

siunitx — A comprehensive (SI) units package*

Joseph Wright[†]

Released 2011/04/04

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

1	Introduction	3
2	Installation	3
3	siunitx for the impatient	4
4	Using the siunitx package	6
4.1	Loading the package	6
4.2	Numbers	6
4.3	Units	7

*This file describes v2.1p, last revised 2011/04/04.

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

4.4	The unit macros	9
4.5	Creating new macros	12
4.6	Tabular material	14
5	The key-value control system	17
5.1	Detecting fonts	17
5.2	Output font families	19
5.3	Parsing numbers	20
5.4	Post-processing numbers	22
5.5	Printing numbers	25
5.6	Multi-part numbers	30
5.7	Angles	31
5.8	Creating units	32
5.9	Loading additional units	33
5.10	Using units	38
5.11	Numbers with units	41
5.12	Tabular material	44
5.13	Symbols	57
5.14	Other options	58
5.15	Local configurations	58
6	Localisation	58
7	Hints for using siunitx	59
7.1	Ensuring text or math output	59
7.2	Expanding content in tables	59
7.3	Using siunitx with datatool	61
7.4	Using units such as $\mu\text{m s}^{-1}$ in headings	62
7.5	Symbols and X _g T _E X	62
7.6	Scaled document fonts with X _g T _E X	63
7.7	Interaction with tex4ht	63
7.8	Maximising performance	63

7.9	Transferring settings to <code>pgf</code>	64
7.10	Using <code>siunitx</code> with the <code>cellspace</code> package	64
7.11	Special considerations for the <code>\kW</code> unit	64
7.12	Adding items after the last column of a <code>tabular</code>	65
7.13	Creating a column with numbers and units	66
7.14	Tables with heading rows	67
7.15	Associating a locale with a <code>babel</code> language	68
8	Information for those upgrading from version 1	68
9	Correct application of (SI) units	73
9.1	Units	73
9.2	Mathematical meaning	74
9.3	Graphs and tables	76
10	Making suggestions and reporting bugs	78
11	Thanks	79
	Change History	80
	Index	83

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 L^AT_EX₃ support as provided in the `expl3` and `xpackages` bundles. Both of these are available on [CTAN](#) as ready-to-install zip files. Suitable versions are available in MiK_TE_X 2.9 and T_EX Live 2010 (updating the relevant packages online may be necessary). L^AT_EX₃, and so `siunitx`, requires the ϵ -T_EX extensions: these are available on all modern T_EX systems.

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

- `amsmath`
- `booktabs`
- `caption`
- `cleveref`
- `csquotes`
- `helvet`
- `mathpazo`
- `listings`
- `pgfplots`
- `xcolor`

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

3 `siunitx` for the impatient

The package provides the user macros:

- `\num[options]{number}`

- `\si[options]{unit}`
- `\SI[options]{number}[pre-unit]{unit}`
- `\numlist[options]{numbers}`
- `\numrange[options]{numbers}{number2}`
- `\SIlist[options]{numbers}{unit}`
- `\SIRange[options]{number1}{number2}{unit}`
- `\ang[options]{angle}`
- `\sisetup{options}`

plus the S and s column types for decimal alignments and units in tables. These macros are designed for typesetting 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	<code>\num{12345,67890}</code>	<code>\\</code>
$1 \pm 2i$	<code>\num{1+-2i}</code>	<code>\\</code>
0.3×10^{45}	<code>\num{.3e45}</code>	<code>\\</code>
$1.654 \times 2.34 \times 3.430$	<code>\num{1.654 x 2.34 x 3.430}</code>	

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

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

Simple lists and ranges of numbers can be handled.

<code>\numlist{10;20;30}</code>	<code>\\</code>
<code>\SIlist{0.13;0.67;0.80}{\milli\metre}</code>	<code>\\</code>
<code>\numrange{10}{20}</code>	<code>\\</code>
<code>\SIRange{0.13}{0.67}{\milli\metre}</code>	
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, serif 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 L^AT_EX 2_ε way.

```
\usepackage{siunitx}
```

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

```
\usepackage[load-configurations = version-1]{siunitx}
```

to use options from version 1 of the package.

4.2 Numbers

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

Numbers are automatically formatted by the `\num` macro. This takes one optional argument, *options*, and one mandatory one, *number*. The contents of *number* are automatically formatted. The formatter removes ‘hard’ spaces (`\`, and `~`), automatically identifies exponents (by default marked using `e`, `E`, `d` or `D`) and adds the appropriate spacing of large numbers. With the standard settings a leading zero is added before a decimal marker, if needed: both `'.'` and `' , '` are recognised as decimal markers.

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

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 \numlist[options]{numbers}
```

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` `\numrange[options]{number1}{number2}`
 Simple ranges of numbers can be handled using the `\numrange` function. This acts in the same way as `\num`, but inserts a phrase or other text between the two entries. This function should be used in text mode.

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

`\ang` `\ang[options]{angle}`
 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°	<code>\ang{10}</code>	<code>\</code>
12.3°	<code>\ang{12.3}</code>	<code>\</code>
4.5°	<code>\ang{4,5}</code>	<code>\</code>
1°2'3"	<code>\ang{1;2;3}</code>	<code>\</code>
1"	<code>\ang{;;1}</code>	<code>\</code>
10°	<code>\ang{+10;;}</code>	<code>\</code>
-0°1'	<code>\ang{-0;1;}</code>	

4.3 Units

`\si` `\si[options]{unit}`
 The symbol for a unit can be typeset using the `\si` macro: this provides full control over output format for the unit. Like the `\num` macro, `\si` takes one optional and one mandatory argument. The unit formatting system can accept two types of input. When the *unit* contains literal items (for example letters or numbers) then siunitx converts . and ~ into inter-unit separators and correctly positions sub- and superscripts specified using _ and ^. The formatting methods will work with both math and text mode.

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

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

¹The 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.

```

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

```

```

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

```

On its own, this is less convenient than the direct method, although it does use meaning rather than appearance for input. However, the 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` `\SI[options]{number}[preunit]{unit}`

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 `unit` arguments work exactly like those for the `\num` and `\si` macros, respectively. `preunit` is a unit to be typeset *before* the numerical value (most likely to be a currency).

```

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

```

```

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

```

It is possible 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 5.8.

`\SIlist` `\SIlist[options]{numbers}{unit}`

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}

```


Table 1: SI base units.

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

`\SIrange` `\SIrange[options]{number1}{number2}{unit}`
 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}`

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`.

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 official sanctioned [8].

`\deka` In addition to the units themselves, `siunitx` provides pre-defined macros for all of the SI prefixes (Table 6, [5]). The spelling ‘`\deka`’ is provided for US users as an alternative to `\deca`.

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

`\cubic`
`\cubed`

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

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

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

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

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

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

Table 5: Other non-SI units.

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

Table 6: SI prefixes.

