

siunitx — A comprehensive (SI) units package*

Joseph Wright[†]

Released 2008/09/03

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 L^AT_EX packages have been developed in the past for formatting units: Slunits, Slstyle, 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	6	Angles	11
I Introduction	4	7 Units and values	12
II Using the siunitx package	5	7.1 Literal units	12
1 For the impatient	5	7.2 The unit interpreter . .	13
2 Requirements	6	7.3 Powers of units	13
3 Loading the package	6	7.4 Units with no values . .	14
4 Numbers	6	7.5 Free-standing units . . .	14
5 Tabular material	8	7.6 Pre-defined units, prefixes and powers	14
5.1 Aligning numbers	8	7.7 Prefixed and abbreviated units	15
5.2 Columns of units	10	7.8 Defining new units . . .	19
	8	8 Specialist units	20
		8.1 Binary units (binary) .	20
		8.2 Synthetic chemistry (synchem)	20
		8.3 High-energy physics (hep)	21

*This file describes version v1.0l, last revised 2008/09/03.

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

8.4	Astronomy (astro) . . .	22	14.8	Expanding content in tables	41
9	Font control	22	14.9	Adding items after the last column of a tabular	43
10	Package options	22	15	Reporting a problem	44
10.1	Font family and style .	23	16	Feature requests	45
10.2	Spacing and separators	24	17	Acknowledgements	45
10.3	Number formatting . .	25	III	Correct application of (SI) units	46
10.4	Angle formatting	27	18	Background	46
10.5	Tabular material	28	19	Units	46
10.6	Units	30	19.1	SI base units	46
10.7	Symbols	31	19.2	SI derived units	46
10.8	Colour	32	19.3	SI prefixes	47
10.9	International support .	32	19.4	Other units	47
10.10	Package control	33	20	Units and values in print	48
10.11	Back-compatibility options	33	20.1	Mathematical meaning .	48
10.12	Summary of all options	33	20.2	Unit multiplication and division	48
11	Emulation of other packages	37	20.3	Repeating units	49
12	Configuration files	37	20.4	Clarity in writing values of quantities	49
13	Common questions	38	20.5	Graphs and tables . . .	49
13.1	Why do I need \per more than once?	38	IV	Implementation	52
13.2	Why is the order of my units changed?	38	21	Main package	52
13.3	Why are compound units not recommended outside of \SI/\si? . .	38	21.1	Setup code	52
13.4	How do I set superscripts to use lining numbers?	38	21.2	Logging	55
13.5	Why do most of the examples use J mol ⁻¹ K ⁻¹ ?	39	21.3	String comparison . . .	56
13.6	What can numprint do that siunitx cannot? . . .	39	21.4	Option handling	57
14	Tricks and known issues	39	21.5	Compatibility options .	74
14.1	Ensuring maths mode .	39	21.6	Constants	76
14.2	Using . and fixed spaces in units	39	21.7	Symbols	77
14.3	Passing unprocessed digits through an S column	40	21.8	Handling fractions . . .	78
14.4	Limitations of \mathrm	40	21.9	Font control	80
14.5	Entire document in sans serif font	41	21.10	Formatting numbers . .	84
14.6	Effects of emulation . . .	41	21.11	Formatting angles . . .	105
14.7	Centring columns on non-decimal input	41	21.12	Tabular material	112
			21.13	Units	121
			21.14	Locales	140
			21.15	Output routine	142

21.16 Finalisation	143	24.1 United Kingdom	156
22 Loadable modules	148	24.2 United States	157
22.1 Multiple prefixes	148	24.3 Germany	157
22.2 Derived units with specific names	149	24.4 South Africa	157
22.3 Units with prefixes	150	25 Emulation code	157
22.4 Abbreviated units	152	25.1 units	157
22.5 Additional (temporary) SI units	153	25.2 unitsdef	159
22.6 Units accepted for use with SI	154	25.3 Slstyle	164
22.7 Units based on physical measurements	154	25.4 Slunits	166
		25.5 hepunits	173
		25.6 fancynum	174
		25.7 fancyunits	175
23 Additional configurations	155	V Notes	178
23.1 Synthetic chemistry	155	26 Change History	178
23.2 High-energy physics	155	27 Index	178
23.3 Astronomy	156	28 References	197
23.4 Binary units	156		
24 Loadable locales	156		

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

siunitx is intended as a complete replacement for Slunits, Slstyle, 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.

¹General Conference on Weights and Measures.

²International 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 changes 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 cm³
 1 × 10³⁴
 1°2'3''
 16.7 m s⁻¹
 30 × 10³ 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 $\varepsilon\text{-}\TeX$ -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 $\mathcal{A}\mathcal{M}\mathcal{S}\TeX$ support bundle (the $\mathcal{A}\mathcal{M}\mathcal{S}$ 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 \LaTeX 3 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 10; 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	<code>\num{1}</code>	<code>\num{123}</code>	<code>\num{1234}</code>	<code>\num{12345}\</code>
0.1 0.123 0.1234 0.123 45	<code>\num{0.1}</code>	<code>\num{0.123}</code>	<code>\num{0, 1234}</code>	<code>\num{.12345}\</code>
1×10^{10} 3.45×10^{-4} -10^{10}	<code>\num{1e10}</code>	<code>\num{3.45d-4}</code>	<code>\num{-e10}</code>	

³If you have an old \LaTeX try “`elatex`” rather than “`latex`”.

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

⁵On 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 $\langle number \rangle$ 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.

$1.234(5) = 1.234 \pm 0.005$	<code>\$\num{1.234(5)} = \num[seperr]{1.234(5)}\$\\</code>
$1.234(5) \times 10^6 = (1.234 \pm 0.005) \times 10^6$	<code>\$\num{1.234(5)e6} = \num[seperr]{1.234(5)e6}\$</code>

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

$m_e = 9.109\,389\,7(54) \times 10^{-31} \text{ kg}$	<code>\$\m_{\mathrm{e}}\$</code>
$m_e = (9.109\,389\,7 \pm 0.000\,005\,4) \times 10^{-31} \text{ kg}$	<code>= \SI{9.1093897(54)e-31}{\kg} \$\\</code>
	<code>\$\m_{\mathrm{e}}\$</code>
	<code>= \SI[seperr]{9.1093897(54)e-31}{\kg} \$</code>

A number of effects are available as options. These are fully explained in Section 10. 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

$1\,234\,567\,890$	<code>\num{1234567890}</code>	<code>\num[mode=text]{1234567890}</code>
--------------------	-------------------------------	--

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

$1\,1.0$	<code>\num{1.}</code>	<code>\num[padnumber=all]{1.}\\</code>
$2+2$	<code>\num{2}</code>	<code>\num[addsign=all]{2}\\</code>
$3 \times 10^4 + 3 \times 10^4 + 3 \times 10^{+4}$	<code>\num{3e4}</code>	<code>\num[addsign=mant]{3e4} \num[addsign=all]{3e4}\\</code>
$0.5\,5$	<code>\num{.5}</code>	<code>\num[padnumber=none]{.5}</code>

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

$1234\,1234$	<code>\num{1234}</code>	<code>\num[sepfour=true]{1234}\\</code>
$12\,345\,12,345$	<code>\num{12345}</code>	<code>\num[digitsep=comma]{12345}\\</code>
12345	<code>\num{12345}</code>	<code>\num[digitsep=none]{12345}</code>

The formatting of exponents is also customisable.

1×10^{10}	<code>\num{1e10}</code>	<code>\num[expproduct=cdot]{1e10}\\</code>
2×10^{20}	<code>\num{2e20}</code>	<code>\num[expbase=5]{2e20}\\</code>
3×10^{30}	<code>\num{3e30}</code>	<code>\num[expproduct=tighttimes]{3e30}</code>

`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	<code>\num{-1}\\</code>
-2	<code>\sisetup{colourneg} \num{-2}\\</code>
3×10^{-3}	<code>\num{3e-3}\\</code>
-4	<code>\num[negcolour=blue]{-4}</code>

⁶This is common in chemical crystallography, for example.

⁷This 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.

1.23456	<code>\num{1.23456}\</code>
1.23	<code>\sisetup{dp=2}</code>
9.80	<code>\num{1.23456}\</code>
-10.43	<code>\num{9.8}\</code>
44.3221(2)	<code>\num{-10.432}\</code>
	<code>\num{44.3221(2)}</code>

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 `dcolum` 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}
```

⁸The approach used is actually a combination of `dcolum` 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 from `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}	$\text{m}^2 \text{s}^{-1}$
2.83×10^{-6}	$\text{m}^2 \text{s}^{-1}$
7.39×10^3	$\text{Pa m}^3 \text{mol}^{-1}$
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.

$10^\circ 12.3^\circ 4.5^\circ$	<code>\ang{10} \ang{12.3} \ang{4,5}\</code>
$1^\circ 2' 3'' 0^\circ 1''$	<code>\ang{1;2;3} \ang{;;1}\</code>
$10^\circ -0^\circ 1'$	<code>\ang{+10;;} \ang{-0;1;}</code>

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.

$0^\circ 0' 1'' 1''$	<code>\ang{;;1} \ang[padangle=none]{;;1}\</code>
$2^\circ 2' 0''$	<code>\ang{2;;} \ang[padangle=all]{2;;}\</code>
$0^\circ 3' 0'' 4^\circ 0' 0'' 0^\circ 0' 5''$	<code>\sisetup{padangle=all} \ang{;3;} \ang{4;;} \ang{;;5}</code>

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.

$1.05^\circ 1.05^\circ$	<code>\ang{1.05} \ang[decimalsymbol=comma]{1.05}\</code>
$3.67890^\circ 3.67890^\circ$	<code>\ang{3.67890} \ang[digitsep=comma]{3.67890}\</code>
$9^\circ 8' 7'' 9^\circ 8' 7''$	<code>\ang{9;8;7} \ang[anglesep=thin]{9;8;7}</code>

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 `Slstyle` 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	<code>\SI[mode=text]{1.23}{J.mol^{-1}.K^{-1}}\</code>
0.23 × 10 ⁷ cd	<code>\SI{.23e7}{\candela}\</code>
£1.99/kg	<code>\SI[per=slash]{1.99}{\pounds\perkilogram}\</code>
70 m s ⁻¹	<code>\SI{70}{\metre\persecond}\</code>
1.345 A/mol	<code>\SI[per=frac,fraction=nice]{1,345}{\ampere\per\mole}</code>

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 `Slstyle` and `Slunits`. The `Slstyle`-like method uses literal input. Four characters have a special meaning:

- “^” The superscript character is 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 `unitsspace` option are typeset by a tilde.

This allows ready input of units.

10 kg m s ⁻²	<code>\SI{10}{kg.m.s^{-2}}\</code>
1.453 g/cm ³	<code>\SI{1.453}{g/cm^3}\</code>
33.562 cd s	<code>\SI{33.562}{cd~s}\</code>
100 m s ⁻²	<code>\SI[unitsep=medium]{100}{m.s^{-2}}</code>

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

10 μm	<code>\SI{10}{\mu m}\</code>
20 °C	<code>\SI{20}{^{\circ}C}\</code>
30 Å	<code>\SI{30}{\AA}\</code>

⁹Currently this works with X_YL_AT_EX and inputenc using the `latin1`, `latin5` and `latin9` encodings.

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 kg m s^{-2}	<code>\SI{10}{\kilo\gram\metre\per\second\squared}\</code>
1.453 g cm^{-3}	<code>\SI{1.453}{\gram\per\cubic\centi\metre}\</code>
33.562 cd s	<code>\SI{33.562}{\candela\second}\</code>
100 m s^{-2}	<code>\SI[unitsep=thin]{100}{\metre\per\Square\second}\</code>
$4.56 \times 10^3 \text{ m s}^{-1}$	<code>\SI[prefixsymbolic=false]{4.56}{\kilo\metre\per\second}\</code>

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.

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

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

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

10 m^2	<code>\SI{10}{\metre\squared}\</code>
20 m^2	<code>\SI{20}{\Square\metre}\</code>
30 m^3	<code>\SI{30}{\metre\cubed}\</code>
40 m^3	<code>\SI{40}{\cubic\metre}\</code>

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

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

`\tothe` For powers not defined above or with `\newpower`, the `\tothe` macro can be

`\raiseto`

¹⁰Users of Slunits should note the lack of need for a `\usk`-type macro.

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 m ⁴	<code>\SI{16.86}{\metre\tothe{4}}\%</code>
7.895 N ⁻⁶	<code>\SI{7.895}{\raiseto{-6}\newton}\%</code>
1.34 K ⁻⁷	<code>\SI{1.34}{\per\kelvin\tothe{7}}\%</code>

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.

kg m/s ²	<code>\SI{}{kg.m/s^2}\%</code>
kg m/s ²	<code>\si{kg.m/s^2}\%</code>
mol dm ⁻³	<code>\si[mode=text,unitsep=thin]{\mole\per\cubic\deci\metre}\%</code>

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 lose font control of the value. When used in this way, the units *do not* take an optional keyval argument.

123 m	<code>\sisetup{prespace,allowoptarg}</code>
123 K	<code>123\metre\%</code>
234 A	<code>\kelvin[123]\%</code>
6 s	<code>\sisetup[mode=text]{\ampere[234]\%}</code> <code>6\second</code>

7.6 Pre-defined units, prefixes and powers

`\metre` 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` 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

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.

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

¹³The official SI spelling for the unit is “metre”.

Table 5: The seven base SI units

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

Table 6: The SI prefixes (`load=prefix`)

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

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.

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=prefix`). 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.

¹⁴The macros `\ohm` and `\celsius` are not defined by `siunitx` if the `gensymb` package is loaded.

¹⁵These are supposed to be replaced over time by SI units.

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

Unit	Macro	Symbol	Unit	Macro	Symbol
becquerel	\becquerel	Bq	newton	\newton	N
celsius	\celsius	°C	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

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

Unit	Macro	Symbol	Abbreviation
<i>Masses</i>			
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
<i>Lengths</i>			
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
<i>Times</i>			
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

Continued on next page

Unit	Macro	Symbol	Abbreviation
<i>Moles</i>			
femtomole	\femtomole	fmol	\fmol
picomole	\picomole	pmol	\pmol
nanomole	\nanomole	nmol	\nmol
micromole	\micromole	μmol	\micmol
millimole	\millimole	mmol	\mmol
<i>Currents</i>			
picoampere	\picoampere	pA	\pA
nanoampere	\nanoampere	nA	\nA
microampere	\microampere	μA	\micA
milliampere	\milliampere	mA	\mA
kiloampere	\kiloampere	kA	\kA
<i>Areas</i>			
square centimetre	\squarecentimetre	cm^2	\cms
	\centimetresquared	cm^2	
square metre	\squaremetre	m^2	
square kilometre	\squarekilometre	km^2	
<i>Volumes</i>			
microlitre	\microlitre	μl	\micl
millilitre	\millilitre	ml	\ml
cubic centimetre	\cubiccentimetre	cm^3	\cmc
	\centimetrecubed	cm^3	
cubic decimetre	\cubicdecimetre	dm^3	\dmc
<i>Frequencies</i>			
hertz	\hertz	Hz	\Hz
millihertz	\millihertz	mHz	\mHz
kilohertz	\kilohertz	kHz	\kHz
megahertz	\megahertz	MHz	\MHz
gigahertz	\gigahertz	GHz	\GHz
terahertz	\terahertz	THz	\THz
<i>Potentials</i>			
millivolt	\millivolt	mV	\mV
kilovolt	\kilovolt	kV	\kV
<i>Energies</i>			
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

Continued on next page

Unit	Macro	Symbol	Abbreviation
<i>Powers</i>			
milliwatt	<code>\milliwatt</code>	mW	
kilowatt	<code>\kilowatt</code>	kW	
megawatt	<code>\megawatt</code>	MW	
<i>Capacitances</i>			
femtofarad	<code>\femtofarad</code>	fF	
picofarad	<code>\picofarad</code>	pF	
nanofarad	<code>\nanofarad</code>	nF	
microfarad	<code>\microfarad</code>	μF	
millifarad	<code>\millifarad</code>	mF	
<i>Resistances</i>			
kiloohm	<code>\kiloohm</code>	kΩ	
megaohm	<code>\megaohm</code>	MΩ	
gigaohm	<code>\gigaohm</code>	GΩ	
millisiemens	<code>\millisiemens</code>	mS	
<i>Forces</i>			
millinewton	<code>\millinewton</code>	mN	
kilonewton	<code>\kilonewton</code>	kN	
<i>Other units</i>			
hectopascal	<code>\hectopascal</code>	hPa	
megabecquerel	<code>\megabecquerel</code>	MBq	
millisievert	<code>\millisievert</code>	mSv	

7.8 Defining new units

`\newunit` 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°          \SI{3.1415}{\degree}\
12 345XXX 67 890 XXX \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

¹⁶Although the `\ang` macro is preferred for this job.

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` Powers are defined: `\newpower[post]{<power>}{<num>}`. Here, `<power>` is
`\renewpower` the name of the power macro and `<num>` is the (positive) number it represents.
`\providepower` 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 `\newpower{\quartic}{4}`
 m^4 `\newpower[post]{\totheforth}{4}\`
`\si{\kilogram\totheforth}\`
`\si{\quartic\metre}`

`\newprefix` The standard SI powers of ten are defined by the package, and are described above. However, the user can define new prefixes with `\newprefix`.
`\renewprefix` This has syntax `\newunit[binary]{<prefix>}{<symbol>}{<powers-ten>}`, where
`\provideprefix` `<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}`

8 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>`.

8.1 Binary units (binary)

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

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

8.2 Synthetic chemistry (synchem)

`\mmHg` The `synchem` file adds the common chemistry units `\mmHg`, `\molar`, `\Molar`,
`\molar` `\torr` and `\dalton` to `siunitx`. The `\Molar` macro is somewhat awkward, as it
`\Molar`
`\torr`
`\dalton`

¹⁷The `\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

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

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

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.

1 M HCl	\SI{1}{\Molar} HCl\
760 Torr	\SI{760}{\torr}\
0.01 mmHg	\SI{0.01}{\mmHg}\
3.0 mol dm ⁻³	\SI{3.0}{\molar}\
106.42 Da	\SI{106.42}{\dalton}

8.3 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.

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

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

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/c² `\SI[per=slash,eVcorrb=0.4ex]{4.657}{\mega\eVperc\squared}`

8.4 Astronomy (**astro**)

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

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

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

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

10 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.

10.1 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.

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).

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.

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

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 (\dots) maths. The `obeyitalic` option does *not* have the same facility (maths is italic anyway).

The font commands used by the package to achieve the above are all available for user modification. The options `mathsrn`, `mathssf` and `mathstt` hold the command sequences used in maths mode,²⁰ while `textrn`, `textsf` and `texttt` do the same for text mode. By default, these contain the obvious command names, for example `mathsrn=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 `mathsrn` option may be split into `valuemathsrn` and `unitmathsrn`.

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	$x = 1.2 \times 10^3 \text{ kg K cd}$	<pre>\sffamily Some text \sisetup{obeyall} \[x = \SI{1.2e3}{\kg\kelvin\candela} \]</pre>
More text	$y = 3 \text{ m s mol}$	<pre>More text \sisetup{detectdisplay=false} \[y = \SI{3}{\metre\second\mole} \]</pre>

10.2 Spacing and separators

The separators between items can all be set using options taking a list of pre-defined 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**.

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.

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

10.3 Number formatting

numdigits	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 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). 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	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 dependant 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 .
digitsep	
sepfour	
1234	<code>\num{1234}\</code>
1 234	<code>\num[sepfour]{1234}</code>
seperr	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=)].
numopenerr	
numcloseerr	
1.234 ± 0.005	<code>\sisetup{seperr}\</code>
(1.234 ± 0.005) × 10 ⁶	<code>\num{1.234(5)}\</code> <code>\num{1.234(5)e6}</code>
trapambigerr	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 ± sign is normally set by T _E X. However, using the tightpm option will cause this to be reduced to a minimum.
openerr	
closeerr	
tightpm	
1.234 × 10 ⁶ ± 0.005 × 10 ⁶	<code>\sisetup{seperr}\</code> <code>\num[trapambigerr=false]{1.234(5)e6}\</code>
1.234 m ± 0.005 m	<code>\SI{1.234(5)}{\metre}\</code>
(1.234 ± 0.005) m	<code>\sisetup{repeatunits=false}</code> <code>\SI{1.234(5)}{\metre}\</code>
1.234 ± 0.005 m	<code>\SI[trapambigerr=false]{1.234(5)}{\metre}\</code>
1.234±0.005	<code>\num[tightpm]{1.234(5)}</code>

²¹fullstop also has aliases stop and period.

`numprod` The number processor can recognise products in the numerical input; the
`repeatunits` 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 latter 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`.

$1 \times 2 \times 3$
 $4\text{ m} \times 5\text{ m} \times 6\text{ m}$
 $1.2 \times 3.4 \times 5.6\text{ mm}^3$
 $(1.234 \pm 0.005)\text{ m}$
 $9.1093897 \pm 0.0000054 \times 10^{-31}\text{ kg}$
 $9.1093897 \times 10^{-31}\text{ kg} \pm 0.0000054 \times 10^{-31}\text{ kg}$
 $1 \times 2 \times 3\text{ m}^3$

```
\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` The formatting of exponents is controlled by `expproduct` and `expbase`.
`expbase` `expproduct` sets the symbol used to indicate a product for exponents (e.g. the
`allowzeroexp` \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.²³
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` Additions to the input can take the form of implicit signs and padded ze-
`sign` ros. The `addsign` option takes a list of potential sites to add a sign: **none**,
`retainplus` **mantissa**, **exponent** and **both**.²⁴ 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.²⁵ No additional precision is added by this option; integer input will not
add a decimal point.

²²This is a very simply option: do not expect it to work with anything except areas and volumes.

²³The **tighttimes** and **tightcdot** options give the rather questionable results: 1×10^2 and $1 \cdot 10^2$,
as opposed to 1×10^2 and $1 \cdot 10^2$.

²⁴Aliases are provided: `mant` = `mantissa`, `exp` = `exponent`, `all` = `true` = `both`, `false` = `none`.

²⁵Aliases: `all` = `true` = `both`, `false` = `none`.

0.1	<code>\num[padnumber=leading]{.1}\</code>
2	<code>\num[padnumber=leading]{2.}\</code>
3.0	<code>\num[padnumber=trailing]{3.}\</code>
4.0	<code>\num[padnumber=both]{4.}\</code>
0.5	<code>\num[padnumber=both]{.5}\</code>
6	<code>\num[padnumber=both]{6}\</code>
7	<code>\num[padnumber=none]{7.}\</code>
.8	<code>\num[padnumber=none]{.8}\</code>

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

1.000	<code>\sisetup{fixdp,dp=3}</code>
53.900	<code>\num{1}\</code>
4.568	<code>\num{53.9}\</code>
-1.294	<code>\num{4.56783}\</code>
-1.295	<code>\num{-1.2942}\</code>
10.000	<code>\num{-1.2949}\</code>
	<code>\num{9.9999}\</code>

10.4 Angle formatting

`padangle` The angle formatter uses `\num` to format numbers: any options for numbers are
`strictarc` 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.²⁶ 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°	<code>\ang[padangle=none]{1;}\</code>
2°0'0"	<code>\ang[padangle=large]{2;}\</code>
3°	<code>\ang[padangle=small]{3;}\</code>
0°4'0"	<code>\ang[padangle=both]{;4;}\</code>
5'5"	<code>\ang[padangle=none]{;5;5}\</code>
1°2'	<code>\ang[strictarc=false]{1;2}\</code>
1°2'3"	<code>\ang[strictarc=false]{1;2;3;4;}\</code>

`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**.²⁷ 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.

