

siunitx — A comprehensive (SI) units package*

Joseph Wright[†]

Released 2010/04/25

Abstract

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

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

Contents

1	Introduction	3
2	Installation	3
3	siunitx for the impatient	4
4	Using the siunitx package	5
4.1	Loading the package	5
4.2	Numbers	5
4.3	Units	6
4.4	The unit macros	7
4.5	Creating new macros	11
4.6	Tabular material	12

*This file describes v2.obeta (revision 435), last revised 2010/04/25.

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

5	The key–value control system	15
5.1	Detecting fonts	16
5.2	Output font families	17
5.3	Parsing numbers	19
5.4	Post-processing numbers	20
5.5	Printing numbers	22
5.6	Multi-part numbers	25
5.7	Angles	27
5.8	Creating units	28
5.9	Loading additional units	29
5.10	Using units	31
5.11	Numbers with units	35
5.12	Tabular material	37
5.13	Symbols	46
5.14	Other options	47
6	Hints for using siunitx	48
6.1	Ensuring text or math output	48
6.2	Numbers greater than, less than and so forth	48
6.3	Expanding content in tables	48
6.4	Using siunitx with datatool	50
6.5	Using units such as $\mu\text{m s}^{-1}$ in headings	51
6.6	Symbols and XeTeX	51
6.7	Scaled document fonts with XeTeX	52
6.8	Maximising performance	52
6.9	Transferring settings to pgf	52
7	Correct application of (SI) units	53
7.1	Units	53
7.2	Mathematical meaning	54
7.3	Graphs and tables	56
8	Thanks	58

Change History 59

Index 60

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

2 Installation

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

The package requires LaTeX₃ 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 MiKTeX 2.8 and TeX Live 2009 (updating the relevant packages online may be necessary). LaTeX₃, and so siunitx, requires the ϵ -TeX extensions: these are available on all modern TeX systems.

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

- `amsmath`
- `booktabs`
- `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]{value}[pre-unit]{unit}`
- `\numrange[options]{number1}{number2}`
- `\SIrange[options]{number1}{number2}{unit}`
- `\ang[options]{angle}`
- `\sisetup{options}`

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

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

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

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

```

\si{kg.m.s^{-1}}          \\  

\si{\kilogram\metre\per\second} \\  

\si[per-mode=symbol]  

  {\kilogram\metre\per\second} \\  

\si[per-mode=symbol]  

  {\kilogram\metre\per\ampere\per\second}

```

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

Simple ranges of numbers can be handled.

```

10 to 20                    \numrange{10}{20} \\  

0.13 mm to 0.67 mm        \SIRange{0.13}{0.67}{\milli\metre}

```

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 LaTeX2e way.

```
\usepackage{siunitx}
```

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

```
\usepackage[load-configuration = 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 marker.

123	<code>\num{123}</code>	<code>\\</code>
1234	<code>\num{1234}</code>	<code>\\</code>
12345	<code>\num{12345}</code>	<code>\\</code>
0.123	<code>\num{0.123}</code>	<code>\\</code>
0.1234	<code>\num{0,1234}</code>	<code>\\</code>
0.12345	<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.

`\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.

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

`\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.

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

```
\SI[mode=text]{1.23}{J.mol-1.K-1} \\
\SI{.23e7}{\candela} \\
\SI[per-mode=symbol]{1.99}[\$]{\per\kilogram} \\
\SI[per-mode=fraction]{1,345}{\coulomb\per\mole}
1.23 J mol-1 K-1
0.23 × 107 cd
$1.99/kg
1.345  $\frac{C}{mol}$ 
```

It is possible to set up the unit macros to be available outside of the `\SI` and `\si` functions. This is not the standard behavior 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.

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

Ranges of numbers with units can be handled using the `\SIrange` function. The behavior of this function is similar to `\numrange`, but with the addition of a unit to each number.

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

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

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

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

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

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

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

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].

<code>\deka</code>	In addition to the units themselves, <code>siunitx</code> provides pre-defined macros for all of the SI prefixes (Table 6, [5]). The spelling “ <code>\deka</code> ” is provided for US users as an alternative to <code>\deca</code> .
<code>\square</code> <code>\squared</code> <code>\cubic</code> <code>\cubed</code>	A small number of pre-defined powers are provided as macros. <code>\square</code> and <code>\cubic</code> are intended for use before units, with <code>\squared</code> and <code>\cubed</code> going after the unit.
	$\begin{array}{l} \text{Bq}^2 \\ \text{J}^2 \text{lm}^{-1} \\ \text{lx}^3 \text{VT}^3 \end{array} \qquad \begin{array}{l} \code{\si{\square\becquerel}} \code{\backslash} \\ \code{\si{\joule\squared\per\lumen}} \code{\backslash} \\ \code{\si{\cubic\lux\volt\tesla\cubed}} \end{array}$

Table 5: Other non-SI units.

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

Table 6: SI prefixes.

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

`\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:

