siunitx — A comprehensive (SI) units package[*]

Joseph Wright[†]

Released 2009/02/07

Abstract

Typesetting values with units requires care to ensure that the combined mathematical meaning of the value plus unit combination is clear. In particular, the SI units system lays down a consistent set of units with rules on how these are to be used. However, different countries and publishers have differing conventions on the exact appearance of numbers (and units).

The siunitx package provides a set of tools for authors to typeset numbers and units in a consistent way. The package has an extended set of configuration options which make it possible to follow varying typographic conventions with the same input syntax. The package includes automated processing of numbers and units, and the ability to control tabular alignment of numbers.

A number of LaTeX packages have been developed in the past for formatting units: SIunits, SIstyle, unitsdef, units, fancyunits and fancynum. Support for users of all of these packages is available as emulation modules in siunitx. In addition, siunitx can carry out many of the functions of the dcolumn, rccol and numprint packages.

Contents

[*] This file describes version v2.0alpha, last revised 2009/02/07.

[†] E-mail: joseph.wright@morningstar2.co.uk

1 Introduction

The correct application of units of measurement is very important in technical applications. For this reason, carefully-crafted definitions of a coherent units system have been laid down by the *Conférence Générale des Poids et Mesures*[1] (CGPM): this has resulted in the *Système International d'Unités*[2] (SI). At the same time, typographic conventions for correctly displaying both numbers and units exist to ensure that no loss of meaning occurs in printed matter.

siunitx aims to provide a unified method for LaTeX users to typeset units and values correctly and easily. The design philosophy of siunitx is to follow the agreed rules by default, but to allow variation through option settings. In this way, users can use siunitx to follow the requirements of publishers, co-authors, universities, *etc*. without needing to alter the input at all.

siunitx is intended as a complete replacement for SIunits, SIstyle, unitsdef, units, fancyunits and fancynum. As such, emulation modes are provided for all of these packages. Where possible, conventions from the existing solutions have been used here. For example, the macros `\num`, `\ang` and `\SI` act in a very similar fashion to those in existing packages.

2 Installation

The entire bundle is supplied with the TDS-ready `zip` file, `siunitx.tds.zip`. Simply unzip this into your local texmf tree and run your hash program (`texhash` for TeXLive or `initextmf -u` for MiKTeX).

To extract the package `siunitx.sty` and the configuration files from `siunitx.dtx`, two methods are available. To extract the files using the `ins` file, simply run (pdf)TeX on `siunitx.ins`. This will produce all of the package files, and also `README.txt`. To extract the files and build the documentation, run (pdf)LaTeX on `siunitx.dtx`. Three (pdf)LaTeX runs with `\write18` enabled will also build the index and table of contents in the PDF.

Compilation of the package documentation requires the l3doc class, from the expl3 bundle produced by the LaTeX3 team. To compile the package documentation, you will need to get a recent version of expl3 from the LaTeX project website.

3 siunitx for the impatient

The package provides the user macros:

- `\SI[`⟨*options*⟩`]{`⟨*value*⟩`}[`⟨*pre-unit*⟩`]{`⟨*unit*⟩`}`

[1] General Conference on Weights and Measures.

[2] International System of Units.

- `\si[`⟨*options*⟩`]{`⟨*unit*⟩`}`
- `\num[`⟨*options*⟩`]{`⟨*number*⟩`}`
- `\ang[`⟨*options*⟩`]{`⟨*angle*⟩`}`
- `\sisetup{`⟨*options*⟩`}`

plus the `S` and `s` column types for decimal alignments and units in tables. These macros are designed for typesetting units and values with control of appearance and with intelligent processing.

12 345.678 90
$1 \pm 2\mathrm{i}$
0.3×10^{45}

```
\num{12345,67890} \\
\num{1+-2i}       \\
\num{.3e45}
```

By default, all text is typeset in the current upright, serif maths font. This can be changed by setting the appropriate options: `\sisetup{font/detect/all}` will use the current font for typesetting.

4 Using the siunitx package

4.1 Loading the package

The package should be loaded in the usual LaTeX 2ε way.

```
\usepackage{siunitx}
```

The package does not use load-time options, although it does support those from version 1 of the package and predecessor packages.

