

siunitx — A comprehensive (SI) units package*

Joseph Wright[†]

Released 2009/12/10

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.

Contents

1	Introduction	2
2	Installation	2
3	siunitx for the impatient	3
4	Using the siunitx package	4
4.1	Loading the package	4
4.2	Numbers	5
4.3	Units	5
4.4	The unit macros	6
4.5	Creating new macros	9

*This file describes v2.0alpha (revision 220), last revised 2009/12/10.

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

5	The key–value control system	11
5.1	Detecting fonts	12
5.2	Output font families	13
5.3	Parsing numbers	15
5.4	Post-processing numbers	16
5.5	Printing numbers	18
5.6	Multi-part numbers	21
5.7	Creating units	22
5.8	Loading additional units	23
5.9	Using units	26
5.10	Numbers with units	28
5.11	Symbols	29
6	Usage tips	30
6.1	Ensuring text or maths output	30
6.2	Using units such as $\mu\text{m s}^{-1}$ in headings	31
	Change History	31
	Index	32

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* (CGPM): this has resulted in the *Système International d’Unités* (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 L^AT_EX 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.

2 Installation

The package is supplied in dtx format and as a pre-extracted zip file, `siunitx.tds.zip`. The later is most convenient for most users: simply unzip this in your local `texmf` directory and run `texhash` to update the database of file locations. If you want to unpack the dtx yourself, running `tex siunitx.dtx` will extract the package whereas `latex siunitx.dtx` will extract it and also typeset the documentation.

The package requires L^AT_EX₃ support as provided in the `expl3` and `xpackages` bundles. Both of these are available on [CTAN](#) as ready-to-install zip files. Suitable versions are available in MiK_TE_X 2.8 and T_EX Live 2009 (updating the relevant packages online may be necessary). L^AT_EX₃, and so `siunitx`, requires the ϵ -T_EX extensions: these are available on all modern T_EX systems.

Typesetting the documentation requires a number of packages in addition to those needed to use the package. This is mainly because of the number of demonstration items included in the text. To compile the documentation without error, you will need the packages:

- `amsmath`
- `booktabs`
- `caption`
- `csquotes`
- `helvet`
- `mathpazo`
- `listings`

The `xfrac` package is also loaded if available, but is not required to typeset the documentation.

3 `siunitx` for the impatient

The package provides the user macros:

- `\num[options]{number}`
- `\si[options]{unit}`
- `\SI[options]{value}[pre-unit]{unit}`
- `\numrange[options]{number1}{number2}`

- `\SIrange[⟨options⟩]{⟨number1⟩}{⟨number2⟩}{⟨unit⟩}`
- `\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.

Numbers are processed with understanding of exponents, complex numbers and multiplication.

12 345.678 90	<code>\num{12345,67890}</code>	<code>\\</code>
$1 \pm 2i$	<code>\num{1+-2i}</code>	<code>\\</code>
0.3×10^{45}	<code>\num{.3e45}</code>	<code>\\</code>
$1.654 \times 2.34 \times 3.430$	<code>\num{1.654 x 2.34 x 3.430}</code>	

The unit system can interpret units given as text to be used directly or as macro-based units. In the later case, different formatting is possible.

<code>\si{kg.m.s^{-1}}</code>	<code>\\</code>
<code>\si{\kilogram\metre\per\second}</code>	<code>\\</code>
<code>\si[per-mode=symbol]</code>	
<code>{\kilogram\metre\per\second}</code>	<code>\\</code>
<code>\si[per-mode=symbol]</code>	
<code>{\kilogram\metre\per\ampere\per\second}</code>	
kg m s^{-1}	
kg m s^{-1}	
$(\text{kg m})/\text{s}$	
$(\text{kg m})/(\text{A s})$	

Simple ranges of numbers can be handled.

10 to 20	<code>\numrange{10}{20}</code>	<code>\\</code>
0.13 mm to 0.67 mm	<code>\SIrange{0.13}{0.67}{\milli\metre}</code>	

By default, all text is typeset in the current upright, serif maths font. This can be changed by setting the appropriate options: `\sisetup{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 L^AT_EX 2_ε way.

```
\usepackage{siunitx}
```

4.2 Numbers

`\num` `\num[options]{number}`

Numbers are automatically formatted by the `\num` macro. This takes one optional argument, `<options>`, and one mandatory one, `<number>`. The contents of `<number>` are automatically formatted. The formatter removes “hard” spaces (`\`, and `~`), automatically identifies exponents (by default marked using `e`, `E`, `d` 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.

123	<code>\num{123}</code>	<code>\</code>
1234	<code>\num{1234}</code>	<code>\</code>
12 345	<code>\num{12345}</code>	<code>\</code>
0.123	<code>\num{0.123}</code>	<code>\</code>
0.1234	<code>\num{0,1234}</code>	<code>\</code>
0.123 45	<code>\num{.12345}</code>	<code>\</code>
3.45×10^4	<code>\num{3.45d-4}</code>	<code>\</code>
-10^{10}	<code>\num{-e10}</code>	

`\numrange` `\numrange[options]{number1}{number2}`

Simple ranges of numbers can be handled using the `\numrange` function. This acts in the same way as `\num`, but inserts a phrase or other text between the two entries. This function will not accept multi-part numbers.

10 to 30	<code>\numrange{10}{30}</code>
----------	--------------------------------

4.3 Units

`\si` `\si[options]{unit}`

The symbol for a unit can be typeset using the `\si` macro: this provides full control over output format for the unit. Like the `\num` macro, `\si` takes one optional and one mandatory argument. The unit formatting system can accept two types of input. When `<unit>` contains one or more literal items, the output is processed in the same manner as with the `sistyle` package. Sub- and superscripts can be input without concern over maths mode, and the tokens `.` and `~` are converted into inter-unit separators.

kg m/s^2	<code>\si{kg.m/s^2}</code>	<code>\</code>
$\text{g}_{\text{polymer}} \text{mol}_{\text{cat}} \text{s}^{-1}$	<code>\si{g_{polymer}~mol_{cat}.s^{-1}}</code>	

The second operation mode for the `\si` macro is an “interpreted” system. Here, each unit, SI multiple prefix and power is given a macro name. These are entered in a method very similar to the reading of the unit name in English.

```

\si{kilo\gram\metre\per\square\second} \\
\si{gram\per\cubic\centi\metre} \\
\si{square\volt\cubic\lumen\per\farad} \\
\si{metre\squared\per\gray\cubic\lux} \\
\si{henry\second}

```

```

kg m s-2
g cm-3
V2 lm3 F-1
m2 Gy-1 lx3
H s

```

On its own, this is less convenient than the direct method, although it does use meaning rather than appearance for input. However, the the package allows you to define new unit macros; a large number of pre-defined abbreviations are also supplied. More importantly, by defining macros for units, instead of literal values, new functionality is made available. Units may be re-defined to give different output, and handling of reciprocal values can be altered.