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

Bq ²	<code>\si{\square\becquerel} \\\</code>
J ² lm ⁻¹	<code>\si{\joule\squared\per\lumen} \\\</code>
lx ³ VT ³	<code>\si{\cubic\lux\volt\tesla\cubed}</code>

`\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	<code>\si{\henry\tothe{5}} \\\</code>
rad ^{4.5}	<code>\si{\raiseto{4.5}\radian}</code>

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

Jmol ⁻¹ K ⁻¹	<code>\si{\joule\per\mole\per\kelvin} \\\</code>
Jmol ⁻¹ K	<code>\si{\joule\per\mole\kelvin} \\\</code>
H ⁻⁵	<code>\si{\per\henry\tothe{5}} \\\</code>
Bq ⁻²	<code>\si{\per\square\becquerel}</code>

`\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}}
kgmetal
0.1 mmol(cat) kg(prod)-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
103
```

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.

<code>\DeclareSIUnit</code> <code>\DeclareSIUnitWithOptions</code>	<code>\DeclareSIUnit{⟨unit⟩}{⟨symbol⟩}</code> <code>\DeclareSIUnitWithOptions{⟨unit⟩}{⟨symbol⟩}{⟨options⟩}</code>
---	--

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 using the `\DeclareSIUnitWithOptions` function, where the *⟨options⟩* argument can be any suitable options, and applies 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:

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

The spacing can still be altered at point of use:

```
\SI{67890}{\degree} \\  

\SI[number-unit-separator = \;]{67890}{\degree}
```

67 890°
67890 °

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.

<code>\DeclareSIPrefix</code> <code>\DeclareBinaryPrefix</code>	<code>\DeclareSIPrefix{⟨prefix⟩}{⟨symbol⟩}{⟨powers-ten⟩}</code> <code>\DeclareBinaryPrefix{⟨prefix⟩}{⟨symbol⟩}{⟨powers-two⟩}</code>
--	--

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}
```

<code>\DeclareSIPostPower</code> <code>\DeclareSIPrePower</code>	<code>\DeclareSIPostPower{⟨power⟩}{⟨num⟩}</code> <code>\DeclareSIPrePower{⟨power⟩}{⟨num⟩}</code>
---	---

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

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

with the functions then used in the document as:

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

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

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

and then in the body the document could read:

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

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

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

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}

```

As a complement to the S column, `siunitx` also provides a second column type, `s`. This is intended for producing columns of units. The letters chosen are intended to be similar

Table 9: Units in tables.

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

to `\SI` and `\si`, respectively. This allows both macro-based and explicit units to be printed easily (Table 9).

```
\begin{table}
  \centering
  \caption{Units in tables.}
  \label{tab:s:demo}
  \begin{tabular}{s}
    \toprule
    \multicolumn{1}{c}{Unit} \\
    \midrule
    \metre\squared\per\second \\
    \pascal \\
    \text{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 10. Notice that the braces 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}
```


Table 10: The `s` column processes everything.

Unit	Unit
m^3	m^3
kg	kg

5 The key–value control system

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

The package uses a range of different key types:

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

Integer Requires a number as the argument.

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

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.

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

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.

5.1 Detecting fonts

The `siunitx` package controls the font used to print output independently of the surrounding material. The standard method is to ignore the surroundings entirely, and to use the current body fonts. However, the package can detect and follow surrounding bold, italic and font family changes. The font detection options are summarised in Table 11.

<code>detect-weight</code>	The options <code>detect-weight</code> and <code>detect-shape</code> set detection of the prevailing font weight and font shape states, respectively. The font shape state is only checked if
<code>detect-family</code>	
<code>detect-shape</code>	
<code>detect-mode</code>	

Table 11: Font detection options.

Option name	Type	Default
<code>detect-all</code>	Meta	<code><none></code>
<code>detect-display-math</code>	Switch	<code>false</code>
<code>detect-family</code>	Switch	<code>false</code>
<code>detect-inline-family</code>	Choice	<code>text</code>
<code>detect-inline-weight</code>	Choice	<code>text</code>
<code>detect-mode</code>	Switch	<code>false</code>
<code>detect-none</code>	Meta	<code><none></code>
<code>detect-shape</code>	Switch	<code>false</code>
<code>detect-weight</code>	Switch	<code>false</code>

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`.

```

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

```

Table 12: Font options (also available as `number-...` and `unit-...` versions).

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

`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.²

```

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

$$x = 1.2 \times 10^3 \text{ kg Kcd}$$

More text

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


```

5.2 Output font families

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

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

²Here, ‘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.

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

```

1234          \sisetup{detect-family}%
1234          \num{1234} \\
99 m          { \sffamily \num{1234} } \\
99 m          \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 the user to load `color` or a more powerful colour package.

```

\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

```

Table 13: 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	<i>none</i>
input-open-uncertainty	Literal	(
input-protect-tokens	Literal	\approx\ge\geq\gg\le \leq\ll\mp\pi\pm\sim
input-signs	Literal	+-\pm\mp
input-symbols	Literal	\pi
parse-numbers	Switch	true

5.3 Parsing numbers

The package uses a sophisticated parsing system to understand numbers. This allows siunitx to carry out a range of formatting, as described later. All of the input options take lists of literal tokens, and are summarised in Table 13.

input-digits	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-decimal-markers	
input-signs	
input-exponent-markers	
input-symbols	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-ignore	
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:
	$\begin{aligned} < 10 & \qquad \qquad \qquad \backslash\text{num}\{< 10\} \backslash\backslash \\ \gg 5\text{m} & \qquad \qquad \qquad \backslash\text{SI}\{>> 5\}\{\text{metre}\} \backslash\backslash \\ \le 0.12 & \qquad \qquad \qquad \backslash\text{num}\{\le 0.12\} \end{aligned}$
input-open-uncertainty	In some fields, it is common to give the uncertainty in a number in brackets
input-close-uncertainty	

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`.

`input-complex-roots` When using complex numbers in input, the complex root ($i = \sqrt{-1}$) 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 + 88.8i          \num{9.99 + 88.8i} \\
9.99 + 88.8i          \num{9.99 + i88.8}
```

`input-protect-tokens` Some symbols can be problematic under expansion in $\text{\LaTeX} 2_{\epsilon}$. To allow these to be used in input without issue, the package can protect these tokens while expanding input. Symbols to be protected in this way should be listed in `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}

 $\sqrt{2}$ 
 $\sqrt{3}\text{m}$ 
```

5.4 Post-processing numbers

Before typesetting numbers, various post-processing steps can be carried out. These involve adding or removing information from the number in a systematic way; the options are summarised in Table 14.

`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) is set using the `round-precision` option. No rounding will take place if the number contains an uncertainty component.

Table 14: Number post-processing options.

Option name	Type	Default
add-decimal-zero	Switch	true
add-integer-zero	Switch	true
explicit-sign	Literal	<i>none</i>
retain-explicit-plus	Switch	false
retain-unity-mantissa	Switch	true
retain-zero-exponent	Switch	false
round-integer-to-decimal	Switch	false
round-minimum	Literal	0
round-mode	Choice	off
round-precision	Integer	2
scientific-notation	Switch	false
scientific-notation-exponent	Integer	0

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

`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	<code>\num[round-mode = figures]{1} \\</code>
1	<code>\num[round-mode = places]{1} \\</code>
1.0	<code>\ssetup{round-integer-to-decimal}</code>
1.00	<code>\num[round-mode = figures]{1} \\</code>
	<code>\num[round-mode = places]{1}</code>

`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          \ssetup{round-mode = places}
0.00          \num{0.0055} \\
0.01          \num{0.0045} \\
< 0.01       \ssetup{round-minimum = 0.01}%
              \num{0.0055} \\
              \num{0.0045}

```

`add-decimal-zero` It is possible to give real (floating point) numbers as input omitting the decimal
`add-integer-zero` 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        \num{123.} \\
456          \num{456} \\
0.789        \num{.789} \\
123.         \ssetup{
456          add-decimal-zero = false,
.789         add-integer-zero = false,
            }%
            \num{123.} \\
            \num{456} \\
            \num{.789}

```

`explicit-sign` The inclusion of a leading plus sign is usually unnecessary for positive numbers, and
`retain-explicit-plus` so the `retain-explicit-plus` option is available to control whether these are printed. 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          \num{+345} \\
+345        \num[retain-explicit-plus]{+345} \\
-345        \num[explicit-sign = -]{345} \\
345         \num[explicit-sign = -]{+345}

```

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

```

\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 Numbers can be converted to scientific notation by the package. This is controlled by
fixed-exponent the `scientific-notation` option, which takes choices `false`, `true` and `fixed`. In the
 later case, the exponent value used will be set by the `fixed-exponent` option.