```
H5                                \si{\henry\tothe{5}} \\  
rad4.5                             \si{\raiseto{4.5}\radian}
```

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

```
J mol-1 K-1                          \si{\joule\per\mole\per\kelvin} \\  
J mol-1 K                              \si{\joule\per\mole\kelvin} \\  
H-5                                    \si{\per\henry\tothe{5}} \\  
Bq-2                                   \si{\per\square\becquerel}
```

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

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

```
k                                \si{\kilo} \\  
μ                                \si{\micro}
```

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

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

4.5 Creating new macros

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

```
\DeclareSIUnit{<unit>}{<symbol>}  
\DeclareSIUnitWithOptions{<unit>}{<symbol>}{<options>}  
\DeclareSIUnitWithOptions New units are produced using the \DeclareSIUnit macro. <symbol> can contain literal
```

values, 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.

```
\DeclareSIPrefix{<prefix>}{<symbol>}{<powers-ten>}
\DeclareBinaryPrefix{<prefix>}{<symbol>}{<powers-two>}
\DeclareSIPrefix
\DeclareBinaryPrefix
```

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

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

```
\DeclareSIPostPower{<power>}{<num>}
\DeclareSIPrePower{<power>}{<num>}
\DeclareSIPostPower
\DeclareSIPrePower
```

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

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

with the functions then used in the document as:

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

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

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

and then in the body the document could read:

```
\SI{1.234}{\gram\polymer\per\mole\catalyst\per\hour}
```

$1.234 \text{ g}_{\text{pol}} \text{ mol}_{\text{cat}}^{-1} \text{ h}^{-1}$

4.6 Tabular material

Centring 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 center of the cell and to align the material appropriately (Table 7).

```
\begin{table}
  \caption{Standard behavior 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}
```

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 color applied by `siunitx`.

Table 7: Standard behavior 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

```

\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 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}

```

Table 9: Units in tables.

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

Table 10: The s column processes everything.

Unit	Unit
m^3	m^3
kg	kg

```

\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 coloring of the cell contents.

```

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

```

5 The key–value control system

`\sisetup` The behavior 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.

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.

`detect-weight` The options `detect-weight` and `detect-italic` set detection of the prevailing font weight and italic states, respectively. The italic state is only checked if the surrounding material is not in math mode (as math text is always italic). Detecting the current

`detect-family` family (roman, sans serif or monospaced) is controlled by the `detect-family` setting,

`detect-italic` while the current mode (text or math) is detected using the `detect-mode` switch.

`detect-mode`

`detect-inline-weight` Font weight detection is influenced by the value of `detect-inline-weight`, which takes values `text`, `math` and `combined`. The package can detect the local value of font weight for either the surrounding text, or the surrounding inline (\dots) math. The `combined` option checks both and uses the prevailing weight if either test is true.

Table 11: Font detection options.

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

```

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

```

`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.

```

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

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

More text

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


```

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>

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`, `math` and `text`. When using `math` mode, `text` is printed using a `math` font whereas in `text` mode a `text` font is used. The extent to which this is visually obvious depends on the fonts in use in the document. This manual uses old style (lower-case) figures in `text` mode to highlight the differences. This option has no effect if the `detect-mode` switch is true.
- `math-rm` If font family detection is inactive, `siunitx` uses the font family stored in either `math-rm` or `text-rm` for output. The choice of `math` or `text` depends on the `mode` setting.
- `text-rm`
- `math-sf` If font family detection is active, `siunitx` may be using a sans serif or monospaced font for output. In `math` mode, these are stored in `math-sf` and `math-tt`, and for `text` mode in
- `math-tt` `text-sf` and `text-tt`. Notice that the detected and output font families can differ.
- `text-sf`
- `text-tt`

```

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

```

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

Table 13: Options for number parsing.

Option name	Type	Default
input-close-uncertainty	Literal)
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	\mp\pi\pm
input-signs	Literal	+-\pm\mp
input-symbols	Literal	\pi
parse-numbers	Switch	true

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

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

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

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

4 Å
4 Å
4 Å
```

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.

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`, `input-signs` and `input-exponent-markers`.

`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 `exponent markers` option.

`input-symbols` As well as “normal” digits, the package will interpret symbolic “numbers” (such as π) correctly if they are included in the `input-symbols` list. 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-open-uncertainty` In some fields, it is common to give the uncertainty in a value in brackets after the main part of the number, for example “1.234(5)”. The opening and closing symbols used for this type of input are set as `input-open-uncertainty` and `input-close-uncertainty`.

`input-complex-roots` When using complex numbers in input, the complex root ($\sqrt{-1}$) 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 LaTeX2e. 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` 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-mode	Choice	off
round-precision	Integer	2

```

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

```

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} \\
\sisetup{
  add-decimal-zero = false,
  add-integer-zero = false,
}%
123.0      \num{123.} \\
456        \num{456} \\
.789       \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. As the same time, it may be useful to force all numbers to have a sign. This behavior 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 × 104
104
444
444 × 100
```

