

The ionumbers package*

Christian Schneider
<software(at)chschneider(dot)eu>

November 2, 2008

Warning: This is alpha software and may contain serious bugs!

Contents

1	Conflicts with other packages	2
2	Details of number handling	2
2.1	General rules	2
2.2	Caveats	3
3	Usage	3
3.1	Package options concerning the separators in the input	3
3.2	Package options concerning the separators in the output	4
3.3	Package options concerning automatic grouping	4
3.4	Local style changes	5
3.5	User-defined values for output separators	5
3.6	Enabling and disabling features	5
4	License	5
5	Acknowledgements	6
6	Bugs, problems, and suggestions	6
7	Implementation	6
7.1	Default/global configuration	6
7.2	Local style changes	7
7.3	User-defined values for output separators	8
7.4	Internal macros holding definitions for $\langle key \rangle = \langle value \rangle$ pairs	9
7.5	Enabling and disabling features	10
7.6	Definitions of active characters	11
7.7	Test for conflicts with other packages	17
7.8	Commands for current number	19

*This document corresponds to ionumbers v0.2.1-alpha, dated 2008/11/02. Copyright 2007-2008 Christian Schneider <software(at)chschneider(dot)eu>.

Abstract

`ionumbers` stands for ‘input/output numbers’.

This package restyles numbers in math mode. If a number in the input file is written, e.g., as `$3,231.44$` as commonly used in English texts, this package is able to restyle it to be output as ‘3 231,44’ as commonly used in German texts (and vice versa). This may be very useful, if you have a large table and want to include it in texts with different output conventions without the need of changing the table.

Furthermore this package can automatically group digits left to the decimal separator (*thousands*) and right to the decimal separator (*thousandths*) in triplets without the need of specifying commas (English) or points (German) as separators. E.g., the input `1234.567890` can be output as ‘1 234. 567 890’.

Finally, an `e` starts the exponent of the number. For example, `$21e6$` may be output as ‘ 26×10^6 ’.

1 Conflicts with other packages

This package potentially conflicts with any other package that defines a macro for any of the following characters: `.,+-0123456789`

There are tests for these cases and warning or error messages may be output. Please load `ionumbers` as last package to be able to detect as many conflicts as possible. As there is no way to detect conflicts in all cases, please report any package known to conflict with `ionumbers` to the author.

2 Details of number handling

2.1 General rules

Every input *in math mode* consisting of the following characters is treated by this package: `.,+-0123456789` These characters get macro definitions. A number is any combination of these characters without anything—not even white spaces—in between them. There are two exceptions/special cases:

1. The separator characters `.` and `,` are not treated as part of the number at its end. This avoids problems with lists like `1, 2, 3, \ldots`, where the commas are not part of in the numbers. Note, however, that the commas are treated as part of the numbers in the first two appearances in `1,2,3,\ldots`, as the commas are immediately followed by a digit. Please input lists with white spaces after the separators as shown in the first case.
2. The sign characters `+` and `-` will only be considered as part of the number, if they appear at the begining of a number.

The lower case letter `e` plays a special role. An `e` following immediatly a number (without any white space or other input in between) can be configured as begining of the exponential part of a number. The letter is eaten from the input in this case and substituted by some configurable output. (It is virtually impossible to handle it the same way as the other and assigning a macro definition to `e`.)

2.2 Caveats

Please be aware that the first decimal separator of a number marks the beginning of the thousandths part of a number; every part of a number appearing left to the first decimal separator is the thousands part. That is why, the input `$1.234.567$` with (only) the package option `autothousandths=true` (‘.’ is the thousands separator; option will be explained later) will lead to ‘1.234.567’ in the output. Note the small space after the second point as a result of 234.567 being treated as thousandths part. The thousandths separator—by default a small space—will be output between the third and fourth digit of the thousandths part; the additional point from the input will not be omitted. The input is syntactically incorrect (there must not be two decimal separators in one number) and the output is *not* a bug of this package.

A number following an `e` treated as exponential part of a number may be typeset as superscript (depending on the configuration). In the case of an exponential part there *may* be arbitrary input between `e` and the number in the exponent. Especially, the input `$1e \Pi^2` with package option `exponent=timestento` (will be explained later) leads to a superscript 2 in the output: ‘ $1 \times 10 \Pi^2$ ’. Again, the input is syntactically incorrect and there is no easy way to circumvent `e` from being treated as beginning of the exponential part in this case (at least, I did not find it).

3 Usage

Package options are used to globally configure a default behaviour of `ionumbers` for the whole document. These options usually consist of a $\langle key \rangle = \langle value \rangle$ pair. Local changes from this global configuration for arbitrary parts of the document can be applied with special commands.

3.1 Package options concerning the separators in the input

The following options configure the meaning of separators in the \LaTeX input file:

`comma`= $\langle value \rangle$ comma ‘,’ will be treated as $\langle value \rangle$
`point`= $\langle value \rangle$ point ‘.’ will be treated as $\langle value \rangle$

The following $\langle value \rangle$ s can be chosen for both of them:

<code>ignore</code>	the separator will be ignored (no output)
<code>decimal</code>	decimal separator (separating the thousands from the thousandths part of a number)
<code>thousands</code>	thousands separator (used for grouping of thousands part)
<code>default</code>	default behaviour of <code>ionumbers</code> (<code>decimal</code> for <code>point</code> ; <code>thousands</code> for <code>comma</code>)

The separator for exponents is always the lowercase letter `e`. A thousandths separator does not exist in input files; such a separator will only be output, if automatic grouping of the thousandths part is enabled (see below).

3.2 Package options concerning the separators in the output

The previously described options assign a *meaning* to separators in the input file. The *output* of the *meanings* is configured via the following options:

thousands=*<value>* thousands separator will be output as *<value>*
decimal=*<value>* decimal separator will be output as *<value>*
thousandths=*<value>* thousandths separator will be *<value>*
exponent=*<value>* exponent separator will be output as *<value>*

The list of valid *<value>*s for **thousands**, **decimal**, and **thousandths** is:

none will be ignored (no output)
point normal point; this is the default point without ionumbers
comma normal comma
punctpoint punctuation point (point followed by small space)
punctcomma punctuation comma (point followed by small space); this is the default comma without ionumbers
apostrophe apostrophe (actually $\text{\textasciitilde\prime}$; *not* for decimal)
phantom space with width of a point (\textasciitilde; *not* for decimal)
space small space (\textasciitilde\, ; *not* for decimal)
default default behaviour of ionumbers (**punctcomma** for thousands; **point** for decimal; **space** for thousandths)

The list of valid *<value>*s for **exponent** is:

none will be ignored (no output)
ite/itE italic lower/upper case letter ‘e’
rme/rmE roman lower/upper case letter ‘e’
timestento $\text{\textasciitilde\times10}$, \$ with following number output as superscript
cdottento $\text{\textasciitilde\cdot10}$, \$ with following number output as superscript
wedge $\text{\textasciitilde\wedge}$
default default behaviour of ionumbers (**ite**)

3.3 Package options concerning automatic grouping

Automatic grouping is a feature that automatically adds the thousands and thousandths separator, respectively. The separator will be added after each triplet of digits. Automatic grouping can be enable or disabled with the following options:

autothousands=*<value>* automatic grouping of thousands (digits left to decimal separator)
autothousandths=*<value>* automatic grouping of thousandths (digits right to decimal separator)

The available *<value>*s are **true** and **false** (default).