4.2 Numbers

`\num` Numbers are automatically formatted by the `\num` macro. This takes one optional and one mandatory argument: `\num[`⟨*options*⟩`]{`⟨*number*⟩`}`. The contents of ⟨*number*⟩ are automatically formatted. The formatter removes "hard" spaces (`\,` and `~`), automatically identifies exponents (by default marked using `e` or `d`) and adds the appropriate spacing of large numbers. A leading zero is added before a decimal marker, if needed: both "`.`" and "`,`" are recognised as decimal marker.

```
\num{123}      \num{1234}   \num{12345}  \\
\num{0.123}    \num{0,1234} \num{.12345} \\
\num{3.45d-4} \num{-e10}
```

123 1234 12 345
0.123 0.1234 0.123 45
3.45×10^{-4} $- \times 10^{10}$

5 The key–value control system

\sisetup The behaviour of the siunitx package is controlled by a number of key–value options. These can be given globally using the \sisetup function or locally as the optional argument to the user macros.

All of the keys are controlled using the pgfkeys approach to organisation. This means that the keys are split into "paths" of related keys. A single key is set by giving the path plus key name; if you need to set several keys on the same path, you can "change" to the appropriate path and give the key name alone. For example, valid numerical input is controlled by keys in the numbers/input/ path:

```
\sisetup{
  numbers/input/signs            = +-\pm\mp,
  numbers/input/exponent markers = dDeE
}
```

or

```
\sisetup{
  numbers/input/.cd,
  signs            = +-\pm\mp,
  exponent markers = dDeE
}
```

are both valid.

The package uses a range of different key types:

Choice Takes a limited number of choices, which are described separately for each key.

Literal A key which uses the value(s) given directly, either to check input (for example the numbers/input keys) or in output.

Maths Similar to a literal option, but the input is always used in maths mode, irrespective of other siunitx settings. Thus to text-mode only input must be placed inside the argument of a \text macro.

Macro Requires a macro, which may need a single argument.

Style A key which contains a number of other keys to set. Only the key name should be given: no value is required. This type of key is user-definable, as described in Section **??**.

Switch These are on–off switches, and recognise true, on and yes to turn on, and false, off and no to turn off. Giving just the key name also turns the key on.

The tables of option names use these descriptions to indicate how the keys should be used.

Table 1: `font/detect/` options

Option name	Type	Default
`all`	Style	⟨*none*⟩
`bold`	Switch	`false`
`display maths`	Switch	`false`
`family`	Switch	`false`
`inline bold`	Choice	`text`
`italic`	Switch	`false`
`mode`	Switch	`false`
`none`	Style	⟨*none*⟩

In all cases, UK and US English spellings are available for both option names and for settings. Thus `centre` and `center` can be used for alignment options, and `maths` or `math` is valid in the names of font options. In the rest of this document, UK English spelling is used.

5.1 Detecting fonts

The siunitx package controls the font used to print output independently of the surrounding material. The standard method is to ignore the surroundings entirely, and to use the current upright maths font for all printing.[3] However, the package can detect and follow surrounding bold, italic and font family changes. The font detection options are available in path `font/detect/` and are summarised in Table 1.

`font/detect/bold`
`font/detect/family`
`font/detect/italic`
`font/detect/mode`

The four basic options `bold` and `italic` set detection of the prevailing bold and italic states, respectively. The italic state is only checked if the surrounding material is not in maths mode (as maths text is always italic). Detecting the current family (roman, sans serif or monospaced) is controlled by the `family` setting, while the current mode (text or maths) is detected using the `mode` switch.

`font/detect/all`
`font/detect/none`

The two style options `all` and `none` can be used to turn on or off all of the detection functions in one go. These are style options, and so need no value.

[3] This will typically use `\mathrm`.