²⁶Aliases: `all = true = both`, `false = none`.

²⁷Aliases: `decimal = dec`, `arc = dms`, `unchanged = none`.

$$\begin{aligned}\$ \backslash \text{ang}\{1;2;3\} &= \backslash \text{ang}[angformat=dec]\{1;2;3\} \$ \\ \$ \backslash \text{ang}\{4.56\} &= \backslash \text{ang}[angformat=arc]\{4.56\} \$\end{aligned}$$

1°2'3.4"	<code>\ang{1;2;3.4}\</code>
5°6'7''8	<code>\ang[astroang]{5;6;7.8}</code>

10.5 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`. `tabnumalign` takes values **centredecimal**, `centre`, `left` and `right`.²⁸ 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 $\langle pre \rangle \langle dec \rangle \langle post \rangle$; $\langle pre \rangle$ and $\langle post \rangle$ 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` When `tabformat` contains exponents, two possibilities are available for alignment. The first method is to place the exponent parts so that the “ $\times 10$ ” 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 not line up (Table 14). This is controlled by the `tabexpalign` option, which can be set to `true` or `false`.

```
\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}
```

Table 14: The `tabalignexp` option

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

Table 15: The `tabautofit` option

Header	Header	Header
1.2	1.200	1.2
1.2345	1.235	1.2345

`tabtextalign` Cells containing no numbers are handled by `siunitx` in a manner similar to `\multicolumn`. The setting of `tabtextalign` is taken from the list **centre**, `tabunitalign` **right** and **left**.²⁹ 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` 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}
```

²⁹Alias `centre = center`.

10.6 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`.³⁰ The default option uses `\per` to indicate reciprocal powers, whereas `slash` causes the package to use `"/` to show division.

`fraction` 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 T_EX `\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.³¹ Finally, the setting `fraction=sfrac` uses the `\sfrac` macro from the `xfrac` package, when available.³² The `slash` option sets the symbol used when `per=slash` is in force. This recognises the single keyword `slash`; anything else is used literally.

$\frac{m}{s}$	<code>\sisetup{per=fraction}</code>
m/s	<code>\si[fraction=frac]{\metre\per\second}\\</code>
m_s	<code>\si[fraction=nice]{\metre\per\second}\\</code>
m/s	<code>\si[fraction=sfrac]{\metre\per\second}\\</code>
	<code>\si[per=slash]{\metre\per\second}</code>

`stickyper` By default, `\per` applies only to the next unit given.³³ By setting the `stickyper` flag, this behaviour is changed so that `\per` applies to all subsequent units.³⁴

$kg\,m^{-1}\,A$	<code>\si{\kilogram\per\metre\ampere}\\</code>
$kg\,m^{-1}\,A^{-1}$	<code>\si[stickyper]{\kilogram\per\metre\ampere}</code>

`trapambigfrac` 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.

$m/(s\,kg)$	<code>\sisetup{trapambigfrac,openfrac=(,closefrac=),per=slash}\\</code>
$m/s\,kg$	<code>\si{\metre\per\second\per\kilogram}\\</code>
	<code>\si[trapambigfrac=false]{\metre\per\second\per\kilogram}</code>

`prefixsymbolic` The unit prefixes (`\kilo`, *etc.*) are normally given as letters. However, the package can convert these into numerical powers.³⁵ 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

³⁰Aliases `reciprocal = rp = power`, `fraction = frac`.

³¹Similar to the `ugly` option of the `nicefrac` package.

³²`xfrac` is part of the experimental system for L^AT_EX₃. 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`.

³³This is the standard method of reading units in English: for example, $J\,mol^{-1}\,K^{-1}$ is pronounced “joules per mole per kelvin”.

³⁴This is the behaviour in `Slunits`.

³⁵Provided things are not too complex!

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

mis the symbol for metres	<code>\metre</code> is the symbol for metres\\
No, m is the correct symbol	<code>\sisetup{xspace}</code>
30 m	No, <code>\metre</code> is the correct symbol\\
Do not use m in running text	<code>\sisetup{prespace}</code>
40 m	30 <code>\metre</code> \\
	Do not use <code>\metre</code> in running text\\
	<code>\sisetup{allowoptarg}</code>
	<code>\metre[40]</code>

10.7 Symbols

User access to control the symbols used for Ω , μ , $^\circ$, $'$, $''$, \AA and $^\circ\text{C}$ 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`;
- `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` 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.³⁶

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

10.8 Colour

`colourall` 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 `colorall` behave in the same way. With colour turned off, no `\color` command is issued internally, and output follows the surrounding text.

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

`colourneg` `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.

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

³⁶This document uses `eVcorra=0.1ex`.

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

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

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

10.10 Package control

`load` 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.³⁷

`log` 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.

10.11 Back-compatibility options

`emulate` The package can emulate `Slunits`, `Slstyle`, `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.

10.12 Summary of all options

Table 16 lists a summary of the package options (excluding those for backward-compatibility). A reminder of the input format is also provided.

Table 16: All package options

Option	Type	Description
<code>addsign</code>	List	Add sign to number
<code>allowzeroexp</code>	Boolean	Allow 10^0

Continued on next page

³⁷`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.

Option	Type	Description
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
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 places of numbers
fraction	List	Method used when setting <code>per=frac</code>
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	"°C" 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	"°C" symbol in maths mode
mathsdegree	Literal	"°" symbol in maths mode
mathsecond	Literal	"″" symbol in maths mode
mathsf	Csname	Sans serif maths font

Continued on next page

Option	Type	Description
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
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, 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
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 and products
retainplus	Boolean	Retain explicit plus sign
seper	Boolean	Separate number and error
sepfour	Boolean	Separate four-digit numbers
sign	List or literal	Sign to add to numbers
slash	List or literal	Symbol used for "/"

Continued on next page

Option	Type	Description
<code>stickyper</code>	Boolean	Require <code>\per</code> only once
<code>strict</code>	Boolean	Obey the rules strictly
<code>strictarc</code>	Boolean	Require exactly zero or two semi-colons
<code>tabalign</code>	List	Positioning of all column data
<code>tabalignexp</code>	Boolean	Alignment of exponents in tables
<code>tabautofit</code>	Boolean	Switch for rounding numbers to length given by <code>tabformat</code>
<code>tabformat</code>	Number	Space reserved in table for numbers
<code>tabnumalign</code>	List	Alignment of S column numbers
<code>tabtextalign</code>	List	Positioning of text in S columns
<code>tabunitalign</code>	List	Positioning of units in s columns
<code>textcelsius</code>	Literal	"°C" symbol in text mode
<code>textdegree</code>	Literal	"°" symbol in text mode
<code>textminute</code>	Literal	"'" symbol in text mode
<code>textmu</code>	Literal	"μ" symbol in text mode
<code>textomega</code>	Literal	"Ω" symbol in text mode
<code>textringa</code>	Literal	"Å" symbol in maths mode
<code>textrm</code>	Csname	Roman text font
<code>textsecond</code>	Literal	"'" symbol in text mode
<code>textsf</code>	Csname	Sans serif text font
<code>texttt</code>	Csname	Fixed-width text font
<code>tightpm</code>	Boolean	Reduce space around \pm
<code>trapambigerr</code>	Boolean	Check for ambiguous errors
<code>trapambigfrac</code>	Boolean	Check for ambiguous fractions
<code>unitcolor</code>	Literal	Switch for colouring units
<code>unitcolour</code>	Literal	Switch for colouring units
<code>unitmathrm</code>	Csname	Roman maths font for units
<code>unitmathsf</code>	Csname	Sans serif maths font for units
<code>unitmathsrn</code>	Csname	Roman maths font for units
<code>unitmathssf</code>	Csname	Sans serif maths font for units
<code>unitmathstt</code>	Csname	Fixed-width maths font for units
<code>unitmathtt</code>	Csname	Fixed-width maths font for units
<code>unitmode</code>	List	As <code>mode</code> , for units only
<code>unitsep</code>	List or literal	Separator for units
<code>unitspace</code>	List or literal	Space used for "~" in units
<code>valuecolor</code>	Literal	Switch for colouring value
<code>valuecolour</code>	Literal	Switch for colouring value
<code>valuemathrm</code>	Csname	Roman maths font for values
<code>valuemathsf</code>	Csname	Sans serif maths font for values
<code>valuemathsrn</code>	Csname	Roman maths font for values
<code>valuemathssf</code>	Csname	Sans serif maths font for values
<code>valuemathstt</code>	Csname	Fixed-width maths font for values
<code>valuemathtt</code>	Csname	Fixed-width maths font for values
<code>valuemode</code>	List	As <code>mode</code> , for values only
<code>valuesep</code>	List or literal	Separator between value and unit
<code>xspace</code>	Boolean	Use <code>xspace</code> after units

11 Emulation of other packages

siunitx has been designed as a replacement for Slunits, Slstyle, 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.³⁸ This means for example that the `\num` macro takes an optional star when emulating Slstyle. 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 provided here. To use an numprint document with siunitx, the `\numprint` macro could be provided using the following code.

```

-123 456
-123 456 N/mm2

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

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.

12 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⟩}`.

```

Some text as filler
TEST This example is rather trivial!

\addtolocale{DE}{TEST}
Some text as filler \SI[obeymode]{-123456}{N/mm^2}
\sisetup{locale=DE}
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`.

³⁸User macros means that they are described in the package documentation; simply not containing an @ does not mean they will have been emulated.

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.

13 Common questions

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

13.2 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.³⁹

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

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

13.3 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`.

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

```
\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.

13.4 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.

³⁹“Double division” (1 s/m/kg) is mathematically incorrect.

1234 m s⁻¹
1234 m s⁻¹

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

13.5 Why do most of the examples use 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!

13.6 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:

- 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.

14 Tricks and known issues

14.1 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 $\mathcal{A}\mathcal{M}\mathcal{S}\mathcal{T}\mathcal{E}\mathcal{X}$ bundle) to ensure text mode.

14.2 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 `unitsspace=space` to get a full space here.

10 V vs. NHE

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

Table 17: Passing single digit characters

Heading
1-2
1
1
-

14.3 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 17).

```
\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}
```

14.4 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.⁴⁰ The use of the \mathnormal font *may* get around this issue.

$4\text{\textcircled{B}} \times 10^{-7}$	<code>\num[numaddn=\pi]{4\pi e-7}\</code>
$4\pi \times 10^{-7}$	<code>\num[numaddn=\pi,mathsrn=mathnormal]{4\pi e-7}</code>

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

$4\pi \times 10^{-7}$	<code>\DeclareRobustCommand*{\numpi}{\ensuremath{\pi}}</code>
	<code>\num[numaddn=numpi,mode=text]{4\numpi e-7}</code>

⁴⁰This depends on your font setup; this document uses T1 encoding, which shows the issue, whereas using OT1 does not.

14.5 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	<code>\sffamily</code>
1×10^2	Some text <code>\</code>
3 N	<code>\sisetup{obeyfamily=false,mathrm=mathsf,textrm=sffamily}</code>
	<code>\num{1e2} \</code>
$4 \times 10^5 \text{Pa}$	<code>\SI{3}{\newton}</code>
	<code>\[\num{4e5} \si{\pascal} \]</code>

14.6 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.

14.7 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 18). Odd things may happen: use with care!

```
\begin{table}
  \caption{Non-standard \texttt{S} column}
  \label{tab:dcolumn}
  \centering
  \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}
```

14.8 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 19). To avoid the expansion of single macros, you must either wrap them in two

Table 18: Non-standard S column

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

Table 19: Values as macros

Some Values
1.234
20.345
0.987654
1.234
88.88

sets of braces or make them robust (using either `\DeclareRobustCommand` or `\protected`).

```
\begin{table}
  \caption{Values as macros}
  \label{tab:vmacros}
  \centering
  \newcommand*{\myvala}{1.234}%
  \newcommand*{\myvalb}{20.345}%
  \newcommand*{\myvalc}{0.987654}%
  \DeclareRobustCommand*{\myvald}{88.88}%
  \begin{tabular}{S[tabletextalign=left]}
    \toprule
    {Some Values} \\
    \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 20). 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

Table 20: Calculated values

Value	Doubled
66.7012	133.4024
66.0212	132.0424
64.9026	129.8052

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}

```

14.9 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 `\cr` line ending for the plain \TeX `\cr` will give the correct spacing, but does not allow adjustment of inter-row distance (Table 21).⁴¹ 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}} }

```

⁴¹For 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 21: Correcting spacing in last S column

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

```

\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}

```

15 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 .

```

1 \listfiles
2 \documentclass{article}

```

Load other packages needed here.

```

3 \usepackage{siunitx}

```

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

```

4 \sisetup{debug}
5 \begin{document}
6 This is the bug test-case document for the \textsf{siunitx}

```

```

7package.\
8Please put your demonstration here, and e-mail to the package
9author.
10\begin{center}
11  \texttt{joseph.wright@morningtar2.co.uk}
12\end{center}
13\end{document}

```

16 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.

17 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 (`Slstyle`), Marcel Heldoorn (`Slunits`), 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 `bi-latex`. 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

18 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.

19 Units

19.1 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.⁴² The base units have been chosen such that all physical quantities can be expressed using an appropriate combination of these units, needing no others and with no redundancy. The kilogram is slightly different from the other base units as it is still defined in terms of a “prototype” held in Paris.⁴³

19.2 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)

⁴²Some base units need others defined first; there is therefore a required order of definition.

⁴³At the time of writing, this is under review and will be altered.

have a special name and symbol.⁴⁴ 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 m m^{-1} . The result of this division (“1”) is therefore regarded as a derived SI unit in this context.

19.3 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 $1 \text{ m}^\circ\text{C}$. The only exception to this rule is for degrees, minutes and seconds of an arc: $1^\circ 2' 3''$.

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

19.4 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 in many areas, and are not without merit. The units provided by `siunitx` by default do not include any of these; only units which are part of the SI set or are accepted for use with SI units are defined. However, several other sets of units can be loaded as optional modules. The binary prefixes and units (Section 8.1 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 particularly 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 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 che-

⁴⁴The 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.

mists 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.

20 Units and values in print

20.1 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 ($^{\circ}$, $'$ 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 °C
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**: 54F
But perhaps in a running block,
it might look better: 54F

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

20.2 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 permissible 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.⁴⁵

⁴⁵`\kWh` and `\kilowatthour` are defined by siunitx in this way.

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.⁴⁶ 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.

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

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

20.3 Repeating units

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

20.4 Clarity in writing values of quantities

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

20.5 Graphs and tables

In tables and graphs, 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.⁴⁷ Much better is to show division of all values by the unit, which leaves the entries as unitless ratios. This is illustrated in Table 22 and Figure 1.

```
\begin{table}
\centering
\caption{An example of table labelling}
\label{tab:label}
\begin{tabular}{cS[tableformat=1.4,tabnumalign=centre]}
\toprule
{Entry} & {Length/\si{\metre}} \\
\end{tabular}
\end{table}
```

⁴⁶Notice that a virgule and a solidus are not the same symbol.

⁴⁷For example, for an acceleration a , the expression $[a]$ is the dimensions of a , i.e. length per time squared in this case.

Table 22: An example of table labelling

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

```

\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=$t/\text{second}$,ylabel=$d/\text{metre}$]
\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 23**. The use of `\multicolumn` is needed here due to the “<”; without `\multicolumn`, the titles are followed by “kg”!

```

\begin{table}
\centering
\caption{Good and bad columns}
\label{tab:exp}
\sisetup{tabnumalign=centre}
\begin{tabular}{c}
c \\
S[tabformat=1.3e1]<\,\text{kilogram}\}
\end{tabular}
\end{table}

```

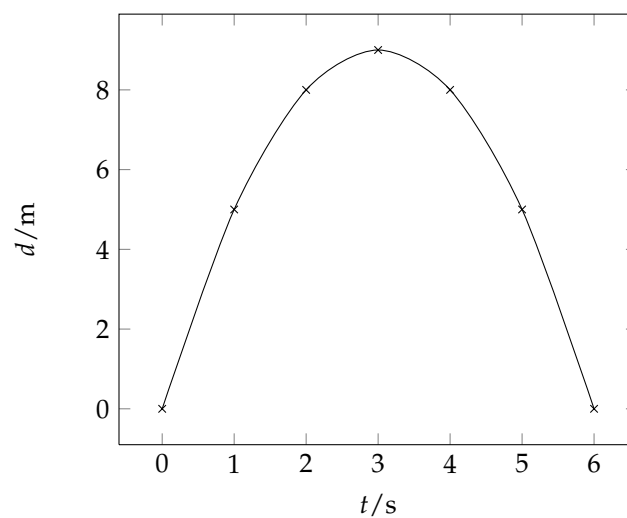


Figure 1: An example of graph labelling

Table 23: Good and bad columns

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

```

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

Implementation

21 Main package

Much of the code here is taken, with little or no modification, from the existing packages. These are all released under the LPPL, and so this use is entirely allowed. Rather than confuse the source here with repeated references, note that code here could be copied from `SIstyle`, `SIunits`, `numprint`, `unitsdef` or `units`. Some ideas have also been borrowed from `biblatex`; again these will not be specifically noted. Code from other packages will be marked when used.

User-space commands (those not containing `@`) defined here should give the same result as macros with the same name in the older packages.⁴⁸ However, internal package macros will behave differently; if the user has redefined internal macros, then compatibility will be impaired.

The code used here uses \LaTeX rather than \TeX commands where possible.⁴⁹ For example, `\newcommand*` is used in place of `\def`, unless custom parameters are needed. Hopefully, this will aid future maintenance. Grouping is used where possible to limit the scope of temporary assignments.

21.1 Setup code

As always, the package starts with identification.

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{siunitx}
3 [2008/09/03 v1.01 A comprehensive (SI) units package]
```

The package requires $\epsilon\text{-TeX}$, so the usual test is made.

```
4 \begingroup
5   \@ifundefined{eTeXversion}
6   {\PackageError{siunitx}
7     {Not running under e-TeX}
8     {This package requires e-TeX. Try compiling the document
9       with\MessageBreak 'elatex' instead of 'latex'. When
10        using pdfTeX, try 'pdfelatex'\MessageBreak instead of
11         'pdflatex'}%
12   \endgroup\endinput}
13 {\endgroup}
```

`\si@catcodes` There are circumstances under which some odd category codes might be in place. The category codes for `{`, `}`, `[`, `]` and `#` are assumed to be okay; if issues arise, this can be altered. \LaTeX will have set `@` as a letter on loading `siunitx`.

```
14 \edef\si@catcodes{%
15   \catcode\string'\string ` \the\catcode\string''\relax
16   \catcode\string'\string = \the\catcode\string''=\relax
17   \catcode\string'\string ^ \the\catcode\string''^\relax
18   \catcode\string'\string ~ \the\catcode\string''~\relax
19   \catcode\string'\string : \the\catcode\string'':\relax}
```

⁴⁸Although extra optional arguments may be added.

⁴⁹This applies to \LaTeX kernel commands only; for example, `ifthenelse` is not used.

```

20 \catcode\string'\string - \the\catcode\string'\-\relax
21 \catcode\string'\string + \the\catcode\string'\+\relax
22 \catcode\string'\string ; \the\catcode\string'\;\relax
23 \catcode\string'\string , \the\catcode\string'\,\relax
24 \catcode\string'\string . \the\catcode\string'\.\relax
25 \catcode\string'\` 12\relax
26 \catcode\'= 12\relax
27 \catcode\'^ 7\relax
28 \catcode\'~ \active\relax
29 \@makeother{\:}
30 \@makeother{\-}
31 \@makeother{\+}
32 \@makeother{\;}
33 \@makeother{\,}
34 \@makeother{\.}

```

Packages needed for functionality are loaded. xkeyval handles the package options, while amstext from the \mathcal{AMS} bundle is needed for \text. array is needed for the new column type for tabular material. xspace provides “magic” spacing after macros, if requested.

```

35 \RequirePackage{xkeyval}[2005/05/07]
36 \RequirePackage{amstext,array,xspace}

```

\si@tempa Some scratch commands are defined; apart from where a known value is carried through, these could contain anything.

```

\si@tempb
\si@tempc 37 \newcommand*\si@tempa{}
38 \newcommand*\si@tempb{}
39 \newcommand*\si@tempc{}

```

\ifsi@switch Various items will need a switch. To avoid name pollution, a single switch is defined here; grouping will keep the definition local.

```

40 \newif\ifsi@switch

```

\si@tempboxa Some boxes are also needed.

```

\si@tempboxb 41 \newbox\si@tempboxa
\si@tempboxc 42 \newbox\si@tempboxb
\si@tempboxd 43 \newbox\si@tempboxc
44 \newbox\si@tempboxd

```

\si@temptoks A token register is also handy.

```

45 \newtoks\si@temptoks

```

\si@packagecheck As siunitx is intended to replace the other unit-management packages, these are tested for before any further processing. If any are loaded, the package halts compilation; name clashes or unexpected results could occur if this is not tested. Notice that Slunits and Slstyle could be loaded with variable capitalisation (at least on Windows); both possibilities are tested.

```

46 \newcommand*\si@blockpkgs{SIunits,sistyle,siunits,SIstyle,%
47 unitsdef,fancyunits}
48 \newcommand*\si@checkpkgs{units,hepunits,fancynum}
49 \newcommand*\si@packagecheck{%
50 \begingroup

```

```

51 \@for\si@tempa:=\si@blockpkgs\do{
52   \@ifpackageloaded{\si@tempa}
53     {\PackageError{siunitx}
54       {Package '\si@tempa' incompatible}
55       {The \si@tempa\space package and siunitx are
56         incompatible.\MessageBreak Use the
57         'emulate=\si@tempa' package option when loading
58         siunitx}}
59   {}}

```

Some packages should not cause a clash, but are emulated and would be better handled that way.

```

60 \@for\si@tempa:=\si@checkpkgs\do{%
61   \@ifpackageloaded{\si@tempa}
62     {\PackageWarning{siunitx}
63       {Consider loading the siunitx package
64         with\MessageBreak option 'emulate=\si@tempa', rather
65         than\MessageBreak loading both \si@tempa\space and
66         siunitx}}
67     {}}
68 \endgroup}

```

The check is carried out on loading and at the beginning of the document, so that packages loaded both before and after siunitx are caught.

```

69 \si@packagecheck
70 \AtBeginDocument{\si@packagecheck}

```

`\si@ifdefinable` #1 : macro

Using `\@ifdefinable` to check macro definitions gives a generic error. To give something more helpful, `\@ifundefined` is used, but this needs some `\expandafter` work. This way it can also be used as a form of `\@ifundefined` for macro names.

```

71 \newcommand*\si@ifdefinable}[1]{%
72   \expandafter\expandafter\expandafter\@ifundefined%
73     \expandafter\expandafter\expandafter%
74     {\expandafter\gobble\string#1}}

```

`\si@addtolist` #1 : macro
#2 : items

It is quite useful to be able to add to a comma-separated list of expandable items.

```

75 \newcommand*\si@addtolist}[2]{%
76   \ifx\@empty#1\@empty
77     \edef#1{#2}%
78   \else
79     \edef#1{#1,#2}%
80   \fi}