Notes on automatic grouping:

1. Grouping of thousandths requires **autothousandths=true** in any case, as there is no thousandths separator for explicitly specifying separations in the input.
2. Automatic grouping of thousands will be skipped in a number, if it contains a thousands separator in the input.

3.4 Local style changes

`\ionumbersstyle` The command `\ionumbersstyle{⟨option list⟩}` changes the global style definitions as specified as package options for the rest of the group. The `⟨option list⟩` may contain any of the package options described in sections 3.1–3.3. An additional `⟨value⟩` for all `⟨key⟩`s is available inside `\ionumbersstyle` to switch back to the configuration specified as package options: `reset`.

`\ionumbersresetstyle` The command `\ionumbersresetstyle` resets all `⟨value⟩`s to the configuration specified as package options. Actually, it is only a shorthand for `\ionumbersstyle{comma=reset,point=reset,decimal=reset,...}`.

3.5 User-defined values for output separators

A user may specify further output separators. Any user-defined `⟨value⟩`s for `thousands`, `decimal`, `thousandths`, and `exponent` can be used like the built-in options in section 3.2.

`\newionumbersthousands` The command `\newionumbersthousands{⟨value⟩}{⟨definition⟩}` has two mandatory arguments. The first one is the name of the newly defined `⟨value⟩` for the `thousands` `⟨key⟩` and the second one its definition. The commands `\newionumbersdecimal`, `\newionumbersthousandths`, and `\newionumbersexponent` work the same way for the `decimal`, `thousandths`, and `exponent` `⟨key⟩`, respectively. There is a starred version of `\newionumbersexponent` (called `\newionumbersexponent*`) that typesets the following number as superscript.

`\renewionumbersthousands` To redefine an existing `⟨key⟩` definition there are `\renew...` versions of the previously described commands.

`\renewionumbersdecimal` Notes on definitions:

1. All `⟨definition⟩`s are set inside `\ionumbersoff` (see section 3.6). This means that numbers appearing in the `⟨definition⟩`s are not treated by this package.
2. The value `curr` has an internal meaning and should *not* be defined/redefined by the user.

3.6 Enabling and disabling features

`\ionumbers` The command `\ionumbers` makes comma, point, signs, and digits active in math mode. This is equivalent to enabling the features of this package. This command applies to the end of the current group.

`\endionumbers` To disable the features by making comma, point, signs, and digits inactive again the command `\endionumbers` can be used. This command applies to the end of the current group.

`\ionumbersoff` The command `\ionumbersoff{⟨stuff⟩}` disables the features only for `⟨stuff⟩`.

4 License

`ionumbers` is free software: you can redistribute it and/or modify it under the terms of the GNU General Public License version 3 as published by the Free Software Foundation, not any later version.

`ionumbers` is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY

or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with ionumbers. If not, see <<http://www.gnu.org/licenses/>>.

5 Acknowledgements

The idea and parts of this package are based on `ziffer.sty` v2.1 by Martin V  th <vaeth@mathematik.uni-wuerzburg.de>.

Furthermore the `\l@addto@macro` (with changed name) from `koma-script bundle` v2.9t by Markus Kohm and Frank Neukam is used in this package.

Thanks to Martin V  th and Markus Kohm for permitting to use their code in this package.

6 Bugs, problems, and suggestions

Please report bugs and problems or send suggestions for this package to Christian Schneider. Check for updates before reporting bugs at the website mentioned above. Do *not* bother Martin V  th, Markus Kohm, or Frank Neukam with bugs, problems or suggestions concerning this package!

7 Implementation

The implementation is briefly described in this section. First of all, we need the `keyval` package for `<key>=<value>` options:

```
1 \RequirePackage{keyval}
```

7.1 Default/global configuration

In principle the definitions of all available `<key>=<value>` pairs is contained in the internal macros `\ion@<key>@<value>`. Setting a package option `<key>=<value>` defines `\ion@<key>@reset` to be `\ion@<key>@<value>`.

The following `ifs` will be required to remember, if automatic grouping is enabled.

```
2 \newif\ifion@autothousands
3 \newif\ifion@autothousandths
```

These shorthands are used to define the `<key>`s for package options and set their `<value>`s using `keyval`, respectively.

```
4 \newcommand*\ion@defpackopts{\define@key{ion@packopts}}
5 \newcommand*\ion@setpackopts{\setkeys{ion@packopts}}
```

Next the `<key>`s are defined.

```
6 \ion@defpackopts{comma}{%
7   \def\ion@comma@reset{\csname ion@comma@#1\endcsname}%
8   \def\ion@aftercomma@reset{\csname ion@aftercomma@#1\endcsname}}
9 \ion@defpackopts{point}{%
10  \def\ion@point@reset{\csname ion@point@#1\endcsname}%
```

```

11 \def\ion@afterpoint@reset{\csname ion@afterpoint@#1\endcsname}}
12 \ion@defpackopts{decimal}{\def\ion@decimal@reset{%
13 \csname ion@decimal@#1\endcsname}}
14 \ion@defpackopts{thousands}{\def\ion@thousands@reset{%
15 \csname ion@thousands@#1\endcsname}}
16 \ion@defpackopts{thousandths}{\def\ion@thousandths@reset{%
17 \csname ion@thousandths@#1\endcsname}}
18 \ion@defpackopts{exponent}{\def\ion@exponent@reset{%
19 \csname ion@exponent@#1\endcsname}}
20 \ion@defpackopts{autothousands}[true]{\def\ion@autothousands@reset{%
21 \csname ion@autothousands@#1\endcsname}\ion@autothousands@reset}
22 \ion@defpackopts{autothousandths}[true]{\def\ion@autothousandths@reset{%
23 \csname ion@autothousandths@#1\endcsname}\ion@autothousandths@reset}

Finally, the default ⟨value⟩s are set and—if specified by the user as package
option—overwritten with the user’s configuration.

24 \ion@setpackopts{comma=default,point=default,thousands=default,%
25 decimal=default,thousandths=default,exponent=default,autothousands=false,%
26 autothousandths=false}
27 \DeclareOption*{\expandafter\ion@setpackopts\expandafter{\CurrentOption}}
28 \ProcessOptions\relax

```

7.2 Local style changes

The currently active configuration of a *⟨key⟩* is stored in the macro `\ion@⟨key⟩@curr`. The `\ion@⟨key⟩@curr` macros for all *⟨key⟩*s are defined using the mechanism for local configuration changes.