	<code>\num{0.001} \\</code>
	<code>\num{0.00100} \\</code>
	<code>\num{1200} \\</code>
0.001	<code>\ssetup{scientific-notation = true}%</code>
0.00100	<code>\num{0.001} \\</code>
1200	<code>\num{0.00100} \\</code>
1×10^{-3}	<code>\num{1200} \\</code>
1.00×10^{-3}	<code>\ssetup{</code>
1.200×10^3	<code>fixed-exponent = 2,</code>
0.00001×10^2	<code>scientific-notation = fixed,</code>
0.0000100×10^2	<code>}%</code>
12.00×10^2	<code>\num{0.001} \\</code>
	<code>\num{0.00100} \\</code>
	<code>\num{1200}</code>

5.5 Printing numbers

Actually printing numbers is controlled by a number of settings, which apply ideas such as differing decimal markers, digit grouping and so on. All of these options are concerned with the appearance of output, rather than the data it conveys. The options are summarised in Table 15.

group-digits Grouping digits into blocks of three is a common method to increase the ease of reading
group-decimal-digits of numbers. The `group-digits` choice turns this behaviour on and off, with grouping
group-integer-digits for numbers of exactly four digits controlled by the `group-four-digits` choice. Note
group-four-digits that the later only applies if `group-digits` is turned on. The separator used between
group-separator groups of digits is stored by the `group-separator` option. This takes literal input and
 is used in math mode: for a text-mode full space use `\text{~}`.

```

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

Table 15: Output options for numbers.

Option name	Type	Default
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-decimal-digits	Switch	true
group-digits	Switch	true
group-four-digits	Switch	false
group-integer-digits	Switch	true
group-separator	Math	\,
negative-color	Literal	<i>none</i>
open-bracket	Literal	(
output-close-uncertainty	Literal)
output-complex-root	Math	\mathrm{i}
output-decimal-marker	Math	.
output-open-uncertainty	Literal	(
separate-uncertainty	Switch	false
tight-spacing	Switch	false
uncertainty-separator	Math	<i>none</i>

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

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

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

12345.67890
12345.678 90
12 345.67890
```

`output-complex-root` The decimal marker used in output is set using the `output-decimal-marker` option. This can differ from the input marker, as can the root of $\sqrt{-1}$, which is stored in the `output-complex-root` option. The later is always in math mode, but the standard setting uses `\mathrm` 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 + 2i
1 + 2i
1 + 2j
1 + 2j
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 - 0.9i
67 - i0.9
67 - 0.9i
```

`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 × 102
1 · 102
1 × 22
```

`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) \num{8.2(13)} \\
8.2 ± 1.3 \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.

Table 16: Output options for lists and ranges of numbers.

Option name	Type	Default
<code>list-final-separator</code>	Literal	<code>□and□</code>
<code>list-separator</code>	Literal	<code>,□</code>
<code>range-phrase</code>	Literal	<code>□to□</code>

```

\num{1+2i e10} \\
\num[bracket-numbers = false]{1+2i e10} \\
\sisetup{
  open-bracket = \{,
  close-bracket = \},
}
\num{1+2i e10}
(1 + 2i) × 1010
1 + 2i × 1010
{1 + 2i} × 1010

```

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

```

-15 673
-15 673
\num{-15673} \\
\num[negative-color = red]{-15673}

```

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

```

\num{1 \pm 2i e3} \\
\num[tight-spacing = true]{1 \pm 2i e3}
(1 ± 2i) × 103
(1±2i)×103

```

Lists and ranges of numbers have a small number of specialised options, which apply on to these more unusual input forms (Table 16).

`list-final-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. Any spaces needed should be included in the option settings: none are added within the code. These two items are printed in text mode.

Table 17: Multi-part number options.

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

```

\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}

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

```

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

```

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

```

5.6 Multi-part numbers

`siunitx` recognises the idea of products and quotients in numbers, both with and without units. These multi-part numbers have a number of options affecting how they are processed. The options are summarised in Table 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 × 2 × 3 \num{1 x 2 x 3} \\\
1 × 104 × 2(3) × 3/4 \num{1e4 x 2(3) x 3/4} \\\
4 × 5 × 6 \num[input-product=*]{4 * 5 * 6} \\\
1/(2 × 104) \num{ 1 / 2e4 } \\\
1 × 102/(3 × 104) \num{ 1e2 / 3e4 }

```

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

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

4.87 · 5.321 · 6.90545
1 div 2
```

`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 × 104)

$$\frac{1}{2 \times 10^4}$$

```

`fraction-function` The function used when `quotient-mode = fraction` is set is determined by the
`fraction-function` option. This should be set to a function which takes two argu-
ments, and presumably creates some type of fraction. Most alternatives to the standard
`\frac` function will involve loading additional packages: the demonstrations here need
`amsmath` and `xfrac`.³

```
\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}


$$\frac{1}{1} \frac{1}{2} \frac{1}{3} \frac{1}{4}$$

```

5.7 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 18).

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

```
\ang{2.67} \\
\num[number-angle-separator = \,]{2.67}

2.67°
2.67 °
```

Table 18: 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	Math	false
number-angle-separator	Math	<i>empty</i>

`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.