```

`\si@addtociname` #1 : ciname
#2 : tokens

A second item to add to a command sequence.

```

81 \newcommand*\si@addtociname}[2]{%
82   \@ifundefined{#1}
83     {\expandafter\gdef\ciname #1\endciname{#2}}
84     {\si@temptoks\expandafter\expandafter\expandafter{#2}}

```

```

85         \csname #1\endcsname#2}%
86         \expandafter\xdef\csname #1\endcsname{\the\si@temptoks}}

\si@ifmtarg    To keep down dependance on other packages, the very short code block from
\si@xifmtarg    ifmtarg is copied here with an internal name.
\si@ifnotmtarg 87 \begingroup
88     \catcode`\Q=3
89     \long\gdef\si@ifmtarg#1{%
90         \si@xifmtarg#1QQ\@secondoftwo\@firstoftwo\@nil}
91     \long\gdef\si@xifmtarg#1#2Q#3#4#5\@nil{#4}
92     \long\gdef\si@ifnotmtarg#1{%
93         \si@xifmtarg#1QQ\@firstofone\@gobble\@nil}
94 \endgroup

\si@newrobustcmd #1 : macro
\si@newcommand [#2]: num-args
\si@newcmd [#3]: default
\si@xargdef #4 : def
    Some more copying, this time from etoolbox. Various macros need to be really
    robust. This is achieved using the  $\epsilon$ -TeX \protected primitive in various places.
    However, it would be nice to have a \protected version of \newcommand.
    etoolbox has code for that, but to avoid needing to load it, the necessary stuff
    is copied here. The only changes from the original are names, and the use of
    \newcommand* for the \si@newcommand macro.

95 \ifpackageloaded{etoolbox}
96   {\let\si@newrobustcmd\newrobustcmd}
97   {\protected\def\si@newrobustcmd{%
98       \ifstar
99         {\let\l@ngrel@x\protected\si@newcommand}
100        {\def\l@ngrel@x{\protected\long}\si@newcommand}}
101   \newcommand*\si@newcommand}[1]{\@testopt{\si@newcmd#1}0}
102   \def\si@newcmd#1[#2]{%
103       \@ifnextchar[%
104         {\si@xargdef#1[#2]}
105         {\@argdef#1[#2]}}
106   \long\def\si@xargdef#1[#2][#3]#4{%
107       \@ifdefinable#1{%
108         \expandafter\protected
109         \expandafter\def
110         \expandafter#1%
111         \expandafter{%
112             \expandafter\@testopt
113             \csname\string#1\endcsname{#3}}}%
114       \expandafter\@yargdef
115       \csname\string#1\endcsname\tw@{#2}{#4}}}
```

21.2 Logging

```

\ifsi@debug    To control logging, some new switches are declared.
\ifsi@logmin 116 \newif\ifsi@debug
\ifsi@lognone 117 \newif\ifsi@logmin
118 \newif\ifsi@lognone
```

`\si@log@err` #1 : **error**
#2 : **explanation**
Some handy re-usable macros are defined here. These all take names beginning
These pop up in various places. First errors are handled.

```

119 \newcommand*{\si@log@err}[2]{%
120   \ifsi@lognone\else
121     \ifsi@logmin
122       \PackageWarning{siunitx}{#1}%
123     \else
124       \PackageError{siunitx}{#1}{#2}%
125     \fi
126   \fi}

```

`\si@log@warn` #1 : **text**
`\si@log@inf` Then warnings and information.

```

127 \newcommand*{\si@log@warn}[1]{%
128   \ifsi@lognone\else
129     \ifsi@logmin\else
130       \PackageWarning{siunitx}{#1}%
131     \fi
132   \fi}
133 \newcommand*{\si@log@inf}[1]{%
134   \ifsi@lognone\else
135     \ifsi@logmin\else
136       \PackageInfo{siunitx}{#1}%
137     \fi
138   \fi}

```

`\si@log@debug` #1 : **debug-info**
The debug macro only gives output if the appropriate package option is set.

```

139 \newcommand*{\si@log@debug}[1]{%
140   \ifsi@lognone\else
141     \ifsi@debug
142       \PackageInfo{siunitx}{#1}%
143     \fi
144   \fi}

```

21.3 String comparison

`\si@str@ifchrstr` #1 : **char**
`\si@str@chrstr` #2 : **string**

At various points, the package needs to compare two strings, to find if one occurs in the other. The first test is if a single character is part of a second string; this is used, for example, to check that a character is valid as input. The first argument is not expanded further, but the second is two allow division into individual units.

```

145 \newcommand*{\si@str@ifchrstr}[2]{%
146   \begingroup
147     \si@switchfalse
148     \renewcommand*{\si@tempa}{#1}%
149     \protected@edef\si@tempb{#2}%
150     \expandafter\si@str@chrstr\si@tempb\@empty\@empty\@empty

```



```

151     \ifsi@switch
152     \aftergroup\@firstoftwo
153     \else
154     \aftergroup\@secondoftwo
155     \fi
156 \endgroup}
157 \def\si@str@chrstr#1#2\@empty{%
158   \renewcommand*\si@tempc{#1}%
159   \ifx\si@tempa\si@tempc
160     \expandafter\si@switchtrue
161   \else
162     \ifx\@empty#2\@empty\else
163       \si@str@chrstr#2\@empty\@empty
164     \fi
165   \fi}

```

```

\si@str@ifonlychrs #1 : string
\si@str@onlychrs  #2 : chars

```

The second test builds on the first. Here, a check is made to see if the first string is made up only of characters from the second string. In this case, the first string is expanded before testing. The second string will be expanded by the internal character by character test.

```

166 \newcommand*\si@str@ifonlychrs}[2]{%
167   \begingroup
168     \si@switchtrue
169     \protected@edef\si@tempa{#1}%
170     \renewcommand*\si@tempb{#2}%
171     \expandafter\si@str@onlychrs\si@tempa\@empty\@empty\@empty
172     \ifsi@switch
173     \aftergroup\@firstoftwo
174     \else
175     \aftergroup\@secondoftwo
176     \fi
177   \endgroup}
178 \def\si@str@onlychrs#1#2\@empty{%
179   \si@str@ifchrstr{#1}{\si@tempb}
180   {}{\si@switchfalse}%
181   \ifx\@empty#2\@empty\else
182     \si@str@onlychrs#2\@empty\@empty
183   \fi}

```

21.4 Option handling

```
\sisetup #1 : options
```

To allow modification of options at run time, a setup macro is provided.

```
184 \newcommand*\sisetup{\setkeys{si}{key}}
```

```

\si@opt@key #1 : keyname
            #2 : code

```

To aid maintenance, some shortcuts are defined for generating keys. These also allow the debugging messages to be added automatically to every key. First of all the basic key definition.

```

185 \newcommand*{\si@opt@key}[2]{%
186   \define@key[si]{key}{#1}
187   {#2\si@log@debug{Option #1 set to ##1}}}

```

`\si@opt@cmdkey` [#1]: default
 #2 : keyname
 #3 : function

For a single command key, a function must be specified.

```

188 \newcommand*{\si@opt@cmdkey}[3][[]]{%
189   \define@cmdkey[si]{key}[si@]{#2}[#1]{#3}}

```

`\si@opt@cmdkeys` [#1]: default
 #2 : keyname

Whereas multiple definitions do not take a function.

```

190 \newcommand*{\si@opt@cmdkeys}[2][[]]{%
191   \define@cmdkeys[si]{key}[si@]{#2}[#1]}

```

`\si@opt@boolkey` [#1]: function
 #2 : keyname

Keys which only take switch values; anything other than true or false will generate a warning from `xkeyval`.

```

192 \newcommand*{\si@opt@boolkey}[2][[]]{%
193   \define@boolkey[si]{key}[si@]{#2}[true]
194   {#1\si@log@debug{Option #2 set to ##1}}}

```

`\si@opt@choicekey` [#1]: default
 #2 : keyname
 #3 : choices
 #4 : function

A “fill in the blanks” choice key. In all cases, `\si@tempa` is used to hold the value given to the key, so that `\ifx` testing can occur.

```

195 \newcommand*{\si@opt@choicekey}[4][[]]{%
196   \define@choicekey*+[si]{key}{#2}[\si@tempa]{#3}[#1]
197   {#4\si@log@debug{Option #2 set to ##1}}
198   {\si@log@warn{Unknown value ‘##1’ for option #2}}}

```

`\si@opt@xchoicekey` #1 : keyname
 #2 : choices
 #3 : initial-value

Several of the package options can take either a choice from a list of known options, or a value to be interpreted literally. To aid maintenance, the necessary code can be set up here. These keys all define a new macro, which must exist. The `\si@opt@xchoicekey` macro therefore ensures that this is defined, as well as setting up the `xkeyval` key.

```

199 \newcommand*{\si@opt@xchoicekey}[3]{%
200   \define@choicekey*+[si]{key}{#1}[\si@tempa]{#2}[#1]

```

This code will execute if the option is on the list. There will be a “fixed” macro with a matching name, which is used for this.

```

201   {\si@log@debug{Option #1 set to ##1}}%
202   \expandafter\renewcommand\expandafter*\expandafter{%
203     \csname si@#1\endcsname}{\@nameuse{si@fix@##1}}}

```

The user has given something that is not on the list as an argument. It is used literally.

```
204      {\si@log@debug{Option #1 set to ##1}%
205      \expandafter\renewcommand\expandafter*\expandafter{%
206      \csname si@#1\endcsname}{##1}}
```

Finally, the initial value of the macro is set up.

```
207      \expandafter\newcommand\expandafter*\expandafter{%
208      \csname si@#1\endcsname}%
209      {\@nameuse{si@fix@#3}}}
```

```
\si@opt@compatkey #1 : package
                  #2 : keyname
```

An all-in-one definition for a back-compatibility key. These should only be used at load time, so are automatically disabled once the package is loaded. Emulation is also automatically turned on.

```
210 \newcommand*{\si@opt@compatkey}[2]{%
211   \define@boolkey[si]{key}[si@old@]{#2}[true]
212   {\si@log@debug{Emulating #1 package option\MessageBreak #2}%
213   \sisetup{emulate=#1}%
214   \si@log@debug{Option #2 set to ##1}}
215   \AtEndOfPackage{\si@opt@disablekey{#2}
216   {Compatibility option #2 only\MessageBreak
217   available when loading siunitx package}}}
```

```
\si@opt@disablekey #1 : keyname
                   #2 : text
```

The ability to disable a key with a meaningful message is a must; the warning will come from siunitx, and not from xkeyval.

```
218 \newcommand*{\si@opt@disablekey}[2]{%
219   \key@ifundefined[si]{key}{#1}
220   {}
221   {\si@log@debug{Disabling key #1}%
222   \si@opt@key{#1}{\si@log@warn{#2}}}}
```

The xkeyval package option for logging is declared. This is then processed to set the switches correctly.

```
223 \si@opt@choicekey[normal]{log}{debug,verbose,normal,errors,none}
```

A series of comparisons are made to assign the logging mode. The normal option is not tested, as executing the option sets the switches appropriately.

```
224 {\si@debugfalse
225   \si@logminfalse
226   \si@lognonefalse
227   \renewcommand*{\si@tempb}{none}%
228   \ifx\si@tempa\si@tempb
229     \si@lognonetrue
230   \fi
231   \renewcommand*{\si@tempb}{minimal}%
232   \ifx\si@tempa\si@tempb
233     \si@logmintrue
234   \fi
235   \renewcommand*{\si@tempb}{debug}%
```

```

236 \ifx\si@tempa\si@tempb
237 \si@debugtrue
238 \fi
239 \renewcommand*\si@tempb}{verbose}%
240 \ifx\si@tempa\si@tempb
241 \si@debugtrue
242 \fi}

```

A quick method to set log=debug.

```

243 \si@opt@boolkey{debug}

```

`\ifsi@strict` It would be useful to be able to disable some keys, when strict interpretation of the rules is desired. This is a load-time option, and has to disable various options.

```

244 \si@opt@boolkey[%
245 \ifsi@strict
246 \sisetup{
247 obeymode=false,
248 obeybold=false,
249 obeyitalic=false,
250 mode=maths,
251 repeatunits=true,
252 trapambigerr=true,
253 trapambigfrac=true}
254 \@for\si@tempa:=obeyall,obeymode,obeyitalic,mode,unitmode,%
255 valuemode,textmode,obeybold,repeatunits,trapambigerr,%
256 trapambigfrac\do{%
257 \begingroup
258 \edef\si@tempb{\endgroup
259 \noexpand\si@opt@disablekey{\si@tempa}
260 {Option '\si@tempa' forbidden in strict mode}}}%
261 \si@tempb}
262 \fi]{strict}
263 \AtEndOfPackage{
264 \si@opt@disablekey{strict}
265 {Option 'strict' only available when\MessageBreak
266 loading package}}

```

`\si@emulate` The emulate option is used for back-compatibility mode; the option is only valid when loading siunitx.

```

267 \newcommand*\si@emulate{}
268 \si@opt@key{emulate}{\si@addtolist{\si@emulate}{#1}}
269 \AtEndOfPackage{
270 \si@opt@disablekey{emulate}
271 {Option 'emulate' only available when\MessageBreak
272 loading package}}

```

`\si@unitsep` `\si@unitspace` `\si@errspace` `\si@valuesep` The two . . . sep options control the size of spaces between the number and the unit (`\si@valuesep`), and that used to represent a product (`\si@unitsep`). Known values here are thin, med, medium, thick, cdot, tightcdot⁵⁰ and none;⁵¹ other entries will be treated as custom spaces.

⁵⁰Both `\cdot`-based options only valid for unitsep.

⁵¹Only valid for valuesep.

```

273 \si@opt@xchoicekey{unitsep}
274 {thin,med,medium,thick,none,comma,stop,fullstop,period,
275 times,tighttimes,cdot,tightcdot}{thin}
276 \si@opt@xchoicekey{unitSPACE}{space,thin,med,medium,thick,
277 none}{thin}
278 \si@opt@xchoicekey{errSPACE}{space,thin,med,medium,thick,
279 none}{none}
280 \si@opt@xchoicekey{valueSEP}
281 {thin,med,medium,thick,none,comma,stop,fullstop,period,
282 times,tighttimes,cdot,tightcdot}{thin}

```

`\si@digitsep` Separation of digits in large numbers is controlled by the `digitsep` option. As with the other `sep` values, this one has a choice of possible values. The list is quite long, so that a range of options are handled automatically. Notice that `digitsep=none` will be used for no separation at all.

```

283 \si@opt@xchoicekey{digitsep}
284 {thin,med,medium,thick,none,comma,stop,fullstop,period,
285 times,tighttimes,cdot,tightcdot}{thin}

```

`\si@decimalsymbol` The symbol used for the decimal position is varied here. There are only two real options, but options are given for the name of a full stop.

```

286 \si@opt@xchoicekey{decimalsymbol}{comma,stop,fullstop,period,
287 cdot,tightcdot}{fullstop}

```

`\si@anglesep` The separator between degrees and minutes, and between minutes and seconds, when using `\ang`.

```

288 \si@opt@xchoicekey{anglesep}
289 {thin,med,medium,thick,none,comma,stop,fullstop,period,
290 times,tighttimes,cdot,tightcdot}{none}

```

`\ifsi@obeymode` The first test for the font control is whether to respect the surrounding maths or text mode.

```

291 \si@opt@boolkey{obeymode}

```

`\ifsi@numtextmode` The output of the package can be typeset using either text or maths mode fonts.
`\ifsi@unittextmode` Two switches are needed, for numbers and units.

```

292 \newif\ifsi@numtextmode
293 \newif\ifsi@unittextmode

```

The `textmode` option has to set both flags.

```

294 \si@opt@choicekey[true]{textmode}{true,false}
295 {\si@numtextmodefalse
296 \si@unittextmodefalse
297 \renewcommand*{\si@tempb}{true}%
298 \ifx\si@tempa\si@tempb
299 \si@numtextmodetrue
300 \si@unittextmodetrue
301 \fi}

```

The `mode` option applies to numbers and units.

```

302 \si@opt@choicekey{mode}{math,maths,text}
303 {\si@numtextmodefalse
304 \si@unittextmodefalse

```

```

305 \renewcommand*{\si@tempb}{text}%
306 \ifx\si@tempa\si@tempb
307   \si@numtextmodetrue
308   \si@unittextmodetrue
309 \fi}

```

Now the two options for numbers or units alone.

```

310 \si@opt@choicekey{valuemode}{math, maths, text}
311 {\si@numtextmodetrue
312 \renewcommand*{\si@tempb}{text}%
313 \ifx\si@tempa\si@tempb
314   \si@numtextmodetrue
315 \fi}
316 \si@opt@choicekey{unitmode}{math, maths, text}
317 {\si@unittextmodetrue
318 \renewcommand*{\si@tempb}{text}%
319 \ifx\si@tempa\si@tempb
320   \si@unittextmodetrue
321 \fi}

```

`\ifsi@obeyfamily` The package can work to match the font family (serif, sans serif, typewriter) of the surrounding text. This is controlled by a Boolean option.

```
322 \si@opt@boolkey{obeyfamily}
```

`\ifsi@obeybold` The package can attempt to respect bold, or may ignore it.

```
323 \si@opt@boolkey{obeybold}
```

`\ifsi@inlinebtext` For inline maths, two options for checking what is bold are available, the maths environment (*i.e.* `\boldmath`) and the surrounding text (`\textbf` or `\bfffamily`).

```

324 \newif\ifsi@inlinebtext
325 \si@opt@choicekey{inlinebold}{text, maths, math}
326 {\si@inlinebtextfalse
327 \renewcommand*{\si@tempb}{text}%
328 \ifx\si@tempa\si@tempb
329   \si@inlinebtexttrue
330 \fi}

```

`\ifsi@obeyitalic` Italic is slightly different to bold, as there is no convenient switch for maths. Thus a choice key is used, with the appropriate check.

```
331 \si@opt@boolkey{obeyitalic}
```

`\ifsi@detectdisplay` For handling display mathematics, a setting is needed for whether to treat it differently from other maths.

```
332 \si@opt@boolkey{detectdisplay}
```

The option to obey all font switching commands is Boolean-like but needs alternative setup.

```

333 \si@opt@choicekey[true]{obeyall}{true, false}
334 {\si@obeyboldfalse
335 \si@obeyitalicfalse
336 \si@obeymodetrue
337 \si@obeyfamilyfalse
338 \renewcommand*{\si@tempb}{true}%

```

```

339 \ifx\si@tempa\si@tempb
340 \si@obeyboldtrue
341 \si@obeyitalictrue
342 \si@obeymodetrue
343 \si@obeyfamilytrue
344 \fi}

```

\si@valuemathsrms The fonts used by the package default to the obvious L^AT_EX ones; however, this needs to be exposed to user modification. First the maths mode fonts are sorted out.

```

\si@unitmathsrms 345 \si@opt@cmdkeys{valuemathsrms,valuemathsssf,valuemathstt}
\si@unitmathsssf 346 \si@opt@cmdkeys{unitmathsrms,unitmathsssf,unitmathstt}
\si@unitmathstt

```

To make life easier for the user, US spellings are provided for the maths keys.

```

347 \si@opt@key{valuemathrm}{\sisetup{valuemathsrms=#1}}
348 \si@opt@key{valuemathssf}{\sisetup{valuemathsssf=#1}}
349 \si@opt@key{valuemathstt}{\sisetup{valuemathstt=#1}}
350 \si@opt@key{unitmathrm}{\sisetup{unitmathsrms=#1}}
351 \si@opt@key{unitmathssf}{\sisetup{unitmathsssf=#1}}
352 \si@opt@key{unitmathstt}{\sisetup{unitmathstt=#1}}

```

The combined options are given, for setting numbers and units at the same time.

```

353 \si@opt@key{mathsrms}{\sisetup{valuemathsrms=#1,unitmathsrms=#1}}
354 \si@opt@key{mathsssf}{\sisetup{valuemathsssf=#1,unitmathsssf=#1}}
355 \si@opt@key{mathstt}{\sisetup{valuemathstt=#1,unitmathstt=#1}}
356 \si@opt@key{mathrm}{\sisetup{valuemathsrms=#1,unitmathsrms=#1}}
357 \si@opt@key{mathssf}{\sisetup{valuemathsssf=#1,unitmathsssf=#1}}
358 \si@opt@key{mathstt}{\sisetup{valuemathstt=#1,unitmathstt=#1}}

```

\si@valuetextrms The same thing for text mode fonts. Once again the default values are pretty obvious.

```

\si@valuetextsf
\si@valuetexttt 359 \si@opt@cmdkeys{valuetextrms,valuetextsf,valuetexttt}
\si@unittextrms 360 \si@opt@cmdkeys{unittextrms,unittextsf,unittexttt}
\si@unittextsf 361 \si@opt@key{textrms}{\sisetup{unittextrms=#1,valuetextrms=#1}}
\si@unittexttt 362 \si@opt@key{textsf}{\sisetup{unittextsf=#1,valuetextsf=#1}}
363 \si@opt@key{texttt}{\sisetup{unittexttt=#1,valuetexttt=#1}}

```

\si@numdigits The list of possible valid characters for parsing numbers is set up. This is similar to numprint, but with the extra class, and with characters ignored with no output renamed as gobble.

```

\si@numdecimal
\si@numexp
\si@numprod 364 \si@opt@cmdkeys{numdigits,numdecimal,numexp,numgobble,numsign,%
\si@numgobble 365 numcloseerr,numopenerr,numaddn,numprod}

```

\si@numsign The various valid characters are collected together in a single macro for later. In common with the above macros, this one starts \si@num. . . . The order here is the order the values are tested later on.

```

\si@numcloseerr
\si@numopenerr
\si@numaddn
366 \newcommand*{\si@numextra}{%
367 \si@numopenerr\si@numcloseerr\si@numaddn}
368 \newcommand*{\si@numvalid}{\si@numgobble\si@numexp\si@numsign
369 \si@numdecimal\si@numdigits\si@numextra\si@numprod}

```

\ifsi@seperr An option to control whether numerical error values are printed with or separate from the number.

```

\ifsi@trapambigerr
\si@openerr
\si@closeerr

```

```

370 \si@opt@boolkey{seperr}
371 \si@opt@boolkey{trapambigerr}
372 \si@opt@cmdkeys{openerr,closeerr}

\ifsi@sepfour   With four digits in a number, separating may or may not be desired. Note that
                 this option is the same as one for numprint.
373 \si@opt@boolkey{sepfour}

\ifsi@retainplus   An option to keep an explicit positive sign.
374 \si@opt@boolkey{retainplus}

\si@expbase   The options for exponents are set up.
\si@expproduct 375 \si@opt@xchoicekey{expproduct}{times,tighttimes,
376   cdot,tightcdot}{times}
377 \si@opt@xchoicekey{expbase}{ten}{ten}

\ifsi@allowzeroexp   The package normally prevents  $10^0$ .
378 \si@opt@boolkey{allowzeroexp}

\si@prefixproduct   The marker for multiplication in prefixes.
379 \si@opt@xchoicekey{prefixproduct}{times,tighttimes,cdot,
380   tightcdot,none}{times}

\si@prefixbase   In the same area, the power for prefixes is variable. Here, two choices are needed.
381 \si@opt@xchoicekey{prefixbase}{ten,two}{ten}

\ifsi@prefixsymbolic   Unit prefixes can be given as either symbols or numerically.
382 \si@opt@boolkey{prefixsymbolic}

\ifsi@num@padlead   A setting is needed to indicate when to add zeros to decimal numbers, either
\ifsi@num@padtrail   before the decimal marker (. 1 giving “0.1”) or after (1 . giving “1.0”).
383 \newif\ifsi@num@padlead
384 \newif\ifsi@num@padtrail
385 \si@opt@choicekey[all]{padnumber}
386   {leading,lead,trailing,trail,all,both,true,none,false}
387   {\si@num@padleadfalse
388     \si@num@padtrailfalse
389     \renewcommand*{\si@tempb}{leading}%
390     \ifx\si@tempa\si@tempb
391       \si@num@padleadtrue
392     \fi
393     \renewcommand*{\si@tempb}{lead}%
394     \ifx\si@tempa\si@tempb
395       \si@num@padleadtrue
396     \fi
397     \renewcommand*{\si@tempb}{trailing}%
398     \ifx\si@tempa\si@tempb
399       \si@num@padtrailtrue
400     \fi
401     \renewcommand*{\si@tempb}{trail}%
402     \ifx\si@tempa\si@tempb
403       \si@num@padtrailtrue
404     \fi

