`\zhnum_digits_zero:n` 快捷方式。

```
\zhnum_digits_null:n
351 \cs_new:Npn \zhnum_digits_zero:n
352   { \zhnum_digits:Nn \c_true_bool }
353 \cs_new:Npn \zhnum_digits_null:n
354   { \zhnum_digits:Nn \c_false_bool }
355 \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。

```
356 \zhnum_expand_wrap:wn
357 \cs_new:Npn \zhnum_digits:Nn #1#2
358   {
359     \exp_after:wN \__zhnum_read_digits:w
360     \int_value:w
361     \exp_after:wN \__zhnum_read_sign_loop:NN \exp_after:wN #1
362     \exp:w \exp_end_continue_f:w \use:n
363     #2 \exp_stop_f: \q_recursion_tail \q_recursion_stop
364   }
365 \cs_new:Npn \__zhnum_read_sign_loop:NN #1#2
366   {
367     \if:w + \if:w - \exp_not:N #2 + \fi: \exp_not:N #2
368     \exp_after:wN \__zhnum_read_sign_loop:NN \exp_after:wN #1
369     \exp:w \exp_end_continue_f:w \exp_after:wN \use:n
370     \else:
371       1 \exp_after:wN ;
372       \exp_after:wN \__zhnum_read_digits_loop:NN
373       \exp_after:wN #1
374       \exp_after:wN #2
375     \fi:
376   }
377 \cs_new:Npn \__zhnum_read_digits_loop:NN #1#2
378   {
379     \zhnum_if_digit:NTF #2
380     { \__zhnum_output_digits:NN #1#2 }
381     {
382       \quark_if_recursion_tail_stop:N #2
383       \token_if_eq_charcode:NNT #2 .
384       { \zhnum_output:n { dot } }
385     }
386     \exp_after:wN \__zhnum_read_digits_loop:NN \exp_after:wN #1
387     \exp:w \exp_end_continue_f:w \use:n
388   }
389 \cs_new:Npn \__zhnum_read_digits:w #1 ;
390   {
391     \int_compare:nNnF {#1} = \c_one_int
392     { \zhnum_output:n { minus } }
393   }
394 \cs_new:Npn \__zhnum_output_digits:NN #1#2
395   {
396     \zhnum_output:n
397     {
398       \if_int_compare:w #2 = \c_zero_int
399       \bool_if:NTF #1 { zero } { null }
400       \else:
401         #2
402       \fi:
403     }
404   }
405 \cs_generate_variant:Nn \zhnum_digits:Nn { Ne }
```

`\c__zhnum_rod_engine_bool` 算筹码位在pdfTeX 与upTeX 下无法表示:这两个引擎是 8-bit 的,实测 `\char_generate:nn` 直接报 LaTeX Error: Charcode requested out of engine range. 这是引擎能力缺失,不是缺字体,因此没有退化写法可用。

不能复用 `\c__zhnum_unicode_engine_bool`: 那个布尔量把 upTeX 也算作真(它判的是「是否不必把汉字首字节设为活动字符」), 而 upTeX 恰好不能表示算筹。两者是不同粒度的引擎分类, 这里需要单独一个只含 XTeX 与 LuaTeX 的判据。

```
406 \bool_const:Nn \c__zhnum_rod_engine_bool
407 {
408   \bool_lazy_or_p:nn
409     { \sys_if_engine_xetex_p: }
410     { \sys_if_engine luatex_p: }
411 }
```

算筹的七个配置变量, 对应`\zhrodsetup`的七个键。

```
\l__zhnum_rod_units_vertical_bool
\l__zhnum_rod_zero_omit_bool
\l__zhnum_rod_minus_overlay_bool
\l__zhnum_rod_font_tl
\l__zhnum_rod_kern_tl
\l__zhnum_rod_zero_tl
\l__zhnum_rod_dot_tl
412 \bool_new:N \l__zhnum_rod_units_vertical_bool
413 \bool_set_true:N \l__zhnum_rod_units_vertical_bool
414 \bool_new:N \l__zhnum_rod_zero_omit_bool
415 \bool_new:N \l__zhnum_rod_minus_overlay_bool
416 \tl_new:N \l__zhnum_rod_font_tl
417 \tl_new:N \l__zhnum_rod_kern_tl
418 \tl_set:Nn \l__zhnum_rod_kern_tl { -.14 em }
```

零、小数点与负号这三个字符不走`\zhnumsetup`的样式与编码体系: 算筹只在 Unicode 引擎下可用, 没有 GBK / Big5 分支要照顾, 而它的零是○、小数点在史料里也不写汉字「点」。故直接用独立变量, 不并入`\zhnum_set_digits_map:nnn`。

```
419 \tl_new:N \l__zhnum_rod_zero_tl
420 \tl_set:Nn \l__zhnum_rod_zero_tl { ○ }
421 \tl_new:N \l__zhnum_rod_dot_tl
422 \tl_set:Nn \l__zhnum_rod_dot_tl { . }
```

`\__zhnum_rod_vertical:n` 这两个宏各从一个码区取出一个字符。注意区名与字形朝向相反: Unicode 的 COUNTING ROD UNIT DIGIT 区 (U+1D360 起) 是横画, TENS DIGIT 区 (U+1D369 起) 是纵画。实测 LXGW WenKaiGB Lite-Regular.ttf 下 U+1D360 的 wd 为 10pt, ht 为 4.05pt(扁), U+1D369 的 ht 为 8.03pt(高), 换 NotoSans Symbols2-Regular.ttf 复核一致, 所以这是 Unicode 的编排而非某款字体的取舍。故纵式取 TENS 区、横式取 UNIT 区, 按区名直译会得到相反的结果。

```
423 \cs_new:Npn \__zhnum_rod_vertical:n #1
424 { \char_generate:nn { \int_eval:n { "1D368 + #1 } } { 12 } }
425 \cs_new:Npn \__zhnum_rod_horizontal:n #1
426 { \char_generate:nn { \int_eval:n { "1D35F + #1 } } { 12 } }
```

`\__zhnum_rod_glyph:nn` 取一位数字 #1 与它的位值 #2, 产出对应字符。位值奇偶决定朝向: 偶位值(个位、百位、万位……)用`\zhrodsetup`的 units 所指定的那一式, 奇位值用另一式。

零单独处理: 默认排出`\l__zhnum_rod_zero_tl`(即○), zero=omit 时跳过。跳过会产生歧义, 理由见用户文档。

```
427 \cs_new:Npn \__zhnum_rod_glyph:nn #1#2
428 {
429   \zhnum_if_digit:NTF #1
430     { \__zhnum_rod_digit:nn {#1} {#2} }
431     { \exp_not:n {#1} }
432 }
```

非数字的输入原样输出, 不塞进`\int_eval:n`的算术里。否则`\zhrod{1a}`会把 a 当成数字参与`"1D368 + a`, 把`\__int_eval:w`一类内部记号漏到页面上(实测在筹码之后多排出 a=0 与一个○; 这里不把筹码字形写进`\verb`, 等宽字体没有算筹字形, 会多出缺字警告)。注意这与`\zhdigits`的策略相反而不是「取齐」: `\zhdigits`把不认识的字符丢掉(`\zhdigits{1a4}`得到「一四」), `\zhrod`把它原样留下(得到筹码、a、筹码)。两者都不报错, 但保留原字符更便于发现输入有问题。

```
433 \cs_new:Npn \__zhnum_rod_digit:nn #1#2
434 {
435   \int_compare:nNnTF {#1} = { 0 }
```

```

436 {
437   \bool_if:NF \l__zhnum_rod_zero_omit_bool
438   { \zhnum_exp_not:o \l__zhnum_rod_zero_tl }
439 }
440 {
441   \__zhnum_rod_kern:
442   \int_compare:nNnTF { \int_mod:nn {#2} { 2 } } = { 0 }
443   {
444     \bool_if:NTF \l__zhnum_rod_units_vertical_bool
445     { \__zhnum_rod_vertical:n {#1} }
446     { \__zhnum_rod_horizontal:n {#1} }
447   }
448   {
449     \bool_if:NTF \l__zhnum_rod_units_vertical_bool
450     { \__zhnum_rod_horizontal:n {#1} }
451     { \__zhnum_rod_vertical:n {#1} }
452   }
453 }
454 }

```

```

\__zhnum_rod_int:n
\__zhnum_rod_int_loop:nn
\__zhnum_rod_int_head:wn

```

整数部分逐位处理:位值从(位数 - 1)递减到0,交给__zhnum_rod_glyph:nn。

现有的两条读法都不能复用。zhnum_digits:Nn 那条逐位循环 (__zhnum_read_digits_loop:NN)全程不携带「当前是第几位」:它的布尔参数只是最初传入的 zero/null 开关,小数点靠字符比较识别、负号在循环前一次性剥离。zhnum_integer:n 那条虽然统计了长度,但那是为了按四位一段做万/亿制分组,不是位值奇偶。所以算筹需要单独写一条带位值的循环。

```

455 \cs_new:Npn \__zhnum_rod_int:n #1
456 { \__zhnum_rod_int_loop:nn {#1} { \tl_count:n {#1} - 1 } }

```

位值用 \tl_count:n 数,因此参数里的花括号分组会算作一位,让各位的纵横整体错开(分组里含非数字时更会直接报错并以非零状态退出)——zhrod{12\ver}(\ver 展开为 {2026})与zhrod{122026} 排出的朝向完全相反,且不报错。这是已知限制,手册里写明了,因为修它要把这两条循环改成 zhnum_digits:Nn 那种 \exp:w \exp_end_continue_f:w \use:n 的单记号取参形式(那种形式天然展开一层分组),同时位值计数也要改成先展平再数,改动范围比这个功能本身还大。zhdigits 不受影响:它逐位取字、不需要知道位置。

```

457 \cs_new:Npn \__zhnum_rod_int_loop:nn #1#2
458 { \tl_if_blank:nF {#1} { \__zhnum_rod_int_head:wn #1 \q_stop {#2} } }
459 \cs_new:Npn \__zhnum_rod_int_head:wn #1#2 \q_stop #3
460 {
461   \__zhnum_rod_glyph:nn {#1} { \int_eval:n {#3} }
462   \__zhnum_rod_int_loop:nn {#2} { \int_eval:n { #3 - 1 } }
463 }

```

```

\__zhnum_rod_frac:n
\__zhnum_rod_frac_loop:nn
\__zhnum_rod_frac_head:wn

```

小数部分:位值从 -1 继续递减,交替因此跨小数点自然延续(分位与个位异式、厘位与个位同式)。

```

464 \cs_new:Npn \__zhnum_rod_frac:n #1
465 { \__zhnum_rod_frac_loop:nn {#1} { -1 } }
466 \cs_new:Npn \__zhnum_rod_frac_loop:nn #1#2
467 { \tl_if_blank:nF {#1} { \__zhnum_rod_frac_head:wn #1 \q_stop {#2} } }
468 \cs_new:Npn \__zhnum_rod_frac_head:wn #1#2 \q_stop #3
469 {
470   \__zhnum_rod_glyph:nn {#1} { \int_eval:n {#3} }
471   \__zhnum_rod_frac_loop:nn {#2} { \int_eval:n { #3 - 1 } }
472 }

```

```

\__zhnum_rod_number:n
\__zhnum_rod_split:www
\__zhnum_rod_sign:w
\__zhnum_rod_overlay_if_base:n
\__zhnum_rod_base_mark:n

```

拆负号与小数点,再分派给上面两条循环。负号排在哪里取决于 minus 的取值,见下文。

```

473 \cs_new:Npn \__zhnum_rod_slash:
474 { \char_generate:nn { "FF3C } { 12 } }
475 \cs_new:Npn \__zhnum_rod_overlay:
476 { \char_generate:nn { "20E5 } { 12 } }

```

两种负号的位置不同,不能共用一个输出点:

- slash 是独立字符,排在筹码之前;
- overlay 是组合字符(U+20E5),必须紧跟在最后一位筹码之后,才会叠加到那一位上。
规范说的正是「叠加在最右一位上」。

初版两者都排在最前,于是 overlay 叠加到了算筹前面那个字符上: $X\text{zhrod}\{-1234\}Y$ 会把斜线叠到 X 上,而单独放进盒子时它不叠任何东西,负数与正数的盒子宽度完全相同(实测 JuliaMono 下都是 24pt),等于负号没排出来。这个缺陷由本地盲审发现。

overlay 还要求确实有一个基字符可叠。以下输入不排出任何可叠的字符:zero=omit 时整串全是 0 的数(-0、-0.0)、zerochar 置空时同样的输入,以及只有负号的-。若照样追加 U+20E5,它会叠到小数点或算筹前面的正文上。`\__zhnum_rod_overlay_if_base:n` 逐个检查输入的记号,只要有一个能排出可叠的字符就追加组合字符,否则不追加:0 只有在 zero=fill 且 zerochar 非空时才排出○,小数点不算,其余记号(非零数字排出筹码,其他字符原样输出)都可以作基字符。此时数值就是零,负零按零处理,输出与 0、0.0 相同。slash 的 \ 是独立字符,不会叠到别的字符上,保持原样。这个缺陷由 PR 审查发现。

两种记法依赖的字符在现有字体里恰好互补:逐一检查 TeX Live 里含全部算筹字形的 19 款字体,lgw-fonts 的 5 款只有 U+FF3C、juliamono 的 14 款只有 U+20E5,没有一款同时具备。因此这两个选项都必须保留,谁也不能当作另一个的替代。

