siunitx — A comprehensive (SI) units package*

Joseph Wright†

Released 2009/05/31

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

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

Contents

*This file describes version v1.2k, last revised 2009/05/31.

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

Part I
Introduction

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

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

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

[1] General Conference on Weights and Measures.
[2] International System of Units.

Part II
Using the siunitx package

1 For the impatient

siunitx provides the user macros:

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

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

By default, all text is typeset in the current upright, serif maths font. This can be changed by setting the appropriate package options: `obeyall` will use the current font for typesetting.

The package includes a "unit processor", which allows the use of named units or literal values. Named units are processed to correctly include powers.

10 g
23.4 g cm^3
1×10^{34}
1°2′3″
16.7 m s^{-1}
30 × 10^3 Hz
1.2 mm × 3.56 mm × 9.2 mm
−4.5 cm
J mol^{-1} K^{-1}
$\frac{\text{J}}{\text{mol K}}$
1.2346
9.8000

Heading
1.3
134.2
3.56
74.7

```
\SI{10}{\gram}\\
\SI{23.4}{g.cm^3}\\
\num{1e34}\\
\ang{1;2;3}\\
\emph{\SI{16,7}{\metre\per\second}}\\
\textbf{\SI{30e3}{\Hz}}\\
\SI{1.2 x 3.56 x 9.2}{\milli\metre}\\
\sisetup{obeyall}
\textbf{\SI{-4.5}{\cm}}\\
\si{\joule\per\mole\per\kelvin}\\
\si[per=frac]{\joule\per\mole\per\kelvin}\\
\num[dp=4]{1.23456}\\
\num[dp=4]{9.8}\\
\begin{tabular}{S[tabformat=3.2]}
  \toprule
  {Heading}\\
  \midrule
  1.3 \\
  134.2 \\
  3.56 \\
  74,7 \\
  \bottomrule
\end{tabular}
```

2 Requirements

siunitx requires a reasonably up to date TEX system. The package requires ε-TEX-extensions, which should be available on most systems.[3] The following packages are also needed:

- array and xspace: from the tools bundle, which should be available to everyone;
- xkeyval: this processes the option handling, and needs to be at least v2.5;
- amstext: from the $\mathcal{AMS}$TEX support bundle (the $\mathcal{AMS}$ fonts are also needed to provide the default upright μ).

Hopefully most people using the package will have access to all of those items.

To use the `fraction=sfrac` option, the xfrac package is needed. This needs various experimental LATEX3 packages. As a result, siunitx does not load xfrac. If you want to use `fraction=sfrac`, *you* need to load xfrac in your preamble before siunitx.[4] If the package is not loaded, `fraction=sfrac` falls back on a nicefrac-like method. The interested user should look at the xfrac documentation for reasons this might not be ideal.[5]

3 Loading the package

siunitx is loaded by the usual LATEX method.

```
\usepackage[<options>]{siunitx}
```

As is shown in the example, the package can be loaded with one or more options, using the key–value system. The full range of package options are described in Section 11; some options are described in the along with the appropriate user macros. Most of the user macros accept the same key–value settings as an optional argument.

4 Numbers

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

1 123 1234 12 345
0.1 0.123 0.1234 0.123 45
1×10^{10} 3.45×10^{-4} -10^{10}

```
\num{1}     \num{123}      \num{1234}   \num{12345}\\
\num{0.1}   \num{0.123}    \num{0,1234} \num{.12345}\\
\num{1e10}  \num{3.45d-4}  \num{-e10}
```

[3] If you have an old LATEX try "`elatex`" rather than "`latex`".

[4] This document has been compiled in this way. You have to load xfrac first as otherwise very nasty things happen with xkeyval. MiKTEX users should note that the packaged versions of expl3, template and xparse will not work with xfrac: download copies from CTAN!

[5] On the other hand, some fractional units will look really bad with `\sfrac`. Use this option with caution.

Various error-checking systems are built into the package, so that if ⟨*number*⟩ does not contain any numeric characters, a warning is issued. Isolated signs are also detected. The package recognises `(` and `)` as "extra" characters, which can be used to indicate the error in a number.[6] The `seperr` causes this data to be given as a separate error value. If the number also contains an exponent, then brackets are re-added after the separation to ensure that meaning is not lost.

$1.234(5) = 1.234 \pm 0.005$
$1.234(5) \times 10^{6} = (1.234 \pm 0.005) \times 10^{6}$

```
$\num{1.234(5)} = \num[seperr]{1.234(5)}$\\
$\num{1.234(5)e6} = \num[seperr]{1.234(5)e6}$
```

The same applies to the unit and value macro `\SI`, described later, for example the rest mass of an electron [1]:

$m_\mathrm{e} = 9.109\,389\,7(54) \times 10^{-31}\,\mathrm{kg}$
$m_\mathrm{e} = (9.109\,389\,7 \pm 0.000\,005\,4) \times 10^{-31}\,\mathrm{kg}$

```
$ m_{\mathrm{e}}
  = \SI{9.1093897(54)e-31}{\kg} $\\
$ m_{\mathrm{e}}
  = \SI[seperr]{9.1093897(54)e-31}{\kg} $
```

A number of effects are available as options. These are fully explained in Section 11. Some of the more useful options are illustrated here. By default, the output of the package is typeset in maths mode. However, the use of the current text font can be forced.[7]

1 234 567 890 1 234 567 890

```
\num{1234567890} \num[mode=text]{1234567890}
```

siunitx can automatically add zeros and signs to numbers. This can be altered as desired.

1 1.0
2 +2
3×10^{4} $+3 \times 10^{4}$ $+3 \times 10^{+4}$
0.5 .5

```
\num{1.}   \num[padnumber=all]{1.}\\
\num{2}    \num[addsign=all]{2}\\
\num{3e4}  \num[addsign=mant]{3e4} \num[addsign=all]{3e4}\\
\num{.5}   \num[padnumber=none]{.5}
```

The separation of digits can be turned on and off, and the output changed.

1234 1 234
12 345 12,345
12345

```
\num{1234}  \num[sepfour=true]{1234}\\
\num{12345} \num[digitsep=comma]{12345}\\
\num[digitsep=none]{12345}
```

The formatting of exponents is also customisable.

1×10^{10} $1 \cdot 10^{10}$
2×10^{20} 2×5^{20}
3×10^{30} $3{\times}10^{30}$

```
\num{1e10} \num[expproduct=cdot]{1e10}\\
\num{2e20} \num[expbase=5]{2e20}\\
\num{3e30} \num[expproduct=tighttimes]{3e30}
```

siunitx can automatically add colour to negative numbers, which is often useful for highlighting purposes. This is turned on with the `colourneg` option; the colour used is set by `negcolour`. Both of these are available with the US spellings: `colorneg` and `negcolor`.

-1
-2
3×10^{-3}
-4

```
\num{-1}\\
\sisetup{colourneg} \num{-2}\\
\num{3e-3}\\
\num[negcolour=blue]{-4}
```

[6] This is common in chemical crystallography, for example.

[7] This document is typeset using lowercase numbers in text mode, which emphasises the effect here.

siunitx can automatically zero-fill and round to a fixed number of decimal places. This is controlled by two options `fixdp` and `dp`. The later is an integer which specifies how many places to fix to; setting this option automatically sets `fixdp` to `true`. The place-fixing system will only alter pure numbers: for example, any error component will result in the input being left unchanged.

```
\num{1.23456}\\
\sisetup{dp=2}
\num{1.23456}\\
\num{9.8}\\
\num{-10.432}\\
\num{44.3221(2)}
```

1.234 56
1.23
9.80
−10.43
44.3221(2)

5 Tabular material

5.1 Aligning numbers

Centring numbers in tabular content is handled by a new column type, the `S` column. This is based closely on the dcolumn method for centring numbers in columns, but adds the functionality of the `\num` macro.[8]

By default, the decimal marker of the number is placed at the centre of the column, which then resizes to accommodate the width of the contents (Table 1). This behaviour is set by the `tabnumalign=centredecimal` option. By setting the `tabnumalign` option to `centre`, the centre of the space reserved for the number is placed at the centre of the column. The space reserved is stored in `tabformat`, which is of the form ⟨*before*⟩⟨*dec*⟩⟨*after*⟩, where ⟨*before*⟩ is the number of characters before the decimal marker and ⟨*after*⟩ is the number after. Thus in the example, `tabformat=2.4` provides space for two digits before the decimal marker and four after. `tabnumalign` can also be set to `left` and `right`, with the expected results.

```
\begin{table}
  \caption{Behaviour of \texttt{S} column type}
  \label{tab:default}
  \centering
  \begin{tabular}{%
    S%
    S[tabnumalign=centre,tabformat=2.4]%
    S[tabnumalign=right,tabformat=2.4]%
    S[tabnumalign=centre,tabformat=2.4,decimalsymbol=comma]}
  \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}
```

[8] The approach used is actually a combination of dcolumn for centring the material and numprint for processing it. It will therefore give rather different results than the `n` and `N` column types in numprint.

Table 1: Behaviour of `S` column type

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

The `tabformat` setting can also be used to reserve space for numbers containing exponents. This is given in the same format as above, but with a mantissa and exponent part (Table 2). Notice that this is designed to expect that numbers will contain a mantissa. Exponents can either be aligned so that the "×" symbols match up vertically, or the exponent part can be allowed to move across as needed. Space for signs is added by using any sign in the `tabformat`, so for example `tabformat=+2.2` and `tabformat=-2.2` have exactly the same effect. Setting `tabformat` will automatically switch `tabnumalign` from `centredecimal` to `centre`, if the former is currently set. In other cases, the current alignment option is retained.

```
\begin{table}
  \caption{Exponents in tables}
  \label{tab:exptab}
  \centering
  \begin{tabular}{%
    S[tabnumalign=right,tabformat=2.2e2]%
    S[tabnumalign=centre,tabformat=2.2e1.1]%
    S[tabnumalign=centre,tabformat=2.2e1.1,tabalignexp=false]%
    S[tabnumalign=centre,tabformat=+2.2]}
  \toprule
      {Longer values}
    & {Longer values}
    & {Longer values}
    & {Values} \\
  \midrule
  2.3e1    &  2.34e1  &  2.34e1  & +2.31 \\
  34.23e45 & 34.23e45 & 34.23e45 & 34.23 \\
  56.78    & 56.78    & 56.78    & -56.78 \\
  1.0e34   & 1.0e34   & 1.0e34   & +-1.0 \\
  \bottomrule
  \end{tabular}
\end{table}
```

Data not to be processed as a number should be protected by wrapping it in braces: this is most likely to be true for column headers (again as illustrated). By default, the contents of non-numeric cells are centred. This can be altered by setting `tabtextalign`, which can be set to `left`, `right` or `centre`. The use of digit separators in table columns is accounted for: extra space is reserved if digit separators will be added.

Table 2: Exponents in tables

Longer values	Longer values	Longer values	Values
2.3×10^{1}	2.34×10^{1}	2.34×10^{1}	2.31
34.23×10^{45}	34.23×10^{45}	34.23×10^{45}	34.23
56.78	56.78	56.78	−56.78
1.0×10^{34}	1.0×10^{34}	1.0×10^{34}	±1.0

Table 3: Number and units in tables

Value	Unit
2.16×10^{-5}	$\mathrm{m^2\,s^{-1}}$
2.83×10^{-6}	$\mathrm{m^2\,s^{-1}}$
7.39×10^{3}	$\mathrm{Pa\,m^3\,mol^{-1}}$
1.0×10^{5}	Pa

5.2 Columns of units

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. The alignment of material in `s` columns is governed by the `tabunitalign` option.

```
\begin{table}
\centering
\caption{Number and units in tables}
\label{tab:num-unit}
  \begin{tabular}{%
    S[tabformat=1.2e-1,tabnumalign=centre]%
    s[tabunitalign=left]}
  \toprule
    {Value} & \multicolumn{1}{c}{Unit} \\
  \midrule
    2.16e-5 & \metre\squared\per\second \\
    2.83e-6 & \metre\squared\per\second \\
    7.39e3  & \pascal\cubic\metre\per\mole \\
    1.0e5   & \pascal \\
  \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 4. Notice that the braces do not prevent processing and colouring of the cell contents.