1234
1234
1234
1234
1234
1234
1234
1234
1234
1234

```
\sisetup{font/detect/none}%
$\num{1234}$ \\
\num{1234} \\
\emph{\num{1234}} \\
\textbf{\num{1234}} \\
\textbf{$\num{1234}$} \\
\sisetup{font/detect/all}%
$\num{1234}$ \\
\num{1234} \\
\emph{\num{1234}} \\
\textbf{\num{1234}} \\
\textbf{$\num{1234}$} \\\
```

`font/detect/inline bold` Bold detection is influenced by the value of `inline bold`, which takes values `text` and `maths`. The package can detect the local value of bold for either the surrounding text, or the surrounding inline (`$`...`$`) maths.

1234
1234
1234
1234
1234

```
\sisetup{
  font/detect/bold = on,
  font/detect/inline bold = maths
}%
$\num{1234}$ \\
{ \boldmath $\num{1234}$ } \\
{ \bfseries $\num{1234}$ } \\
\sisetup{
  font/detect/inline bold = text
}
{ \boldmath $\num{1234}$ } \\
{ \bfseries $\num{1234}$ }
```

`font/detect/display maths` The font detection system can treat displayed mathematical content in two ways. This is controlled by the `display` option. When set `on`, display mathematics is treated independently from the body of the document. Thus the local *maths* font is checked for matching. In contrast, when set `off`, display material is treated with the current running text font.

```
\sffamily
Some text
\sisetup{
  font/detect/all,
  font/detect/display maths = true
}
\[ x = \SI{1.2e3}{\kg\kelvin\candela} \]
More text
\sisetup{font/detect/display maths = false}
\[ y = \SI{3}{\metre\second\mole} \]
```

Table 2: `font/` options (all also apply in `font/units/` and `font/numbers`)

Option name	Type	Default
`maths rm`	Macro	`\mathrm`
`maths sf`	Macro	`\mathsf`
`maths tt`	Macro	`\mathtt`
`mode`	Choice	`maths`
`text rm`	Macro	`\rmfamily`
`text sf`	Macro	`\sffamily`
`text tt`	Macro	`\ttfamily`

Some text

$$x = 1.2 \times 10^{3}$$

More text

$$y = 3$$

5.2 Output font families

The relationship between font family detected and font family used for output is not fixed. The font detected by the package in the surrounding material does not have to match that used for output. This is controlled by the `font/output` options.

`mode` The `mode` option determines whether siunitx uses maths or text mode when printing output. The choices are `maths`, `math` and `text`. When using maths mode, text is printed using a maths font whereas in text mode a text font is used. The extent to which this is visually obvious depends on the fonts in use in the document. This manual uses old style (lower-case) figures in text mode to highlight the differences. This option has no effect if the `font/detect/mode` switch is `on`.

`maths rm` `text rm` `maths sf` `maths tt` `text sf` `text tt` If font family detection is inactive, siunitx uses the font family stored in either `maths rm` or `text rm` for output. The choice of `maths` or `text` depends on the `mode` setting. If font family detection is active, siunitx may be using a sans serif or monospaced font for output. In maths mode, these are stored in `maths sf` and `maths tt`, and for text mode in `text sf` and `text tt`. Notice that the detected and output font families can differ.

1234
1234
99
99

```
\sisetup{font/detect/family = yes}%
\num{1234} \\
{ \sffamily \num{1234} } \\
\SI{99}{\metre} \\
\sisetup{font/maths rm = \mathtt}%
\SI{99}{\metre}
```

This can be used to good effect to change all output from siunitx without needing to detect the font. For example, when creating beamer presentations the settings

Table 3: `numbers/input/` options

Option name	Type	Default
`complex roots`	Literal	`ij`
`close uncertainty`	Literal	`)`
`decimal markers`	Literal	`.,`
`digits`	Literal	`0123456789`
`exponent markers`	Literal	`dDeE`
`ignore`	Literal	⟨*none*⟩
`open uncertainty`	Literal	`(`
`signs`	Literal	`+-\pm\mp`
`symbols`	Literal	`\pi`

```
\sisetup{
  font/maths rm = \mathsf,
  font/text rm  = \sffamily
```

given all output in sans serif font without font detection.

Every one of the font options can be given independently for units and number, with the option paths `font/units/` and `font/numbers/`, respectively. This allows fine control of output.