```

477 \cs_new:Npn \__zhnum_rod_number:n #1
478 { \tl_if_blank:nF {#1} { \__zhnum_rod_sign:w #1 \q_stop } }
479 \cs_generate_variant:Nn \__zhnum_rod_number:n { e }
480 \cs_new:Npn \__zhnum_rod_sign:w #1#2 \q_stop
481 {
482   \str_if_eq:nnTF {#1} { - }
483   {
484     \bool_if:NF \l__zhnum_rod_minus_overlay_bool { \__zhnum_rod_slash: }
485     \__zhnum_rod_split:www #2 . \q_nil . \q_stop
486     \bool_if:NT \l__zhnum_rod_minus_overlay_bool
487     { \__zhnum_rod_overlay_if_base:n {#2} }
488   }
489   {
490     \str_if_eq:nnTF {#1} { + }
491     { \__zhnum_rod_split:www #2 . \q_nil . \q_stop }
492     { \__zhnum_rod_split:www #1#2 . \q_nil . \q_stop }
493   }
494 }
495 \cs_new:Npn \__zhnum_rod_overlay_if_base:n #1
496 {
497   \tl_if_blank:eF { \tl_map_function:nN {#1} \__zhnum_rod_base_mark:n }
498   { \__zhnum_rod_overlay: }
499 }
500 \cs_new:Npn \__zhnum_rod_base_mark:n #1
501 {
502   \str_if_eq:nnTF {#1} { 0 }
503   {
504     \bool_lazy_or:nnF
505     { \l__zhnum_rod_zero_omit_bool }
506     { \tl_if_blank_p:V \l__zhnum_rod_zero_tl }
507     { * }
508   }
509   { \str_if_eq:nnF {#1} { . } { * } }
510 }
511 \cs_new:Npn \__zhnum_rod_split:www #1 . #2 . #3 \q_stop
512 {
513   \__zhnum_rod_int:n {#1}
514   \quark_if_nil:nF {#2}
515   {
516     \zhnum_exp_not:o \l__zhnum_rod_dot_tl
517     \__zhnum_rod_frac:n {#2}
518   }
519 }

```

`\zhrod` 可展开的算筹输出,与 `\zhdigits` 同级。它只产出字符,不切字体、不加字距,因此可以进 .toc 与 PDF 书签:实测这一形式在书签里得到正确的代理对、零条 Token not allowed in a PDF string 警告,而下面 `\zhrodbox` 那种切字体加字距的形式在同一处会退化并报警告。

pdfTeX/UpTeX 下定义成直接报错的版本,不做静默退化:算筹没有 8-bit 的退化写法,退化只会让使用者以为自己排出了算筹。

```

520 \msg_new:nnn { zhnumber } { rod-engine-unsupported }
521 {
522   \token_to_str:N \zhrod \c_space_tl requires ~ XeTeX ~ or ~ LuaTeX. \\
523   Counting-rod ~ code ~ points ~ (U+1D360--U+1D371) ~ cannot ~ be ~
524   represented ~ by ~ 8-bit ~ engines.
525 }
526 \bool_if:NTF \c__zhnum_rod_engine_bool
527 {
528   \zhnum_expand_wrap:wn
529   \cs_new:Npn \zhrod #1
530     { \__zhnum_rod_number:e {#1} }
531 }
532 {
533   \cs_new_protected:Npn \zhrod #1
534     { \msg_error:nn { zhnumber } { rod-engine-unsupported } }
535 }

```

`\zhrodbox` 排版层:切字体并在相邻字符之间压紧字距。它不可展开,不能写进辅助文件。

字距默认为 `-.14em`,这是个临时视觉参数,尚未按史料核对,所以做成可调键而不是写死。

```

536 \bool_if:NTF \c__zhnum_rod_engine_bool
537 {
538   \cs_new_protected:Npn \zhrodbox #1
539     {
540       \group_begin:
541       \tl_use:N \l__zhnum_rod_font_tl
542       \bool_set_true:N \l__zhnum_rod_kerning_bool
543       \bool_set_false:N \l__zhnum_rod_kern_seen_bool
544       \__zhnum_rod_number:e {#1}
545       \group_end:
546     }
547 }
548 {
549   \cs_new_protected:Npn \zhrodbox #1
550     { \msg_error:nn { zhnumber } { rod-engine-unsupported } }
551 }

```

`\__zhnum_rod_kern:` 把已经展开成字符序列的算筹逐字排出,字符之间插入字距。先用 `e` 变体把 `\__zhnum_rod_number:n` 展开成纯字符,再逐字处理,这样字距逻辑不必重复位值判断。

用 `\kern` 而不是 `\skip_horizontal:n`:后者排出的是 glue,而 glue 是合法的断行点,于是一串筹码会被拆到两行去(实测 `\hsize=60pt` 下 123456789 断成三行)。`\kern` 不构成断点。这个缺陷由本地盲审发现。

但这只保证本命令自己插入的间距不构成断点,并不保证整串筹码一定不断行:在 `ctexart` 一类加载了 `xeCJK` 的文档里,筹码本身不是汉字 (`\XeTeXcharclass` 为 0),而占位的 `○` 与 `minus=slash` 的 `\` 都是 (为 1),`xeCJK` 会在这两类字符的交界处插入 `\CJKecglue`,那是 glue、因而是可断点。所以 `zero=fill` 时每个 `○` 两侧各有一处, `minus=slash` 时负号与首位筹码之间也有一处。

这些位置是否真的断,取决于断行的代价:实测 `fontset=lxgw`、`\hsize=60pt` 下, `\zhrodbox{10203040}` 在 `zero=fill` 时会断开(`○` 提供了多个可断点),而同一串在 `zero=omit` 时不断(没有 `○`,也就没有那些可断点);负号那一处虽然是 glue,实测在 `20pt` 与 `60pt` 下都不会实际断开,因为断在那里会把 `\` 单独留下、代价过高。

结论是:`\kern` 保证的只是本命令自己插入的间距不构成断点。要整串筹码一定不断,请自己把它放进 `\mbox`,或把 `\CJKecglue` 设为 `\kern`。

```

552 \cs_new:Npn \__zhnum_rod_kern:

```

```

553 {
554   \bool_if:NT \l__zhnum_rod_kerning_bool
555   { \exp_not:V \c__zhnum_rod_kern_node_tl }
556 }
557 \bool_new:N \l__zhnum_rod_kerning_bool
558 \tl_const:Nn \c__zhnum_rod_kern_node_tl { \__zhnum_rod_kern_node: }

```

字距排在每个筹码之前,因此第一个筹码前会多出一个。它不能靠「是不是第一位」来判断——zero=omit 时前导零不排除任何东西,「第一位」与「第一个实际排出来的筹码」不是一回事(初版按位置插字距,于是\zhrodbox{100} 在 omit 下留下两个多余的字距,把后面的内容拉进筹码里,实测宽度 7.2pt 而可展开版是 10.0pt)。

做法是让 __zhnum_rod_kern_node: 在盒子里第一次调用时什么都不排、其后才排字距。它只在 \zhrodbox 的分组内使用,所以这个状态随分组自动复位;\zhrod 不开 \l__zhnum_rod_kerning_bool,展开结果里连这个记号都不会出现。

```

559 \cs_new_protected:Npn \__zhnum_rod_kern_node:
560 {
561   \bool_if:NTF \l__zhnum_rod_kern_seen_bool
562   { \tex_kern:D \dim_eval:n { \l__zhnum_rod_kern_tl } \scan_stop: }
563   { \bool_set_true:N \l__zhnum_rod_kern_seen_bool }
564 }
565 \bool_new:N \l__zhnum_rod_kern_seen_bool

```

\zhdate 输出中文日期。

```

566 \msg_new:nnn { zhnumber } { weekday-unavailable }
567 { Weekday ~ is ~ unavailable. ~ Please ~ assign ~ the ~ `day'. }
568 \NewExpandableDocumentCommand \zhdate { +s +m }
569 {
570   \__zhnum_date:www #2 / / / \q_stop
571   \bool_if:NT #1 { \__zhnum_date_star:www #2 / / / \q_stop }
572 }

```

为确保在输入 yyyy 或 yyyy/mm 的情况下提供足够的 delimiter, 辅助函数 __zhnum_date:www 添加一个额外的/#4 并在调用时添加等量的 delimiter¹。

```

573 \cs_new:Npn \__zhnum_date:www #1 / #2 / #3 / #4 \q_stop
574 { \__zhnum_date_aux:nnn {#1} {#2} {#3} }
575 \cs_new:Npn \__zhnum_date_star:www #1 / #2 / #3 / #4 \q_stop
576 {
577   \tl_if_blank:ETF {#3}
578   { \msg_expandable_error:nnn { zhnumber } { weekday-unavailable } { } }
579   { \__zhnum_week_day:ww #1/#2/#3 \q_stop }
580 }

```

\zhtoday 输出当天日期。

```

581 \cs_new:Npn \zhtoday
582 {
583   \__zhnum_date_aux:nnn
584   { \int_value:w \tex_year:D }
585   { \tex_month:D }
586   { \tex_day:D }
587 }

```

```

\__zhnum_date_aux:nnn 588 \cs_new:Npn \__zhnum_date_aux:nnn
589 {
590   \bool_if:NTF \l__zhnum_time_bool
591   { \__zhnum_date_aux:NNnnnn \zhnum_digits_null:n \zhnum_int:n { } }
592   { \__zhnum_date_aux:Nnnnn \int_to_arabic:n { \l__zhnum_arabic_sep_tl } }
593 }
594 \cs_new:Npn \__zhnum_date_aux:Nnnnn #1#2
595 { \__zhnum_date_aux:NNnnnn #1#1 { \zhnum_exp_not:o {#2} } }
596 \zhnum_expand_wrap:wn
597 \cs_new:Npn \__zhnum_date_aux:NNnnnn #1#2#3#4#5#6

```

¹<https://tex.stackexchange.com/a/764551/299948>

```

598 {
599   \tl_if_empty:xF {#4} { #1 {#4} #3 \zhnum_output:n { year } }
600   \tl_if_empty:xF {#5} { #3 #2 {#5} #3 \zhnum_output:n { month } }
601   \tl_if_empty:xF {#6} { #3 #2 {#6} #3 \zhnum_output:n { day } }
602 }

```

\zhweekday 输出星期

```

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

```

605 \zhnum_expand_wrap:wn
606 \cs_new:Npn \__zhnum_week_day:www #1/#2/#3 \q_stop
607 {
608   \if_case:w \zhnum_Zeller:nnn {#1} {#2} {#3} \exp_stop_f:
609     \zhnum_output:n { sat }
610     \or: \zhnum_output:n { sun }
611     \or: \zhnum_output:n { mon }
612     \or: \zhnum_output:n { tue }
613     \or: \zhnum_output:n { wed }
614     \or: \zhnum_output:n { thu }
615     \or: \zhnum_output:n { fri }
616   \fi:
617 }

```

\zhnum_Zeller:nnn 用 Zeller 公式² 计算星期几。

```

\zhnum_Zeller_aux:Nnnn
\zhnum_two_digits:n
618 \cs_new:Npn \zhnum_Zeller:nnn #1#2#3
619 {
620   \int_compare:nNnTF
621     { #1 \zhnum_two_digits:n {#2} \zhnum_two_digits:n {#3} } > { 1582 10 04 }
622     { \__zhnum_Zeller_aux:Nnnn \zhnum_Zeller_Gregorian:nnn }
623     { \__zhnum_Zeller_aux:Nnnn \zhnum_Zeller_Julian:nnn }
624   {#1} {#2} {#3}
625 }
626 \cs_new:Npn \__zhnum_Zeller_aux:Nnnn #1#2#3#4
627 {
628   \int_compare:nNnTF {#3} < 3
629     { #1 { #2 - 1 } { #3 + 12 } {#4} }
630     { #1 {#2} {#3} {#4} }
631 }
632 \cs_new:Npn \zhnum_two_digits:n #1
633 {
634   \int_compare:nNnT {#1} < { 10 } { 0 }
635   \int_eval:n {#1}
636 }

```

\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 $Y --$ 。

```

637 \cs_new:Npn \zhnum_Zeller_Gregorian:nnn #1#2#3
638 {
639   \int_mod:nn
640     {
641       (#3)
642       + \int_div_truncate:nn { 26 * ( #2 + 1 ) } { 10 }
643       + (#1)
644       + \int_div_truncate:nn {#1} { 4 }
645       + 6 * \int_div_truncate:nn {#1} { 100 }
646       + \int_div_truncate:nn {#1} { 400 }
647     }
648     { 7 }
649 }

```

²http://en.wikipedia.org/wiki/Zeller's_congruence

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

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

```

650 \cs_new:Npn \zhnum_Zeller_Julian:nnn #1#2#3
651 {
652   \int_mod:nn
653   {
654     (#3)
655     + \int_div_truncate:nn { 26 * ( #2 + 1 ) } { 10 }
656     + (#1)
657     + \int_div_truncate:nn {#1} { 4 }
658     + 5
659   }
660   { 7 }
661 }

```

\zhctime 输出时间。

```

662 \zhnum_expand_wrap:wn
663 \cs_new:Npn \zhctime #1
664 { \__zhnum_time:ww #1 \q_stop }
665 \use:e
666 {
667   \cs_new:Npn \exp_not:N \__zhnum_time:ww
668     #1 \c_colon_str #2 \exp_not:N \q_stop
669   }
670 { \__zhnum_time_aux:nn {#1} {#2} }