The local options are defined and set—analogueous to the package option case—with two shorthands using `keyval`. The latter is publically available to the user.

```

29 \newcommand*\ion@deflocopts{\define@key{ion@locopts}}

```

`\ionnumberstyle`

```

30 \newcommand*\ionnumbersstyle[1]{\setkeys{ion@locopts}{#1}}

```

Now the *⟨key⟩*s for the local options are defined (just as in the case of the package options):

```

31 \ion@deflocopts{comma}{%
32 \def\ion@comma@curr{\csname ion@comma@#1\endcsname}%
33 \def\ion@aftercomma@curr{\csname ion@aftercomma@#1\endcsname}}
34 \ion@deflocopts{point}{%
35 \def\ion@point@curr{\csname ion@point@#1\endcsname}%
36 \def\ion@afterpoint@curr{\csname ion@afterpoint@#1\endcsname}}
37 \ion@deflocopts{decimal}{\def\ion@decimal@curr{%
38 \csname ion@decimal@#1\endcsname}}
39 \ion@deflocopts{thousands}{\def\ion@thousands@curr{%
40 \csname ion@thousands@#1\endcsname}}
41 \ion@deflocopts{thousandths}{\def\ion@thousandths@curr{%
42 \csname ion@thousandths@#1\endcsname}}
43 \ion@deflocopts{exponent}{\def\ion@exponent@curr{%
44 \csname ion@exponent@#1\endcsname}}
45 \ion@deflocopts{autothousands}[true]{\csname ion@autothousands@#1\endcsname}
46 \ion@deflocopts{autothousandths}[true]{\csname ion@autothousandths@#1\endcsname}

```

Finally, the command for resetting all $\langle key \rangle$ s is defined.

```
\ionumbersresetstyle
47 \newcommand*\ionumbersresetstyle{%
48   \ionumbersstyle{comma=reset,point=reset,thousands=reset,%
49     decimal=reset,thousandths=reset,exponent=reset,autothousands=reset,%
50     autothousandths=reset}}
```

This command is issued at the end of the package to make the configuration of the package options active (and have no undefined $\backslash ion@ \langle key \rangle @ curr$ macros).

```
51 \AtEndOfPackage{\ionumbersresetstyle}
```

7.3 User-defined values for output separators

The commands for user-defined $\langle value \rangle$ s for output separators just (re)define the internal macro $\backslash ion@ \langle key \rangle @ \langle value \rangle$ storing the definition for the $\langle key \rangle = \langle value \rangle$ pair.

```
\newionumbersthousands
52 \newcommand*\newionumbersthousands[2]{\expandafter\newcommand%
53   \expandafter*\csname ion@thousands@#1\endcsname{\ionumbersoff{#2}}}
```

```
\newionumbersdecimal
54 \newcommand*\newionumbersdecimal[2]{\expandafter\newcommand%
55   \expandafter*\csname ion@decimal@#1\endcsname{\ionumbersoff{#2}}}
```

```
\newionumbersthousandths
56 \newcommand*\newionumbersthousandths[2]{\expandafter\newcommand%
57   \expandafter*\csname ion@thousandths@#1\endcsname{\ionumbersoff{#2}}}
```

```
\newionumbersexponent
58 \newcommand*\newionumbersexponent{%
59   \@ifstar{\newionumbersexponent@@}{\newionumbersexponent@}}
60 \newcommand*\newionumbersexponent@[2]{\expandafter\newcommand%
61   \expandafter*\csname ion@exponent@#1\endcsname{\ionumbersoff{#2}}}
62 \newcommand*\newionumbersexponent@@[2]{\expandafter\newcommand%
63   \expandafter*\csname ion@exponent@#1\endcsname{\ionumbersoff{#2}}%
64   \ion@exponent@superscripttrue}}
```

```
\renewionumbersthousands
65 \newcommand*\renewionumbersthousands[2]{\expandafter\renewcommand%
66   \expandafter*\csname ion@thousands@#1\endcsname{\ionumbersoff{#2}}}
```

```
\renewionumbersdecimal
67 \newcommand*\renewionumbersdecimal[2]{\expandafter\renewcommand%
68   \expandafter*\csname ion@decimal@#1\endcsname{\ionumbersoff{#2}}}
```

```
\renewionumbersthousandths
69 \newcommand*\renewionumbersthousandths[2]{\expandafter\renewcommand%
70   \expandafter*\csname ion@thousandths@#1\endcsname{\ionumbersoff{#2}}}
```


`\renewionnumbersexponent`

```

71 \newcommand*\renewionnumbersexponent{%
72   \@ifstar{\renewionnumbersexponent@@}{\renewionnumbersexponent@}}
73 \newcommand*\renewionnumbersexponent@[2]{\expandafter\renewcommand%
74   \expandafter*\csname ion@exponent@#1\endcsname{\ionnumbersoff{#2}}}%
75 \newcommand*\renewionnumbersexponent@@[2]{\expandafter\renewcommand%
76   \expandafter*\csname ion@exponent@#1\endcsname{\ionnumbersoff{#2}%
77   \ion@exponent@superscripttrue}}

```

7.4 Internal macros holding definitions for $\langle key \rangle = \langle value \rangle$ pairs

First of all, macros with the original character definitions of `.`, `+`, `-` are defined.

```

78 \mathchardef\ion@point@original="013A
79 \mathchardef\ion@comma@original="613B
80 \mathchardef\ion@plus@original="202B
81 \mathchardef\ion@minus@original="2200

```

Here the `\ion@ $\langle key \rangle$ @ $\langle value \rangle$` macros are defined, beginning with the definitions for the comma as input separator.

```

82 \def\ion@comma@ignore{}
83 \def\ion@comma@decimal{\ion@decimal@curr}
84 \def\ion@comma@thousands{\ion@thousands@curr}
85 \def\ion@comma@default{\ion@comma@thousands}

```

The macros `\ion@comma@ $\langle value \rangle$` contain the output for a comma appearing in the input. Actually, a second set of `\ion@aftercomma@ $\langle value \rangle$` macros is required containing commands to be issued whenever a comma appears. If comma is the decimal separator, the appearance of comma in the input will mean that input of the thousands part is complete and the thousandths thousands part starts (`\ion@beforedecimalfalse` must be issued). If comma is the thousands separator, the automatic grouping of thousands will be switched of for that number (`\ion@noexplicitthousandsfalse` must be issued).

```

86 \def\ion@aftercomma@ignore{}
87 \def\ion@aftercomma@decimal{\ion@beforedecimalfalse}
88 \def\ion@aftercomma@thousands{\ion@noexplicitthousandsfalse}
89 \def\ion@aftercomma@default{\ion@aftercomma@thousands}

```

An analogous set of macros is defined for the point as input separator.

```

90 \def\ion@point@ignore{}
91 \def\ion@point@decimal{\ion@decimal@curr}
92 \def\ion@point@thousands{\ion@thousands@curr}
93 \def\ion@point@default{\ion@point@decimal}

```

For the same reasons as mentioned before a set of `\ion@afterpoint@ $\langle value \rangle$` macros is required.

```

94 \def\ion@afterpoint@ignore{}
95 \def\ion@afterpoint@decimal{\ion@beforedecimalfalse}
96 \def\ion@afterpoint@thousands{\ion@noexplicitthousandsfalse}
97 \def\ion@afterpoint@default{\ion@afterpoint@decimal}

