siunitx — A comprehensive (SI) units package*

Joseph Wright†

Released 2017/06/13

Abstract

Physical quantities have both numbers and units, and each physical quantity should be expressed as the product of a number and a unit. Typesetting physical quantities requires care to ensure that the combined mathematical meaning of the number–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 quantities 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

*This file describes v2.7h, last revised 2017/06/13.

†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* (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 LaTeX users to typeset numbers and units 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 LaTeX3 support as provided in the l3kernel and l3packages bundles. Both of these are available on CTAN as ready-to-install zip files. Suitable versions are available in MiKTeX 2.9 and TeX Live 2015 (updating the relevant packages online may be necessary). LaTeX3, and so siunitx, requires the ε-TeX extensions: these are available on all modern TeX 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
- cancel
- caption
- cleveref
- colortbl
- csquotes
- helvet
- mathpazo
- multirow
- listings
- pgfplots
- xcolor

3 siunitx for the impatient

The package provides the user macros:

- `\ang[`⟨*options*⟩`]{`⟨*angle*⟩`}`
- `\num[`⟨*options*⟩`]{`⟨*number*⟩`}`
- `\si[`⟨*options*⟩`]{`⟨*unit*⟩`}`
- `\SI[`⟨*options*⟩`]{`⟨*number*⟩`}[`⟨*pre-unit*⟩`]{`⟨*unit*⟩`}`
- `\numlist[`⟨*options*⟩`]{`⟨*numbers*⟩`}`
- `\numrange[`⟨*options*⟩`]{`⟨*numbers*⟩`}{`⟨*number2*⟩`}`
- `\SIlist[`⟨*options*⟩`]{`⟨*numbers*⟩`}{`⟨*unit*⟩`}`
- `\SIrange[`⟨*options*⟩`]{`⟨*number1*⟩`}{`⟨*number2*⟩`}{`⟨*unit*⟩`}`
- `\sisetup{`⟨*options*⟩`}`
- `\tablenum[`⟨*options*⟩`]{`⟨*number*⟩`}`

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

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

$12\,345.678\,90$
$1 \pm 2\mathrm{i}$
0.3×10^{45}
$1.654 \times 2.34 \times 3.430$

```
\num{12345,67890} \\
\num{1+-2i}       \\
\num{.3e45}       \\
\num{1.654 x 2.34 x 3.430}
```

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

```
\si{kg.m.s^{-1}}                \\
\si{\kilogram\metre\per\second} \\
\si[per-mode=symbol]
  {\kilogram\metre\per\second}  \\
\si[per-mode=symbol]
  {\kilogram\metre\per\ampere\per\second}
```

kg m s^{-1}
kg m s^{-1}
kg m/s
kg m/(A s)

Simple lists and ranges of numbers can be handled.

```
\numlist{10;20;30}                   \\
\SIlist{0.13;0.67;0.80}{\milli\metre} \\
\numrange{10}{20}                    \\
\SIrange{0.13}{0.67}{\milli\metre}
```

10, 20 and 30
0.13 mm, 0.67 mm and 0.80 mm
10 to 20
0.13 mm to 0.67 mm

By default, all text is typeset in the current upright math 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 LaTeX 2ε way.

```
\usepackage{siunitx}
```

The key–value options described later in this document can be used when loading the package, for example

```
\usepackage[version-1-compatibility]{siunitx}
```

to use options from version 1 of the package.

4.2 Numbers

`\num` 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 both 'soft' (␣) and 'hard' spaces (`\,` and `~`), automatically identifies exponents (by default marked using `e`, `E`, `d` or `D`) and adds the appropriate spacing of large numbers. With the standard settings a leading zero is added before a decimal marker, if needed: both '`.`' and '`,`' are recognised as decimal markers.

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

Note that numbers are parsed before typesetting, which does have a performance overhead (only obvious with very large amounts of numerical input). The parser understands a range of input syntaxes, as demonstrated above.

`\numlist` Lists of numbers may be processed using the `\numlist` function. Each ⟨*number*⟩ is given within the list of ⟨*numbers*⟩ within a brace pair, as the list can have a flexible length. This function should be used in text mode, a common feature of all of the list and range functions provided by siunitx.[1]

10, 30, 50 and 70	`\numlist{10;30;50;70}`

`\numrange` 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 should be used in text mode.

10 to 30	`\numrange{10}{30}`

`\ang` Angles can be typeset using the `\ang` command. The ⟨*angle*⟩ can be given either as a decimal number or as a semi-colon separated list of degrees, minutes and seconds, which is called 'arc format' in this document. The numbers which make up an angle are processed using the same system as other numbers.

10°	`\ang{10} \\`
12.3°	`\ang{12.3} \\`
4.5°	`\ang{4,5} \\`
1°2′3″	`\ang{1;2;3} \\`
1″	`\ang{;;1} \\`
10°	`\ang{+10;;} \\`
−0°1′	`\ang{-0;1;}`

4.3 Units

`\si` 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 the ⟨*si*⟩ contains literal items (for example letters or numbers) then siunitx converts `.` and `~` into inter-unit product and correctly positions sub- and superscripts specified using `_` and `^`. The formatting methods will work with both math and text mode.

$\mathrm{kg\,m/s^2}$	`\si{kg.m/s^2} \\`
$\mathrm{g_{polymer}\,mol_{cat}\,s^{-1}}$	`\si{g_{polymer}~mol_{cat}.s^{-1}}`

[1] The reason for this restriction is that the separators between items may involve text-mode spaces which must be able to vanish at line breaks. It is not possible to achieve this effect from inside math mode.

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}
```

$\mathrm{kg\,m\,s^{-2}}$
$\mathrm{g\,cm^{-3}}$
$\mathrm{V^{2}\,lm^{3}\,F^{-1}}$
$\mathrm{m^{2}\,Gy^{-1}\,lx^{3}}$
$\mathrm{H\,s}$

On its own, this is less convenient than the direct method, although it does use meaning rather than appearance for input. However, the 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 input, new functionality is made available. By altering the settings used by the package, the same input can yield a variety of different output formats. For example, the \per macro can give reciprocal powers, slashes or be used to construct units as fractions.

\SI Very often, numbers and units are given together. Formally, the value of a quantity is the product of the number and the unit, the space being regarded as a multiplication sign [9]. The \SI macro combines the functionality of \num and \si, and makes this both possible and easy. The ⟨*number*⟩ and ⟨*si*⟩ 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
$0.23 \times 10^{7}\,\mathrm{cd}$
\$1.99/kg
$1.345\,\frac{\mathrm{C}}{\mathrm{mol}}$

It is possible to 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 5.10.

\SIlist Lists of numbers with units can be handled using the \SIlist function. The behaviour of this function is similar to \numlist, but with the addition of a unit to each number. This function should be used in text mode.

10 m, 30 m and 45 m `\SIlist{10;30;45}{\metre}`

\SIrange Ranges of numbers with units can be handled using the \SIrange function. The behaviour of this function is similar to \numrange, but with the addition of a unit to each number. This function should be used in text mode.

10 m to 30 m `\SIrange{10}{30}{\metre}`

Table 1 – SI base units.

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

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

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

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].

\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 [3].

\celsius

The SI also lists a number of units which have special names and symbols [4]: these are listed in Table 2. As a short-cut for the degree Celsius, the unit \celsius is defined equivalent to \degreeCelsius .

\percent

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 [6]. Some units are fundamental physical quantities, and these are non-SI but can be used within the SI (Table 4, [7]). There are also a set of non-SI units which are used in certain defined circumstances (Table 5), although they are not necessarily officially sanctioned [8]. The package also predefines the \percent macro. While this is not a unit, it is used in the same way in many cases and is therefore provided on the grounds of realism.

\deka

In addition to the units themselves, siunitx provides pre-defined macros for all

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

Unit	Macro	Symbol
day	`\day`	d
degree	`\degree`	°
hectare	`\hectare`	ha
hour	`\hour`	h
litre	`\litre`	l
	`\liter`	L
minute (plane angle)	`\arcminute`	′
minute (time)	`\minute`	min
second (plane angle)	`\arcsecond`	″
tonne	`\tonne`	t

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

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

Table 5 – Other non-SI units.

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

Table 6 – SI prefixes.

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

of the SI prefixes (Table 6, [5]). The spelling '\deka' is provided for US users as an alternative to \deca.

\square \squared \cubic \cubed

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.

Bq^2
$\mathrm{J}^2\,\mathrm{lm}^{-1}$
$\mathrm{lx}^3\,\mathrm{V}\,\mathrm{T}^3$

```
\si{\square\becquerel} \\
\si{\joule\squared\per\lumen} \\
\si{\cubic\lux\volt\tesla\cubed}
```

\tothe \raiseto

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:

H^5
$\mathrm{rad}^{4.5}$

```
\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.

$\mathrm{J}\,\mathrm{mol}^{-1}\,\mathrm{K}^{-1}$
$\mathrm{J}\,\mathrm{mol}^{-1}\,\mathrm{K}$
H^{-5}
Bq^{-2}

```
\si{\joule\per\mole\per\kelvin} \\
\si{\joule\per\mole\kelvin} \\
\si{\per\henry\tothe{5}} \\
\si{\per\square\becquerel}
```

\of

As for generic powers, generic qualifiers are also available using the \of function:

```
\si{\kilogram\of{metal}} \\
\SI[qualifier-mode = brackets]
  {0.1}{\milli\mole\of{cat}\per\kilogram\of{prod}}
```

$\mathrm{kg}_{\mathrm{metal}}$
$0.1\,\mathrm{mmol(cat)}\,\mathrm{kg(prod)}^{-1}$

\cancel \highlight

If the cancel package is loaded, it is possible to 'cancel out' units using the \cancel macro. This applies to the next unit, in a similar manner to a prefix. The \highlight macro is also available to selectively colour units. Both \cancel and \highlight are of course outside of the normal semantic meaning of units, but are provided as they may be useful in some cases.

```
\si[per-mode = fraction]
  {\cancel\kilogram\metre\per\cancel\kilogram\per\second} \\
\si{\highlight{red}\kilogram\metre\per\second} \\
\si[unit-color = purple]
  {\highlight{red}\kilogram\metre\per\second}
```

$\frac{\cancel{\mathrm{kg}}\,\mathrm{m}}{\cancel{\mathrm{kg}}\,\mathrm{s}}$
kg m s^{-1}
kg m s^{-1}

When using the unit macros, the package is able to validate the input given. As part of this, stand-alone unit prefixes can be used with the \si macro

```
\si{\kilo}  \\
\si{\micro} \\
\si[prefixes-as-symbols = false]{\kilo}
```

k
µ
10^3

However, the package only allows a single prefix to be used in this way: multiple prefixes will give an error, as will trying to give a number without a unit. So the following will raise errors:

```
\si{\kilo\gram\micro} \\
\SI{10}{\micro}
```

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 New units are produced using the \DeclareSIUnit macro. ⟨*symbol*⟩ can contain literal input, other units, multiple prefixes, powers and \per, although literal text should not be intermixed with unit macros. Units can be created with ⟨*options*⟩ from the usual list understood by siunitx, and apply the specific unit macro only. The (first) optional argument to \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:

```
\DeclareSIUnit[number-unit-product = {}]
  \degree{\SIUnitSymbolDegree}
```

The spacing can still be altered at point of use:

```
\SI{67890}{\degree} \\
\SI[number-unit-product = \,]{67890}{\degree}
```

67 890°
67 890 °

The meaning of a pre-defined unit can be altered by using `\DeclareSIUnit` after loading siunitx. This will overwrite the original definition with the newer version.

`\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` 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:

kg^4	`\si{\kilogram\tothefourth}\\`
m^4	`\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}
```

$1.234\,\mathrm{g_{pol}\,mol_{cat}^{-1}\,h^{-1}}$

4.6 Tabular material

Aligning numbers in tabular content is handled by a new column type, the `S` column. This new column type can align material using a number of different strategies, with the aim of flexibility of output without needing to alter the input. The method used as standard is to place the decimal marker in the number at the centre of the cell and to align the material appropriately (Table 7).

```
\begin{table}
  \caption{Standard behaviour of the \texttt{S} column type.}
  \label{tab:S:standard}
  \centering
  \begin{tabular}{S}
  \toprule
    {Some Values} \\
  \midrule
      2.3456 \\
     34.2345 \\
     -6.7835 \\
     90.473  \\
```

Table 7 – Standard behaviour of the S column type.

Some Values
2.3456
34.2345
-6.7835
90.473
5642.5
1.2×10^3
10^4

Table 8 – Detection of surrounding material in an S column.

Some Values
12.34
975.31
44.268[a]

```
    5642.5    \\
        1.2e3 \\
          e4  \\
  \bottomrule
  \end{tabular}
\end{table}
```

The S column will attempt to automatically detect material which should be placed before or after a number, and will maintain the alignment of the numerical data (Table 8). If the material could be mistaken for part of a number, it should be protected by braces. The use of \color in a table cell will also be detected and will override any general colour applied by siunitx.

```
\begin{table}
  \caption{Detection of surrounding material in an \texttt{S}
    column.}
  \label{tab:S:extras}
  \centering
  \begin{tabular}{S[color=orange]}
  \toprule
    {Some Values} \\
  \midrule
    12.34 \\
    \color{purple} 975,31 \\
    44.268 \textsuperscript{\emph{a}} \\
  \bottomrule
  \end{tabular}
\end{table}
```

\tablenum Within more complex tables, aligned numbers may be desirable within the argu-

Table 9 – Controlling complex alignment with the \tablenum macro.

Heading	Heading
Info	More info
Info	More info
12.34	
333.5567	
4563.21	

Heading	Heading
88.999	aaa
	bbb
33.435	ccc
	ddd

ment of \multicolumn or \multirow.[2] The \tablenum function is available to achieve alignment in these situations: this is, in effect, a macro version of the S macro (Table 9).

```
\begin{table}
  \caption{Controlling complex alignment with the \cs{tablenum}
    macro.}
  \label{tab:tablenum}
  \centering
  \begin{tabular}{lr}
    \toprule
    Heading & Heading \\
    \midrule
    Info & More info \\
    Info & More info \\
    \multicolumn{2}{c}{\tablenum[table-format = 4.4]{12,34}}    \\
    \multicolumn{2}{c}{\tablenum[table-format = 4.4]{333.5567}} \\
    \multicolumn{2}{c}{\tablenum[table-format = 4.4]{4563.21}}  \\
    \bottomrule
  \end{tabular}
  \hfil
  \begin{tabular}{lr}
    \toprule
    Heading & Heading \\
    \midrule
    \multirow{2}*{\tablenum{88,999}} & aaa \\
                                     & bbb \\
    \multirow{2}*{\tablenum{33,435}} & ccc \\
                                     & ddd \\
    \bottomrule
  \end{tabular}
\end{table}
```

As a complement to the S column type, siunitx also provides a second column type, s. This is intended for producing columns of units. This allows both macro-based and explicit units to be printed easily (Table 10).