```

\zhcurrtime 输出当前时间。

```

671 \zhnum_expand_wrap:wn
672 \cs_new:Npn \zhcurrtime
673 {
674   \__zhnum_time_aux:nn
675   { \int_div_truncate:nn \tex_time:D { 60 } }
676   { \int_mod:nn \tex_time:D { 60 } }
677 }

```

```

\__zhnum_time_aux:nn 678 \cs_new:Npn \__zhnum_time_aux:nn
\__zhnum_time_aux:Nnnn 679 {
680   \bool_if:NTF \l__zhnum_time_bool
681   { \__zhnum_time_aux:Nnnn \zhnum_int:n { } }
682   { \__zhnum_time_aux:Nnnn \int_to_arabic:n { \l__zhnum_arabic_sep_tl } }
683 }
684 \cs_new:Npn \__zhnum_time_aux:Nnnn #1#2#3#4
685 {
686   #1 {#3} #2 \zhnum_output:n { hour } #2
687   #1 {#4} #2 \zhnum_output:n { minute }
688 }

```

\zhnum_scale_map:n 大数系统的映射。

```

\zhnum_scale_map_loop:n 689 \cs_new:Npn \zhnum_scale_map:n #1
690 {
691   \tl_if_exist:cTF { l__zhnum_s #1 _tl }
692   { \zhnum_output:n { s #1 } }
693   { \zhnum_scale_map_hook:n {#1} }
694 }
695 \cs_new:Npn \zhnum_scale_map_loop:n #1
696 { \zhnum_scale_map:n { \int_mod:nn {#1} \l__zhnum_scale_int } }
697 \cs_generate_variant:Nn \zhnum_scale_map:n { f }
698 \int_new:N \l__zhnum_scale_int
699 \int_set:Nn \l__zhnum_scale_int { 11 }
700 \cs_new_eq:NN \zhnum_scale_map_hook:n \zhnum_scale_map_loop:n
701 \tl_new:c { l__zhnum_s 0 _tl }

```

`\zhnumExtendScaleMap` 扩展进位系统。

```

702 \NewDocumentCommand \zhnumExtendScaleMap { > { \TrimSpaces } +o +m }
703 {
704   \int_zero:N \l__zhnum_tmp_int
705   \clist_map_function:nN {#2} \zhnum_set_scale:n
706   \tl_if_novalue:nF {#1}
707   { \cs_set:Npn \zhnum_scale_map_hook:n ##1 {#1} }
708 }

\zhnum_set_scale:n 709 \cs_new_protected:Npn \zhnum_set_scale:n #1
710 {
711   \int_incr:N \l__zhnum_tmp_int
712   \exp_args:Nc \__zhnum_set_scale:Nn
713   { l__zhnum_s \int_eval:n { \l__zhnum_tmp_int + 11 } _tl }
714   {#1}
715 }
716 \cs_new_protected:Npn \__zhnum_set_scale:Nn #1
717 {
718   \tl_if_exist:NF #1
719   {
720     \tl_new:N #1
721     \int_incr:N \l__zhnum_scale_int
722   }
723   \tl_set:Nn #1
724   {
725     \int_new:N \l__zhnum_tmp_int

```

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

```

726 \zhnum_expand_wrap:wn
727 \cs_new:Npn \zhnum_ganzhi_normal:nnn #1#2#3
728 {
729   \int_compare:nNnF {#1} < \c_one_int
730   {
731     \cs_if_free:cF { l__zhnum_ #2 _ #1 _tl }
732     { \zhnum_output:n { #2 _ #1 } }
733   }
734 }

```

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

`\__zhnum_ganzhi_cyclic_mod:nnnn`

```

735 \zhnum_expand_wrap:wn
736 \cs_new:Npn \zhnum_ganzhi_cyclic:nnn #1#2#3
737 {
738   \int_compare:nNnF {#1} = \c_zero_int
739   {
740     \tl_if_exist:cTF { l__zhnum_ #2 _ #1 _tl }
741     { \zhnum_output:n { #2 _ #1 } }
742     {
743       \__zhnum_ganzhi_cyclic_mod:fnnn
744       { \int_mod:nn {#1} {#3} } {#1} {#2} {#3}
745     }
746   }
747 }
748 \cs_new:Npn \__zhnum_ganzhi_cyclic_mod:nnnn #1#2#3#4
749 {
750   \int_compare:nNnTF {#2} > \c_zero_int
751   { \zhnum_output:n { #3 _ #1 } }
752   {
753     \int_compare:nNnTF {#1} = \c_zero_int
754     { \zhnum_output:n { #3 _ 1 } }
755     { \zhnum_output:n { #3 _ \int_eval:n { #1 + #4 + 1 } } }
756   }
757 }
758 \cs_generate_variant:Nn \__zhnum_ganzhi_cyclic_mod:nnnn { f }

```

`\zhnum_ganzhi:nnn` 默认不对超出范围的数字取模。

```

759 \cs_new_eq:NN \zhnum_ganzhi:nnn \zhnum_ganzhi_normal:nnn

```

```
760 \cs_generate_variant:Nn \zhnum_ganzhi:nnn { f }
```

`\zhtiangan` 天干。

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

`\zhdizhi` 地支。

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

`\zhganzhi` 干支。

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

`\zhganzhinian` 干支纪年。

```
767 \cs_new:Npn \zhganzhinian #1
768 { \zhnum_ganzhi_nian:f { \int_eval:n {#1} } }
```

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

```
769 \zhnum_expand_wrap:wn
770 \cs_new:Npn \zhnum_ganzhi_nian:n #1
771 {
772   \int_compare:nNnTF {#1} > \c_zero_int
773   { \zhnum_output:n { ganzhi_ \int_mod:nn { #1 + 57 } { 60 } } }
774   {
775     \int_compare:nNnF {#1} = \c_zero_int
776     {
777       \zhnum_output:n
778       {
779         ganzhi_ \int_eval:n
780         { \int_mod:nn { #1 - 2 } { 60 } + 60 }
781       }
782     }
783   }
784 }
785 \cs_generate_variant:Nn \zhnum_ganzhi_nian:n { f }
```

根据需要设置中文阿拉伯数字。

```
786 \tl_new:N \l__zhnum_kv_tl
787 \tl_new:N \l__zhnum_tmp_tl
788 \group_begin:
789 \tl_build_begin:N \l__zhnum_kv_tl
790 \int_step_inline:nn { 10 }
791 {
792   \tl_new:c { l__zhnum_ #1 _tl }
793   \tl_build_put_right:Ne \l__zhnum_kv_tl
794   {
795     #1 .tl_set:N = \exp_not:c { l__zhnum_normal_ #1 _tl } ,
796     F#1 .tl_set:N = \exp_not:c { l__zhnum_financial_ #1 _tl } ,
797     E \int_eval:n { #1 * 4 }
798     .tl_set:N = \exp_not:c { l__zhnum_ s#1 _tl } ,
799   }
800 }
801 \clist_map_inline:nn { 0 , 100 , 1000 }
802 {
803   \tl_new:c { l__zhnum_ #1 _tl }
804   \tl_build_put_right:Ne \l__zhnum_kv_tl
805   {
806     #1 .tl_set:N = \exp_not:c { l__zhnum_normal_ #1 _tl } ,
807     F#1 .tl_set:N = \exp_not:c { l__zhnum_financial_ #1 _tl } ,
808   }
809 }
810 \clist_map_inline:nn
```

```

811 {
812     20 , 30 , 40 , 200 ,
813     dot , and , parts , year , month , day , hour , minute
814 }
815 {
816     \tl_build_put_right:Ne \l__zhnum_kv_tl
817     { #1 .tl_set:N = \exp_not:c { l__zhnum_ #1 _tl } , }
818 }
819 \tl_build_put_right:Ne \l__zhnum_kv_tl
820 {
821     - .tl_set:N = \exp_not:N \l__zhnum_minus_tl ,
822     -0 .tl_set:N = \exp_not:N \l__zhnum_null_tl ,
823     E2 .tl_set:N = \exp_not:c { l__zhnum_normal_ 100 _tl } ,
824     E3 .tl_set:N = \exp_not:c { l__zhnum_normal_ 1000 _tl } ,
825     FE2 .tl_set:N = \exp_not:c { l__zhnum_financial_ 100 _tl } ,
826     FE3 .tl_set:N = \exp_not:c { l__zhnum_financial_ 1000 _tl } ,
827     E44 .tl_set:N = \exp_not:c { l__zhnum_s11 _tl }
828 }

```

expl3 2023-12-14 起, \tl_build_get:NN 是废弃命令。当 l3debug 的 deprecation 开启, 废弃命令被重定义为 \outer 的, 无法在其他命令的参数中使用, 所以这里用 \use:c 绕过。

```

829 \cs_if_exist:NTF \tl_build_get_intermediate:NN
830 { \tl_build_get_intermediate:NN }
831 { \use:c { tl_build_get:NN } }
832 \l__zhnum_kv_tl \l__zhnum_tmp_tl
833 \cs_set:Npn \__zhnum_tmp:w #1 . #2 \q_stop
834 { , #1 .groups:n = { user } }
835 \clist_map_inline:Nn \l__zhnum_tmp_tl
836 {
837     \tl_build_put_right:Ne \l__zhnum_kv_tl
838     { \__zhnum_tmp:w #1 \q_stop }
839 }
840 \tl_build_put_right:Nn \l__zhnum_kv_tl
841 {
842     ,
843     weekday .tl_set:N = \l__zhnum_weekday_tl ,
844     weekday .groups:n = { user , pre , weekday } ,
845 }
846 \clist_map_inline:nn
847 { mon , tue , wed , thu , fri , sat , sun }
848 {
849     \tl_build_put_right:Ne \l__zhnum_kv_tl
850     {
851         #1 .tl_set:N = \exp_not:c { l__zhnum_ #1 _tl } ,
852         #1 .groups:n = { user , pos , day } ,
853     }
854 }

```

下面三组键的目标变量必须各自独立: Tn 写 \l__zhnum_tiangang_#1_tl、Dn 写 \l__zhnum_dizhi_#1_tl、GZn 写 \l__zhnum_ganzhi_#1_tl。zhganzhi 的输出由天干与地支两个变量组合而成, 只有在用户显式设置 GZn 时才直接取干支变量, 所以三者不能共用同一个目标。

#1077 之前 Tn 抄成了 \l__zhnum_ganzhi_#1_tl, 与 GZn 指向同一个变量: 设置 T1 改掉的是 \zhganzhi{<1>} 而不是 \zhtiangan{<1>}。此处笔误单看 Tn 那两行是看不出来的——步数 10(天干正好十个)与 groups 里的 tiandi 都是对的, 只有变量名错。判断这类问题要把三组键并排比对, 而不是逐组检查其内部自治性。

```

855 \int_step_inline:nn { 10 }
856 {
857     \tl_build_put_right:Ne \l__zhnum_kv_tl
858     {
859         T#1 .tl_set:N = \exp_not:c { l__zhnum_tiangang_ #1 _tl } ,
860         T#1 .groups:n = { user , pre , tiandi } ,
861     }

```

```

862     }
863     \int_step_inline:nn { 12 }
864     {
865         \tl_build_put_right:Ne \l__zhnum_kv_tl
866         {
867             D#1 .tl_set:N = \exp_not:c { l__zhnum_dizhi_ #1 _tl } ,
868             D#1 .groups:n = { user , pre , tiandi } ,
869         }
870     }
871     \int_step_inline:nn { 60 }
872     {
873         \tl_build_put_right:Ne \l__zhnum_kv_tl
874         {
875             GZ#1 .tl_set:N = \exp_not:c { l__zhnum_ganzhi_ #1 _tl } ,
876             GZ#1 .groups:n = { user , pos , ganzhi } ,
877         }
878     }
879     \tl_build_end:N \l__zhnum_kv_tl
880     \exp_args:NNno \group_end:
881     \keys_define:nn
882     { zhnum / options }
883     { \l__zhnum_kv_tl }

```

将配置文件中的中文数字保存到 prop 变量中。

```

884 \cs_new_protected:Npn \zhnum_set_digits_map:nn #1
885 { \prop_put:Nnn \l__zhnum_cfg_map_prop {#1} }
886 \cs_new_protected:Npn \zhnum_set_digits_map:nnn #1#2#3
887 {
888     \prop_put_if_new:Nnn \l__zhnum_cfg_map_prop {#1} {#3}
889     \prop_put:Nnn \l__zhnum_cfg_map_var_prop {#1_#2} {#3}
890 }
891 \cs_new_protected:Npn \zhnum_set_financial_map:nn #1
892 { \prop_put:Nnn \l__zhnum_cfg_map_finan_prop {#1} }
893 \cs_new_protected:Npn \zhnum_set_financial_map:nnn #1#2#3
894 {
895     \prop_put_if_new:Nnn \l__zhnum_cfg_map_finan_prop {#1} {#3}
896     \prop_put:Nnn \l__zhnum_cfg_map_var_prop { financial_#1_#2 } {#3}
897 }
898 \cs_new_protected:Npn \zhnum_set_tiangang_map:nn #1
899 { \prop_put:Nnn \l__zhnum_cfg_map_ganzhi_prop { tiangan_#1 } }
900 \cs_new_protected:Npn \zhnum_set_dizhi_map:nn #1
901 { \prop_put:Nnn \l__zhnum_cfg_map_ganzhi_prop { dizhi_#1 } }
902 \prop_new:N \l__zhnum_cfg_map_prop
903 \prop_new:N \l__zhnum_cfg_map_var_prop
904 \prop_new:N \l__zhnum_cfg_map_finan_prop
905 \prop_new:N \l__zhnum_cfg_map_ganzhi_prop