`\SI` [*options*] {*number*} [*preunit*] {*unit*}

Very often, numbers and values are given together. Mathematically, these form a single entity, and should be separated by a non-breaking space. The `\SI` macro combines the functionality of `\num` and `\si`, and makes this both possible and easy. The *number* and *unit* arguments work exactly like those for the `\num` and `\si` macros, respectively. *preunit* is a unit to be typeset *before* the numerical value (most likely to be a currency).

```

\SI[mode=text]{1.23}{J.mol-1.K-1} \\
\SI{.23e7}{\candela} \\
\SI[per-mode=symbol]{1.99}[\$]{\per\kilogram} \\
\SI[per-mode=fraction]{1,345}{\coulomb\per\mole}

```

```

1.23 J mol-1 K-1
(0.23 × 107) cd
$1.99 /kg
1.345  $\frac{C}{mol}$ 

```

It is possible set up the unit macros to be available outside of the `\SI` and `\si` functions. This is not the standard behaviour as there is the risk of name clashes (for example, `\bar` is used by other packages, and several packages define `\degree`). Full details of using “stand alone” units are found in Section [5.7](#).

4.4 The unit macros

The package always defines the basic set of SI units with macro names. This includes the base SI units, the derived units with special names and the prefixes. A small number of powers are also given pre-defined names. Full details of units in the SI are available on-line [\[1\]](#).

Table 1: SI base units

Unit	Macro	Symbol
ampere	<code>\ampere</code>	A
candela	<code>\candela</code>	cd
kelvin	<code>\kelvin</code>	K
kilogram	<code>\kilogram</code>	kg
metre	<code>\metre</code>	m
mole	<code>\mole</code>	mol
second	<code>\second</code>	s

Table 2: Coherent derived units in the SI with special names and symbols

Unit	Macro	Symbol	Unit	Macro	Symbol
becquerel	<code>\becquerel</code>	Bq	newton	<code>\newton</code>	N
degree Celsius	<code>\degreeCelsius</code>	°C	ohm	<code>\ohm</code>	Ω
coulomb	<code>\coulomb</code>	C	pascal	<code>\pascal</code>	Pa
farad	<code>\farad</code>	F	radian	<code>\radian</code>	rad
gray	<code>\gray</code>	Gy	siemens	<code>\siemens</code>	S
hertz	<code>\hertz</code>	Hz	sievert	<code>\sievert</code>	Sv
henry	<code>\henry</code>	H	steradian	<code>\steradian</code>	sr
joule	<code>\joule</code>	J	tesla	<code>\tesla</code>	T
katal	<code>\katal</code>	kat	volt	<code>\volt</code>	V
lumen	<code>\lumen</code>	lm	watt	<code>\watt</code>	W
lux	<code>\lux</code>	lx	weber	<code>\weber</code>	Wb

`\meter` The seven base SI units are always defined (Table 1). In addition, the macro `\meter` is available as an alias for `\metre`, for users of US spellings. The full details of the base units are given in the SI Brochure [2].

`\celsius` The SI also lists a number of units which have special names and symbols [3]: these are listed in Table 2. As a short-cut for the degree Celsius, the unit `\celsius` is defined equivalent to `\degreeCelsius`.

In addition to the official SI units, `siunitx` also provides macros for a number of units which are accepted for use in the SI although they are not SI units. Table 3 lists the “accepted” units [5]. Some units are fundamental physical quantities, and these are non-SI but can be used with in the SI (Table 4, [6]). There are also a set of non-SI units which are used in certain defined circumstances (Table 5), although they are not necessarily official sanctioned [7].

`\deka` In addition to the units themselves, `siunitx` provides pre-defined macros for all of the SI prefixes (Table 6, [4]). The spelling “`\deka`” is provided for US users as an alternative to `\deca`.

Table 3: Non-SI units accepted for use with the International System of Units

Unit	Macro	Symbol
day	<code>\day</code>	d
degree	<code>\degree</code>	°
hectare	<code>\hectare</code>	ha
hour	<code>\hour</code>	h
litre	<code>\litre</code>	l
	<code>\liter</code>	L
minute (plane angle)	<code>\arcminute</code>	'
minute (time)	<code>\minute</code>	min
second (plane angle)	<code>\arcsecond</code>	"
second (time)	<code>\second</code>	s
tonne	<code>\tonne</code>	t

Table 4: Non-SI units whose values in SI units must be obtained experimentally

Unit	Macro	Symbol
astronomical unit	<code>\astronomicalunit</code>	ua
atomic mass unit	<code>\atomicmassunit</code>	u
bohr	<code>\bohr</code>	a_0
speed of light	<code>\clight</code>	c_0
dalton	<code>\dalton</code>	Da
electron mass	<code>\electronmass</code>	m_e
electronvolt	<code>\electronvolt</code>	eV
elementary charge	<code>\elementarycharge</code>	e
hartree	<code>\hartree</code>	e_h
reduced Planck constant	<code>\planckbar</code>	\hbar

Table 5: Other non-SI units

Unit	Macro	Symbol
ångström	<code>\angstrom</code>	Å
bar	<code>\bar</code>	bar
barn	<code>\barn</code>	b
bel	<code>\bel</code>	B
decibel	<code>\decibel</code>	dB
knot	<code>\knot</code>	kn
millimetre of mercury	<code>\mmHg</code>	mmHg
nautical mile	<code>\nauticalmile</code>	M
neper	<code>\neper</code>	Np

Table 6: SI prefixes

Prefix	Macro	Symbol	Power	Prefix	Macro	Symbol	Power
yocto	<code>\yocto</code>	y	-24	deca	<code>\deca</code>	da	1
zepto	<code>\zepto</code>	z	-21	hecto	<code>\hecto</code>	h	2
atto	<code>\atto</code>	a	-18	kilo	<code>\kilo</code>	k	3
femto	<code>\femto</code>	f	-15	mega	<code>\mega</code>	M	6
pico	<code>\pico</code>	p	-12	giga	<code>\giga</code>	G	9
nano	<code>\nano</code>	n	-9	tera	<code>\tera</code>	T	12
micro	<code>\micro</code>	μ	-6	peta	<code>\peta</code>	P	15
milli	<code>\milli</code>	m	-3	exa	<code>\exa</code>	E	18
centi	<code>\centi</code>	c	-2	zetta	<code>\zetta</code>	Z	21
deci	<code>\deci</code>	d	-1	yotta	<code>\yotta</code>	Y	24

`\square` A small number of pre-defined powers are provided as macros. `\square` and `\cubic` are intended for use before units, with `\squared` and `\cubed` going after the unit.

`\cubic`

`\cubed`

Bq^2

$\text{J}^2 \text{lm}^{-1}$

$\text{lx}^3 \text{VT}^3$

`\si{\square\becquerel} \\\`

`\si{\joule\squared\per\lumen} \\\`

`\si{\cubic\lux\volt\tesla\cubed}`

`\tothe` Generic powers can be inserted on a one-off basis using the `\tothe` and `\raiseto` macros. These are the only macros for units which take an argument:

`\raiseto`

H^5

$\text{rad}^{4.5}$

`\si{\henry\tothe{5}} \\\`

`\si{\raiseto{4.5}\radian}`

`\per` Reciprocal powers are indicated using the `\per` macro. This applies to the next unit only, unless the `sticky-per` option is turned on.

$\text{J mol}^{-1} \text{K}^{-1}$

$\text{J mol}^{-1} \text{K}$

H^{-5}

Bq^{-2}

`\si{\joule\per\mole\per\kelvin} \\\`

`\si{\joule\per\mole\kelvin} \\\`

`\si{\per\henry\tothe{5}} \\\`

`\si{\per\square\becquerel}`

4.5 Creating new macros

The various macro components of a unit have to be defined before they can be used. The package supplies a number of common definitions, but new definitions are also possible. As the definition of a logical unit should remain the same in a single document, these creation functions are all preamble-only.