```
\begin{table}
  \centering
  \caption{Units in tables.}
  \label{tab:s:demo}
  \begin{tabular}{s}
```

[2]Provided by the multirow package

Table 10 – Units in tables.

Unit
$\mathrm{m^2\,s^{-1}}$
Pa
$\mathrm{m\,s^{-1}}$

Table 11 – The s column processes everything.

Unit	Unit
$\mathrm{m^3}$	$\mathrm{m^3}$
kg	kg

```
    \toprule
      \multicolumn{1}{c}{Unit} \\
    \midrule
      \metre\squared\per\second \\
      \pascal \\
      m.s^{-1} \\
    \bottomrule
  \end{tabular}
\end{table}
```

As the `\si` macro can take literal or macro input, the `s` column cannot validate the input. *Everything* in an `s` column is therefore passed to the `\si` macro for processing. To prevent this, you have to use `\multicolumn`, as is shown in Table 11. Notice that braces alone do not prevent processing and colouring of the cell contents.

```
\begin{table}
  \centering
  \caption{The \texttt{s} column processes everything.}
  \label{tab:s:processing}
  \sisetup{color = orange}
  \begin{tabular}{ss}
    \toprule
      {Unit}       & \multicolumn{1}{c}{Unit}\\
    \midrule
      {\si{m^3}} & \multicolumn{1}{c}{\si{m^3}} \\
      \kilogram    & \kilogram \\
    \bottomrule
  \end{tabular}
\end{table}
```

5 Comprehensive details of package control options

5.1 The key–value control system

`\sisetup` The behaviour of the siunitx package is controlled by a number of key–value options.

Table 12 – Font detection options.

Option name	Type	Default
`detect-all`	Meta	⟨*none*⟩
`detect-display-math`	Switch	`false`
`detect-family`	Switch	`false`
`detect-inline-family`	Choice	`text`
`detect-inline-weight`	Choice	`text`
`detect-mode`	Switch	`false`
`detect-none`	Meta	⟨*none*⟩
`detect-shape`	Switch	`false`
`detect-weight`	Switch	`false`

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.

`Integer` Requires a number as the argument.

`Length` Requires a length, either as a literal value such as `2.0 cm`, or stored as a LaTeX length, or TeX dimension.

`Literal` A key which uses the value(s) given directly, either to check input (for example the `input-digits` key) or in output.

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

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

`Meta` These are options which actually apply a number of other options. As such, they do not take any value at all.

`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.

5.2 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 12.

`detect-weight` `detect-family` `detect-shape` `detect-mode`

The options `detect-weight` and `detect-shape` set detection of the prevailing font weight and font shape states, respectively. The font shape state is only checked if the surrounding material is not in math mode (as math text is always italic). The `detect-shape` option is an extension of the older `detect-italic` option, which is retained for backward compatibility. Detecting the current family (roman, sans serif or monospaced) is controlled by the `detect-family` setting, while the current mode (text or math) is detected using the `detect-mode` switch. For convenience, all of the preceding options can be turned on or off in one go using the `detect-all` and `detect-none` options. As the names indicate, `detect-all` sets all of `detect-weight`, `detect-family`, `detect-shape` and `detect-mode` to `true`, while `detect-none` sets all of them to `false`.

`detect-all`
`detect-none`

`detect-inline-family`
`detect-inline-weight`

When siunitx macros are used in in-line math, the detection of font weight and font family can be tuned using the `detect-inline-family` and `detect-inline-weight` options. Both of these take the choices `text` and `math`.

1234
1234
1234
1234
1234
5678
5678
5678
5678
5678

```
\sisetup{
  detect-family        = true,
  detect-inline-family = math
}%
$\num{1234}$ \\
{ \sffamily $\num{1234}$ } \\
$ \mathsf { \num{1234}} $  \\
 \sisetup{detect-inline-family = text}
{ \sffamily $\num{1234}$ } \\
$ \mathsf { \num{1234} } $ \\
\sisetup{
  detect-weight        = true,
  detect-inline-weight = math
}%
$\num{5678}$ \\
{ \boldmath $\num{5678}$ } \\
{ \bfseries $\num{5678}$ } \\
 \sisetup{detect-inline-weight = text}
{ \boldmath $\num{5678}$ } \\
{ \bfseries $\num{5678}$ }
```

`detect-display-math`

The font detection system can treat displayed mathematical content in two ways. This is controlled by the `detect-display-math` option. When set `true`, display mathematics is treated independently from the body of the document. Thus the local *math* font is checked for matching. In contrast, when set `false`, display material is treated with the current running text font.[3]

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

[3]Here, 'display' math means either typeset in TeX's display math mode or using the AMS display-like environments. Simply using `\displaystyle` will not make otherwise in line math be detected as display math.

Table 13 – Font options (also available as `number-...` and `unit-...` versions).

Option name	Type	Default
`color`	Literal	⟨*none*⟩
`math-rm`	Macro	`\mathrm`
`math-sf`	Macro	`\mathsf`
`math-tt`	Macro	`\mathtt`
`mode`	Choice	`math`
`text-rm`	Macro	`\rmfamily`
`text-sf`	Macro	`\sffamily`
`text-tt`	Macro	`\ttfamily`

Some text

$$x = 1.2 \times 10^{3}\,\mathrm{kg\,K\,cd}$$

More text

$$y = 3\,\mathsf{m\,s\,mol}$$

5.3 Font settings

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 (Table 13).

`mode` The `mode` option determines whether siunitx uses math or text mode when printing output. The choices are `math` and `text`. When using math mode, text is printed using a math 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`.

`math-rm` `math-sf` `math-tt` `text-rm` `text-sf` `text-tt` If font family detection is inactive, siunitx uses the font family stored in either `math-rm` or `text-rm` for output. The choice of `math` 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 math mode, these are stored in `math-sf` and `math-tt`, and for text mode in `text-sf` and `text-tt`. Notice that the detected and output font families can differ.

1234
1234
99 m
99 m

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

`color` The colour of printed output can be set using the `color` 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 such as xcolor.

Table 14 – Options for number parsing.

Option name	Type	Default
input-close-uncertainty	Literal	)
input-comparators	Literal	<=>\approx\ge\geq \gg\le\leq\ll \sim
input-complex-roots	Literal	ij
input-decimal-markers	Literal	.,
input-digits	Literal	0123456789
input-exponent-markers	Literal	dDeE
input-ignore	Literal	⟨*none*⟩
input-open-uncertainty	Literal	(
input-protect-tokens	Literal	\approx\dots\ge\geq\gg\le \leq\ll\mp\pi\pm\sim
input-signs	Literal	+-\pm\mp
input-uncertainty-signs	Literal	\pm
input-symbols	Literal	\pi\dots
parse-numbers	Switch	true

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

Some text
4 m Sv^{-1}
More text
4 m Sv^{-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.

```
\SI{4}{\angstrom} \\
\SI[number-color = green]{4}{\angstrom} \\
\SI[unit-color = green]{4}{\angstrom}
```

4 Å
4 Å
4 Å

5.4 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 14.

input-digits
input-decimal-markers
input-signs
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 input-digits, input-decimal-markers and input-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 `input-exponent-markers` option.

`input-symbols`
`input-ignore`

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

`input-comparators`

In addition to signs, siunitx can recognise comparators, such as `<`. The package will automatically carry out conversions for `«`, `»`, `<=` and `>=` to `\ll`, `\gg`, `\le` and `\ge`, respectively:

<10
$\gg 5\,\mathrm{m}$
≤ 0.12

```
\num{< 10} \\
\SI{>> 5}{\metre} \\
\num{\le 0.12}
```

`input-open-uncertainty`
`input-close-uncertainty`
`input-uncertainty-signs`

In some fields, it is common to give the uncertainty in a number 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 `input-open-uncertainty` and `input-close-uncertainty`. Alternatively, the uncertainty may be given as a separate part following a sign. Which signs are valid for this operation is determined by the `input-uncertainty-signs` option. As with other signs, the combination `+-` will automatically be converted to `\pm` internally.

9.99(9)
9.99(9)
9.99(9)
123.0(45)
12.3(60)

```
\num{9.99(9)}       \\
\num{9.99 +- 0.09}  \\
\num{9.99 \pm 0.09} \\
\num{123 +- 4.5}    \\
\num{12.3 +- 6}
```

`input-complex-roots`

When using complex numbers in input, the complex root ($\mathrm{i} = \sqrt{-1}$) is indicated by one of the tokens stored in `input-complex-roots`. The parser understands complex root symbols given either before or after the associated number (but will detect any invalid arrangement):

$9.99 + 88.8\mathrm{i}$
$9.99 + 88.8\mathrm{i}$

```
\num{9.99 + 88.8i} \\
\num{9.99 + i88.8}
```

`input-protect-tokens`

Some symbols can be problematic under expansion in LaTeX 2ε. 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 `input-protect-tokens`.

`parse-numbers`

The `parse-numbers` 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 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 math mode (requiring `\text` to protect any text in the number).

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

$\sqrt{2}$
$\sqrt{3}\,\mathrm{m}$

Table 15 – Number post-processing options.

Option name	Type	Default
add-decimal-zero	Switch	true
add-integer-zero	Switch	true
explicit-sign	Literal	⟨*none*⟩
fixed-exponent	Integer	0
minimum-integer-digits	Integer	0
omit-uncertainty	Switch	false
retain-explicit-plus	Switch	false
retain-unity-mantissa	Switch	true
retain-zero-exponent	Switch	false
round-half	Choice	up
round-integer-to-decimal	Switch	false
round-minimum	Literal	0
round-mode	Choice	off
round-precision	Integer	2
scientific-notation	Switch	false
zero-decimal-to-integer	Switch	false

5.5 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 15.

round-mode
round-precision

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 digits used (either decimal places or significant figures in the mantissa) is set using the `round-precision` option. No rounding will take place if the number contains an uncertainty component.

1.234 56
14.23
0.123 45(9)
1.235
14.230
0.123 45(9)
1.23
14.2
0.123 45(9)

```
\num{1.23456} \\
\num{14.23} \\
\num{0.12345(9)} \\
\sisetup{
  round-mode      = places,
  round-precision = 3
}%
\num{1.23456} \\
\num{14.23} \\
\num{0.12345(9)} \\
\sisetup{
  round-mode      = figures,
  round-precision = 3
}%
\num{1.23456} \\
\num{14.23} \\
\num{0.12345(9)}
```

round-integer-to-decimal

The standard settings for siunitx do not add a decimal part if none was given in the input. The round-integer-to-decimal option can be used to allow this conversion as part of the rounding process.

1
1
1.0
1.00

```
\num[round-mode = figures]{1} \\
\num[round-mode = places]{1} \\
\sisetup{round-integer-to-decimal}
\num[round-mode = figures]{1} \\
\num[round-mode = places]{1}
```

zero-decimal-to-integer

It may be desirable to convert decimals to integers if the decimal part is zero. This is set up using the zero-decimal-to-integer option.

2.0
2.1
2
2.1

```
\num{2.0} \\
\num{2.1} \\
\sisetup{zero-decimal-to-integer}
\num{2.0} \\
\num{2.1}
```

round-minimum

There are cases in which rounding will result in the number reaching zero. It may be desirable to show such results as below a threshold value. This can be achieved by setting round-minimum to the threshold value. There will be no effect when rounding to a number of significant figures as it is not possible to obtain the value zero in these cases.

0.01
0.00
0.01
$<$0.01

```
\sisetup{round-mode = places}%
\num{0.0055} \\
\num{0.0045} \\
\sisetup{round-minimum = 0.01}%
\num{0.0055} \\
\num{0.0045}
```

round-half

In cases where the rounded part of a number is exactly half, there are two common methods for 'breaking the tie'. The choice of method is determined by the option round-half, which recognises the choices up and even.

```
\sisetup{round-mode = places, round-half = up}%
\num{0.055} \\
\num{0.045} \\
\sisetup{round-half = even}%
\num{0.055} \\
\num{0.045}
```

0.06
0.05
0.06
0.04

add-decimal-zero
add-integer-zero

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-decimal-zero and add-integer-zero allow the package to 'fill in' the missing zero.

123.0
456
0.789
123.
456
.789

```
\num{123.} \\
\num{456}  \\
\num{.789} \\
\sisetup{
  add-decimal-zero = false,
  add-integer-zero = false,
}%
\num{123.} \\
\num{456}  \\
\num{.789}
```

minimum-integer-digits

Related is the minimum-integer-digits option. This applies only to the integer part of the mantissa, and ensures that it will contain at least the specified number of digits. This is achieved by padding with zeros if needed.

```
\num{123} \\
\num[minimum-integer-digits = 1]{123} \\
\num[minimum-integer-digits = 2]{123} \\
\num[minimum-integer-digits = 3]{123} \\
\num[minimum-integer-digits = 4]{123}
```

123
123
123
123
0123

explicit-sign
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. At 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
+345
−345
345

```
\num{+345} \\
\num[retain-explicit-plus]{+345} \\
\num[explicit-sign = -]{345}     \\
\num[explicit-sign = -]{+345}
```

retain-unity-mantissa
retain-zero-exponent

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

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

1×10^4
10^4
444
444×10^0

scientific-notation
fixed-exponent

Numbers can be converted to scientific notation by the package. This is controlled by the scientific-notation option, which takes choices false, true, fixed and engineering. The fixed setting will use the exponent value by the fixed-exponent option. When engineering is set, the exponent is always a power of three.

```
\num{0.001}   \\
\num{0.0100} \\
\num{1200}    \\
\sisetup{scientific-notation = true}%
\num{0.001}   \\
\num{0.0100} \\
\num{1200}    \\
\sisetup{scientific-notation = engineering}%
\num{0.001}   \\
\num{0.0100} \\
\num{1200}    \\
\sisetup{
  fixed-exponent      = 2,
  scientific-notation = fixed,
}%
\num{0.001}   \\
\num{0.0100} \\
\num{1200}
```

0.001
0.0100
1200
1×10^{-3}
1.00×10^{-2}
1.200×10^{3}
1×10^{-3}
10.0×10^{-3}
1.200×10^{3}
$0.000\,01 \times 10^{2}$
$0.000\,100 \times 10^{2}$
12.00×10^{2}

When used with a `fixed-exponent` of zero, this may be used to remove scientific notation from the input

```
\num{1.23e4} \\
\num[scientific-notation = fixed, fixed-exponent = 0]{1.23e4}
```

1.23×10^{4}
12 300

Note that these options apply after any removal of unit mantissa, zero exponent, *etc.*

`omit-uncertainty` The printing of an uncertainty can be suppressed entirely using the `omit-uncertainty` option.

0.01(2)
0.01

```
\num{0.01(2)} \\
\num[omit-uncertainty]{0.01(2)}
```

5.6 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 16.

`group-digits` `group-four-digits` `group-separator` Grouping digits into blocks of three is a common method to increase the ease of reading of numbers. The `group-digits` choice controls whether this behaviour applies,

Table 16 – Output options for numbers.

Option name	Type	Default
bracket-negative-numbers	Switch	false
bracket-numbers	Switch	true
close-bracket	Literal	)
complex-root-position	Choice	after-number
copy-complex-root	Choice	false
copy-decimal-marker	Choice	false
exponent-base	Literal	10
exponent-product	Math	\times
group-digits	Choice	true
group-minimum-digits	integer	5
group-separator	Literal	\,
negative-color	Literal	⟨*none*⟩
open-bracket	Literal	(
output-close-uncertainty	Literal	)
output-complex-root	Literal	\ensuremath{\mathrm{i}}
output-decimal-marker	Literal	.
output-exponent-marker	Literal	⟨*none*⟩
output-open-uncertainty	Literal	(
separate-uncertainty	Switch	false
tight-spacing	Switch	false
uncertainty-separator	Literal	⟨*none*⟩

and takes the values `true`, `false`, `decimal` and `integer`. Grouping can be activated separately for the integer and decimal parts of a number using the appropriately-named values.

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

12 345.678 90
12345.67890
12345.678 90
12 345.67890

The separator used between groups of digits is stored by the `group-separator` option. This takes literal input and may be used in math mode: for a text-mode full space use `\text{~}`.