```
\begin{table}
\centering
\caption{The \texttt{s} column processes everything}
```

Table 4: The `s` column processes everything

Unit	Unit
m^3	m^3
kg	kg

```
\label{tab:s-limits}
  \begin{tabular}{%
    s[colourall,colour=orange]%
    s[colourall,colour=orange]}
  \toprule
    {Unit}     & \multicolumn{1}{c}{Unit}\\
  \midrule
    {m^3}      & \multicolumn{1}{c}{\si{m^3}} \\
    \kilogram & \kilogram \\
  \bottomrule
  \end{tabular}
\end{table}
```

6 Angles

`\ang` Angles can be typeset using the `\ang` command. This takes two arguments, `\ang[`⟨*options*⟩`]{`⟨*angle*⟩`}`, where ⟨*options*⟩ can be any of the package options to apply only to this value. ⟨*angle*⟩ can be given either as a decimal number or as a semi-colon separated list of degrees, minutes and seconds, *i.e.* `\ang{`⟨*decimal angle*⟩`}` or `\ang{`⟨*degrees*⟩`;`⟨*minutes*⟩`;`⟨*seconds*⟩`}`. By default, no space is introduced between angles and the degrees, minutes and seconds markers.

10° 12.3° 4.5°
1°2′3″ 0°0′1″
10° −0°1′

```
\ang{10} \ang{12.3} \ang{4,5}\\
\ang{1;2;3} \ang{;;1}\\
\ang{+10;;} \ang{-0;1;}
```

By default, angles with no degrees (or minutes) are zero-filled; angles with degrees but no minutes or seconds are not filled. This behaviour can be altered using the package options.

0°0′1″ 1″
2° 2°0′0″
0°3′0″ 4°0′0″ 0°0′5″

```
\ang{;;1} \ang[padangle=none]{;;1}\\
\ang{2;;} \ang[padangle=all]{2;;}\\
\sisetup{padangle=all} \ang{;3;} \ang{4;;} \ang{;;5}
```

The `\num` macro is used to typeset each number of the angle, so the options for `\num` also apply here. The `anglesep` value can be used to separate degrees, minutes and seconds.

1.05° 1,05°
3.678 90° 3.678,90°
9°8′7″ 9° 8′ 7″

```
\ang{1.05}    \ang[decimalsymbol=comma]{1.05}\\
\ang{3.67890} \ang[digitsep=comma]{3.67890}\\
\ang{9;8;7}   \ang[anglesep=thin]{9;8;7}
```

The degrees, minutes and seconds signs can be placed over the decimal sign using the `astroang` option. This is designed on the assumption that only the last number given has a decimal part.

1.2° 1°2
1°2.3′ 1°2′.3
1°2′3.4″ 1°2′3″.4

```
\ang{1.2} \ang[astroang]{1.2}\\
\ang{1;2.3;} \ang[astroang]{1;2.3;}\\
\ang{1;2;3.4} \ang[astroang]{1;2;3.4}
```

7 Units and values

`\SI` The core aim of siunitx is correctly typesetting values which have units. The main output macro here is `\SI`, which has the same syntax as the macros with the same name in SIstyle and unitsdef packages. The `\SI` macro takes two mandatory arguments, in addition to the optional set up argument, and a second optional argument: `\SI[`⟨*options*⟩`]{`⟨*number*⟩`}[`⟨*preunit*⟩`]{`⟨*unit*⟩`}`. The ⟨*number*⟩ argument operates in exactly the same manner as the equivalent argument of the `\num` macro. ⟨*unit*⟩ will be typeset with a non-breakable space between it and the preceding number, with font control as outlined earlier. Finally, ⟨*preunit*⟩ is a unit to be typeset *before* the numerical value (most likely to be a currency). Some examples illustrate the general power of the macro.

1.23 J mol^{-1} K^{-1}
0.23 × 10^{7} cd
£1.99/kg
70 m s^{-1}
1.345 $^{A}/_{mol}$

```
\SI[mode=text]{1.23}{J.mol^{-1}.K^{-1}}\\
\SI{.23e7}{\candela}\\
\SI[per=slash]{1.99}[\pounds]{\per\kilogram}\\
\SI{70}{\metre\per\second}\\
\SI[per=frac,fraction=nice]{1,345}{\ampere\per\mole}
```

The use of unit macros outside of the `\SI` macro is described later.

7.1 Literal units

Units can be input in two ways, inspired by SIstyle and SIunits. The SIstyle-like method uses literal input. Five characters have a special meaning:

- "^" and "_" The superscript and subscript characters can be used without the usual need for surrounding maths characters (`$`);
- "." and ",": the full stop (point) symbol and comma are made active, and produce the current contents of the `unitsep` option;
- "~" The contents of the `unitspace` option are typeset by a tilde.

This allows ready input of units.

10 kg m s^{-2}
1.453 g/cm^{3}
6.345 $\mathrm{kg_{pol}\,mmol_{cat}^{-1}}$
33.562 cd s
100 m s^{-2}

```
\SI{10}{kg.m.s^{-2}}\\
\SI{1.453}{g/cm^3}\\
\SI{6.345}{kg_{pol}.mmol_{cat}^{-1}}\\
\SI{33.562}{cd~s}\\
\SI[unitsep=medium]{100}{m.s^{-2}}
```

The literal unit system will correctly typeset input containing the symbols µ (micro), ° (degree) and Å (ring-A).[9]

10 µm
20 °C
30 Å

```
\SI{10}{µm}\\
\SI{20}{°C}\\
\SI{30}{Å}
```

allowlitunits

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 `allowlitunits` option to `false`: the standard setting is **`true`**.

7.2 The unit interpreter

The second operation mode for the `\SI` macro is based on the behaviour of Slunits. 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.

$10\,\mathrm{kg\,m\,s^{-2}}$ `\SI{10}{\kilo\gram\metre\per\second\squared}\\`
$1.453\,\mathrm{g\,cm^{-3}}$ `\SI{1.453}{\gram\per\cubic\centi\metre}\\`
$33.562\,\mathrm{cd\,s}$ `\SI{33.562}{\candela\second}\\`
$100\,\mathrm{m\,s^{-2}}$ `\SI[unitsep=medium]{100}{\metre\per\Square\second}\\`
$4.56 \times 10^{3}\,\mathrm{m\,s^{-1}}$ `\SI[prefixsymbolic=false]{4.56}{\kilo\metre\per\second}`

On its own, this is very similar to Slunits, and is less convenient than the direct input method.[10] 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. Units may be re-defined to give different output, and handling of reciprocal values can be altered.

$10\,\frac{\mathrm{g\,m}}{\mathrm{s^2}}$ `\SI[per=frac,fraction=frac]{10}{\gram\metre\per\second\squared}\\`
$1.453\,\mathrm{g/cm^3}$ `\SI[per=slash]{1.453}{\gram\per\cubic\centi\metre}\\`
$33.562\,\mathrm{cd\,s}$ `\SI{33.562}{\candela\second}\\`
$100\,\mathrm{m/s^2}$ `\SI[per=frac,fraction=nice]{100}{\metre\per\Square\second}`

The unit processor will trap *some* errors in the input and give the "best guess" result. However, it is down to the user to check the output.

7.3 Powers of units

\Square
\ssquare
\squared
\cubic
\cubed

Including powers in units is handled using a "natural language" method. Thus preceding a unit by `\Square` or `\cubic` which raise the unit to the appropriate power, while `\squared` or `\cubed` follow the unit they apply to. The `\Square` macro is capitalised to avoid a name clash with pstricks; the alternative `\ssquare` is also provided.

$10\,\mathrm{m^2}$ `\SI{10}{\metre\squared}\\`
$20\,\mathrm{m^2}$ `\SI{20}{\Square\metre}\\`
$30\,\mathrm{m^3}$ `\SI{30}{\metre\cubed}\\`
$40\,\mathrm{m^3}$ `\SI{40}{\cubic\metre}`

\per

The `\per` macro intelligently creates reciprocal powers, and also adds the power -1 when appropriate.

[9] Currently this works with XeTeX and inputenc using the `latin1`, `latin5` and `latin9` encodings.
[10] Users of Slunits should note the lack of need for a `\usk`-type macro.

$10\,\mathrm{s}^{-2}$ `\SI{10}{\per\second\squared}\\`
$20\,\mathrm{s}^{-2}$ `\SI{20}{\per\Square\second}\\`
$30\,{}^{1}/_{\mathrm{s}^{3}}$ `\SI[per=frac,fraction=nice]{30}{\per\second\cubed}\\`
$40/\mathrm{s}^{3}$ `\SI[per=slash]{40}{\per\cubic\second}\\`
$50\,\mathrm{s}^{-1}$ `\SI{50}{\per\second}\\`
$60\,\mathrm{m}^{-1}\,\mathrm{cd}^{2}$ `\SI{60}{\per\metre\Square\candela}`

`\tothe` `\raiseto` For powers not defined above or with `\newpower`, the `\tothe` macro can be used "in line" to produce a power. As follows from standard English usage, this comes after the unit. `\raiseto` achieves the same, but is used *before* a unit to add a power *after*.[11]

$16.86\,\mathrm{m}^{4}$ `\SI{16.86}{\metre\tothe{4}}\\`
$7.895\,\mathrm{N}^{-6}$ `\SI{7.895}{\raiseto{-6}\newton}\\`
$1.34\,\mathrm{K}^{-7}$ `\SI{1.34}{\per\kelvin\tothe{7}}`

7.4 Units with no values

`\si` For typesetting the symbol for a unit on its own, with the full font control and without extra spaces, the `\si` macro is provided.[12] The macro name avoids a clash with the functionality of the earlier packages, but is similar to `\ilu` from the unitsdef package.

$\mathrm{kg\,m/s^{2}}$ `\SI{}{kg.m/s^2}\\`
$\mathrm{kg\,m/s^{2}}$ `\si{kg.m/s^2}\\`
$\mathrm{mol\,dm^{-3}}$ `\si[mode=text,unitsep=thin]{\mole\per\cubic\deci\metre}`

7.5 Free-standing units

Users of the unitsdef package will be accustomed to using unit macros on their own (following a value) or with an optional argument containing a number. In both cases, only a single unit macro could be used. siunitx supports both operation modes, with the limitation that units trailing values loose font control of the value. When used in this way, the units *do not* take an optional keyval argument.

$123\,\mathrm{m}$ `\sisetup{prespace,allowoptarg}`
$123\,\mathrm{K}$ `123\metre\\`
$234\,\mathrm{A}$ `\kelvin[123]\\`
$6\,\mathrm{s}$ `\sisetup{mode=text} \ampere[234]\\`
`6\second`

7.6 Pre-defined units, prefixes and powers

`\metre` `\meter` The package always defines the seven base SI units, irrespective of any package options given (Table 5). The kilogram is notable as by default it is a *base* unit with a prefix. Thus, when the package is loaded with the option `load={}`, `\kilo`

[11] `\raiseto` acts in the same way as `\tothe` when used in a literal context: the power will be produced where the macro is, rather than moving after the next item.

[12] The same effect can be achieved using the `\SI` macro with an empty numerical argument.

Table 5: The seven base SI units

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

Table 6: The SI prefixes (load=prefix)

Prefix	Macro	Power	Symbol	Prefix	Macro	Power	Symbol
yocto	\yocto	10^{-24}	y	deca	\deca	10^{1}	da
zepto	\zepto	10^{-21}	z	hecto	\hecto	10^{2}	h
atto	\atto	10^{-18}	a	kilo	\kilo	10^{3}	k
femto	\femto	10^{-15}	f	mega	\mega	10^{6}	M
pico	\pico	10^{-12}	p	giga	\giga	10^{9}	G
nano	\nano	10^{-9}	n	tera	\tera	10^{12}	T
micro	\micro	10^{-6}	µ	peta	\peta	10^{15}	P
milli	\milli	10^{-3}	m	exa	\exa	10^{18}	E
centi	\centi	10^{-2}	c	zetta	\zetta	10^{21}	Z
deci	\deci	10^{-1}	d	yotta	\yotta	10^{24}	Y

and \gram *are not defined*. As metre is often spelled as "meter" in the US, the macro \meter is provided in addition to the \metre macro.[13]

By default, a number of additional definitions are created by the package. These are controlled by the load and noload options. Unless specifically requested with the option noload=prefix, siunitx defines the standard prefixes for powers of ten (Table 6). This leads to the redefinition of \kilogram as \kilo\gram. The macro \deka is provided, as this is used as an alias for \deca in some places. The package also defines a number of derived SI units which have assigned names and symbols (Table 7). Note that \Gray is capitalised to avoid a name clash with the pstricks package.[14]

In addition to these units, there are three other groups of units for use with the SI system which do not fit into the above. These are those derived from physical measurements (Table 8), those considered "accepted" (Table 9), and those accepted temporarily (Table 10).[15] The unit "litre" is often spelled "liter" in the US; both spellings are provided by siunitx, with \liter giving L and \litre producing l.

\litre
\liter

[13] The official SI spelling for the unit is "metre".
[14] The macros \ohm and \celsius are not defined by siunitx if the gensymb package is loaded.
[15] These are supposed to be replaced over time by SI units.

Table 7: The derived SI units with defined names (`load=named`)

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

Table 8: Units derived from experiments (`load=physical`)

Unit	Macro	Symbol
electron volt	`\electronvolt`	eV
unified atomic mass unit	`\atomicmassunit`	u
	`\atomicmass`	u

Table 9: Units accepted for use with SI (`load=accepted`)

Unit	Macro	Symbol
bel	`\bel`	B
day	`\Day`	d
	`\dday`	d
degree	`\degree`	°
hour	`\hour`	h
litre	`\litre`	l
	`\liter`	L
minute	`\minute`	min
minute (arc)	`\arcmin`	′
neper	`\neper`	Np
percent	`\percent`	%
second (arc)	`\arcsec`	″
tonne	`\tonne`	t

Table 10: Additional (temporary) SI units (load=addn)

Unit	Macro	Symbol
ångström	\angstrom	Å
are	\are	a
curie	\curie	Ci
bar	\BAR	bar
	\bbar	bar
barn	\barn	b
gal	\gal	Gal
hectare	\hectare	ha
millibar	\millibar	mbar
rad	\rad	rad
rem	\rem	rem
roentgen	\roentgen	R

7.7 Prefixed and abbreviated units

Many basic units have prefixes which are commonly used with the unit, such as centimetre or megahertz. The package therefore defines a number of common prefixed units (load=prefixed). Several of these also have obvious abbreviations (such as \MHz for \megahertz), which are made available by load=abbr. In common with the units discussed above, the prefixed and abbreviated unit definitions are loaded by default.

Table 11: Prefixed (load=prefixed) and abbreviated (load=abbr) units

Unit	Macro	Symbol	Abbreviation
Masses			
kilogram	\kilogram	kg	\kg
femtogram	\femtogram	fg	\fg
picogram	\picogram	pg	\pg
nanogram	\nanogram	ng	\nanog
microgram	\microgram	µg	\micg
milligram	\milligram	mg	\mg
atomic mass	\atomicmass	u	\amu
Lengths			
picometre	\picometre	pm	\picom
nanometre	\nanometre	nm	\nm
micrometre	\micrometre	µm	\micm
millimetre	\millimetre	mm	\mm
centimetre	\centimetre	cm	\cm
decimetre	\decimetre	dm	\dm
kilometre	\kilometre	km	\km

Continued on next page

Unit	Macro	Symbol	Abbreviation
Times			
second	`\second`	s	`\Sec`
attosecond	`\attosecond`	as	`\as`
femtosecond	`\femtosecond`	fs	`\fs`
picosecond	`\picosecond`	ps	`\ps`
nanosecond	`\nanosecond`	ns	`\ns`
microsecond	`\microsecond`	µs	`\mics`
millisecond	`\millisecond`	ms	`\ms`
Moles			
femtomole	`\femtomole`	fmol	`\fmol`
picomole	`\picomole`	pmol	`\pmol`
nanomole	`\nanomole`	nmol	`\nmol`
micromole	`\micromole`	µmol	`\micmol`
millimole	`\millimole`	mmol	`\mmol`
kilomole	`\kilomole`	kmol	`\kmol`
Currents			
picoampere	`\picoampere`	pA	`\pA`
nanoampere	`\nanoampere`	nA	`\nA`
microampere	`\microampere`	µA	`\micA`
milliampere	`\milliampere`	mA	`\mA`
kiloampere	`\kiloampere`	kA	`\kA`
Areas			
square centimetre	`\squarecentimetre`	cm^2	`\cms`
	`\centimetresquared`	cm^2	
square metre	`\squaremetre`	m^2	
square kilometre	`\squarekilometre`	km^2	
Volumes			
microlitre	`\microlitre`	µl	`\micl`
millilitre	`\millilitre`	ml	`\ml`
microliter	`\microliter`	µL	`\micL`
milliliter	`\milliliter`	mL	`\mL`
cubic centimetre	`\cubiccentimetre`	cm^3	`\cmc`
	`\centimetrecubed`	cm^3	
cubic decimetre	`\cubicdecimetre`	dm^3	`\dmc`
Frequencies			
hertz	`\hertz`	Hz	`\Hz`
millihertz	`\millihertz`	mHz	`\mHz`
kilohertz	`\kilohertz`	kHz	`\kHz`
megahertz	`\megahertz`	MHz	`\MHz`
gigahertz	`\gigahertz`	GHz	`\GHz`
terahertz	`\terahertz`	THz	`\THz`

Continued on next page

Unit	Macro	Symbol	Abbreviation
Potentials			
millivolt	`\millivolt`	mV	`\mV`
kilovolt	`\kilovolt`	kV	`\kV`
Energies			
kilojoule	`\kilojoule`	kJ	`\kJ`
electronvolt	`\electronvolt`	eV	`\eV`
millielectronvolt	`\millielectronvolt`	meV	`\meV`
kiloelectronvolt	`\kiloelectronvolt`	keV	`\keV`
megaelectronvolt	`\megaelectronvolt`	MeV	`\MeV`
gigaelectronvolt	`\gigaelectronvolt`	GeV	`\GeV`
teraelectronvolt	`\teraelectronvolt`	TeV	`\TeV`
kilowatthour	`\kilowatthour`	kWh	`\kWh`
Powers			
milliwatt	`\milliwatt`	mW	
kilowatt	`\kilowatt`	kW	
megawatt	`\megawatt`	MW	
Capacitances			
femtofarad	`\femtofarad`	fF	
picofarad	`\picofarad`	pF	
nanofarad	`\nanofarad`	nF	
microfarad	`\microfarad`	μF	
millifarad	`\millifarad`	mF	
Resistances			
kilohm	`\kilohm`	kΩ	
megohm	`\megohm`	MΩ	
gigaohm	`\gigaohm`	GΩ	
millisiemens	`\millisiemens`	mS	
Forces			
millinewton	`\millinewton`	mN	
kilonewton	`\kilonewton`	kN	
Other units			
hectopascal	`\hectopascal`	hPa	
megabecquerel	`\megabecquerel`	MBq	
millisievert	`\millisievert`	mSv	

7.8 Defining new units

`\newunit` `\renewunit` `\provideunit`

New units are produced using the `\newunit` macro. This works as might be expected: `\newunit[`⟨*options*⟩`]{`⟨*unit*⟩`}{`⟨*symbol*⟩`}`, where ⟨*symbol*⟩ can contain literal values, other units, multiple prefixes, powers and `\per`. The ⟨*options*⟩ argument can be any suitable options, and applies the specific unit macro only.

The most obvious example for using this macro is the `\degree` unit.[16] The (first) optional argument to `\SI` and `\si` can be used to override the settings for the unit. The `\renewunit` and `\provideunit` macros take the same arguments.

3.1415°
12 345XXX 67 890 XXX