```

将 prop 表转化到单独的 tl 变量。

```

906 \cs_new_protected:Npn \zhnum_parse_config:
907 {
908     \tl_clear_new:N \l__zhnum_reset_tl
909     \tl_clear_new:N \l__zhnum_reset_simp_tl
910     \tl_clear_new:N \l__zhnum_reset_trad_tl
911     \tl_clear_new:N \l__zhnum_set_ancient_tl
912     \tl_clear_new:N \l__zhnum_set_normal_tl
913     \tl_clear_new:N \l__zhnum_reset_ancient_tl
914     \tl_clear_new:N \l__zhnum_reset_normal_tl
915     \tl_clear_new:N \l__zhnum_reset_financial_tl
916     \prop_map_function:NN \l__zhnum_cfg_map_prop \zhnum_check_simp:nn
917     \zhnum_set_ganzhi:
918     \zhnum_reset_all:
919 }
920 \cs_new_protected:Npn \zhnum_check_simp:nn #1#2
921 {
922     \prop_get:NnNTF \l__zhnum_cfg_map_var_prop
923     { #1_ancient } \l__zhnum_ancient_tl

```

```

924     { \__zhnum_add_reset_ancient:nN {#1} \l__zhnum_ancient_tl }
925     {
926         \__zhnum_check_simp_aux:nn {#2} {#1}
927         \prop_get:NnNT \l__zhnum_cfg_map_finan_prop {#1} \l__zhnum_tmp_tl
928         {
929             \exp_args:No \__zhnum_check_simp_aux:nn
930             { \l__zhnum_tmp_tl } { financial_ #1 }
931             \__zhnum_add_reset_financial:n {#1}
932         }
933     }
934 }
935 \cs_new_protected:Npn \__zhnum_check_simp_aux:nn #1#2
936 {
937     \prop_get:NnNTF \l__zhnum_cfg_map_var_prop
938     { #2 _trad } \l__zhnum_trad_tl
939     {
940         \prop_get:NnNF \l__zhnum_cfg_map_var_prop
941         { #2 _simp } \l__zhnum_simp_tl
942         { \tl_set:Nn \l__zhnum_simp_tl {#1} }
943         \__zhnum_add_reset_simp:nNN
944         {#2} \l__zhnum_simp_tl \l__zhnum_trad_tl
945     }
946     { \__zhnum_add_reset:nn {#2} {#1} }
947 }
948 \tl_new:N \l__zhnum_simp_tl
949 \tl_new:N \l__zhnum_trad_tl
950 \tl_new:N \l__zhnum_ancient_tl
951 \cs_new_protected:Npn \__zhnum_add_reset:nn #1#2
952 {
953     \tl_put_right:Ne \l__zhnum_reset_tl
954     {
955         \tl_set:Nn \exp_not:c { l__zhnum_ #1 _tl }
956         { \exp_not:n {#2} }
957     }
958 }
959 \cs_new_protected:Npn \__zhnum_add_reset_simp:nNN #1#2#3
960 {
961     \tl_put_right:Ne \l__zhnum_reset_simp_tl
962     {
963         \tl_set:Nn \exp_not:c { l__zhnum_ #1 _tl }
964         { \exp_not:o {#2} }
965     }
966     \tl_put_right:Ne \l__zhnum_reset_trad_tl
967     {
968         \tl_set:Nn \exp_not:c { l__zhnum_ #1 _tl }
969         { \exp_not:o {#3} }
970     }
971 }
972 \cs_new_protected:Npn \__zhnum_add_reset_financial:n #1
973 {
974     \tl_put_right:Ne \l__zhnum_set_normal_tl
975     {
976         \tl_set_eq:NN
977         \exp_not:c { l__zhnum_normal_ #1 _tl }
978         \exp_not:c { l__zhnum_ #1 _tl }
979     }
980     \tl_put_right:Ne \l__zhnum_reset_normal_tl
981     {
982         \tl_set_eq:NN
983         \exp_not:c { l__zhnum_ #1 _tl }
984         \exp_not:c { l__zhnum_normal_ #1 _tl }
985     }
986     \tl_put_right:Ne \l__zhnum_reset_financial_tl
987     {
988         \tl_set_eq:NN
989         \exp_not:c { l__zhnum_ #1 _tl }
990         \exp_not:c { l__zhnum_financial_ #1 _tl }

```

```

991     }
992 }
993 \cs_new_protected:Npn \__zhnum_add_reset_ancient:nN #1#2
994 {
995     \tl_put_right:Ne \l__zhnum_reset_ancient_tl
996     {
997         \tl_set:Nn \exp_not:c { l__zhnum_ #1 _tl }
998         { \exp_not:o {#2} }
999     }
1000     \tl_put_right:Ne \l__zhnum_set_ancient_tl
1001     {
1002         \tl_concat:NNN
1003         \exp_not:c { l__zhnum_ #1 _tl }
1004         \exp_not:c { l__zhnum_ \str_head:n {#1} _tl }
1005         \exp_not:c { l__zhnum_ 1 \str_tail:n {#1} _tl }
1006     }
1007 }

\zhnum_set_week_day: 1008 \cs_new_protected:Npn \zhnum_set_week_day:
\zhnum_reset_week_day: 1009 {
1010     \cs_set_protected:Npe \zhnum_reset_week_day:
1011     {
1012         \__zhnum_set_week_day:nn { mon } { 1 }
1013         \__zhnum_set_week_day:nn { tue } { 2 }
1014         \__zhnum_set_week_day:nn { wed } { 3 }
1015         \__zhnum_set_week_day:nn { thu } { 4 }
1016         \__zhnum_set_week_day:nn { fri } { 5 }
1017         \__zhnum_set_week_day:nn { sat } { 6 }
1018         \__zhnum_set_week_day:nn { sun } { day }
1019     }
1020 }
1021 \cs_new_eq:NN \zhnum_reset_week_day: \prg_do_nothing:
1022 \cs_new:Npn \__zhnum_set_week_day:nn #1#2
1023 {
1024     \tl_set:Ne \exp_not:c { l__zhnum_ #1 _tl }
1025     {
1026         \exp_not:N \exp_not:o { \exp_not:N \l__zhnum_weekday_tl }
1027         \exp_not:N \exp_not:n { \exp_not:v { l__zhnum_ #2 _tl } }
1028     }
1029 }

\zhnum_set_ganzhi: 1030 \cs_new_protected:Npn \zhnum_set_ganzhi:
\zhnum_reset_ganzhi: 1031 {
1032     \prop_map_function:NN
1033     \l__zhnum_cfg_map_ganzhi_prop
1034     \__zhnum_add_reset:nn
1035 }
1036 \cs_new_protected:Npn \__zhnum_reset_ganzhi:nn #1#2
1037 { \tl_set:cn { l__zhnum_ #1 _tl } {#2} }
1038 \cs_new:Npn \zhnum_zero_mod:nn #1#2
1039 { \exp_args:Nf \__zhnum_zero_mod_aux:nn { \int_mod:nn {#1} {#2} } {#2} }
1040 \cs_new:Npn \__zhnum_zero_mod_aux:nn #1#2
1041 { \int_compare:nNnTF {#1} = \c_zero_int {#2} {#1} }
1042 \tl_new:c { l__zhnum_dizhi_ 0 _tl }
1043 \tl_new:c { l__zhnum_ganzhi_ 0 _tl }
1044 \tl_new:c { l__zhnum_tiangang_ 0 _tl }
1045 \group_begin:
1046 \cs_set:Npn \__zhnum_tmp:w #1
1047 {
1048     \tl_concat:NNN
1049     \exp_not:c { l__zhnum_ganzhi_ #1 _tl }
1050     \exp_not:c { l__zhnum_tiangang_ \zhnum_zero_mod:nn {#1} { 10 } _tl }
1051     \exp_not:c { l__zhnum_dizhi_ \zhnum_zero_mod:nn {#1} { 12 } _tl }
1052 }
1053 \cs_new_protected:Npe \zhnum_reset_ganzhi:
1054 {
1055     \tl_set_eq:NN
1056     \exp_not:c { l__zhnum_dizhi_ 0 _tl }

```

```

1057     \exp_not:c { l__zhnum_dizhi_12_tl }
1058     \tl_set_eq:NN
1059     \exp_not:c { l__zhnum_tiangang_0_tl }
1060     \exp_not:c { l__zhnum_tiangang_10_tl }
1061     \int_step_function:nN { 60 } \__zhnum_tmp:w
1062     \tl_set_eq:NN
1063     \exp_not:c { l__zhnum_ganzhi_0_tl }
1064     \exp_not:c { l__zhnum_ganzhi_60_tl }
1065   }
1066 \group_end:

\zhnum_reset_config: 1067 \cs_new_protected:Npn \zhnum_reset_config:
    \zhnum_reset_all: 1068 { \zhnum_load_cfg:o { \l__zhnum_encoding_str } }
\zhnum_reset_style: 1069 \cs_new_protected:Npn \zhnum_reset_all:
1070   {
1071     \zhnum_reset_main:
1072     \zhnum_reset_simp:
1073     \zhnum_set_week_day:
1074     \zhnum_reset_week_day:
1075     \zhnum_reset_ganzhi:
1076     \zhnum_reset_normal:
1077   }
1078 \cs_new_protected:Npn \zhnum_reset_main:
1079   {
1080     \tl_use:N \l__zhnum_reset_tl
1081     \tl_use:N \l__zhnum_set_normal_tl
1082     \tl_concat:NNN
1083       \l__zhnum_reset_normal_tl
1084       \l__zhnum_reset_normal_tl
1085       \c__zhnum_set_zero_tl
1086     \tl_concat:NNN
1087       \l__zhnum_reset_financial_tl
1088       \l__zhnum_reset_financial_tl
1089       \c__zhnum_set_zero_tl
1090     \tl_concat:NNN
1091       \l__zhnum_reset_financial_tl
1092       \l__zhnum_reset_financial_tl
1093       \l__zhnum_set_ancient_tl
1094   }
1095 \tl_const:Ne \c__zhnum_set_zero_tl
1096   {
1097     \tl_set_eq:NN
1098     \exp_not:N \l__zhnum_zero_tl
1099     \exp_not:c { l__zhnum_0_tl }
1100   }
1101 \tl_new:N \l__zhnum_zero_tl
1102 \cs_new_protected:Npn \zhnum_reset_style:
1103   {
1104     \zhnum_reset_simp:
1105     \zhnum_reset_normal:
1106   }
1107 \cs_new_protected:Npn \zhnum_reset_simp:
1108   {
1109     \bool_if:NTF \l__zhnum_simp_bool
1110       { \tl_use:N \l__zhnum_reset_simp_tl }
1111       { \tl_use:N \l__zhnum_reset_trad_tl }
1112   }
1113 \cs_new_protected:Npn \zhnum_reset_normal:
1114   {
1115     \bool_if:NTF \l__zhnum_normal_bool
1116       {
1117         \tl_use:N \l__zhnum_reset_normal_tl
1118         \__zhnum_reset_ancient:
1119         \__zhnum_reset_zero:
1120       }
1121       { \tl_use:N \l__zhnum_reset_financial_tl }
1122   }
1123 \cs_new_protected:Npn \__zhnum_reset_ancient:

```

```

1124 {
1125   \bool_if:NTF \l__zhnum_ancient_bool
1126   { \tl_use:N \l__zhnum_reset_ancient_tl }
1127   { \tl_use:N \l__zhnum_set_ancient_tl }
1128 }
1129 \cs_new_protected:Npe \__zhnum_reset_zero:
1130 {
1131   \exp_not:n { \bool_if:NT \l__zhnum_null_bool }
1132   {
1133     \tl_set_eq:NN
1134     \exp_not:c { l__zhnum_0_tl }
1135     \exp_not:N \l__zhnum_null_tl
1136   }
1137 }