`\DeclareSIUnit` `\DeclareSIUnitWithOptions`
`\DeclareSIUnit` `\DeclareSIUnitWithOptions`

`\DeclareSIUnit` `\DeclareSIUnitWithOptions`

New units are produced using the `\DeclareSIUnit` macro. `\SI` and `\si` can be used to override the settings for the unit. A typical example is the `\degree` unit.

`3.1415°` `\SI{3.1415}{\degree}`

This is declared in the package as:

```

\DeclareSIUnitWithOptions\degree{\SIUnitSymbolDegree}
{number-unit-separator={}}

```

The spacing can still be altered at point of use:

```

\SI{67890}{\degree} \ \
\SI[number-unit-separator = \;]{67890}{\degree}
67 890°
67 890 °

```

`\DeclareSIPrefix` `\DeclareBinaryPrefix`
`\DeclareSIPrefix` `\DeclareBinaryPrefix`

`\DeclareSIPrefix` `\DeclareBinaryPrefix`

The standard SI powers of ten are defined by the package, and are described above. However, the user can define new prefixes with `\DeclareSIPrefix`. The `\DeclareBinaryPrefix` function is also available for creating binary prefixes, with the same syntax (`\powers-ten` being replaced by `\powers-two`). For example, `\kilo` and `\kibi` are defined:

```

\DeclareSIPrefix\kilo{k}{3}
\DeclareBinaryPrefix\kibi{Ki}{10}

```

`\DeclareSIPostPower` `\DeclareSIPrePower`
`\DeclareSIPostPower` `\DeclareSIPrePower`

`\DeclareSIPostPower` `\DeclareSIPrePower`

These create power macros to appear before or after the unit they apply to. For example, the preamble to a document might contain:

```

\DeclareSIPrePower\quartic{4}
\DeclareSIPostPower\tothefourth{4}

```

with the functions then used in the document as:

```

kg4 \si{\kilogram\tothefourth}\ \
m4 \si{\quartic\metre}

```

`\DeclareSIQualifier` Following the syntax of the other macros, qualifiers are created with the syntax `\DeclareSIQualifier{<qualifier>}{<symbol>}`. In contrast to the other parts of a unit, there are no pre-defined qualifiers. It is therefore entirely up to the user to create these. For example, to identify the mass of a product created when using a particular catalyst, the preamble could contain:

```
\DeclareSIQualifier\polymer{pol}
\DeclareSIQualifier\catalyst{cat}
```

and then in the body the document could read:

```
\SI{1.234}{\gram\polymer\per\mole\catalyst\per\hour}
1.234 gpol molcat-1 h-1
```

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.

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 `input-digits` key) 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.

Switch These are on–off switches, and recognise `true` and `false`. 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.

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.

Table 7: Font detection options.

Option name	Type	Default
<code>detect-all</code>	Meta	<code><none></code>
<code>detect-bold</code>	Switch	<code>false</code>
<code>detect-display-maths</code>	Switch	<code>false</code>
<code>detect-family</code>	Switch	<code>false</code>
<code>detect-inline-bold</code>	Choice	<code>text</code>
<code>detect-italic</code>	Switch	<code>false</code>
<code>detect-mode</code>	Switch	<code>false</code>
<code>detect-none</code>	Meta	<code><none></code>

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 body fonts. However, the package can detect and follow surrounding bold, italic and font family changes. The font detection options are summarised in Table 7.

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

`detect-inline-bold` Bold detection is influenced by the value of `detect-inline-bold`, which takes values
`text`, `maths` and `combined`. The package can detect the local value of bold for either
the surrounding text, or the surrounding inline (`$. . . $`) maths. The `combined` option
checks both and uses bold if either test is true.

```

\sisetup{
  detect-bold      = true,
  detect-inline-bold = maths
}%
1234
1234
1234
1234
1234
\sisetup{detect-inline-bold = text}
{ \boldmath $\num{1234}$ } \\
{ \bfseries $\num{1234}$ } \\

```

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

Table 8: Font options (also available as `number-...` and `unit-...` versions).

Option name	Type	Default
<code>colour</code>	Literal	<code><i>none</i></code>
<code>maths-rm</code>	Macro	<code>\mathrm</code>
<code>maths-sf</code>	Macro	<code>\mathsf</code>
<code>maths-tt</code>	Macro	<code>\mathtt</code>
<code>mode</code>	Choice	<code>maths</code>
<code>text-rm</code>	Macro	<code>\rmfamily</code>
<code>text-sf</code>	Macro	<code>\sffamily</code>
<code>text-tt</code>	Macro	<code>\ttfamily</code>

```

\sfamily
Some text
\sisetup{
  detect-all,
  detect-display-maths = true
}
\[ x = \SI{1.2e3}{\kilogram\kelvin\candela} \]
More text
\sisetup{detect-display-maths = false}
\[ y = \SI{3}{\metre\second\mole} \]
Some text

$$x = (1.2 \times 10^3) \text{ kg K cd}$$

More text

$$y = 3 \text{ m s mol}$$


```

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.

- `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 `detect-mode` switch is `true`.
- `maths-rm` 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.
- `maths-sf` If font family detection is active, `siunitx` may be using a sans serif or monospaced font for output.
- `maths-tt` In `maths` mode, these are stored in `maths-sf` and `maths-tt`, and for `text` mode
- `text-sf` in `text-sf` and `text-tt`. Notice that the detected and output font families can differ.
- `text-tt`

```

1234          \sisetup{detect-family}%
1234          \num{1234} \\
1234          { \sffamily \num{1234} } \\
99 m          \SI{99}{\metre} \\
99 m          \sisetup{maths-rm = \mathtt}%
99 m          \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

```

\sisetup{
  maths-rm = \mathsf,
  text-rm = \sffamily
}

```

given all output in sans serif font without font detection.

colour The colour of printed output can be set using the colour option. When no colour is given, printing follows the surrounding text. In contrast, when a specific colour is given, it is used irrespective of the surroundings. As there are a number of different colour models available, it is left to user to load color or a more powerful colour package.

```

\color{red}
Some text \\
\SI{4}{\metre\per\sievert} \\
More text \\
\SI[colour = blue]{4}{\metre\per\sievert} \\
Still red here!

```

```

Some text
4 mSv-1
More text
4 mSv-1
Still red here!

```

Every one of the font options can be given independently for units and number, with the prefixes unit- and number-, respectively. This allows fine control of output.