5.5 Printing numbers

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

`group-digits` Grouping digits into blocks of three is a common method to increase the ease of reading of numbers. The `group-digits` choice turns this behavior on and off, with grouping for numbers of exactly four digits controlled by the `group-four-digits` choice. Note that the later only applies if `group-digits` is turned on. The separator used between groups of digits is stored by the `group-separator` option. This takes literal input and is used in 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
close-bracket	Literal)
complex-root-position	Choice	after-number
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	(
range-phrase	Literal	to
separate-uncertainty	Switch	false
tight-spacing	Switch	false
uncertainty-separator	Math	<i>none</i>
use-brackets	Switch	true

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.
`output-decimal-marker` This can differ from the input marker, as can the root of $\sqrt{-1}$, which is stored in the
`copy-decimal-marker` `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 decimal marker from the input can be used in the output by setting the `copy-decimal-marker` option.

```
\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-decimal-marker]{555,555}

1.23
1,23
1 + 2i
1 + 2i
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` When exponents are present in the input, the `exponent-base` and `exponent-product`
`exponent-product` options set the obvious parts of the output. Notice that the base is in the current mode,
but the product sign is always in math mode.

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

$$1 \times 10^2$$

$$1 \cdot 10^2$$

$$1 \times 2^2$$

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

$$1.234(5)$$

$$1.234 \pm 0.005$$

$$1.234 [5]$$

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

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

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

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

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

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

Table 16: 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>

`-15673` `\num{-15673} \\\`
`-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}`

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

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

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

5.6 Multi-part numbers

`siunitx` recognises the idea of products and quotients in numbers, both with and without units. These multi-part numbers have a number of options affecting how they are processed. The options are summarised in Table 16.

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

$1 \times 2 \times 3$ `\num{1 x 2 x 3} \\\`
 $1 \times 10^4 \times 2(3) \times 3/4$ `\num{1e4 x 2(3) x 3/4} \\\`
 $4 \times 5 \times 6$ `\num[input-product=*]{4 * 5 * 6} \\\`
 $1/(2 \times 10^4)$ `\num{ 1 / 2e4 } \\\`
 $1 \times 10^2/(3 \times 10^4)$ `\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
slash and fraction.

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

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

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

```
\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 17).

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 17: 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 behavior 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}
```

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

Table 18: 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 18). 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 behavior can be altered using the `overwrite-functions` switch.

`space-before-unit` When “free standing” unit macros are created, their behavior 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 behavior for the inputs `\SI{10}{\metre}`

and

`\metre[10]`.

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

The first group of add-on units are those which provide convenient abbreviations (Table 19). 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 19: 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
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	\ns	ns
microsecond	\us	μ s
millisecond	\ms	ms
second	\s	s
femtomole	\fmol	fmol
picomole	\pmol	pmol
nanomole	\nmol	nmol
micromole	\umol	μ mol
millimole	\mmol	mmol
kilomole	\kmol	kmol
picoampere	\pA	pA
nanoampere	\nA	nA
microampere	\uA	μ A
milliampere	\mA	mA
kiloampere	\kA	kA
microlitre	\u.l	μ l
millilitre	\ml	ml
microliter	\uL	μ L
milliliter	\mL	mL
millihertz	\mHz	mHz
hertz	\Hz	Hz
kilohertz	\kHz	kHz
megahertz	\MHz	MHz
gigahertz	\GHz	GHz
terahertz	\THz	THz
millivolt	\mV	mV
kilovolt	\kV	kV
kilojoule	\kJ	kJ
electronvolt	\eV	eV
millielectronvolt	\meV	meV
kiloelectronvolt	\keV	keV
megaelectronvolt	\MeV	MeV
gigaelectronvolt	\GeV	GeV

Continued on next page

Table 20: 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

Table 21: Astronomy-related units
(load-configurations=astronomy).

Unit	Macro	Symbol
parsec	\parsec	pc
lightyear	\lightyear	ly

Continued from previous page

Unit	Abbreviation	Symbol
teraelectronvolt	\TeV	TeV
kilowatt hour	\kWh	kWh

\bit Binary data is expressed in units of bits and bytes. These are normally given prefixes
 \byte 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 20). The units \bit and \byte are also available.

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

Units for specialist areas are given in Tables 21 to 24. These are not SI units, but are defined here as they are in common usage in the subject areas concerned.

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 behavior of units is therefore controlled by a number of options

Table 22: Chemical engineering units
(load-configurations=chemical-engineering).

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

Table 23: Chemistry-related units
(load-configurations=chemistry).

Unit	Macro	Symbol
molar	<code>\molar</code>	mol dm^{-3}
	<code>\Molar</code>	M
torr	<code>\torr</code>	Torr

Table 24: Geophysics-related units
(load-configurations=geophysics).

Unit	Macro	Symbol
gon	<code>\gon</code>	gon

Table 25: Unit output options.

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

which alter either the processing of units or the output directly (Table 25).

`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 centered dot. To get the correct spacing it is necessary to use `{ }\cdot{ }` in the later case.

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