```
\SI{3.1415}{\degree}\\
\newunit[valuesep=none]{\oddunit}{XXX}
\SI{12345}{\oddunit}
\SI[valuesep=thick]{67890}{\oddunit}
```

As with the LaTeX commands `\newcommand`, *etc.*, the choice of `\newunit`, `\renewunit` or `\provideunit` depends on the presence of an existing definition. While `\newunit` should be used when a unit has not been previously defined, `\renewunit` will issue a warning if the named unit does not already exist. `\provideunit` defines the unit if it does not exist, and otherwise does nothing at all. The same behaviour is seen with `\providepower` and `\provideprefix` (*vide infra*).

Output that is only valid in maths mode requires `\ensuremath`, text-only input requires `\text`. In the example below, `\mathnormal` is used to force the font choice only for the single character.[17]

$10\,\mathrm{m}\,\pi^{-2}$

```
\newunit{\SIpi}{\ensuremath{\mathnormal{\pi}}}
\SI{10}{\metre\per\SIpi\squared}
```

`\newpower`
`\renewpower`
`\providepower`

Powers are defined: `\newpower[post]{⟨power⟩}{⟨num⟩}`. Here, ⟨*power*⟩ is the name of the power macro and ⟨*num*⟩ is the (positive) number it represents. The later argument is always processed internally by `\num`, but *must* be a number. Giving the optional argument `post` indicates to the package that the power will come after the unit it applies to; by default it is assumed that it will come before.

kg^{4}
m^{4}

```
\newpower{\quartic}{4}
\newpower[post]{\totheforth}{4}\\
\si{\kilogram\totheforth}\\
\si{\quartic\metre}
```

`\newprefix`
`\renewprefix`
`\provideprefix`

The standard SI powers of ten are defined by the package, and are described above. However, the user can define new prefixes with `\newprefix`. This has syntax `\newunit[binary]{⟨prefix⟩}{⟨symbol⟩}{⟨powers-ten⟩}`, where ⟨*powers-ten*⟩ is the number of powers of ten the prefix represents. When the `binary` option is given, the prefix is a power of two. For example, `\kilo` and `\kibi` are defined:

```
\newprefix{\kilo}{k}{3}
\newprefix[binary]{\kibi}{Ki}{10}
```

`\newqualifier`
`\renewqualifier`
`\providequalifier`

It is possible to create unit "qualifiers", which add subscript descriptions to units. Although not encouraged, this is sometimes necessary to make the meaning clear. No qualifiers are pre-defined by siunitx, and so the user needs to declare them using `\newqualifier`. Currently, only subscript qualifiers can be created; a more extended set of options is planned for the next release of siunitx.

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

```
\newqualifier{\polymer}{pol}
\newqualifier{\catalyst}{cat}
\si{\kg\polymer\per\mole\catalyst\per\hour}
```

[16] Although the `\ang` macro is preferred for this job.
[17] The `\mathrm` font used for this document has an "ß" at the π position.

Table 12: Binary prefixes (`alsoload=binary`)

Prefix	Macro	Power	Symbol
kibi	`\kibi`	2^{10}	Ki
mebi	`\mebi`	2^{20}	Mi
gibi	`\gibi`	2^{30}	Gi
tebi	`\tebi`	2^{40}	Ti
pebi	`\pebi`	2^{50}	Pi
exbi	`\exbi`	2^{60}	Ei

8 Ranges of numbers

`\numrange` `\SIrange`

To aid the input of ranges of numbers, the macros `\numrange` and `\SIrange` are available. These both take an optional settings argument, then two numbers. The `\SIrange` macro then needs a unit. Depending on package settings, the phrase for the range and the repetition of units can be altered.

1 to 10
0.4–0.6
1.0 m to 1.4 m
12 Pa … 15 Pa
(44 to 45) kg

```
\numrange{1}{10}\\
\numrange[tophrase=dash]{0.4}{0.6}\\
\SIrange{1.0}{1.4}{\metre}\\
\SIrange[tophrase=dots]{12}{15}{\pascal}\\
\SIrange[repeatunits=false]
  {44}{45}{\kilo\gram}\\
```

9 Specialist units

In some subject area, there are units which are in common use even though they are outside of the SI system. Unlike the units discussed earlier, these specialist units are not loaded by default. In each case, they should be requested with the option `alsoload=`⟨*name*⟩.

9.1 Binary units (`binary`)

`\bit` `\byte`

The binary prefixes, `\bit`, `\byte` (Table 12) are not formally part of the SI system. They are available by giving the `alsoload=binary` option.

100 MiB

```
\SI{100}{\mebi\byte}
```

9.2 Synthetic chemistry (`synchem`)

`\mmHg` `\molar` `\Molar` `\torr` `\dalton`

The `synchem` file adds the common chemistry units `\mmHg`, `\molar`, `\Molar`, `\torr` and `\dalton` to siunitx. The `\Molar` macro is somewhat awkward, as it can be given as either "M" or "M". The later is obviously easily confused with the sign for the prefix mega. By default, siunitx uses the UK default of a small-caps symbol. The `\dalton` unit is defined here as this name is not recognised by the various international bodies: the symbol u is preferred.

Table 13: High-energy physics units (`alsoload=hep`)

Unit	Macro	Symbol	Abbreviation
Areas			
yoctobarn	`\yoctobarn`	yb	`\yb`
zeptobarn	`\zeptobarn`	zb	`\zb`
attobarn	`\attobarn`	ab	`\ab`
femtobarn	`\femtobarn`	fb	`\fb`
picobarn	`\picobarn`	pb	`\pb`
nanobarn	`\nanobarn`	nb	`\nb`
Other units			
micron	`\micron`	µm	
millirad	`\mrad`	mrad	
gauss	`\gauss`	G	

1 M HCl
760 Torr
0.01 mmHg
3.0 mol dm^{-3}
106.42 Da