```

140 dB
4 Å

```

```

\sisetup{
  number-math-rm = \mathtt,
  number-text-rm = \ttfamily,
  unit-math-rm = \mathsf,
  unit-text-rm = \sffamily
}
\SI{140}{\decibel} \\
\SI[number-colour=green]{4}{\angstrom}

```

Table 9: Options for number parsing

Option name	Type	Default
<code>input-close-uncertainty</code>	Literal)
<code>input-complex-roots</code>	Literal	ij
<code>input-decimal-markers</code>	Literal	. ,
<code>input-digits</code>	Literal	0123456789
<code>input-exponent-markers</code>	Literal	dDeE
<code>input-ignore</code>	Literal	<i>none</i>
<code>input-open-uncertainty</code>	Literal	(
<code>input-protect-tokens</code>	Literal	<code>\mp\pi\pm</code>
<code>input-signs</code>	Literal	<code>+-\pm\mp</code>
<code>input-symbols</code>	Literal	<code>\pi</code>
<code>parse-numbers</code>	Switch	true

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 9.

<code>input-digits</code>	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 <code>input-digits</code> , <code>input-decimal-markers</code> and <code>input-signs</code> , 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 <code>exponent markers</code> option.
<code>input-decimal-markers</code>	
<code>input-signs</code>	
<code>input-exponent-markers</code>	
<code>input-ignore</code>	As well as “normal” digits, the package will interpret symbolic “numbers” (such as <code>\pi</code>) correctly if they are included in the <code>input-symbols</code> list. Tokens given in the <code>input-ignore</code> list are totally passed over by <code>siunitx</code> : they will be removed from the input with no further processing.
<code>input-symbols</code>	
<code>input-open-uncertainty</code>	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 <code>input-open-uncertainty</code> and <code>input-close-uncertainty</code> .
<code>input-close-uncertainty</code>	
<code>input-complex-roots</code>	When using complex numbers in input, the complex root ($\sqrt{-1}$) is indicated by one of the tokens stored in <code>input-complex-roots</code> .
<code>input-protect-tokens</code>	Some symbols can be problematic under expansion in $\text{\LaTeX} 2_{\epsilon}$. To allow these to be used in input without issue, the package can protect these tokens while expanding input. Symbols to be protected in this way should be listed in <code>input-protect-tokens</code> .
<code>parse-numbers</code>	The <code>parse-numbers</code> option turns the entire parsing system on and off. The option is made available for two reasons. First, if all of the numbers in a document are to be reproduced “as given”, turning off the parser will represent a significant saving in

Table 10: Number post-processing options

Option name	Type	Default
<code>add-zero-decimal</code>	Switch	<code>true</code>
<code>add-zero-integer</code>	Switch	<code>true</code>
<code>explicit-sign</code>	Literal	<code>\langle none \rangle</code>
<code>retain-explicit-plus</code>	Switch	<code>false</code>
<code>retain-unity-mantissa</code>	Switch	<code>true</code>
<code>retain-zero-exponent</code>	Switch	<code>false</code>
<code>round-mode</code>	Choice	<code>off</code>
<code>round-figures</code>	Number	<code>2</code>
<code>round-places</code>	Number	<code>2</code>

processing required. Second, it allows the use of arbitrary \TeX code in numbers. If the parser is turned off, the input will be printed in maths mode (requiring `\text` to protect any text in the number).

```
\num[parse-numbers = false]{\sqrt{2}}      \\  
\SI[parse-numbers = false]{\sqrt{3}}{\metre}
```

$$\sqrt{2}$$

$$\sqrt{3}\text{m}$$

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 10.

`round-mode` 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. No rounding will take place if the number contains an uncertainty component.

	<code>\num{1.23456} \\</code>
	<code>\num{14.23} \\</code>
	<code>\num{0.12345(9)} \\</code>
	<code>\ssetup{</code>
	<code>round-mode = places,</code>
	<code>round-places = 3</code>
	<code>}%</code>
1.23456	<code>\num{1.23456} \\</code>
14.23	<code>\num{14.23} \\</code>
0.12345(9)	<code>\num{0.12345(9)} \\</code>
1.235	<code>\ssetup{</code>
14.230	<code>round-mode = figures,</code>
0.12345(9)	<code>round-figures = 3</code>
1.23	<code>}%</code>
14.2	<code>\num{1.23456} \\</code>
0.12345(9)	<code>\num{14.23} \\</code>
	<code>\num{0.12345(9)}</code>

`add-zero-decimal` It is possible to give real (floating point) numbers as input omitting the decimal
`add-zero-integer` 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.

	<code>\num{123.} \\</code>
	<code>\num{456} \\</code>
	<code>\num{.789} \\</code>
	<code>\ssetup{</code>
	<code>add-zero-decimal = false,</code>
	<code>add-zero-integer = false,</code>
	<code>}%</code>
123.0	<code>\num{123.} \\</code>
456	<code>\num{456} \\</code>
0.789	<code>\num{.789}</code>
123	
456	
.789	

`explicit-sign` The inclusion of a leading plus sign is usually unnecessary for positive numbers, and
`retain-explicit-plus` 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 `explicit-sign` option: this is used if given and if no sign was
present in the input.

345	<code>\num{+345} \\</code>
345	<code>\num[retain-explicit-plus]{+345} \\</code>
345	<code>\num[explicit-sign = -]{345}</code>

`retain-unity-mantissa` The retention of a zero exponent is controlled by the `retain-zero-exponent` option.
`retain-zero-exponent` The retention of a mantissa of one is likewise controlled by the `retain-unity-mantissa`
option.

Table 11: Output options for numbers

Option name	Type	Default
close-bracket	Literal)
complex-root-after-number	Switch	true
exponent-base	Literal	10
exponent-product	Maths	\times
group-decimal-digits	Switch	true
group-digits	Switch	true
group-four-digits	Switch	false
group-integer-digits	Switch	true
group-separator	Maths	\,
negative-colour	Literal	\langle none \rangle
open-bracket	Literal	(
output-close-uncertainty	Literal)
output-complex-root	Maths	\mathrm{i}
output-decimal-marker	Maths	.
output-open-uncertainty	Literal	(
range-phrase	Literal	_to_
separate-uncertainty	Switch	false
tight-spacing	Switch	false
uncertainty-separator	Maths	\langle none \rangle
use-brackets	Switch	true

```

\num{1e4} \\
\num[retain-unity-mantissa = false]{1e4} \\
\num{444e0} \\
\num[retain-zero-exponent = true]{444e0}

```

```

1 × 104
104
444
444 × 100

```

5.5 Printing numbers

Actually printing 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 11.

group-digits Grouping digits into blocks of three is a common method to increase the ease of reading
group-decimal-digits of numbers. The group-digits choice turns this behaviour on and off, with grouping
group-integer-digits for numbers of exactly four digits controlled by the group-four-digits choice. Note
group-four-digits that the later only applies if group-digits is turned on. The separator used between
group-separator

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[group-digits = false]{12345} \\
\num{1234} \\
\num[group-four-digits = true]{1234} \\
\num{12345} \\
\num[group-separator = {,}]{12345} \\
\num[group-separator = \text{~}]{12345}

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

Grouping can be activated separately for the integer and decimal parts of a number using the `group-integer-digits` and `group-decimal-digits` options.

```
\sisetup{group-digits = false}%
\num{12345.67890} \\
\num[group-decimal-digits]{12345.67890} \\
\num[group-integer-digits]{12345.67890}

12345.67890
12345.678 90
12 345.67890
```

`output-complex-root`
`output-decimal-marker`

The decimal marker used in output is set using the `output-decimal-marker` option. This can differ from the input marker, as can the root of $\sqrt{-1}$, which is stored in the `output-complex-root` option. The later is always in maths mode, but the standard setting uses `\mathrm` to give an upright “i”: this can easily be altered.