```
-1°              \ang{-1;;} \\
-2'              \ang{;-2;} \\
-3"              \ang{;;-3} \\
-1°              \ssetup{add-arc-degree-zero}
-1°              \ang{-1;;} \\
-0°2'           \ang{;-2;} \\
-0°3"           \ang{;;-3} \\
-1°0'           \ssetup{add-arc-minute-zero}
-0°2'           \ang{-1;;} \\
-0°0'3"         \ang{;-2;} \\
-1°0'0"         \ang{;;-3} \\
-0°2'0"         \ssetup{add-arc-second-zero}
-0°0'3"         \ang{-1;;} \\
                 \ang{;-2;} \\
                 \ang{;;-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}
```

³If `xfrac` is not available when typesetting this document, the demonstration of `\sfrac` will have the wrong appearance.

Table 19: Unit creation options.

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

45.697°
6°7'6.5"
45°697
6°7'6!5

5.8 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 19). As a result, these options all apply in the preamble only.

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

`space-before-unit` When ‘free standing’ unit macros are created, their behaviour can be adjusted by 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.

5.9 Loading additional units

`load-configurations` siunitx includes a set of configuration files which are intended to make life easier for the user. The files are loaded using the `load-configurations` option, which accepts a comma-separated list of file names.

The first configuration file provides a number of convenient abbreviations for the SI units (Table 20). 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).

Table 20: Abbreviated units
(`load-configurations=abbreviations`).

Unit	Abbreviation	Symbol
femtogram	<code>\fg</code>	fg
picogram	<code>\pg</code>	pg
nanogram	<code>\ng</code>	ng
microgram	<code>\ug</code>	μg
milligram	<code>\mg</code>	mg
gram	<code>\g</code>	g
kilogram	<code>\kg</code>	kg
atomic mass unit	<code>\amu</code>	u
picometre	<code>\pm</code>	pm
nanometre	<code>\nm</code>	nm
micrometre	<code>\um</code>	μm
millimetre	<code>\mm</code>	mm
centimetre	<code>\cm</code>	cm
decimetre	<code>\dm</code>	dm
metre	<code>\m</code>	m
kilometre	<code>\km</code>	km
attosecond	<code>\as</code>	as
femtosecond	<code>\fs</code>	fs
picosecond	<code>\ps</code>	ps

Continued on next page

Continued from previous page

Unit	Abbreviation	Symbol
nanosecond	<code>\ns</code>	ns
microsecond	<code>\us</code>	μ s
millisecond	<code>\ms</code>	ms
second	<code>\s</code>	s
femtomole	<code>\fmol</code>	fmol
picomole	<code>\pmol</code>	pmol
nanomole	<code>\nmol</code>	nmol
micromole	<code>\umol</code>	μ mol
millimole	<code>\mmol</code>	mmol
mole	<code>\mol</code>	mol
kilomole	<code>\kmol</code>	kmol
picoampere	<code>\pA</code>	pA
nanoampere	<code>\nA</code>	nA
microampere	<code>\uA</code>	μ A
milliampere	<code>\mA</code>	mA
ampere	<code>\A</code>	A
kiloampere	<code>\kA</code>	kA
microlitre	<code>\u1</code>	μ l
millilitre	<code>\m1</code>	ml
litre	<code>\1</code>	l
microliter	<code>\uL</code>	μ L
milliliter	<code>\mL</code>	mL
liter	<code>\L</code>	L
millihertz	<code>\mHz</code>	mHz
hertz	<code>\Hz</code>	Hz
kilohertz	<code>\kHz</code>	kHz
megahertz	<code>\MHz</code>	MHz
gigahertz	<code>\GHz</code>	GHz
terahertz	<code>\THz</code>	THz
newton	<code>\N</code>	N
millinewton	<code>\mN</code>	mN
kilonewton	<code>\kN</code>	kN
meganewton	<code>\MN</code>	MN

Continued on next page

Continued from previous page

Unit	Abbreviation	Symbol
pascal	\Pa	Pa
kilopascal	\kPa	kPa
megapascal	\MPa	MPa
gigapascal	\GPa	GPa
milliohm	\mohm	mΩ
kilohm	\kohm	kΩ
megohm	\Mohm	MΩ
millivolt	\mV	mV
volt	\V	V
kilovolt	\kV	kV
watt	\W	W
milliwatt	\mW	mW
kilowatt	\kW	kW
megawatt	\MW	MW
gigawatt	\GW	GW
joule	\J	J
kilojoule	\kJ	kJ
electronvolt	\eV	eV
millielectronvolt	\meV	meV
kiloelectronvolt	\keV	keV
megaelectronvolt	\MeV	MeV
gigaelectronvolt	\GeV	GeV
teraelectronvolt	\TeV	TeV
kilowatt hour	\kWh	kWh
kelvin	\K	K

\bit 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 (Table 21). The units \bit and \byte are also available.

```
\SI{100}{\mebi\byte} \\
\SI[prefixes-as-symbols=false]{30}{\kibi\bit}
100 MiB
30 × 210 bit
```

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 `load-configurations = version-1`. This is intended to allow easy transition to version 2: users should update their source to use the new interfaces and functions.

Table 21: Binary prefixes
(load-configurations=binary).

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

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

```

% Astronomy
\DeclareSIUnit\parse{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\c\light}
\DeclareSIUnit\nanobarn{\nano\barn}
\DeclareSIUnit\picobarn{\pico\barn}
\DeclareSIUnit\femtobarn{\femto\barn}

```

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

Table 22: Unit output options.

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

```

\DeclareSIUnit\attobarn{\atto\barn}
\DeclareSIUnit\zeptobarn{\zepto\barn}
\DeclareSIUnit\yottobarn{\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 5.15.

5.10 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 22).

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

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

```

\si{\farad\squared\lumen\candela} \\
\si[inter-unit-separator={}\cdot{}]{\farad\squared\lumen\candela}
F2 lm cd
F2 · lm · cd

```

per-mode The handling of `\per` is altered using the `per-mode` choice option. The standard setting is `reciprocal`, meaning that `\per` generates reciprocal powers for units. Setting the `per-symbol` 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}
```

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

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}
```

$\text{A mol}^{-1} \text{s}$
 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`. The output for `per-symbol` is always made in math mode, and so `\text` will be needed to print textual information.

```
\sisetup{per-mode = symbol}%
\si{\joule\per\mole\per\kelvin} \\
\si{\metre\per\second\squared} \\
\si[per-symbol = \text{\~div~}]{\joule\per\mole\per\kelvin} \\
\si[bracket-unit-denominator = false]{\joule\per\mole\per\kelvin}
```

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

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}
```

$\text{J}/\text{mol}/\text{K}$

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}%
\langle \si{\joule\per\mole\per\kelvin} \rangle
\lceil \si{\joule\per\mole\per\kelvin} \rceil
\si{\joule\per\mole\per\kelvin} \\
\langle
\displaystyle
\si{\joule\per\mole\per\kelvin}
\rangle
\lceil
\textstyle
\si{\joule\per\mole\per\kelvin}
\rceil
J/(molK)
\frac{J}{molK}
J/(molK)
\frac{J}{molK}
J/(molK)
```

`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}
Pa Gy-1 H
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.

```
\sisetup{unit-mode = text}%
\si{\metre\per\second\squared} \\
\si[power-font = unit]{\metre\per\second\squared}
ms-2
ms-2
```

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

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


```

\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}

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

```

`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.

```

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

ml mol-1 dA
10 kg2 ds
10-4 l mol-1 A
10 × 10-1 kg2 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 \SI{300}{\MHz} \\
300 MHz \SI[parse-units = false]{300}{\MHz}

```

5.11 Numbers with units

Some options apply to the combination of units and numbers, rather than to units or numbers alone (Table 23).

`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.

Table 23: Options for numbers with units.