```
F2 lm cd
F2 · lm · cd
```

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

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

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

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}

A mol-1 s
A smol-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`. 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}

J/(mol K)
m/s2
J div (mol K)
J/mol K
```

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

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

J/mol/K
```

Finally, it is possible for the behavior of the `\per` function to depend on the surrounding environment. Setting the option to `symbol-or-fraction`, causes the same behavior as `symbol` when the surroundings are in line, and the same behavior as `fraction` when in a display context.

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

J/(mol K)


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


J/(mol K)
```

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

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

```

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

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

```

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

```

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

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

```

prefixes-as-symbols The unit prefixes (`\kilo`, *etc.*) are normally given as letters. However, the package can 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
300 MHz
\SI{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 26).

Table 26: Options for numbers with units.

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

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

```

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

```

`number-unit-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}

2.67F
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

product-units When a product of values 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`.

```
\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 m × 3 m × 4 m
 (2 × 3 × 4) m
 (2 × 3 × 4) m³
 2 × 3 × 4 m³
 2 m × 3 m × 4 m
 2 × 3 × 4 m

range-units The `range-units` option determines how the `\SIrange` function displays units. The standard setting is `repeat`, where both parts of the range have a unit. Alternatives are `brackets` and `single`.

```
\SIrange{2}{4}{\degreeCelsius} \\
\SIrange[range-units = brackets]{2}{4}{\degreeCelsius} \\
\SIrange[range-units = repeat]{2}{4}{\degreeCelsius} \\
\SIrange[range-units = single]{2}{4}{\degreeCelsius} \\
2 °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 27).

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 28.

Table 27: Options for tabular material.

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

```

\begin{table}
  \centering
  \caption{Parsing without aligning in an \texttt{S} column.}
  \label{tab:S:parse}
  \begin{tabular}
    {
      S
      S[table-parse-only]
    }
  \toprule
    {Decimal-centered} &
    {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-align-numbers` The alignment of numbers with the boundaries of the S column is controlled using the `table-align-numbers` 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 center of the column. This does not need any information in advance, and so is the standard setting. It works best for approximately symmetrical

Table 28: Parsing without aligning in an S column.

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

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 29, which uses somewhat exaggerated column headings to show the relative position of the cell contents.

```
\begin{table}
  \caption{Aligning the \texttt{S} column.}
  \label{tab:S:align}
  \centering
  \sisetup{
    table-figures-integer = 2,
    table-figures-decimal = 4
  }
  \begin{tabular}{
    S
    S[table-align-numbers = center]
    S[table-align-numbers = left]
    S[table-align-numbers = 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-align-numbers = center-decimal-marker` is set, as this mode always centers 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

Table 29: 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

table-sign-exponent
table-sign-mantissa

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 30.

```

\begin{table}
  \caption{Reserving space in \texttt{S} columns.}
  \label{tab:S:space}
  \sisetup{
    table-align-numbers = center,
    table-figures-integer = 2
  }
  \centering
  \begin{tabular}{
    S
    S[table-align-numbers = 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 30: 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

The table-printing code will omit any part of a number which has no space reserved, placing a warning in the LaTeX log. This means that uncertainties and exponents will not be printed if no space is reserved for them.

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

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

is equivalent to

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

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

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

will have the same effect as

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

It is important to note that any parts of a number *not* specified in the table format argument are set to be absent (the number of figures is set to zero). Setting the `table-format` option also resets `table-align-numbers` to center (Table 31).

Table 31: 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

```

\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
table-space-text-post

Space for material before and after the S column can be reserved by giving model text for the options table-space-text-pre and ...-post. This is then used to provide the necessary gap while maintaining alignment (Table 32).

```

\begin{table}
  \caption{Text before and after numbers.}
  \label{tab:S:ends}
  \centering
  \sisetup{
    table-figures-integer = 2,
    table-figures-decimal = 4,
    table-space-text-pre = $ > $,
  }

```

Table 32: Text before and after numbers.

Values
2.3456
34.2345 ^a
56.7835
> 90.473

```

table-space-text-post =
  \textsuperscript{\emph{a}}
}
\begin{tabular}{S}
\toprule
{Values} \\
\midrule
2.3456 \\
34.2345 \textsuperscript{\emph{a}} \\
56.7835 \\
$ > $ 90.473 \\
\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 33.

```

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

```

Table 33: The table-auto-round option.

Header	Header
1.2	1.200
1.2345	1.235

```

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 34).

```

\begin{table}
\caption{Aligning without parsing.}
\label{tab:S:nonparsed}
\sisetup{
  parse-numbers = false,
  table-figures-integer = 2,
  table-figures-decimal = 3
}
\centering
\begin{tabular}{
  S
  S[table-align-numbers = center]
  S[table-align-numbers = right]
  S[table-align-numbers = 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} \\
\bottomrule
\end{tabular}
\end{table}

```

Table 34: Aligning without parsing.