```
\num{1.23} \\
\num[output-decimal-marker = {,}]{1.23} \\
\num{1+2i} \\
\num[output-complex-root = \text{\ensuremath{i}}]{1+2i}

1.23
1,23
1 + 2i
1 + 2i
```

`complex-root-after-number`

The position of the complex root can be adjusted to place it either before or after the associated numeral in a complex number using the `complex-root-after-number` option.

```

\num{67-0.9i} \\
\num[complex-root-after-number = false]{67-0.9i}
67 - 0.9i
67 - i0.9

```

`exponent-base` When exponents are present in the input, the `exponent-base` and `exponent-product`
`exponent-product` options 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[exponent-product = \times]{1e2} \\
\num[exponent-product = \cdot]{1e2} \\
\num[exponent-base = 2]{1e2}
1 \times 10^2
1 \cdot 10^2
1 \times 2^2

```

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

```

\num{1.234(5)} \\
\num[separate-uncertainty = true]{1.234(5)} \\
\sisetup{
  output-open-uncertainty = [,
  output-close-uncertainty = ],
  uncertainty-separator = {\,}
}
\num{1.234(5)}
1.234(5)
1.234 ± 0.005
1.234 [5]

```

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

```

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

```

Table 12: Multi-part number options

Option name	Type	Default
<code>fraction-function</code>	Macro	<code>\frac</code>
<code>input-product</code>	Literal	<code>x</code>
<code>input-quotient</code>	Literal	<code>/</code>
<code>output-product</code>	Maths	<code>\times</code>
<code>output-quotient</code>	Maths	<code>/</code>
<code>product-mode</code>	Choice	<code>symbol</code>
<code>quotient-mode</code>	Choice	<code>symbol</code>

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

`negative-colour` `siunitx` can detect negative mantissa values and alter print colour accordingly. This is disabled by setting the option to an empty value.

$15\,673$ `\num{-15673} \\\`
 $15\,673$ `\num[negative-colour = red]{-15673}`

`tight-spacing` Under some circumstances it may be desirable to “squeeze” the output spacing. This is turned on using the `tight-spacing` switch, which compresses spacing where possible.

`\num{1 \pm 2i e3} \\\`
`\num[tight-spacing = true]{1 \pm 2i e3}`
 $(1 \pm 2i) \times 10^3$
 $(1\pm 2i)\times 10^3$

`range-phrase` Ranges of numbers can be given as input. These will have an appropriate word or symbol inserted between the two entries: this is stored using the `range-phrase` option. The phrase should include any necessary spaces: no extra space is added.

5 to 100 `\numrange{5}{100} \\\`
 5 – 100 `\numrange[range-phrase = --]{5}{100}`

5.6 Multi-part numbers

`siunitx` recognises the idea of products and quotients in numbers, both with and without units. These multi-part numbers have a number of options affecting how they are processed. The options are summarised in Table 12.

`input-product` `input-quotient` The options `input-product` and `input-quotient` contain the tokens used to determine if a number contains multiple parts.

$1 \times 2 \times 3$	<code>\num{1 x 2 x 3} \\</code>
$1 \times 10^4 \times 2(3) \times 3/4$	<code>\num{1e4 x 2(3) x 3/4} \\</code>
$4 \times 5 \times 6$	<code>\num[input-product=*]{4 * 5 * 6} \\</code>
$1/(2 \times 10^4)$	<code>\num{ 1 / 2e4 } \\</code>
$(1 \times 10^2)/(3 \times 10^4)$	<code>\num{ 1e2 / 3e4 }</code>

`output-product` The symbols used for printing products and quotients are stored using the options
`output-quotient` `output-product` and `output-quotient`.

```
\num[output-product = \cdot]{4.87 x 5.321 x 6.90545} \\
\num[output-quotient = \text{ div }]{1 / 2}
```

4.87 · 5.321 · 6.90545
1 div 2

`product-mode` For quotients, there is the possibility to print output either using a slash, or using the
`quotient-mode` `\frac` macro. This is controlled by the `quotient choice` option, which takes values
slash and fraction.

```
\num{1 / 2e4} \\
\num[quotient-mode = fraction]{1 / 2e4}
```

$1/(2 \times 10^4)$
 $\frac{1}{2 \times 10^4}$

`fraction-function` The function used when `quotient-mode = fraction` is set is determined by the
`fraction-function` option. This should be set to a function which takes two argu-
ments, and presumably creates some type of fraction. Most alternatives to the standard
`\frac` function will involve loading addition packages: the demonstrations here need
`amsmath` and `xfrac`.¹

$\frac{1}{1} \frac{1}{2} \frac{1}{3} \frac{1}{4}$	<code>\ssetup{quotient-mode = fraction}</code>
	<code>\num{1 / 1}</code>
	<code>\num[fraction-function=\dfrac]{1 / 2}</code>
	<code>\num[fraction-function=\sfrac]{1 / 3}</code>
	<code>\num[fraction-function=\tfrac]{1 / 4}</code>

5.7 Creating units

The various macro units are created at the start of the document. These can be defined only inside the `\si` and `\SI` macros, or can also be made available in the document body. There are a number of settings which control this creation process (Table 13). As a result, these options all apply in the preamble only.

`free-standing-units` The `free-standing-units` option controls whether the unit macros exist outside of the
`overwrite-functions` `overwrite-functions`

¹If `xfrac` is not available when typesetting this document, the demonstration of `\sfrac` will have the wrong appearance.

Table 13: Unit creation options

Option name	Type	Default
<code>free-standing-units</code>	Switch	false
<code>overwrite-functions</code>	Switch	false
<code>space-before-unit</code>	Switch	false
<code>unit-optional-argument</code>	Switch	false
<code>use-xspace</code>	Switch	false

`\si` and `\SI` arguments. When this option is true, `siunitx` creates the macros for general use. The standard method to achieve this does not overwrite any existing macros: this behaviour can be altered using the `overwrite-functions` switch.

`space-before-unit` When “free standing” unit macros are created, their behaviour can be adjusted by
`unit-optional-argument` a number of options. These are mainly intended for emulating the input syntax of
`use-xspace` older packages. The option `unit-optional-argument` gives the same behaviour for the
inputs

`\SI{10}{\metre}`

and

`\metre[10]`.

The `space-before-unit` and `use-xspace` options control the behaviour at the “ends” of the unit macros. Activating `space-before-unit` inserts the number–unit space before the unit is printed. This is suitable for the input syntax

`30\metre`

but does mean that the unit macros are incorrectly spaced in running text. On the other hand, the `use-xspace` option attempts to correctly space input such as

`\metre` is the symbol for metres.

5.8 Loading additional units

`load-configurations` There are a number of additional unit definitions supplied with `siunitx` that are not loaded as standard. Some of these are specialist units for certain subject areas, others are not accepted with the SI and some are for the users convenience. A comma-separated list of configurations to load should be given as the argument to the `load-configurations` option.

The first group of add-on units are those which provide either abbreviations or combined names for SI units.