Option name	Type	Default
allow-number-unit-breaks	Switch	false
list-units	Choice	repeat
multi-part-units	Choice	brackets
number-unit-separator	Math	\,
product-units	Choice	repeat
range-units	Choice	repeat

```

\begin{minipage}{2.55 cm}
  % Gives an underfull hbox
  Some filler text \SI{10}{\metre} \\
  10m
  Some filler text 10
  m
\end{minipage}

```

number-unit-separator The separator between the number and unit is set using the `number-unit-separator` option. This is always printed in math mode, and so anything which must be printed as text should be placed inside a `\text` macro.

```

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

2.67 F
2.67 F
2.67F

```

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 ± 0.4) kg
(12.3 ± 0.4) kg
12.3 kg ± 0.4 kg
12.3 ± 0.4 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}
```

$$1.234 \pm 0.005 \times 10^{-4}$$

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

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}
```

$$2 \text{ m} \times 3 \text{ m} \times 4 \text{ m}$$

$$(2 \times 3 \times 4) \text{ m}$$

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

$$2 \times 3 \times 4 \text{ m}^3$$

$$2 \text{ m} \times 3 \text{ m} \times 4 \text{ m}$$

$$2 \times 3 \times 4 \text{ m}$$

`list-units` `range-units` The `list-units` and `range-units` options determines how the `\SIlist` and `\SIrange` function displays 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 sets 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}
```

Table 24: Options for tabular material.

Option name	Type	Default
<code>table-align-exponent</code>	Switch	true
<code>table-align-text-post</code>	Switch	true
<code>table-align-uncertainty</code>	Switch	true
<code>table-alignment</code>	Choice	<i>none</i>
<code>table-auto-round</code>	Switch	false
<code>table-comparator</code>	Switch	false
<code>table-figures-decimal</code>	Integer	2
<code>table-figures-exponent</code>	Integer	0
<code>table-figures-integer</code>	Integer	3
<code>table-figures-uncertainty</code>	Integer	0
<code>table-format</code>	Special	<i>none</i>
<code>table-number-alignment</code>	Choice	center-decimal-marker
<code>table-parse-only</code>	Switch	false
<code>table-omit-exponent</code>	Switch	true
<code>table-space-text-pre</code>	Literal	<i>empty</i>
<code>table-space-text-post</code>	Literal	<i>empty</i>
<code>table-sign-exponent</code>	Switch	false
<code>table-sign-mantissa</code>	Switch	false
<code>table-text-alignment</code>	Choice	center
<code>table-unit-alignment</code>	Choice	center

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

5.12 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 24).

`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 25.

```
\begin{table}
  \centering
  \caption{Parsing without aligning in an \texttt{S} column.}
```

Table 25: 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

```

\label{tab:S:parse}
\begin{tabular}
{
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 26, 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}{

```

Table 26: 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

```

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} \\
\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 27.

`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

```

Table 27: 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

```

\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}
  & {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 28).

```

\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

```

Table 28: 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	$\gg 10^6$

```

\begin{tabular}{S}
S[table-comparator = true]
}
\toprule
{Values}
& {Values} \\
\midrule
2.3 & < 2.3e8 \\
34.23 & = 34.23 \\
56.78 & \geq 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, placing a warning in the L^AT_EX 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 29).

```
\begin{table}
  \caption{Using the \opt{table-format} option.}
  \label{tab:S:format}
  \centering
  \begin{tabular}{c}
    S
    S[table-format = 2.2]
    S[table-format = 2.2(1)]
    S[table-format = +2.2]
    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 Space for material before and after the S column can be reserved by giving model text
table-space-text-post for the options table-space-text-pre and ...-post. This is then used to provide the
necessary gap while maintaining alignment (Table 30).

```
\begin{table}
  \caption{Text before and after numbers.}
  \label{tab:S:ends}
  \centering
  \sisetup{
```

Table 29: 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

Table 30: Text before and after numbers.

Values
2.3456
34.2345 ^a
56.7835
now 90.473

```

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}
\end{table}

```

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 31). 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 32).

```

\begin{table}
\centering
\caption{The \opt{table-align-exponent} option}

```

Table 31: The table-align-exponent option

Header	Header
1.2×10^3	1.2×10^3
1.234×10^{56}	1.234×10^{56}

Table 32: The table-align-uncertainty option

Header	Header
1.2 ± 0.1	1.2 ± 0.3
1.234 ± 0.005	1.234 ± 0.005

```

\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} \\
    \midrule
      1.2(1) & 1.2(3) \\
      1.234(5) & 1.234(5) \\
    \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 33: 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

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 33); 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
    ]
  }
  \toprule
    {Header} & {Header / \num{e3}} \\
  \midrule
    1.2e3 & 1.2e3 \\
    3e2 & 3e2 \\
    1.0e4 & 1.0e4 \\
  \bottomrule
  \end{tabular}
\end{table}
```

`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-post` option (Table 34).

```
\begin{table}
  \caption{Closing notes up to text.}
  \label{tab:S:notes}
  \newcommand\NoteMark[1]{%
    \textsuperscript{\emph{#1}}%
  }
  \centering
  \sisetup{
    table-number-alignment = center,
  }
```

Table 34: Closing notes up to text.

Values	Values
2.3456	2.3456
34.234 ^a	34.234 ^a
56.78 ^b	56.78 ^b
90.4 ^c	90.4 ^c

```

    table-figures-integer = 2,
    table-figures-decimal = 4,
}
\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} \\
\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 35.

```

\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

```

Table 35: The table-auto-round option.

Header	Header
1.2	1.200
1.2345	1.235

```

{Header} & {Header} \\
\midrule
1.2 & 1.2 \\
1.2345 & 1.2345 \\
\bottomrule
\end{tabular}
\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 36).

```

\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}{c}
S
S[table-number-alignment = center]
S[table-number-alignment = right]
S[table-number-alignment = left]
}
\toprule
\multicolumn{1}{c}{Some values}
& \multicolumn{1}{c}{Some values}
& \multicolumn{1}{c}{Some values}
& \multicolumn{1}{c}{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}

```

Table 36: 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
$\sqrt{2}$	$\sqrt{2}$	$\sqrt{2}$	$\sqrt{2}$

```

\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 37).

```

\begin{table}
\caption{Aligning text in \texttt{S} columns.}
\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 37: 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

Table 38: Alignment options in s columns.

Left-aligned	Centred text	Right-aligned
m s^{-1}	m s^{-1}	m s^{-1}
kg	kg	kg

`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}
  \begin{tabular}
    {
      s[table-unit-alignment = right]
      s
      s[table-unit-alignment = left]
    }
  \toprule
    {Left-aligned} &
    {Centred text} &
    {Right-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 39: Symbol options.

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

5.13 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 39. 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` 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
`\SIUnitSymbolArcminute`
`\SIUnitSymbolArcsecond`
`\SIUnitSymbolCelsius`
`\SIUnitSymbolDegree`
`\SIUnitSymbolMicro`
`\SIUnitSymbolOhm`

`\SIUnitSymbolMicro`. Notice that these names capitalise the unit name (to make reading the macro name easier!).⁶

5.14 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.

```
1.234m          \SI{1.234}{\metre}\
6,789m          \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.

5.15 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 L^AT_EX document headers.

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

⁶The function `\SIUnitSymbolAngstrom` uses the name without accents.

`\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} },
  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 and German as options, and also loads the `babel` package:

1 m, 2 m and 3 m	<code>% In English by default</code>
1 °C to 10 °C	<code>\SIlist{1;2;3}{\metre} \\ \SIRange{1}{10}{\degreeCelsius} \\ \selectlanguage{french}%</code>
1 m, 2 m et 3 m	<code>\SIlist{1;2;3}{\metre} \\ \SIRange{1}{10}{\degreeCelsius} \\ \selectlanguage{german}%</code>
1 °C à 10 °C	<code>\SIlist{1;2;3}{\metre} \\ \SIRange{1}{10}{\degreeCelsius} \\ \selectlanguage{german}%</code>
1 m, 2 m und 3 m	<code>\SIlist{1;2;3}{\metre} \\ \SIRange{1}{10}{\degreeCelsius} \\ \selectlanguage{german}%</code>
1 °C bis 10 °C	<code>\SIlist{1;2;3}{\metre} \\ \SIRange{1}{10}{\degreeCelsius} \\ \selectlanguage{german}%</code>

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. $\LaTeX_{2\epsilon}$ robust commands are also detected and are not expanded (Table 40). 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 $\epsilon\text{-}\TeX$ 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 $\LaTeX_{2\epsilon}$ robust commands can lead to unexpected results.)

Table 40: Values as macros in S columns.

Some Values		
12	348.8	12 34
12	348.8	1234
1234	8.8	1234
1234	8.8	1234
1234	8.8	1234