```
\num{12345} \\
\num[group-separator = {,}]{12345} \\
\num[group-separator = \text{~}]{12345}
```

12 345
12,345
12 345

`group-minimum-digits` Grouping is not always applied to smaller numbers, and the option `group-minimum-digits` is available to specify how many digits must be present before grouping is applied. The number of digits is considered separately for the integer and decimal parts of the number: grouping does not 'cross the boundary'.

```
\num{1234} \\
\num[group-minimum-digits = 4]{1234} \\
\num{1234.5678} \\
\num[group-minimum-digits = 4]{1234.5678}
```

1234
1 234
1234.5678
1 234.567 8

output-complex-root
output-decimal-marker
copy-complex-root
copy-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 standard setting uses `\mathrm` in math mode to give an upright 'i': this can easily be altered. The complex root or decimal marker from the input can be used in the output by setting the `copy-complex-root` and `copy-decimal-marker` options, respectively.

```
\num{1.23} \\
\num[output-decimal-marker = {,}]{1.23} \\
\num{1+2i} \\
\num[output-complex-root = \text{\ensuremath{i}}]{1+2i} \\
\num[output-complex-root = j]{1+2i} \\
\num[copy-complex-root]{1+2j} \\
\num[copy-decimal-marker]{555,555}
```

1.23
1,23
$1 + 2\mathrm{i}$
$1 + 2i$
$1 + 2\mathrm{j}$
$1 + 2\mathrm{j}$
555,555

complex-root-position

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-position` option.

```
\num{67-0.9i} \\
\num[complex-root-position = before-number]{67-0.9i} \\
\num[complex-root-position = after-number]{67-0.9i}
```

$67 - 0.9\mathrm{i}$
$67 - \mathrm{i}0.9$
$67 - 0.9\mathrm{i}$

exponent-base
exponent-product

When exponents are present in the input, the `exponent-base` and `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 math mode.

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

1×10^{2}
$1 \cdot 10^{2}$
1×2^{2}

output-exponent-marker

Alternatively, if the output-exponent-marker option is set then the value stored will be used in place of the normal product and base combination. This will normally be set up to ensure math or text mode.

```
\num[output-exponent-marker = \text{e}]{1e2} \\
\num[output-exponent-marker = \ensuremath{\mathrm{E}}]{1e2}
```

1e2
1E2

separate-uncertainty
uncertainty-separator
output-open-uncertainty
output-close-uncertainty

When input is given including an uncertainty in a number, 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 number and the uncertainty: this is stored using the uncertainty-separator option. The opening and closing brackets used are stored in 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]

Notice that siunitx correctly interprets uncertainties which cross the decimal marker position whether these are separated out or not.

8.2(13)
8.2 ± 1.3

```
\num{8.2(13)} \\
\num[separate-uncertainty]{8.2(13)}
```

bracket-numbers
open-bracket
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 bracket-numbers switch. The opening and closing brackets used are stored in open-bracket and close-bracket, respectively. Note that bracket-numbers only applies to numbers without units: for numbers with units see the multi-part-units option.

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

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

negative-color

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

Table 17 – Multi-part number options.

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

−15 673
−15 673

```
\num{-15673} \\
\num[negative-color = red]{-15673}
```

bracket-negative-numbers

A common means to display negative numbers in financial situations is to place them in brackets. This can be carried out automatically using the bracket-negative-numbers option.

```
\num{-15673} \\
\num[bracket-negative-numbers]{-15673} \\
\SI{-10}{\metre} \\
\SI[bracket-negative-numbers]{-10}{\metre}
```

−15 673
(15 673)
−10 m
(10) m

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 \pm 2i e3} \\
\num[tight-spacing = true]{1 \pm 2i e3}
```

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

5.7 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 17.

input-product
input-quotient

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

$1 \times 2 \times 3$
$1 \times 10^4 \times 2(3) \times 3/4$
$4 \times 5 \times 6$
$1/(2 \times 10^4)$
$1 \times 10^2/(3 \times 10^4)$

```
\num{1 x 2 x 3} \\
\num{1e4 x 2(3) x 3/4} \\
\num[input-product=*]{4 * 5 * 6} \\
\num{ 1 / 2e4 } \\
\num{ 1e2 / 3e4 }
```

output-product
output-quotient

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

Table 18 – Output options for lists and ranges of numbers.

Option name	Type	Default
list-final-separator	Literal	␣and␣
list-pair-separator	Literal	␣and␣
list-separator	Literal	,␣
range-phrase	Literal	␣to␣

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

$4.87 \cdot 5.321 \cdot 6.905\,45$
1 div 2

quotient-mode

For quotients, there is the possibility to print output either using a slash, or using the \frac macro. This is controlled by the quotient choice option, which takes values symbol 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 arguments, and presumably creates some type of fraction. Most alternatives to the standard \frac function will involve loading additional packages: the demonstrations here need amsmath and xfrac.

```
\sisetup{quotient-mode = fraction}
\num{1 / 1}
\num[fraction-function = \dfrac]{1 / 2}
\num[fraction-function = \sfrac]{1 / 3}
\num[fraction-function = \tfrac]{1 / 4}
```

$\frac{1}{1}$ $\dfrac{1}{2}$ $^1/_3$ $\tfrac{1}{4}$

5.8 Lists and ranges of numbers

Lists and ranges of numbers have a small number of specialised options, which apply to these more unusual input forms (Table 18).

list-final-separator
list-pair-separator
list-separator

Lists of numbers are printed with a separator between each item, which is stored using the list-separator option. The separator before the last item of a list may be different, and is therefore set using the list-final-separator option. The separator used for exactly two items is set using the list-pair-separator option. Any spaces needed should be included in the option settings: none are added within the code. These items are printed in text mode.

Table 19 – Angle options.

Option name	Type	Default
add-arc-degree-zero	Switch	false
add-arc-minute-zero	Switch	false
add-arc-second-zero	Switch	false
angle-symbol-over-decimal	Switch	false
arc-separator	Literal	false
number-angle-product	Literal	⟨*empty*⟩

```
\numlist{0.1;0.2;0.3}                              \\
\numlist[list-separator = {; }]{0.1;0.2;0.3}       \\
\numlist[list-final-separator = {, }]{0.1;0.2;0.3} \\
\numlist[
  list-separator       = { and },
  list-final-separator = { and finally }
]{0.1;0.2;0.3} \\
\numlist{0.1;0.2} \\
\numlist[list-pair-separator = {, and }]{0.1;0.2}
```

0.1, 0.2 and 0.3
0.1; 0.2 and 0.3
0.1, 0.2, 0.3
0.1 and 0.2 and finally 0.3
0.1 and 0.2
0.1, and 0.2

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
5–100

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

For lists and ranges when a single unit is given, siunitx will automatically 'compress' exponents when a fixed exponent is in use.

$(1 \text{ to } 7) \times 10^3$ m
$(1, 2 \text{ and } 3) \times 10^3$ kg

```
\sisetup{
  fixed-exponent      = 3        ,
  list-units          = brackets ,
  range-units         = brackets ,
  scientific-notation = fixed
}%
\SIrange{1e3}{7e3}{\metre} \\
\SIlist{1e3;2e3;3e3}{\kg}
```

5.9 Angles

Angle processing provided by the \ang function has a set of options which apply in addition to the general ones set up for number processing (Table 19).

number-angle-product

The separator between the number and angle symbol (degrees, minutes or seconds) can be set using the number-angle-product option, independent of the related number-unit-product option used by the \SI function.

2.67°	`\ang{2.67} \\`
2.67 °	`\ang[number-angle-product = \,]{2.67}`

`arc-separator`

When angles are printed in arc format, the separation of the different parts is set up using the `arc-separator` option.

6°7′6.5″	`\ang{6;7;6.5} \\`
6° 7′ 6.5″	`\ang[arc-separator = \,]{6;7;6.5}`

`add-arc-degree-zero`
`add-arc-minute-zero`
`add-arc-second-zero`

Zero-filling for the degree, minute or second parts of an arc is controlled using the `add-arc-degree-zero`, `add-arc-minute-zero` and `add-arc-second-zero` options. All are off as standard.

```
\ang{-1;;} \\
\ang{;-2;} \\
\ang{;;-3} \\
\sisetup{add-arc-degree-zero}
\ang{-1;;} \\
\ang{;-2;} \\
\ang{;;-3} \\
\sisetup{add-arc-minute-zero}
\ang{-1;;} \\
\ang{;-2;} \\
\ang{;;-3} \\
\sisetup{add-arc-second-zero}
\ang{-1;;} \\
\ang{;-2;} \\
\ang{;;-3}
```

−1°
−2′
−3″
−1°
−0°2′
−0°3″
−1°0′
−0°2′
−0°0′3″
−1°0′0″
−0°2′0″
−0°0′3″

`angle-symbol-over-decimal`

In some subject areas, most notably astronomy, the angle symbols are given over the decimal marker, rather than at the end of the number. This behaviour is available using the `angle-symbol-over-decimal` option.

```
\ang{45.697}  \\
\ang{6;7;6.5} \\
\ang[angle-symbol-over-decimal]{45.697} \\
\ang[angle-symbol-over-decimal]{6;7;6.5}
```

45.697°
6°7′6.5″
45°.697
6°7′6″.5

5.10 Creating units

The various macro units are created at the start of the document. siunitx can define these such that they are only available for use within the `\si` and `\SI` functions, or can make the unit macros available throughout the document body. There are a number of settings which control this creation process (Table 20). As a result, these options all apply in the preamble only.

`free-standing-units`
`overwrite-functions`

The `free-standing-units` option controls whether the unit macros exist outside of the `\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`
`unit-optional-argument`
`use-xspace`

When 'free standing' unit macros are created, their behaviour can be adjusted by

Table 20 – Unit creation options.

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

a number of options. These are mainly intended for emulating the input syntax of 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.
```

5.11 Loading additional units

`abbreviations` As standard, siunitx loads a set of abbreviated versions of the SI units (Table 21). The standard siunitx settings only create these abbreviations within the scope of the `\si` and `\SI` functions, meaning that no clashes should occur (for example with the standard `\pm` symbol). Loading of these abbreviations can be prevented by setting the option `abbreviations = false` in the preamble.

Table 21 – Abbreviated units.

Unit	Abbreviation	Symbol
femtogram	\fg	fg
picogram	\pg	pg
nanogram	\ng	ng

Continued on next page

Continued from previous page

Unit	Abbreviation	Symbol
microgram	`\ug`	µg
milligram	`\mg`	mg
gram	`\g`	g
kilogram	`\kg`	kg
atomic mass unit	`\amu`	u
picometre	`\pm`	pm
nanometre	`\nm`	nm
micrometre	`\um`	µm
millimetre	`\mm`	mm
centimetre	`\cm`	cm
decimetre	`\dm`	dm
metre	`\m`	m
kilometre	`\km`	km
attosecond	`\as`	as
femtosecond	`\fs`	fs
picosecond	`\ps`	ps
nanosecond	`\ns`	ns
microsecond	`\us`	µs
millisecond	`\ms`	ms
second	`\s`	s
femtomole	`\fmol`	fmol
picomole	`\pmol`	pmol
nanomole	`\nmol`	nmol
micromole	`\umol`	µmol
millimole	`\mmol`	mmol
mole	`\mol`	mol
kilomole	`\kmol`	kmol
picoampere	`\pA`	pA
nanoampere	`\nA`	nA
microampere	`\uA`	µA
milliampere	`\mA`	mA
ampere	`\A`	A
kiloampere	`\kA`	kA
microlitre	`\ul`	µl
millilitre	`\ml`	ml
litre	`\l`	l

Continued on next page

Continued from previous page

Unit	Abbreviation	Symbol
hectolitre	`\hl`	hl
microliter	`\uL`	μL
milliliter	`\mL`	mL
liter	`\L`	L
hectoliter	`\hL`	hL
millihertz	`\mHz`	mHz
hertz	`\Hz`	Hz
kilohertz	`\kHz`	kHz
megahertz	`\MHz`	MHz
gigahertz	`\GHz`	GHz
terahertz	`\THz`	THz
millinewton	`\mN`	mN
newton	`\N`	N
kilonewton	`\kN`	kN
meganewton	`\MN`	MN
pascal	`\Pa`	Pa
kilopascal	`\kPa`	kPa
megapacal	`\MPa`	MPa
gigapascal	`\GPa`	GPa
milliohm	`\mohm`	mΩ
kilohm	`\kohm`	kΩ
megohm	`\Mohm`	MΩ
picovolt	`\pV`	pV
nanovolt	`\nV`	nV
microvolt	`\uV`	μV
millivolt	`\mV`	mV
volt	`\V`	V
kilovolt	`\kV`	kV
watt	`\W`	W
microwatt	`\uW`	μW
milliwatt	`\mW`	mW
kilowatt	`\kW`	kW
megawatt	`\MW`	MW
gigawatt	`\GW`	GW
joule	`\J`	J
kilojoule	`\kJ`	kJ

Continued on next page

Table 22 – Binary prefixes.

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

Continued from previous page

Unit	Abbreviation	Symbol
electronvolt	\eV	eV
millielectronvolt	\meV	meV
kiloelectronvolt	\keV	keV
megaelectronvolt	\MeV	MeV
gigaelectronvolt	\GeV	GeV
teraelectronvolt	\TeV	TeV
kilowatt hour	\kWh	kWh
farad	\F	F
femtofarad	\fF	fF
picofarad	\pF	pF
kelvin	\K	K
decibel	\dB	dB

binary-units
\bit
\byte

Binary data is expressed in units of bits and bytes. These are normally given prefixes which use powers of two, rather than the powers of ten used by the SI prefixes. As these binary prefixes are closely related to the SI prefixes, they are defined by siunitx but do have to be loaded using `binary-units = true` (or simply `binary-units`) (Table 22). The units `\bit` and `\byte` are then also available.

```
\SI{100}{\mebi\byte} \\
\SI[prefixes-as-symbols=false]{30}{\kibi\bit}
```

100 MiB
30×2^{10} bit

version-1-compatibility

A configuration file is also included which will use settings and define macros as defined by version 1 of siunitx: this can be accessed with the option `version-1-compatibility`. This is intended to allow easy transition to version 2: users should update their source to use the new interfaces and functions.

Users upgrading from version 1 of siunitx will notice that the various 'specialist' units available in version 1 are no longer provided as loadable options.[4] These are not included in version 2 as the criteria for inclusion of such units are far from clear, and

[4] They are included in the loaded configuration file `version-1`, but this is intended purely to ease transition to version 2.

it is difficult to justify providing clearly non-SI units in the package. For reference, appropriate definitions for the units which where provided in version 1 are as follows.