`\bit` Binary data requires the units bits and bites, and the special prefixes which use powers
`\byte`

Table 14: Abbreviated units
(load-configurations=abbreviations)

Unit	Abbreviation	Symbol	Unit	Abbreviation	Symbol
picoampere	\pA	pA	hertz	\Hz	Hz
nanoampere	\nA	nA	millihertz	\mHz	mHz
microampere	\uA	μ A	kilohertz	\kHz	kHz
milliampere	\mA	mA	megahertz	\MHz	MHz
kiloampere	\kA	kA	gigahertz	\GHz	GHz
terahertz	\THz	THz			
picomole	\pmol	pmol	kilovolt	\kV	kV
nanomole	\nmol	nmol	millivolt	\mV	mV
micromole	\umol	μ mol	millilitre	\ml	ml
millimole	\mmol	mmol	microlitre	\ul	μ l

Table 15: Binary prefixes
(load-configurations=binary)

Prefix	Macro	Symbol	Power
kibi	\kibi	Ki	10
mebi	\mebi	Mi	20
gibi	\gibi	Gi	30
tebi	\tebi	Ti	40
pebi	\pebi	Pi	50
exbi	\exbi	Ei	60
zebi	\zebi	Zi	70
yobi	\yobi	Yi	80

of two rather than of ten (Table 15) have been defined for this purpose. Appropriate unit prefixes, along with the units \bit and \byte are available for use in the computing field.

```
\SI{100}{\mebi\byte} \\
\SI[prefixes-as-symbols=false]{30}{\kibi\bit}
100 MiB
 $30 \times 2^{10}$  bit
```

Units for specialist areas are given in Tables 16 to 19. These are not SI units, but are defined here as they are in common usage in the subject areas concerned.

Table 16: Astronomy-related units
(load-configurations=astronomy)

Unit	Macro	Symbol
parsec	\parsec	pc
lightyear	\lightyear	ly

Table 17: Chemical engineering units
(load-configurations=chemical-engineering)

Unit	Macro	Symbol
gram-mole	\gmol	g-mol
kilogram-mole	\kgmol	kg-mol
pound-mole	\lbmol	lb-mol

Table 18: Chemistry-related units
(load-configurations=chemistry)

Unit	Macro	Symbol
molar	\molar	mol dm ⁻³
	\Molar	M
torr	\torr	Torr

Table 19: Geophysics-related units
(load-configurations=geophysics)

Unit	Macro	Symbol
gon	\gon	gon

Table 20: Unit output options

Option name	Type	Default
<code>forbid-literal-units</code>	Switch	<code>false</code>
<code>inter-unit-separator</code>	Maths	<code>\,</code>
<code>per-mode</code>	Choice	<code>reciprocal</code>
<code>per-symbol</code>	Maths	<code>/</code>
<code>prefixes-as-symbols</code>	Switch	<code>true</code>
<code>qualifier-mode</code>	Choice	<code>subscript</code>
<code>sticky-per</code>	Switch	<code>false</code>

5.9 Using units

Part of the power of `siunitx` is the ability to alter the output format for units without changing the input. The behaviour of units is therefore controlled by a number of options which alter either the processing of units or the output directly (Table 20).

`forbid-literal-units` Some users may prefer to completely disable the use of literal input in units, for example to enforce consistency. This can be accomplished by setting the `forbid-literal-units` switch. With this option enabled, only macro-based units can be used in a document.

`inter-unit-separator` The separator between each unit is stored using the `inter-unit-separator` option. The standard setting is a thin space: another common choice is a centred dot. To get the correct spacing it is necessary to use `{ }\cdot{ }` in the later case.

```
\si{\farad\squared\lumen\candela} \\
\si[inter-unit-separator={ }\cdot{ }]{\farad\squared\lumen\candela}
```

```
F2 lm cd
F2 · lm · cd
```

`per-mode` The handling of `\per` is altered using the `per-mode` choice option. The standard setting is reciprocal, meaning that `\per` generates reciprocal powers for units. Setting the option to `fraction` uses the `\frac` function to typeset the positive and negative powers of a unit separately.

```
\si{\joule\per\mole\per\kelvin} \\
\si{\metre\per\second\squared} \\
\si[per-mode=fraction]{\joule\per\mole\per\kelvin} \\
\si[per-mode=fraction]{\metre\per\second\squared}
```

```
J mol-1 K-1
m s-2

$$\frac{\text{J}}{\frac{\text{mol}}{\text{m}} \text{K} \frac{\text{s}^2}{\text{s}^2}}$$

```

It is possible to use a symbol (usually /) to separate the two parts of a unit by setting `per-mode` to `symbol`; the symbol used is stored using the setting `per-symbol`. This method for displaying units can be ambiguous, and so brackets are added unless `use-brackets` is set to `false`. The output for `per-symbol` is always made in maths mode, and so `\text` will be needed to print textual information.

```
\sisetup{per-mode=symbol}%
\si{\joule\per\mole\per\kelvin} \\
\si{\metre\per\second\squared} \\
\si[per-symbol=\text{-div-}]{\joule\per\mole\per\kelvin} \\
\si[use-brackets=false]{\joule\per\mole\per\kelvin}

J/(molK)
m/s2
J div (molK)
J/molK
```

Finally, the often-requested (but mathematically invalid) `repeated-symbol` option is also available to repeat the symbol for each `\per`.

```
\si[per-mode=repeated-symbol]{\joule\per\mole\per\kelvin}

J/mol/K
```

`sticky-per` By default, `\per` applies only to the next unit given.² By setting the `sticky-per` flag, this behaviour is changed so that `\per` applies to all subsequent units.

```
\si{\pascal\per\gray\henry} \\
\si[sticky-per]{\pascal\per\gray\henry}

Pa Gy-1 H
Pa Gy-1 H-1
```

`qualifier-mode` Unit qualifiers can be printed in three different formats, set by the `qualifier-mode` option. The standard setting is subscript, while the options `brackets` and `space` are also possible. With the last settings, powers can lead to ambiguity and are automatically detected and brackets added as appropriate.

```
\si{\kilogram\polymer\squared\per\mole\catalyst\per\hour} \\
\si[qualifier-mode=brackets]
  {\kilogram\polymer\squared\per\mole\catalyst\per\hour} \\
\si[qualifier-mode=space]
  {\kilogram\polymer\squared\per\mole\catalyst\per\hour}

kgpol2 molcat-1 h-1
kg(pol)2 mol(cat)-1 h-1
(kg pol)2 (mol cat)-1 h-1
```

`prefixes-as-symbols` The unit prefixes (`\kilo`, *etc.*) are normally given as letters. However, the package can

²This is the standard method of reading units in English: for example, $\text{J mol}^{-1} \text{K}^{-1}$ is pronounced “joules per mole per kelvin”.

Table 21: Options for numbers with units

Option name	Type	Default
allow-number-unit-breaks	Switch	false
number-unit-separator	Maths	\,
repeat-units	Switch	true
unit-product-to-power	Switch	false

convert these into numerical powers. This is controlled by the `prefixes-as-symbols` switch option.

```
\si{\milli\litre\per\mole\deci\ampere} \\
\SI{10}{\kilo\gram\squared\deci\second} \\
\si[prefixes-as-symbols=false]{\milli\litre\per\mole\deci\ampere} \\
\SI[prefixes-as-symbols=false]{10}{\kilo\gram\squared\deci\second}

ml mol-1 dA
10 kg2 ds
10-4 l mol-1 A
10 × 105 g2 s
```

5.10 Numbers with units

Some options apply to the combination of units and numbers, rather than to units or numbers alone (Table 21).