```

Next the definitions for the decimal output separator, ...

```

98 \mathchardef\ion@decimal@point="013A
99 \mathchardef\ion@decimal@comma="013B
100 \mathchardef\ion@decimal@punctpoint="613A
101 \mathchardef\ion@decimal@punctcomma="613B
102 \def\ion@decimal@default{\ion@decimal@point}

... the thousands output separator, ...

103 \def\ion@thousands@none{}
104 \mathchardef\ion@thousands@comma="013B
105 \mathchardef\ion@thousands@point="013A
106 \mathchardef\ion@thousands@punctcomma="613B
107 \mathchardef\ion@thousands@punctpoint="613A
108 \def\ion@thousands@apostrophe{^{\prime}}
109 \def\ion@thousands@phantom{\phantom{\ion@point@original}}
110 \def\ion@thousands@space{\,,}
111 \def\ion@thousands@default{\ion@thousands@punctcomma}

... the thousandths output separator, ...

112 \def\ion@thousandths@none{}
113 \mathchardef\ion@thousandths@comma="013B
114 \mathchardef\ion@thousandths@point="013A
115 \mathchardef\ion@thousandths@punctcomma="613B
116 \mathchardef\ion@thousandths@punctpoint="613A
117 \def\ion@thousandths@apostrophe{^{\prime}}
118 \def\ion@thousandths@phantom{\phantom{\ion@point@original}}
119 \def\ion@thousandths@space{\,,}
120 \def\ion@thousandths@default{\ion@thousandths@space}

... and the exponent output separator are given.

121 \def\ion@exponent@none{}
122 \mathchardef\ion@exponent@ite="7165
123 \mathchardef\ion@exponent@itE="7145
124 \mathchardef\ion@exponent@rme="7065
125 \mathchardef\ion@exponent@rmE="7045
126 \def\ion@exponent@timestento{\times 10\,,\ion@exponent@superscripttrue}
127 \def\ion@exponent@cdottento{\cdot 10\,,\ion@exponent@superscripttrue}
128 \def\ion@exponent@wedge{^{\wedge}}
129 \def\ion@exponent@default{\ion@exponent@ite}

```

7.5 Enabling and disabling features

The following helper macros make different subsets of `.`, `+`, `-`, `0`, `1`, `2`, `3`, `4`, `5`, `6`, `7`, `8`, `9` active.

```

130 \def\ion@separators@active{\catcode'\,=\active\catcode'\.=\active\relax}
131 \def\ion@signs@active{\catcode'\+=\active\catcode'\-=\active\relax}
132 \def\ion@digits@active{\catcode'\,=\active\catcode'\.=\active%
133   \catcode'\0=\active\catcode'\1=\active\catcode'\2=\active%
134   \catcode'\3=\active\catcode'\4=\active\catcode'\5=\active%
135   \catcode'\6=\active\catcode'\7=\active\catcode'\8=\active%
136   \catcode'\9=\active\relax}

```

An analogous set of macros makes subsets of these characters active/inactive in math mode.

```

137 \def\ion@separators@math@active{\mathcode'',"8000\mathcode'.'"8000\relax}
138 \def\ion@separators@math@inactive{\mathcode'',"613B\mathcode'.'"013A\relax}
139 \def\ion@signs@math@active{\mathcode'+'"8000\mathcode'-'8000\relax}
140 \def\ion@signs@math@inactive{\mathcode'+'"202B\mathcode'-'2200\relax}
141 \def\ion@digits@math@active{\mathcode'0'"8000\mathcode'1'"8000\mathcode'2'"8000%
142   \mathcode'3'"8000\mathcode'4'"8000\mathcode'5'"8000\mathcode'6'"8000%
143   \mathcode'7'"8000\mathcode'8'"8000\mathcode'9'"8000\relax}
144 \def\ion@digits@math@inactive{\mathcode'0'"7030\mathcode'1'"7031%
145   \mathcode'2'"7032\mathcode'3'"7033\mathcode'4'"7034\mathcode'5'"7035%
146   \mathcode'6'"7036\mathcode'7'"7037\mathcode'8'"7038\mathcode'9'"7039\relax}

```

Next the user interface for making .,+-0123456789 active/inactive follows.

`\ionnumbers`

```

147 \def\ionnumbers{\ion@separators@math@active\ion@signs@math@active%
148   \ion@digits@math@active}

```

`\endionnumbers`

```

149 \def\endionnumbers{\ion@separators@math@inactive\ion@signs@math@inactive%
150   \ion@digits@math@inactive}

```

`\ionnumbersoff`

```

151 \newcommand\ionnumbersoff[1]{\begingroup\endionnumbers#1\ionnumbers\endgroup}

```

Of course, at the begining of the document the charactars shall be active by default.

```

152 \AtBeginDocument{\ionnumbers}

```

7.6 Definitions of active characters

The macro definitions for the characters .,+-0123456789 are hold in the following macros. Number processing works by looking at the next character and performing on or more from the following actions:

- the currently configured output for the character will be added to the end of `\ion@comma@curr` by `\ion@currnum@pushback`; `\ion@comma@curr` stores the currently processed number
- only for comma/point: the corresponding `after...` macro will be issued
- the currently processed number will be output via `\ion@currnum@output`
- the `e` will be eaten and replaced by its configured output

The conditions in the macro definitions should be self-explanatory for each character.

```

153 \def\ion@comma{%
154   \ion@ifnextdigit{%
155     \ion@currnum@pushback*{\ion@comma@curr}\ion@aftercomma@curr%
156   }{%
157     \ion@ifnextseparator{%
158       \ion@currnum@pushback*{\ion@comma@curr}\ion@aftercomma@curr%
159       \@warning{Too many separators}%

```

```

160   }{%
161     \ion@ifnextchar e{%
162       \ion@currnum@pushback*{\ion@comma@curr}\ion@aftercomma@curr%
163       \ion@currnum@output\ion@exponent@curr\@gobble%
164     }{%
165       \ion@currnum@output\ion@comma@original%
166     }%
167   }%
168 }%
169 }
170 \def\ion@point{%
171   \ion@ifnextdigit{%
172     \ion@currnum@pushback*{\ion@point@curr}\ion@afterpoint@curr%
173   }{%
174     \ion@ifnextseparator{%
175       \ion@currnum@pushback*{\ion@point@curr}\ion@afterpoint@curr%
176       \@warning{Too many separators}%
177     }{%
178       \ion@ifnextchar e{%
179         \ion@currnum@pushback*{\ion@point@curr}\ion@afterpoint@curr%
180         \ion@currnum@output\ion@exponent@curr\@gobble%
181       }{%
182         \ion@currnum@output\ion@point@original%
183       }%
184     }%
185   }%
186 }
187 \def\ion@plus{%
188   \ion@ifnextdigit{%
189     \ion@currnum@pushback*{\ion@plus@original}%
190   }{%
191     \ion@ifnextseparator{%
192       \ion@currnum@pushback*{\ion@plus@original}%
193     }{%
194       \ion@ifnextsign{%
195         \ion@currnum@pushback*{\ion@plus@original}\@warning{Too many signs}%
196       }{%
197         \ion@currnum@output\ion@plus@original%
198       }%
199     }%
200   }%
201 }
202 \def\ion@minus{%
203   \ion@ifnextdigit{%
204     \ion@currnum@pushback*{\ion@minus@original}%
205   }{%
206     \ion@ifnextseparator{%
207       \ion@currnum@pushback*{\ion@minus@original}%
208     }{%
209       \ion@ifnextsign{%
210         \ion@currnum@pushback*{\ion@minus@original}\@warning{Too many signs}%
211       }{%
212         \ion@currnum@output\ion@minus@original%
213       }%