```



```

405 \renewcommand*{\si@tempb}{all}%
406 \ifx\si@tempa\si@tempb
407   \si@num@padleadtrue
408   \si@num@padtrailtrue
409 \fi
410 \renewcommand*{\si@tempb}{true}%
411 \ifx\si@tempa\si@tempb
412   \si@num@padleadtrue
413   \si@num@padtrailtrue
414 \fi
415 \renewcommand*{\si@tempb}{both}%
416 \ifx\si@tempa\si@tempb
417   \si@num@padleadtrue
418   \si@num@padtrailtrue
419 \fi}

```

\si@sign **Some new switches for adding signs to numbers**

```

\ifsi@num@signmant 420 \newif\ifsi@num@signmant
\ifsi@num@signexp 421 \newif\ifsi@num@signexp

```

Signs can be added to numbers by default. Two options are needed here; whether to add a sign by default, and what the sign is.

```

422 \si@opt@xchoicekey{sign}{plus,minus,pm,mp}{plus}
423 \si@opt@choicekey[all]{addsign}
424 {mantissa,exponent,mant,exp,all,both,true,none,false}

```

The option is now processed.

```

425 {\si@num@signmantfalse
426   \si@num@signexpfalse
427   \renewcommand*{\si@tempb}{mantissa}%
428   \ifx\si@tempa\si@tempb
429     \si@num@signmanttrue
430   \fi
431   \renewcommand*{\si@tempb}{mant}%
432   \ifx\si@tempa\si@tempb
433     \si@num@signmanttrue
434   \fi
435   \renewcommand*{\si@tempb}{exponent}%
436   \ifx\si@tempa\si@tempb
437     \si@num@signexptrue
438   \fi
439   \renewcommand*{\si@tempb}{exp}%
440   \ifx\si@tempa\si@tempb
441     \si@num@signexptrue
442   \fi
443   \renewcommand*{\si@tempb}{all}%
444   \ifx\si@tempa\si@tempb
445     \si@num@signmanttrue
446     \si@num@signexptrue
447   \fi
448   \renewcommand*{\si@tempb}{true}%
449   \ifx\si@tempa\si@tempb
450     \si@num@signmanttrue
451     \si@num@signexptrue

```

```

452 \fi
453 \renewcommand*{\si@tempb}{both}%
454 \ifx\si@tempa\si@tempb
455     \si@num@signmanttrue
456     \si@num@signexptrue
457 \fi}

```

`\ifsi@tightpm` To reduce spacing, it might be necessary to use a “tight” \pm sign.

```

\si@pm 458 \si@opt@boolkey{tightpm}
459 \newcommand*{\si@pm}{%
460     \ifsi@tightpm
461         \si@fix@tightpm
462     \else
463         \si@fix@pm
464 \fi}

```

`\ifsi@ang@padsmall` A switch for determining whether to typeset `\ang{; ; 1}` as $0^{\circ}0'1''$ or $1''$. First,
`\ifsi@ang@padlarge` two new Boolean switches are needed to indicate padding.

```

465 \newif\ifsi@ang@padsmall
466 \newif\ifsi@ang@padlarge
467 \si@opt@choicakey[all]{padangle}
468 {small,large,all,both,true,none,false}
469 {\si@ang@padsmallfalse
470  \si@ang@padlargefalse
471  \renewcommand*{\si@tempb}{small}%
472  \ifx\si@tempa\si@tempb
473      \si@ang@padsmalltrue
474  \fi
475  \renewcommand*{\si@tempb}{large}%
476  \ifx\si@tempa\si@tempb
477      \si@ang@padlargetrue
478  \fi
479  \renewcommand*{\si@tempb}{all}%
480  \ifx\si@tempa\si@tempb
481      \si@ang@padsmalltrue
482      \si@ang@padlargetrue
483  \fi
484  \renewcommand*{\si@tempb}{true}%
485  \ifx\si@tempa\si@tempb
486      \si@ang@padsmalltrue
487      \si@ang@padlargetrue
488  \fi
489  \renewcommand*{\si@tempb}{both}%
490  \ifx\si@tempa\si@tempb
491      \si@ang@padsmalltrue
492      \si@ang@padlargetrue
493  \fi}

```

`\ifsi@ang@toarc` To control whether angles are formatted as decimals or degrees–minutes–seconds,
`\ifsi@ang@toddec` a package option plus two switches are needed. The later format is referred to as
“arc” most readily. An option to leave the input unchanged is also provided.

```

494 \newif\ifsi@ang@toarc
495 \newif\ifsi@ang@toddec
496 \si@opt@choicakey[all]{angformat}

```

```

497 {dec,decimal,arc,dms,unchanged,none}
498 {\si@ang@toarcfalse
499 \si@ang@todectfalse
500 \renewcommand*{\si@tempb}{dec}%
501 \ifx\si@tempa\si@tempb
502 \si@ang@todecttrue
503 \fi
504 \renewcommand*{\si@tempb}{decimal}%
505 \ifx\si@tempa\si@tempb
506 \si@ang@todecttrue
507 \fi
508 \renewcommand*{\si@tempb}{arc}%
509 \ifx\si@tempa\si@tempb
510 \si@ang@toarctrue
511 \fi
512 \renewcommand*{\si@tempb}{dms}%
513 \ifx\si@tempa\si@tempb
514 \si@ang@toarctrue
515 \fi}

```

`\ifsi@astroang` A slightly odd option to allow the method used by astronomers for angles.

```
516 \si@opt@boolkey{astroang}
```

`\ifsi@strictarc` For the `\ang` macro, the default is to require two semi-colons in the input for arc angles. This is controlled here.

```
517 \si@opt@boolkey{strictarc}
```

`\ifsi@tab@fixed` To determine the control of table alignment, two options are provided. The `\si@tabnumalign` option controls which centring method is used, and the fills used for achieving this.

```

\si@tab@rfill@S \si@tab@lfill@S
\si@tab@lfill@S 518 \newif\ifsi@tab@fixed
519 \si@opt@choicakey{tabnumalign}
520 {centredecimal,centerdecimal,right,left,centre,center}
521 {\si@tab@fixedtrue
522 \let\si@tab@rfill@S\hfil
523 \let\si@tab@lfill@S\hfil
524 \renewcommand*{\si@tempb}{right}%
525 \ifx\si@tempa\si@tempb
526 \let\si@tab@lfill@S\hfill
527 \fi
528 \renewcommand*{\si@tempb}{left}%
529 \ifx\si@tempa\si@tempb
530 \let\si@tab@rfill@S\hfill
531 \fi
532 \renewcommand*{\si@tempb}{centredecimal}%
533 \ifx\si@tempa\si@tempb
534 \expandafter\si@tab@fixedfalse
535 \fi
536 \renewcommand*{\si@tempb}{centerdecimal}%
537 \ifx\si@tempa\si@tempb
538 \expandafter\si@tab@fixedfalse
539 \fi}
540 \si@opt@key{tabalign}{\sisetup{tabnumalign=#1,tabtextalign=#1,
541 tabunitalign=#1}}

```

```

\ifsi@tabalignexp  A switch for aligning exponents.
542 \si@opt@boolkey{tabalignexp}

\si@tab@mantprecnt  To process the format information, various internal number-processing macros
\si@tab@mantpostcnt are used. First, some storage areas are created.
\si@tab@expprecnt 543 \newcount\si@tab@mantprecnt
\si@tab@exppostcnt 544 \newcount\si@tab@mantpostcnt
\ifsi@tab@mantsign 545 \newcount\si@tab@expprecnt
\ifsi@tab@expsign 546 \newcount\si@tab@exppostcnt
547 \newif\ifsi@tab@mantsign
548 \newif\ifsi@tab@expsign

```

The input is split into a mantissa and exponent, then passed to a re-useable macro for further processing.

```

549 \si@opt@cmdkey{tabformat}
550 {\si@num@fixpm
551 \renewcommand*{\si@num@arg}{tabformat data}%
552 \renewcommand*{\si@num@exp}{}%
553 \renewcommand*{\si@num@mant}{}%
554 \si@tab@mantsignfalse
555 \si@tab@expsignfalse
556 \si@switchfalse
557 \si@num@sepmantexp{#1}%

```

When checking for a sign, the internal flag for finding but deleting a plus sign is used.

```

558 \si@num@sepsign{mant}%
559 \ifx\@empty\si@num@mantsign\@empty
560 \ifsi@num@delplus
561 \expandafter\expandafter\expandafter\si@tab@mantsigntrue
562 \fi
563 \else
564 \expandafter\si@tab@mantsigntrue
565 \fi
566 \si@num@sepsign{exp}%
567 \ifx\@empty\si@num@expsign\@empty
568 \ifsi@num@delplus
569 \expandafter\expandafter\expandafter\si@tab@expsigntrue
570 \fi
571 \else
572 \expandafter\si@tab@expsigntrue
573 \fi
574 \si@opt@proctform{mant}%
575 \si@opt@proctform{exp}%

```

If both the integer and decimal parts are empty, then something has probably gone wrong.

```

576 \ifnum\si@tab@mantpostcnt=\z@\relax
577 \ifnum\si@tab@mantprecnt=\z@\relax
578 \si@log@err{Empty mantissa argument for tabformat}
579 {The argument '#1' contains no valid entry for
580 a mantissa\MessageBreak It should be of the
581 form 'm.n', where m and n are integers}%
582 \fi
583 \fi

```

If `tabformat` has been given with `tabnumalign=centredecimal` active, then the alignment is changed to centred.

```
584 \ifsi@tab@fixed\else
585   \sisetup{tabnumalign=centre}%
586 \fi
587 \let\pm\si@num@pm
588 \let\mp\si@num@mp}
```

`\si@opt@proctform` #1 : either mant or exp

Processing the number further uses the `\si@num@digits` macro. The results are stored in the appropriate counter.

```
589 \newcommand*{\si@opt@proctform}[1]{%
590   \renewcommand*{\si@num@predec}{}%
591   \renewcommand*{\si@num@postdec}{}%
592   \si@switchfalse
593   \expandafter\si@ifnotmtarg\expandafter{%
594     \csname si@num@#1\endcsname}
595     {\expandafter\expandafter\expandafter\si@num@digits
596       \csname si@num@#1\endcsname\@empty\@empty}%
597   \csname si@tab@#1precnt\endcsname\z@\relax
598   \csname si@tab@#1postcnt\endcsname\z@\relax
599   \ifx\@empty\si@num@predec\@empty\else
600     \csname si@tab@#1precnt\endcsname\si@num@predec\relax
601   \fi
602   \ifx\@empty\si@num@postdec\@empty\else
603     \csname si@tab@#1postcnt\endcsname\si@num@postdec\relax
604   \fi}
```

The alignment of tabular material when not processed by `\num` needs to be available.

```
605 \si@opt@choicakey{tabtextalign}{left,right,centre,center}
```

`\si@tab@rfill@t` By default, centring happens on both sides of the content of tabular material.

```
\si@tab@lfill@t 606 {\let\si@tab@rfill@t\hfill
607   \let\si@tab@lfill@t\hfill
608   \renewcommand*{\si@tempb}{right}%
609   \ifx\si@tempa\si@tempb
610     \let\si@tab@rfill@t\relax
611   \fi
612   \renewcommand*{\si@tempb}{left}%
613   \ifx\si@tempa\si@tempb
614     \let\si@tab@lfill@t\relax
615   \fi}
```

The alignment of unit columns for tabular material has a similar control option.

```
616 \si@opt@choicakey{tabunitalign}{left,right,centre,center}
```

`\si@tab@rfill@s` By default, centring happens on both sides of the content of tabular material.

```
\si@tab@lfill@s 617 {\let\si@tab@rfill@s\hfill
618   \let\si@tab@lfill@s\hfill
619   \renewcommand*{\si@tempb}{right}%
620   \ifx\si@tempa\si@tempb
621     \let\si@tab@rfill@s\relax
```

```

622 \fi
623 \renewcommand*{\si@tempb}{left}%
624 \ifx\si@tempa\si@tempb
625 \let\si@tab@lfill@s\relax
626 \fi}

```

`\ifsi@fixdp` To allow control of rounding, two options are needed. One sets how many fixed digits to use, the second turns this function on and off.

```

627 \si@opt@boolkey{fixdp}
628 \newcount\si@num@dp
629 \si@opt@key{dp}{%
630 \si@str@ifonlychrs{#1}{0123456789}
631 {}
632 {\si@log@err{Invalid input for dp option}
633 {The dp option must be given a positive integer}}%
634 \si@num@dp#1\relax
635 \si@fixdptrue}

```

`\ifsi@tabautofit` To apply rounding automatically in a table, a separate option is used.

```

636 \si@opt@boolkey{tabautofit}

```

`\ifsi@xspace` Unit macros on their own may need `xspace`.

```

637 \si@opt@boolkey{xspace}

```

`\ifsi@prespace`

```

638 \si@opt@boolkey
639 [\si@unt@numfalse
640 \ifsi@prespace
641 \si@unt@numtrue
642 \fi]
643 {prespace}

```

`\ifsi@allowoptarg` For `unitsdef` users, a method to absorb optional arguments is needed.

```

644 \si@opt@boolkey{allowoptarg}

```

`\ifsi@frac` The option processing for formatting units with `\per` in them needs two switches.

```

\ifsi@slash 645 \newif\ifsi@slash
\ifsi@stickyper 646 \newif\ifsi@frac
647 \si@opt@boolkey{stickyper}
648 \si@opt@choickey[reciprocal]{per}
649 {reciprocal,rp,power,slash,frac,fraction}
650 {\si@slashfalse
651 \si@fracfalse
652 \renewcommand*{\si@tempb}{slash}%
653 \ifx\si@tempa\si@tempb
654 \si@fractrue
655 \si@slashttrue
656 \let\si@frac\si@frc@slash
657 \fi
658 \renewcommand*{\si@tempb}{frac}%
659 \ifx\si@tempa\si@tempb
660 \si@fractrue
661 \fi

```

```

662 \renewcommand*{\si@tempb}{fraction}%
663 \ifx\si@tempa\si@tempb
664 \si@fractrue
665 \fi}

```

`\si@slash` For the slash option, the separator can be customised.

```

666 \si@opt@xchoicekey{slash}{slash}{slash}

```

`\ifsi@repeatunits` An option is needed for cases where units should be repeated.

```

\ifsi@addunitpower 667 \newif\ifsi@repeatunits
668 \newif\ifsi@addunitpower
669 \si@opt@choicekey[true]{repeatunits}{true,false,power}
670 {\si@repeatunitsfalse
671 \si@addunitpowerfalse
672 \renewcommand*{\si@tempb}{true}%
673 \ifx\si@tempa\si@tempb
674 \si@repeatunitstrue
675 \fi
676 \renewcommand*{\si@tempb}{power}%
677 \ifx\si@tempa\si@tempb
678 \si@addunitpowertrue
679 \fi}

```

`\ifsi@trapambigfrac` Macros for the right and left brackets added to potentially ambiguous denominators.

```

\si@closefrac
\si@openfrac 680 \si@opt@boolkey{trapambigfrac}
681 \si@opt@cmdkeys{closefrac,openfrac}

```

In the case of fractional handling of the `\per` operator, further refinement is available.

```

682 \si@opt@choicekey[frac]{fraction}
683 {frac,nicefrac,nice,sfrac,xfrac,uglyfrac,ugly}
684 {\let\si@frac\si@frc@frac
685 \renewcommand*{\si@tempb}{nicefrac}%
686 \ifx\si@tempa\si@tempb
687 \let\si@frac\si@frc@nice
688 \fi
689 \renewcommand*{\si@tempb}{uglyfrac}%
690 \ifx\si@tempa\si@tempb
691 \let\si@frac\si@frc@ugly
692 \fi
693 \renewcommand*{\si@tempb}{nice}%
694 \ifx\si@tempa\si@tempb
695 \let\si@frac\si@frc@nice
696 \fi
697 \renewcommand*{\si@tempb}{sfrac}%
698 \ifx\si@tempa\si@tempb
699 \let\si@frac\si@frc@sfrac
700 \fi
701 \renewcommand*{\si@tempb}{xfrac}%
702 \ifx\si@tempa\si@tempb
703 \let\si@frac\si@frc@sfrac
704 \fi

```

```

705 \renewcommand*{\si@tempb}{ugly}%
706 \ifx\si@tempa\si@tempb
707 \let\si@frac\si@frc@ugly
708 \fi}

```

`\si@load` Loading of support files is controlled by two keys. The first defines a list of files that may be loaded, the second a list that will not. This makes it easy to exclude a single file from a long list.

```

709 \si@opt@cmdkeys{load,noload}
710 \si@opt@key{alsoload}{\si@addtolist{\si@load}{#1}}
711 \AtEndOfPackage{
712 \si@opt@disablekey{load}
713 {Configuration files can only be used\MessageBreak
714 when loading package}
715 \si@opt@disablekey{noload}
716 {Configuration files can only be used\MessageBreak
717 when loading package}}
718 \AtEndOfPackage{
719 \si@opt@key{alsoload}{%
720 \@for\si@tempa:=#1\do{\si@loadfile{\si@tempa}}}}

```

`\ifsi@colourunits` Colour is turned on and off using two switches and the appropriate options. US spellings are also provided.

```

\ifsi@colourvalues
\si@unitcolour 721 \si@opt@boolkey{colourunits}
\si@valuecolour 722 \si@opt@boolkey{colourvalues}
723 \si@opt@choicekey[true]{colorunits}
724 {true,false}
725 {\si@colourunitsfalse
726 \renewcommand*{\si@tempb}{true}%
727 \ifx\si@tempa\si@tempb
728 \si@colourunitstrue
729 \fi}
730 \si@opt@choicekey[true]{colorvalues}
731 {true,false}
732 {\si@colourvaluesfalse
733 \renewcommand*{\si@tempb}{true}%
734 \ifx\si@tempa\si@tempb
735 \si@colourvaluestru
736 \fi}
737 \si@opt@choicekey[true]{colorall}
738 {true,false}
739 {\si@colourvaluesfalse
740 \si@colourunitsfalse
741 \renewcommand*{\si@tempb}{true}%
742 \ifx\si@tempa\si@tempb
743 \si@colourunitstrue
744 \si@colourvaluestru
745 \fi}
746 \si@opt@choicekey[true]{colourall}
747 {true,false}
748 {\si@colourvaluesfalse
749 \si@colourunitsfalse
750 \renewcommand*{\si@tempb}{true}%
751 \ifx\si@tempa\si@tempb

```



```

752     \si@colourunitstrue
753     \si@colourvaluesttrue
754   \fi}
755 \si@opt@cmdkeys{unitcolour,valuecolour}
756 \si@opt@key{unitcolor}{\sisetup{unitcolour=#1}}
757 \si@opt@key{valuecolor}{\sisetup{valuecolour=#1}}
758 \si@opt@key{colour}{\sisetup{unitcolour=#1,valuecolour=#1}}
759 \si@opt@key{color}{\sisetup{unitcolour=#1,valuecolour=#1}}

```

`\ifsi@colourneg` The set up for colouring negative numbers is similar.

```

\si@negcolour 760 \si@opt@boolkey{colourneg}
761 \si@opt@choicakey[true]{colorneg}
762 {true,false}
763 {\si@colournegfalse
764 \renewcommand*{\si@tempb}{true}%
765 \ifx\si@tempa\si@tempb
766 \si@colournegtrue
767 \fi}
768 \si@opt@cmdkeys{negcolour}
769 \si@opt@key{negcolor}{\sisetup{negcolour=#1}}

```

`\si@textOmega` The various non-Latin symbols need to be handled, and given user interfaces.