`allow-number-unit-breaks` Usually, the combination of a number and unit is regarded as a single mathematical entity which should not be split across lines. However, there are cases (very long units, narrow columns, *etc.*) where breaks may be needed. This can be turned on using the `allow-number-unit-breaks` option.

```
Some filler text \begin{minipage}{2.6 cm}
10 m Some filler text \SI{10}{\metre} \\
Some filler text 10 \ssetup{allow-number-unit-breaks}
m Some filler text \SI{10}{\metre} \\
\end{minipage}
```

`number-unit-separator` The separator between the number and units is stored using the `inter-unit-separator` option.

```
\SI{2.67}{\farad} \\
\SI[number-unit-separator=\text{~}]{2.67}{\farad} \\
\SI[number-unit-separator=]{2.67}{\farad}

2.67 F
2.67 F
2.67F
```

`repeat-units` When brackets are not in use to make the meaning of number–unit combinations clear, it may be necessary to repeat units to maintain the mathematical meaning of the output. This is handled by the `repeat-units` option, which may be true or false. At the same time, *simple* products (areas and volumes) may be given with the repetition of the unit converted into a power. This behaviour is set by the `unit-product-to-power` option, which will only act if units are not being printed repeatedly.

```

\SI{10 x 20 x 30}{\metre} \\
\SI[unit-product-to-power]{10 x 20 x 30}{\metre} \\

\sisetup{use-brackets = false}
\SI{10 x 20 x 30}{\metre} \\
\SI[unit-product-to-power]{10 x 20 x 30}{\metre} \\
\SI[repeat-units=false]{10 x 20 x 30}{\metre} \\
\SI[repeat-units=false,unit-product-to-power]
  {10 x 20 x 30}{\metre} \\

\sisetup{separate-uncertainty}
\SI{12.3(4)}{\kilo\gram} \\
\SI[repeat-units=false]{12.3(4)}{\kilo\gram}

(10 × 20 × 30) m
(10 × 20 × 30) m3
10 m × 20 m × 30 m
10 m × 20 m × 30 m
10 × 20 × 30 m
10 × 20 × 30 m3
12.3 kg ± 0.4 kg
12.3 ± 0.4 kg

```

5.11 Symbols

Most units use letters as the symbol for the unit, and these are all very easy to control. However, a small number of units use other symbols, and matching these to the body text requires more work. `siunitx` provides appropriate symbols for commonly-used units, but the definitions may need adjustment depending on the body font used in a document.

`redefine-symbols` The package provides one general option for the handling of symbols. If the packages `textcomp` or `upgreek` are loaded, symbols can be taken from these for units, rather than using the `siunitx` default values. The switch `redefine-symbols` can be used to turn this behaviour on or off: the standard setting is true.

The individual symbols are set up independently for maths and text output, and are summarised in Table 22. Many of the definitions are variations using `\text` or `\ensuremath` to produce the correct output, as the symbols available in the document may vary considerably. In the case of the micro symbol (μ), `siunitx` provides a suitable low-level definition for the symbol. Depending on the fonts available, this may need to

Table 22: Symbol options

Option name	Type	Default
<code>maths-angstrom</code>	Literal	<code>\text{\AA}</code>
<code>maths-arcminute</code>	Literal	<code>{ }^{\prime}</code>
<code>maths-arcsecond</code>	Literal	<code>{ }^{\prime\prime}</code>
<code>maths-celsius</code>	Literal	<code>{ }^{\circ}</code> <code>\kern -\scriptspace \mathrm{C}</code>
<code>maths-degree</code>	Literal	<code>{ }^{\circ}</code>
<code>maths-micro</code>	Literal	<i>(see text)</i>
<code>maths-ohm</code>	Literal	<code>\Omega</code>
<code>redefine-symbols</code>	Switch	<code>true</code>
<code>text-angstrom</code>	Literal	<code>\AA</code>
<code>text-arcminute</code>	Literal	<code>\ensuremath{{ }^{\prime}}</code>
<code>text-arcsecond</code>	Literal	<code>\ensuremath{{ }^{\prime\prime}}</code>
<code>text-celsius</code>	Literal	<code>\ensuremath{{ }^{\circ}}</code> <code>\kern -\scriptspace \text{C}</code>
<code>text-degree</code>	Literal	<code>\ensuremath{{ }^{\circ}}</code>
<code>text-micro</code>	Literal	<i>(see text)</i>
<code>text-ohm</code>	Literal	<code>\ensuremath{\Omega}</code>

be replaced by an alternative by the user. The ohm symbol (Ω) is usually set to `\Omega`, but will check that this has not been redefined as a slanted letter. If `\Omega` has been redefined, an alternative definition is used.

`\SIUnitSymbolAngstrom`
`\SIUnitSymbolArcminute`
`\SIUnitSymbolArcsecond`
`\SIUnitSymbolCelsius`
`\SIUnitSymbolDegree`
`\SIUnitSymbolMicro`
`\SIUnitSymbolOhm`

The maths and text symbols defined above are wrapped up into mode independent functions with user names. These are then used in the definitions of the appropriate units. For example, the micro symbol can be accessed using the macro `\SIUnitSymbolMicro`. Notice that these names capitalise the unit name (to make reading the macro name easier!).³

6 Usage tips

6.1 Ensuring text or maths output

The macros `\ensuremath` and `\text` should be used to ensure that a particular item is always printed in the desired mode. Some mathematical output does not work well in `\mathrm` (the standard font used by `siunitx` for printing). The easiest way to solve this is to use the construction `\text{\ensuremath{...}}`, which will print the material in the standard mathematics font without affecting the rest of the output. In some cases, simply forcing `\mathnormal` will suffice, but this is less reliable with non-Latin characters.

³The function `\SIUnitSymbolAngstrom` uses the name without accents.

6.2 Using units such as $\mu\text{m s}^{-1}$ in headings

The `siunitx` code is designed to work correctly with functions in headings. They will print correctly in headings and in the table of contents. As illustrated here, the standard behaviour is to ignore font changes. When the `hyperref` package is loaded, the functions automatically “degrade gracefully” to produce useful information in PDF bookmarks. If you want more control over the bookmark text, use the `\texorpdfstring` function from `hyperref`, for example:

```
\section{Some text
  \texorpdfstring
    {\si{\joule\per\mole\per\kelvin}}
    {J mol-1 K-1}%
}
```

References

- [1] *The International System of Units (SI)*, <http://www.bipm.org/en/si/>.
- [2] *SI base units*, http://www.bipm.org/en/si/si_brochure/chapter2/2-1/.
- [3] *Units with special names and symbols; units that incorporate special names and symbols*, http://www.bipm.org/en/si/si_brochure/chapter2/2-2/2-2-2.html.
- [4] *SI Prefixes*, http://www.bipm.org/en/si/si_brochure/chapter3/prefixes.html.
- [5] *Non-SI units accepted for use with the International System of Units*, http://www.bipm.org/en/si/si_brochure/chapter4/table6.html.
- [6] *Non-SI units whose values in SI units must be obtained experimentally*, http://www.bipm.org/en/si/si_brochure/chapter4/table7.html.
- [7] *Other non-SI units*, http://www.bipm.org/en/si/si_brochure/chapter4/table8.html.