```
% Astronomy
\DeclareSIUnit\parsec{pc}
\DeclareSIUnit\lightyear{ly}

% Chemical engineering
\DeclareSIUnit\gmol{g\text{-}mol}
\DeclareSIUnit\kgmol{kg\text{-}mol}
\DeclareSIUnit\lbmol{lb\text{-}mol}

% Chemistry
\DeclareSIUnit\molar{\mole\per\cubic\deci\metre}
\DeclareSIUnit\Molar{\textsc{m}}
\DeclareSIUnit\torr{torr}

% Geophysics
\DeclareSIUnit\gon{gon}

% High energy physics
\DeclareSIUnit\micron{\micro\metre}
\DeclareSIUnit\mrad{\milli\rad}
\DeclareSIUnit\gauss{G}
\DeclareSIUnit\eVperc{\eV\per\clight}
\DeclareSIUnit\nanobarn{\nano\barn}
\DeclareSIUnit\picobarn{\pico\barn}
\DeclareSIUnit\femtobarn{\femto\barn}
\DeclareSIUnit\attobarn{\atto\barn}
\DeclareSIUnit\zeptobarn{\zepto\barn}
\DeclareSIUnit\yoctobarn{\yocto\barn}
\DeclareSIUnit\nb{\nano\barn}
\DeclareSIUnit\pb{\pico\barn}
\DeclareSIUnit\fb{\femto\barn}
\DeclareSIUnit\ab{\atto\barn}
\DeclareSIUnit\zb{\zepto\barn}
\DeclareSIUnit\yb{\yocto\barn}
```

Users can use a local configuration file to make additional units available on a local basis, as described in Section 5.17.

5.12 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 23).

`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-product` The separator between each unit is stored using the `inter-unit-product` 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 `\ensuremath{}\cdot{}` in the latter case.

Table 23 – Unit output options.

Option name	Type	Default
bracket-unit-denominator	Switch	true
forbid-literal-units	Switch	false
literal-superscript-as-power	Switch	true
inter-unit-product	Literal	\,
parse-units	Switch	true
per-mode	Choice	reciprocal
per-symbol	Literal	/
power-font	Choice	number
prefixes-as-symbols	Switch	true
qualifier-mode	Choice	subscript
sticky-per	Switch	false

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

$\mathrm{F^2\,lm\,cd}$
$\mathrm{F^2\cdot lm\cdot cd}$

per-mode
per-symbol
bracket-unit-denominator

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}
```

$\mathrm{J\,mol^{-1}\,K^{-1}}$
$\mathrm{m\,s^{-2}}$
$\frac{\mathrm{J}}{\mathrm{mol\,K}}$
$\frac{\mathrm{m}}{\mathrm{s^2}}$

The closely-related reciprocal-positive-first setting acts in the same way but places all of the positive powers before any negative ones.

```
\si{\ampere\per\mole\second} \\
\si[per-mode = reciprocal-positive-first]
  {\ampere\per\mole\second}
```

$\mathrm{A\,mol^{-1}\,s}$
$\mathrm{A\,s\,mol^{-1}}$

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 bracket-unit-denominator is set to false. Notice that bracket-unit-denominator only applies when per-mode is set to symbol or symbol-or-fraction.

```
\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[bracket-unit-denominator = false]{\joule\per\mole\per\kelvin}
```

J/(mol K)
$\mathrm{m/s^2}$
J div (mol K)
J/mol K

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

Finally, it is possible for the behaviour of the `\per` function to depend on the prevailing math style. Setting `per-mode` to `symbol-or-fraction` will use the `symbol` setting for in line math, and the `fraction` setting when used in `\displaystyle` math.

```
\sisetup{per-mode = symbol-or-fraction}%
\( \si{\joule\per\mole\per\kelvin} \)
\[ \si{\joule\per\mole\per\kelvin} \]
   \si{\joule\per\mole\per\kelvin} \\
\(
  \displaystyle
  \si{\joule\per\mole\per\kelvin}
\)
\[
  \textstyle
  \si{\joule\per\mole\per\kelvin}
\]
```

J/(mol K)

$$\frac{\mathrm{J}}{\mathrm{mol\,K}}$$

J/(mol K)
$\displaystyle\frac{\mathrm{J}}{\mathrm{mol\,K}}$

$$\mathrm{J/(mol\,K)}$$

`sticky-per` By default, `\per` applies only to the next unit given.[5] 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}
```

$\mathrm{Pa\,Gy^{-1}\,H}$
$\mathrm{Pa\,Gy^{-1}\,H^{-1}}$

`power-font` The font used for the powers in units can be typeset using the current number or unit font. This may be of use when the font used for numbers and units are very different, for example when the euler package is loaded. Note that this setting applies to all printing options for numbers, including the color used for displaying the number.

[5] This is the standard method of reading units in English: for example, $\mathrm{J\,mol^{-1}\,K^{-1}}$ is pronounced 'joules per mole per kelvin'.

```
\si{\metre\per\second\squared} \\
\si[power-font = unit]{\metre\per\second\squared}
```

$\mathrm{m\,s^{-2}}$
$\mathrm{m\,s^{-2}}$

literal-superscript-as-power When printing units in 'literal' mode, it is possible that simply printing superscripts 'as is' may lead to poor appearance for the numbers. This is most likely if the text font of the document uses old style (lower case) numerals, but the math font uses lining (upper case) numerals. It is therefore possible to treat superscripts within literal units as powers, and thus for the power-font option to apply within these literal units. This behaviour is controlled using the literal-superscript-as-power switch.

```
\si{m.s^{2}} \\
\si[literal-superscript-as-power = false]{m.s^{2}}
```

$\mathrm{m\,s^{2}}$
$\mathrm{m\,s^{2}}$

qualifier-mode
qualifier-phrase

Unit qualifiers can be printed in three different formats, set by the qualifier-mode option. The standard setting is subscript, while the options brackets, phrase, space and text 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 = subscript]
  {\kilogram\polymer\squared\per\mole\catalyst\per\hour} \\
\si[qualifier-mode = space]
  {\kilogram\polymer\squared\per\mole\catalyst\per\hour}  \\
\si[qualifier-mode = text]
  {\deci\bel\isotropic}
```

$\mathrm{kg_{pol}^{2}\,mol_{cat}^{-1}\,h^{-1}}$
$\mathrm{kg(pol)^{2}\,mol(cat)^{-1}\,h^{-1}}$
$\mathrm{kg_{pol}^{2}\,mol_{cat}^{-1}\,h^{-1}}$
$\mathrm{(kg\ pol)^{2}\,(mol\ cat)^{-1}\,h^{-1}}$
dBi

The phrase setting for qualifier-mode uses the text stored using qualifier-phrase to separate the qualifier.

```
\si[qualifier-mode = phrase]
  {\kilogram\polymer\squared\per\mole\catalyst\per\hour} \\
\si[qualifier-mode = phrase, qualifier-phrase = { by }]
  {\kilogram\polymer\squared\per\mole\catalyst\per\hour} \\
```

$\mathrm{(kgofpol)^{2}\,(molofcat)^{-1}\,h^{-1}}$
$\mathrm{(kgbypol)^{2}\,(molbycat)^{-1}\,h^{-1}}$

prefixes-as-symbols

The unit prefixes (\kilo, *etc.*) are normally given as letters. However, the package can convert these into numerical powers. This is controlled by the prefixes-as-symbols switch option. This correctly deals with the kilogram, which is a base unit even though it involves a prefix.

Table 24 – Options for numbers with units.

Option name	Type	Default
allow-number-unit-breaks	Switch	false
exponent-to-prefix	Switch	false
list-units	Choice	repeat
multi-part-units	Choice	brackets
number-unit-product	Literal	\,
product-units	Choice	repeat
range-units	Choice	repeat

```
\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 kg^2 ds
10^{-4} l mol^{-1} A
10 × 10^{-1} kg^2 s

parse-units

Normally, siunitx is used with the unit parse enabled, and only prints units directly if there is literal input. However, if the input is known to be essentially consistent and high performance is desired, then the parser can be turned off using the parse-units switch.

300 MHz
300 MHz

```
\SI{300}{\MHz} \\
\SI[parse-units = false]{300}{\MHz}
```

5.13 Numbers with units

Some options apply to the combination of units and numbers, rather than to units or numbers alone (Table 24).

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
10 m
Some filler text 10
m

```
\begin{minipage}{2.55 cm}
  % Gives an underfull hbox
  Some filler text \SI{10}{\metre} \\
  \sisetup{allow-number-unit-breaks}
  Some filler text \SI{10}{\metre}
\end{minipage}
```

number-unit-product

The product symbol between the number and unit is set using the number-unit-product option.

```
\SI{2.67}{\farad} \\
\SI[number-unit-product = \ ]{2.67}{\farad} \\
\SI[number-unit-product = ]{2.67}{\farad}
```

2.67 F
2.67 F
2.67F

multi-part-units When a number has multiple parts (such as a separate uncertainty) then the unit must apply to all parts of the number. How this is shown is controlled using the `multi-part-units` options. The standard setting is `brackets`, which will place the entire numerical part in brackets and use a single unit symbol. Alternative options are `repeat` (print the unit for each part of the number) and `single` (print only one unit symbol: mathematically incorrect).

```
\sisetup{separate-uncertainty}%
\SI{12.3(4)}{\kilo\gram} \\
\SI[multi-part-units = brackets]{12.3(4)}{\kilo\gram} \\
\SI[multi-part-units = repeat]{12.3(4)}{\kilo\gram}   \\
\SI[multi-part-units = single]{12.3(4)}{\kilo\gram}
```

$(12.3 \pm 0.4)\,\mathrm{kg}$
$(12.3 \pm 0.4)\,\mathrm{kg}$
$12.3\,\mathrm{kg} \pm 0.4\,\mathrm{kg}$
$12.3 \pm 0.4\,\mathrm{kg}$

It is important to notice that numbers with units are not affected by the setting of `bracket-numbers`, which applies to 'pure' numbers only. For example:

```
\sisetup{separate-uncertainty,bracket-numbers = false}%
\num{1.234(5)e-4} \\
\SI{1.234(5)e-4}{\metre}
```

$1.234 \pm 0.005 \times 10^{-4}$
$(1.234 \pm 0.005) \times 10^{-4}\,\mathrm{m}$

The reason is that the requirements to bracket values with units are fundamentally different from those for numbers alone. Some combinations which are mathematically valid in the absence of a unit become invalid when a unit is present.

product-units When a product of quantities is given, the resulting units can be displayed in a number of ways, set using the `product-units` option. The standard setting is `repeat`, which prints one unit symbol for each numbers. Alternatives are `brackets`, `brackets-power`, `power`, `repeat` and `single`. This option does not affect the application of brackets for each number within the product list: it only sets those around the entire list.

```
\SI{2 x 3 x 4}{\metre} \\
\SI[product-units = brackets]{2 x 3 x 4}{\metre}       \\
\SI[product-units = brackets-power]{2 x 3 x 4}{\metre} \\
\SI[product-units = power]{2 x 3 x 4}{\metre}          \\
\SI[product-units = repeat]{2 x 3 x 4}{\metre}         \\
\SI[product-units = single]{2 x 3 x 4}{\metre}
```

$2\,\mathrm{m} \times 3\,\mathrm{m} \times 4\,\mathrm{m}$
$(2 \times 3 \times 4)\,\mathrm{m}$
$(2 \times 3 \times 4)\,\mathrm{m}^3$
$2 \times 3 \times 4\,\mathrm{m}^3$
$2\,\mathrm{m} \times 3\,\mathrm{m} \times 4\,\mathrm{m}$
$2 \times 3 \times 4\,\mathrm{m}$

list-units
range-units The `list-units` and `range-units` options determine how the `\SIlist` and `\SIrange` functions display units, respectively. The standard setting for both is `repeat`, where each number will be printed with a unit. Alternatives are `brackets` and `single`. Any brackets needed on individual numbers within a product are controlled by the `brackets-numbers` option (*i.e.* they are treated as pure numbers). These options do not

affect the application of brackets for each number within the list or range: they only set those around the entire group.

```
\SIlist{2;4;6;8}{\tesla} \\
\SIlist[list-units = brackets]{2;4;6;8}{\tesla} \\
\SIlist[list-units = repeat]{2;4;6;8}{\tesla} \\
\SIlist[list-units = single]{2;4;6;8}{\tesla} \\
\SIrange{2}{4}{\degreeCelsius} \\
\SIrange[range-units = brackets]{2}{4}{\degreeCelsius} \\
\SIrange[range-units = repeat]{2}{4}{\degreeCelsius} \\
\SIrange[range-units = single]{2}{4}{\degreeCelsius}
```

2 T, 4 T, 6 T and 8 T
(2, 4, 6 and 8) T
2 T, 4 T, 6 T and 8 T
2, 4, 6 and 8 T
2 °C to 4 °C
(2 to 4) °C
2 °C to 4 °C
2 to 4 °C

`exponent-to-prefix` When the `exponent-to-prefix` option is set `true`, the package will attempt to convert any exponents in quantities into unit prefixes, and will attach these to the first unit given. This process is only possible if the exponent is one for which a prefix is available, and retains the number of significant figures in the input.

```
\SI{1700}{\g}  \\
\SI{1.7e3}{\g} \\
\sisetup{exponent-to-prefix}%
\SI{1700}{\g}  \\
\SI{1.7e3}{\g} \\
\sisetup{fixed-exponent = 3, scientific-notation = fixed}%
\SI{1700}{\g}  \\
\SI{1.7e3}{\g}
```

1700 g
1.7×10^3 g
1700 g
1.7 kg
1.700 kg
1.7 kg

5.14 Tabular material

Processing of material in tables obeys the same settings as described for the functions already described. However, there are some settings which apply only to the layout of tabular material (Table 25).

`table-parse-only` The main use of the `S` column is to control the alignment of the resulting output. However, it is possible to turn off alignment entirely and only use the number parser of siunitx. This is achieved using the `table-parse-only` switch, as illustrated in Table 26.

```
\begin{table}
  \centering
  \caption{Parsing without aligning in an \texttt{S} column.}
  \label{tab:S:parse}
  \begin{tabular}
```

Table 25 – Options for tabular material.

Option name	Type	Default
`table-align-comparator`	Switch	`true`
`table-align-exponent`	Switch	`true`
`table-align-text-pre`	Switch	`true`
`table-align-text-post`	Switch	`true`
`table-align-uncertainty`	Switch	`true`
`table-alignment`	Choice	⟨*none*⟩
`table-auto-round`	Switch	`false`
`table-column-width`	Length	`0 pt`
`table-comparator`	Switch	`false`
`table-figures-decimal`	Integer	`2`
`table-figures-exponent`	Integer	`0`
`table-figures-integer`	Integer	`3`
`table-figures-uncertainty`	Integer	`0`
`table-format`	Special	⟨*none*⟩
`table-number-alignment`	Choice	`center-decimal-marker`
`table-parse-only`	Switch	`false`
`table-omit-exponent`	Switch	`true`
`table-space-text-pre`	Literal	⟨*empty*⟩
`table-space-text-post`	Literal	⟨*empty*⟩
`table-sign-exponent`	Switch	`false`
`table-sign-mantissa`	Switch	`false`
`table-text-alignment`	Choice	`center`
`table-unit-alignment`	Choice	`center`

Table 26 – Parsing without aligning in an S column.

Decimal-centred	Simple centring
12.345	12.345
6.78	6.78
−88.8(9)	−88.8(9)
4.5×10^3	4.5×10^3

```
    {
      S
      S[table-parse-only]
    }
    \toprule
      {Decimal-centred} &
      {Simple centring} \\
    \midrule
      12.345   & 12.345   \\
      6,78     & 6,78     \\
      -88.8(9) & -88.8(9) \\
      4.5e3    & 4.5e3    \\
    \bottomrule
  \end{tabular}
\end{table}
```

`table-number-alignment`

The alignment of numbers with the boundaries of the S column is controlled using the table-number-alignment option, which takes the values center-decimal-marker, center, left and right. The center-decimal-marker places the decimal marker for the number at the centre of the column. This does not need any information in advance, and so is the standard setting. It works best for approximately symmetrical input (equal numbers of digits before and after the decimal). On the other hand, the center, left and right options require space to be reserved for the numbers, and then use this fixed space to align with the edges of the column. The different alignment choices are illustrated in Table 27, which uses somewhat exaggerated column headings to show the relative position of the cell contents.