```
\SI{1}{\Molar} HCl\\
\SI{760}{\torr}\\
\SI{0.01}{\mmHg}\\
\SI{3.0}{\molar}\\
\SI{106.42}{\dalton}
```

9.3 High-energy physics (hep)

In contrast to hepunits, siunitx does not define a long list of compound units for high-energy physics.[18] Instead, a small selection of new units are defined (Table 13). The mechanisms provided by siunitx should avoid the need for large numbers of abbreviations. For example, the hepunits `\MinveV` can be given as `\per\MeV` in siunitx, which requires only one more character.

`\clight` `\eVperc`

The `hep` option defines two units which are slightly unusual. `\clight` gives c, which is recognised as a unit when used in the appropriate circumstances. The second unit provided is `\eVperc`, which is commonly-used and clear enough for a compound definition. Notice that the value of `eVcorrb` will need to be adjusted when using this unit.

4.657 MeV/c^2

```
\SI[per=slash,eVcorrb=0.4ex]{4.657}{\mega\eVperc\squared}
```

9.4 Astronomy (astro)

`\parsec` `\lightyear`

For astronomers, the `\parsec` and `\lightyear` units are available, and give the obvious results.

12 pc
1 ly

```
\SI{12}{\parsec}\\
\SI{1}{\lightyear}
```

9.5 Geophysics (geophys)

`\gon`

For geophysics, the unit `\gon` is available.

[18] Using the `emulate=hepunits` option will load a file defining those.

12 gon

```
\SI{12}{\gon}
```

9.6 Chemical engineering (chemeng)

`\gmol` `\kgmol` `\lbmol`

For chemical engineers, the units `\gmol`, `\kgmol` and `\lbmol` can be loaded.

1 g-mol
5 kg-mol
10 lb-mol

```
\SI{1}{\gmol} \\
\SI{5}{\kgmol} \\
\SI{10}{\lbmol} \\
```

10 Font control

Following the lead of SIstyle, siunitx provides control over the font used to typeset output. By default, all text is typeset using the current upright serif maths font, whether the macros are given in text or maths mode. Some examples will show the effect.

10 10
20° 20°
30 kg
40 kg
50

```
\num{10} $\num{10}$\\
\sffamily \ang{20} $\ang{20}$\\
\textbf{\SI{30}{\kilo\gram}}\\
\boldmath $\SI{40}{\kilo\gram}$
\[ \num{50} \]
```

In contrast, by setting `obeyall`, the current font is used: this may be maths or text, depending on the context.

1°1′1″ 1°1′1″
2°2′2″ 2°2′2″
3°3′3″ 3°3′3″
4°4′4″ 4°4′4″
5°5′5″

```
\sisetup{obeyall}\\
\ang{1;1;1} $\ang{1;1;1}$\\
\sffamily \ang{2;2;2} $\ang{2;2;2}$\\
\textbf{\ang{3;3;3}} \boldmath $\ang{3;3;3}$\\
\emph{\ang{4;4;4}} \emph{$\ang{4;4;4}$}
\[ \ang{5;5;5} \]
```

Fine control of which elements of the local font are used is available with the `obeyfamily`, `obeybold`, `obeyitalic` and `obeymode` options.

1°1′1″ 1°1′1″
2°2′2″ 2°2′2″
3°3′3″ 3°3′3″
4°4′4″ 4°4′4″
5°5′5″

```
\sisetup{obeyfamily}\\
\ang{1;1;1} $\ang{1;1;1}$\\
\sffamily \ang{2;2;2} $\ang{2;2;2}$\\
\sisetup{obeybold}
\textbf{\ang{3;3;3}} \boldmath $\ang{3;3;3}$\\
\emph{\ang{4;4;4}} \emph{$\ang{4;4;4}$}
\[ \ang{5;5;5} \]
```

11 Package options

`\sisetup`

The "native" options for the package are all given using the key–value method. Most of the package options can be given both when loading the package and at any point in the document. This is achieved using the `\sisetup` macro.

The package options take a number of different forms.

- `option=⟨bool⟩` Simple true/false values. These macros all default to `option=true`, meaning that giving the option name along will set the appropriate flag.
- `option=⟨choice⟩` Take a single item from a pre-determined list. Depending on the value, one or more internal states will be altered. Values not on the list are ignored (with a warning).
- `option=⟨choice,literal⟩` If the given value is a ⟨*choice*⟩, then the internal settings for that choice are used. Any other value is used directly.
- `option=⟨literal⟩` The given value is used as a literal by the package.
- `option=⟨csname⟩` These options expect a command sequence as a value.
- `option=⟨length⟩` Requires a TEX length, for example `0.5ex`.
- `option=⟨list⟩` Takes a list of one or more items, which are not determined in advance.
- `option=⟨number⟩` Takes a number (possibly including an exponent part).

The package has a large range of options, to allow full control of the various features of the package. These control differing aspects of the package, and are given below in groups based on function. Where the key has a default value, it is given in bold.

11.1 Font family and style

The font used when typesetting material can be tightly controlled using siunitx. A number of options affect how the package matches the surrounding font, and the font families used to achieve this. The default is to use the current upright maths serif font with no variation.

`mode` `textmode` `valuemode` `unitmode` `obeymode`

The output of siunitx can occur using either text or maths mode. The package option `mode` determines which is used: valid options are **`maths`** and `text`.[19] The shortcut `textmode` is provided for setting `mode=text` quickly. Further refinement is possible using the `valuemode` and `unitmode` options. These apply to numbers (the output of `\num` and the first mandatory argument of `\SI`) and units (all other output), respectively. By setting the `obeymode` flag, the package will use the local typesetting mode (maths or text).

`obeyall` `obeyfamily` `obeybold` `obeyitalic`

The detection and matching of surrounding text can be controlled using a number of Boolean package options. `obeyall` turns on all of the detection. Thus output with `obeyall` in force will always match the local text appearance. `obeyfamily` instructs the package on detecting the surrounding font family (Roman, sans serif, fixed width), but does not detect bold or italic. `obeybold` detects the local bold setting, whilst `obeyitalic` picks up italic fonts.

Bold detection is influenced by the value of `inlinebold`, which takes values **`text`** and `maths`. The package can detect the local value of bold for either the surrounding text, or the surrounding inline (`$...$`) maths. The `obeyitalic` option does *not* have the same facility (maths is italic anyway).

`mathsrm` `mathssf` `mathstt` `textrm` `textsf` `texttt`

The font commands used by the package to achieve the above are all available

[19] Here and in all other cases, either UK or US spelling may be used. Thus `mode=maths` or `mode=math` have exactly the same effect.

for user modification. The options `mathsrm`, `mathssf` and `mathstt` hold the command sequences used in maths mode,[20] while `textrm`, `textsf` and `texttt` do the same for text mode. By default, these contain the obvious command names, for example `mathsrm=mathrm` and `texttt=ttfamily`. However, they can be set at will: the macro names indicate the nature of the surrounding text detected. For example, the value of `mathssf` is used in maths mode when the surrounding text is sans serif.

Each of the font options can be given separately for the contents of numbers and units. The option names include `value` or `unit` before the mode name. For example, the `mathsrm` option may be split into `valuemathsrm` and `unitmathsrm`.

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

Some text

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

More text

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

```
\sffamily
Some text
\sisetup{obeyall}
\[ x = \SI{1.2e3}{\kg\kelvin\candela} \]
More text
\sisetup{detectdisplay=false}
\[ y = \SI{3}{\metre\second\mole} \]
```

11.2 Spacing and separators

`unitsep` `valuesep` `digitsep` `anglesep` The separators between items can all be set using options taking a list of predefined items or a literal value. The "`sep`" options (`unitsep`, `valuesep`, `digitsep` and `anglesep`) all recognise `thin`, `medium`, `med`, `thick`, `space`, `cdot`, `times`, `tightcdot`, `tighttimes`, `fullstop`, `stop`, `period` and `none`. The named spaces are the normal maths separations, with `space` representing a full (non-breakable text) space, and with the obvious meanings for `cdot` and `times`. The `tight` variants reduce the spacing available. Three possible values are provided for "`.`", and `none` yields no space at all. In all cases, other values are treated literally and are typeset in maths mode. The default value is **`thin`** for all separations except `anglesep`, which is set to **`none`**.

`unitspace` `errspace` The `unitspace` and `errspace` options again take a list or literal value, but only the "real" spaces `thin`, `medium`, `med`, `thick`, `space` and `none` are recognised in the list. The `unitspace` option controls the output generated by an explicit space (`~`) inside a unit macro, while `errspace` is used to separate a bracketed error from the main number.

11.3 Number formatting

`numdigits` `numdecimal` `numdiv` `numexp` `numsign` `numaddn` `numgobble` There are two groups of options for formatting numbers. The first group all begin with "`num`", and take literal values used by the package to parse numbers. `numdigits` contains the valid number symbols (**`0123456789`**), with

[20] These can also be set using `mathrm`, `mathsf` and `mathtt`

numdecimal containing the decimal markers (.,). As in the numprint package, numexp (the list of exponent markers) recognises **deDE** as valid by default. numsign contains the sign markers for numbers (**+-\pm\mp**). numdiv holds the division marker in numbers (**/**). numaddn and numgobble both control which other characters do not give an error when present in a number. numaddn contains valid characters which should be included in the final output "as is", whereas numgobble lists the characters that are completely ignored. In all cases, the content of the options is a simple string, for example numdigits=1234567890.

decimalsymbol
digitsep
sepfour

The second group of number options control the output of numbers after parsing. The symbol used by siunitx as a decimal marker is set by the decimalsymbol option, which can take a list of choices or a literal. The valid choices here are **fullstop**, comma, cdot and tightcdot.[21] Notice that this does not have to agree with the input marker. The other separator for numerical output is the division of digits into groups of three. The result is dependent on two options. The previously-described digitsep option controls the spacing added between groups of three numbers. For numbers consisting of exactly four digits, the sepfour Boolean option controls whether separation occurs in these cases. The default is **false**.

1234
1 234

```
\num{1234}\\
\num[sepfour]{1234}
```

seperr
numopenerr
numcloseerr

For numbers given with an error [*e.g.* 1.23(4)], the package can separate out the error part, to give for example 1.23 ± 0.04. This behaviour is activated by the seperr option, and requires that numopenerr and numcloseerr contain the left- and right-hand delimiters for the error [defaults numopenerr=(and numcloseerr=)].

1.234 ± 0.005
$(1.234 \pm 0.005) \times 10^{6}$

```
\sisetup{seperr}\\
\num{1.234(5)}\\
\num{1.234(5)e6}
```

trapambigerr
openerr
closeerr
tightpm

If the number has an exponent, or if units are not repeated, then the result can be considered ambiguous. By default, the package adds the markers stored in openerr and closeerr to remove the ambiguity; the options have the same default values as the input error markers. Detection of a potentially-ambiguous error is controlled by the trapambigerr option, although for numbers with units the repeatunits option is also important. The spacing around the $\pm$ sign is normally set by TEX. However, using the tightpm option will cause this to be reduced to a minimum.

$1.234 \times 10^{6} \pm 0.005 \times 10^{6}$
1.234 m ± 0.005 m
(1.234 ± 0.005) m
1.234 ± 0.005 m
1.234±0.005

```
\sisetup{seperr}\\
\num[trapambigerr=false]{1.234(5)e6}\\
\SI{1.234(5)}{\metre}\\
\sisetup{repeatunits=false}
\SI{1.234(5)}{\metre}\\
\SI[trapambigerr=false]{1.234(5)}{\metre}\\
\num[tightpm]{1.234(5)}
```

numprod
repeatunits

The number processor can recognise products in the numerical input; the

[21] fullstop also has aliases stop and period.

symbols used for products are stored in `numprod`, with the default value of "`x`". By default, the `\SI` macro will repeat the units for a number given in this way. This behaviour is altered by the `repeatunits` option, which takes the values **`true`**, `false` and `power`. The later applies only when providing multiplied numbers with units, and converts the unit to the appropriate power rather than repeating the units.[22] Notice that when applied to errors, `repeatunits` takes priority over `trapambigerr`.[23]

$1 \times 2 \times 3$
$4\,\mathrm{m} \times 5\,\mathrm{m} \times 6\,\mathrm{m}$
$1.2 \times 3.4 \times 5.6\,\mathrm{mm}^3$
$(1.234 \pm 0.005)\,\mathrm{m}$
$9.109\,389\,7 \pm 0.000\,005\,4 \times 10^{-31}\,\mathrm{kg}$
$9.109\,389\,7 \times 10^{-31}\,\mathrm{kg} \pm 0.000\,005\,4 \times 10^{-31}\,\mathrm{kg}$
$1 \times 2 \times 3\,\mathrm{m}^3$