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

`table-align-text` 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-align-text` option, which recognises the values center, left and right (Table 35).

```
\begin{table}
  \caption{Aligning text in \texttt{S} columns.}
  \label{tab:S:text}
  \sisetup{
    table-align-numbers = center,
    table-figures-integer = 4,
    table-figures-decimal = 4
  }
  \centering
  \begin{tabular}{
    S
    S[table-align-text = left]
    S[table-align-text = 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-align-units` The contents of s columns can be centered or aligned to the left or right using the `table-align-units` option. As for the other alignment options, this recognises the choices center, left and right.

Table 35: 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 36: Alignment options in s columns.

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

```

\begin{table}
  \centering
  \caption{Alignment options in \texttt{s} columns.}
  \label{tab:s:align}
  \begin{tabular}
    {
      s[table-align-units = right]
      s
      s[table-align-units = 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-align` The three table alignment options (`table-align-numbers`, `table-align-text` and `table-align-units`) can be set to the same value using the `table-align` option. This will set all three alignment options to the same value (one of `center`, `right` or `left`).

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

Table 37: 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-oh m</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>\ker n -\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>

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 behavior 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 37. 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 `\SIUnitSymbolMicro`. Notice that these names capitalise the unit name (to make reading the macro name easier!).³

`\SIUnitSymbolArcminute`

`\SIUnitSymbolArcsecond`

`\SIUnitSymbolCelsius`

`\SIUnitSymbolDegree`

`\SIUnitSymbolMicro`

`\SIUnitSymbolOhm`

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

5.14 Other options

`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 behavior could not be turned off. The preamble-only option `strict` resets package behavior 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.

6 Hints for using `siunitx`

6.1 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.

6.2 Numbers greater than, less than and so forth

The numerical relationships `>`, `<`, *etc.* are not signs, and so the results obtained by using them as such with `siunitx` are not predictable. The supported method for including relations is to use them outside of numbers:

$1 \times 10^4 > 1 \times 10^{-4}$	<code>\(\num{1e4} > \num{1e-4} \) \\\</code>
$> 34.5 \text{ m}$	<code>\(\{>\} \SI{34.5}{\metre} \) \)</code>

Notice the extra braces in the second example line, needed to prevent TeX including more space around the `>`.

6.3 Expanding content in tables

When processing tables, `siunitx` will expand anything stored inside a macro, unless it is long or protected. LaTeX2e robust commands are also detected and are not expanded (Table 38). Values which would otherwise be expanded can be protected by wrapping them in a set of braces. As TeX itself will expand the first token in a table cell before `siunitx` can act on it, using the ϵ -TeX protected mechanism is the recommended course of action to prevent expansion of macros in table cells. (As is shown in the table, TeX's expansion of LaTeX2e robust commands can lead to unexpected results.)

Table 38: Values as macros in S columns.

Some Values		
12348.8	1234	34
12348.8	1234	
12348.8	1234	
12348.8	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 ϵ -TeX protected mechanism can be used to prevent this; using the `etoolbox` package provides a convenient way to apply this protection to existing functions. The general approach is illustrated in Table 39. 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 39: 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-align-numbers = 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}

```

6.4 Using siunitx with datatool

As illustrated in Table 39, 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 40).

```

\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 40: Displaying a datatool table.

value
6.7012
66.0212
64.902

```

\label{tab:xmpl:datatool}
\centering
\sisetup{
  parse-numbers      = false,
  table-align-numbers = center,
  table-figures-integer = 2,
  table-figures-decimal = 4
}
\renewcommand*\dtlrealalign{S}
\DTLdisplaydb{moredata}
\end{table}

```

6.5 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 behavior 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}%
}

```

6.6 Symbols and XeTeX

A small number of non-Latin symbols are needed by siunitx, notably Ω and μ . The package picks glyphs for these which are correct in the sense that they are upright (not italic) symbols, and match the LaTeX standard Computer Modern font. However, this does not make them the best choice if other fonts are in use, which is particularly common when XeTeX is being used.

XeTeX users will probably need to choose appropriate symbols themselves. The correct choice depends on the fonts in use, but many system fonts include Greek letters and

other symbols (which is not the case with most TeX-specific fonts). An appropriate setting could then be to use the text μ symbol in all cases:

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

6.7 Scaled document fonts with XeTeX

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

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

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

6.8 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:

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

6.9 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.

7 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 units and values. Spacing and so forth is handled in such a way as to make implementing the rules (relatively) easy.

7.1 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.⁴ 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.

⁴Some 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 20) 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.

7.2 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 value. 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 behavior. 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, centered) 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.⁵ 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}
```

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

Products and errors should show what unit applies to each value given. Thus (2 × 3) m is an ordered set of lengths of a geometric area, whereas 2 × 3 m is a length (and equal to 6 m). Thus, × 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 × 3 m². Although this does display the correct overall units, it is potentially-confusing and is not encouraged.