```
\begin{table}
  \caption{Aligning the \texttt{S} column.}
  \label{tab:S:align}
  \centering
  \sisetup{
    table-figures-integer = 2,
    table-figures-decimal = 4
  }
  \begin{tabular}{
    S
    S[table-number-alignment = center]
    S[table-number-alignment = left]
    S[table-number-alignment = right]
  }
  \toprule
  {Some Values} & {Some Values} & {Some Values} & {Some Values} \\
```

Table 27 – Aligning the S column.

Some Values	Some Values	Some Values	Some Values
2.3456	2.3456	2.3456	2.3456
34.2345	34.2345	34.2345	34.2345
56.7835	56.7835	56.7835	56.7835
90.473	90.473	90.473	90.473

```
  \midrule
     2.3456  &  2.3456  &  2.3456 &  2.3456 \\
    34.2345  & 34.2345  & 34.2345 & 34.2345 \\
    56.7835  & 56.7835  & 56.7835 & 56.7835 \\
    90.473   & 90.473   & 90.473  & 90.473  \\
  \bottomrule
  \end{tabular}
\end{table}
```

Many of the other table options do not apply when `table-number-alignment = center-decimal-marker` is set, as this mode always centres the marker at the expense of any other choices.

table-figures-decimal
table-figures-exponent
table-figures-integer
table-figures-uncertainty

The space reserved by siunitx for a number is controlled by two families of options. The first family cover the number of digits allowed for in different parts of the number, for example `table-figures-integer` controls the space for integer digits in the mantissa. If the number of figures is set to 0, then no space is reserved and some output will either be out of position or not printed at all (although a warning will result). Reserving space for a given part of number will automatically include space for any associated items (for example the '×' symbol for exponents). The second family of options are switches which govern whether space is reserved for a sign: `table-sign-exponent` and `table-sign-mantissa`. The effect of altering some of these settings is shown in Table 28.

table-sign-exponent
table-sign-mantissa

```
\begin{table}
  \caption{Reserving space in \texttt{S} columns.}
  \label{tab:S:space}
  \sisetup{
    table-number-alignment = center,
    table-figures-integer  = 2
  }
  \centering
  \begin{tabular}{
    S
    S[table-number-alignment = right]
    S[table-figures-uncertainty = 1]
    S[
      separate-uncertainty,
      table-figures-uncertainty = 1
    ]
    S[table-sign-mantissa]
    S[table-figures-exponent = 1]
    }
  \toprule
      {Values}
```

Table 28 – Reserving space in `S` columns.

Values	Values	Values	Values	Values	Values
2.3	2.3	2.3(5)	2.3 ± 0.5	2.3	2.3×10^{8}
34.23	34.23	34.23(4)	34.23 ± 0.04	34.23	34.23
56.78	56.78	56.78(3)	56.78 ± 0.03	-56.78	56.78×10^{3}
3.76	3.76	3.76(2)	3.76 ± 0.02	± 3.76	10^{6}

```
    & {Values}
    & {Values}
    & {Values}
    & {Values}
    & {Values} \\
  \midrule
     2.3  &  2.3  &  2.3(5)  &  2.3(5)  &   2.3  &  2.3e8  \\
    34.23 & 34.23 & 34.23(4) & 34.23(4) &  34.23 & 34.23   \\
    56.78 & 56.78 & 56.78(3) & 56.78(3) & -56.78 & 56.78e3 \\
     3,76 &  3,76 &  3,76(2) &  3.76(2) & +-3.76 &      e6 \\
  \bottomrule
  \end{tabular}
\end{table}
```

`table-comparator` Space can also be reserved in a table for a comparator (greater than, less than, and so forth). This is activated using the `table-comparator` switch (Table 29).

```
\begin{table}
  \caption{Reserving space for comparators in \texttt{S} columns.}
  \label{tab:S:comparators}
  \sisetup{
    table-number-alignment = center,
    table-figures-integer  = 2,
    table-figures-decimal  = 2,
    table-figures-exponent = 2,
  }
  \centering
  \begin{tabular}{
    S
    S[table-comparator = true]
    }
  \toprule
      {Values}
    & {Values} \\
  \midrule
     2.3  & <   2.3e8  \\
    34.23 & =   34.23  \\
    56.78 & >= 56.78e3 \\
     3,76 & \gg     e6 \\
  \bottomrule
  \end{tabular}
\end{table}
```

The table-printing code will omit any part of a number which has no space reserved,

Table 29 – Reserving space for comparators in S columns.

Values	Values
2.3	$<$ 2.3 $\times 10^{8}$
34.23	$=$34.23
56.78	$\geq$56.78 $\times 10^{3}$
3.76	$\gg$ 10^{6}

placing a warning in the LaTeX log. This means that uncertainties and exponents will not be printed if no space is reserved for them.

table-format

As a short cut for the preceding options, siunitx also provides the `table-format` option. This can be used to give the same information about the space to reserve for a number in a 'compressed' manner. The input to `table-format` should consist of a number showing how many figures to reserve in each part of the input. Thus

```
\sisetup{
  table-format = 3.2
}
```

is equivalent to

```
\sisetup{
  table-figures-integer = 3,
  table-figures-decimal = 2
}
```

The `table-format` option will also correctly interpret the presence of a sign, so that

```
\sisetup{
  table-format = +3.2e+4
}
```

will have the same effect as

```
\sisetup{
  table-figures-integer  = 3,
  table-figures-decimal  = 2,
  table-figures-exponent = 4,
  table-sign-mantissa,
  table-sign-exponent
}
```

It is important to note that any parts of a number *not* specified in the table format argument are set to be absent (the number of figures is set to zero). Setting the `table-format` option also resets `table-number-alignment` to `center` (Table 30).

```
\begin{table}
  \caption{Using the \opt{table-format} option.}
  \label{tab:S:format}
  \centering
  \begin{tabular}{
    S
    S[table-format =  2.2]
    S[table-format =  2.2(1)]
    S[table-format = +2.2]
```

Table 30 – Using the `table-format` option.

Values	Values	Values	Values	Values
2.3	2.3	2.3(5)	2.3	2.3×10^8
34.23	34.23	34.23(4)	34.23	34.23
56.78	56.78	56.78(3)	-56.78	56.78×10^3
3.76	3.76	3.76(2)	± 3.76	10^6

```
    S[table-format =  2.2e1]
    }
  \toprule
      {Values}
    & {Values}
    & {Values}
    & {Values}
    & {Values} \\
  \midrule
     2.3  &  2.3  &  2.3(5)  &   2.3  &  2.3e8  \\
    34.23 & 34.23 & 34.23(4) &  34.23 & 34.23   \\
    56.78 & 56.78 & 56.78(3) & -56.78 & 56.78e3 \\
     3,76 &  3,76 &  3.76(2) & +-3.76 &      e6 \\
  \bottomrule
  \end{tabular}
\end{table}
```

`table-space-text-pre`
`table-space-text-post`

Space for material before and after the `S` column can be reserved by giving model text for the options `table-space-text-pre` and `...-post`. This is then used to provide the necessary gap while maintaining alignment (Table 31).

```
\begin{table}
  \caption{Text before and after numbers.}
  \label{tab:S:ends}
  \centering
  \sisetup{
    table-number-alignment = center,
    table-figures-integer  = 2,
    table-figures-decimal  = 4,
    table-space-text-pre   = now~,
    table-space-text-post  =
      \textsuperscript{\emph{a}}
  }
  \begin{tabular}{S}
  \toprule
  {Values} \\
  \midrule
          2.3456  \\
         34.2345  \textsuperscript{\emph{a}}\\
         56.7835  \\
    now~ 90.473   \\
  \bottomrule
  \end{tabular}
```

Table 31 – Text before and after numbers.

Values
2.3456
34.2345[a]
56.7835
now 90.473

Table 32 – The `table-align-exponent` option

Header	Header
$1.2 \quad \times 10^{3}$	1.2×10^{3}
1.234×10^{56}	1.234×10^{56}

```
\end{table}
```

`table-align-comparator`
`table-align-exponent`
`table-align-uncertainty`

When printing exponents in tables, there is a choice of aligning the exponent parts or having these close up to the mantissa. This is controlled by the `table-align-exponent` option (Table 32). Similarly, uncertainty parts which are printed separately from the mantissa can be aligned or closed up. This is set by the `table-align-uncertainty` option (Table 33). Finally, the same approach is available for the comparator with the `table-align-comparator` option (Table 34).

```
\begin{table}
  \centering
  \caption{The \opt{table-align-exponent} option}
  \label{tab:align:exp}
  \sisetup{table-format = 1.3e2, table-number-alignment = center}
  \begin{tabular}{SS[table-align-exponent = false]}
    \toprule
      {Header} & {Header} \\
    \midrule
      1.2e3     & 1.2e3     \\
      1.234e56 & 1.234e56 \\
    \bottomrule
  \end{tabular}
\end{table}
```

```
\begin{table}
  \centering
  \caption{The \opt{table-align-uncertainty} option}
  \label{tab:align:uncert}
  \sisetup{
    separate-uncertainty,
    table-format = 1.3(1),
  }
  \begin{tabular}{SS[table-align-uncertainty = false]}
    \toprule
      {Header} & {Header} \\
```

Table 33 – The `table-align-uncertainty` option

Header	Header
1.2 $\pm$ 0.1	1.2 $\pm$ 0.3
1.234 $\pm$ 0.005	1.234 $\pm$ 0.005

Table 34 – The `table-align-comparator` option

Header	Header
$>$ 1.2	$>$1.2
$<$12.34	$<$12.34

```
    \midrule
      1.2(1)   & 1.2(3)    \\
      1.234(5) & 1.234(5) \\
    \bottomrule
  \end{tabular}
\end{table}
```

```
\begin{table}
  \centering
  \caption{The \opt{table-align-comparator} option}
  \label{tab:align:comp}
  \sisetup{table-format = >2.2}
  \begin{tabular}{SS[table-align-comparator = false]}
    \toprule
      {Header} & {Header} \\
    \midrule
      >  1.2  & >  1.2  \\
      < 12.34 & < 12.34 \\
    \bottomrule
  \end{tabular}
\end{table}
```

`table-omit-exponent` In cases where data cover a range of values, printing using a fixed exponent in a table may make presentation clearer. In these cases, omitting the exponent value from the table is useful. The package offers the `table-omit-exponent` option to do this (Table 35); this automatically sets `scientific-notation = fixed` for the table column.

```
\begin{table}
  \centering
  \caption{The \opt{table-omit-exponent} option}
  \label{tab:exp:omit}
  \begin{tabular}{
    S[table-format = 1.1e1]
    S[
      fixed-exponent      = 3,
      table-format        = 2.1,
      table-omit-exponent
```

Table 35 – The table-omit-exponent option

Header	Header / 10^3
1.2×10^3	1.2
3×10^2	0.3
1.0×10^4	10

```
    ]
  }
    \toprule
      {Header} & {Header / \num{e3}} \\
    \midrule
      1.2e3 & 1.2e3 \\
       3e2  &   3e2 \\
     1.0e4  & 1.0e4 \\
    \bottomrule
  \end{tabular}
\end{table}
```

table-align-text-pre
table-align-text-post

Note markers are often given in tables after the numerical content. It may be desirable for these to close up to the numbers. Whether this takes place is controlled by the table-align-text-pre and ...-post option (Table 36).

```
\begin{table}
  \caption{Closing notes up to text.}
  \label{tab:S:notes}
  \newrobustcmd\NoteMark[1]{%
    \textsuperscript{\emph{#1}}%
  }
  \centering
  \sisetup{
    table-number-alignment = center,
    table-figures-integer  = 2,
    table-figures-decimal  = 4,
    table-space-text-pre   = \NoteMark{a}
  }
  \begin{tabular}{
    S
    S[table-align-text-pre = false]
  }
  \toprule
    {Values}             & {Values} \\
  \midrule
                2.3456 &               2.3456 \\
    \NoteMark{a}  4.234  & \NoteMark{a}  4.234 \\
    \NoteMark{b}   .78   & \NoteMark{b}   .78  \\
    \NoteMark{d} 88      & \NoteMark{d} 88     \\
  \bottomrule
  \end{tabular}
  \hfil
  \sisetup{table-space-text-post = \NoteMark{a}}
```

Table 36 – Closing notes up to text.

Values	Values
2.3456	2.3456
[a] 4.234	[a]4.234
[b] 0.78	[b]0.78
[d]88	[d]88

Values	Values
2.3456	2.3456
34.234 [a]	34.234[a]
56.78 [b]	56.78[b]
90.4 [c]	90.4[c]
88 [d]	88[d]

```
  \begin{tabular}{
    S
    S[table-align-text-post = false]
  }
  \toprule
    {Values}             & {Values} \\
  \midrule
     2.3456              & 2.3456 \\
    34.234  \NoteMark{a} & 34.234 \NoteMark{a} \\
    56.78   \NoteMark{b} & 56.78  \NoteMark{b} \\
    90.4    \NoteMark{c} & 90.4   \NoteMark{c} \\
    88      \NoteMark{d} & 88     \NoteMark{d} \\
  \bottomrule
  \end{tabular}
\end{table}
```

`table-auto-round` The contents of table cells can automatically be rounded or zero-filled to the number of decimal digits given for the `table-figures-decimal` option. This mode is activated using the `table-auto-round` switch, as illustrated in Table 37.

```
\begin{table}
  \centering
  \caption{The \opt{table-auto-round} option.}
  \label{tab:S:auto}
  \sisetup{
    table-number-alignment = center,
    table-figures-integer  = 1,
    table-figures-decimal  = 3
  }
  % Notice the overfull hbox which results with
  % the first column
  \begin{tabular}{
    S
    S[table-auto-round]
  }
    \toprule
      {Header} & {Header} \\
    \midrule
      1.2    & 1.2    \\
      1.2345 & 1.2345 \\
    \bottomrule
  \end{tabular}
```

Table 37 – The `table-auto-round` option.

Header	Header
1.2	1.200
1.2345	1.235

```
\end{table}
```

`parse-numbers` When the `parse-numbers` option is set to `false`, then the alignment code for tables takes a different approach. The output is always set in math mode, and alignment takes place at the first decimal marker. This is achieved by making it active in math mode. When reserving space for content only the integer and decimal values for the mantissa are considered (Table 38).