```

\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
    {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 $\epsilon\text{-}\TeX$ 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 41. 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}\DTLnewbentry{data}{value}{66.7012}
\DTLnewrow{data}\DTLnewbentry{data}{value}{66.0212}
\DTLnewrow{data}\DTLnewbentry{data}{value}{64.9026}
\begin{table}

```

Table 41: Calculated values.

Value	Doubled
66.7012	133.4024
66.0212	132.0424
64.9026	129.8052

```

\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
{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 41, 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 42).

```

\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.}

```

Table 42: Displaying a datatool table.

	value
	6.7012
	66.0212
	64.902

```

\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 $\mu\text{m s}^{-1}$ in headings

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

```

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

```

7.5 Symbols and X_YTeX

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

X_YTeX 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:

```
\ssetup{
  math-micro = \text{\mu},
  text-micro  = \mu
}
```

7.6 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:

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

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

7.7 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.

7.8 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:

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

7.9 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
...
```

7.10 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.

7.11 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                \si{\kWh} \
kWh m-1          \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
```

7.12 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 spacing will be wrong. 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 instead (Table 43).

```
\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 43: Correcting spacing in last S column

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

7.13 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 44). 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}
\end{table}

```

Table 44: Tables where numbers have different units

	One	Two	Heading
$a/\text{\AA}$	1.234(2)	5.678(4)	1.234 m
$\beta/^\circ$	90.34(4)	104.45(5)	0.835 cd
μ/mm^{-1}	0.532	0.894	4.23 J mol ⁻¹

Table 45: Header row in a table

123.456
23.45
123.4
3.456

```
\hfil
\end{table}
```

7.14 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 45, 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}
```

7.15 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 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[load-configurations = version-1]{siunitx}
```

Many of the options from version 1 map to similar ones in version 2 (Table 46). 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 46: Mapping of version 1 options to version 2.

Version 1	See in version 2
<code>addsign</code>	<code>explicit-sign</code>
<code>allowlitunits</code>	<code>free-standing-units</code>
<code>allowoptarg</code>	<code>unit-optional-argument</code>
<code>allowzeroexp</code>	<code>retain-zero-exponent</code>
<code>anglesep</code>	<code>arc-separator</code>
<code>astroang</code>	<code>angle-symbol-over-decimal</code>
<code>closeerr</code>	<code>close-bracket</code>
<code>closefrac</code>	<code>close-bracket</code>
<code>closerange</code>	<code>close-bracket</code>

Continued on next page

Continued from previous page

Version 1	See in version 2
colour	color
colorall	color
colourall	color
colorunits	unit-color
colorneg	negative-color
colourneg	negative-color
colourunits	unit-color
colorvalues	value-color
colourvalues	value-color
decimalsymbol	output-decimal-marker
detectdisplay	detect-display-math
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

Continued on next page

Continued from previous page

Version 1	See in version 2
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
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

Continued on next page

Continued from previous page

Version 1	See in version 2
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
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
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

Continued on next page

Continued from previous page

Version 1	See in version 2
textrm	text-rm
textsecond	text-arcsecond
textsf	text-sf
texttt	text-tt
tightpm	tight-spacing
topphrase	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
unitmathsrn	unit-math-rm
unitmathssf	unit-math-sf
unitmathstt	unit-math-tt
unitmathttt	unit-math-tt
unitmode	unit-mode
unitsep	inter-unit-separator
unitspace	inter-unit-separator
valuecolor	value-color
valuecolour	value-color
valuemathrm	value-math-rm
valuemathsf	value-math-sf
valuemathsrn	value-math-rm
valuemathssf	value-math-sf
valuemathstt	value-math-tt
valuemathttt	value-math-tt
valuemode	value-mode
valuesep	number-unit-separator
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 from earlier packages), and so the `emulate` option no longer applies.

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 appropriate 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 m°C.

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

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 $1 \times 10^{-6} \text{ kg} = 1 \text{ mg}$.

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 in 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 21) 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 two parts should not be separated. With the exception of the symbols for plane angles ($^\circ$, $'$ and $''$), it is usual to have a space between the unit and the number. This should therefore be a thin non-breaking space between the two.

```
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 serif font. Italic, bold and sans serif are all used mathematically to convey other meanings. The `siunitx` package defaults again follow this convention: any local settings are ignored, and uses the current upright serif 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-separator = { } \cdot { } }  

\(< \si{\metre\second} = \text{metre second} \) \\  

\(< \si{\milli\second} = \text{millisecond} \) \)
```

ms = metre second
ms = millisecond
m · s = metre second
ms = millisecond

There are some circumstances under which it is permissible to omit any spaces. The classic example is kWh, where ‘kWh’ 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.

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.

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

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

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

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

Products and errors should show what unit applies to each number given. Thus $(2 \times 3) \text{ m}$ is an ordered set of lengths of a geometric area, whereas $2 \times 3 \text{ m}$ 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: $2 \times 3 \text{ m}^2$. Although this does display the correct overall units, it is potentially-confusing and is not encouraged.

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

Care must be taken when writing ranges of numbers. For purely numerical values, it is common to use an en-dash to show a range, for example ‘see pages 1–5’. On the other hand, physical quantities could be misinterpreted 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 47 and Fig. 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}
```

```
\begin{figure}
  \centering
  \begin{tikzpicture}
    \begin{axis}[
      xlabel = \(\ t/\si{\second} \),
```

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

Table 47: An example of table labelling.

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

```

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 48. 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
]