```
\num{1 x 2 x 3}\\
\SI{4 x 5 x 6}{\metre}\\
\sisetup{repeatunits=false}
\SI{1.2 x 3.4 x 5.6}{\milli\metre\cubed}\\
\SI[seperr]{1.234(5)}{\metre}\\
\SI[seperr,
  trapambigerr=false]
  {9.1093897(54)e-31}{\kilo\gram}\\
\SI[seperr,repeatunits,
  trapambigerr=false]
  {9.1093897(54)e-31}{\kilo\gram}\\
\SI[repeatunits=power]{1 x 2 x 3}{\metre}
```

`expproduct` `expbase` `allowzeroexp`

The formatting of exponents is controlled by `expproduct` and `expbase`. `expproduct` sets the symbol used to indicate a product for exponents (*e.g.* the $\times$ in 2×10^2), while the value of `expbase` sets the power used (the 10 in the example). Both options accept a very short list of options: **`times`**, `tighttimes`, `cdot` and `tightcdot` for the product, and **`ten`** and `two` for the power.[24] Other choices are used literally. Also relevant to exponent processing is the `allowzeroexp` option. By default, the package will suppress a zero exponent, but setting the flag will allow the output of 10^0.

`addsign` `sign` `retainplus`

Additions to the input can take the form of implicit signs and padded zeros. The `addsign` option takes a list of potential sites to add a sign: **`none`**, `mantissa`, `exponent` and `both`.[25] If no sign is given in the input, the setting here determines if one is added. The sign to add is stored in `sign`, which takes the list of choices **`plus`**, `minus`, `pm` and `mp`, or uses the input literally (in maths mode). For positive numbers, the `retainplus` option causes a $+$ sign explicitly in the input to be retained. By default, the package will remove such signs.

`padnumber`

The `padnumber` option controls the addition of zeros to the input, to "pad" the result. The option takes a list of choices: **`leading`**, `trailing`, `both` and `none`.[26] No additional precision is added by this option; integer input will not add a decimal point.

[22] This is a very simply option: do not expect it to work with anything except areas and volumes.
[23] `repeatunits` also applies to the range macro `\SIrange`.
[24] The `tighttimes` and `tightcdot` options give the rather questionable results: 1×10^2 and $1\cdot10^2$, as opposed to 1×10^2 and $1 \cdot 10^2$.
[25] Aliases are provided: `mant` = `mantissa`, `exp` = `exponent`, `all` = `true` = `both`, `false` = `none`.
[26] Aliases: `all` = `true` = `both`, `false` = `none`.

0.1
2
3.0
4.0
0.5
6
7
.8

```
\num[padnumber=leading]{.1}\\
\num[padnumber=leading]{2.}\\
\num[padnumber=trailing]{3.}\\
\num[padnumber=both]{4.}\\
\num[padnumber=both]{.5}\\
\num[padnumber=both]{6}\\
\num[padnumber=none]{7.}\\
\num[padnumber=none]{.8}
```

`fixdp` `dp`

In contrast to the `padnumber` option, the package can alter the precision of the input number if the `fixdp` option is set. The `fixdp` option will fix the decimal places of the output to the number stored in the `dp` option. The later should be a positive integer or zero.

1.000
53.900
4.568
−1.294
−1.296
10.000
2.126×10^{90}
12

```
\sisetup{fixdp,dp=3}
\num{1}\\
\num{53.9}\\
\num{4.56783}\\
\num{-1.2942}\\
\num{-1.2959}\\
\num{9.9999}\\
\num{2.1264e90}\\
\num[dp=0]{12.345}
```

`fixsf` `sf`

In contrast to the `dp` option, the `fixsf` and `sf` options control the number of significant figures in the output. This never adds extra digits to the output, as significant figures are concerned with accuracy.

1
53.9
4.57
−1.29
−1.30
10.0
2.13×10^{90}
773 000

```
\sisetup{fixsf,sf=3}
\num{1}\\
\num{53.9}\\
\num{4.56783}\\
\num{-1.2942}\\
\num{-1.2959}\\
\num{9.9999}\\
\num{2.1264e90}\\
\num{773322}
```

`tophrase`

When using the range macros `\numrange` and `\SIrange`, the phrase used for "to" is set using the option `tophrase`. This can take three pre-defined choices **`to`**, `dots` and `dash`, or arbitrary input in text mode.

1 to 10
1 . . . 10
1–10
1 bis 10

```
\numrange{1}{10}\\
\numrange[tophrase=dots]{1}{10}\\
\numrange[tophrase=dash]{1}{10}\\
\numrange[tophrase={{ bis }}]{1}{10}\\
```

Notice that any space required must be included in the phrase itself, and that this may require a second set of braces.

`trapambigrange` `openrange` `closerange`

When `repeatunits` is false, the `trapambigrange` option controls whether brackets are added to keep unit–value agreement. As with other options for trapping ambiguous output, the bracket symbols used here are stored as `openrange` and `closerange`.

1 kg to 10 kg
(1 to 10) kg
1 to 10 kg

```
\SIrange{1}{10}{\kg}\\
\SIrange[repeatunits=false]{1}{10}{\kg}\\
\SIrange[repeatunits=false,trapambigrange=false]
  {1}{10}{\kg}
```

Notice that any space required must be included in the phrase itself, and that this may require a second set of braces.

11.4 Angle formatting

padangle
strictarc

The angle formatter uses `\num` to format numbers: any options for numbers are therefore applicable here. The `padangle` option takes choices **`small`**, `large`, `all` and `none`, and controls how angles are padded when given in degrees, minutes and seconds.[27] When giving angles as arcs (in degrees, minutes and seconds), the package can detect if the correct number of semi-colons have been given. This is controlled by the `strictarc` option, which is a Boolean switch with a **`true`** default. With `strictarc` set to `false`, an incomplete arc is interpreted as degrees and minute, while an over-complete one will drop excess input.

1°
2°0′0″
3°
0°4′0″
5′5″
1°2′
1°2′3″

```
\ang[padangle=none]{1;;}\\
\ang[padangle=large]{2;;}\\
\ang[padangle=small]{3;;}\\
\ang[padangle=both]{;4;}\\
\ang[padangle=none]{;5;5}\\
\ang[strictarc=false]{1;2}\\
\ang[strictarc=false]{1;2;3;4;}
```

angformat

The angle formatting system can convert between decimal angles and those given as degrees, minutes and seconds. This is controlled by the `angformat` option, which takes choices **`unchanged`**, `decimal` and `arc`.[28] When set to `unchanged`, nothing is done to the input. The conversion is based on TEX dimensions, and is therefore limited in accuracy. For this reason, the output is automatically rounded: output as a decimal angle is limited to three places, and that as an arc is given to a single decimal place for the seconds component.

1°2′3″ = 1.034°
4.56° = 4°33′36.0″

```
$\ang{1;2;3} = \ang[angformat=dec]{1;2;3}$\\
$\ang{4.56} = \ang[angformat=arc]{4.56}$
```

astroang

For astronomers, the `astroang` option is provided. This moves the degrees, minutes or seconds symbol (as appropriate) over the decimal marker rather than after the number.

1°2′3.4″
5°6′7.″8

```
\ang{1;2;3.4}\\
\ang[astroang]{5;6;7.8}
```

11.5 Tabular material

tabnumalign

Material typeset in `S` columns is processed internally by the `\num` macro. Thus, as with angles, the number options also apply here. The positioning of tabular material is controlled by the two options `tabnumalign` and `tabformat`.

[27] Aliases: `all` = `true` = `both`, `false` = `none`.
[28] Aliases: `decimal` = `dec`, `arc` = `dms`, `unchanged` = `none`.

Table 14: The tabalignexp option

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

tabnumalign takes values **centredecimal**, centre, left and right.[29] When using centredecimal, the package places the decimal marker of the mantissa at the centre of the column, which then grows to accommodate the widest number given. For equal numbers of digits before and after the decimal sign, this is the easiest option. The other choices use a fixed-width box to store the number; the box is then aligned with the edges of the column.

tabformat The tabformat option sets the amount of space reserved by siunitx for the alignment box when not using the centredecimal setting of tabnumalign. The numerical parts of tabformat are interpreted as ⟨*pre*⟩⟨*dec*⟩⟨*post*⟩; ⟨*pre*⟩ and ⟨*post*⟩ are the number of digits before and after the decimal sign, respectively. Both signs and exponents can be included in tabformat, resulting in appropriate space being reserved. The entire tabformat input is processed using the \num macro internally. Thus the decimal and exponent signs used in tabformat are checked against numdecimal and numexp, respectively.

tabalignexp tabexpalign When tabformat contains exponents, two possibilities are available for alignment. The first method is to place the exponent parts so that the "×10" parts form a column, with whitespace after shorter mantissa components. In the second method, no additional space is added after the mantissa, and the exponents do no line up (Table 14). This is controlled by the tabalignexp option, which can be set to **true** or false. The alias tabexpalign is available for this option.

```
\begin{table}
  \centering
  \caption{The \opt{tabalignexp} option}
  \label{tab:alignexp}
  \sisetup{tabformat=1.3e2,tabnumalign=centre}
  \begin{tabular}{SS[tabalignexp=false]}
    \toprule
    {Header} & {Header} \\
    \midrule
    1.2e3    & 1.2e3    \\
    1.234e56 & 1.234e56 \\
    \bottomrule
  \end{tabular}
\end{table}
```

tabtextalign tabunitalign tabalign Cells containing no numbers are handled by siunitx in a manner similar to \multicolumn. The setting of tabtextalign is taken from the list **centre**, right and left.[30] As would be expected, these settings centre, right- or left-align the cell contents. In s columns, all content is treated as input to the \si

[29] Aliases centerdecimal = centredecimal; center = centre.
[30] Alias centre = center.

Table 15: The tabautofit option

Header	Header	Header
1.2	1.200	1.2
1.2345	1.235	1.2345

macro. The alignment of the contents relative to the cell is controlled by the tabunitalign option, which takes options **left**, right and centre. The settings for tabnumalign, tabtextalign and tabunitalign can be set to the same value in one go with the tabalign option.

tabautofit The contents of table cells can automatically be rounded or zero-filled to the number of decimal places given in tabformat. This is activated by the tabautofit Boolean option. As tabformat does not apply to columns with alignment centredecimal, tabautofit is also inactive for these columns (Table 15).

```
\begin{table}
  \centering
  \caption{The \opt{tabautofit} option}
  \label{tab:autofit}
  \sisetup{tabformat=1.3,tabnumalign=centre}
  % Notice the overfull hbox which results with
  % the first column
  \begin{tabular}{%
    S%
    S[tabautofit]%
    S[tabautofit,tabnumalign=centredecimal]}
    \toprule
    {Header} & {Header} & {Header} \\
    \midrule
    1.2    & 1.2    & 1.2    \\
    1.2345 & 1.2345 & 1.2345 \\
    \bottomrule
  \end{tabular}
\end{table}
```

tabparseonly It is possible to turn off the automatic alignment of numbers while retaining the ability to parse the S column input. Setting tabparseonly to true will still parse the column contents, but will not align the decimal markers. The alignment of the cell is governed by tabnumalign (Table 16).