```
\begin{table}
  \caption{Aligning without parsing.}
  \label{tab:S:nonparsed}
  \sisetup{
    parse-numbers = false,
    table-figures-integer = 2,
    table-figures-decimal = 3
  }
  \centering
  \begin{tabular}{
    S
    S[table-number-alignment = center]
    S[table-number-alignment = right]
    S[table-number-alignment = left]
    }
  \toprule
      {Some values}
    & {Some values}
    & {Some values}
    & {Some values} \\
  \midrule
     2.35  &    2.35  &    2.35  &   2.35   \\
    34.234 &   34.234 &   34.234 &  34.234  \\
    56.783 &   56.783 &   56.783 &  56.783  \\
     3,762 &    3,762 &    3,762 &   3.762  \\
  \sqrt{2} & \sqrt{2} & \sqrt{2} & \sqrt{2} \\
  \bottomrule
  \end{tabular}
\end{table}
```

`table-text-alignment` Cell contents which are not part of a number can be protected using braces, as illustrated. Cells which contain no numerical data at all are aligned using the setting specified by the `table-text-alignment` option, which recognises the values `center`, `left` and `right` (Table 39).

```
\begin{table}
  \caption{Aligning text in \texttt{S} columns.}
```

Table 38 – Aligning without parsing.

Some values	Some values	Some values	Some values
2.35	2.35	2.35	2.35
34.234	34.234	34.234	34.234
56.783	56.783	56.783	56.783
3.762	3.762	3.762	3.762
$\sqrt{2}$	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{2}$

Table 39 – Aligning text in S columns.

Values	Values	Values
992.435	992.435	992.435
7734.2344	7734.2344	7734.2344
56.7834	56.7834	56.7834
3.7462	3.7462	3.7462

```
  \label{tab:S:text}
  \sisetup{
    table-number-alignment = center,
    table-figures-integer  = 4,
    table-figures-decimal  = 4
  }
  \centering
  \begin{tabular}{
    S
    S[table-text-alignment = left]
    S[table-text-alignment = right]
    }
  \toprule
      {Values}
    & {Values}
    & {Values} \\
  \midrule
    992.435  &  992.435  &  992.435  \\
   7734.2344 & 7734.2344 & 7734.2344 \\
     56.7834 &   56.7834 &   56.7834 \\
      3,7462 &    3,7462 &    3,7462 \\
  \bottomrule
  \end{tabular}
\end{table}
```

`table-unit-alignment` The contents of s columns can be centred or aligned to the left or right using the `table-unit-alignment` option. As for the other alignment options, this recognises the choices `center`, `left` and `right`.

```
\begin{table}
  \centering
  \caption{Alignment options in \texttt{s} columns.}
  \label{tab:s:align}
```

Table 40 – Alignment options in s columns.

Right – aligned	Centredtext	Left – aligned
$\mathrm{m\,s^{-1}}$	$\mathrm{m\,s^{-1}}$	$\mathrm{m\,s^{-1}}$
kg	kg	kg

```
  \begin{tabular}
    {
      s[table-unit-alignment = right]
      s
      s[table-unit-alignment = left]
    }
    \toprule
      {Right-aligned}  &
      {Centred text}  &
      {Left-aligned} \\
    \midrule
      \metre\per\second & \metre\per\second & \metre\per\second \\
      \kilogram         & \kilogram         & \kilogram         \\
    \bottomrule
  \end{tabular}
\end{table}
```

`table-alignment` The three table alignment options (`table-number-alignment`, `table-text-alignment` and `table-unit-alignment`) can be set to the same value using the `table-alignment` option. This will set all three alignment options to the same value (one of `center`, `right` or `left`).

`table-column-width` Usually, the width of the `S` and `s` columns is allowed to vary depending on the content. However, there are cases where a strictly fixed width is desirable. For these cases, the `table-column-width` option is available. The standard setting, `0 pt`, indicates that no fixing takes place. If a value is set for this option then the tabular material is typeset to the specified width (Table 41).

```
\begin{table}
  \centering
  \caption{Fixed-width columns.}
  \label{tab:width:fixed}
  \begin{tabular}
    {
      s
      s[table-column-width = 2 cm]
      S
      S[table-column-width = 2 cm]
    }
    \toprule
      {Flexible} &
      {Fixed}    &
      {Flexible} &
      {Fixed}    \\
    \midrule
      \metre\per\second & \metre\per\second & 1.23 & 1.23 \\
```

Table 41 – Fixed-width columns.

Flexible	Fixed	Flexible	Fixed
$m\,s^{-1}$	$m\,s^{-1}$	1.23	1.23
kg cd	kg cd	45.6	45.6

Table 42 – Right-aligning under a heading.

Long header
12.33
2
1234

```
      \kilogram\candela & \kilogram\candela & 45.6 & 45.6 \\
    \bottomrule
  \end{tabular}
\end{table}
```

The table-column-width option can also be used to achieve special effects. One example is centring a column of numbers under a wide heading, with the numbers themselves right-aligned (Table 42).

```
\begin{table}
  \centering
  \caption{Right-aligning under a heading.}
  \label{tab:width:special}
  \settowidth\mylength{Long header}
  \sisetup{
    table-format           = 4         ,
    table-number-alignment = center    ,
    table-column-width     = \mylength ,
    input-decimal-markers  =           ,
    input-symbols          = .         ,
  }
  \begin{tabular}{S}
  \toprule
  {Long header} \\
  \midrule
      12.33 \\
       2    \\
    1234    \\
   \bottomrule
  \end{tabular}
\end{table}
```

Table 43 – Symbol options.

Option name	Type	Default
math-angstrom	Literal	\text{\AA}
math-arcminute	Literal	{}^{\prime}
math-arcsecond	Literal	{}^{\prime\prime}
math-celsius	Literal	⟨*see text*⟩
math-degree	Literal	{}^{\circ}
math-micro	Literal	⟨*see text*⟩
math-ohm	Literal	\Omega
redefine-symbols	Switch	true
text-angstrom	Literal	\AA
text-arcminute	Literal	\ensuremath{{}^{\prime}}
text-arcsecond	Literal	\ensuremath{{}^{\prime\prime}}
text-celsius	Literal	\ensuremath{{}^{\circ}} \kern -\scriptspace C
text-degree	Literal	\ensuremath{{}^{\circ}}
text-micro	Literal	⟨*see text*⟩
text-ohm	Literal	\ensuremath{\Omega}

5.15 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 math and text output, and are summarised in Table 43. 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 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 math 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!).[6]

When using XeTeX or LuaTeX, if fontspec is loaded these options are redefined to use UTF-8 characters directly.

[6] The function \SIUnitSymbolAngstrom uses the name without accents.

5.16 Other options

locale siunitx allows the user to switch between the typographic conventions of different (geographical) areas by using locales. Currently, the package is supplied with configurations for locales UK, US, DE (Germany), FR (French) and ZA (South Africa). The locale option is used to switch to a particular locale.

1.234 m	`\SI{1.234}{\metre}\\`
6,789 m	`\SI[locale = DE]{6.789}{\metre}`

strict Some users will want to stick closely to the official rules for typesetting units. This could be made complicated if the options for non-standards behaviour could not be turned off. The preamble-only option strict resets package behaviour to follow the rules closely, and disables options which deviate from this. If the package is loaded with the strict option, all output is made using the upright (serif) font.

5.17 Local configurations

The siunitx package will check for a local configuration file siunitx.cfg during package loading. This occurs before applying any setting given in the optional argument to \usepackage. A typical configuration file may include settings (using \sisetup) and locally-defined units, for example

```
\ProvidesFile{siunitx.cfg}
\sisetup{
  output-decimal-marker = {,},
  per-mode              = symbol,
}
\DeclareSIUnit\torr{torr}
```

As units are always declared, overwriting any existing definition, units may safely be created in the configuration file even when also included in individual LaTeX document headers.

Installing a local configuration file on your system is very much like doing a local installation of a package. The exact method depends on the TeX system in use. For advice on this, a good start is the TeX.SX question on local installation.

6 Localisation

The translator package provides a structured framework for localisation of words and phrases, and is part of the larger beamer bundle. The translator package provides the \translate macro, which will provide appropriate translations based on the current babel or polyglossia language setting.

If translator is available, siunitx will load it and alter the standard settings for the list-final-separator and range-phrase options to read:

```
\sisetup{
  list-final-separator = { \translate{and} },
  list-pair-separator  = { \translate{and} },
  range-phrase         = { \translate{to (numerical range)} },
}
```

If the current language is known to the translator package then the result will be localised text. The preamble for this manual loads English, French, German and Spanish as options, and also loads the babel package:

1 m, 2 m and 3 m
1 °C to 10 °C
1 m, 2 m et 3 m
1 °C à 10 °C
1 m, 2 m und 3 m
1 °C bis 10 °C
1 m, 2 m y 3 m
1 °C a 10 °C

```
% In English by default
\SIlist{1;2;3}{\metre} \\
\SIrange{1}{10}{\degreeCelsius} \\
\selectlanguage{french}%
\SIlist{1;2;3}{\metre} \\
\SIrange{1}{10}{\degreeCelsius} \\
\selectlanguage{german}%
\SIlist{1;2;3}{\metre} \\
\SIrange{1}{10}{\degreeCelsius} \\
\selectlanguage{spanish}%
\SIlist{1;2;3}{\metre} \\
\SIrange{1}{10}{\degreeCelsius} \\
```

Note that the in order for this to work correctly, languages should be given as global (class) options rather than as package options for babel.

7 Hints for using siunitx

7.1 Ensuring text or math 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.

7.2 Expanding content in tables

When processing tables, siunitx will expand anything stored inside a macro, unless it is long or protected. LaTeX 2ε robust commands are also detected and are not expanded (Table 44). Values which would otherwise be expanded can be protected by wrapping them in a set of braces. As TeX itself will expand the first token in a table cell before siunitx can act on it, using the ε-TeX protected mechanism is the recommended course of action to prevent expansion of macros in table cells. (As is shown in the table, TeX's expansion of LaTeX 2ε robust commands can lead to unexpected results.)

```
\begin{table}
  \centering
  \caption{Values as macros in \texttt{S} columns.}
  \label{tab:xmpl:macro}
  \newcommand*\myvaluea{1234}
  \newcommand\myvalueb{1234}
  \DeclareRobustCommand*\myvaluec{1234}
  \protected\def\myvalued{1234}
  \begin{tabular}{S}
  \toprule
```

Table 44 – Values as macros in `S` columns.

Some Values
12 348.812 34
12 348.8 1234
12348.8 1234
12348.81234
1234 8.8 1234

```
    {Some Values} \\
  \midrule
     \myvaluea  8.8 \myvaluea  \\ % Both expanded
     \myvalueb  8.8 \myvalueb  \\ % First expanded by TeX
                                  % to numbers
     \myvaluec  8.8 \myvaluec  \\ % First expanded by TeX
                                  % but not to numbers!
     \myvalued  8.8 \myvalued  \\ % Neither expanded
    {\myvaluea\ 8.8 \myvaluea} \\ % Neither expanded
  \bottomrule
  \end{tabular}
\end{table}
```

It is possible to use calculated values in tables. For this to work, the calculation must take place before attempting to parse the number. An added complication is that TeX itself will expand the first macro in a table cell until it finds something unexpandable. The ε-TeX protected mechanism can be used to prevent this; using the etoolbox package provides a convenient way to apply this protection to existing functions. The general approach is illustrated in Table 45. The macro `\DTLmul` is made robust inside the table using the `\robustify` command from etoolbox, before constructing the table using an extra column to contain the calculation.

```
\DTLnewdb{data}
\DTLnewrow{data}\DTLnewdbentry{data}{value}{66.7012}
\DTLnewrow{data}\DTLnewdbentry{data}{value}{66.0212}
\DTLnewrow{data}\DTLnewdbentry{data}{value}{64.9026}
\begin{table}
  \caption{Calculated values.}
  \label{tab:xmpl:calc}
  \centering
  \robustify\DTLmul
  \sisetup{
    table-number-alignment = center,
    table-figures-integer  = 2,
    table-figures-decimal  = 4
  }
  \begin{tabular}{
    S
    S[table-figures-integer = 3]
    @{}l
  }
    \toprule
```

Table 45 – Calculated values.

Value	Doubled
66.7012	133.402 400 000
66.0212	132.042 400 000
64.9026	129.805 200 000

Table 46 – Displaying a datatool table.

value
6.7012
66.0212
64.902

```
      {Value} & {Doubled}  &
    \DTLforeach{data}{\myvalue=value}{%
      \DTLiffirstrow {\\ \midrule}{\\}%
      \myvalue & % First column
      \DTLmul{\myvalue}{\myvalue}{2} \myvalue % second column
      & }\\
    \bottomrule
  \end{tabular}
\end{table}
```

7.3 Using **siunitx** with **datatool**

As illustrated in Table 45, siunitx can be used to typeset data stored using datatool. For quickly displaying the contents of tables, datatool offers the \DTLshowtable macro. This will only work with S columns if number parsing is turned off (Table 46).

```
\DTLnewdb{moredata}
\DTLnewrow{moredata}\DTLnewdbentry{moredata}{value}{ 6.7012}
\DTLnewrow{moredata}\DTLnewdbentry{moredata}{value}{66.0212}
\DTLnewrow{moredata}\DTLnewdbentry{moredata}{value}{64.902 }
\begin{table}
  \caption{Displaying a \textsf{datatool} table.}
  \label{tab:xmpl:datatool}
  \centering
  \sisetup{
    parse-numbers           = false,
    table-number-alignment = center,
    table-figures-integer  = 2,
    table-figures-decimal  = 4
  }
  \renewcommand*\dtlrealalign{S}
  \DTLdisplaydb{moredata}
\end{table}
```

7.4 Using units such as $\mu m\,s^{-1}$ 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}%
  }
```

7.5 A left-aligned column visually centred under a heading

When you have a column of non-related numbers, the usual advice is to make these left-aligned and then centre the resulting column under the heading. With the dcolumn package, that would be done with something like `D{x}{}{5.0}`. This is something of an abuse of the nature of a number, but can also be done using siunitx (47).

```
\begin{table}
  \caption{Formatting unrelated numbers}
  \label{tbl:xmpl:unrel}
  \centering
  \begin{tabular}
    {
      S[
        table-format = 5.0,
        parse-numbers = false,
        input-symbols=.,
        input-decimal-markers = x
      ]
    }
    \toprule
    \multicolumn{1}{c}{Header} \\
    \midrule
    120   \\
    12.3  \\
    12340 \\
    12.02 \\
    123   \\
    1     \\
    \bottomrule
  \end{tabular}
\end{table}
```

7.6 Symbols and XeTeX

A small number of non-Latin symbols are needed by siunitx, notably Ω and μ. The package picks glyphs for these which are correct in the sense that they are upright

Table 47 – Formatting unrelated numbers

Header
120
12.3
12340
12.02
123
1

(not italic) symbols, and match the LaTeX standard Computer Modern font. However, this does not make them the best choice if other fonts are in use, which is particularly common when XeTeX is being used.