⁵Notice 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 between values, for example “see pages 1–5”. On the other hand, values with units could be misinterpreted as negative values if written in this way. As the unit–value 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.

7.3 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.⁶ Much better is to show division of all values by the unit, which leaves the entries as unitless ratios. This is illustrated in Table 41 and Fig. 1.

```
\begin{table}
  \centering
  \caption{An example of table labelling.}
  \label{tab:xmpl:unitless}
  \sisetup{
    table-align-numbers = 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 = \(\ t/\si{\second} \),
```

⁶For example, for an acceleration a , the expression $[a]$ is the dimensions of a , *i.e.* length per time squared in this case.

Table 41: 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 = \(\text{d}/\text{si}\{\text{metre}\} \),
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 42. 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-align-numbers = center}
\begin{tabular}{c}
S[
table-figures-integer = 1,
table-figures-decimal = 3,
table-figures-exponent = 1
]

```

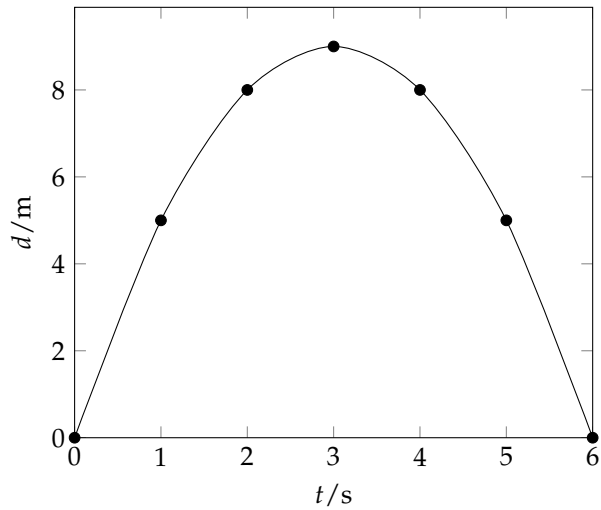


Figure 1: An example of graph labelling.