```
\begin{table}
  \centering
  \caption{The \opt{tabparseonly} option}
  \label{tab:parse}
  \begin{tabular}{%
    S[tabparseonly]%
    S[tabparseonly,tabnumalign=left]%
    S[tabparseonly,tabnumalign=right]}
    \toprule
    {Header} & {Header} & {Header} \\
```

Table 16: The tabparseonly option

Header	Header	Header
14.2	14.2	14.2
1.234 56	1.234 56	1.234 56
1.2×10^3	1.2×10^3	1.2×10^3

```
    \midrule
    14.2    & 14.2    & 14.2    \\
    1.23456 & 1.23456 & 1.23456 \\
    1.2e3   & 1.2e3   & 1.2e3   \\
    \bottomrule
  \end{tabular}
\end{table}
```

11.6 Units

per

Most of the unit options are concerned with the processing of named units. The processor for units given as macro names can be influenced to give a variety of output formats. The per option defines how the keyword macro \per is handled. This option takes a choice from the list **reciprocal**, slash and fraction.[31] The default option uses \per to indicate reciprocal powers, whereas slash causes the package to use "/" to show division.

fraction
slash

The fraction option defines how per=fraction is interpreted. The list of applicable values here is **frac**, nice, ugly and sfrac. In each case, the unit is typeset as a fraction, but the macro use to achieve this varies. frac uses the TEX \frac, macro, while nice makes use of a \nicefrac-like method. The ugly option uses a slash in text mode and \frac in maths mode.[32] Finally, the setting fraction=sfrac uses the \sfrac macro from the xfrac package, when available.[33] The slash option sets the symbol used when per=slash is in force. This recognises the single keyword slash; anything else is used literally.

$\frac{\mathrm{m}}{\mathrm{s}}$
m/s
m/s
m/s

```
\sisetup{per=fraction}
\si[fraction=frac]{\metre\per\second}\\
\si[fraction=nice]{\metre\per\second}\\
\si[fraction=sfrac]{\metre\per\second}\\
\si[per=slash]{\metre\per\second}
```

stickyper

By default, \per applies only to the next unit given.[34] By setting the stickyper flag, this behaviour is changed so that \per applies to all subsequent units.[35]

[31] Aliases reciprocal = rp = power, fraction = frac.

[32] Similar to the ugly option of the nicefrac package.

[33] xfrac is part of the experimental system for LATEX3. As it requires a number of additional packages to work, siunitx does not load xfrac. If it is unavailable, the sfrac setting will fall back to using \nicefrac. See the xfrac documentation for reasons to prefer \sfrac to \nicefrac.

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

[35] This is the behaviour in Slunits.

$\mathrm{kg\,m^{-1}\,A}$
$\mathrm{kg\,m^{-1}\,A^{-1}}$

```
\si{\kilogram\per\metre\ampere}\\
\si[stickyper]{\kilogram\per\metre\ampere}
```

`trapambigfrac` `openfrac` `closefrac`

When using `per=slash`, multiple units in the denominator will yield a potentially ambiguous result. The `trapambigfrac` determines whether the package checks for this: this takes **`true`** and `false`. When set, the contents of `openfrac` are inserted before the denominator, and `closefrac` is inserted after.

$\mathrm{m/(s\,kg)}$
$\mathrm{m/s\,kg}$

```
\sisetup{trapambigfrac,openfrac=(,closefrac=),per=slash}\\
\si{\metre\per\second\per\kilogram}\\
\si[trapambigfrac=false]{\metre\per\second\per\kilogram}
```

`prefixsymbolic` `prefixbase` `prefixproduct`

The unit prefixes (`\kilo`, *etc.*) are normally given as letters. However, the package can convert these into numerical powers.[36] This is controlled by the `prefixsymbolic` Boolean option, which by default is **`true`**. If `prefixsymbolic` is set to `false`, the format of the prefix is controlled by `prefixbase` and `prefixproduct`, which work in the same way as `expbase` and `expproduct`.

`prespace` `xspace` `allowoptarg`

By default, the single unit macros (*e.g.* `\metre`) add no space either before or after the unit. Setting the `xspace` flag to true means that the single macros are followed by the `\xspace` command (when used outside of `\SI`/`\si`). For users of unitsdef, the `prespace` macro changes the behaviour of the unit macros, so that they can immediately follow a number. As a result, the unit macros will *always* be preceded by a fixed space when the `prespace` flag is true: this will be in addition to any other space. Also relevant to users moving from unitsdef is the `allowoptarg` option. This allows single unit macros to take an optional numerical argument, in the same way that occurs in that package.

mis the symbol for metres
No, m is the correct symbol
30 m
Do not use m in running text
40 m

```
\metre is the symbol for metres\\
\sisetup{xspace}
No, \metre is the correct symbol\\
\sisetup{prespace}
30\metre\\
Do not use \metre in running text\\
\sisetup{allowoptarg}
\metre[40]
```

11.7 Symbols

User access to control the symbols used for Ω, µ, °, ′, ″, Å and ℃ is provided here. These are all literal options, which are available in text and maths mode variants. For example, `textmicro` is the code used for the µ symbol in text mode. The text mode macros should be safe when forced into text, and the maths ones when forced into maths. The symbols defined in this way are:

- `textOmega`;
- `mathsOmega`;
- `textmu`;

[36] Provided things are not too complex!

- mathsmu;
- textdegree;
- mathsdegree;
- textminute;
- mathsminute;
- textsecond;
- mathssecond;
- textringA;
- mathsringA;
- textcelsius;
- mathscelsius.

redefsymbols When siunitx is loaded, it can check for the presence of the textcomp and upgreek packages, to provide better symbols for certain items. To prevent this, set the redefsymbols option false (the default is **true**).

eVcorra eVcorrb The eV symbol requires some fine-tuning, and so has two options of its own, both TEX lengths. eVcorra is the correction applied to the gap between "e" and "V" of the unit: the default is **0.3ex**. eVcorrb is the correction applied to the gap between "V" of the unit and whatever follows; the default is **0ex**. The optimal value for these options will depend on the current font settings.[37]

eV/m
eV/m

```
\si[per=slash]{\electronvolt\per\metre}\\
\si[per=slash,eVcorrb=0.7ex]{\electronvolt\per\metre}
```

11.8 Colour

colourall colourunits colourvalues The package provides internal hooks for applying colour to part or all of the output. This requires the user to load the color or xcolor package to support colour in the output; siunitx will ignore a colour request if support is unavailable. The Boolean options colourall, colourunits and colourvalues are used to turn application of a given colour on or off for all output, only units and only values, respectively. All three switches are available with US spelling, *e.g.* colorall and colourall behave in the same way. With colour turned off, no \color command is issued internally, and output follows the surrounding text.

colour unitcolour valuecolour The colour names to use for colouring output are set by the colour, unitcolour and valuecolour options (all also available with US spelling). The colour option internally sets both unitcolour and valuecolour.

Text 50
10 m
Text 70
40 s

```
{\color{brown} Text \num{50}}\\
\sisetup{colourall,colour=green}
\SI{10}{\metre}\\
{\color{purple} Text \num{70}}\\
\sisetup{colourunits=true,colourvalues=false}
\SI{40}{\second}
```

colourneg
negcolour

siunitx can automatically add a colour to negative numbers. This is turned on using the `colourneg` switch. The colour used is set by the `negcolour` option; both options are available using US spellings.

11.9 International support

locale

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

1.234 m
6,789 m

```
\SI{1.234}{\metre}\\
\SI[locale=DE]{6.789}{\metre}
```

loctolang

Locales are distinct from babel languages, as typographic conventions are not tightly integrated with language. However, it is useful to be able to associate a particular locale with a babel language. The option `loctolang` handles this, and expects pairs of values: `loctolang=`⟨*locale*⟩`:`⟨*language*⟩.

$6.022 \times 10^{23}\,\mathrm{mol}^{-1}$

$6{,}022 \cdot 10^{23}\,\mathrm{mol}^{-1}$

```
\sisetup{loctolang={UK:UKenglish,DE:german}}\\
\SI{6.022e23}{\per\mole}\\
\selectlanguage{german}\\
\SI{6.022e23}{\per\mole}
```

11.10 Package control

load
noload
alsoload

The package keeps most of the unit and abbreviations definitions in files separate from `siunitx.sty`. To control what is loaded, three complementary options are provided, all of which take a list of one or more choices. `load` and `alsoload` define which support configuration files are loaded. The list in `load` recognises the value `default`, which is expanded to the normal list before loading. The difference between `load` and `alsoload` is that `load` specifies the *complete* list of files to load, whereas `alsoload` adds to the existing list. To use the `load` option successfully requires knowing everything that is needed. The `noload` option can be used to delete one or more items from the `load` list, without needing to know what is on it.[38]

log
debug

To control data written to the `.log` file, the `log` option is provided. This takes a value from the list **`normal`**, `none`, `minimal`, `errors` and `debug`. As would be expected, these indicate the amount of detail written to the log file. As a shortcut to `log=debug`, the package also recognises the `debug` option directly.

strict

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

[37] This document uses `eVcorra=0.1ex`.

[38] `noload` does not prevent the loading of a file needed by one which is loaded. Thus the package may internally override a `noload` value if needed.

11.11 Back-compatibility options

`emulate` The package can emulate SIunits, SIstyle, unitsdef, units, hepunits, fancyunits and fancynum. Giving the `emulate=`⟨*package*⟩ option will give the desired emulation, and combinations which would be possible with the real packages will also work here. The package will recognise the options of the emulated packages. This will automatically cause emulation to be switched on.

11.12 Summary of all options

Table 17 lists a summary of the package options (excluding those for backward-compatibility). A reminder or the input format is also provided.

Table 17: All package options

Option	Type	Description
addsign	List	Add sign to number
allowlitunits	Boolean	Allow literal input of units
allowoptarg	Boolean	Allow optional argument to unit macros
allowzeroexp	Boolean	Allow 10^0
angformat	List	Conversion of angle format
anglesep	List or literal	Space between angle components
astroang	Boolean	Astronomy-style angles
closeerr	Literal	Closes potential-ambiguous error
closefrac	Literal	Closes potential-ambiguous fraction
closerange	Literal	Closes potential-ambiguous range
color	Literal	Colour used for units and values
colour	Literal	Colour used for units and values
colorall	Boolean	Switch for colouring all output
colourall	Boolean	Switch for colouring all output
colorneg	Boolean	Colour negative numbers
colourneg	Boolean	Colour negative numbers
colorunits	Boolean	Switch for colouring units
colourunits	Boolean	Switch for colouring units
colorvalues	Boolean	Switch for colouring values
colourvalues	Boolean	Switch for colouring values
decimalsymbol	List or literal	Decimal symbol
debug	Boolean	Write debugging data to log
detectdisplay	Boolean	Treat display maths separately
digitsep	List	Separation of digits in large numbers
dp	Integer	Number of decimal places to output numbers to
emulate	Modules	Emulation modules to load
errspace	List	Spacing of bracketed error
eVcorra	Length	Spacing correction in eV
eVcorrb	Length	Spacing correction after eV
expbase	List or literal	Base used for exponents
expproduct	List or literal	Product sign for exponents
fixdp	Boolean	Switch for fixing decimal

Continued on next page

Option	Type	Description
		places of numbers
fixsf	Boolean	Switch for fixing significant figures of numbers
fraction	List	Method used when setting per=frac
inlinebold	List	Select how inline bold is tested
load	Modules	Modules to load
locale	Modules	Locale to follow
loctolang	Special	Associate locale with babel language
log	List	Amount of data added to log
mathOmega	Literal	"Ω" symbol in maths mode
mathcelsius	Literal	"℃" symbol in maths mode
mathdegree	Literal	"°" symbol in maths mode
mathminute	Literal	"′" symbol in maths mode
mathmu	Literal	"μ" symbol in maths mode
mathringA	Literal	"Å" symbol in maths mode
mathrm	Csname	Roman maths font
mathsOmega	Literal	"Ω" symbol in maths mode
mathscelsius	Literal	"℃" symbol in maths mode
mathsdegree	Literal	"°" symbol in maths mode
mathsecond	Literal	"″" symbol in maths mode
mathsf	Csname	Sans serif maths font
mathsminute	Literal	"′" symbol in maths mode
mathsmu	Literal	"μ" symbol in maths mode
mathsringA	Literal	"Å" symbol in maths mode
mathsrm	Csname	Roman maths font
mathssecond	Literal	"″" symbol in maths mode
mathssf	Csname	Sans serif maths font
mathstt	Csname	Fixed-width maths font
mathtt	Csname	Fixed-width maths font
mode	List	Use text or maths mode for typesetting
negcolor	Literal	Colour used for negative numbers
negcolour	Literal	Colour used for negative numbers
noload	Modules	Modules not to load
numaddn	Literal	Additional input allowed in numbers
numcloseerr	Literal	Character indicating end of numerical error
numdecimal	Literal	Decimal symbols in numbers
numdigits	Literal	Digit characters in numbers
numdiv	Literal	Division characters in numbers
numexp	Literal	Exponent characters in numbers
numgobble	Literal	Characters to ignore in numbers
numopenerr	Literal	Character indicating start of numerical error
numprod	Literal	Characters used for a product
numsign	Literal	Sign characters in numbers
obeyall	Boolean	Combination of obeybold,

Continued on next page

Option	Type	Description
		obeyitalic and obeymode
obeybold	Boolean	Check local bold setting
obeyitalic	Boolean	Check local italic setting
obeymode	Boolean	Check local mode (text/maths)
openerr	Literal	Opens potentially-ambiguous error
openfrac	Literal	Opens potentially-ambiguous fraction
openrange	Literal	Opens potentially-ambiguous range
padangle	List	Add zeros to blank parts of angles
padnumber	List	Add zeros to blank parts of numbers
per	List	Behaviour of \per
prefixbase	List or literal	Base used when making prefixes numerical
prefixproduct	List or literal	Product sign for prefixes
prefixsymbolic	Boolean	Behaviour of unit prefixes
prespace	Boolean	Add space before units
redefsymbols	Boolean	Use better symbols if available
repeatunits	List	Repeat units with separated errors, products and in ranges
retainplus	Boolean	Retain explicit plus sign
seperr	Boolean	Separate number and error
sepfour	Boolean	Separate four-digit numbers
sf	Integer	Number of significant figures to output numbers to
sign	List or literal	Sign to add to numbers
slash	List or literal	Symbol used for "/"
stickyper	Boolean	Require \per only once
strict	Boolean	Obey the rules strictly
strictarc	Boolean	Require exactly zero or two semi-colons
tabalign	List	Positioning of all column data
tabalignexp	Boolean	Alignment of exponents in tables
tabautofit	Boolean	Switch for rounding numbers to length given by tabformat
tabformat	Number	Space reserved in table for numbers
tabnumalign	List	Alignment of S column numbers
tabparseonly	Boolean	Do not align S columns on decimal marker
tabtextalign	List	Positioning of text in S columns
tabunitalign	List	Positioning of units in s columns
textcelsius	Literal	"°C" symbol in text mode
textdegree	Literal	"°" symbol in text mode
textminute	Literal	"′" symbol in text mode
textmu	Literal	"μ" symbol in text mode
textomega	Literal	"Ω" symbol in text mode
textringa	Literal	"Å" symbol in maths mode
textrm	Csname	Roman text font
textsecond	Literal	"″" symbol in text mode

Continued on next page

Option	Type	Description
textsf	Csname	Sans serif text font
texttt	Csname	Fixed-width text font
tightpm	Boolean	Reduce space around $\pm$
tophrase	List or literal	The phrase "to" for ranges
trapambigerr	Boolean	Check for ambiguous errors
trapambigfrac	Boolean	Check for ambiguous fractions
trapambigrange	Boolean	Check for ambiguous ranges
unitcolor	Literal	Switch for colouring units
unitcolour	Literal	Switch for colouring units
unitmathrm	Csname	Roman maths font for units
unitmathsf	Csname	Sans serif maths font for units
unitmathsrm	Csname	Roman maths font for units
unitmathssf	Csname	Sans serif maths font for units
unitmathstt	Csname	Fixed-width maths font for units
unitmathtt	Csname	Fixed-width maths font for units
unitmode	List	As mode, for units only
unitsep	List or literal	Separator for units
unitspace	List or literal	Space used for "~" in units
valuecolor	Literal	Switch for colouring value
valuecolour	Literal	Switch for colouring value
valuemathrm	Csname	Roman maths font for values
valuemathsf	Csname	Sans serif maths font for values
valuemathsrm	Csname	Roman maths font for values
valuemathssf	Csname	Sans serif maths font for values
valuemathstt	Csname	Fixed-width maths font for values
valuemathtt	Csname	Fixed-width maths font for values
valuemode	List	As mode, for values only
valuesep	List or literal	Separator between value and unit
xspace	Boolean	Use xspace after units

12 Emulation of other packages

siunitx has been designed as a replacement for SIunits, SIstyle, unitsdef, units, hepunits, fancyunits and fancynum. It therefore provides options reproduce the functions of all of these packages. In this way, siunitx should be usable as a straight replacement for the older packages.[39] This means for example that the \num macro takes an optional star when emulating SIstyle. However, there are some points that should be remembered. In particular, siunitx validates numerical input, meaning that places where a number is expected in the older packages *require* a number when emulated by siunitx.

The numprint package has provided many useful ideas for the code used here for number formatting. The basic use of the \numprint (or \np) macro can be reproduced using siunitx. However, numprint is large and complex, with its own backward-compatibility options. As a result, emulation of numprint is not

[39] User macros means that they are described in the package documentation; simply not containing an @ does not mean they will have been emulated.

provided here. To use a numprint document with siunitx, the `\numprint` macro could be provided using the following code.