XeTeX users will probably need to choose appropriate symbols themselves. The correct choice depends on the fonts in use, but many system fonts include Greek letters and other symbols (which is not the case with most TeX-specific fonts). An appropriate setting could then be to use the text µ symbol in all cases:

```
\sisetup{
  math-micro = \text{µ},
  text-micro  = µ
}
```

It may also be desirable in these cases to select a fixed font using the fontspec package, for example

```
\sisetup{
  math-micro = \fontspec{Minion Pro} \textmu,
  text-micro = \fontspec{Minion Pro} \textmu
}
```

7.7 Scaled document fonts with XeTeX

The fontspec package makes it possible to scale the document body font. This can lead to unexpected problems with printing for siunitx, as some symbols will not scale while numbers and text will. The problem is best avoided by forcing siunitx to use the default math font for all output:

```
\sisetup{
  mode    = math,
  math-rm = \ensuremath
}
```

This will cause all siunitx output *not* to scale at all, consistent with other mathematical content.

7.8 Interaction with tex4ht

siunitx will detect when tex4ht is in use, and makes some changes to the way output is printed. Text mode printing is automatically selected, and certain items (such as spaces) are printed in text mode rather than as math. This is designed to reduce the likelihood of spurious formulae appearing in, for example, output converted to OpenOffice format.

7.9 Maximising performance

Both the number and unit parsers require significant effort in terms of TeX programming. For input that does not require such processing, the maximum performance for siunitx can therefore be obtained by turning off both systems:

7.3 Hz
7.3 Hz
7.3 Hz

```
\SI{7.3}{\Hz} \\
\SI[parse-units = false]{7.3}{\Hz} \\
\SI[
  parse-numbers = false,
  parse-units   = false
]{7.3}{\Hz}
```

7.10 Transferring settings to pgf

\SendSettingsToPgf The numerical engine in the pgf package has settings similar to those in siunitx. To enable working with both packages easily, the macro \SendSettingsToPgf is available. It will set some commonly-used numerical formatting options in pgf to the current values used by siunitx to make using the two packages together more convenient for end users. This function can be used at any point after loading both the pgf and siunitx packages.

```
\documentclass{article}
\usepackage{pgf,siunitx}
\sisetup{...}
\SendSettingsToPgf
...
```

7.11 Using siunitx with the cellspace package

Both siunitx and cellspace use the letter S for a new column type. This obviously leads to a problem. If both are loaded, siunitx will retain the S column, and moves the functionality of cellspace to the letter C. This allows the normal use of cellspace with standard column types: it does *not* work with the siunitx S or s columns.

7.12 Special considerations for the \kWh unit

The abbreviations configuration file provides the unit \kWh, which is set up with no spacing between the 'kW' and the 'h' unit to give 'kWh'. However, this only applies when the unit is given on its own: combinations will follow the normal rules

kWh
$\mathrm{kW\,h\,m^{-1}}$

```
\si{\kWh} \\
\si{\kWh\per\metre}
```

This is because the unit \kWh is defined so that it can still be varied by altering \kilo, \watt and \hour, and so that the prefix can still be turned into a number. However, some users may prefer to have a non-flexible macro which never adds a space. This can be achieved by redefining \kWh with \DeclareSIUnit, by added an alternative definition

```
\DeclareSIUnit\kWh{kWh}
\DeclareSIUnit\KWH{kWh}
```

or of course by using literal unit input.

kWh m^{-1}	`\si{\KWH\per\metre}\\`
kWh m^{-1}	`\si{kWh.m^{-1}}`

Another point to notice is that the `\per` macro applies to the next unit, and not an entire unit combination. Thus in

cd kW^{-1} h	`\si{\candela\per\kWh}`

`\per` applies to the watts but not to the hours. In this case, the units need to be written out in full or the `sticky-per` option should be used.

```
\si{\candela\per\kilo\watt\per\hour} \\
\si[sticky-per]{\candela\per\kWh}
```

cd kW^{-1} h^{-1}
cd kW^{-1} h^{-1}

7.13 Adding items after the last column of a tabular

When using the array package '<' construct to insert material after an `S` or `s` column, the alignment of the final column may be wrong if the standard tabular row terminator `\\` is used. This is due to the way that LaTeX constructs tables at a low level. The incorrect spacing can be avoided by using the TeX `\cr` primitive to end the table rows (Table 48).

```
\begin{table}
  \caption{Correcting spacing in last \texttt{S} column}
  \label{tab:cr}
  \hfil
  \begin{tabular}{S<{\,\si{\kg}}S<{\,\si{\kg}}}
    \toprule
    \multicolumn{1}{c}{Long header} &
      \multicolumn{1}{c}{Long header} \\
    \midrule
    1.23 & 1.23 \\
    4.56 & 4.56 \\
    7.8  & 7.8  \\
    \bottomrule
  \end{tabular}
  \hfil
  \begin{tabular}{S<{\,\si{\kg}}S<{\,\si{\kg}}}
    \toprule
    \multicolumn{1}{c}{Long header} &
      \multicolumn{1}{c}{Long header} \\
    \midrule
    1.23 & 1.23 \cr
    4.56 & 4.56 \cr
    7.8  & 7.8  \cr
    \bottomrule
  \end{tabular}
  \hfil
\end{table}
```

Table 48 – Correcting spacing in last S column

Long header	Long header
1.23 kg	1.23 kg
4.56 kg	4.56 kg
7.8 kg	7.8 kg

Long header	Long header
1.23 kg	1.23 kg
4.56 kg	4.56 kg
7.8 kg	7.8 kg

7.14 Creating a column with numbers and units

Usually, numbers in a table should be given with the units in the column heading. However, there are cases where a series of data are best presented in a table but have different units. There are two ways to do this (Table 49). The first is to place the units in the first column of the table, which makes sense if there are several related items in the table. The second method is to generate two columns, one for numbers and a second for units, and then to format these to give the visual effect of a single column. The later effect is most appropriate when only one set of numbers are presented in a table.

```
\begin{table}
  \caption{Tables where numbers have different units}
  \label{tab:xmpl:mixed}
  \hfil
  \begin{tabular}
    {
      >{$}l<{$}
      S[table-format = 2.3(1)]
      S[table-format = 3.3(1)]
    }
    \toprule
      & {One} & {Two} \\
    \midrule
    a / \si{\angstrom}    &  1.234(2) &   5.678(4) \\
    \beta / \si{\degree} & 90.34(4)  & 104.45(5)  \\
    \mu / \si{\per\mm}    &  0.532    &   0.894    \\
    \bottomrule
  \end{tabular}
  \hfil
  \begin{tabular}
    {S[table-format=1.3]@{\,}s[table-unit-alignment = left]}
    \toprule
    \multicolumn{2}{c}{Heading} \\
    \midrule
    1.234 & \metre   \\
    0.835 & \candela \\
    4.23  & \joule\per\mole \\
    \bottomrule
  \end{tabular}
  \hfil
\end{table}
```

Table 49 – Tables where numbers have different units

	One	Two
$a/\text{Å}$	1.234(2)	5.678(4)
$\beta/°$	90.34(4)	104.45(5)
μ/mm^{-1}	0.532	0.894

Heading
1.234 m
0.835 cd
4.23 J mol^{-1}

Table 50 – Header row in a table

123.456
23.45
123.4
3.456

7.15 Tables with heading rows

A common format for tables is to make the heading row visually distinct using a background colour and bold text. If numbers appear in such a heading row within an `S` column then getting the appearance right can be challenging. The best approach is to make the `\bfseries` macro 'robust' (as demonstrated in Section 7.2), then to use this macro to make the heading cells bold. This approach is illustrated in Table 50, along with the use of `\rowcolor` to provide a background colour.

```
\begin{table}
  \caption{Header row in a table}
  \label{tab:xmpl:headers}
  \robustify\bfseries
  \centering
  \begin{tabular}
    {S[detect-weight,table-format = 3.3]}
    \rowcolor[gray]{0.9}
    \bfseries 123.456 \\
     23.45   \\
    123.4    \\
      3.456 \\
  \end{tabular}
\end{table}
```

7.16 Associating a locale with a babel language

It is possible to instruct the babel package to switch to a particular siunitx locale when changing language. This can be done using the babel `\extras⟨language⟩` system. For example, to associate the `DE` locale with the `german` babel language, the appropriate code would be

```
\addto\extrasgerman{\sisetup{locale = DE}}
```

8 Information for those upgrading

8.1 Upgrading from version 1

The key–value control system of siunitx has been completely rewritten for version 2, and at the same time some of the macros provided by the package have been renamed and reworked. The package can be loaded with a configuration file to provide most of the same options and defaults as in version 1:

```
\usepackage[version-1-compatibility]{siunitx}
```

Many of the options from version 1 map to similar ones in version 2 (Table 51). The correspondence often includes a syntax change: consult details of the new options for the correct syntax for the new options. In some cases, the new approach is different to the older one, and in these cases the most appropriate option new has been listed in the table.

Table 51 – Mapping of version 1 options to version 2.

Version 1	See in version 2
addsign	explicit-sign
allowlitunits	free-standing-units
allowoptarg	unit-optional-argument
allowzeroexp	retain-zero-exponent
anglesep	arc-separator
astroang	angle-symbol-over-decimal
closeerr	close-bracket
closefrac	close-bracket
closerange	close-bracket
colour	color
colorall	color
colourall	color
colorunits	unit-color
colorneg	negative-color
colourneg	negative-color
colourunits	unit-color
colorvalues	number-color
colourvalues	number-color
decimalsymbol	output-decimal-marker
detectdisplay	detect-display-math

Continued on next page

Continued from previous page

Version 1	See in version 2
`digitsep`	`group-separator`
`dp`	`round-mode`
	`round-precision`
`errspace`	`uncertainty-separator`
`expbase`	`exponent-base`
`expproduct`	`exponent-product`
`fixdp`	`round-mode`
`fixsf`	`round-mode`
`fraction`	`fraction-function`
`inlinebold`	`detect-inline-weight`
`locale`	`locale`
`mathOmega`	`math-ohm`
`mathcelsius`	`math-celsius`
`mathdegree`	`math-degree`
`mathminute`	`math-arcminute`
`mathmu`	`math-micro`
`mathringA`	`math-angstrom`
`mathrm`	`math-rm`
`mathsOmega`	`math-ohm`
`mathscelsius`	`math-celsius`
`mathsdegree`	`math-degree`
`mathsecond`	`math-arcsecond`
`mathsf`	`math-sf`
`mathsminute`	`math-arcminute`
`mathsmu`	`math-micro`
`mathsringA`	`math-angstrom`
`mathsrm`	`math-rm`
`mathssecond`	`math-arcsecond`
`mathssf`	`math-sf`
`mathstt`	`math-tt`
`mathtt`	`math-tt`
`mode`	`mode`
`negcolor`	`negative-color`
`negcolour`	`negative-color`
`numaddn`	`input-symbols`
`numcloseerr`	`input-close-uncertainty`

Continued on next page

Continued from previous page

Version 1	See in version 2
`numdecimal`	`input-decimal-markers`
`numdigits`	`input-digits`
`numdiv`	`input-quotient`
`numexp`	`input-exponent-markers`
`numgobble`	`input-ignore`
`numopenerr`	`input-open-uncertainty`
`numprod`	`input-product`
`numsign`	`input-signs`
`obeyall`	`detect-all`
`obeybold`	`detect-weight`
`obeyfamily`	`detect-family`
`obeyitalic`	`detect-shape`
`obeymode`	`detect-mode`
`openerr`	`open-bracket`
`openfrac`	`open-bracket`
`openrange`	`open-bracket`

`padangle`	`add-arc-degree-zero`
	`add-arc-minute-zero`
	`add-arc-second-zero`

`padnumber`	`add-decimal-zero` `add-integer-zero`
`per`	`per-mode`
`prefixsymbolic`	`prefixes-as-symbols`
`prespace`	`space-before-unit`
`redefsymbols`	`redefine-symbols`

`repeatunits`	`multi-part-units`
	`product-units`
	`range-units`

`retainplus`	`retain-explicit-plus`
`seperr`	`separate-uncertainty`
`sepfour`	`group-four-digits`
`sf`	`round-mode` `round-precision`
`sign`	`explicit-sign`
`slash`	`per-symbol`
`stickyper`	`sticky-per`

Continued on next page

Continued from previous page

Version 1	See in version 2
strict	strict
tabalign	table-alignment
tabalignexp	table-align-exponent
tabautofit	table-auto-round
tabformat	table-format
tabnumalign	table-number-alignment
tabparseonly	table-parse-only
tabexpalign	table-align-exponent
tabtextalign	table-text-alignment
tabunitalign	table-unit-alignment
textcelsius	text-celsius
textdegree	text-degree
textminute	text-arcminute
textmode	mode
textmu	text-micro
textOmega	text-ohm
textringA	text-angstrom
textrm	text-rm
textsecond	text-arcsecond
textsf	text-sf
texttt	text-tt
tightpm	tight-spacing
tophrase	range-phrase
trapambigerr	multi-part-units
trapambigfrac	bracket-numbers
trapambigrange	range-units
unitcolor	unit-color
unitcolour	unit-color
unitmathrm	unit-math-rm
unitmathsf	unit-math-sf
unitmathsrm	unit-math-rm
unitmathssf	unit-math-sf
unitmathstt	unit-math-tt
unitmathtt	unit-math-tt
unitmode	unit-mode
unitsep	inter-unit-product

Continued on next page

Continued from previous page

Version 1	See in version 2
unitspace	inter-unit-product
valuecolor	number-color
valuecolour	number-color
valuemathrm	number-math-rm
valuemathsf	number-math-sf
valuemathsrm	number-math-rm
valuemathssf	number-math-sf
valuemathstt	number-math-tt
valuemathtt	number-math-tt
valuemode	value-mode
valuesep	number-unit-product
xspace	use-xspace

A small number of the options from version 1 are used unchanged in version 2, for example the `mode` setting. These are listed above but require no action on the part of the user. There are also a few options which are no longer used at all, and are therefore ignored by the current code.

Loading configuration files has been completely changed, and this means that the options `alsoload`, `load` and `noload` are ignored by version 2. In the same way the options `debug` and `log` are not used by the current release of siunitx, as this information is usually only needed by the package author. Emulation of older packages is no longer offered (it was intended to help with the transition form earlier packages), and so the `emulate` option no longer applies.

8.2 Upgrading from version 2.0 or 2.1

User feedback on siunitx means that over time some renaming takes place. The following functions and options have been deprecated in version 2.2. They are therefore available in version 2.2, but should be replaced in new or updated documents with the successor names.

`angle-unit-separator`
`inter-unit-separator`
`number-unit-separator`

These options have been replaced by the options

- `angle-unit-product`
- `inter-unit-product`
- `number-unit-product`

as these items are formally products, and the new option names emphasise this.

`\DeclareSIUnitWithOptions`

The `\DeclareSIUnit` function has been extended to take a first optional argument, which removes the need for `\DeclareSIUnitWithOptions`. This function is therefore depreciated but retained for compatibility.

8.3 Upgrading from version 2.2

`load-configurations`

The option `load-configurations` has been deprecated in favour of the three options `abbreviations`, `binary-units` and `version-1-compatibility`. At the same time, loading of the abbreviations is now the standard behaviour, and so in most cases no explicit configuration file loading will be needed.

group-decimal-digits
group-integer-digits
group-four-digits

The digit grouping options have been revised, and the options `group-decimal-digits` and `group-integer-digits` are now integrated into `group-digits`. At the same time, the `group-four-digits` option has been extended to the new option `group-minimum-digits`.

literal-superscript-as-power

The new literal-superscript-as-power option means that the standard behaviour now uses the current math font for superscripts, even when units are printed literally. This will only be obvious in documents such as this manual, where the text and math mode numerals are (deliberately) different. To restore the previous behaviour, set `literal-superscript-as-power = false`.