```

@{\, \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}

```

8 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

Table 42: 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

for the `Slstyle` 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.

Change History

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

numbers added	1	v2.0
v1.3		
General: Better definition for micro symbol	1	General: Complete re-write of package to add many new features
		1

Index

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

A		close-bracket (option)	24
add-arc-degree-zero (option)	27	\cm	30
add-arc-minute-zero (option)	27	color (option)	18
add-arc-second-zero (option)	27	complex-root-position (option)	24
add-decimal-zero (option)	21	copy-decimal-marker (option)	23
add-integer-zero (option)	21	\coulomb	8
allow-number-unit-breaks (option)	35	\cubed	10
\ampere	8	\cubic	10
\amu	30	D	
\ang	6	\dalton	9
angle-symbol-over-decimal (option)	28	\day	9
\angstrom	9	\deca	10
arc-separator (option)	27	\deci	10
\arcminute	9	\decibel	9
\arcsecond	9	\DeclareBinaryPrefix	12
\as	30	\DeclareSIPostPower	12
\astronomicalunit	9	\DeclareSIPrefix	12
\atomicmassunit	9	\DeclareSIPrePower	12
\atto	10	\DeclareSIQualifier	12
B		\DeclareSIUnit	11
\bar	9	\DeclareSIUnitWithOptions	11
\barn	9	\degree	9
\becquerel	8	\degreeCelsius	8
\bel	9	\deka	8
\bit	31	detect-display-math (option)	17
\bohr	9	detect-family (option)	16
bracket-unit-denominator (option)	33	detect-inline-weight (option)	16
\byte	31	detect-italic (option)	16
C		detect-mode (option)	16
\candela	8	detect-weight (option)	16
\celsius	8	\dm	30
\centi	10	E	
\clight	9	\electronmass	9

<code>\electronvolt</code>	9	<code>input-open-uncertainty</code> (option)	19
<code>\elementarycharge</code>	9	<code>input-product</code> (option)	25
<code>\eV</code>	31	<code>input-protect-tokens</code> (option)	20
<code>\exa</code>	10	<code>input-quotient</code> (option)	25
<code>\exbi</code>	32	<code>input-signs</code> (option)	19
<code>explicit-sign</code> (option)	21	<code>input-symbols</code> (option)	19
<code>exponent-base</code> (option)	24	<code>inter-unit-separator</code> (option)	33
<code>exponent-product</code> (option)	24		
		J	
F		<code>\joule</code>	8
<code>\farad</code>	8		
<code>\femto</code>	10	K	
<code>\fg</code>	30	<code>\kA</code>	30
<code>\fmol</code>	30	<code>\katal</code>	8
<code>forbid-literal-units</code> (option)	33	<code>\kelvin</code>	8
<code>fraction-function</code> (option)	26	<code>\keV</code>	31
<code>free-standing-units</code> (option)	28	<code>\kg</code>	30
<code>\fs</code>	30	<code>\kgmol</code>	32
		<code>\kHz</code>	31
G		<code>\kibi</code>	32
<code>\g</code>	30	<code>\kilo</code>	10
<code>\GeV</code>	31	<code>\kilogram</code>	8
<code>\GHz</code>	31	<code>\kJ</code>	31
<code>\gibi</code>	32	<code>\km</code>	30
<code>\giga</code>	10	<code>\kmol</code>	30
<code>\gmol</code>	32	<code>\knot</code>	9
<code>\gon</code>	32	<code>\kV</code>	31
<code>\gray</code>	8	<code>\kWh</code>	31
<code>group-decimal-digits</code> (option)	22		
<code>group-digits</code> (option)	22	L	
<code>group-four-digits</code> (option)	22	<code>\lbmol</code>	32
<code>group-integer-digits</code> (option)	22	<code>\lightyear</code>	32
<code>group-separator</code> (option)	22	<code>\liter</code>	9
		<code>\litre</code>	9
H		<code>load-configurations</code> (option)	29
<code>\hartree</code>	9	<code>\lumen</code>	8
<code>\hectare</code>	9	<code>\lux</code>	8
<code>\hecto</code>	10		
<code>\henry</code>	8	M	
<code>\hertz</code>	8	<code>\mA</code>	30
<code>\hour</code>	9	<code>math-rm</code> (option)	17
<code>\Hz</code>	31	<code>math-sf</code> (option)	17
		<code>math-tt</code> (option)	17
I		<code>\mebi</code>	32
<code>input-close-uncertainty</code> (option)	19	<code>\mega</code>	10
<code>input-complex-roots</code> (option)	19	<code>\meter</code>	8
<code>input-decimal-markers</code> (option)	19	<code>\metre</code>	8
<code>input-digits</code> (option)	19	<code>\MeV</code>	31
<code>input-exponent-markers</code> (option)	19	<code>\meV</code>	31
<code>input-ignore</code> (option)	19	<code>\mg</code>	30

<code>\MHz</code>	31	<code>complex-root-position</code>	24
<code>\mHz</code>	31	<code>copy-decimal-marker</code>	23
<code>\micro</code>	10	<code>detect-display-math</code>	17
<code>\milli</code>	10	<code>detect-family</code>	16
<code>\minute</code>	9	<code>detect-inline-weight</code>	16
<code>\mL</code>	31	<code>detect-italic</code>	16
<code>\ml</code>	31	<code>detect-mode</code>	16
<code>\mm</code>	30	<code>detect-weight</code>	16
<code>\mmHg</code>	9	<code>explicit-sign</code>	21
<code>\mmol</code>	30	<code>exponent-base</code>	24
<code>mode (option)</code>	17	<code>exponent-product</code>	24
<code>\Molar</code>	32	<code>forbid-literal-units</code>	33
<code>\molar</code>	32	<code>fraction-function</code>	26
<code>\mole</code>	8	<code>free-standing-units</code>	28
<code>\ms</code>	30	<code>group-decimal-digits</code>	22
<code>multi-part-units (option)</code>	36	<code>group-digits</code>	22
<code>\mV</code>	31	<code>group-four-digits</code>	22
N			
<code>\nA</code>	30	<code>group-integer-digits</code>	22
<code>\nano</code>	10	<code>group-separator</code>	22
<code>\nauticalmile</code>	9	<code>input-close-uncertainty</code>	19
<code>negative-color (option)</code>	25	<code>input-complex-roots</code>	19
<code>\neper</code>	9	<code>input-decimal-markers</code>	19