```

```

214     }%
215 }%
216 }
217 \def\ion@zero{\ion@currnum@pushback{\mathchar"7030}%
218   \ion@ifnextdigit{%
219     %% nothing
220   }{%
221     \ion@ifnextseparator{%
222       %% nothing
223     }{%
224       \ion@ifnextchar e{%
225         \ion@currnum@output\ion@exponent@curr\@gobble%
226       }{%
227         \ion@currnum@output%
228       }%
229     }%
230   }%
231 }
232 \def\ion@one{\ion@currnum@pushback{\mathchar"7031}%
233   \ion@ifnextdigit{%
234     %% nothing
235   }{%
236     \ion@ifnextseparator{%
237       %% nothing
238     }{%
239       \ion@ifnextchar e{%
240         \ion@currnum@output\ion@exponent@curr\@gobble%
241       }{%
242         \ion@currnum@output%
243       }%
244     }%
245   }%
246 }
247 \def\ion@two{\ion@currnum@pushback{\mathchar"7032}%
248   \ion@ifnextdigit{%
249     %% nothing
250   }{%
251     \ion@ifnextseparator{%
252       %% nothing
253     }{%
254       \ion@ifnextchar e{%
255         \ion@currnum@output\ion@exponent@curr\@gobble%
256       }{%
257         \ion@currnum@output%
258       }%
259     }%
260   }%
261 }
262 \def\ion@three{\ion@currnum@pushback{\mathchar"7033}%
263   \ion@ifnextdigit{%
264     %% nothing
265   }{%
266     \ion@ifnextseparator{%
267       %% nothing

```

```

268   }{%
269     \ion@ifnextchar e{%
270       \ion@currnum@output\ion@exponent@curr\@gobble%
271     }{%
272       \ion@currnum@output%
273     }%
274   }%
275 }%
276 }
277 \def\ion@four{\ion@currnum@pushback{\mathchar"7034}%
278   \ion@ifnextdigit{%
279     %% nothing
280   }{%
281     \ion@ifnextseparator{%
282       %% nothing
283     }{%
284       \ion@ifnextchar e{%
285         \ion@currnum@output\ion@exponent@curr\@gobble%
286       }{%
287         \ion@currnum@output%
288       }%
289     }%
290   }%
291 }
292 \def\ion@five{\ion@currnum@pushback{\mathchar"7035}%
293   \ion@ifnextdigit{%
294     %% nothing
295   }{%
296     \ion@ifnextseparator{%
297       %% nothing
298     }{%
299       \ion@ifnextchar e{%
300         \ion@currnum@output\ion@exponent@curr\@gobble%
301       }{%
302         \ion@currnum@output%
303       }%
304     }%
305   }%
306 }
307 \def\ion@six{\ion@currnum@pushback{\mathchar"7036}%
308   \ion@ifnextdigit{%
309     %% nothing
310   }{%
311     \ion@ifnextseparator{%
312       %% nothing
313     }{%
314       \ion@ifnextchar e{%
315         \ion@currnum@output\ion@exponent@curr\@gobble%
316       }{%
317         \ion@currnum@output%
318       }%
319     }%
320   }%
321 }

```

```

322 \def\ion@seven{\ion@currnum@pushback{\mathchar"7037}}%
323 \ion@ifnextdigit{%
324   %% nothing
325 }{%
326   \ion@ifnextseparator{%
327     %% nothing
328   }{%
329     \ion@ifnextchar e{%
330       \ion@currnum@output\ion@exponent@curr\@gobble%
331     }{%
332       \ion@currnum@output%
333     }%
334   }%
335 }%
336 }
337 \def\ion@eight{\ion@currnum@pushback{\mathchar"7038}}%
338 \ion@ifnextdigit{%
339   %% nothing
340 }{%
341   \ion@ifnextseparator{%
342     %% nothing
343   }{%
344     \ion@ifnextchar e{%
345       \ion@currnum@output\ion@exponent@curr\@gobble%
346     }{%
347       \ion@currnum@output%
348     }%
349   }%
350 }%
351 }
352 \def\ion@nine{\ion@currnum@pushback{\mathchar"7039}}%
353 \ion@ifnextdigit{%
354   %% nothing
355 }{%
356   \ion@ifnextseparator{%
357     %% nothing
358   }{%
359     \ion@ifnextchar e{%
360       \ion@currnum@output\ion@exponent@curr\@gobble%
361     }{%
362       \ion@currnum@output%
363     }%
364   }%
365 }%
366 }

```

The macro `\ion@define@charmacros` is used to assign the above macros to the (active) characters `.,+-0123456789`. It will be executed later in the conflict test section.

```

367 \begingroup
368 \ion@separators@active\ion@signs@active\ion@digits@active
369 \gdef\ion@define@charmacros{%
370   \global\let,=\ion@comma%
371   \global\let.=\ion@point%

```

```

372 \global\let+=\ion@plus%
373 \global\let-=\ion@minus%
374 \global\let0=\ion@zero%
375 \global\let1=\ion@one%
376 \global\let2=\ion@two%
377 \global\let3=\ion@three%
378 \global\let4=\ion@four%
379 \global\let5=\ion@five%
380 \global\let6=\ion@six%
381 \global\let7=\ion@seven%
382 \global\let8=\ion@eight%
383 \global\let9=\ion@nine%
384 }
385 \endgroup

```

Now the macros for the conditions in the above definitions follow. There are tests for a digit 0123456789, ...

```

386 \long\def\ion@ifnextdigit#1#2{%
387 \def\reserved@a{#1}%
388 \def\reserved@b{#2}%
389 \futurelet\@let@token\ion@ifnextdigit@}
390 \def\ion@ifnextdigit@{%
391 \ifx\@let@token1\let\reserved@c\reserved@a\else%
392 \ifx\@let@token2\let\reserved@c\reserved@a\else%
393 \ifx\@let@token3\let\reserved@c\reserved@a\else%
394 \ifx\@let@token4\let\reserved@c\reserved@a\else%
395 \ifx\@let@token5\let\reserved@c\reserved@a\else%
396 \ifx\@let@token6\let\reserved@c\reserved@a\else%
397 \ifx\@let@token7\let\reserved@c\reserved@a\else%
398 \ifx\@let@token8\let\reserved@c\reserved@a\else%
399 \ifx\@let@token9\let\reserved@c\reserved@a\else%
400 \ifx\@let@token0\let\reserved@c\reserved@a\else%
401 \let\reserved@c\reserved@b%
402 \fi%
403 \fi%
404 \fi%
405 \fi%
406 \fi%
407 \fi%
408 \fi%
409 \fi%
410 \fi%
411 \fi%
412 \reserved@c}