`\si@mathsOmega` Some definitions are more complex than others; for Ω things are easy.

```

770 \si@opt@cmdkeys{textOmega,mathsOmega}
771 \si@opt@key{mathOmega}{\sisetup{mathsOmega=#1}}
772 \newcommand*{\si@mathsOmega}{\text{\ensuremath{\Omega}}}
773 \newcommand*{\si@textOmega}{\ensuremath{\Omega}}

```

`\si@textmu` For the μ symbol, some direct loading of symbols is needed as the maths mu sign (μ) is wrong.

```

\si@mathsmu 774 \si@opt@cmdkeys{textmu,mathsmu}
775 \si@opt@key{mathmu}{\sisetup{mathsmu=#1}}
776 \DeclareFontEncoding{TS1}{}{}
777 \DeclareFontSubstitution{TS1}{cmr}{m}{n}
778 \DeclareTextSymbol{\si@textmu}{TS1}{181}
779 \DeclareTextSymbolDefault{\si@textmu}{TS1}
780 \ifpackageloaded{upgreek}
781 {}
782 {\DeclareFontFamily{OML}{eur}{\skewchar\font'177}
783 \DeclareFontShape{OML}{eur}{m}{n}{%
784 <-6> eurm5 <6-8> eurm7 <8-> eurm10}{}%
785 \AtBeginDocument{
786 \ifpackageloaded{upgreek}
787 {\let\si@mathsmu\upmu}
788 {\DeclareSymbolFont{si@greek}{OML}{eur}{m}{n}
789 \DeclareMathSymbol{\si@mathsmu}{\mathord}{si@greek}{"16}}}

```

`\si@textdegree` The angle signs.

```

\si@mathsdegree 790 \si@opt@cmdkeys{textdegree,mathsdegree,textminute,mathsminute,
\si@textminute 791 textsecond,mathssecond}
\si@mathsminute 792 \si@opt@key{mathdegree}{\sisetup{mathsdegree=#1}}
\si@textsecond 793 \si@opt@key{mathminute}{\sisetup{mathsminute=#1}}
\si@mathssecond 794 \si@opt@key{mathsecond}{\sisetup{mathssecond=#1}}

```

```

795 \newcommand*{\si@textdegree}{\ensuremath{{}^{\circ}}}
796 \newcommand*{\si@mathsdegree}{{}^{\circ}}
797 \newcommand*{\si@textminute}{\ensuremath{{}^{\prime}}}
798 \newcommand*{\si@mathsminute}{{}^{\prime}}
799 \newcommand*{\si@textsecond}{\ensuremath{{}^{\prime\prime}}}
800 \newcommand*{\si@mathssecond}{{}^{\prime\prime}}

```

`\si@textcelsius` Finally, degrees Celsius, which may need the degree symbol.

```

\si@mathscelsius 801 \si@opt@cmdkeys{textcelsius,mathscelsius}
802 \si@opt@key{mathcelsius}{\sisetup{mathscelsius=#1}}
803 \newcommand*{\si@textcelsius}{%
804   \si@textdegree\kern-\scriptspace C}
805 \newcommand*{\si@mathscelsius}{%
806   \si@mathsdegree\kern-\scriptspace\mathrm{C}}

```

`\si@textringA` The Å sign.

```

\si@mathsringA 807 \si@opt@cmdkeys{textringA,mathsringA}
808 \si@opt@key{mathringA}{\sisetup{mathsringA=#1}}
809 \newcommand*{\si@textringA}{\AA}
810 \newcommand*{\si@mathsringA}{\text{\AA}}

```

`\ifsi@redefsymbols` A flag for using `textcomp` and `upgreek` to provide better symbols.

```

811 \si@opt@boolkey{redefsymbols}
812 \AtBeginDocument{
813   \si@opt@disablekey{redefsymbols}
814   {Symbols can only be redefined\MessageBreak
815     when loading siunitx}}

```

`\si@eVcorra`

```

\si@eVcorrb 816 \newlength\si@eVcorra
817 \newlength\si@eVcorrb
818 \si@opt@key{eVcorra}{\setlength\si@eVcorra{#1}}
819 \si@opt@key{eVcorrb}{\setlength\si@eVcorrb{#1}}

```

`\si@locale` Handling typographic conventions needs three keys. `locale` is used to set the
`\si@loctolang` locale, `loctolang` to bind to `babel`.

```

820 \si@opt@key{locale}{%
821   \si@loc@load{#1}%
822   \si@loc@set{#1}}%
823 \si@opt@key{loctolang}{\si@loc@ltol{#1}}

```

21.5 Compatibility options

`\ifsi@old@ugly` With the options for the package set up, the next stage is to provide support for
`\ifsi@old@nice` users of the older packages. These all set up switches, but do not do anything.
`\ifsi@old@loose` That is left to the emulation files, loaded at the end of the package. First of all,
`\ifsi@old@tight` the units options are dealt with; there are not many.

```

824 \si@opt@compatkey{units}{ugly}
825 \si@opt@compatkey{units}{nice}
826 \si@opt@compatkey{units}{loose}
827 \si@opt@compatkey{units}{tight}

```

```

\ifsi@old@OHM      The unitsdef package is unfortunately much more profligate with options. The
\ifsi@old@ohm      first set are to do with support for gensymb.
\ifsi@old@redef-gensymb 828 \si@opt@compatkey{unitsdef}{OHM}
\ifsi@gensymb      829 \si@opt@compatkey{unitsdef}{ohm}
                   830 \si@opt@compatkey{unitsdef}{redef-gensymb}
                   831 \newif\ifsi@gensymb

\ifsi@old@LITER    The second set are more general functionality.
\ifsi@old@liter    832 \si@opt@compatkey{unitsdef}{LITER}
\ifsi@old@noxspace 833 \si@opt@compatkey{unitsdef}{liter}
\ifsi@old@noconfig 834 \si@opt@compatkey{unitsdef}{noxspace}
                   835 \si@opt@compatkey{unitsdef}{noconfig}

\ifsi@old@noabbr   The final set are for control of abbreviations, and are a good demonstration of
\ifsi@old@noamperageabbr why to use xkeyval!
\ifsi@old@nofrequncyabbr 836 \si@opt@compatkey{unitsdef}{noabbr}
\ifsi@old@nomolabbr 837 \si@opt@compatkey{unitsdef}{noampereageabbr}
\ifsi@old@novoltageabbr 838 \si@opt@compatkey{unitsdef}{nofrequncyabbr}
\ifsi@old@novolumeabbr 839 \si@opt@compatkey{unitsdef}{nomolabbr}
\ifsi@old@noweightabbr 840 \si@opt@compatkey{unitsdef}{novoltageabbr}
\ifsi@old@noenergyabbr 841 \si@opt@compatkey{unitsdef}{novolumeabbr}
\ifsi@old@nolengthabbr 842 \si@opt@compatkey{unitsdef}{noweightabbr}
\ifsi@old@notimeabbr 843 \si@opt@compatkey{unitsdef}{noenergyabbr}
\ifsi@old@notimeabbr 844 \si@opt@compatkey{unitsdef}{nolengthabbr}
\ifsi@old@notimeabbr 845 \si@opt@compatkey{unitsdef}{notimeabbr}

\ifsi@old@cdot     The Slunits package has lots of options. These ones are all related to spacing.
\ifsi@old@thickspace 846 \si@opt@compatkey{SIunits}{cdot}
\ifsi@old@mediumspace 847 \si@opt@compatkey{SIunits}{thickspace}
\ifsi@old@thinspace 848 \si@opt@compatkey{SIunits}{mediumspace}
\ifsi@old@thickqspace 849 \si@opt@compatkey{SIunits}{thinspace}
\ifsi@old@mediumqspace 850 \si@opt@compatkey{SIunits}{thickqspace}
\ifsi@old@thingqspace 851 \si@opt@compatkey{SIunits}{mediumqspace}
\ifsi@old@thingqspace 852 \si@opt@compatkey{SIunits}{thingqspace}

\ifsi@old@amssymb   These options are used by Slunits to control clashes with other packages.
\ifsi@old@squaren 853 \si@opt@compatkey{SIunits}{amssymb}
\ifsi@old@pstricks 854 \si@opt@compatkey{SIunits}{squaren}
\ifsi@old@Gray 855 \si@opt@compatkey{SIunits}{pstricks}
\ifsi@old@italian 856 \si@opt@compatkey{SIunits}{Gray}
\ifsi@old@italian 857 \si@opt@compatkey{SIunits}{italian}

\ifsi@old@textstyle The miscellaneous options.
\ifsi@old@binary 858 \si@opt@compatkey{SIunits}{textstyle}
\ifsi@old@noams 859 \si@opt@compatkey{SIunits}{binary}
\ifsi@old@derivedinbase 860 \si@opt@compatkey{SIunits}{noams}
\ifsi@old@derived 861 \si@opt@compatkey{SIunits}{derivedinbase}
\ifsi@old@derived 862 \si@opt@compatkey{SIunits}{derived}

\ifsi@old@noprefixcmds The hepunits package only has one option.
\ifsi@old@noprefixcmds 863 \si@opt@compatkey{hepunits}{noprefixcmds}

```

```

\ifsi@old@english  fancynum provides a few options. First the rather oddly named english and
\ifsi@old@french    french ones.
                    864 \si@opt@compatkey{fancyntum}{english}
                    865 \si@opt@compatkey{fancyntum}{french}

\ifsi@old@tight     A couple for spacing multiplication.
\ifsi@old@loose     866 \si@opt@compatkey{fancyntum}{tight}
                    867 \si@opt@compatkey{fancyntum}{loose}

\ifsi@old@thinspaces Three for digit separation.
\ifsi@old@commas    868 \si@opt@compatkey{fancyntum}{thinspaces}
\ifsi@old@plain     869 \si@opt@compatkey{fancyntum}{commas}
                    870 \si@opt@compatkey{fancyntum}{plain}

\ifsi@old@spaceqspace The fancyunits package provides one option not available with Slunits.
                    871 \si@opt@compatkey{fancyunits}{spaceqspace}

```

21.6 Constants

A number of macros are needed by the package that provide a non-changing output. These are defined here; the intention is that these should not be macros that the user is likely to need to alter. All of these macros have preface `\si@fix@`, to flag that that are intended as constants. The package may rely on the contents of these macros for functionality.

```

\si@fix@thin       First, there are the various space macros. To allow both med and medium to be
\si@fix@med         used as a space description, two macros are needed for the same output.
\si@fix@medium     872 \newcommand*{\si@fix@thin}{\,}
\si@fix@thick      873 \newcommand*{\si@fix@med}{\:}
\si@fix@space      874 \newcommand*{\si@fix@medium}{\;}
                    875 \newcommand*{\si@fix@thick}{\;}
                    876 \newcommand*{\si@fix@space}{\text{ }}

\si@fix@cdot       Next there are macros for material that is not simply whitespace. To allow several
\si@fix@comma      options, the full-stop gets lots of names.
\si@fix@stop       877 \newcommand*{\si@fix@cdot}{{}\cdot{}}
\si@fix@fullstop   878 \newcommand*{\si@fix@comma}{{,}}
\si@fix@period     879 \newcommand*{\si@fix@stop}{{.}}
\si@fix@times      880 \newcommand*{\si@fix@fullstop}{{.}}
\si@fix@tighttimes 881 \newcommand*{\si@fix@period}{{.}}
\si@fix@tightcdot  882 \newcommand*{\si@fix@times}{\times}
                    883 \newcommand*{\si@fix@tighttimes}{\bgroup\times\egroup}
                    884 \newcommand*{\si@fix@tightcdot}{\bgroup\cdot\egroup}

\si@fix@plus       Signs for numbers are needed.
\si@fix@minus      885 \newcommand*{\si@fix@plus}{+}
\si@fix@pm         886 \newcommand*{\si@fix@minus}{-}
\si@fix@tightpm    887 \newcommand*{\si@fix@pm}{\pm}
\si@fix@mp         888 \newcommand*{\si@fix@tightpm}{\bgroup\pm\egroup}
                    889 \newcommand*{\si@fix@mp}{\mp}

```

`\si@fix@two` The literals “2” and “10” are needed for exponents.

```

\si@fix@ten 890 \newcommand*{\si@fix@two}{2}
            891 \newcommand*{\si@fix@ten}{10}

\si@fix@slash Another optional component that will probably not be used by many people.
            892 \newcommand*{\si@fix@slash}{/}

\si@fix@none Finally for spacing, there is the possibility of nothing at all
            893 \newcommand*{\si@fix@none}{}

```

21.7 Symbols

`\si@symbol` Each of the symbol macros needs to be set up; the options give a maths and text mode sign, but internally a single macro is needed for each.

```

894 \newcommand*{\si@symbol}[1]{%
895   \expandafter\protected\expandafter\def
896     \csname si@sym@#1\endcsname{%
897     \ifmode
898       \expandafter\csname si@maths#1\expandafter\endcsname
899     \else
900       \expandafter\csname si@text#1\expandafter\endcsname
901     \fi}}

\si@sym@Omega The various symbols are now declared.
\si@sym@ringA 902 \si@symbol{Omega}
\si@sym@mu    903 \si@symbol{ringA}
\si@sym@degree 904 \si@symbol{mu}
\si@sym@minute 905 \si@symbol{degree}
\si@sym@second 906 \si@symbol{minute}
\si@sym@celsius 907 \si@symbol{second}
\si@sym@celsius 908 \si@symbol{celsius}

```

The issue of redefinition of symbols now arises. `siunitx` can check for the loading of a number of support package, and can then redefine the appropriate symbols.

```

909 \AtBeginDocument{%
910   \ifsi@redefsymbols
911     \@ifpackageloaded{textcomp}{
912       {\si@log@debug{Redefining symbols using textcomp}}%
913       \renewcommand*{\si@textdegree}{\textdegree}%
914       \renewcommand*{\si@mathsdegree}{\text{\textdegree}}%

```

`mathptmx` will give issues with `textcomp` and the Ω sign.

```

915     \@ifpackageloaded{mathptmx}{}
916     {\renewcommand*{\si@textmu}{\textmu}%
917     \renewcommand*{\si@textOmega}{\textohm}}%

```

The Å symbol is only redefined if the encoding is OT1; other encodings should have a proper glyph used for `\AA`. The `\encodingdefault` macro is `\long` for some reason.

```

918     \renewcommand*{\si@tempa}{OT1}%
919     \ifx\si@tempa\encodingdefault
920       \renewcommand*{\si@mathsringA}{%
921         \text{\capitalring{A}}}%

```

```

922         \renewcommand*{\si@textringA}{\capitalring{A}}
923         \fi}{}
924     \@ifpackageloaded{upgreek}
925     {\si@log@debug{Redefining symbols using upgreek}%
926     \renewcommand*{\si@mathsOmega}{\Upomega}}{}
927 \fi}

```

21.8 Handling fractions

`\si@frac` Various methods of handling fractions are provided.

```

\si@frc@hook 928 \newcommand*{\si@frc@frac}[2]{%
\si@frc@frac 929 \ensuremath{\si@frc@hook\frac{%
\si@frc@slash 930 \expandafter\si@unt@out\expandafter{#1}}%
\si@frc@nice 931 {\expandafter\si@unt@out\expandafter{#2}}}%
\si@frc@sfrac 932 \let\si@frac\si@frc@frac
933 \newcommand*{\si@frc@hook}{}
934 \newcommand*{\si@frc@slash}[2]{%
935 \expandafter\si@unt@out\expandafter{#1}%
936 \si@out{\ensuremath{\si@slash}}}%
937 \expandafter\si@unt@out\expandafter{#2}}
938 \newcommand*{\si@frc@nice}[2]{%
939 \ensuremath{\si@frc@nicefrac{\expandafter\si@unt@out%
940 \expandafter{#1}}{\expandafter\si@unt@out\expandafter
941 {#2}}}%
942 \newcommand*{\si@frc@sfrac}[2]{%
943 \sfrac{\expandafter\si@unt@out\expandafter{#1}}%
944 {\expandafter\si@unt@out\expandafter{#2}}}%
945 \AtBeginDocument{
946 \@ifpackageloaded{xfrac}
947 {}
948 {\si@log@inf{xfrac package unavailable\MessageBreak
949 using 'fraction=sfrac' will fall back on\MessageBreak
950 nicefrac-like method}%
951 \renewcommand*{\si@frc@sfrac}[2]{%
952 \si@log@warn{xfrac package unavailable}%
953 \si@frc@nice{#1}{#2}}}%

```

`\si@frc@nicefrac` To avoid needing units installed, the `\nicefrac` macro needs to be emulated here. The code is taken (with permission) from `kgnicefrac`.⁵²

```

\si@frc@displen 954 \newlength\si@frc@displen
\si@frc@suplen 955 \newlength\si@frc@suplen
\si@frc@ssuplen 956 \newlength\si@frc@ssuplen
957 \newlength\si@frc@ssuplen
958 \newcommand*{\si@frc@nicefrac}{%
959 \ifmmode
960 \expandafter\si@frc@mathsnf
961 \else
962 \expandafter\si@frc@textnf
963 \fi}

```

`\si@frc@mathsnf` #1 : numerator

⁵²The original is licensed under the GPL; thanks to the author Axel Reichert for permission to copy the code here.

#2 : denominator
The maths mode system.

```

964 \newcommand*{\si@frc@mathsnf}[2]{%
965   \begingroup
966     \settoheight{\si@frc@displen}{\ensuremath{%
967       \displaystyle{M}}}%
968     \settoheight{\si@frc@textlen}{\ensuremath{%
969       \textstyle{M}}}%
970     \settoheight{\si@frc@suplen}{\ensuremath{%
971       \scriptstyle{M}}}%
972     \settoheight{\si@frc@ssuplen}{%
973       \ensuremath{\scriptscriptstyle{M}}}%
974     \addtolength{\si@frc@displen}{-\si@frc@ssuplen}%
975     \addtolength{\si@frc@textlen}{-\si@frc@ssuplen}%
976     \addtolength{\si@frc@suplen}{-\si@frc@ssuplen}%
977     \mathchoice
978       {\raisebox{\si@frc@displen}{\ensuremath{%
979         \scriptstyle{#1}}}}%
980       {\raisebox{\si@frc@textlen}{\ensuremath{%
981         \textstyle{#1}}}}%
982       {\raisebox{\si@frc@suplen}{%
983         \ensuremath{\scriptscriptstyle{#1}}}}%
984       {\raisebox{\si@frc@ssuplen}{%
985         \ensuremath{\scriptscriptstyle{#1}}}}%
986     \mkern-2mu\relax/\mkern-1mu\relax
987   \bgroup
988     \mathchoice
989       {\scriptstyle}%
990       {\scriptstyle}%
991       {\scriptscriptstyle}%
992       {\scriptscriptstyle}%
993     {#2}%
994   \egroup
995 \endgroup}

```

\si@frc@textnf #1 : numerator
#2 : denominator

A stripped down version of the nicefrac system for text mode.

```

996 \newcommand*{\si@frc@textnf}[2]{%
997   \begingroup
998     \settoheight{\si@frc@textlen}{M}%
999     \settoheight{\si@frc@ssuplen}{\fontsize\sf@size\z@\relax
1000       \selectfont{M}}%
1001     \addtolength{\si@frc@textlen}{-\si@frc@ssuplen}%
1002     \raisebox{\si@frc@textlen}{\fontsize\sf@size\z@\relax
1003       \selectfont{#1}}%
1004     \hspace{-0.25ex}/\hspace{-0.25ex}%
1005     \hbox{\fontsize\sf@size\z@\selectfont{#2}}%
1006   \endgroup}

```

\si@frc@ugly #1 : numerator

The \si@frc@ugly macro is needed to emulate the ugly option in units, where output depends on the current mode.

```

1007 \newcommand*{\si@frc@ugly}[1]{%
1008   \renewcommand*{\si@tempa}{#1}%
1009   \ifmmode
1010     \expandafter\si@frc@frac
1011   \else
1012     \renewcommand*{\si@tempb}{1}%
1013     \ifx\si@tempa\si@tempb

```

The slash switch cannot be used, so the possibility of the numerator being one is handled here.

```

1014       \setbox\si@tempboxa=\hbox{\ensuremath{\si@valuesep}}}%
1015       \hskip-\wd\si@tempboxa\relax
1016       \renewcommand*{\si@tempa}{}%
1017   \fi
1018   \expandafter\si@frc@slash
1019 \fi
1020 {\si@tempa}}

```

21.9 Font control

A number of controls and tests are needed to control the font used for output. Underlying all of this is the $\mathcal{A}\mathcal{M}\mathcal{S}$ package `amstext` package, providing the `\text` command. Much of the font control system here is taken more or less verbatim from `SIstyle`; modifications have been made to fit the `siunitx` interface.

`\si@fam@sf` The package needs to know the maths font families in use. This is set right at the start of the document, after any changes can have been made by, for example, `fontspec`.

```

1021 \g@addto@macro{\document}{%
1022   \ifdefined\mathsf
1023     \setbox\si@tempboxa=\hbox{%
1024       $\mathsf{\global\chardef\si@fam@sf=\fam}$}%
1025   \else
1026     \si@log@inf{\string\mathsf not found}%
1027     \global\chardef\si@fam@sf=99\relax
1028   \fi
1029   \ifdefined\mathtt
1030     \setbox\si@tempboxa=\hbox{%
1031       $\mathtt{\global\chardef\si@fam@tt=\fam}$}%
1032   \else
1033     \si@log@inf{\string\mathtt not found}%
1034     \global\chardef\si@fam@tt=99\relax
1035   \fi}

```

`\si@fam@ifbtext` #1 : code

`\si@fam@ifbmaths` These tests check for bold in text and maths mode, respectively.

```

1036 \newcommand*{\si@fam@ifbtext}[1]{%
1037   \if b\expandafter\@car\@fseries\@nil
1038     #1\fi}
1039 \newcommand*{\si@fam@ifbmaths}[1]{%
1040   \renewcommand*{\si@tempa}{bold}%
1041   \ifx\math@version\si@tempa
1042     #1\fi}

```


`\si@fam@ifbinline` For compatibility with units, a method to change the behaviour when in inline maths is needed for the bold detector.

```
1043 \newcommand*{\si@fam@ifbinline}{%
1044   \ifsi@inlinebtext
1045     \expandafter\si@fam@ifbtext
1046   \else
1047     \expandafter\si@fam@ifbmaths
1048   \fi}
```

`\si@fam@ifitext` #1 : code

This test check for italic or slanted text in text mode, by negation (upright text is n).

```
1049 \newcommand*{\si@fam@ifitext}[1]{%
1050   \if n\expandafter\@car\f@series\@nil\else
1051     #1\fi}
```

`\si@fam@mode` Detection of the current mode needs to happen“early” (before any change of `\ensuremath`). So a short macro is provided to do the job.

```
1052 \newcommand*{\si@fam@mode}{%
1053   \ifsi@obeymode
1054     \ifmmode
1055       \sisetup{mode=maths}%
1056     \else
1057       \sisetup{mode=text}%
1058     \fi
1059   \fi}
```

`\si@fam@colourcmd` The colour command is set up.

```
1060 \AtBeginDocument{
1061   \@ifpackageloaded{color}
1062     {\let\si@fam@colourcmd\color}
1063     {\let\si@fam@colourcmd\@gobble}}
```

`\ifsi@fam@set` A marker is set up to check if font-matching has been taken place. A second flag
`\ifsi@textmode` is used to track the use of text mode.

```
1064 \newif\ifsi@fam@set
1065 \newif\ifsi@textmode
```

`\si@fam@set` Using the code from `Slstyle` as a base, a set of tests are used to set the current font families and weights. To begin with, the mode to use is set up.

```
1066 \newcommand*{\si@fam@set}{%
1067   \ifsi@out@num
1068     \ifsi@numtextmode
1069       \expandafter\expandafter\expandafter\si@textmodetrue
1070     \else
1071       \expandafter\expandafter\expandafter\si@textmodefalse
1072     \fi
1073   \else
1074     \ifsi@unittextmode
1075       \expandafter\expandafter\expandafter\si@textmodetrue
1076     \else
1077       \expandafter\expandafter\expandafter\si@textmodefalse
1078     \fi
1079   \fi}
```

```

\si@mathsrms The appropriate font macros are now established, if necessary.
\si@mathssfi1080 \ifsi@fam@set\else
\si@mathstt1081 \let\si@colourcmd\@gobble
\si@textrm1082 \ifsi@out@num
\si@textsf1083 \let\si@mathsrms\si@valuemathsrms
\si@texttt1084 \let\si@mathssf\si@valuemathssf
\si@colourcmd1085 \let\si@mathstt\si@valuemathstt
\si@colour1086 \let\si@textrm\si@valuetextrm
1087 \let\si@textsf\si@valuetextsf
1088 \let\si@texttt\si@valuetexttt
1089 \ifsi@colourvalues
1090 \let\si@colourcmd\si@fam@colourcmd
1091 \fi
1092 \let\si@colour\si@valuecolour
1093 \else
1094 \let\si@mathsrms\si@unitmathsrms
1095 \let\si@mathssf\si@unitmathssf
1096 \let\si@mathstt\si@unitmathstt
1097 \let\si@textrm\si@unittextrm
1098 \let\si@textsf\si@unittextsf
1099 \let\si@texttt\si@unittexttt
1100 \ifsi@colourunits
1101 \let\si@colourcmd\si@fam@colourcmd
1102 \fi
1103 \let\si@colour\si@unitcolour
1104 \fi
1105 \fi
1106 \si@fam@settrue

The temporary macros are needed for the \ifx tests, which need to be expanded
once.
1107 \edef\si@tempa{\sfdefault}%
1108 \edef\si@tempb{\ttdefault}%

\si@fam@maths The surrounding font family is only tested if matching is requested. First, the
\si@fam@text defaults are set up assuming no detection takes place.
1109 \expandafter\let\expandafter\si@fam@maths
1110 \csname\si@mathsrms\endcsname
1111 \expandafter\let\expandafter\si@fam@text
1112 \csname\si@textrm\endcsname
1113 \ifsi@obeyfamily
1114 \si@log@debug{Font detection: checking font}%

The detection code has to check the mode currently in operation. Display
mathematics can be handled in two ways, so this means some code is repeated:
it is spun out to separate routines.
1115 \ifmmode
1116 \ifinner
1117 \si@log@debug{Font detection: inline maths}%
1118 \si@fam@detttext
1119 \else
1120 \si@log@debug{Font detection: display maths}%
1121 \ifsi@detectdisplay