```

\zhnum_load_cfg:n 根据选定编码载入配置文件。

```

1138 \cs_new_protected:Npn \zhnum_load_cfg:n #1
1139 {
1140   \zhnum_set_cfg_name:Nn \l__zhnum_cfg_str {#1}
1141   \str_if_eq:NNTF \l__zhnum_cfg_str \l__zhnum_last_cfg_str
1142   { \zhnum_reset_all: }
1143   {
1144     \zhnum_update_cfg:n {#1}
1145     \zhnum_parse_config:
1146   }
1147 }
1148 \cs_generate_variant:Nn \zhnum_load_cfg:n { o }
1149 \cs_new_protected:Npn \zhnum_update_cfg:n #1
1150 {
1151   \prop_if_exist:cTF { g__zhnum_cfg_ \l__zhnum_cfg_str _prop }
1152   { \__zhnum_set_cfg_prop: }
1153   { \zhnum_input_cfg:n {#1} }
1154 }
1155 \cs_new_protected:Npn \__zhnum_set_cfg_prop:
1156 {
1157   \str_set_eq:NN \l__zhnum_last_cfg_str \l__zhnum_cfg_str
1158   \__zhnum_update_cfg_prop:N \prop_set_eq:Nc
1159 }
1160 \cs_new_protected:Npn \zhnum_input_cfg:n #1
1161 {
1162   \file_get_full_name:nNTF { zhnumber - #1 .cfg } \l__zhnum_cfg_file_tl
1163   {
1164     \bool_set_false:N \l__zhnum_reset_bool
1165     \__zhnum_update_cfg_prop:N \__zhnum_prop_initial:Nn
1166     \group_begin:
1167       \zhnum_set_catcode:
1168       \exp_args:No \file_input:n { \l__zhnum_cfg_file_tl }
1169       \__zhnum_update_cfg_prop:N \__zhnum_prop_gset_eq:Nn
1170     \group_end:
1171     \__zhnum_set_cfg_prop:
1172   }
1173   { \msg_error:nne { zhnumber } { file-not-found } {#1} }
1174 }
1175 \tl_new:N \l__zhnum_cfg_file_tl
1176 \cs_new_protected:Npn \__zhnum_update_cfg_prop:N
1177 { \exp_args:No \__zhnum_update_cfg_prop_aux:nN { \l__zhnum_cfg_str } }
1178 \cs_new_protected:Npn \__zhnum_update_cfg_prop_aux:nN #1#2
1179 {
1180   #2 \l__zhnum_cfg_map_prop { g__zhnum_cfg_ #1 _prop }
1181   #2 \l__zhnum_cfg_map_var_prop { g__zhnum_cfg_var_ #1 _prop }
1182   #2 \l__zhnum_cfg_map_finan_prop { g__zhnum_cfg_finan_ #1 _prop }
1183   #2 \l__zhnum_cfg_map_ganzhi_prop { g__zhnum_cfg_ganzhi_ #1 _prop }
1184 }
1185 \cs_new_protected:Npn \__zhnum_prop_initial:Nn #1#2
1186 {
1187   \prop_clear:N #1
1188   \prop_new:c {#2}

```

```

1189 }
1190 \cs_new_protected:Npn \__zhnum_prop_gset_eq:Nn #1#2
1191 { \prop_gset_eq:cN {#2} #1 }
1192 \str_new:N \l__zhnum_cfg_str
1193 \str_new:N \l__zhnum_last_cfg_str
1194 \bool_new:N \l__zhnum_reset_bool
1195 \msg_new:nnnn { zhnumber } { file-not-found }
1196 { File~`#1'~not~found. }
1197 {
1198   The~requested~file~could~not~be~found~in~the~current~directory,~
1199   in~the~TeX~search~path~or~in~the~LaTeX~search~path.
1200 }

```

使用 $\text{\texttt{upTeX}}$ 的时候,也不必将汉字的首字符设置为活动字符。判断 $\text{\texttt{~~~~0021}}$ 是否为单个记号的办法对 $\text{\texttt{upTeX}}$ 不适用。

```

1201 \bool_const:Nn \c__zhnum_unicode_engine_bool
1202 {
1203   \bool_lazy_any_p:n
1204   {
1205     { \sys_if_engine_xetex_p: }
1206     { \sys_if_engine luatex_p: }
1207     { \sys_if_engine_uptex_p: }
1208   }
1209 }

```

设置与恢复配置文件前后的 `catcode`。 $\text{\texttt{pdf\LaTeX}}$ 需要将汉字的首字节设置为活动字符。

```

\zhnum_set_catcode:
\zhnum_set_cfg_name:Nn
\zhnum_reset_config:
1210 \bool_if:NTF \c__zhnum_unicode_engine_bool
1211 {
1212   \cs_new_eq:NN \zhnum_set_catcode: \prg_do_nothing:
1213   \cs_new_protected:Npn \zhnum_set_cfg_name:Nn #1#2
1214   {
1215     \str_set:Ne \l__zhnum_encoding_str {#2}
1216     \str_set_eq:NN #1 \l__zhnum_encoding_str
1217   }
1218 }
1219 {
1220   \cs_new_protected:Npn \zhnum_set_catcode:
1221   {
1222     \bool_if:NTF \l__zhnum_active_char_bool
1223     { \zhnum_set_active: }
1224     { \zhnum_set_other: }
1225   }
1226   \cs_new_protected:Npe \zhnum_set_active:
1227   {
1228     \int_step_function:nnN
1229     { 128 } { 255 } \char_set_catcode_active:n
1230   }
1231   \cs_new_protected:Npe \zhnum_set_other:
1232   {
1233     \int_step_function:nnN
1234     { 128 } { 255 } \char_set_catcode_other:n
1235   }
1236   \cs_new_protected:Npn \zhnum_set_cfg_name:Nn #1#2
1237   {
1238     \str_set:Ne \l__zhnum_encoding_str {#2}
1239     \str_set:Ne #1
1240     {
1241       \l__zhnum_encoding_str
1242       \bool_if:NTF \l__zhnum_active_char_bool
1243       { / active }
1244       { / other }
1245     }
1246   }
1247   \bool_new:N \l__zhnum_active_char_bool
1248   \bool_set_true:N \l__zhnum_active_char_bool
1249 }

```

\zhnum_set_encoding:n 设置编码。

```
1250 \cs_new_protected:Npn \zhnum_set_encoding:n #1
1251 {
1252   \str_set:Nx \l__zhnum_encoding_str
1253     { \str_lowercase:n {#1} }
1254   \zhnum_load_cfg:o { \l__zhnum_encoding_str }
1255 }
1256 \str_new:N \l__zhnum_encoding_str
```

encoding 宏包设置选项。

```
style
null
reset
1257 \keys_define:nn { zhnum / options }
1258 {
1259   encoding .choices:nn =
1260     { UTF8 , GBK , Big5 }
1261     { \zhnum_set_encoding:n {#1} } ,
1262   encoding .default:n = { GBK } ,
1263   encoding / Bg5 .meta:n = { encoding = Big5 } ,
1264   encoding / unknown .code:n =
1265     { \msg_error:nnn { zhnumber } { encoding-invalid } {#1} } ,
1266   style .multichoice: ,
1267   style / Normal .code:n =
1268     {
1269       \bool_set_false:N \l__zhnum_ancient_bool
1270       \bool_set_true:N \l__zhnum_normal_bool
1271     } ,
1272   style / Financial .code:n =
1273     {
1274       \bool_set_false:N \l__zhnum_ancient_bool
1275       \bool_set_false:N \l__zhnum_normal_bool
1276     } ,
1277   style / Ancient .code:n =
1278     {
1279       \bool_set_true:N \l__zhnum_ancient_bool
1280       \bool_set_true:N \l__zhnum_normal_bool
1281     } ,
1282   style / Simplified .code:n = { \bool_set_true:N \l__zhnum_simp_bool } ,
1283   style / Traditional .code:n = { \bool_set_false:N \l__zhnum_simp_bool } ,
1284   style .default:n = { Normal , Simplified } ,
1285   style .groups:n = { style } ,
1286   null .bool_set:N = \l__zhnum_null_bool ,
1287   null .groups:n = { style } ,
1288   time .choice: ,
1289   time / Chinese .code:n = { \bool_set_true:N \l__zhnum_time_bool } ,
1290   time / Arabic .code:n = { \bool_set_false:N \l__zhnum_time_bool } ,
1291   time .default:n = { Arabic } ,
1292   reset .bool_set:N = \l__zhnum_reset_bool ,
1293   activechar .bool_set:N = \l__zhnum_active_char_bool ,
1294   ganzhi-cyclic .choice: ,
1295   ganzhi-cyclic / true .code:n =
1296     { \cs_set_eq:NN \zhnum_ganzhi:nnn \zhnum_ganzhi_cyclic:nnn } ,
1297   ganzhi-cyclic / false .code:n =
1298     { \cs_set_eq:NN \zhnum_ganzhi:nnn \zhnum_ganzhi_normal:nnn } ,
1299   ganzhi-cyclic .default:n = { true } ,
1300   arabicsep .tl_set:N = \l__zhnum_arabic_sep_tl
1301 }
```

算筹的键单独一个模块。它们与 \zhnumsetup 那批键作用不同:那批管的是「每个数字换成哪个汉字」(样式与编码),算筹管的是「按位值奇偶选哪个码区」,并且只在 Unicode 引擎下有意义。混进同一模块会让 style 一类键在算筹下显得可用而实际无效。

zero 与既有的 null 作用也不同:null 管「0 映射成零还是〇」(\zhdigits 默认已用〇),zero 管「写不写」。

```
1302 \keys_define:nn { zhnum / rod }
1303 {
1304   units .choice: ,
```

```

1305 units / vertical .code:n =
1306 { \bool_set_true:N \l__zhnum_rod_units_vertical_bool } ,
1307 units / horizontal .code:n =
1308 { \bool_set_false:N \l__zhnum_rod_units_vertical_bool } ,
1309 units .default:n = { vertical } ,
1310 zero .choice: ,
1311 zero / fill .code:n =
1312 { \bool_set_false:N \l__zhnum_rod_zero_omit_bool } ,
1313 zero / omit .code:n =
1314 { \bool_set_true:N \l__zhnum_rod_zero_omit_bool } ,
1315 zero .default:n = { fill } ,
1316 minus .choice: ,
1317 minus / slash .code:n =
1318 { \bool_set_false:N \l__zhnum_rod_minus_overlay_bool } ,
1319 minus / overlay .code:n =
1320 { \bool_set_true:N \l__zhnum_rod_minus_overlay_bool } ,
1321 minus .default:n = { slash } ,
1322 font .tl_set:N = \l__zhnum_rod_font_tl ,
1323 kern .tl_set:N = \l__zhnum_rod_kern_tl ,
1324 zerochar .tl_set:N = \l__zhnum_rod_zero_tl ,
1325 dot .tl_set:N = \l__zhnum_rod_dot_tl
1326 }
1327 \NewDocumentCommand \zhrodsetup { +m }
1328 {
1329 \keys_set:nn { zhnum / rod } { #1 }
1330 \tex_ignorespaces:D
1331 }

1332 \bool_new:N \l__zhnum_simp_bool
1333 \bool_new:N \l__zhnum_normal_bool
1334 \bool_new:N \l__zhnum_ancient_bool
1335 \msg_new:nnnn { zhnumber } { encoding-invalid }
1336 { The~encoding~`#1'~is~invalid. }
1337 { Available~encodings~are~`UTF8',~`GBK'~and~`Big5'. }

```

\zhnumsetup 在文档中设置 zhnumber 的接口。

```

1338 \NewDocumentCommand \zhnumsetup { +m }
1339 {
1340 \zhnum_set:n { #1 }
1341 \tex_ignorespaces:D
1342 }
1343 \cs_new_protected:Npn \zhnum_set:n #1
1344 {
1345 \bool_set_false:N \l__zhnum_reset_bool
1346 \keys_set_exclude_groups:nnnN { zhnum / options }
1347 { style , user } { #1 } \l__zhnum_kv_tl
1348 \tl_if_empty:NF \l__zhnum_kv_tl
1349 { \__zhnum_set_style: }
1350 \bool_if:NT \l__zhnum_reset_bool
1351 { \zhnum_reset_config: }
1352 }
1353 \cs_new_protected:Npn \__zhnum_set_style:
1354 {
1355 \keys_set_exclude_groups:nnoN { zhnum / options }
1356 { user } { \l__zhnum_kv_tl } \l__zhnum_tmp_tl
1357 \tl_if_empty:NTF \l__zhnum_tmp_tl
1358 { \zhnum_reset_style: }
1359 {
1360 \tl_if_eq:NNF \l__zhnum_tmp_tl \l__zhnum_kv_tl
1361 { \zhnum_reset_style: }
1362 \bool_if:NF \l__zhnum_reset_bool
1363 { \__zhnum_set_user: }
1364 }
1365 }
1366 \cs_new_protected:Npn \__zhnum_set_user:
1367 {
1368 \keys_set_exclude_groups:nnoN { zhnum / options }

```

```

1369     { pre , pos } { \l__zhnum_tmp_tl } \l__zhnum_kv_tl
1370     \tl_if_eq:NNF \l__zhnum_kv_tl \l__zhnum_tmp_tl
1371     { \zhnum_reset_normal: }
1372     \tl_if_empty:NF \l__zhnum_kv_tl
1373     { \__zhnum_set_pre_pos: }
1374   }
1375 \cs_new_protected:Npn \__zhnum_set_pre_pos:
1376 {
1377   \keys_set_exclude_groups:nnoN { zhnum / options }
1378   { pos } { \l__zhnum_kv_tl } \l__zhnum_tmp_tl
1379   \tl_if_eq:NNF \l__zhnum_tmp_tl \l__zhnum_kv_tl
1380   {
1381     \zhnum_reset_week_day:
1382     \zhnum_reset_ganzhi:
1383   }
1384   \tl_if_empty:NF \l__zhnum_kv_tl
1385   { \keys_set:no { zhnum / options } { \l__zhnum_kv_tl } }
1386 }

```

初始化设置和执行宏包选项。

```

1387 \keys_set:nn { zhnum / options }
1388 { style , time , arabicsep = { ~ } }
1389 \cs_if_exist:NTF \ProcessKeyOptions
1390 { \ProcessKeyOptions [ zhnum / options ] }
1391 {
1392   \RequirePackage { l3keys2e }
1393   \ProcessKeysOptions { zhnum / options }
1394 }