```
\newcommand*{\numprint}[2][]{\SI[obeymode]{#2}{#1}}\\
\numprint{-123456} \\
\numprint[N/mm^2]{-123456}
```

−123 456
−123 456 N/mm²

siunitx can be used more-or-less directly to replace both dcolumn and rccol. As is explained in the code section, much of the column-alignment system here is taken from dcolumn, while rccol provided a model for a customisable system. However, neither package has been directly emulated here. The `S` column type can be used to replace both `D` and `R` columns by setting the appropriate package options.

13 Configuration files

siunitx is a modular package. The unit definitions, abbreviations and locales are all stored in configuration files. These all take names of the form `si-`⟨*name*⟩`.cfg`, where ⟨*name*⟩ is the part of the filename used as an option in `\sisetup` or when loading siunitx. Producing new configuration files therefore consists of making a suitably-named file and adding it to the path searched by TEX. The files should normally consist of settings (in `\sisetup`) and unit definitions, *etc*.

`\addtolocale` To allow arbitrary macros to be stored in locales, the `\addtolocale` macro is provided. This ensures that arbitrary text is only executed when using a locale, not when loading it. The macro takes two arguments, `{`⟨*locale*⟩`}` and `{`⟨*code*⟩`}`.

```
\addtolocale{DE}{TEST}
Some text as filler \\
\sisetup{locale=DE}
This example is rather trivial!
```

Some text as filler
TEST This example is rather trivial!

`\requiresiconfigs` To load one or more configuration files from inside another configuration file, the `\requiresiconfigs` macro is provided. This accepts a comma-separated list of configuration names, in the same way as `load` or `noload`.

In addition to the various configuration files provided with the package, a local file `siunitx.cfg` may be provided. This is read at the end of loading siunitx, and allows the user to include any local definitions or settings easily.

14 Common questions

14.1 Why do I need `\per` more than once?

The unit engine of siunitx is based around the English method for reading units out loud. Thus $\mathrm{J\,mol^{-1}\,K^{-1}}$ is pronounced "joules per mole per kelvin". Hence by default you need to put `\per` before each item in the denominator of a unit. The behaviour can be altered by setting the `stickyper` option.

14.2 Why is the order of my units changed?

Then using `per=reciprocal`, the units are typeset as given. However, when using `per=slash` or `per=fraction`, the package needs to find which ones are

in the denominator. It then prints the numerator and denominator separately. So if you give a unit in the denominator *before* one in the numerator, they have to be re-ordered.[40]

$88\,\mathrm{kg^{-1}\,m}$
$66\,\mathrm{m/kg}$

```
\SI{88}{\per\kilogram\metre} \\
\SI[per=slash]{66}{\per\kilogram\metre}
```

14.3 Why are compound units not recommended outside of `\SI/\si`?

To fully process units made up of several parts, the processor has to know where the end of the unit is. When the unit macros are used outside of `\SI/\si`, this is not the case. The package therefore does its best, but results may be sub-optimal. To get consistent results, either define a new *single* unit, or keep compound units inside `\SI` and `\si`.

$\mathrm{m/\mu s}$
$\mathrm{m\,\mu s^{-1}}$
$\mathrm{m\,\mu s^{-1}}$
$\mathrm{kg\,m\,\mu s^{-1}}$
$\mathrm{kgm\,\mu s^{-1}}$

```
\metre\per\micro\second \\
\si{\metre\per\micro\second} \\
\newunit{\myunit}{\metre\per\micro\second}
\myunit
\newunit{\myunittwo}{\kilogram\myunit} \\
\myunittwo \\
\kilogram\myunit
```

Notice the difference in behaviour of `\per` in the first two lines, and the spacing error on the last line in the example.

14.4 How do I set superscripts to use lining numbers?

Lowercase ("old style") numbers are favoured by many people for use in running text. However, this does not necessarily look good in superscripts. The mode used to typeset data can be varied, so that maths mode numbers are used for the unit part of the output.

$1234\,\mathrm{m\,s^{-1}}$
$1234\,\mathrm{m\,s^{-1}}$

```
\SI[mode=text]{1234}{\metre\per\second}\\
\SI[valuemode=text,unitmode=maths]{1234}{\metre\per\second}
```

14.5 Why do most of the examples use $\mathrm{J\,mol^{-1}\,K^{-1}}$?

The package author is a chemist, and this is the unit of entropy (disorder). It nicely demonstrates the use of the `\per` macro, and so it crops up a lot. It is also in the subsection heading here to act as a test with hyperref and moving arguments!

14.6 What can numprint do that siunitx cannot?

siunitx uses a lot of ideas from numprint: a reasonable amount of the number-processing code here is based on that in numprint. However, the two packages have somewhat different aims, and as a result there are things that numprint can do that siunitx does not implement. The main features of numprint not available here are:

[40] "Double division" ($1\,\mathrm{s/m/kg}$) is mathematically incorrect.

- General support for numbers with base other than 10 (see nbaseprt);
- Alignment of the decimal marker of powers in tables;
- Alignment of numbers in running text;
- Specific formatting commands for TEX counters and lengths.

15 Tricks and known issues

15.1 Ensuring maths mode

Due to the possibility of output in either maths or text mode, any input which requires a particular mode needs to be protected. You cannot use `$...$`, as this can get "caught out", but also as it may give hard-to-follow errors. Always use `\ensuremath` to force maths processing, and `\text` (from the $\mathcal{AMS}$TEX bundle) to ensure text mode.

15.2 Using . and fixed spaces in units

To use a literal . in a unit, it has to be within an extra set of braces. This does not need any extra protection, unlike the situation with SIstyle (for example, no `\text` macro is needed). The fixed space (~) is more problematic: set `unitspace=space` to get a full space here.

```
\newunit[unitspace=space]{\myunit}{V~vs{.}~NHE}%
\SI{10}{\myunit}
```

10 V vs. NHE

15.3 Passing unprocessed digits through an S column

The method used to detect numbers in an `S` column will pick up material wrapped inside braces if there is more than a single character inside the braces. If you want to pass a *single* numerical character without processing it, you need two sets of braces (Table 18).

```
\begin{table}
  \centering
  \caption{Passing single digit characters}
  \label{tab:S-limits}
  \begin{tabular}{S[colourall,colour=orange]}
    \toprule
      {Heading} \\
    \midrule
      {1-2} \\
      {1}   \\
      {{1}} \\
      {{-}} \\ % Using {-} gives an error!
    \bottomrule
  \end{tabular}
\end{table}
```

Table 18: Passing single digit characters

Heading
1-2
1
1
-

15.4 Limitations of `\mathrm`

The package uses the `\mathrm` font family by default to typeset output in maths mode. This however has a few side-effects. For example, the Greek alphabet can give odd results.[41] The use of the `\mathnormal` font *may* get around this issue.

$4ß \times 10^{-7}$
$4\pi \times 10^{-7}$

```
\num[numaddn=\pi]{4\pi e-7}\\
\num[numaddn=\pi,mathsrm=mathnormal]{4\pi e-7}
```

On the other hand, you may want to use text mode, in which case `\ensuremath` is needed. Depending on the exact circumstances, the LaTeX protection mechanism (`\DeclareRobustCommand`) may be sufficient; in some cases, this will fail and the ε-TeX `\protected` system may be required. There are several potential pitfalls in this area; experimentation may well be needed.

$4\pi \times 10^{-7}$

```
\DeclareRobustCommand*{\numpi}{\ensuremath{\pi}}
\num[numaddn=\numpi,mode=text]{4\numpi e-7}
```

15.5 Entire document in sans serif font

If your entire document is not in a Roman font, using the font detection system is not the most efficient method for setting the siunitx output. Instead, the `mathrm` and `textrm` package options can be redefined.

Some text
1×10^{2}
3 N
4×10^{5} Pa

```
\sffamily
Some text \\
\sisetup{obeyfamily=false,mathrm=mathsf,textrm=sffamily}
\num{1e2} \\
\SI{3}{\newton}
\[ \SI{4e5}{\pascal} \]
```

15.6 Effects of emulation

The package has been designed so that almost everything can be set using the options. In the emulation code, some internal macros are redefined. This is because the legacy packages do odd things, which are deliberately not implemented by siunitx. Using an emulation file will prevent subsequent loading of the real package. This is to prevent errors or, worse, difficult to diagnose changes to output.

[41] This depends on your font setup; this document uses T1 encoding, which shows the issue, whereas using OT1 does not.

Table 19: Non-standard `S` column

Some Values
2.3456 ± 0.02
34.2345 ± 0.001
56.7835 ± 0.067
90.473 ± 0.021

15.7 Centring columns on non-decimal input

The dcolumn manual suggests using that package to align a column on a $\pm$ sign. The same type of output is possible using siunitx, but some care is needed (Table 19). Odd things may happen: use with care!

```
\begin{table}
  \centering
  \caption{Non-standard \texttt{S} column}
  \label{tab:dcolumn}
  \begin{tabular}{%
    S[digitsep=none,decimalsymbol={\,\pm\,},
      numdigits={0123456789.},numdecimal=+]}
  \toprule
  {Some Values} \\
  \midrule
  2.3456  + 0.02  \\
  34.2345 + 0.001 \\
  56.7835 + 0.067 \\
  90.473  + 0.021 \\
  \bottomrule
  \end{tabular}
\end{table}
```

15.8 Expanding content in tables

When processing `S` columns, siunitx works hard to expand any items which may give numbers. So for example you can define values as macros (Table 20). To avoid the expansion of single macros, you must either wrap them in two sets of braces or make them robust (using either `\DeclareRobustCommand` or `\protected`).