```

```

1122         \si@fam@detmaths
1123     \else
1124         \si@fam@detttext
1125     \fi
1126 \fi
1127 \else
1128     \si@log@debug{Font detection: text}%
1129     \si@fam@detttext
1130 \fi
1131 \else
1132     \si@log@debug{Font detection: inactive}%
1133 \fi

```

`\si@fam@bold` With the font family set, the next check is for bold text. This again needs to examine the current mode. Things are a bit more complex than in `Slstyle` as it is possible to be typesetting in either text or maths mode. The bold command is set up with `\def`, as nested calls can occur.

```

1134 \def\si@fam@bold{\unboldmath\mdseries}%
1135 \ifsi@obeybold
1136     \si@log@debug{Weight detection: checking weight}%
1137     \ifmmode
1138         \ifdim\displaywidth>0pt\relax
1139             \ifsi@detectdisplay
1140                 \expandafter\si@fam@ifbmaths
1141             \else
1142                 \expandafter\si@fam@ifbtext
1143             \fi
1144             \si@fam@setbold
1145         \else
1146             \si@fam@ifbinline\si@fam@setbold
1147         \fi
1148     \else
1149         \si@fam@ifbtext\si@fam@setbold
1150     \fi
1151 \fi

```

`\si@fam@italic` The value of `obeyitalic` is now tested; as this does nothing in maths mode, a reminder is added to the log.

```

1152 \let\si@fam@italic\upshape
1153 \ifsi@obeyitalic
1154     \si@log@debug{Italic detection: checking italic}%
1155     \si@fam@ifitext
1156     {\let\si@fam@italic\relax
1157     \si@log@debug{Italic detection: italic}}%
1158 \fi}

```

`\si@fam@detmaths` Two detection macros are needed for maths and text mode. This allows handling
`\si@fam@detttext` of the various combinations without needing too many code lines.

```

1159 \newcommand*{\si@fam@detmaths}{%
1160     \ifnum\the\fam=\si@fam@sf
1161         \si@log@debug{Font detection: sf}%
1162         \expandafter\let\expandafter\si@fam@maths
1163         \csname\si@mathssf\endcsname

```

```

1164 \expandafter\let\expandafter\si@fam@text
1165 \csname\si@textsf\endcsname
1166 \else
1167 \ifnum\the\fam=\si@fam@tt
1168 \si@log@debug{Font detection: tt}%
1169 \expandafter\let\expandafter\si@fam@maths
1170 \csname\si@mathstt\endcsname
1171 \expandafter\let\expandafter\si@fam@text
1172 \csname\si@texttt\endcsname
1173 \else
1174 \si@log@debug{Font detection: rm}%
1175 \expandafter\let\expandafter\si@fam@maths
1176 \csname\si@mathsrms\endcsname
1177 \expandafter\let\expandafter\si@fam@text
1178 \csname\si@textrm\endcsname
1179 \fi
1180 \fi}
1181 \newcommand*{\si@fam@detttext}{%
1182 \ifx\f@family\si@tempa
1183 \si@log@debug{Font detection: sf}%
1184 \expandafter\let\expandafter\si@fam@maths
1185 \csname\si@mathssf\endcsname
1186 \expandafter\let\expandafter\si@fam@text
1187 \csname\si@textsf\endcsname
1188 \else
1189 \ifx\f@family\si@tempb
1190 \si@log@debug{Font detection: tt}%
1191 \expandafter\let\expandafter\si@fam@maths
1192 \csname\si@mathstt\endcsname
1193 \expandafter\let\expandafter\si@fam@text
1194 \csname\si@texttt\endcsname
1195 \else
1196 \si@log@debug{Font detection: rm}%
1197 \expandafter\let\expandafter\si@fam@maths
1198 \csname\si@mathsrms\endcsname
1199 \expandafter\let\expandafter\si@fam@text
1200 \csname\si@textrm\endcsname
1201 \fi
1202 \fi}

```

`\si@fam@setbold` For setting bold, a couple of control macros are needed.

```

\si@fam@boldify1203 \newcommand*{\si@fam@setbold}{%
1204 \si@log@debug{Weight detection: bold weight}%
1205 \let\si@fam@bold\si@fam@boldify}
1206 \newcommand*{\si@fam@boldify}{\boldmath\bfseries}

```

21.10 Formatting numbers

`\num` [#1]: keyval options
#2 : number

The system used here is modelled on that in `numprint`; the input is broken down into single tokens, each one is examined and the result is re-assembled into an output number. However, various changes have been made to the system used,

and so the macros here are not simply renamed copies of those in numprint. The user macro `\num` sets any local keys, then calls the number formatting macro on the processed number.

```

1207 \si@newrobustcmd*{\num}[2][]{%
1208   \begingroup
1209     \sisetup{#1}%
1210     \si@fam@mode
1211     \si@num@intabfalse
1212     \si@log@debug{Processing \string\num\space input `#2'}%
1213     \expandafter\si@out@num\expandafter{\si@num{#2}}%
1214   \endgroup}

```

`\ifsi@num@intab` A flag for processing inside a table is needed.

```

1215 \newif\ifsi@num@intab

```

`\si@num` #1 : number

This is the main processing macro. Unlike the related macro in numprint, the output of this macro is not subjected to any font changes. That is left to one of the `\si@out@...` macros. No grouping is applied here; any call to `\si@num` (or any of the sub-macros) must be within a group as the definitions used rely on this. Grouping is not applied here so that other macros can get the various separated parts of the input.

```

1216 \newcommand*{\si@num}[1]{%

```

The argument of the macro is fully expanded before any processing. By using `\scantokens`, any odd problems from packages with active characters can be avoided.

```

1217   \si@num@fixpm
1218   \begingroup
1219     \makeatletter
1220     \@makeother{\,%
1221     \@makeother{.}%
1222     \@makeother{+}%
1223     \@makeother{-}%
1224     \def~{}%
1225     \def\,%{}%
1226     \catcode`\~= \active\relax
1227     \catcode`\^= \active\relax
1228     \everyeof{\noexpand}%
1229     \endlinechar\m@ne
1230     \protected@xdef\si@tempa{\scantokens{#1}}%
1231   \endgroup

```

Processing only takes place if there is actually something in the argument. This is tested once “hard” spaces have been stripped out; if there is input other than spaces, the processor first checks the validity of the input, then moves on to format it.

```

1232   \si@ifnotmtarg{\si@tempa}
1233   {\si@num@ifvalid{\si@tempa}
1234     {\si@num@format{\si@tempa}}

```

The parser has to bailed-out, and so no further processing of the input is done. Instead, whatever was passed to the macro is returned as supplied.

```

1235      {\si@log@err{Invalid character '#1' in numerical input}%
1236      {Only characters from the list
1237      '\si@numvalid'\MessageBreak should be present in the
1238      argument of the \string\num\space macro\MessageBreak
1239      (or derivative such as an 's' column)}}%
1240      {#1}}}}

```

`\si@num@fixpm` With certain packages loaded, there can be issues with `\pm` and `\mp`. To avoid this, the ϵ -TeX `\protected` system is employed; this is only used within local groups.

```

\pm1241 \newcommand*{\si@num@fixpm}{%
\mp1242 \let\si@num@pm\pm
1243 \let\si@num@mp\mp
1244 \protected\def\pm{\si@num@pm}%
1245 \protected\def\mp{\si@num@mp}}

```

`\si@num@ifvalid` #1 : chars

`\si@num@valid` Assuming that there is a non-space argument to `\si@num`, every character is checked to ensure it is valid in the context, so that further processing can occur without sanity checks. If the character is valid, recursion occurs.

```

1246 \newcommand*{\si@num@ifvalid}[1]{%
1247 \begingroup
1248 \si@switchtrue
1249 \expandafter\si@num@valid#1\@empty\@empty
1250 \ifsi@switch
1251 \aftergroup\@firstoftwo
1252 \else
1253 \aftergroup\@secondoftwo
1254 \fi
1255 \endgroup}
1256 \def\si@num@valid#1#2\@empty{%
1257 \si@str@ifchrstr{#1}{\si@numvalid}
1258 {\ifx\@empty#2\@empty\else
1259 \si@num@valid#2\@empty\@empty\@empty
1260 \fi}
1261 {\si@switchfalse}}

```

`\si@num@in` Various storage macros are needed.

```

\si@num@out1262 \newcommand*{\si@num@in}{}
\si@num@exp1263 \newcommand*{\si@num@out}{}
\si@num@expsign1264 \newcommand*{\si@num@exp}{}
\si@num@mant1265 \newcommand*{\si@num@expsign}{}
\si@num@mantsign1266 \newcommand*{\si@num@mant}{}
\si@num@err1267 \newcommand*{\si@num@mantsign}{}
\si@num@xpart1268 \newcommand*{\si@num@err}{}
\si@num@ambig1269 \newcommand*{\si@num@xpart}{}
\si@tab@out1270 \newcommand*{\si@num@ambig}{}
\si@tab@expout1271 \newcommand*{\si@tab@out}{}
1272 \newcommand*{\si@tab@expout}{}

```

`\ifsi@num@erropen` A flag is set up for tracking unclosed errors.

```

1273 \newif\ifsi@num@erropen

```

\si@num@format #1 : number

\si@num@arg The number processor starts by saving #1 (odd things happen otherwise). A hook is also provided to allow modifications by other macros.

```
1274 \newcommand*{\si@num@arg}{}
1275 \newcommand*{\si@num@format}[1]{%
1276   \protected@edef\si@num@arg{#1}%
1277   \si@log@debug{Formatting number '\si@num@arg'}%
```

The storage areas are emptied.

```
1278 \renewcommand*{\si@num@in}{}%
1279 \renewcommand*{\si@num@exp}{}%
1280 \renewcommand*{\si@num@expsign}{}%
1281 \renewcommand*{\si@num@mant}{}%
1282 \renewcommand*{\si@num@mantsign}{}%
1283 \renewcommand*{\si@num@err}{}%
1284 \renewcommand*{\si@num@xpart}{}%
```

Any “x-part” is now found, leaving the first number in \si@num@in and anything else in \si@num@xpart.

```
1285 \si@switchfalse
1286 \expandafter\si@num@findxpart\si@num@arg\@empty\@empty
```

The input is split into an mantissa and an exponent; the flag is used here to indicate if an exponent is found. The mantissa will end up in \si@num@mant, and the exponent in \si@num@exp.

```
1287 \si@switchfalse
1288 \si@num@sepmantexp{\si@num@in}%
```

The sign and value of the mantissa and exponent are separated; the mantissa is done after the exponent as this makes life easier when using the table-formatting fork. Checks are then needed, as a sign with no value is potentially-valid for the mantissa (for example -10^{10}).

```
1289 \si@num@sepsign{exp}%
1290 \si@num@sepsign{mant}%
1291 \ifx\@empty\si@num@exp\@empty
1292   \ifx\@empty\si@num@expsign\@empty\else
1293     \si@log@warn{Sign but no number for '\si@num@arg'}%
1294   \fi
1295   \let\si@num@expsign\@empty
1296 \fi
1297 \ifx\@empty\si@num@mant\@empty
1298   \ifx\@empty\si@num@mantsign\@empty\else
1299     \ifx\@empty\si@num@exp\@empty
1300       \si@log@warn{Sign but no number for '\si@num@arg'}%
1301       \let\si@num@mantsign\@empty
1302     \fi
1303   \fi
1304 \fi
```

A check for negative mantissa values is made, to allow some colour-based trickery.

```
1305 \renewcommand*{\si@tempa}{\{-}%
1306 \ifx\si@num@mantsign\si@tempa
1307   \ifsi@colourneg
1308     \expandafter\expandafter\expandafter\si@fam@colourcmd
1309   \else
```

```

1310     \expandafter\expandafter\expandafter\@gobble
1311     \fi
1312   \else
1313     \expandafter\@gobble
1314   \fi
1315   {\si@negcolour}%

```

The next stage is to process the remaining data, to find the decimal marker and reformat correctly. These two macros control this entire process. The exponent is processed first as this makes life easier when the system is used to typeset tabular material.⁵³

```

1316   \si@num@procnum{exp}%
1317   \si@num@procnum{mant}%

```

If the exponent is zero, then it might need to be deleted.

```

1318   \si@str@ifonlychrs{\si@num@exp}{0\si@numdecimal}
1319   {\ifsi@allowzeroexp\else
1320     \renewcommand*{\si@num@exp}{}%
1321     \ifx\@empty\si@num@mant\@empty
1322       \renewcommand*{\si@num@mant}{1}%
1323     \fi
1324   \fi}%

```

To build up the number for typesetting, a rather complex series of tests is needed. First, the “ambiguous error” flag is set if there is an exponent and the package has been asked to check this. The same flag will already be true if a unit is present and checking is active.

```

1325   \ifx\@empty\si@num@exp\@empty\else
1326     \ifsi@trapambigerr
1327       \expandafter\expandafter\expandafter\si@num@ambigertrue
1328     \fi
1329   \fi

```

\si@num@out Processing now divides, as when used with the S column some extra steps are needed. Inside a table, the macro \si@tab@out holds the part of the mantissa before the decimal sign. The post-decimal part then ends up in \si@num@out. \si@tab@out \si@tab@expout On the other hand, under normal circumstances the entire mantissa should be copied to \si@num@out.

```

1330   \protected@edef\si@num@out{%
1331     \ensuremath{{\si@num@mantsign}}\si@num@mant}%
1332   \renewcommand*{\si@tempa}{num}%
1333   \ifsi@num@intab
1334     \protected@edef\si@tab@out{%
1335       \ensuremath{{\si@num@mantsign}}\si@num@predec}%
1336     \protected@edef\si@num@out{\si@num@postdec}%
1337     \renewcommand*{\si@tempa}{tab}%
1338   \fi

```

Now there is the error part to handle. For tables, a check has to be made so the error ends up in the correct part of the number. The value of \si@tempa is used to track this, and so is set up here.

```

1339   \ifx\@empty\si@num@postdec\@empty\else
1340     \renewcommand*{\si@tempa}{num}%

```

⁵³The contents of \si@num@predec and \si@num@postdec are needed for the mantissa.


```

1341 \fi
1342 \ifx\@empty\si@num@err\@empty\else

```

If there is an error, and it is begin separated, the problem arises of the potential for ambiguous values. This can only apply outside of a table, as seperr is disabled in tabular material.

```

1343 \ifsi@seperr
1344 \ifsi@num@ambigerr
1345 \protected@edef\si@num@out{%
1346 \ensuremath{\si@openerr}\si@num@out}%
1347 \si@repeatunitsfalse
1348 \expandafter\si@num@erropentrue
1349 \else

```

If there is an exponential part and an error, errors are being separated and ambiguous errors are not trapped, then there is work to do. The exponent part is added to the *error*, and deleted from the mantissa if necessary.

```

1350 \ifsi@trapambigerr\else
1351 \ifx\@empty\si@num@exp\@empty\else
1352 \protected@edef\si@num@err{%
1353 \si@num@err\expandafter\car\si@num@exp\@nil
1354 \si@num@expsign\si@num@exp}%
1355 \ifsi@repeatunits\else
1356 \renewcommand*\si@num@exp{ }%
1357 \renewcommand*\si@num@expsign{ }%
1358 \fi
1359 \fi
1360 \fi
1361 \fi

```

If seperr is not in force, the error and mantissa have to be recombined. This is handled so that the same macro deals with tables and normal processing.

```

1362 \else
1363 \expandafter\protected@edef\csname
1364 si@\si@tempa @out\endcsname{%
1365 \si@num@out\ensuremath{\si@errspace}\ensuremath
1366 {\si@openerr}\si@num@err\ensuremath{\si@closeerr}}%
1367 \renewcommand*\si@num@err{ }%
1368 \fi
1369 \fi

```

The main reconstruction can now occur. This performs various checks on the validity of the input, and adds the necessary “filler” between the supplied data.

```

1370 \renewcommand*\si@tempa{num@out}%
1371 \ifsi@num@erropen
1372 \renewcommand*\si@tempa{num@ambig}%
1373 \fi
1374 \ifsi@num@intab
1375 \renewcommand*\si@tempa{tab@expout}%
1376 \fi
1377 \ifx\@empty\si@num@exp\@empty
1378 \ifx\@empty\si@num@mant\@empty
1379 \si@log@err{Invalid number format '\si@num@arg'}
1380 {Something is wrong with the number format; does it
1381 contain \MessageBreak any numbers (from the list

```

```

1382         '\si@numdigits')?}%
1383         \renewcommand*\si@num@out{}}%
1384     \fi
1385 \else
1386     \ifx\@empty\si@num@mant\@empty\else
1387         \expandafter\protected@edef\csname
1388             si@\si@tempa\endcsname{%
1389             \csname si@\si@tempa\endcsname\ensuremath{}}%
1390             \si@expproduct{}}}%
1391     \fi
1392     \expandafter\protected@edef\csname
1393         si@\si@tempa\endcsname{%
1394         \csname si@\si@tempa\endcsname\si@expbase
1395         \textsuperscript{\ensuremath{\si@num@expsign}}%
1396         \si@num@exp}}}%
1397 \fi

```

With everything done, the result is output.

```

1398 \ifsi@num@intab\else
1399     \expandafter\si@num@out
1400 \fi

```

If there is anything inside the “x-part”, then there is now more work to do.

```

1401 \ifx\@empty\si@num@err\@empty\else
1402     \expandafter\si@num@procerr
1403 \fi
1404 \ifsi@num@erropen
1405     \expandafter\si@out@num\expandafter{%
1406         \ensuremath{\si@closeerr}}%
1407     \ifx\@empty\si@num@ambig\@empty\else
1408         \expandafter\si@out@num\expandafter{\si@num@ambig}%
1409         \renewcommand*\si@num@ambig{}}%
1410     \fi
1411 \fi
1412 \si@num@erropenfalse
1413 \ifx\@empty\si@num@xpart\@empty\else
1414     \expandafter\si@num@sepxpart
1415 \fi}

```

`\si@num@findxpart` Before processing the number, any multiplied parts have to be found and removed. As there can be more than one product sign, the building process here has no error checking.

```

1416 \def\si@num@findxpart#1#2\@empty{%
1417     \si@str@ifchrstr{#1}{\si@numprod}
1418     {\si@switchtrue\si@seperrfalse}}}%
1419 \ifsi@switch
1420     \protected@edef\si@num@xpart{\si@num@xpart#1}%
1421 \else
1422     \protected@edef\si@num@in{\si@num@in#1}%
1423 \fi
1424 \ifx\@empty#2\@empty\else
1425     \si@num@findxpart#2\@empty
1426 \fi}

```

`\si@num@sepmanexp` #1 : number
`\si@num@mantexp` Splitting the mantissa and exponent first checks for characters to gobble, which are simply thrown away. For any other input, there are two possibilities. If the character is an exponent marker, then the package switches from collecting the mantissa to collecting the exponent (after a sanity check). All other characters are added to either the mantissa or the exponent, as appropriate.

```

1427 \newcommand*{\si@num@sepmanexp}[1]{%
1428   \expandafter\si@num@mantexp#1\@empty\@empty}
1429 \def\si@num@mantexp#1#2\@empty{%
1430   \si@str@ifchrstr{#1}{\si@num@gobble}
1431   {\si@log@debug{Gobbling '#1' in \si@num@arg}}
1432   {\si@str@ifchrstr{#1}{\si@num@exp}
1433    {\ifsi@switch
1434     \si@log@err{Duplicate exponent marker found}
1435     {Only a single exponent character \MessageBreak
1436      (from the list '\si@num@exp')\MessageBreak may
1437      occur in a numerical argument}%
1438     \else
1439     \si@log@debug{Exponent marker '#1' found in
1440      '\si@num@arg'}%
1441     \fi
1442     \si@switchtrue}%
1443   {\ifsi@switch
1444     \expandafter\si@num@addexp
1445     \else
1446     \expandafter\si@num@addmnt
1447     \fi
1448     {#1}}}%

```

If the recursion has not bottomed out, another loop occurs.

```

1449 \ifx\@empty#2\@empty
1450   \expandafter\@gobble
1451 \else
1452   \expandafter\si@num@sepmanexp
1453 \fi
1454 {#2}}

```

`\si@num@addmnt` #1 : char

`\si@num@addexp` To allow `\expandafter` use in the above, the actual addition to the appropriate macro is handled here. Two helper macros are needed.

```

1455 \newcommand*{\si@num@addmnt}[1]{%
1456   \si@num@addmntexp{#1}{mant}{mantissa}}
1457 \newcommand*{\si@num@addexp}[1]{%
1458   \si@num@addmntexp{#1}{exp}{exponent}}

```

`\si@num@addmntexp` #1 : char

#2 : either mant or exp

#3 : info-text

Then the business end.

```

1459 \newcommand*{\si@num@addmntexp}[3]{%
1460   \si@log@debug{Adding '#1' to #3 for '\si@num@arg'}%
1461   \expandafter\protected@edef\csname si@num@#2\endcsname{%
1462     \csname si@num@#2\endcsname#1}}

```

`\si@num@sepsign` #1 : either mant or exp
The input is now tested for a sign. If one exists, it is transferred into the `\si@num@#1sign` storage macro, with the remained of the number in `\si@num@#1`. The only check made directly here is that there is something to process.

```

1463 \newcommand*{\si@num@sepsign}[1]{%
1464   \expandafter\ifx\expandafter\@empty
1465     \csname si@num@#1\endcsname\@empty
1466     \expandafter\@gobble
1467   \else
1468     \expandafter\si@num@gensign
1469   \fi
1470   {#1}}

```

`\si@num@gensign` #1 : either mant or exp
The sign generator starts by calling the procedure to check if the input contains a valid one- or two-digit sign. The results are returned as `\si@num@sign` and `\si@num@value`.

```

1471 \newcommand*{\si@num@gensign}[1]{%
1472   \expandafter\expandafter\expandafter\si@num@findsign
1473   \csname si@num@#1\endcsname\@empty\@empty

```

If no sign has been found, then there may be a need to add one anyway.

```

1474   \ifx\@empty\si@num@sign\@empty
1475     \ifx\@empty\si@num@value\@empty
1476       \expandafter\expandafter\expandafter\@gobble
1477     \else
1478       \expandafter\expandafter\expandafter\si@num@addsign
1479     \fi
1480   \else
1481     \expandafter\@gobble
1482   \fi
1483   {#1}%

```

The appropriate storage areas are now assigned.

```

1484   \expandafter\let\csname si@num@#1sign\endcsname\si@num@sign
1485   \expandafter\let\csname si@num@#1\endcsname\si@num@value}

```

`\si@num@findsign` The first one or two characters of the mantissa or exponent may contain a sign.
`\si@num@sign` To test for this, the first two characters of the number are split off, and examined.
`\si@num@value` Two characters are used so that `\pm` and `\mp` can be represented by `++` and `--`, respectively. To allow the user to alter the valid signs, but retain this conversion, the generic character test is used before checking specific matches.

```

1486 \newcommand*{\si@num@sign}{}
1487 \def\si@num@findsign#1#2#3\@empty{%
1488   \si@num@delplusfalse
1489   \si@str@ifchrstr{#1}{\si@num@sign}{%
1490     \si@str@ifchrstr{#2}{\si@num@sign}{%
1491       \if ++#1%
1492       \if --#2%
1493         \si@log@debug{Found sign combination +- for
1494           '\si@num@arg'}%
1495         \renewcommand*{\si@num@sign}{\si@pm}}%