<code>\newton</code>	8	<code>input-digits</code>	19
<code>\ng</code>	30	<code>input-exponent-markers</code>	19
<code>\nm</code>	30	<code>input-ignore</code>	19
<code>\nmol</code>	30	<code>input-open-uncertainty</code>	19
<code>\ns</code>	30	<code>input-product</code>	25
<code>\num</code>	5	<code>input-protect-tokens</code>	20
<code>number-angle-separator (option)</code>	27	<code>input-quotient</code>	25
<code>number-unit-separator (option)</code>	36	<code>input-signs</code>	19
<code>\numrange</code>	6	<code>input-symbols</code>	19
O			
<code>\of</code>	10	<code>inter-unit-separator</code>	33
<code>\ohm</code>	8	<code>load-configurations</code>	29
<code>open-bracket (option)</code>	24	<code>math-rm</code>	17
options:		<code>math-sf</code>	17
<code>add-arc-degree-zero</code>	27	<code>math-tt</code>	17
<code>add-arc-minute-zero</code>	27	<code>mode</code>	17
<code>add-arc-second-zero</code>	27	<code>multi-part-units</code>	36
<code>add-decimal-zero</code>	21	<code>negative-color</code>	25
<code>add-integer-zero</code>	21	<code>number-angle-separator</code>	27
<code>allow-number-unit-breaks</code>	35	<code>number-unit-separator</code>	36
<code>angle-symbol-over-decimal</code>	28	<code>open-bracket</code>	24
<code>arc-separator</code>	27	<code>output-close-uncertainty</code>	24
<code>bracket-unit-denominator</code>	33	<code>output-complex-root</code>	23
<code>close-bracket</code>	24	<code>output-decimal-marker</code>	23
<code>color</code>	18	<code>output-open-uncertainty</code>	24
		<code>output-product</code>	26
		<code>output-quotient</code>	26
		<code>overwrite-functions</code>	28
		<code>parse-numbers</code>	20, 43

parse-units	35
per-mode	33
per-symbol	33
prefixes-as-symbols	35
product-units	36
qualifier-mode	35
quotient-mode	26
range-phrase	25
range-units	37
redefine-symbols	46
retain-explicit-plus	21
retain-unity-mantissa	21
retain-zero-exponent	21
round-mode	20
round-precision	20
separate-uncertainty	24
space-before-unit	28
sticky-per	34
strict	47
table-align	46
table-align-numbers	38
table-align-text	44
table-align-units	45
table-auto-round	43
table-figures-decimal	39
table-figures-exponent	39
table-figures-integer	39
table-figures-uncertainty	39
table-format	40
table-parse-only	37
table-sign-exponent	40
table-sign-mantissa	40
table-space-text-post	42
table-space-text-pre	42
text-rm	17
text-sf	17
text-tt	17
tight-spacing	25
uncertainty-separator	24
unit-optional-argument	28
use-brackets	24
use-xspace	28
output-close-uncertainty (option)	24
output-complex-root (option)	23
output-decimal-marker (option)	23
output-open-uncertainty (option)	24
output-product (option)	26
output-quotient (option)	26
overwrite-functions (option)	28
P	
\pA	30
parse-numbers (option)	20, 43
parse-units (option)	35
\parsec	32
\pascal	8
\pebi	32
\per	10
per-mode (option)	33
per-symbol (option)	33
\peta	10
\pg	30
\pico	10
\planckbar	9
\pm	30
\pmol	30
prefixes-as-symbols (option)	35
product-units (option)	36
\ps	30
Q	
qualifier-mode (option)	35
quotient-mode (option)	26
R	
\radian	8
\raiseto	10
range-phrase (option)	25
range-units (option)	37
redefine-symbols (option)	46
retain-explicit-plus (option)	21
retain-unity-mantissa (option)	21
retain-zero-exponent (option)	21
round-mode (option)	20
round-precision (option)	20
S	
\s	30
\second	8, 9
\SendSettingsToPgf	52
separate-uncertainty (option)	24
\SI	7
\si	6
\siemens	8
\sievert	8
\SIrange	7
\ssetup	15
\SIUnitSymbolAngstrom	47
\SIUnitSymbolArcminute	47
\SIUnitSymbolArcsecond	47

<code>\SIUnitSymbolCelsius</code>	47	<code>\THz</code>	31
<code>\SIUnitSymbolDegree</code>	47	<code>tight-spacing (option)</code>	25
<code>\SIUnitSymbolMicro</code>	47	<code>\tonne</code>	9
<code>\SIUnitSymbolOhm</code>	47	<code>\torr</code>	32
<code>space-before-unit (option)</code>	28	<code>\tothe</code>	10
<code>\square</code>	10		
<code>\squared</code>	10	U	
<code>\steradian</code>	8	<code>\uA</code>	30
<code>sticky-per (option)</code>	34	<code>\ug</code>	30
<code>strict (option)</code>	47	<code>\uL</code>	31
		<code>\ul</code>	31
T		<code>\um</code>	30
<code>table-align (option)</code>	46	<code>\umol</code>	30
<code>table-align-numbers (option)</code>	38	<code>uncertainty-separator (option)</code>	24
<code>table-align-text (option)</code>	44	<code>unit-optional-argument (option)</code>	28
<code>table-align-units (option)</code>	45	<code>\us</code>	30
<code>table-auto-round (option)</code>	43	<code>use-brackets (option)</code>	24
<code>table-figures-decimal (option)</code>	39	<code>use-xspace (option)</code>	28
<code>table-figures-exponent (option)</code>	39		
<code>table-figures-integer (option)</code>	39	V	
<code>table-figures-uncertainty (option)</code>	39	<code>\volt</code>	8
<code>table-format (option)</code>	40		
<code>table-parse-only (option)</code>	37	W	
<code>table-sign-exponent (option)</code>	40	<code>\watt</code>	8
<code>table-sign-mantissa (option)</code>	40	<code>\weber</code>	8
<code>table-space-text-post (option)</code>	42		
<code>table-space-text-pre (option)</code>	42	Y	
<code>\tebi</code>	32	<code>\yobi</code>	32
<code>\tera</code>	10	<code>\yocto</code>	10
<code>\tesla</code>	8	<code>\yotta</code>	10
<code>\TeV</code>	31		
<code>text-rm (option)</code>	17	Z	
<code>text-sf (option)</code>	17	<code>\zebi</code>	32
<code>text-tt (option)</code>	17	<code>\zepto</code>	10
		<code>\zetta</code>	10