```

如果没有选定编码,则选用 UTF8。

```

1395 \str_if_empty:NTF \l__zhnum_encoding_str
1396 { \zhnum_set_encoding:n { UTF8 } }
1397 { \zhnum_reset_style: }
1398 </package>

```

第 4 节 中文数字配置文件

```

1399 <*config>
1400 <!*big5>
1401 \zhnum_set_digits_map:nnn { minus } { simp } { 负 }
1402 \zhnum_set_digits_map:nnn { minus } { trad } { 負 }
1403 </!big5>
1404 <*big5>
1405 \zhnum_set_digits_map:nn { minus } { 負 }
1406 </big5>
1407 \zhnum_set_digits_map:nn { 0 } { 零 }
1408 <!*big5>
1409 \zhnum_set_digits_map:nn { null } { 〇 }
1410 </!big5>
1411 <*big5>
1412 \zhnum_set_digits_map:nn { null } { 〇 }
1413 </big5>
1414 \zhnum_set_digits_map:nn { 1 } { 一 }
1415 \zhnum_set_digits_map:nn { 2 } { 二 }
1416 \zhnum_set_digits_map:nn { 3 } { 三 }
1417 \zhnum_set_digits_map:nn { 4 } { 四 }
1418 \zhnum_set_digits_map:nn { 5 } { 五 }
1419 \zhnum_set_digits_map:nn { 6 } { 六 }
1420 \zhnum_set_digits_map:nn { 7 } { 七 }
1421 \zhnum_set_digits_map:nn { 8 } { 八 }
1422 \zhnum_set_digits_map:nn { 9 } { 九 }
1423 \zhnum_set_digits_map:nn { 10 } { 十 }
1424 \zhnum_set_digits_map:nn { 100 } { 百 }

```

```

1425 \zhnum_set_digits_map:nn { 1000 } { 千 }
1426 \zhnum_set_digits_map:nnn { 20 } { ancient } { 廿 }
1427 \zhnum_set_digits_map:nnn { 30 } { ancient } { 卅 }
1428 \zhnum_set_digits_map:nnn { 40 } { ancient } { 卌 }
1429 \zhnum_set_digits_map:nnn { 200 } { ancient } { 佰 }
1430 <*/big5>
1431 \zhnum_set_digits_map:nnn { dot } { simp } { 点 }
1432 \zhnum_set_digits_map:nnn { dot } { trad } { 點 }
1433 </big5>
1434 <*big5>
1435 \zhnum_set_digits_map:nn { dot } { 點 }
1436 </big5>
1437 \zhnum_set_digits_map:nn { and } { 又 }
1438 \zhnum_set_digits_map:nn { parts } { 分之 }
1439 <*/big5>
1440 \zhnum_set_digits_map:nnn { s1 } { simp } { 万 }
1441 \zhnum_set_digits_map:nnn { s1 } { trad } { 萬 }
1442 \zhnum_set_digits_map:nnn { s2 } { simp } { 亿 }
1443 \zhnum_set_digits_map:nnn { s2 } { trad } { 億 }
1444 </big5>
1445 <*big5>
1446 \zhnum_set_digits_map:nn { s1 } { 萬 }
1447 \zhnum_set_digits_map:nn { s2 } { 億 }
1448 </big5>
1449 \zhnum_set_digits_map:nn { s3 } { 兆 }
1450 \zhnum_set_digits_map:nn { s4 } { 京 }
1451 \zhnum_set_digits_map:nn { s5 } { 垓 }
1452 \zhnum_set_digits_map:nn { s6 } { 秭 }
1453 \zhnum_set_digits_map:nn { s7 } { 穰 }
1454 <*/big5>
1455 \zhnum_set_digits_map:nnn { s8 } { simp } { 沟 }
1456 \zhnum_set_digits_map:nnn { s8 } { trad } { 溝 }
1457 \zhnum_set_digits_map:nnn { s9 } { simp } { 涧 }
1458 \zhnum_set_digits_map:nnn { s9 } { trad } { 澗 }
1459 </big5>
1460 <*big5>
1461 \zhnum_set_digits_map:nn { s8 } { 溝 }
1462 \zhnum_set_digits_map:nn { s9 } { 澗 }
1463 </big5>
1464 \zhnum_set_digits_map:nn { s10 } { 正 }
1465 <*/big5>
1466 \zhnum_set_digits_map:nnn { s11 } { simp } { 载 }
1467 \zhnum_set_digits_map:nnn { s11 } { trad } { 載 }
1468 </big5>
1469 <*big5>
1470 \zhnum_set_digits_map:nn { s11 } { 載 }
1471 </big5>
1472 \zhnum_set_digits_map:nn { year } { 年 }
1473 \zhnum_set_digits_map:nn { month } { 月 }
1474 \zhnum_set_digits_map:nn { day } { 日 }
1475 <*/big5>
1476 \zhnum_set_digits_map:nnn { hour } { simp } { 时 }
1477 \zhnum_set_digits_map:nnn { hour } { trad } { 時 }
1478 </big5>
1479 <*big5>
1480 \zhnum_set_digits_map:nn { hour } { 時 }
1481 </big5>
1482 \zhnum_set_digits_map:nn { minute } { 分 }
1483 \zhnum_set_digits_map:nn { weekday } { 星期 }
1484 \zhnum_set_financial_map:nn { null } { 零 }
1485 \zhnum_set_financial_map:nn { 0 } { 零 }
1486 \zhnum_set_financial_map:nn { 1 } { 壹 }
1487 <*/big5>
1488 \zhnum_set_financial_map:nnn { 2 } { simp } { 贰 }
1489 \zhnum_set_financial_map:nnn { 2 } { trad } { 貳 }
1490 \zhnum_set_financial_map:nnn { 3 } { simp } { 叁 }
1491 \zhnum_set_financial_map:nnn { 3 } { trad } { 叁 }