```

```

1496         \else
1497         \si@log@inf{Unknown sign combination '#1#2'}%
1498         \renewcommand*\si@num@sign{{#1#2}}%
1499     \fi
1500 \else
1501     \if -#1%
1502     \if +#2%
1503     \si@log@debug{Found sign combination +- for
1504     '\si@num@arg'}%
1505     \renewcommand*\si@num@sign{{\mp}}%
1506     \else
1507     \si@log@inf{Unknown sign combination '#1#2'}%
1508     \renewcommand*\si@num@sign{{#1#2}}%
1509     \fi
1510 \else
1511     \si@log@inf{Unknown sign combination '#1#2'}%
1512     \renewcommand*\si@num@sign{{#1#2}}%
1513     \fi
1514 \fi
1515 \protected@edef\si@num@value{#3}}%

```

Only one valid sign character.

```

1516     {\si@log@debug{Found single sign character '#1' for
1517     '\si@num@arg'}%
1518     \renewcommand*\si@num@sign{{#1}}%
1519     \if +#1%
1520     \ifsi@retainplus\else
1521     \expandafter\expandafter\expandafter\si@num@killsign
1522     \fi
1523     \fi
1524     \protected@edef\si@num@value{#2#3}}}%

```

No valid sign, so \@empty is returned for the sign .

```

1525     {\si@log@debug{No sign found for '\si@num@arg'}%
1526     \renewcommand*\si@num@sign{}}%
1527     \protected@edef\si@num@value{#1#2#3}}}%

```

`\ifsi@num@delplus` A simple spin-out to remove a plus sign. A flag is set as it might be useful to know this.

```

1528 \newif\ifsi@num@delplus
1529 \newcommand*\si@num@killsign{%
1530     \si@num@delplustrue
1531     \renewcommand*\si@num@sign{}}%

```

`\si@num@addsign` #1 : either mant or exp

`\si@num@assign` The macro to add a sign to an unsigned number has to check whether this is a mantissa or an exponent. The result is still placed in `\si@num@sign` for ease of processing later.

```

1532 \newcommand*\si@num@addsign[1]{%
1533     \begingroup
1534     \renewcommand*\si@tempa{#1}%
1535     \renewcommand*\si@tempb{mant}%
1536     \ifx\si@tempa\si@tempb
1537     \aftergroup\@firstoftwo

```

```

1538     \else
1539         \aftergroup\@secondoftwo
1540     \fi
1541 \endgroup
1542 {\ifsi@num@signmant
1543     \expandafter\si@num@assign
1544     \else
1545         \expandafter\@gobble
1546     \fi
1547 {mantissa}}
1548 {\ifsi@num@signexp
1549     \expandafter\si@num@assign
1550     \else
1551         \expandafter\@gobble
1552     \fi
1553 {exponent}}
1554 \newcommand*{\si@num@assign}[1]{%
1555     \let\si@num@sign\si@sign
1556     \si@log@debug{Adding sign \si@sign\space to #1 for
1557         '\si@num@arg' }}

```

`\si@num@procnum` #1 : either mant or exp

The control macro for processing the number (plus any extra characters).

```

1558 \newcommand*{\si@num@procnum}[1]{%
1559     \expandafter\ifx\expandafter\@empty
1560         \csname si@num@#1\endcsname\@empty
1561         \expandafter\@gobble
1562     \else
1563         \expandafter\si@num@finddigits
1564     \fi
1565 {#1}}

```

`\si@num@predec` Two new storage areas are defined.

```

\si@num@postdec 1566 \newcommand*{\si@num@predec}{}
1567 \newcommand*{\si@num@postdec}{}

```

`\si@num@finddigits` #1 : either mant or exp

The core digit processor divides the number into the parts before and after the decimal point marker. The temporary switch is used to indicate finding a decimal marker.

```

1568 \newcommand*{\si@num@finddigits}[1]{%
1569     \renewcommand*{\si@num@predec}{}%
1570     \renewcommand*{\si@num@postdec}{}%
1571     \si@switchfalse
1572     \expandafter\expandafter\expandafter\si@num@digits
1573     \csname si@num@#1\endcsname\@empty\@empty

```

Tests are now made to see if padding zeros are needed. The trailing test needs to verify if a decimal marker was found, as well as if a zero is needed.

```

1574     \ifx\@empty\si@num@predec\@empty
1575         \ifsi@num@padlead
1576             \expandafter\expandafter\expandafter\si@num@addprezero
1577         \fi
1578     \fi

```

```

1579 \ifx\@empty\si@num@postdec\@empty
1580 \ifsi@num@padtrail
1581 \ifsi@switch
1582 \expandafter\expandafter\expandafter\expandafter
1583 \expandafter\expandafter\expandafter
1584 \si@num@addpostzero
1585 \fi
1586 \fi
1587 \fi

```

The next checks to make concern input validity in a more mathematical sense. First, if the number is zero, then no sign should be given under any circumstances. Then leading zeros need to be removed from the input. This is slightly complicated by the potential presence of “extra” characters.

```

1588 \si@num@unsign{#1}%
1589 \ifx\@empty\si@num@predec\@empty
1590 \else
1591 \expandafter\si@num@nozero
1592 \fi

```

A sanity check is made to ensure that the supplied number consisted of more than a decimal marker.

```

1593 \ifx\@empty\si@num@predec\@empty
1594 \ifx\@empty\si@num@postdec\@empty
1595 \expandafter\expandafter\expandafter\@gobble
1596 \else
1597 \expandafter\expandafter\expandafter\si@num@sepdigits
1598 \fi
1599 \else
1600 \expandafter\si@num@sepdigits
1601 \fi
1602 {#1}}

```

`\si@num@digits` The `\si@num@digits` macro compares each character in the input against the list of characters valid at this stage: numbers, decimal markers and “extra” characters.

```

1603 \def\si@num@digits#1#2\@empty{%
1604 \si@str@ifchrstr{#1}{\si@numdecimal}
1605 {\ifsi@switch
1606 \si@log@err{Duplicate decimal marker in '\si@num@arg'}
1607 {Only a single decimal marker (from the list
1608 '\si@numdecimal')\MessageBreak may occur in a
1609 numerical argument}%
1610 \else
1611 \si@log@debug{Found decimal marker '#1' in
1612 '\si@num@arg'}%
1613 \expandafter\si@switchtrue
1614 \fi}

```

The earlier code only checks for a sign at the start of the text. A check is therefore needed for a sign after the first two characters; if one is found, it is ignored.

```

1615 {\si@str@ifchrstr{#1}{\si@numsign}
1616 {\si@log@err{Misplaced sign character
1617 '#1' in '\si@num@arg'}
1618 {Sign characters '\si@numsign' can only

```

```
1619          occur\MessageBreak at the start of a number}}
```

The current character is added to the appropriate stack; this is “spun out” to avoid problems with expansion of the switch code.

```
1620      {\ifsi@switch
1621        \expandafter\si@num@post
1622        \else
1623        \expandafter\si@num@pre
1624        \fi
1625        {#1}}}%
1626      \ifx\@empty#2\@empty\else
1627        \si@num@digits#2\@empty\@empty
1628      \fi}
```

```
\si@num@pre  #1 : char
\si@num@post  Storage of the result is spun out.
```

```
1629 \newcommand*{\si@num@pre}[1]{%
1630   \si@num@prepost{#1}{pre}{integer}}
1631 \newcommand*{\si@num@post}[1]{%
1632   \si@num@prepost{#1}{post}{decimal}}
```

```
\si@num@prepost  #1 : char
                  #2 : either pre or post
                  #3 : info-text
                  Then actually stored here.
```

```
1633 \newcommand*{\si@num@prepost}[3]{%
1634   \expandafter\protected@edef\csname si@num@#2dec\endcsname{%
1635     \csname si@num@#2dec\endcsname#1}%
1636   \si@log@debug{Adding ‘#1’ to #3 part for ‘\si@num@arg’}}
```

```
\si@num@addprezero  A similar set of macros are used for the padding zeros.
```

```
\si@num@addpostzero1637 \newcommand*{\si@num@addprezero}{%
1638   \si@num@addpzero{pre}{leading}}
1639 \newcommand*{\si@num@addpostzero}{%
1640   \si@num@addpzero{post}{trailing}}
```

```
\si@num@addpzero  #1 : either pre or post
                  In this case, there is no argument to be passed along.
```

```
1641 \newcommand*{\si@num@addpzero}[2]{%
1642   \si@log@debug{Adding #2 zero for ‘\si@num@arg’}%
1643   \@namedef{si@num@#1dec}{0}}
```

```
\si@num@unsign  #1 : either mant or exp
```

```
\si@num@nosign  The trap for a sign with zero numerical input is made. First, a check is made to
                  see if there is a sign to worry about. The pre- and post-decimal parts are then
                  examined, to see if they contain something other than “o” or an extra character.
```

```
1644 \newcommand*{\si@num@unsign}[1]{%
1645   \expandafter\ifx\expandafter\@empty
1646     \csname si@num@#1sign\endcsname\@empty
1647     \expandafter\@gobble
1648   \else
1649     \expandafter\si@num@nosign
1650   \fi}
```



```

1651 {#1}}
1652 \newcommand*{\si@num@nosign}[1]{%
1653   \begingroup
1654     \si@switchtrue
1655     \si@str@ifonlychrs{\si@num@predec\si@num@postdec}{0}
1656     {\si@switchfalse}{}%
1657     \ifsi@switch
1658       \aftergroup\@gobble
1659     \else
1660       \aftergroup\@firstofone
1661     \fi
1662   \endgroup
1663   {\si@log@debug{Zero value: removing any sign}%
1664   \ifsi@ang@sign\else
1665     \namedef{si@num@#1sign}{}%
1666   \fi}}

```

`\si@num@nozero` A very short test for a totally zero pre-decimal component.

```

1667 \newcommand*{\si@num@nozero}{%
1668   \si@str@ifonlychrs{\si@num@predec}{0}
1669   {\renewcommand*{\si@num@predec}{0}}{}}

```

`\si@num@decimalhook` A hook is needed to attach things inside the group to happen afterwards, if the number is a decimal.

```

1670 \newcommand*{\si@num@decimalhook}{}

```

`\si@num@sepdigits` #1 : either mant or exp

The `\si@num@sepdigits` macro is only called if at least one of the mantissa and exponent contain something to output.

```

1671 \newcommand*{\si@num@sepdigits}[1]{%

```

First an overall check is needed for additional characters. By altering the contents of `\si@numextra`, the same code can be shared by two different checks.

```

1672   \begingroup
1673     \let\si@numextra\si@numaddn
1674     \protected@edef\si@tempa{\si@num@predec\si@num@postdec}%
1675     \si@num@ifextra{\si@tempa}
1676     {\aftergroup\@gobble}
1677     {\aftergroup\@firstofone}%
1678   \endgroup

```

Separation of the error in a number from the number itself is only attempted for the mantissa.

```

1679   {\renewcommand*{\si@tempb}{mant}}%
1680   \renewcommand*{\si@tempc}{#1}%
1681   \ifx\si@tempb\si@tempc
1682     \expandafter\si@num@checkerr
1683   \fi}%

```

If both parts of the number contain only digits, then any rounding can be attempted if there is no error part. Otherwise, the input must be left alone.

```

1684   \protected@edef\si@tempa{\si@num@predec\si@num@postdec}%
1685   \expandafter\si@str@ifonlychrs\expandafter{\si@tempa}
1686   {0123456789}

```

```

1687     {\ifx\@empty\si@num@err\@empty
1688         \ifsi@fixdp
1689             \expandafter\expandafter\expandafter\si@num@fixdp
1690             \fi
1691         \fi}{}%

```

If the pre-decimal part contains nothing except numbers, then digit separation is carried out.

```

1692     \si@num@ifextra{\si@num@predec}{}
1693     {\expandafter\si@num@int\expandafter{\si@num@predec}}%

```

A decision is made about the decimal sign, then digit separation occurs on the post-decimal part of the number.

```

1694     \renewcommand*{\si@tempc}{}%
1695     \ifx\@empty\si@num@postdec\@empty\else
1696         \si@num@decimalhook
1697         \renewcommand*{\si@tempc}{}%
1698         \ensuremath{{\si@decimalsymbol}}}%
1699         \si@num@ifextra{\si@num@postdec}{}
1700         {\expandafter\si@num@dec\expandafter{\si@num@postdec}}%
1701     \fi

```

The construction is finalised by re-combining the number.

```

1702     \expandafter\protected@edef\csname si@num@#1\endcsname
1703     {\si@num@predec\si@tempc\si@num@postdec}%

```

`\si@num@ifextra` #1 : number

`\si@num@extra` A relatively simple test for “extra” characters. Once again, a bit of group trickery is used.

```

1704 \newcommand*{\si@num@ifextra}[1]{%
1705     \begingroup
1706         \si@switchfalse
1707         \expandafter\si@num@extra#1\@empty\@empty
1708         \ifsi@switch
1709             \si@log@debug{Found ‘extra’ characters in ‘#1’}%
1710             \aftergroup\@firstoftwo
1711         \else
1712             \aftergroup\@secondoftwo
1713         \fi
1714     \endgroup}
1715 \def\si@num@extra#1#2\@empty{%
1716     \ifx\@empty#1\@empty\else
1717         \si@str@ifchrstr{#1}{\si@num@extra}{\si@switchtrue}{}%
1718         \ifx\@empty#2\@empty\else
1719             \si@num@extra#2\@empty\@empty
1720         \fi
1721     \fi}

```

`\ifsi@num@ambigerr` When separating out an error from a number, the first step is to see if there is a decimal part to the number. If so, any error must be in that part of the decimal part; it is not possible to have an error starting in the integer part and continuing into the decimal part.

```

1722 \newif\ifsi@num@ambigerr
1723 \newcommand*{\si@num@checkerr}{%

```

```

1724 \ifx\@empty\si@num@postdec\@empty
1725 \expandafter\si@num@preerr
1726 \else
1727 \expandafter\si@num@posterr
1728 \fi}

```

`\si@num@preerr` Errors in integers are easy to handle. After finding the error, the result is simply stored in the error macro `\si@num@err`.

```

1729 \newcommand*{\si@num@preerr}{%
1730 \si@num@seperr{pre}%
1731 \ifx\@empty\si@tempb\@empty\else
1732 \expandafter\renewcommand\expandafter*\expandafter
1733 \si@num@err\expandafter{\si@tempb}%
1734 \fi}

```

`\si@num@posterr` Life is more complex for an error in the decimal part. This is found, then it may need zero-filling or the insertion of a decimal point. This depends on whether the number of error digits is larger than the number of post-decimal digits.

```

1735 \newcommand*{\si@num@posterr}{%
1736 \si@num@seperr{post}%
1737 \ifx\@empty\si@tempb\@empty\else
1738 \ifsi@seperr
1739 \expandafter\expandafter\expandafter\si@num@psterr
1740 \else
1741 \let\si@num@err\si@tempb
1742 \fi
1743 \fi}
1744 \newcommand*{\si@num@psterr}{%
1745 \si@num@cntdigits{\si@tempb}%
1746 \si@tempcntb\si@tempcnta\relax
1747 \si@num@cntdigits{\si@num@postdec}%
1748 \ifnum\si@tempcnta<\si@tempcntb\relax
1749 \expandafter\si@num@largeerr
1750 \else
1751 \expandafter\si@num@smallerr
1752 \fi}

```

`\si@num@seperr` #1 : either pre or post

`\si@num@finderr` The usual hand-off is made for searching for an error.

```

1753 \newcommand*{\si@num@seperr}[1]{%
1754 \si@switchfalse
1755 \renewcommand*{\si@tempa}{}%
1756 \renewcommand*{\si@tempb}{}%
1757 \expandafter\expandafter\expandafter\si@num@finderr
1758 \csname si@num@#1dec\endcsname\@empty\@empty
1759 \ifx\@empty\si@tempb\@empty\else
1760 \expandafter\let\csname si@num@#1dec\endcsname\si@tempa
1761 \fi}
1762 \def\si@num@finderr#1#2\@empty{%

```

First a check for the opening character of an error.

```

1763 \si@str@ifchrstr{#1}{\si@num@openerr}

```

If the switch is set when an error is found, then something is wrong.

```

1764     {\ifsi@switch
1765       \si@log@err{Invalid error in number}
1766       {The numerical argument \si@num@arg\space has two (or
1767        more)\MessageBreak error-opening characters}%
1768     \else
1769       \expandafter\si@switchtrue
1770     \fi}

```

The end-of-error character has to be the last item of the input, and an error needs to start before it ends.

```

1771     {\si@str@ifchrstr{#1}{\si@num@closeerr}
1772     {\ifsi@switch
1773       \ifx\@empty#2\@empty\else
1774         \si@log@err{Invalid error in number}
1775         {The numerical argument \si@num@arg\space has an
1776          error-closing before the last character}%
1777       \fi
1778     \else
1779       \si@log@err{Invalid error in number}
1780       {The numerical argument \si@num@arg\space has an
1781        error-closing character\MessageBreak but no
1782        error-opening one}%
1783     \fi}

```

The input must be a digit. It is stored in the appropriate area.

```

1784     {\ifsi@switch
1785       \expandafter\si@num@addtmpb
1786     \else
1787       \expandafter\si@num@addtmpa
1788     \fi
1789     {#1}}}%
1790 \ifx\@empty#2\@empty\else
1791   \si@num@finderr#2\@empty
1792 \fi}

```

`\si@num@addtmpa` #1 : number

`\si@num@addtmpb` Some quick methods for adding to the temporary macros with `\if` expansion.

```

1793 \newcommand*{\si@num@addtmpa}[1]{\si@num@addtmp{a}{#1}}
1794 \newcommand*{\si@num@addtmpb}[1]{\si@num@addtmp{b}{#1}}

```

`\si@num@addtmp` #1 : either a or b

#2 : number

The addition is committed.

```

1795 \newcommand*{\si@num@addtmp}[2]{%
1796   \expandafter\protected@edef\csname si@temp#1\endcsname{%
1797     \csname si@temp#1\endcsname#2}}

```

`\si@num@cntdigits` #1 : number

`\si@num@cntdgt` A recursive system for counting the number of characters in a given input; only used here to count digits in a number.

```

1798 \newcommand*{\si@num@cntdigits}[1]{%
1799   \si@tempcnta\z\relax
1800   \expandafter\si@num@cntdgt#1\@empty\@empty}
1801 \def\si@num@cntdgt#1#2\@empty{%

```

```

1802 \ifx\@empty#1\@empty\else
1803 \advance\si@tempcnta\@ne\relax
1804 \fi
1805 \ifx\@empty#2\@empty\else
1806 \expandafter\si@num@cntdgt#2\@empty
1807 \fi}

```

`\si@num@smallerr` For handling an error with no more digits than the post-decimal part of a number, zero-padding is undertaken before adding a decimal sign and (possibly) leading zero.

```

1808 \newcommand*{\si@num@smallerr}{%
1809 \si@tempcntb\si@tempcnta\relax
1810 \si@num@serr
1811 \protected@edef\si@num@err{%
1812 \ifsi@num@padlead0\fi\expandafter\@car\si@numdecimal\@nil
1813 \si@tempb}}
1814 \newcommand*{\si@num@serr}{%
1815 \si@num@cntdigits{\si@tempb}%
1816 \ifnum\si@tempcnta=\si@tempcntb\relax\else
1817 \protected@edef\si@tempb{0\si@tempb}%
1818 \expandafter\si@num@serr
1819 \fi}

```

`\si@num@largeerr` When the error has more digits than the decimal part, some shuffling is needed.

```

\si@num@lerr1820 \newcommand*{\si@num@largeerr}{%
\si@num@movedigit1821 \renewcommand*{\si@tempa}{}%
1822 \si@tempcntb\si@tempcnta\relax
1823 \si@num@lerr
1824 \protected@edef\si@num@err{%
1825 \si@tempa\ensuremath{\si@decimalsymbol}\si@tempb}}
1826 \newcommand*{\si@num@lerr}{%
1827 \si@num@cntdigits{\si@tempb}%
1828 \ifnum\si@tempcnta=\si@tempcntb\relax\else
1829 \expandafter\si@num@movedigit\si@tempb\@empty\@empty
1830 \si@num@lerr
1831 \fi}
1832 \def\si@num@movedigit#1#2\@empty{%
1833 \protected@edef\si@tempa{\si@tempa#1}%
1834 \protected@edef\si@tempb{#2}}

```

`\si@num@fixdp` The test for fixing decimal places starts by counting up the number of decimal digits. The number is then padded, rounded or left alone.

```

1835 \newcommand*{\si@num@fixdp}{%
1836 \si@num@cntdigits{\si@num@postdec}%
1837 \ifx\@empty\si@num@postdec\@empty
1838 \si@tempcnta\z@\relax
1839 \fi
1840 \ifnum\si@tempcnta>\si@num@dp\relax
1841 \expandafter\si@num@round
1842 \else
1843 \ifnum\si@tempcnta<\si@num@dp\relax
1844 \expandafter\expandafter\expandafter\si@num@pad
1845 \fi
1846 \fi}

```

`\si@num@pad` Padding a number is a relatively easy matter. The number of digits is increased by adding “0” recursively.

```
1847 \newcommand*{\si@num@pad}{%
1848   \si@log@debug{Padding to \the\si@num@dp\space digits}%
1849   \loop\ifnum\si@tempcnta<\si@num@dp\si@num@pd\repeat}
1850 \newcommand*{\si@num@pd}{%
1851   \advance\si@tempcnta\@ne\relax
1852   \protected@edef\si@num@postdec{\si@num@postdec0}}
```

`\si@num@round` Rounding a number is more complicated. The main macro here relies on several others, and is simply a set-up and hand-off system.

```
\si@num@postrnd1853 \newcommand*{\si@num@prernd}{%
1854   \newcommand*{\si@num@postrnd}{%
1855     \newcommand*{\si@num@round}{%
1856       \si@log@debug{Rounding to \the\si@num@dp\space digits}%
1857       \si@num@reverse{\si@num@postdec}%
1858       \si@num@reverse{\si@num@predec}%
1859       \let\si@num@prernd\si@num@predec
1860       \let\si@num@postrnd\si@num@postdec
1861       \renewcommand*{\si@num@predec}{}%
1862       \renewcommand*{\si@num@postdec}{}%
1863       \si@switchfalse
1864       \si@num@rnd}
```

`\si@num@reverse` #1 : storage-macro

`\si@num@rev` The initial stage is to reverse the entire number, to make life easier. This is then undone by the other macros as the output is constructed.

```
1865 \newcommand*{\si@num@reverse}[1]{%
1866   \renewcommand*{\si@tempa}{}%
1867   \expandafter\si@num@rev#1\@empty\@empty
1868   \let#1\si@tempa}
1869 \def\si@num@rev#1#2\@empty{%
1870   \edef\si@tempa{#1\si@tempa}%
1871   \ifx\@empty#2\@empty\else
1872     \si@num@rev#2\@empty\@empty
1873   \fi}
```

`\si@num@rnd` The core looping macro of the system simply divides the flow between rounding before and after the decimal. The two routines are quite similar, but have subtly different requirements. Both take one character at a time from the input, increment if there is a carry digit, check its value, and add to the output string. This is very similar to the same routines used in other macro packages to achieve the same thing.