```

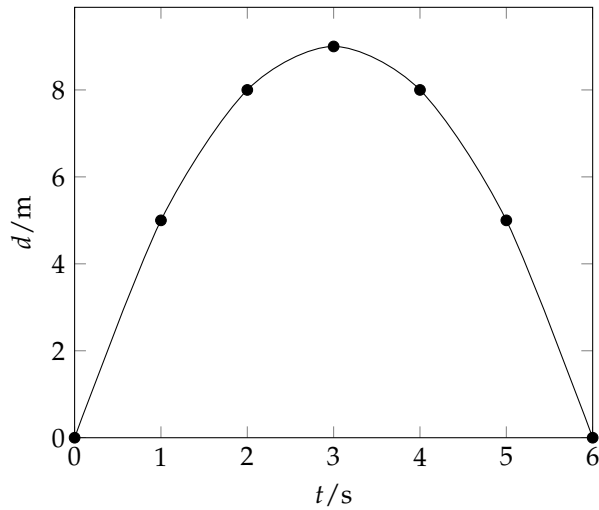


Figure 1: An example of graph labelling.

```

@{\,\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

Table 48: 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

‘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 [BitBucket](#), 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 is will also definitely get attention: joseph.wright@morningstar2.co.uk. I also keep an eye on various groups, for example [comp.text.tex](#) and [The L^AT_EX Community](#). Sometimes I miss posts to these places, so it’s useful if you also e-mail me pointing to the appropriate thread.

11 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 `Slunits` 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/si/>.
- [2] *International System of Units from NIST*, <http://physics.nist.gov/cuu/Units/index.html>.
- [3] *SI base units*, http://www.bipm.org/en/si/si_brochure/chapter2/2-1/.
- [4] *Units with special names and symbols; units that incorporate special names and symbols*, http://www.bipm.org/en/si/si_brochure/chapter2/2-2/2-2-2.html.
- [5] *SI Prefixes*, http://www.bipm.org/en/si/si_brochure/chapter3/prefixes.html.
- [6] *Non-SI units accepted for use with the International System of Units*, http://www.bipm.org/en/si/si_brochure/chapter4/table6.html.
- [7] *Non-SI units whose values in SI units must be obtained experimentally*, http://www.bipm.org/en/si/si_brochure/chapter4/table7.html.
- [8] *Other non-SI units*, http://www.bipm.org/en/si/si_brochure/chapter4/table8.html.
- [9] *Formatting the value of a quantity*, http://www.bipm.org/en/si/si_brochure/chapter5/5-3-2.html#5-3-3.

Change History

<p>vo.6 General: First public testing release (as si) 1</p> <p>v1.0 General: First official release 1</p> <p>v1.1 General: Package extended to a greater range of unit types 1</p> <p>v1.2 General: Correct handling for ranges of numbers added 1</p> <p>v1.3 General: Better definition for micro symbol 1</p> <p>v1.4 General: Detect entire document in non-serif font 1</p>	<p>v2.0 General: Complete re-write of package to add many new features 1</p> <p>Introduced \numlist and \SIlist functions 1</p> <p>v2.0a General: Detect use of version 1 options and automatically load appropriate configuration file 1</p> <p>Fix various errors in version-1 configuration file 1</p> <p>Include high energy physics units in discussion of old configurations and in version-1 configuration file 1</p> <p>Make \SendSettingsToPgf available in document body 1</p>
--	--

v2.ob	General: Further improvements to version-1 configuration file	1	v2.00	General: Extend <code>detect-italic</code> option to other shapes, renaming as <code>detect-shape</code> as a result	1
v2.oc	General: Mixed literal and macro units now print more reliably	1	v2.op	General: Actually get change from v2.00 working	1
v2.od	General: Document special case situations for last cell in table row	1	v2.oq	General: Deal with bad definition of <code>\color</code> by <code>textpos</code> package	1
	Fix error in <code>table-format</code> option concerning exponent signs	1		Errors with free-standing unit code fixed	1
v2.oe	General: Correct behaviour of <code>\pm</code> in numbers when abbreviations configuration is loaded: problem introduced in v2.0c	1	v2.or	General: Error in definition for old <code>decimalsymbol</code> option corrected . .	1
v2.of	General: Fix issue with spacing of multiplication sign in text mode	1	v2.os	General: Correct errors in rounding code when precision requested is zero decimal places	1
v2.og	General: Fix issue with complex numbers in quotients	1		Document how to do mixed bold and normal numbers in tables	1
	Improve handling of complex root token	1	v2.ot	General: Replace <code>\exp_after:wN</code> in code for <code>\per</code> with <code>\exp_after:wN</code>	1
	Introduce localisation for text values	1	v2.ou	General: Fix second possible issue with <code>textpos</code> package and <code>\color</code>	1
	Repair broken <code>bracket-numbers</code> option	1		Prevent infinite loop if <code>\SI</code> function is used with an empty number . . .	1
v2.oh	General: Actually get localisation into the code	1	v2.ov	General: Internal changes reflecting <code>expl3</code> updates	1
v2.oi	General: Correct behaviour of <code>\of</code> function so it actually works (bug introduced in v2.0d)	1	v2.ow	General: Deal with internal function used by REVTeX in tables	1
v2.oj	General: Correct <code>\hartree</code> unit appearance	1	v2.ox	General: Fix bug when detecting single prefixes and converting prefixes to numbers	1
	Ensure symbols specified in <code>input-symbols</code> are always printed in math mode	1	v2.oy	General: Error with <code>tight-spacing</code> option and exponents corrected	1
v2.ok	General: Fix for <code>babel</code> French settings with <code>\fg</code> in tabular material	1	v2.1	General: New <code>copy-complex-root</code> option for moving input complex root to output	1
v2.ol	General: Further adjustments to <code>babel</code> support	1		New <code>input-comparators</code> option for numbers greater than, less than and so on	1
v2.om	General: Re-introduce <code>locale</code> option . .	1			
v2.on	General: More abbreviated units	1			

New <code>power-font</code> option for controlling whether superscript powers are treated as numbers or units . . .	1	v2.1e	General: Fix issues with text mode symbols and <code>fontspec</code> package	1
New <code>round-integer-to-decimal</code> option to convert integers to decimals on rounding	1		Further corrections when applying unit options when <code>free-standing-units</code> is active	1
New <code>round-minimum</code> option to set a floor for rounding numbers downward	1	v2.1f	General: Typo in definition for <code>unit-optional-argument</code> implementation corrected	1
New <code>scientific-notation</code> option for to use exponent form for numbers in all cases	1	v2.1g	General: Checks on the versions of <code>expl3</code> and <code>xparse</code> installed	1
New <code>table-align-exponent</code> and <code>table-align-uncertainty</code> options for additional choices of table formatting	1	v2.1h	General: Detect AMS display-like environments	1
New <code>table-comparator</code> option for reserving space for comparators in tables	1	v2.1i	General: Improved logic for per-mode setting <code>symbol-or-fraction</code>	1
New <code>table-omit-exponent</code> option for simplifying tables	1	v2.1j	General: Allow for loading of <code>inputenc</code> package with no options	1
v2.1a		v2.1k	General: Bug fix when printing superscript minus signs and using <code>fontspec</code> package	1
General: Ensure that output of list separators is in text mode	1		New option <code>detect-inline-family</code>	1