```

```

1492 </!big5>
1493 <*big5>
1494 \zhnum_set_financial_map:nn { 2 } { 貳 }
1495 \zhnum_set_financial_map:nn { 3 } { 參 }
1496 </big5>
1497 \zhnum_set_financial_map:nn { 4 } { 肆 }
1498 \zhnum_set_financial_map:nn { 5 } { 伍 }
1499 <*big5>
1500 \zhnum_set_financial_map:nnn { 6 } { simp } { 陆 }
1501 \zhnum_set_financial_map:nnn { 6 } { trad } { 陸 }
1502 </!big5>
1503 <*big5>
1504 \zhnum_set_financial_map:nn { 6 } { 陸 }
1505 </big5>
1506 \zhnum_set_financial_map:nn { 7 } { 柒 }
1507 \zhnum_set_financial_map:nn { 8 } { 捌 }
1508 \zhnum_set_financial_map:nn { 9 } { 玖 }
1509 \zhnum_set_financial_map:nn { 10 } { 拾 }
1510 \zhnum_set_financial_map:nn { 100 } { 佰 }
1511 \zhnum_set_financial_map:nn { 1000 } { 仟 }
1512 \zhnum_set_tiangang_map:nn { 1 } { 甲 }
1513 \zhnum_set_tiangang_map:nn { 2 } { 乙 }
1514 \zhnum_set_tiangang_map:nn { 3 } { 丙 }
1515 \zhnum_set_tiangang_map:nn { 4 } { 丁 }
1516 \zhnum_set_tiangang_map:nn { 5 } { 戊 }
1517 \zhnum_set_tiangang_map:nn { 6 } { 己 }
1518 \zhnum_set_tiangang_map:nn { 7 } { 庚 }
1519 \zhnum_set_tiangang_map:nn { 8 } { 辛 }
1520 \zhnum_set_tiangang_map:nn { 9 } { 壬 }
1521 \zhnum_set_tiangang_map:nn { 10 } { 癸 }
1522 \zhnum_set_dizhi_map:nn { 1 } { 子 }
1523 \zhnum_set_dizhi_map:nn { 2 } { 丑 }
1524 \zhnum_set_dizhi_map:nn { 3 } { 寅 }
1525 \zhnum_set_dizhi_map:nn { 4 } { 卯 }
1526 \zhnum_set_dizhi_map:nn { 5 } { 辰 }
1527 \zhnum_set_dizhi_map:nn { 6 } { 巳 }
1528 \zhnum_set_dizhi_map:nn { 7 } { 午 }
1529 \zhnum_set_dizhi_map:nn { 8 } { 未 }
1530 \zhnum_set_dizhi_map:nn { 9 } { 申 }
1531 \zhnum_set_dizhi_map:nn { 10 } { 酉 }
1532 \zhnum_set_dizhi_map:nn { 11 } { 戌 }
1533 \zhnum_set_dizhi_map:nn { 12 } { 亥 }
1534 </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

\??? 14
 \\ 5, 6, 7, 522

A

activechar 7
 arabicsep 6

B

bingint internal commands:
 _bingint_read_do:nn 11
 bool commands:
 _bool_const:Nn 406, 1201
 _bool_if:NTF 247, 261, 399,
 437, 444, 449, 484, 486, 526, 536, 554, 561, 571, 590,
 680, 1109, 1115, 1125, 1131, 1210, 1222, 1242, 1350, 1362
 _bool_lazy_all:nTF 287
 _bool_lazy_any_p:n 1203
 _bool_lazy_or:nnTF 504
 _bool_lazy_or_p:nn 408
 _bool_new:N
 ... 412, 414, 415, 557, 565, 1194, 1247, 1332, 1333, 1334
 _bool_set_false:N 543,
 1164, 1269, 1274, 1275, 1283, 1290, 1308, 1312, 1318, 1345
 _bool_set_true:N 413, 542,
 563, 1248, 1270, 1279, 1280, 1282, 1289, 1306, 1314, 1320
 _c_false_bool 14, 252, 253, 354
 _c_true_bool 237, 252, 253, 352
 \BooleanFalse 317

C

char commands:
 _char_generate:nn 15, 424, 426, 474, 476
 _char_set_catcode_active:n 1229
 _char_set_catcode_other:n 1234

clist commands:
 _clist_map_function:nN 705
 _clist_map_inline:Nn 835
 _clist_map_inline:nn 801, 810, 846

cs commands:
 _cs_generate_variant:Nn 59,
 221, 240, 256, 355, 405, 479, 697, 758, 760, 785, 1148
 _cs_if_exist:NTF 12, 829, 1389
 _cs_if_free:NTF 731
 _cs_new:Npn 14, 51, 53, 60, 66, 85, 93, 105, 121, 127, 132,
 145, 146, 155, 167, 177, 185, 187, 189, 199, 207, 209,
 211, 222, 233, 235, 241, 257, 313, 329, 335, 351, 353,
 357, 365, 377, 389, 394, 423, 425, 427, 433, 455, 457,
 459, 464, 466, 468, 473, 475, 477, 480, 495, 500, 511,
 529, 552, 573, 575, 581, 588, 594, 597, 603, 606, 618,

626, 632, 637, 650, 663, 667, 672, 678, 684, 689, 695,
 727, 736, 748, 761, 763, 765, 767, 770, 1022, 1038, 1040
 \cs_new_eq:NN 20, 21, 700, 759, 1021, 1212
 \cs_new_protected:Npe 1053, 1129, 1226, 1231
 \cs_new_protected:Npn
 16, 18, 22, 27, 533, 538, 549, 559, 709,
 716, 884, 886, 891, 893, 898, 900, 906, 920, 935, 951,
 959, 972, 993, 1008, 1030, 1036, 1067, 1069, 1078, 1102,
 1107, 1113, 1123, 1138, 1149, 1155, 1160, 1176, 1178,
 1185, 1190, 1213, 1220, 1236, 1250, 1343, 1353, 1366, 1375
 \cs_set:Npn 707, 833, 1046
 \cs_set_eq:NN 24, 25, 29, 30, 1296, 1298
 \cs_set_protected:Npe 1010

D

dim commands:
 _dim_eval:n 562
 dot 3

E

else commands:
 _else: 160, 172, 203, 370, 400
 encoding 1, 1257
 exp commands:
 _exp:w 17, 151, 159, 162, 171, 183, 362, 369, 387
 _exp_after:wN 148, 150, 158, 159,
 161, 163, 164, 170, 171, 173, 174, 182, 202, 204, 224,
 227, 228, 229, 359, 361, 368, 369, 371, 372, 373, 374, 386
 _exp_args:Nc 15, 110, 124, 318, 712
 _exp_args:Nf 208, 210, 1039
 _exp_args:Nnf 109
 _exp_args:Nnnf 317
 _exp_args:NNno 880
 _exp_args:No 929, 1168, 1177
 _exp_end_continue_f:w
 17, 151, 159, 162, 171, 183, 362, 369, 387
 _exp_not:N
 . 157, 169, 201, 367, 667, 668, 795, 796, 798, 806, 807,
 817, 821, 822, 823, 824, 825, 826, 827, 851, 859, 867,
 875, 955, 963, 968, 977, 978, 983, 984, 989, 990, 997,
 1003, 1004, 1005, 1024, 1026, 1027, 1049, 1050, 1051,
 1056, 1057, 1059, 1060, 1063, 1064, 1098, 1099, 1134, 1135
 _exp_not:n 19, 20,
 21, 29, 30, 431, 555, 956, 964, 969, 998, 1026, 1027, 1131
 _exp_stop_f: 152, 201, 226, 231, 363, 608

F

fi commands:
 _fi: 157, 165, 175, 205, 230, 367, 375, 402, 616
 file commands:
 _file_get_full_name:nNTF 1162

\file_input:n 1168
font 4

G

ganzhi-cyclic 6
group commands:
\group_begin: 45, 116, 324, 346, 540, 788, 1045, 1166
\group_end: 48, 119, 327, 349, 545, 880, 1066, 1170

H

hook commands:
\hook_gput_code:nnn 32, 34

I

if commands:
\if:w 157, 169, 367
\if_case:w 226, 608
\if_int_compare:w 201, 398
\ifnum 2

int commands:

\int_compare:nNnTF 134, 137, 191, 194, 213,
216, 244, 250, 259, 268, 270, 274, 281, 283, 304, 391,
435, 442, 620, 628, 634, 729, 738, 750, 753, 772, 775, 1041
\int_compare:nTF 296
\int_compare_p:nNn 289, 290
\int_div_truncate:nn 225, 642, 644, 645, 646, 655, 657, 675
\int_eval:n
..... 16, 135, 140, 188, 217, 254, 336, 424, 426, 461,
462, 470, 471, 635, 713, 755, 762, 764, 766, 768, 779, 797
\int_if_exist:NnTF 107, 123, 315, 331
\int_incr:N 711, 721
\int_mod:nn 210, 442, 639, 652, 676, 696, 744, 773, 780, 1039
\int_new:N 698, 725
\int_set:Nn 699
\int_step_function:nN 1061
\int_step_function:nnN 1228, 1233
\int_step_inline:nn 790, 855, 863, 871
\int_to_arabic:n 592, 682
\int_use:N 110, 318
\int_value:w 149, 225, 360, 584
\int_zero:N 704
\c_one_int 194, 289, 391, 729
\c_zero_int 134, 137, 153, 191, 216, 244, 250, 259, 268,
270, 281, 283, 290, 304, 398, 738, 750, 753, 772, 775, 1041

int internal commands:

__int_eval:w 16

K

kern 4
keys commands:
\keys_define:nn 881, 1257, 1302
\keys_set:nn 1329, 1385, 1387
\keys_set_exclude_groups:nnnN . 1346, 1355, 1368, 1377

M

minus 3
msg commands:
\msg_error:nn 11, 534, 550

\msg_error:nnn 1173, 1265
\msg_expandable_error:nnn 128, 578
\msg_new:nnn 3, 129, 520, 566
\msg_new:nnnn 1195, 1335

N

\NewDocumentCommand .. 12, 43, 114, 322, 344, 702, 1327, 1338
\NewExpandableDocumentCommand 36, 99, 307, 337, 568
null 6, 1257
\numexpr 2

O

or commands:

\or: 227, 228, 229, 610, 611, 612, 613, 614, 615

P

\pdfstringdefPreHook 33, 35
prg commands:

\prg_do_nothing: 1021, 1212
\ProcessKeyOptions 1389, 1390
\ProcessKeysOptions 1393

prop commands:

\prop_clear:N 1187
\prop_get:NnNTF 922, 927, 937, 940
\prop_gset_eq:NN 1191
\prop_if_exist:NnTF 1151
\prop_map_function:NN 916, 1032
\prop_new:N 902, 903, 904, 905, 1188
\prop_put:Nnn 885, 889, 892, 896, 899, 901
\prop_put_if_new:Nnn 888, 895
\prop_set_eq:NN 1158

Q

quark commands:

\q_mark 64, 66
\q_nil 12, 52, 56, 64, 238, 485, 491, 492
\quark_if_nil:nTF 55, 62, 68, 514
\quark_if_recursion_tail_stop:N 243, 382
\quark_if_recursion_tail_stop_do:Nn 181
\q_recursion_stop 152, 177, 183, 238, 363
\q_recursion_tail 12, 152, 238, 363
\q_stop 52, 53, 56,
60, 64, 66, 458, 459, 467, 468, 478, 480, 485, 491, 492,
511, 570, 571, 573, 575, 579, 604, 606, 664, 668, 833, 838

R

\RequirePackage 13, 1392
reset 7, 1257

S

scan commands:

\scan_stop: 562

skip commands:

\skip_horizontal:n 19

str commands:

\c_colon_str 668
\str_head:n 1004
\str_if_empty:NnTF 1395
\str_if_eq:NNTF 1141

`\str_if_eq:nnTF` 482, 490, 502, 509
`\str_lowercase:n` 1253
`\str_new:N` 1192, 1193, 1256
`\str_set:Nn` 1215, 1238, 1239, 1252
`\str_set_eq:NN` 1157, 1216
`\str_tail:n` 1005
`style` 6, 1257

sys commands:

`\sys_if_engine luatex_p:` 410, 1206
`\sys_if_engine uptex_p:` 1207
`\sys_if_engine xetex_p:` 409, 1205

T

TeX and L^AT_EX_{2_ε} commands:

`\@ifpackagelater` 10
`\@zhdig` 335
`\@zhnum` 145
`\changes` 14
`\CJKecglue` 4, 19
`\CJKfamily` 4
`\edef` 8, 10
`\g@addto@macro` 33, 35
`\int_use:N` 10
`\kern` 4, 19
`\mbox` 4, 19
`\outer` 25
`\pagenumbering` 4
`\pdfstringdef` 7, 8
`\pdfstringdefPreHook` 7
`\tableofcontents` 9
`\tex_expanded:D` 7, 8
`\thesection` 7
`\tiangan` 6
`\tl_set_eq:NN` 10
`\verb` 16
`\XeTeXcharclass` 4, 19
`\zhdate` 4
`\zhdig` 1, 4, 7, 14
`\zhdigits` 1, 2, 5–7, 14, 16, 17, 19, 32
`\zhdigwithoptions` 14
`\zhdizhi` 5
`\zhganzhi` 5, 25
`\zhganzhinian` 5
`\zhnum` 1, 4, 7, 9, 10, 13
`\zhnum_reset_style:` 10
`\zhnumber` 1, 5, 7, 8
`\zhnumberwithoptions` 10
`\zhnumExtendScaleMap` 5
`\zhnumsetup` 1–3, 5, 7, 16, 32
`\zhnumwithoptions` 9
`\zhrod` 2, 4, 16, 20
`\zhrodbox` 2, 4, 19, 20
`\zhrodsetup` 2, 16
`\zhtiangan` 5, 25
`\zhtime` 5
`\zhweekday` 4

tex commands:

`\tex_day:D` 586
`\tex_ignorespaces:D` 1330, 1341
`\tex_kern:D` 562
`\tex_month:D` 585
`\tex_time:D` 675, 676
`\tex_year:D` 584

time

tl commands:

`\c_space_tl` 522
`\tl_build_begin:N` 789
`\tl_build_end:N` 879
`\tl_build_get:NN` 25
`\tl_build_get_intermediate:NN` 829, 830
`\tl_build_put_right:Nn`
 793, 804, 816, 819, 837, 840, 849, 857, 865, 873
`\tl_clear_new:N` ... 908, 909, 910, 911, 912, 913, 914, 915
`\tl_concat:NNN` 1002, 1048, 1082, 1086, 1090
`\tl_const:Nn` 558, 1095
`\tl_count:n` 17, 208, 456
`\tl_if_blank:nTF` 75, 89, 95, 458, 467, 478, 497, 577
`\tl_if_blank_p:n` 506
`\tl_if_empty:N` 1348, 1357, 1372, 1384
`\tl_if_empty:nTF` 599, 600, 601
`\tl_if_eq:NNTF` 1360, 1370, 1379
`\tl_if_exist:N` 691, 718, 740
`\tl_if_no_value:nTF` 38, 101, 309, 339, 706
`\tl_map_function:nN` 497
`\tl_new:N` 416, 417, 419, 421, 701, 720, 786,
 787, 792, 803, 948, 949, 950, 1042, 1043, 1044, 1101, 1175
`\tl_put_right:Nn` . 953, 961, 966, 974, 980, 986, 995, 1000
`\tl_set:Nn`
 .. 418, 420, 422, 723, 942, 955, 963, 968, 997, 1024, 1037
`\tl_set_eq:NN` 976, 982, 988, 1055, 1058, 1062, 1097, 1133
`\tl_use:N` 541, 1080, 1081, 1110, 1111, 1117, 1121, 1126, 1127

tl internal commands:

`\__tl_act:NNNnn` 11

token commands:

`\token_if_eq_charcode:NNTF` 383
`\token_to_str:N` 522
`\TrimSpaces` 702

U

`units` 3

use commands:

`\use:N` 25, 831
`\use:n` 17, 24, 25, 151, 159, 171, 183, 227, 362, 369, 387, 665
`\use_i:nn` 202, 245
`\use_i:nnn` 229
`\use_i_ii:nnn` 228
`\use_ii:nn` 204

Z

`zero` 3
`zerochar` 3
`\zhcurrtime` 5, 671
`\zhdate` 4, 566

- \zhdig [4](#), [7](#), [307](#)
- \zhdigits [2](#), [7](#), [337](#)
- \zhdigitsoptions [317](#), [337](#)
- \zhdigwithoptions [322](#)
- \zhdizhi [5](#), [763](#)
- \zhganzhi [5](#), [765](#)
- \zhganzhinian [5](#), [767](#)
- \zhnum [4](#), [7](#), [99](#)
- zhnum commands:
 - \zhnum_blank_to_zero:n [70](#), [72](#), [80](#), [82](#), [87](#), [93](#), [93](#)
 - \zhnum_check_financial:nn [906](#)
 - \zhnum_check_simp:nn [906](#), [916](#), [920](#)
 - \zhnum_counter:n [102](#), [118](#), [121](#), [121](#)
 - \zhnum_decimal:nn [57](#), [85](#), [85](#)
 - \zhnum_digits:Nn .. [17](#), [340](#), [348](#), [352](#), [354](#), [356](#), [357](#), [405](#)
 - \zhnum_digits_counter:n [14](#), [310](#), [326](#), [329](#)
 - \zhnum_digits_null:n [332](#), [336](#), [351](#), [353](#), [355](#), [591](#)
 - \zhnum_digits_zero:n [91](#), [351](#), [351](#)
 - \zhnum_exp_not:n . [15](#), [20](#), [21](#), [24](#), [25](#), [29](#), [30](#), [438](#), [516](#), [595](#)
 - \zhnum_expand_wrap:wn [14](#),
[16](#), [50](#), [131](#), [356](#), [528](#), [596](#), [605](#), [662](#), [671](#), [726](#), [735](#), [769](#)
 - \zhnum_ganzhi:nnn [759](#), [759](#), [760](#), [762](#), [764](#), [766](#), [1296](#), [1298](#)
 - \zhnum_ganzhi_cyclic:nnn [735](#), [736](#), [1296](#)
 - \zhnum_ganzhi_nian:n [768](#), [769](#), [770](#), [785](#)
 - \zhnum_ganzhi_normal:nnn [726](#), [727](#), [759](#), [1298](#)
 - \zhnum_if_digit:NTF [179](#), [199](#), [199](#), [379](#), [429](#)
 - \zhnum_input_cfg:n [1153](#), [1160](#)
 - \zhnum_int:n [121](#), [124](#), [132](#), [145](#), [591](#), [681](#)
 - \zhnum_integer:n [15](#), [17](#), [63](#), [146](#), [146](#)
 - \zhnum_load_cfg:n [1068](#), [1138](#), [1138](#), [1148](#), [1254](#)
 - \zhnum_number:n [39](#), [47](#), [50](#), [51](#), [59](#), [77](#), [97](#)
 - \zhnum_output:n [14](#), [14](#), [71](#), [78](#),
[81](#), [88](#), [90](#), [96](#), [139](#), [142](#), [192](#), [195](#), [214](#), [247](#), [262](#), [265](#),
[266](#), [271](#), [275](#), [277](#), [278](#), [284](#), [294](#), [297](#), [299](#), [300](#), [305](#),
[384](#), [392](#), [396](#), [599](#), [600](#), [601](#), [609](#), [610](#), [611](#), [612](#), [613](#),
[614](#), [615](#), [686](#), [687](#), [692](#), [732](#), [741](#), [751](#), [754](#), [755](#), [773](#), [777](#)
 - \zhnum_parse_config: [906](#), [906](#), [1145](#)
 - \zhnum_parse_number:n [135](#), [140](#), [207](#), [207](#), [221](#)
 - \zhnum_parse_number:nn [196](#), [207](#), [208](#), [209](#)
 - \zhnum_process_number:NNNNNN [248](#), [257](#), [257](#)
 - \zhnum_reset_all: [918](#), [1067](#), [1069](#), [1142](#)
 - \zhnum_reset_config: [1067](#), [1067](#), [1210](#), [1351](#)
 - \zhnum_reset_ganzhi: [1030](#), [1053](#), [1075](#), [1382](#)
 - \zhnum_reset_main: [1071](#), [1078](#)
 - \zhnum_reset_normal: [1076](#), [1105](#), [1113](#), [1371](#)
 - \zhnum_reset_simp: [1072](#), [1104](#), [1107](#)
 - \zhnum_reset_style: [1067](#), [1102](#), [1358](#), [1361](#), [1397](#)
 - \zhnum_reset_week_day: .. [1008](#), [1010](#), [1021](#), [1074](#), [1381](#)
 - \zhnum_scale_map:n [249](#), [689](#), [689](#), [696](#), [697](#)
 - \zhnum_scale_map_hook:n [693](#), [700](#), [707](#)
 - \zhnum_scale_map_loop:n [689](#), [695](#), [700](#)
 - \zhnum_set:n [46](#), [117](#), [325](#), [347](#), [1340](#), [1343](#)
 - \zhnum_set_active: [1223](#), [1226](#)
 - \zhnum_set_catcode: [1167](#), [1210](#), [1212](#), [1220](#)
 - \zhnum_set_cfg_name:Nn [1140](#), [1210](#), [1213](#), [1236](#)
 - \zhnum_set_digits_map:nn [884](#),
[884](#), [1405](#), [1407](#), [1409](#), [1412](#), [1414](#), [1415](#), [1416](#), [1417](#),
[1418](#), [1419](#), [1420](#), [1421](#), [1422](#), [1423](#), [1424](#), [1425](#), [1435](#),
[1437](#), [1438](#), [1446](#), [1447](#), [1449](#), [1450](#), [1451](#), [1452](#), [1453](#),
[1461](#), [1462](#), [1464](#), [1470](#), [1472](#), [1473](#), [1474](#), [1480](#), [1482](#), [1483](#)
 - \zhnum_set_digits_map:nnn [16](#), [884](#), [886](#), [1401](#),
[1402](#), [1426](#), [1427](#), [1428](#), [1429](#), [1431](#), [1432](#), [1440](#), [1441](#),
[1442](#), [1443](#), [1455](#), [1456](#), [1457](#), [1458](#), [1466](#), [1467](#), [1476](#), [1477](#)
 - \zhnum_set_dizhi_map:nn [884](#), [900](#), [1522](#), [1523](#),
[1524](#), [1525](#), [1526](#), [1527](#), [1528](#), [1529](#), [1530](#), [1531](#), [1532](#), [1533](#)
 - \zhnum_set_encoding:n [1250](#), [1250](#), [1261](#), [1396](#)
 - \zhnum_set_financial_map:nn
..... [884](#), [891](#), [1484](#), [1485](#), [1486](#), [1494](#),
[1495](#), [1497](#), [1498](#), [1504](#), [1506](#), [1507](#), [1508](#), [1509](#), [1510](#), [1511](#)
 - \zhnum_set_financial_map:nnn
..... [884](#), [893](#), [1488](#), [1489](#), [1490](#), [1491](#), [1500](#), [1501](#)
 - \zhnum_set_ganzhi: [917](#), [1030](#), [1030](#)
 - \zhnum_set_other: [1224](#), [1231](#)
 - \zhnum_set_scale:n [705](#), [709](#), [709](#)
 - \zhnum_set_tiangang_map:nn [884](#), [898](#),
[1512](#), [1513](#), [1514](#), [1515](#), [1516](#), [1517](#), [1518](#), [1519](#), [1520](#), [1521](#)
 - \zhnum_set_week_day: [906](#), [1008](#), [1008](#), [1073](#)
 - \zhnum_split_number:nn [217](#), [234](#), [235](#), [235](#), [240](#)
 - \zhnum_split_number:NNNNNNw
..... [237](#), [241](#), [241](#), [252](#), [253](#), [256](#)
 - \zhnum_two_digits:n [618](#), [621](#), [632](#)
 - \zhnum_update_cfg:n [1144](#), [1149](#)
 - \zhnum_Zeller:nnn [608](#), [618](#), [618](#)
 - \zhnum_Zeller_aux:Nnnn [618](#)
 - \zhnum_Zeller_Gregorian:nnn [622](#), [637](#), [637](#)
 - \zhnum_Zeller_Julian:nnn [623](#), [650](#), [650](#)
 - \zhnum_zero_mod:nn [1038](#), [1050](#), [1051](#)
 - zhnum internal commands:
 - \l_zhnum_active_char_bool [1222](#), [1242](#), [1247](#), [1248](#), [1293](#)
 - _zhnum_add_reset:nn [946](#), [951](#), [1034](#)
 - _zhnum_add_reset_ancient:nN [924](#), [993](#)
 - _zhnum_add_reset_financial:n [931](#), [972](#)
 - _zhnum_add_reset_simp:nNN [943](#), [959](#)
 - \l_zhnum_ancient_bool .. [1125](#), [1269](#), [1274](#), [1279](#), [1334](#)
 - \l_zhnum_ancient_tl [923](#), [924](#), [950](#)
 - \l_zhnum_arabic_sep_tl [592](#), [682](#), [1300](#)
 - \l_zhnum_cfg_file_tl [1162](#), [1168](#), [1175](#)
 - \l_zhnum_cfg_map_finan_prop [884](#), [927](#), [1182](#)
 - \l_zhnum_cfg_map_ganzhi_prop [884](#), [1033](#), [1183](#)
 - \l_zhnum_cfg_map_prop [884](#), [916](#), [1180](#)
 - \l_zhnum_cfg_map_var_prop ... [884](#), [922](#), [937](#), [940](#), [1181](#)
 - \l_zhnum_cfg_str .. [1140](#), [1141](#), [1151](#), [1157](#), [1177](#), [1192](#)
 - _zhnum_check_simp_aux:nn [926](#), [929](#), [935](#)
 - _zhnum_counter_error:n [112](#), [125](#), [127](#), [320](#), [333](#)
 - _zhnum_counter_with_options:nn [103](#), [105](#), [105](#)
 - _zhnum_date:www [20](#), [570](#), [573](#)
 - _zhnum_date_aux:nnn [574](#), [583](#), [588](#), [588](#)
 - _zhnum_date_aux:Nnnnn [592](#), [594](#)
 - _zhnum_date_aux:NNnnnn [591](#), [595](#), [597](#)
 - _zhnum_date_star:www [571](#), [575](#)