```

... for a separator ., , ...

```

413 \long\def\ion@ifnextseparator#1#2{%
414 \def\reserved@a{#1}%
415 \def\reserved@b{#2}%
416 \futurelet\@let@token\ion@ifnextseparator@}
417 \def\ion@ifnextseparator@{%
418 \ifx\@let@token.\let\reserved@c\reserved@a\else%
419 \ifx\@let@token,\let\reserved@c\reserved@a\else%
420 \let\reserved@c\reserved@b%
421 \fi%

```



```

422 \fi%
423 \reserved@c}

... and for a sign +- as next character.
424 \long\def\ion@ifnextsign#1#2{%
425 \def\reserved@a{#1}%
426 \def\reserved@b{#2}%
427 \futurelet\@let@token\ion@ifnextsign@}
428 \def\ion@ifnextsign@{%
429 \ifx\@let@token+\let\reserved@c\reserved@a\else%
430 \ifx\@let@token-\let\reserved@c\reserved@a\else%
431 \let\reserved@c\reserved@b%
432 \fi%
433 \fi%
434 \reserved@c}

```

An additional test for an arbitrary character is also added. It obeys white spaces in contrast to L^AT_EX's \@ifnextchar.

```

435 \long\def\ion@ifnextchar#1#2#3{%
436 \let\reserved@d=#1%
437 \def\reserved@a{#2}%
438 \def\reserved@b{#3}%
439 \futurelet\@let@token\ion@ifnextchar@}
440 \def\ion@ifnextchar@{%
441 \ifx\@let@token\reserved@d%
442 \let\reserved@c\reserved@a%
443 \else%
444 \let\reserved@c\reserved@b%
445 \fi%
446 \reserved@c}

```

7.7 Test for conflicts with other packages

First of all we test for some packages known to conflict with ionumbers. This will be done by checking at the beginning of the document, if one of these packages has been loaded and an error/warning will be issued.

```

447 \newcommand*\ion@conflict@package}[1]{%
448 \ifpackageloaded{#1}{\PackageError{ionumbers}%
449 {Packages #1 and ionumbers conflict!\MessageBreak%
450 Do not load both packages in the same document}}{}{}}
451 \newcommand*\ion@problem@package}[2]{%
452 \@ifpackageloaded{#1}{\PackageWarning{ionumbers}%
453 {Loading #1 and ionumbers is\MessageBreak%
454 problematic!\MessageBreak%
455 #2}}{}{}}
456 \AtBeginDocument{%
457 \ion@conflict@package{ziffer}%
458 \ion@problem@package{dcolumn}{Use ‘tabular’s inside \string\ionumbersoff}%
459 }

```

Next the characters .,+0123456789 are checked for macro definitions (by other packages). This way conflicts with other packages may be detected with some probability (but only if the conflicting package has already been loaded).

```

460 \newcommand*\ion@conflict@definedtest[1]{%
461   \ifx#1\@undefined\else\PackageWarning{ionumbers}%
462     {Potential conflict with other package(s) detected.\MessageBreak%
463     '\string#1' has already been defined. I will redefine it.\MessageBreak%
464     This might break other package(s)!\MessageBreak}\fi}
465 \begingroup
466   \ion@separators@active\ion@signs@active\ion@digits@active
467   \ion@conflict@definedtest{,}%
468   \ion@conflict@definedtest{.}%
469   \ion@conflict@definedtest{+}%
470   \ion@conflict@definedtest{-}%
471   \ion@conflict@definedtest{0}%
472   \ion@conflict@definedtest{1}%
473   \ion@conflict@definedtest{2}%
474   \ion@conflict@definedtest{3}%
475   \ion@conflict@definedtest{4}%
476   \ion@conflict@definedtest{5}%
477   \ion@conflict@definedtest{6}%
478   \ion@conflict@definedtest{7}%
479   \ion@conflict@definedtest{8}%
480   \ion@conflict@definedtest{9}%
481 \endgroup

```

After the above test the definitions of the characters of ionumbers can be applied.

```

482 \ion@define@charmacros

```

Additionally, ionumbers tests for redefinitions of the macros of the characters at the beginning of the document.

```

483 \newcommand*\ion@conflict@redefinedtest[2]{%
484   \ifx#1#2\else\PackageWarning{ionumbers}%
485     {Potential conflict with other package(s) detected.\MessageBreak%
486     '\string#1' has been redefined. This might break ionumbers!\MessageBreak}%
487   \fi}
488 \begingroup
489   \ion@separators@active\ion@signs@active\ion@digits@active
490   \gdef\ion@conflict@redefinedtest@macro{%
491     \ion@conflict@redefinedtest{,}{\ion@comma}%
492     \ion@conflict@redefinedtest{.}{\ion@point}%
493     \ion@conflict@redefinedtest{+}{\ion@plus}%
494     \ion@conflict@redefinedtest{-}{\ion@minus}%
495     \ion@conflict@redefinedtest{0}{\ion@zero}%
496     \ion@conflict@redefinedtest{1}{\ion@one}%
497     \ion@conflict@redefinedtest{2}{\ion@two}%
498     \ion@conflict@redefinedtest{3}{\ion@three}%
499     \ion@conflict@redefinedtest{4}{\ion@four}%
500     \ion@conflict@redefinedtest{5}{\ion@five}%
501     \ion@conflict@redefinedtest{6}{\ion@six}%
502     \ion@conflict@redefinedtest{7}{\ion@seven}%
503     \ion@conflict@redefinedtest{8}{\ion@eight}%
504     \ion@conflict@redefinedtest{9}{\ion@nine}%
505   }
506 \endgroup
507 \AtBeginDocument{\ion@conflict@redefinedtest@macro}

```

7.8 Commands for current number

Numbers are processed by first storing one character after the other in an internal macro to be able to automatically group digits. The basic idea when adding single characters is

- remember, whether we processing the thousands or the thousandths part of a number (`\ifion@beforedecimal`)
- calculate the number of digits processed modulo 3 plus 1 in the current part and
 - for the thousands part: add `\ion@thousands@sepa` for 1, `\ion@thousands@sepb` for 2, and `\ion@thousands@sepc` for 3 after a digit
 - for the thousandths part: add `\ion@thousandths@sep` after each third digit

The macros `\ion@thousands@sep...` and `\ion@thousandths@sep` are empty by default. Before outputting the number, the number of digits in the thousands part is known and the correct `\ion@thousands@sep...` macro can be set to the thousands separator for correct grouping.