Print prefixes correctly in text mode when converting to numerical value	1		Remove combined choice for option <code>detect-inline-weight</code>	1
v2.1b		v2.1l	General: Error in font family detection introduced in v2.1k corrected	1
General: Bug in hyphen printing when detecting mode sorted	1	v2.1m	General: Avoid expansion of erroneous literal units when these are forbidden	1
Bug in printing code for complex part with no number fixed	1		Ensure some output occurs in all cases when <code>round-precision</code> is set to 0 and <code>round-mode</code> is set to <code>places</code>	1
v2.1c		v2.1n	General: Consistent behaviour for <code>round-integer-to-decimal</code> when <code>round-precision</code> is 0	1
General: After reviewing internals, <code>\numlist</code> , <code>\numrange</code> , <code>\SIlist</code> and <code>\SIrange</code> are documented as requiring text mode due to issues with spacing and line breaks	1		Set output to 0 when <code>round-mode</code> is <code>figures</code> and <code>round-precision</code> is 0	20
Auto-detect math mode in tables and correct output accordingly	1	v2.1o	General: Account for negative exponents when using <code>fixed-exponent</code> system	1
Discourage line break between number and unit even when it is permitted	1			
New text choice for <code>qualifier-mode</code> option	1			
v2.1d				
General: Apply unit options when <code>free-standing-units</code> is active	1			
Error with definition of version 1 option <code>xspace</code> corrected	1			

Fix incorrect font choice when arev package is loaded	<u>1</u>	some rows contain comparators fixed	<u>1</u>
v2.1p		Poor position of comparators in numbers fixed	<u>1</u>
General: Bad table alignment when			

Index

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

A		\centi	<u>11</u>
\A	<u>35</u>	\clight	<u>10</u>
add-arc-degree-zero (option)	<u>32</u>	close-bracket (option)	<u>28</u>
add-arc-minute-zero (option)	<u>32</u>	\cm	<u>34</u>
add-arc-second-zero (option)	<u>32</u>	color (option)	<u>20</u>
add-decimal-zero (option)	<u>24</u>	complex-root-position (option)	<u>27</u>
add-integer-zero (option)	<u>24</u>	copy-complex-root (option)	<u>27</u>
allow-number-unit-breaks (option)	<u>41</u>	copy-decimal-marker (option)	<u>27</u>
\ampere	<u>9</u>	\coulomb	<u>9</u>
\amu	<u>34</u>	\cubed	<u>10</u>
\ang	<u>7</u>	\cubic	<u>10</u>
angle-symbol-over-decimal (option)	<u>32</u>	D	
\angstrom	<u>11</u>	\dalton	<u>10</u>
arc-separator (option)	<u>31</u>	\day	<u>10</u>
\arcminute	<u>10</u>	\deca	<u>11</u>
\arcsecond	<u>10</u>	\deci	<u>11</u>
\as	<u>34</u>	\decibel	<u>11</u>
\astronomicalunit	<u>10</u>	\DeclareBinaryPrefix	<u>13</u>
\atomicmassunit	<u>10</u>	\DeclareSIPostPower	<u>13</u>
\atto	<u>11</u>	\DeclareSIPrefix	<u>13</u>
B		\DeclareSIPrePower	<u>13</u>
\bar	<u>11</u>	\DeclareSIQualifier	<u>14</u>
\barn	<u>11</u>	\DeclareSIUnit	<u>13</u>
\becquerel	<u>9</u>	\DeclareSIUnitWithOptions	<u>13</u>
\bel	<u>11</u>	\degree	<u>10</u>
\bit	<u>36</u>	\degreeCelsius	<u>9</u>
\bohr	<u>10</u>	\deka	<u>10</u>
bracket-numbers (option)	<u>28</u>	detect-all (option)	<u>17</u>
bracket-unit-denominator (option)	<u>38</u>	detect-display-math (option)	<u>18</u>
\byte	<u>36</u>	detect-family (option)	<u>17</u>
C		detect-inline-family (option)	<u>18</u>
\candela	<u>9</u>	detect-inline-weight (option)	<u>18</u>
\celsius	<u>9</u>	detect-mode (option)	<u>17</u>
		detect-none (option)	<u>17</u>

detect-shape (option)	17		
detect-weight (option)	17		
\dm	34		
E			
\electronmass	10		
\electronvolt	10		
\elementarycharge	10		
\eV	36		
\exa	11		
\exbi	37		
explicit-sign (option)	24		
exponent-base (option)	27		
exponent-product (option)	27		
F			
\farad	9		
\femto	11		
\fg	34		
fixed-exponent (option)	24		
\fmol	34		
forbid-literal-units (option)	38		
fraction-function (option)	31		
free-standing-units (option)	32		
\fs	34		
G			
\g	34		
\GeV	36		
\GHz	35		
\gibi	37		
\giga	11		
\GPa	35		
\gray	9		
group-decimal-digits (option)	25		
group-digits (option)	25		
group-four-digits (option)	25		
group-integer-digits (option)	25		
group-separator (option)	25		
\GW	36		
H			
\hartree	10		
\hectare	10		
\hecto	11		
\henry	9		
\hertz	9		
\hour	10		
\Hz	35		
I			
input-close-uncertainty (option)	21		
input-comparators (option)	21		
input-complex-roots (option)	21		
input-decimal-markers (option)	20		
input-digits (option)	20		
input-exponent-markers (option)	20		
input-ignore (option)	21		
input-open-uncertainty (option)	21		
input-product (option)	30		
input-protect-tokens (option)	22		
input-quotient (option)	30		
input-signs (option)	20		
input-symbols (option)	21		
inter-unit-separator (option)	38		
J			
\J	36		
\joule	9		
K			
\K	36		
\kA	35		
\katal	9		
\kelvin	9		
\keV	36		
\kg	34		
\kHz	35		
\kibi	37		
\kilo	11		
\kilogram	9		
\kJ	36		
\km	34		
\kmol	35		
\kN	35		
\knot	11		
\kohm	35		
\kPa	35		
\kV	36		
\kW	36		
\kWh	36		
L			
\L	35		
\l	35		
list-final-separator (option)	29		
list-separator (option)	29		
list-units (option)	43		
\liter	10		
\litre	10		

load-configurations (option)	33	\ng	34
locale (option)	58	\nm	34
\lumen	9	\nmol	35
\lux	9	\ns	34
M		\num	6
\m	34	number-angle-separator (option)	31
\mA	35	number-unit-separator (option)	42
math-rm (option)	19	\numlist	6
math-sf (option)	19	\numrange	7
math-tt (option)	19	O	
\mebi	37	\of	12
\mega	11	\ohm	9
\meter	9	open-bracket (option)	28
\metre	9	options:	
\MeV	36	add-arc-degree-zero	32
\meV	36	add-arc-minute-zero	32
\mg	34	add-arc-second-zero	32
\MHz	35	add-decimal-zero	24
\mHz	35	add-integer-zero	24
\micro	11	allow-number-unit-breaks	41
\milli	11	angle-symbol-over-decimal	32
\minute	10	arc-separator	31
\mL	35	bracket-numbers	28
\mL	35	bracket-unit-denominator	38
\mm	34	close-bracket	28
\mmHg	11	color	20
\mmol	35	complex-root-position	27
\MN	35	copy-complex-root	27
\mN	35	copy-decimal-marker	27
mode (option)	19	detect-all	17
\Mohm	36	detect-display-math	18
\mohm	35	detect-family	17
\mol	35	detect-inline-family	18
\mole	9	detect-inline-weight	18
\MPa	35	detect-mode	17
\ms	34	detect-none	17
multi-part-units (option)	42	detect-shape	17
\mV	36	detect-weight	17
\MW	36	explicit-sign	24
\mW	36	exponent-base	27
N		exponent-product	27
\N	35	fixed-exponent	24
\nA	35	forbid-literal-units	38
\nano	11	fraction-function	31
\nauticalmile	11	free-standing-units	32
negative-color (option)	29	group-decimal-digits	25
\neper	11	group-digits	25
\newton	9	group-four-digits	25
		group-integer-digits	25

<code>\umol</code>	35	<code>\watt</code>	9
<code>uncertainty-separator</code> (option)	28	<code>\weber</code>	9
<code>unit-optional-argument</code> (option)	32		
<code>\us</code>	34	Y	
<code>use-xspace</code> (option)	32	<code>\yobi</code>	37
V		<code>\yocto</code>	11
<code>\V</code>	36	<code>\yotta</code>	11
<code>\volt</code>	9	Z	
W		<code>\zebi</code>	37
<code>\W</code>	36	<code>\zepto</code>	11
		<code>\zetta</code>	11