```

\__zhnum_digits_counter_with_options:nn ..... 311, 313, 313
\l__zhnum_dizhi_#1_tl ..... 25
\l__zhnum_encoding_str .....
... 1068, 1215, 1216, 1238, 1241, 1252, 1254, 1256, 1395
\__zhnum_expand_wrap_aux:nn ..... 17, 18
\__zhnum_fraction:www ..... 64, 66, 66
\l__zhnum_ganzhi_#1_tl ..... 25
\__zhnum_ganzhi_cyclic_mod:nnnn .. 735, 743, 748, 758
\__zhnum_integer_or_fraction:www ..... 56, 60, 60
\l__zhnum_kv_tl ..... 786, 789, 793, 804, 816,
819, 832, 837, 840, 849, 857, 865, 873, 879, 883, 1347,
1348, 1356, 1360, 1369, 1370, 1372, 1378, 1379, 1384, 1385
\l__zhnum_last_cfg_str ..... 1141, 1157, 1193
\__zhnum_loop_end:wnn ..... 181, 187
\l__zhnum_minus_tl ..... 821
\l__zhnum_normal_bool ... 1115, 1270, 1275, 1280, 1333
\l__zhnum_null_bool ..... 1131, 1286
\l__zhnum_null_tl ..... 822, 1135
\__zhnum_number:www ..... 50, 52, 53
\__zhnum_output:nnwnn ..... 180, 185
\__zhnum_output_digits:NN ..... 380, 394
\__zhnum_parse_number:nnn ..... 210, 211
\__zhnum_prop_gset_eq:Nn ..... 1169, 1190
\__zhnum_prop_initial:Nn ..... 1165, 1185
\__zhnum_read_abs_loop:Nw ..... 173, 177, 177, 182
\__zhnum_read_digits:w ..... 359, 389
\__zhnum_read_digits_loop:NN ..... 17, 372, 377, 386
\__zhnum_read_integer:www ..... 148, 189, 189
\__zhnum_read_sign_loop:N ..... 150, 155, 158
\__zhnum_read_sign_loop:NN ..... 361, 365, 368
\__zhnum_read_zeros_loop:N ..... 163, 167, 170
\__zhnum_reset_ancient: ..... 1118, 1123
\l__zhnum_reset_ancient_tl ..... 913, 995, 1126
\l__zhnum_reset_bool 1164, 1194, 1292, 1345, 1350, 1362
\l__zhnum_reset_financial_tl .....
..... 915, 986, 1087, 1088, 1091, 1092, 1121
\__zhnum_reset_ganzhi:nn ..... 1036
\l__zhnum_reset_normal_tl . 914, 980, 1083, 1084, 1117
\l__zhnum_reset_simp_tl ..... 909, 961, 1110
\l__zhnum_reset_tl ..... 908, 953, 1080
\l__zhnum_reset_trad_tl ..... 910, 966, 1111
\__zhnum_reset_zero: ..... 1119, 1129
\__zhnum_result:nn ..... 153, 185, 186, 187
\__zhnum_rod_base_mark:n ..... 473, 497, 500
\__zhnum_rod_digit:nn ..... 430, 433
\l__zhnum_rod_dot_tl ..... 412, 516, 1325
\c__zhnum_rod_engine_bool ..... 406, 526, 536
\l__zhnum_rod_font_tl ..... 412, 541, 1322
\__zhnum_rod_frac:n ..... 464, 464, 517
\__zhnum_rod_frac_head:wn ..... 464, 467, 468
\__zhnum_rod_frac_loop:nn ..... 464, 465, 466, 471
\__zhnum_rod_glyph:nn ..... 17, 427, 427, 461, 470
\__zhnum_rod_horizontal:n ..... 423, 425, 446, 450
\__zhnum_rod_int:n ..... 455, 455, 513
\__zhnum_rod_int_head:wn ..... 455, 458, 459
\__zhnum_rod_int_loop:nn ..... 455, 456, 457, 462
\__zhnum_rod_kern: ..... 441, 552, 552
\__zhnum_rod_kern_node: ..... 20, 552, 558, 559
\c__zhnum_rod_kern_node_tl ..... 555, 558
\l__zhnum_rod_kern_seen_bool ..... 543, 561, 563, 565
\l__zhnum_rod_kern_tl ..... 412, 562, 1323
\l__zhnum_rod_kerning_bool ..... 20, 542, 554, 557
\l__zhnum_rod_minus_overlay_bool .....
..... 412, 484, 486, 1318, 1320
\__zhnum_rod_number:n ..... 19, 473, 477, 479, 530, 544
\__zhnum_rod_overlay: ..... 475, 498
\__zhnum_rod_overlay_if_base:n .... 18, 473, 487, 495
\__zhnum_rod_sign:w ..... 473, 478, 480
\__zhnum_rod_slash: ..... 473, 484
\__zhnum_rod_split:www ..... 473, 485, 491, 492, 511
\l__zhnum_rod_units_vertical_bool .....
..... 412, 444, 449, 1306, 1308
\__zhnum_rod_vertical:n ..... 423, 423, 445, 451
\l__zhnum_rod_zero_omit_bool 412, 437, 505, 1312, 1314
\l__zhnum_rod_zero_tl ..... 16, 412, 438, 506, 1324
\l__zhnum_scale_int ..... 696, 698, 699, 721
\l__zhnum_set_ancient_tl ..... 911, 1000, 1093, 1127
\__zhnum_set_cfg_prop: ..... 1152, 1155, 1171
\l__zhnum_set_normal_tl ..... 912, 974, 1081
\__zhnum_set_pre_pos: ..... 1373, 1375
\__zhnum_set_scale:Nn ..... 712, 716
\__zhnum_set_style: ..... 1349, 1353
\__zhnum_set_user: ..... 1363, 1366
\__zhnum_set_week_day:nn .....
..... 1012, 1013, 1014, 1015, 1016, 1017, 1018, 1022
\c__zhnum_set_zero_tl ..... 1085, 1089, 1095
\l__zhnum_simp_bool ..... 1109, 1282, 1283, 1332
\l__zhnum_simp_tl ..... 941, 942, 944, 948
\__zhnum_split_number_aux:nnn ..... 218, 222, 222
\__zhnum_split_number_aux:wn ..... 224, 233
\l__zhnum_tiangang_#1_tl ..... 25
\__zhnum_time:ww ..... 664, 667
\__zhnum_time_aux:nn ..... 670, 674, 678, 678
\__zhnum_time_aux:Nnnn ..... 678, 681, 682, 684
\l__zhnum_time_bool ..... 590, 680, 1289, 1290
\__zhnum_tmp:w ..... 833, 838, 1046, 1061
\l__zhnum_tmp_int ..... 704, 711, 713, 725
\l__zhnum_tmp_tl ..... 787,
832, 835, 927, 930, 1356, 1357, 1360, 1369, 1370, 1378, 1379
\l__zhnum_trad_tl ..... 938, 944, 949
\c__zhnum_unicode_engine_bool ..... 16, 1201, 1210
\__zhnum_update_cfg_prop:N .... 1158, 1165, 1169, 1176
\__zhnum_update_cfg_prop_aux:nN ..... 1177, 1178
\__zhnum_week_day:www ..... 579, 604, 605, 606
\l__zhnum_weekday_tl ..... 843, 1026
\__zhnum_Zeller_aux:Nnnn ..... 622, 623, 626
\__zhnum_zero_mod_aux:nn ..... 1039, 1040
\l__zhnum_zero_tl ..... 1098, 1101
\zhnumber ..... 1, 7, 36
\zhnumberwithoptions ..... 40, 43, 109
\zhnumClearWrapper ..... 22, 33, 35

```

\zhnumExtendScaleMap	5, 702	\zhrodsetup	2, 1327
\zhnumResetWrapper	27	\zhtiangan	5, 761
\zhnumsetup	5, 1338	\zhtime	5, 662
\zhnumwithoptions	114	\zhtoday	5, 581
\zhrod	2, 520	\zhweekday	4, 603
\zhrodbox	2, 536		