First of all, the ifs, counters and empty separator macros are initialized.

```

508 \newif\ifion@beforedecimal\ion@beforedecimaltrue
509 \newif\ifion@noexplicitthousands\ion@noexplicitthousandstrue
510 \newif\ifion@exponent@superscript\ion@exponent@superscriptfalse
511 \newcount\ion@thousands@currpos\ion@thousands@currpos=0
512 \newcount\ion@thousandths@currpos\ion@thousandths@currpos=0
513 \def\ion@currnum{}
514 \def\ion@thousands@sepa{}
515 \def\ion@thousands@sepb{}
516 \def\ion@thousands@sepc{}
517 \def\ion@thousandths@sep{}

```

The macro `\ion@currnum@pushback` adds the character in its argument to the end of `\ion@currnum`. In the starred version adding an empty separator macros is omitted.

```

518 \newcommand{\ion@currnum@pushback}{\ifstar{\ion@currnum@pushback@@}%
519   {\ion@currnum@pushback@}}
520 \newcommand*{\ion@currnum@pushback@@}[1]{%
521   \ion@addto@macro{\ion@currnum}{#1}%
522 }
523 \newcommand*{\ion@currnum@pushback@}[1]{%
524   \ifion@beforedecimal%
525     %% push back (empty) separator and character
526     \ifcase\ion@thousands@currpos%
527       \ion@addto@macro{\ion@currnum}{#1}%
528     \or%
529       \ion@addto@macro{\ion@currnum}{\ion@thousands@sepa#1}%
530     \or%
531       \ion@addto@macro{\ion@currnum}{\ion@thousands@sepb#1}%
532     \or%
533       \ion@addto@macro{\ion@currnum}{\ion@thousands@sepc#1}%
534     \fi%
535     %% advance thousands counter

```

```

536 \advance\ion@thousands@currpos by1\relax%
537 \ifnum\ion@thousands@currpos>3%
538 \ion@thousands@currpos=1%
539 \fi%
540 \else%
541 %% push back (empty) separator and character
542 \ifnum\ion@thousandths@currpos=3%
543 \ion@addto@macro{\ion@currnum}{\ion@thousandths@sep#1}%
544 \else%
545 \ion@addto@macro{\ion@currnum}{#1}%
546 \fi%
547 %% advance thousandths counter
548 \advance\ion@thousandths@currpos by1\relax%
549 \ifnum\ion@thousandths@currpos>3%
550 \ion@thousandths@currpos=1%
551 \fi%
552 \fi%
553 }

```

The `\ion@currnum@output` macro defines the empty separator macros (depending on the current configuration), outputs the current number, and resets everything for the next number.

```

554 \newcommand*{\ion@currnum@output}{%
555 \begingroup%
556 %% set automatic thousands separator
557 \ifion@autothousands%
558 \ifion@noexplicitthousands%
559 \ifcase\ion@thousands@currpos%
560 %% do nothing
561 \or%
562 \def\ion@thousands@sepa{\ion@thousands@curr}%
563 \or%
564 \def\ion@thousands@sepb{\ion@thousands@curr}%
565 \or%
566 \def\ion@thousands@sepc{\ion@thousands@curr}%
567 \fi%
568 \fi%
569 \fi%
570 %% set automatic thousandths separator
571 \ifion@autothousandths%
572 \def\ion@thousandths@sep{\ion@thousandths@curr}%
573 \fi%
574 %% output number
575 \ion@exponent@superscript%
576 ~{\ion@currnum}%
577 \else%
578 \ion@currnum%
579 \fi%
580 \endgroup%
581 %% reset stuff for next number
582 \ion@thousands@currpos=0%
583 \ion@thousandths@currpos=0%
584 \def\ion@currnum{}%
585 \ion@beforedecimaltrue%

```


\ifion@beforedecimal	\ion@comma@thousands	\ion@defpackopts ..
..... 508, 524 84, 85	... 4, 6, 9, 12,
\ifion@exponent@superscript	\ion@conflict@definedtest	14, 16, 18, 20, 22
..... 510, 575 460, 467–480	\ion@digits@active .
\ifion@noexplicitthousands	\ion@conflict@package	. 132, 368, 466, 489
..... 509, 558 447, 457	\ion@digits@math@active
\ifnum 537, 542, 549	\ion@conflict@redefinedtest 141, 148
\ifx 391–400, 483, 491–504	\ion@digits@math@inactive
418, 419, 429,	\ion@conflict@redefinedtest@macro 144, 150
430, 441, 461, 484 490, 507	\ion@eight 337, 382, 503
\ion@addto@macro 521,	\ion@currnum .. 513,	\ion@exponent@cdottento
527, 529, 531,	521, 527, 529, 127
533, 543, 545, 588	531, 533, 543,	\ion@exponent@curr .
\ion@aftercomma@curr	545, 576, 578, 584	.. 43, 163, 180,
. 33, 155, 158, 162	\ion@currnum@output	225, 240, 255,
\ion@aftercomma@decimal 163, 165,	270, 285, 300,
..... 87	180, 182, 197,	315, 330, 345, 360
\ion@aftercomma@default	212, 225, 227,	\ion@exponent@default
..... 89	240, 242, 255, 129
\ion@aftercomma@ignore	257, 270, 272,	\ion@exponent@itE . 123
..... 86	285, 287, 300,	\ion@exponent@ite .
\ion@aftercomma@reset	302, 315, 317, 122, 129
..... 8	330, 332, 345,	\ion@exponent@none . 121
\ion@aftercomma@thousands	347, 360, 362, 554	\ion@exponent@reset 18
..... 88, 89	\ion@currnum@pushback	\ion@exponent@rmE . 125
\ion@afterpoint@curr 155,	\ion@exponent@rme . 124
. 36, 172, 175, 179	158, 162, 172,	\ion@exponent@superscriptfalse
\ion@afterpoint@decimal	175, 179, 189, 510, 587
..... 95, 97	192, 195, 204,	\ion@exponent@superscripttrue
\ion@afterpoint@default	207, 210, 217,	.. 64, 77, 126, 127
..... 97	232, 247, 262,	\ion@exponent@timestento
\ion@afterpoint@ignore	277, 292, 307, 126
..... 94	322, 337, 352, 518	\ion@exponent@wedge 128
\ion@afterpoint@reset	\ion@currnum@pushback@	\ion@five . 292, 379, 500
..... 11 519, 523	\ion@four . 277, 378, 499
\ion@afterpoint@thousands	\ion@currnum@pushback@@	\ion@ifnextchar ...
..... 96 518, 520	. 161, 178, 224,
\ion@autothousandsreset	\ion@decimal@comma . 99	239, 254, 269,
..... 20, 21	\ion@decimal@curr .	284, 299, 314,
\ion@autothousandthsreset 37, 83, 91	329, 344, 359, 435
..... 22, 23	\ion@decimal@default	\ion@ifnextchar@ ..
\ion@beforedecimalfalse 102 439, 440
..... 87, 95	\ion@decimal@point .	\ion@ifnextdigit ..
\ion@beforedecimaltrue 98, 102 154, 171,
..... 508, 585	\ion@decimal@punctcomma	188, 203, 218,
\ion@comma 153, 370, 491 101	233, 248, 263,
\ion@comma@curr ...	\ion@decimal@punctpoint	278, 293, 308,
. 32, 155, 158, 162 100	323, 338, 353, 386
\ion@comma@decimal . 83	\ion@decimal@reset . 12	\ion@ifnextdigit@ .
\ion@comma@default . 85	\ion@define@charmacros 389, 390
\ion@comma@ignore . 82 369, 482	\ion@ifnextseparator
\ion@comma@original	\ion@deflocopts 157, 174,
..... 79, 165	. 29, 31, 34, 37,	191, 206, 221,
\ion@comma@reset ... 7	39, 41, 43, 45, 46	236, 251, 266,