```
\begin{table}
  \centering
  \caption{Values as macros}
  \label{tab:vmacros}
  \newcommand*{\myvala}{1.234}%
  \newcommand*{\myvalb}{20.345}%
  \newcommand*{\myvalc}{0.987654}%
  \DeclareRobustCommand*{\myvald}{88.88}%
  \begin{tabular}{S[tabtextalign=left]}
  \toprule
  {Some Values} \\
```

Table 20: Values as macros

Some Values
1.234
20.345
0.987 654
1.234
88.88

```
  \midrule
  \myvala  \\
  \myvalb \\
  \myvalc \\
  {{\myvala}} \\
  \myvald \\
  \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. This means that any calculation code should be wrapped in braces and appear before the use of the number. This is illustrated by using the datatool package to set up a miniature database, and then output the values doubled (Table 21). This example also shows that macros such as `\DLTforeach`, which construct a table while TEX is running, cannot be placed inside an `S` column (or an `s` one). Instead, an additional dummy column has been added with no inter-column space. This is used to contain the tale-building macro.

```
\DTLnewdb{data}
\DTLnewrow{data}\DTLnewdbentry{data}{value}{66.7012}
\DTLnewrow{data}\DTLnewdbentry{data}{value}{66.0212}
\DTLnewrow{data}\DTLnewdbentry{data}{value}{64.9026}
\begin{table}
  \caption{Calculated values}
  \label{tab:calc}
  \centering
  \sisetup{tabformat=2.4}
  \begin{tabular}{SS@{}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}
```

Table 21: Calculated values

Value	Doubled
66.7012	133.4024
66.0212	132.0424
64.9026	129.8052

15.9 Adding items after the last column of a tabular

If you use an `S` or `s` column as the last one in a tabular, and you use the array "`<`" construction to add items after it, the spacing may be wrong. This will occur if the column contents are of differing widths. Changing the LaTeX `\\` line ending for the plain TeX `\cr` will give the correct spacing, but does not allow adjustment of inter-row distance (Table 22).[42] In most cases, this should not be a serious issue. Notice that an extra set of braces is needed here, compared to usual LaTeX use; this is to prevent any expansion of the material by siunitx.

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

[42] For the TeX experts, the issue here is that the system to gather up cell contents is added in using the < construction. Normally, this comes after the cell contents and any other < arguments, so collects the user additions. However, in the last cell the contents include `\\`, which is converted to `\cr` before gathering can occur. By using `\cr` directly, the gathering process receives all of the cell contents as normal.

Table 22: Correcting spacing in last S column

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

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

15.10 Using siunitx with the cellspace package

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

15.11 Using siunitx with the mathabx package

The mathabx package defines its own \second and \degree macros. This is respected by siunitx: if you want the siunitx definitions instead, put the lines

```
\makeatletter
\renewunit{\second}{s}
\renewunit[valuesep=none]{\degree}{\si@sym@degree}
\makeatother
```

after \begin{document}.

15.12 Numbers with no mantissa in S columns

The S column is design for handling numbers which will contain a decimal symbol. The current code deals unreliably with numbers which consist only of an exponent part. This is on the list for version 2 of siunitx: the current code makes this issue difficult to fix without major changes.

16 Reporting a problem

siunitx is quite long and complicated, and works hard to cover all possible eventualities. However, there will be bugs in the code and unexpected interactions with other packages. If you think you have found a bug, please report it. A short test-case demonstrating the problem would be very welcome. The following is a suitable template, and is available as si-bug.ltx in the doc/latex/siunitx directory or by running the .dtx or .ins file through TEX.

```
\listfiles
\documentclass{article}
```

Load other packages needed here.

```
\usepackage{siunitx}
```

Normally, debugging the load procedure will not be needed; the debug option here means that all run-time information is logged.

```
\sisetup{debug}
\begin{document}
This is the bug test-case document for the \textsf{siunitx}
package.\\
Please put your demonstration here, and e-mail to the package
author.
\begin{center}
  \texttt{joseph.wright@morningtar2.co.uk}
\end{center}
\end{document}
```

17 Feature requests

Feature requests for siunitx are welcome. The package maintainer will consider any ideas within the remit of the package (units and values). If suggesting a new feature, an example of how it should work would be appreciated. If new controls are needed, some suggestions for option names would be welcome.

18 Acknowledgements

Many thanks indeed to Stefan Pinnow, who has made a very large number of suggestions and found numerous bugs in the package. His contribution to the package has been vital. The package author has learned LaTeX tricks from far too many people to thank all of them. However, for this package specific thanks must go to the authors of the existing "unit" packages: Danie Els (SIstyle), Marcel Heldoorn (SIunits), Patrick Happel (unitsdef), Axel Reichert (units) and Harald Harders (numprint). Will Robertson and Heiko Oberdiek deserve much credit for demonstrating LaTeX coding best practice. Victor Eijkhout's excellent (and free) *TeX by Topic* has provided some useful coding hints [2]. The idea for combining and extending unit provision in LaTeX was heavily inspired by Philip Lehmann's biblatex. Thanks to the various contributors of ideas for the package: Donald Arseneau, Michele Dondi, Paul Gans, Ben Morrow, Lan Thuy Pham, Alan Ristow, Patrick Heinze, Andrea Blomenhofer, Morten Høgholm, Burkhard Moddemann and Patrick Steegstra.

Part III
Correct application of (SI) units

19 Background

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) [3].

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* (BIPM) in French [4] and English [5]. They also publish a very useful and detailed guide to using units, values and so on, available online in a number of different formats [6].

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.

20 Units

20.1 SI base units

There are seven base SI units, listed in Table 5. Apart from the kilogram, these are defined in terms of a measurable physical quantity needing the definition alone.[43] The base units have been chosen such that all physical quantities can be expressed using an appropriative combination of these units, needing no others and with no redundancy. The kilogram is slightly different from the other base units as it is still defined in terms of a "prototype" held in Paris.[44]

20.2 SI derived units

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 Table 7, Table 8 and Table 9)

[43] Some base units need others defined first; there is therefore a required order of definition.
[44] At the time of writing, this is under review and will be altered.

have a special name and symbol.[45] 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 8).

Some of these units (in Table 9) are regarded as only "temporarily" accepted, as the use of only the base and fully-consistent derived units in Table 7 is encouraged. They are accepted as they are in common use in one or more disciplines; some are still very widespread in the appropriate areas. These units are mainly multiples of base units (for example, a tonne is 1000 kg).

One point to note is that "unitless ratios" are regarded as having base units which cancel out. For example, the radian is regarded as having base unit $\mathrm{m\,m^{-1}}$. The result of this division ("1") is therefore regarded as a derived SI unit in this context.

20.3 SI prefixes

A series of SI prefixes for decimal multiples and submultiples 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. The only exception to this rule is for degrees, minutes and seconds of an arc: $1°2'3''$.

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

Three unit-prefix combinations deserve special note. The units hectare, kilohm and megohm elide the additional vowel (these would otherwise be hectoare, kiloohm and megaohm). All other vowel combinations are retained.[46]

20.4 Other units

The application of SI units is meant to provide a single set of units which ensure consistency and clarity across all areas. However, other units are common is many areas, and are not without merit. The units provided by siunitx by default do not included 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 (Section 9.1 and Table 12) 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 (such as those provided by the modules `synchem`, `hep` and `astro`) 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 particulary true of units derived from basic physical constants; for example reason, the `hep` module defines the speed of light, c, as a unit. For work in basic science, a small number of physical constants are

[45] The nautical mile has a given name but no agreed symbol, and although accepted by the SI is not provided by siunitx as a unit macro.

[46] Thanks to Charles Cameron for raising this issue with the package author.

recognised as units provided the results for comparison with experiment are given in SI units.

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.

21 Units and values in print

21.1 Mathematical meaning

As explained earlier, a unit–value 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 (°, ′ and ″), it is usual to have a space between the unit and the value. This should therefore be a non-breaking space between the two. Different geographical areas have different conventions on the size of this space; a "small" space (`\,`) is the siunitx default.

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

```
A space for \SI{10}{\percent}\\
and also for \SI{100}{\celsius}\\
but not for \ang{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. siunitx package defaults again follow this convention: any local settings are ignored, and uses the current upright serif maths font. However, there are occasions where this may not be the most desirable behaviour. A classic example would be in an all-bold section heading. As the surrounding text is bold, some people feel that any units should follow this.

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

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

21.2 Unit multiplication and division

Symbols for units formed from other units by multiplication are indicated by means of either a half-height (that is, centred) dot or a (thin) space. This document uses a half-height dot as (*i*) this is the recommendation of NIST, amongst others and (*ii*) it avoids potential confusion between unit prefixes and multiplied units.

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

```
$\si{\metre\second} = \mbox{metre second}$ \\
$\si{\milli\second} = \mbox{millisecond}$ \\
\sisetup{unitsep=thin}
$\si{\metre\second} = \mbox{metre second}$ \\
$\si{\milli\second} = \mbox{millisecond}$
```

There are some circumstances under which it is permissable to omit any spaces. The classic example is kWh, where "kW h" does not add any useful information. If using such a unit repeatedly, users of siunitx are advised to create a custom unit to ensure consistency.[47]

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.[48] 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.

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

```
\si{\joule\per\mole\per\kelvin}\\
\si[per=fraction]{\joule\per\mole\per\kelvin}\\
\si[per=slash]{\joule\per\mole\per\kelvin}
```

21.3 Repeating units

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

21.4 Clarity in writing values of quantities

Care must be taken when writing ranges of numbers. For purely numerical values, it is common to use a en-dash between values, for example "see pages 1–5". On the other hand, values with units could be misinterpret as negative values if written in this way. As the unit–value combination is a single mathematic 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.

```
\SI{1}{\metre} to \SI{5}{\metre} long.
```

21.5 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.[49] Much better is to show division of all values by the unit, which leaves the entries as unitless ratios. This is illustrated in Table 23 and Figure 1.

```
\begin{table}
  \centering
```

[47] \kWh and \kilowatthour are defined by siunitx in this way.

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

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

Table 23: An example of table labelling

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

```
  \caption{An example of table labelling}
  \label{tab:label}
  \begin{tabular}{cS[tabformat=1.4,tabnumalign=centre]}
    \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=$t/\si{\second}$,ylabel=$d/\si{\metre}$]
      \addplot[smooth,mark=x]
        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:label}
\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 24. 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}
```

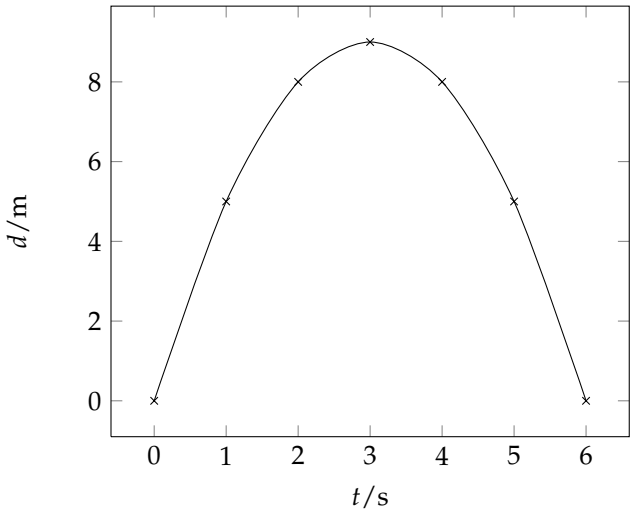

Figure 1: An example of graph labelling

Table 24: Good and bad columns

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

```
  \label{tab:exp}
  \sisetup{tabnumalign=centre}
  \begin{tabular}{%
    c%
    S[tabformat=1.3e1]<{\,\si{\kilogram}}%
    S[tabformat=2.2]}
    \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}
```

Part IV
Notes

22 Change History

23 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

L

M

N

O

P

R

S

T

U

V

24 References

[1] The IUPAC Green Book, 1993. http://old.iupac.org/publications/books/gbook/green_book_2ed.pdf.

[2] Victor Eijkhout. TEX by Topic, 2007. http://www.eijkhout.net/tbt/.

[3] http://physics.nist.gov/cuu/Units/index.html.

[4] http://www.bipm.org/fr/si/.

[5] http://www.bipm.org/en/si/.

[6] http://www.bipm.org/en/si/si_brochure/.