5.3 Parsing numbers

The package uses a sophisticated parsing system to understand numbers. This allows siunitx to carry out a range of formatting, as described later. All of the input options take lists of literal tokens, and are summarised in Table 3.

`numbers/input/digits`
`numbers/input/decimal markers`
`numbers/input/signs`
`numbers/input/exponent markers`

The basic parts of a number are the digits, any sign and a separator between the integer and decimal parts. These are stored in the input options `digits`, `decimal markers` and `signs`, respectively. More than one input decimal marker can be used: it will be converted by the package to the appropriate output marker. Numbers which include an exponent part also require a marker for the exponent: this again is taken from the range of tokens in the `exponent markers` option.

`numbers/input/ignore`
`numbers/input/symbols`

As well as “normal” digits, the package will interpret symbolic “numbers” (such as `\pi`) correctly if they are included in the `symbols` list. Tokens given in the `ignore` list are totally passed over by siunitx: they will be removed from the input with no further processing.

`numbers/input/open uncertainty`
`numbers/input/close uncertainty`

In some fields, it is common to give the uncertainty in a value in brackets after the main part of the number, for example “1.234(5)”. The opening and closing symbols used for this type of input are set as `open uncertainty` and `close uncertainty`.

`numbers/input/complex roots`

When using complex numbers in input, the complex root ($\sqrt{-1}$) is indicated by one of the tokens stored in `complex roots`.

Table 4: `numbers/process/` options

Option name	Type	Default
`add zero decimal`	Switch	`false`
`add zero integer`	Switch	`false`
`explicit sign`	Literal	`+`
`include explicit sign`	Switch	`false`
`retain explicit plus`	Switch	`false`
`retain zero exponent`	Switch	`false`
`round mode`	Choice	`off`
`round figures`	Number	`2`
`round places`	Number	`2`

5.4 Post-processing numbers

Before typesetting numbers, various post-processing steps can be carried out. These involve adding or removing information from the number in a systematic way; the options are summarised in Table 4.

`numbers/process/round mode`
`numbers/process/round figures`
`numbers/process/round places`

The siunitx package can round numerical input to a fixed number of significant figures or decimal places. This is controlled by the `round mode` option, which takes the choices `off`, `figures` and `places`. When rounding is turned on, the number of figures to use is determined by the `round figures` and `round places` option: both of these options require a number.

`numbers/process/add zero decimal`
`numbers/process/add zero integer`

It is possible to give real (floating point) numbers as input omitting the decimal or the integer parts of the number (for example 0.123 or 123.0). The options `add zero decimal` and `add zero integer` allow the package to "fill in" the missing zero.

`numbers/process/explicit sign`
`numbers/process/include explicit sign`
`numbers/process/retain explicit plus`

The inclusion of a leading plus sign is usually unnecessary for positive numbers, and so the `retain explicit plus` option is available to control whether these are printed. As the same time, it may be useful to force all numbers to have a sign. This behaviour is controlled by the `include explicit sign` option, with the sign to use stored by the `explicit sign` option.

5.5 Printing numbers

Actually prinitng numbers is controlled by a number of settings, which apply ideas such as differing decimal markers, digit grouping and so on. All of these options are concerned with the appearance of output, rather than the data it conveys. The options are summarised in Table 5.

`numbers/output/group digits`
`numbers/output/group four digits`
`numbers/output/group separator`

Grouping digits into blocks of three is a common method to increase the ease of reading of numbers. The `group digits` choice turns this behaviour on and off, with grouping for numbers of exactly four digits controlled by the `group four digits` choice. Note