281, 296, 311, 326, 341, 356, 413	<code>\ion@thousands@comma</code> 104	<code>\ion@thousandths@space</code> 119, 120
<code>\ion@ifnextseparator@</code> 416, 417	<code>\ion@thousands@curr</code> 39, 84, 92, 562, 564, 566	<code>\ion@three</code> 262, 377, 498 <code>\ion@two</code> .. 247, 376, 497 <code>\ion@zero</code> . 217, 374, 495
<code>\ion@ifnextsign</code> 194, 209, 424	<code>\ion@thousands@currpos</code> 511, 526, 536–538, 559, 582	<code>\ionnumbers</code> 5, 147, 151, 152
<code>\ion@ifnextsign@</code> 427, 428	<code>\ion@thousands@default</code> 111	<code>\ionnumbersoff</code> 5, 53, 55, 57, 61, 63, 66, 68, 70, 74, 76, 151, 458
<code>\ion@minus</code> 202, 373, 494	<code>\ion@thousands@none</code> 103	<code>\ionnumbersresetstyle</code> 5, 47, 51
<code>\ion@minus@original</code> 81, 204, 207, 210, 212	<code>\ion@thousands@phantom</code> 109	<code>\ionnumbersstyle</code> 5, 30, 48
<code>\ion@nine</code> . 352, 383, 504	<code>\ion@thousands@point</code> 105	<code>\ionnumberstyle</code> 30
<code>\ion@noexplicitthousandsfall</code> 88, 96	<code>\ion@thousands@punctcomma</code> 106, 111	L
<code>\ion@noexplicitthousandstr</code> 509, 586	<code>\ion@thousands@punctpoint</code> 107	<code>\let</code> 370–383, 391–401, 418–420, 429– 431, 436, 442, 444
<code>\ion@one</code> .. 232, 375, 496	<code>\ion@thousands@reset</code> 14	<code>\long</code> . 386, 413, 424, 435
<code>\ion@plus</code> . 187, 372, 493	<code>\ion@thousands@sepa</code> 514, 529, 562	M
<code>\ion@plus@original</code> 80, 189, 192, 195, 197	<code>\ion@thousands@sepb</code> 515, 531, 564	<code>\mathchar</code> 217, 232, 247, 262, 277, 292, 307, 322, 337, 352
<code>\ion@point</code> 170, 371, 492	<code>\ion@thousands@sepc</code> 516, 533, 566	<code>\mathchardef</code> 78–81, 98– 101, 104–107, 113–116, 122–125
<code>\ion@point@curr</code> 35, 172, 175, 179	<code>\ion@thousands@space</code> 110	<code>\mathcode</code> 137–146
<code>\ion@point@decimal</code> 91, 93	<code>\ion@thousandths@apostrophe</code> 117	<code>\MessageBreak</code> 449, 453, 454, 462–464, 485, 486
<code>\ion@point@default</code> . 93	<code>\ion@thousandths@comma</code> 113	N
<code>\ion@point@ignore</code> . 90	<code>\ion@thousandths@curr</code> 41, 572	<code>\newcommand</code> 4, 5, 29, 30, 47, 52, 54, 56, 58, 60, 62, 65, 67, 69, 71, 73, 75, 151, 447, 451, 460, 483, 518, 520, 523, 554, 588
<code>\ion@point@original</code> . 78, 109, 118, 182	<code>\ion@thousandths@currpos</code> 512, 542, 548–550, 583	<code>\newcount</code> 511, 512
<code>\ion@point@reset</code> .. 10	<code>\ion@thousandths@default</code> 120	<code>\newif</code> 2, 3, 508–510
<code>\ion@point@thousands</code> 92	<code>\ion@thousandths@none</code> 112	<code>\newionnumbersdecimal</code> 5, 54
<code>\ion@problem@package</code> 451, 458	<code>\ion@thousandths@phantom</code> 118	<code>\newionnumbersexponent</code> 5, 58
<code>\ion@separators@active</code> . 130, 368, 466, 489	<code>\ion@thousandths@point</code> 114	<code>\newionnumbersexponent@</code> 59, 60
<code>\ion@separators@math@active</code> 137, 147	<code>\ion@thousandths@punctcomma</code> 115	<code>\newionnumbersexponent@@</code> 59, 62
<code>\ion@separators@math@inactive</code> 138, 149	<code>\ion@thousandths@punctpoint</code> 116	
<code>\ion@setpackopts</code> 5, 24, 27	<code>\ion@thousandths@reset</code> 16	
<code>\ion@seven</code> 322, 381, 502	<code>\ion@thousandths@sep</code> 517, 543, 572	
<code>\ion@signs@active</code> . . 131, 368, 466, 489		
<code>\ion@signs@math@active</code> 139, 147		
<code>\ion@signs@math@inactive</code> 140, 149		
<code>\ion@six</code> .. 307, 380, 501		
<code>\ion@thousands@apostrophe</code> 108		

<code>\newionumbersthousands</code>	143, 146, 536, 548	<code>\reserved@b</code> ... 388,
..... 5, <u>52</u>	<code>\renewcommand</code>	401, 415, 420,
<code>\newionumbersthousandths</code>	. 65, 67, 69, 73, 75	426, 431, 438, 444
..... 5, <u>56</u>	<code>\renewionumbersdecimal</code>	<code>\reserved@c</code> 391–401,
<code>\noexpand</code> 590 5, <u>67</u>	412, 418–420,
O	<code>\renewionumbersexponent</code>	423, 429–431,
<code>\or</code> 528, 530, 5, <u>71</u>	434, 442, 444, 446
532, 561, 563, 565	<code>\renewionumbersexponent@</code>	<code>\reserved@d</code> ... 436, 441
 72, 73	
P	<code>\renewionumbersexponent@@</code>	S
<code>\PackageError</code> 448 72, 75	<code>\setkeys</code> 5, 30
<code>\PackageWarning</code> ...	<code>\renewionumbersthousands</code>	<code>\string</code> ... 458, 463, 486
..... 452, 461, 484 5, <u>65</u>	
<code>\phantom</code> 109, 118	<code>\renewionumbersthousandths</code>	T
<code>\prime</code> 108, 117 5, <u>69</u>	<code>\the</code> 590
<code>\ProcessOptions</code> ... 28	<code>\RequirePackage</code> 1	<code>\times</code> 126
	<code>\reserved@a</code> ... 387,	<code>\toks@</code> 589, 590
R	391–400, 414,	
<code>\relax</code> 28, 130,	418, 419, 425,	W
131, 136–140,	429, 430, 437, 442	<code>\wedge</code> 128