8.4 Upgrading from version 2.3

The number of options which assume that the input is given in math mode has been significantly reduced. As most material can be typeset in either math or text mode, the 'intrinsic' math mode options could lead to inconsistent output. The only options which now force math mode are those for products, which will almost always require the use of math mode.

8.5 Upgrading from version 2.4

The process of removing options which assume math mode, begun with version 2.4, has been taken further, and only `output-product` and `exponent-product` now do not require `\ensuremath` for material which must be in math mode. The standard settings have been altered to take account of this, but user-set options for products may need to be updated accordingly.

8.6 Upgrading from version 2.5

All printing now takes place in math mode (*i.e.* `mode = math` is set). This may alter the appearance of some units, and users may want to verify the output is correct.

9 Correct application of (SI) units

Consistent and logical units are a necessity for scientific work, and have applicability everywhere. Historically, a number of systems have been used for physical units. SI units were introduced by the *Conférence Générale des Poids et Mesures* (CGPM) in 1960. SI units are a coherent system based on seven base units, from which all other units may be derived.

At the same time, physical quantities with units are mathematical entities, and as such way that they are typeset is important. In mathematics, changes of type (such as using bold, italic, sans serif typeface and so on) convey information. This means that rules exist not only for the type of units to be used under the SI system, but also the way they should appear in print. Advice on best practice has been made available by the *National Institute of Standards and Technology* (NIST) [2].

As befits an agreed international standard, the full rules are detailed. It is not appropriate to reproduce these in totality here; instead, a useful summary of the key points is provided. The full details are available from the *Bureau International des Poids et Mesures* [1].

siunitx takes account of the information given here, so far as is possible. Thus the package defaults follow the recommendations made for typesetting numbers and units. Spacing and so forth is handled in such a way as to make implementing the rules (relatively) easy.

9.1 Units

There are seven base SI units, listed in Table 1. Apart from the kilogram, these are defined in terms of a measurable physical quantity needing the definition alone.[7] The base units have been chosen such that all physical quantities can be expressed using an appropriative combination of these units, needing no others and with no redundancy. The kilogram is slightly different from the other base units as it is still defined in terms of a 'prototype' held in Paris.

All other units within the SI system are regarded as 'derived' from the seven base units. At the most basic, all other SI units can be expressed as combinations of the base units. However, many units (listed in Tables 2 and 3) have a special name and symbol. Most of these units are simple combinations of one or more base units (raised to powers as appropriate). A small number of units derived from experimental data are allowed as SI units (Table 4).

A series of SI prefixes for decimal multiples and sub-multiples are provided, and can be used as modifiers for any SI unit (either base or derived units) with the exception of the kilogram. The prefixes are listed in Table 6. No space should be used between a prefix and the unit, and only a single prefix should be used. Even the degree Celsius can be given a prefix, for example $1\,\mathrm{m}^\circ\mathrm{C}$.

It is important to note that the kilogram is the only SI unit with a prefix as part of its name and symbol. Only single prefix may be used, and so in the case of the kilogram prefix names are used with the unit name 'gram' and the prefix symbols are used with the unit symbol g. For example $1 \times 10^{-6}\,\mathrm{kg} = 1\,\mathrm{mg}$.

The application of SI units is meant to provide a single set of units which ensure consistency and clarity across all areas. However, other units are common is many areas, and are not without merit. The units provided by siunitx by default do not include any of these; only units which are part of the SI set or are accepted for use with SI units are defined. However, several other sets of units can be loaded as optional modules. The binary prefixes and units (Table 22) are the most obvious example. These are *not* part of the SI specifications, but the prefix names are derived from those in Table 6.

Other units are normally to be avoided where possible. SI units should, in the main, be preferred due to the advantages of clear definition and self-consistency this brings. However, there will probably always be a place for specialist or non-standard units. This is particularly true of units derived from basic physical constants.

There are also many areas where non-standard units are used so commonly that to do otherwise is difficult or impossible. For example, most synthetic chemists measure the pressure inside vacuum apparatus in mmHg, partly because the most common gauge for the task still uses a column of mercury metal. For these reasons, siunitx does define non-SI units.

9.2 Mathematical meaning

As explained earlier, a number–unit combination is a single mathematical entity. This has implications for how both the number and the unit should be printed. Firstly, the

[7] Some base units need others defined first; there is therefore a required order of definition.

two parts should not be separated: a quantity is a product of the number and the unit. With the exception of the symbols for plane angles (°, ′ and ″), the BIPM specifies either a space or half-height (centred) dot should be used [1].

```
A space for \SI{10}{\percent}\\
and also for \SI{100}{\degreeCelsius}\\
but not for \ang{1.23}.
```

A space for 10 %
and also for 100 °C
but not for 1.23°.

The mathematical meaning of units also means that the shape, weight and family are important. Units are supposed to be typeset in an upright, medium weight font. Italic, bold and sans serif are all used mathematically to convey other meanings. (In an all sans serif document, using sans serif for units is reasonable.) The siunitx package defaults again follow this convention: any local settings are ignored, and uses the current upright math font. However, there are occasions where this may not be the most desirable behaviour. A classic example would be in an all-bold section heading. As the surrounding text is bold, some people feel that any units should follow this.

```
Units should \textbf{not be bold: \SI{54}{\farad}}\\
\textbf{But perhaps in a running block,\\
it might look better: \SI[detect-weight]{54}{\farad}}
```

Units should **not be bold:** 54 F
But perhaps in a running block,
it might look better: 54 F

Symbols for units formed from other units by multiplication are indicated by means of either a half-height (that is, centred) dot or a (thin) space.

```
\( \si{\metre\second} = \text{metre second} \) \\
\( \si{\milli\second} = \text{millisecond}  \) \\
\sisetup{inter-unit-product = \ensuremath { { } \cdot { } } }
\( \si{\metre\second} = \text{metre second} \) \\
\( \si{\milli\second} = \text{millisecond}  \)
```

$\mathrm{m\,s} = \text{metre second}$
$\mathrm{ms} = \text{millisecond}$
$\mathrm{m \cdot s} = \text{metre second}$
$\mathrm{ms} = \text{millisecond}$

There are some circumstances under which it is common practice to omit any spaces. The classic example is kWh, where 'kW h' does not add any useful information. If using such a unit repeatedly, users of siunitx are advised to create a custom unit to ensure consistency. It is important to note that while this is common practice, it is *not* allowed by the BIPM [1].

Symbols for units formed from other units by division are indicated by means of a virgule (oblique stroke, slash, /), a horizontal line, or negative exponents.[8] However, to avoid ambiguity, the virgule must not be repeated on the same line unless parentheses are used. This is ensured when using named unit macros in siunitx, which will 'trap' repeated division and format it correctly. In complicated cases, negative exponents are to be preferred over other formats.

[8] Notice that a virgule and a solidus are not the same symbol.

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

$\mathrm{J\,mol^{-1}\,K^{-1}}$
$\frac{\mathrm{J}}{\mathrm{mol\,K}}$
$\mathrm{J/(mol\,K)}$

Products and errors should show what unit applies to each number given. Thus (2×3) m is an ordered set of lengths of a geometric area, whereas 2×3 m is a length (and equal to 6 m). Thus, $\times$ is not a product but is a mathematical operator; in the same way, a 2×3 matrix is not a 6 matrix! In some areas, areas and volumes are given with separated units but a unit raised to the appropriate power: $2 \times 3\,\mathrm{m}^2$. Although this does display the correct overall units, it is potentially-confusing and is not encouraged.

Care must be taken when writing ranges of numbers. For purely numerical values, it is common to use a en-dash to show a range, for example 'see pages 1–5'. On the other hand, physical quantities could be misinterpret as negative values if written in this way. As the unit–number combination is a single mathematical entity, writing the values with an en-dash followed by a single unit is also incorrect. As a result, using the word 'to' is strongly recommended.

1 m to 5 m long.

```
\SIrange{1}{5}{\metre} long.
```

9.3 Graphs and tables

In graphs and tables, repetition of the units following each entry or axis mark is confusing and repetitive. It is therefore best to place the unit in the label part of the information. Placing the unit in square brackets is common but mathematically poor.[9] Much better is to show division of all quantities by the unit, which leaves the entries as unitless ratios. This is illustrated in Table 52 and Figure 1.

```
\begin{table}
  \centering
  \caption{An example of table labelling.}
  \label{tab:xmpl:unitless}
  \sisetup{
    table-number-alignment = center,
    table-figures-integer  = 1,
    table-figures-decimal  = 4
  }
  \begin{tabular}{cS}
    \toprule
      Entry & {Length/\si{\metre}} \\
    \midrule
      1 & 1.1234 \\
      2 & 1.1425 \\
      3 & 1.7578 \\
      4 & 1.9560 \\
    \bottomrule
  \end{tabular}
\end{table}
```

[9] For example, for an acceleration a, the expression $[a]$ is the dimensions of a, *i.e.* length per time squared in this case.

Table 52 – An example of table labelling.

Entry	Length/m
1	1.1234
2	1.1425
3	1.7578
4	1.9560

```
\begin{figure}
  \centering
  \begin{tikzpicture}
    \begin{axis}[
      xlabel = \( t/\si{\second} \),
      xmax   = 6,
      xmin   = 0,
      ylabel = \( d/\si{\metre}  \),
      ymin   = 0
    ]
      \addplot[smooth,mark=*]
        plot coordinates {
          (0,0)
          (1,5)
          (2,8)
          (3,9)
          (4,8)
          (5,5)
          (6,0)
         };
    \end{axis}
  \end{tikzpicture}
  \caption{An example of graph labelling.}
  \label{fig:xmpl:unitless}
\end{figure}
```

In most cases, adding exponent values in the body of a table is less desirable than adding a fixed exponent to column headers. An example is shown in Table 53. The use of \multicolumn is needed here due to the '<'; without \multicolumn, the titles are followed by 'kg'!

```
\begin{table}
  \centering
  \caption{Good and bad columns.}
  \label{tab:good}
  \sisetup{table-number-alignment = center}
  \begin{tabular}{
    c
    S[
      table-figures-integer  = 1,
      table-figures-decimal  = 3,
      table-figures-exponent = 1
    ]
```

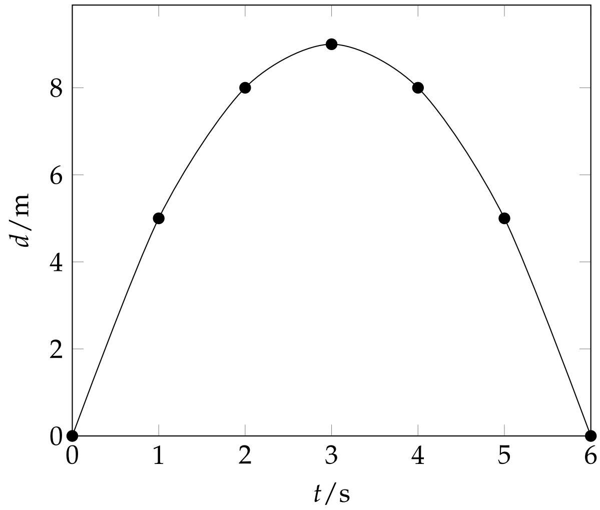

Figure 1 – An example of graph labelling.

Table 53 – Good and bad columns.

Entry	Mass	Mass/10^3 kg
1	4.56×10^3 kg	4.56
2	2.40×10^3 kg	2.40
3	1.345×10^4 kg	13.45
4	4.5×10^2 kg	0.45

```
    @{\,\si{\kilogram}}
    S[
      table-figures-integer  = 2,
      table-figures-decimal  = 2
    ]
  }
    \toprule
      Entry & \multicolumn{1}{c}{Mass} &
        {Mass/\SI{e3}{\kilogram}} \\
    \midrule
      1 & 4.56e3  & 4.56 \\
      2 & 2.40e3  & 2.40 \\
      3 & 1.345e4 & 13.45 \\
      4 & 4.5e2   & 0.45 \\
    \bottomrule
  \end{tabular}
\end{table}
```

10 Making suggestions and reporting bugs

Feedback on siunitx is always welcome, either to make suggestions or to report problems. When sending feedback, it is always useful if a small example file is included, showing the bug being reported or illustrating the desired output. It is helpful if a 'reference rendering' is included, showing what the output should look like. A typical example file might read

```
\listfiles
% Use the article class unless the problem is class-dependent
\documentclass{article}
\usepackage{siunitx}
% Other packages loaded as required
\begin{document}
Reference output: $ 1.23\,\mathrm{m} $

siunitx output: \SI{1.23}{\metre}
\end{document}
```

As illustrated, it is usually best to use the `article` class and to only load packages which are needed to show the issue. It is also useful to include a copy of the log file generate by TeX when reporting a bug (as the versions of packages can be important to solving the issue).

Feedback can be sent in a range of ways. The development code is hosted by GitHub, and the site includes an issue tracker. Adding feedback directly to the database means that other users can see it, and also ensures that it does not get forgotten. E-mailing directly will also definitely get attention: joseph.wright@morningstar2.co.uk.

11 Code level interfaces

At present, most interfaces in siunitx at the code level are internal. This largely reflects the parallel between siunitx and core expl3 development: much of the code here was written whilst the LaTeX team were defining how best to handle this area. Development work on the next major release of siunitx includes a full set of documented code level interfaces. Some of these will be added to this (release) version of the package over time, where this is appropriate.

`\l_siunitx_unit_symbolic_seq`

This sequence contains all of the symbolic ⟨*unit*⟩ names defined: these will be in the form of control sequences such as `\kilogram`. The order of the sequence is unimportant.

12 Thanks

Many users have provided feedback, bug reports and ideas for new features for siunitx: thanks to all of them. Particular thanks to Stefan Pinnow, who has taken the lead role as beta tester for siunitx, finding incorrect output, bad documentation and the odd spelling mistake in the documentation. Thanks also to Danie Els and Marcel Heldoorn for the SIstyle and SIunits packages, respectively, which provided the starting point for the development of siunitx.

References

[1] *The International System of Units (SI)*, http://www.bipm.org/en/measurement-units/.

[2] *International System of Units from NIST*, http://physics.nist.gov/cuu/Units/index.html.

[3] *SI base units*, http://www.bipm.org/en/publications/si-brochure/section2-1.html.

[4] *Units with special names and symbols; units that incorporate special names and symbols*, http://www.bipm.org/en/publications/si-brochure/section2-2-2.html.

[5] *SI Prefixes*, http://www.bipm.org/en/publications/si-brochure/chapter3.html.

[6] *Non-SI units accepted for use with the International System of Units*, http://www.bipm.org/en/publications/si-brochure/table6.html.

[7] *Non-SI units whose values in SI units must be obtained experimentally*, http://www.bipm.org/en/publications/si-brochure/table7.html.

[8] *Other non-SI units*, http://www.bipm.org/en/publications/si-brochure/table8.html.

[9] *Formatting the value of a quantity*, http://www.bipm.org/en/publications/si-brochure/section5-3-3.html.

Change History

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.