Table 5: `numbers/output/` options

Option name	Type	Default
`close bracket`	Literal	`)`
`close uncertainty`	Literal	`)`
`complex root`	Maths	`i`
`decimal marker`	Maths	`.`
`exponent base`	Literal	`10`
`exponent product`	Maths	`\times`
`group digits`	Switch	`true`
`group four digits`	Switch	`false`
`group separator`	Maths	`\,`
`open bracket`	Literal	`(`
`open uncertainty`	Literal	`(`
`separate uncertainty`	Switch	`false`
`tight spacing`	Switch	`false`
`use brackets`	Switch	`true`
`uncertainty space`	Maths	⟨*none*⟩

that the later only applies if `group digits` is turned on. The separator used between groups of digits is stored by the `group separator` option. This takes literal input and is used in maths mode: for a text-mode full space use `\text{~}`.

```
\num{12345} \\
\num[numbers/output/group digits = off]{12345} \\
\num{1234} \\
\num[numbers/output/group four digits = on]{1234} \\
\num{12345} \\
\num[numbers/output/group separator = {,}]{12345} \\
\num[numbers/output/group separator = \text{~}]{12345}
```

12 345
12345
1234
1 234
12 345
12,345
12 345

`numbers/output/complex root`
`numbers/output/decimal marker`

The decimal marker used in output is set using the `decimal marker` option. This can differ from the input marker, as can the root of $\sqrt{-1}$, which is stored in the `complex root` option. The later is always in maths mode, but notice that siunitx uses `\mathrm` by default. Thus an italic *i* is obtained by forcing `\mathnormal`.

```
\num{1.23} \\
\num[numbers/output/decimal marker = {,}]{1.23} \\
\num{1+2i} \\
\num[numbers/output/complex root = \mathnormal{i}]{1+2i}
```

1.23
$1,23$
$1 + 2\mathrm{i}$
$1 + 2i$

`numbers/output/exponent base`
`numbers/output/exponent product`

When exponents are present in the input, the options `exponent base` and `exponent product` set the obvious parts of the output. Notice that the base is in the current mode, but the product sign is always in maths mode.

```
\num[numbers/output/exponent product = \times]{1e2} \\
\num[numbers/output/exponent product = \cdot]{1e2} \\
\num[numbers/output/exponent base = 2]{1e2}
```

1×10^{2}
$1 \cdot 10^{2}$
1×2^{2}

`numbers/output/separate uncertainty`
`numbers/output/uncertainty space`
`numbers/output/open uncertainty`
`numbers/output/close uncertainty`

When input is given including an uncertatinty in a value, it can be printed either with the uncertainty in brackets or as a separate number. This behaviour is controlled by the `separate uncertainty` choice. If the uncertainty is given in brackets, a space may be added between the main value and the uncertainty: this is stored using the `uncertainty space` option. The opening and closing brackets used are stored `open uncertainty` and `close uncertainty`, respectively.

```
\num{1.234(5)} \\
\num[numbers/output/separate uncertainty = on]{1.234(5)} \\
\sisetup{
  numbers/output,
  open uncertainty  = [,
  close uncertainty = ],
  uncertainty space = {\,}
}
\num{1.234(5)}
```

$1.234(5)$
1.234 ± 0.005
$1.234\,[5]$

`numbers/output/use brackets`
`numbers/output/open bracket`
`numbers/output/close bracket`

There are certain combinations of numerical input which can be ambiguous. This can be corrected by adding brackets in the appropriate place, and is controlled by the `use brackets` switch. The opening and closing brackets used are stored `open bracket` and `close bracket`, respectively.

```
\num{1+2i e10} \\
\num[numbers/output/use brackets = false]{1+2i e10} \\
\sisetup{
  numbers/output,
  open bracket  = \{,
  close bracket = \},
}
\num{1+2i e10}
```

$(1+2\mathrm{i})\times10^{10}$
$1+2\mathrm{i}\times10^{10}$
$\{1+2\mathrm{i}\}\times10^{10}$

`numbers/output/tight spacing` Under some circumstances is may be desirable to "squeeze" the output spacing. This is turned on using the `tight spacing` switch, which compresses spacing where possible.

```
\num{1\pm2i e3} \\
\num[numbers/output/tight spacing = true]{1\pm2i e3} \\
```

$(1\pm2\mathrm{i})\times10^{3}$
$(1{\pm}2\mathrm{i}){\times}10^{3}$

Change History

v1.0
General: First official release 1

v1.1
General: Package extended to a greater range of unit types 1

v2.0
General: Complete re-write of package to add many new features 1

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.