Change History

vo.6		v1.1
General: First public testing release (as si)	1	General: Package extended to a greater range of unit types
v1.0		v1.2
General: First official release	1	General: Correct handling for ranges of

numbers added	1	symbol	1
v1.3		v2.0	
General: Better definition for micro		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.

A		\day	7
add-zero-decimal (option)	16	\deca	8
add-zero-integer (option)	16	\deci	8
allow-number-unit-breaks (option)	27	\decibel	7
\ampere	6	\DeclareBinaryPrefix	9
\angstrom	7	\DeclareSIPostPower	9
\arcminute	7	\DeclareSIPrefix	9
\arcsecond	7	\DeclareSIPrePower	9
\astronomicalunit	7	\DeclareSIQualifier	10
\atomicmassunit	7	\DeclareSIUnit	9
\atto	8	\DeclareSIUnitWithOptions	9
B		\degree	7
\bar	7	\degreeCelsius	6
\barn	7	\deka	6
\becquerel	6	detect-bold (option)	11
\bel	7	detect-display-maths (option)	11
\bit	22	detect-family (option)	11
\bohr	7	detect-inline-bold (option)	11
\byte	22	detect-italic (option)	11
C		detect-mode (option)	11
\candela	6	E	
\celsius	6	\electronmass	7
\centi	8	\electronvolt	7
\clight	7	\elementarycharge	7
close-bracket (option)	19	\exa	8
colour (option)	13	\exbi	23
complex-root-after-number (option)	18	explicit-sign (option)	16
\coulomb	6	exponent-base (option)	19
\cubed	8	exponent-product (option)	19
\cubic	8	F	
D		\farad	6
\dalton	7	\femto	8
		forbid-literal-units (option)	25

<code>fraction-function</code> (option)	21	<code>\kilo</code>	8
<code>free-standing-units</code> (option)	21	<code>\kilogram</code>	6
G			
<code>\GHz</code>	23	<code>\knot</code>	7
<code>\gibi</code>	23	<code>\kV</code>	23
<code>\giga</code>	8	L	
<code>\gmol</code>	24	<code>\lbmol</code>	24
<code>\gon</code>	24	<code>\lightyear</code>	24
<code>\gray</code>	6	<code>\liter</code>	7
<code>group-decimal-digits</code> (option)	17	<code>\litre</code>	7
<code>group-digits</code> (option)	17	<code>load-configurations</code> (option)	22
<code>group-four-digits</code> (option)	17	<code>\lumen</code>	6
<code>group-integer-digits</code> (option)	17	<code>\lux</code>	6
<code>group-separator</code> (option)	17	M	
H			
<code>\hartree</code>	7	<code>\mA</code>	23
<code>\hectare</code>	7	<code>maths-rm</code> (option)	12
<code>\hecto</code>	8	<code>maths-sf</code> (option)	12
<code>\henry</code>	6	<code>maths-tt</code> (option)	12
<code>\hertz</code>	6	<code>\mebi</code>	23
<code>\hour</code>	7	<code>\mega</code>	8
<code>\Hz</code>	23	<code>\meter</code>	6
I			
<code>input-close-uncertainty</code> (option)	14	<code>\metre</code>	6
<code>input-complex-roots</code> (option)	14	<code>\MHz</code>	23
<code>input-decimal-markers</code> (option)	14	<code>\mHz</code>	23
<code>input-digits</code> (option)	14	<code>\micro</code>	8
<code>input-exponent-markers</code> (option)	14	<code>\milli</code>	8
<code>input-ignore</code> (option)	14	<code>\minute</code>	7
<code>input-open-uncertainty</code> (option)	14	<code>\ml</code>	23
<code>input-product</code> (option)	20	<code>\mmHg</code>	7
<code>input-protect-tokens</code> (option)	14	<code>\mmol</code>	23
<code>input-quotient</code> (option)	20	<code>mode</code> (option)	12
<code>input-signs</code> (option)	14	<code>\Molar</code>	24
<code>input-symbols</code> (option)	14	<code>\molar</code>	24
<code>inter-unit-separator</code> (option)	25	<code>\mole</code>	6
J			
<code>\joule</code>	6	<code>\mV</code>	23
K			
<code>\kA</code>	23	N	
<code>\katal</code>	6	<code>\nA</code>	23
<code>\kelvin</code>	6	<code>\nano</code>	8
<code>\kgmol</code>	24	<code>\nauticalmile</code>	7
<code>\kHz</code>	23	<code>negative-colour</code> (option)	20
<code>\kibi</code>	23	<code>\neper</code>	7
O			
		<code>\newton</code>	6
		<code>\nmol</code>	23
		<code>\num</code>	4
		<code>number-unit-separator</code> (option)	27
		<code>\numrange</code>	4
		<code>\ohm</code>	6

<code>\pmol</code>	23	<code>sticky-per</code> (option)	26
<code>prefixes-as-symbols</code> (option)	26		
<code>product-mode</code> (option)	21		
		T	
Q		<code>\tebi</code>	23
<code>qualifier-mode</code> (option)	26	<code>\tera</code>	8
<code>quotient-mode</code> (option)	21	<code>\tesla</code>	6
		<code>text-rm</code> (option)	12
R		<code>text-sf</code> (option)	12
<code>\radian</code>	6	<code>text-tt</code> (option)	12
<code>\raiseto</code>	8	<code>\THz</code>	23
<code>range-phrase</code> (option)	20	<code>tight-spacing</code> (option)	20
<code>redefine-symbols</code> (option)	28	<code>\tonne</code>	7
<code>repeat-units</code> (option)	28	<code>\torr</code>	24
<code>retain-explicit-plus</code> (option)	16	<code>\tothe</code>	8
<code>retain-unity-mantissa</code> (option)	16		
<code>retain-zero-exponent</code> (option)	16	U	
<code>round-figures</code> (option)	15	<code>\uA</code>	23
<code>round-mode</code> (option)	15	<code>\ul</code>	23
<code>round-places</code> (option)	15	<code>\umol</code>	23
		<code>uncertainty-separator</code> (option)	19
S		<code>unit-optional-argument</code> (option)	22
<code>\second</code>	6, 7	<code>unit-product-to-power</code> (option)	28
<code>separate-uncertainty</code> (option)	19	<code>use-brackets</code> (option)	19
<code>\SI</code>	5	<code>use-xspace</code> (option)	22
<code>\si</code>	4		
<code>\siemens</code>	6	V	
<code>\sievert</code>	6	<code>\volt</code>	6
<code>\sisetup</code>	10		
<code>\SIUnitSymbolAngstrom</code>	29	W	
<code>\SIUnitSymbolArcminute</code>	29	<code>\watt</code>	6
<code>\SIUnitSymbolArcsecond</code>	29	<code>\weber</code>	6
<code>\SIUnitSymbolCelsius</code>	29		
<code>\SIUnitSymbolDegree</code>	29	Y	
<code>\SIUnitSymbolMicro</code>	29	<code>\yobi</code>	23
<code>\SIUnitSymbolOhm</code>	29	<code>\yocto</code>	8
<code>space-before-unit</code> (option)	22	<code>\yotta</code>	8
<code>\square</code>	8		
<code>\squared</code>	8	Z	
<code>\steradian</code>	6	<code>\zebi</code>	23
		<code>\zepto</code>	8
		<code>\zetta</code>	8