`\si@num@rndpre`
`\si@num@rndpost`

```
1874 \newcommand*{\si@num@rnd}{%
1875   \ifnum\si@tempcnta>\z@\relax
1876     \expandafter\si@num@rndpost
1877   \else
1878     \expandafter\si@num@rndpre
1879   \fi}
1880 \newcommand*{\si@num@rndpre}{%
1881   \expandafter\edef\expandafter\si@tempa\expandafter{%
1882     \expandafter\@car\si@num@prernd\@nil}%
1883   \expandafter\edef\expandafter\si@num@prernd\expandafter{%
```

```

1884 \expandafter\@cdr\si@num@prernd\@nil}%
1885 \si@tempcntb\si@tempa\relax
1886 \ifsi@switch
1887 \advance\si@tempcntb\@ne\relax
1888 \fi
1889 \si@switchfalse
1890 \ifnum\si@tempcntb=10\relax
1891 \si@tempcntb\z@\relax
1892 \expandafter\expandafter\expandafter\si@switchtrue
1893 \fi
1894 \edef\si@num@predec{\the\si@tempcntb\si@num@predec}%
1895 \ifx\@empty\si@num@prernd\@empty
1896 \ifsi@switch
1897 \edef\si@num@predec{1\si@num@predec}%
1898 \fi
1899 \else
1900 \expandafter\si@num@rnd
1901 \fi}
1902 \newcommand*{\si@num@rndpost}{%
1903 \expandafter\edef\expandafter\si@tempa\expandafter{%
1904 \expandafter\@car\si@num@postrnd\@nil}%
1905 \expandafter\edef\expandafter\si@num@postrnd\expandafter{%
1906 \expandafter\@cdr\si@num@postrnd\@nil}%
1907 \si@tempcntb\si@tempa\relax
1908 \ifsi@switch
1909 \advance\si@tempcntb\@ne\relax
1910 \fi
1911 \si@switchfalse
1912 \advance\si@num@dp\@ne\relax
1913 \ifnum\si@tempcnta>\si@num@dp\relax
1914 \advance\si@num@dp\m@ne\relax
1915 \else
1916 \advance\si@num@dp\m@ne\relax
1917 \ifnum\si@tempcnta>\si@num@dp\relax
1918 \ifnum\si@tempcntb>4\relax
1919 \expandafter\expandafter\expandafter\si@switchtrue
1920 \fi
1921 \else
1922 \ifnum\si@tempcntb=10\relax
1923 \si@tempcntb\z@\relax
1924 \expandafter\expandafter\expandafter\si@switchtrue
1925 \fi
1926 \edef\si@num@postdec{\the\si@tempcntb\si@num@postdec}%
1927 \fi
1928 \fi
1929 \advance\si@tempcnta\m@ne\relax
1930 \si@num@rnd}

```

`\si@num@int` #1 : integer-part

The formatting code for separating thousands is taken more-or-less directly from `Sistyle`. A few changes are made to fit the various conventions here. Following on from the code above, `\si@tempa` is used to store the integer part of the number, and `\si@tempb` is used for the decimal part.

```

1931 \newcommand*{\si@num@int}[1]{%

```

```

1932 \renewcommand*{\si@num@predec}{}%
1933 \ifsi@sepfour
1934     \si@num@intfmt{}#1\@empty\@empty\@empty
1935 \else
1936     \si@num@iffive{#1}
1937     {\si@num@intfmt{}#1\@empty\@empty\@empty}
1938     {\renewcommand*{\si@num@predec}{#1}}%
1939 \fi}

\si@num@iffive #1 : number
\si@num@five A test is needed for the presence of more than four characters.
1940 \newcommand*{\si@num@iffive}[1]{%
1941     \si@num@five#1\@empty\@empty\@empty\@empty\@empty\end}
1942 \def\si@num@five#1#2#3#4#5\end{%
1943     \ifx\@empty#5\@empty
1944         \expandafter\@secondoftwo
1945     \else
1946         \expandafter\@firstoftwo
1947     \fi}

\si@num@intfmt The business end of the integer formatter.
\si@num@fiint1948 \newcommand*{\si@num@intfmt}[4]{%
1949     \ifx\@empty#2\@empty
1950         \si@num@intsep#1\relax
1951     \else
1952         \ifx\@empty#3\@empty
1953             \si@num@intsep\@empty\@empty#1#2\relax
1954         \else
1955             \ifx\@empty#4\@empty
1956                 \si@num@intsep\@empty#1#2#3\relax
1957             \else
1958                 \si@num@fiint{#1#2#3#4}%
1959             \fi
1960         \fi
1961     \fi}
1962 \def\si@num@fiint#1\fi\fi\fi{\fi\fi\fi\si@num@intfmt{#1}}

\si@num@intsep For adding separation to integers, an extra function is needed.
1963 \newcommand*{\si@num@intsep}[4]{%
1964     \protected@edef\si@num@predec{\si@num@predec#1#2#3}%
1965     \if\relax#4\relax\else
1966         \protected@edef\si@num@predec{%
1967             \si@num@predec\ensuremath{\noexpand\si@digitsep}}%
1968         \expandafter\si@num@intsep\expandafter#4%
1969     \fi}

\si@num@dec #1 : decimal-part
\si@num@decfmt Formatting a decimal uses a similar mechanism, but with a few alterations
needed.
1970 \newcommand*{\si@num@dec}[1]{%
1971     \renewcommand*{\si@num@postdec}{}%
1972     \ifsi@sepfour
1973         \si@num@decfmt#1\@empty\@empty\@empty\@empty

```



```

1974 \else
1975   \si@num@iffive{#1}
1976   {\si@num@decfmt#1\@empty\@empty\@empty\@empty}
1977   {\protected@edef\si@num@postdec{\si@num@postdec#1}}%
1978 \fi}
1979 \newcommand*{\si@num@decfmt}[4]{%
1980   \protected@edef\si@num@postdec{\si@num@postdec#1#2#3}%
1981   \ifx\@empty#4\@empty%
1982   \else
1983     \protected@edef\si@num@postdec{%
1984       \si@num@postdec\ensuremath{\noexpand\si@digitsep}}%
1985     \expandafter\si@num@decfmt\expandafter#4%
1986   \fi}

```

`\si@num@procerr` Any error is recycled to to formatted correctly. The \pm sign, and any unit, are also added.

```

1987 \newcommand*{\si@num@procerr}{%
1988   \si@num@addunit
1989   \ensuremath{\si@pm}%
1990   \expandafter\si@num\expandafter{\si@num@err}}

```

`\si@num@sepxpart` A similar approach for numbers containing products, except the first token of the input has to be deleted.

```

1991 \newcommand*{\si@num@sepxpart}{%
1992   \si@num@addunit
1993   \ensuremath{\times}%
1994   \expandafter\expandafter\expandafter\si@num\expandafter
1995   \expandafter\expandafter{%
1996     \expandafter\@cdr\si@num@xpart\@nil}}

```

`\si@num@addunit` A short macro to add units if needed.

```

1997 \newcommand*{\si@num@addunit}{%
1998   \si@unt@numtrue
1999   \ifx\@empty\si@unt@unitarg\@empty\else
2000     \ifsi@repeatunits
2001       \si@unt@printunit{\si@unt@unitarg}%
2002     \fi
2003   \fi}

```

21.11 Formatting angles

`\ang` [#1]: keyval options

#2 : angle, either in decimal or deg;min;sec format

The approach used here is similar to that in `SIstyle`, but has been modified in a few ways.

```

2004 \si@newrobustcmd*{\ang}[2][]{%
2005   \beginngroup
2006     \sisetup{#1}%
2007     \si@fam@mode
2008     \si@log@debug{Processing \string\ang\space input `#2'}%
2009     \@makeother{;}%
2010     \makeatletter
2011     \scantokens{\si@ang@parse#2;;;\@nil}}

```

\si@ang@parse With the correct catcodes in place, processing can take place strip out the semi-colons. Here, the input must either contain no semi-colons or two semi-colons.

```

2012 \def\si@ang@parse#1;#2;#3;#4\nil{%
2013   \let\ifsi@ang@fixdp\ifsi@fixdp
2014   \si@fixdpfalse
2015   \si@ifmtarg{#4}
2016   {\si@log@debug{Angle argument contains no
2017     semi-colons:\MessageBreak decimal angle}%
2018     \si@ang@dec{#1}}{}}
2019   {\si@log@debug{Angle argument contains
2020     semi-colons:\MessageBreak degree-minute-second angle}%
2021     \renewcommand*{\si@tempa}{#4}%
2022     \renewcommand*{\si@tempb}{; ;}%
2023     \ifx\si@tempa\si@tempb\else
2024       \ifsi@strictarc
2025         \renewcommand*{\si@tempb}{; ;}%
2026         \ifx\si@tempa\si@tempb
2027           \si@log@err{Insufficient semi-colons in argument
2028             of \string\ang}{The argument of
2029             \string\ang\space must contain either no
2030             semi-colons or exactly two}%
2031         \else
2032           \si@log@err{Excess semi-colons in argument of
2033             \string\ang}{The argument of \string\ang\space
2034             must contain either no semi-colons or exactly
2035             two}%
2036         \fi
2037       \fi
2038     \fi
2039     \si@ang@arc{#1}{#2}{#3}}

```

\si@ang@dec Two tests are needed, in case the input format requires conversion.

```

\si@ang@arc2040 \newcommand*{\si@ang@dec}{%
2041   \let\si@ang@fix\@gobble
2042   \ifsi@ang@toarc
2043     \expandafter\si@ang@dectoarc
2044   \else
2045     \sisetup{padangle=none}\expandafter\si@ang@typeset
2046   \fi}
2047 \newcommand*{\si@ang@arc}{%
2048   \let\si@ang@fix\si@ang@arcfix
2049   \ifsi@ang@todec
2050     \expandafter\si@ang@arctodec
2051   \else
2052     \expandafter\si@ang@typeset
2053   \fi}

```

\ifsi@ang@fixdp #1 : number

\si@ang@fix A check is needed so that rounding and zero filling are only applied to the
\si@ang@arcfix seconds of an arc angle.

```

2054 \newif\ifsi@ang@fixdp
2055 \newcommand*{\si@ang@fix}[1]{%
2056 \newcommand*{\si@ang@arcfix}[1]{%

```

```

2057 \renewcommand*{\si@tempa}{second}%
2058 \renewcommand*{\si@tempb}{#1}%
2059 \ifx\si@tempa\si@tempb
2060   \ifsi@ang@fixdp
2061     \expandafter\expandafter\expandafter\si@fixdptrue
2062   \else
2063     \expandafter\expandafter\expandafter\si@fixdpfalse
2064   \fi
2065 \else
2066   \expandafter\si@fixdpfalse
2067 \fi}

```

`\si@ang@ifnum` #1 : **number**

A test is required to check that the provided data consists of numbers which can actually be processed by \TeX . This is achieved by using `\si@num@ifvalid` with a fixed list of valid characters.

```

2068 \newcommand*{\si@ang@ifnum}[1]{%
2069   \begingroup
2070     \renewcommand*{\si@numvalid}{0123456789,.-+}%
2071     \ifx\@empty#1\@empty
2072       \aftergroup\@firstoftwo
2073     \else
2074       \si@num@ifvalid{#1}
2075       {\aftergroup\@firstoftwo}
2076       {\aftergroup\@secondoftwo}%
2077     \fi
2078   \endgroup}

```

`\si@ang@arctodec` #1 : **degrees**

#2 : **minutes**

#3 : **seconds**

The business end of converting arcs to decimal angles is relatively straightforward, as it only needs one calculation. This has to be divided up, so that disaster does not strike if there are empty arguments; a check is also needed for the sign of the angle, so that the maths makes sense.

```

2079 \newcommand*{\si@ang@arctodec}[3]{%
2080   \let\si@ang@fix@gobble
2081   \ifnum\si@num@dp>\thr@@\relax
2082     \si@num@dp\thr@@\relax
2083   \fi
2084   \si@fixdptrue
2085   \si@ang@ifnum{#1}
2086   {\si@ang@ifnum{#2}
2087     {\si@ang@ifnum{#3}
2088       {\si@tempdima\z@\relax
2089         \renewcommand*{\si@tempa}{+}%
2090         \ifx\@empty#1\@empty\else
2091           \si@tempdima #1pt\relax
2092         \fi
2093         \ifdim\si@tempdima<\z@\relax
2094           \renewcommand*{\si@tempa}{-}%
2095         \fi
2096         \ifx\@empty#2\@empty\else

```

```

2097         \si@tempdima\dimexpr\si@tempdima\si@tempa
2098         #2pt/60\relax
2099     \fi
2100     \ifdim\si@tempdima<\z@\relax
2101         \renewcommand*\si@tempa{-}%
2102     \else
2103     \fi
2104     \ifx\@empty#3\@empty\else
2105         \si@tempdima\dimexpr\si@tempdima\si@tempa
2106         #3pt/3600\relax
2107     \fi
2108     \sisetup{numdecimal=.%}
2109     \expandafter\si@ang@typeset\expandafter{%
2110         \strip@pt\si@tempdima}{\{}}
2111     {\si@ang@notnum{#1}{#2}{#3}}
2112     {\si@ang@notnum{#1}{#2}{#3}}
2113     {\si@ang@notnum{#1}{#2}{#3}}

```

`\si@ang@dectoarc` #1 : number

`\si@ang@arcdeg` The conversion macros for decimal to arc angles. Life is more complex here than
`\si@ang@arcmin` above, even without the need for checks on the input. A number of separation
`\si@ang@arcsec` steps are needed, each of which needs a separate macro.

```

2114 \newcommand*\si@ang@dectoarc}[1]{%
2115   \let\si@ang@fix\si@ang@arcfix
2116   \si@ang@fixdptrue
2117   \ifnum\si@num@dp>\@ne\relax
2118     \si@num@dp\@ne\relax
2119   \fi
2120   \si@ang@ifnum{#1}
2121   {\si@tempdima\z@\relax
2122     \ifx\@empty#1\@empty\else
2123       \si@tempdima #1pt\relax
2124     \fi
2125     \si@ang@sepint{deg}%
2126     \si@tempdima\dimexpr\si@tempdima *60\relax
2127     \si@ang@sepint{min}%
2128     \edef\si@tempa{\the\dimexpr\si@tempdima *60\relax}%
2129     \expandafter\newcommand\expandafter*\expandafter{%
2130       \expandafter\si@ang@arcsec\expandafter}\expandafter{%
2131       \expandafter\si@ang@stript\si@tempa}%

```

A check is made for “o.o” seconds, which should be converted to simply “o”.

```

2132   \si@tempdima\z@\relax
2133   \edef\si@tempa{\the\si@tempdima}%
2134   \expandafter\renewcommand\expandafter*\expandafter{%
2135     \expandafter\si@tempa\expandafter}\expandafter{%
2136     \expandafter\si@ang@stript\si@tempa}%
2137   \ifx\si@tempa\si@ang@arcsec
2138     \renewcommand*\si@ang@arcsec{0}%
2139   \fi

```

To avoid adding zeros where they are not required, each part of the angle is now checked.

```

2140   \renewcommand*\si@tempa{0}%

```

```

2141 \ifx\si@ang@arcdeg\si@tempa
2142 \si@temptoks{}}}%
2143 \else
2144 \si@temptoks{{\si@ang@arcdeg}}}%
2145 \fi
2146 \ifx\si@ang@arcmin\si@tempa
2147 \si@temptoks\expandafter{\the\si@temptoks{}}}%
2148 \else
2149 \si@temptoks\expandafter{\the\si@temptoks{
2150 \si@ang@arcmin}}}%
2151 \fi
2152 \ifx\si@ang@arcsec\si@tempa
2153 \si@temptoks\expandafter{\the\si@temptoks{}}}%
2154 \else
2155 \si@temptoks\expandafter{\the\si@temptoks{
2156 \si@ang@arcsec}}}%
2157 \fi
2158 \expandafter\si@ang@typeset\the\si@temptoks}
2159 {\si@ang@notnum{#1}}{}}{}}

```

`\si@ang@sepint` #1 : either deg or min

`\si@ang@sint` Support macros for the conversion from decimal to arc angles.

```

\si@ang@strippt2160 \newcommand*{\si@ang@sepint}[1]{%
2161 \expandafter\si@ang@sint\the\si@tempdima\@empty
2162 \expandafter\let\csname si@ang@arc#1\endcsname\si@tempa}
2163 \def\si@ang@sint#1.#2\@empty{%
2164 \renewcommand*{\si@tempa}{#1}%
2165 \si@tempdima 0.#2\relax}
2166 \begingroup
2167 \catcode`P=12
2168 \catcode`T=12
2169 \lowercase{
2170 \renewcommand*{\si@tempa}{%
2171 \def\si@ang@strippt##1PT{##1}}}
2172 \expandafter\endgroup
2173 \si@tempa

```

`\si@ang@notnum` #1 : degrees

#2 : minutes

#3 : seconds

Not a TeX number: complain.

```

2174 \newcommand*{\si@ang@notnum}[3]{%
2175 \si@log@warn{Angle '#1;#2;#3' is not a pure
2176 number:\MessageBreak output will be as given}%
2177 \si@ang@typeset{#1}{#2}{#3}}

```

`\ifsi@ang@sign` A flag is needed to leave signs along for angles, where a zero value may still need a sign.

```
2178 \newif\ifsi@ang@sign
```

`\si@ang@typeset` #1 : degrees

#2 : minutes

#3 : seconds

The `\si@ang@set` macro does the work of assigning the degrees, minutes and seconds, and actually typesetting the result.

```
2179 \newcommand*{\si@ang@typeset}[3]{%
```

`\si@ang@deg`s First, the three macros that will contain the measures must exist.

```
\si@ang@mins2180 \ifsi@ang@padlarge
\si@ang@secs2181 \newcommand*{\si@ang@deg}{0\si@sym@degree}%
2182 \newcommand*{\si@ang@min}{0\si@sym@minute}%
2183 \newcommand*{\si@ang@sec}{0\si@sym@second}%
2184 \else
2185 \newcommand*{\si@ang@deg}{}%
2186 \newcommand*{\si@ang@min}{}%
2187 \newcommand*{\si@ang@sec}{}%
2188 \fi
```

`\si@ang@decimalsymbol` The current definition of `\si@decimalsymbol` needs to be saved.

```
2189 \protected@edef\si@ang@decimalsymbol{\si@decimalsymbol}%
```

`\si@ang@movesign` Either the signs need to be moved, or this needs to be killed off.

```
2190 \ifsi@astroang
2191 \let\si@ang@movesign\si@ang@astrosign
2192 \else
2193 \let\si@ang@movesign\@gobble
2194 \fi
```

`\si@ang@secnum` The arguments are now examined in reverse order. If they are empty, then
`\si@ang@minnum` nothing is done. Otherwise, the larger measures are zero-filled, if this has been requested. Some steps are needed to allow for addition of signs to numbers.

```
2195 \newcommand*{\si@ang@secnum}{\si@ang@num{second}}%
2196 \newcommand*{\si@ang@minnum}{\si@ang@num{minute}}%
2197 \si@ifnotmtarg{#3}
2198 {\si@log@debug{Found seconds '#3'}%
2199 \si@ang@ifnum{#3}
2200 {\ifdim #3 pt=\z@\relax\else
2201 \si@ang@signtrue
2202 \fi}}}%
2203 \renewcommand*{\si@ang@sec}{
2204 {\si@ang@secnum{#3}\si@sym@second}%
2205 \renewcommand*{\si@ang@min}{
2206 {\si@ang@pad{0\si@sym@minute}}}%
2207 \renewcommand*{\si@ang@deg}{
2208 {\si@ang@pad{0\si@sym@degree}}}%
2209 \si@ifnotmtarg{#2}
2210 {\si@log@debug{Found minutes '#2'}%
2211 \si@ang@ifnum{#2}
2212 {\ifdim #2 pt=\z@\relax\else
2213 \si@ang@signtrue
2214 \fi}}}%
2215 \renewcommand*{\si@ang@secnum}{%
2216 \si@ang@signlessnum{second}}%
2217 \renewcommand*{\si@ang@min}{
2218 {\si@ang@minnum{#2}\si@sym@minute}%
2219 \renewcommand*{\si@ang@deg}
```

```

2220      {\si@ang@pad{0\si@sym@degree}}}%
2221      \si@ifnotmtarg{#1}
2222      {\si@log@debug{Found degrees `#1'}%
2223      \renewcommand*{\si@ang@secnum}{%
2224      \si@ang@signlessnum{second}}}%
2225      \renewcommand*{\si@ang@minnum}{%
2226      \si@ang@signlessnum{minute}}}%
2227      \renewcommand*{\si@ang@deg}

```

The group here is needed to get the mechanism to move the symbol to work properly.

```

2228      {\si@ang@num{degree}{#1}%
2229      \si@sym@degree}}%
2230      \si@out@num
2231      {\si@ang@deg\si@anglesep\si@ang@mins\si@anglesep
2232      \si@ang@secs}%

```

The group opened by `\ang` is closed.

```

2233      \endgroup}

```

`\si@ang@pad` #1 : number

Padding is only added if requested; the zero is a literal.

```

2234 \newcommand*{\si@ang@pad}[1]{\ifsi@ang@padsmall #1\fi}

```

`\si@ang@num` #1 : one of degree, minute or second

`\si@ang@signlessnum` #2 : number

Modified versions of `\num`, one to typeset angles without a leading sign and the other with.

```

2235 \newcommand*{\si@ang@num}[2]{%
2236   \begingroup
2237     \si@ang@fix{#1}%
2238     \si@ang@movesign{#1}%
2239     \si@num{#2}%
2240   \endgroup}
2241 \newcommand*{\si@ang@signlessnum}[2]{%
2242   \begingroup
2243     \si@ang@fix{#1}%
2244     \si@ang@movesign{#1}%
2245     \sisetup{addsign=none}%
2246     \si@num{#2}%
2247   \endgroup}

```

`\si@ang@killdegree` A mechanism is needed to handle moving the angle unit signs for the `astroang` option. First, some support macros are needed.

`\si@ang@killminute`

```

\si@ang@killsecond2248 \newcommand*{\si@ang@killdegree}{\let\si@sym@degree\relax}

```

```

2249 \newcommand*{\si@ang@killminute}{\let\si@sym@minute\relax}

```

```

2250 \newcommand*{\si@ang@killsecond}{\let\si@sym@second\relax}

```

`\si@ang@astrosign` #1 : one of degree, minute or second

The method needs two steps, producing the sign over the decimal sign and preventing duplicate symbols appearing. This is based on a suggestion from Morten Høgholm, but using `TEX` internals as `\makebox` does not work here.

Note the need to correct for `\scriptspace` (thanks to Donald Arseneau for that).

```

2251 \newcommand*{\si@ang@astro sign}[1]{%
2252   \renewcommand*{\si@decimalsymbol}{%
2253     \setbox\si@tempboxa=\hbox{%
2254       \ensuremath{\{\si@ang@decimalsymbol\}}}%
2255     \si@tempdima\wd\si@tempboxa\relax
2256     \setbox\si@tempboxb=\hbox to\z@{%
2257       \hss\unhbox\si@tempboxa\hss}%
2258     \setbox\si@tempboxa=\hbox{%
2259       \csname si@sym@#1\endcsname\hskip-\scriptspace}%
2260     \si@tempdima\wd\si@tempboxa\relax
2261     \setbox\si@tempboxc=\hbox to\z@{%
2262       \hss\unhbox\si@tempboxa\hss}%
2263     \setbox\si@tempboxd=\hbox{%
2264       \usebox\si@tempboxb\usebox\si@tempboxc}%
2265     \ifdim\si@tempdima>\si@tempdima\relax
2266       \setbox\si@tempboxa=\hbox to\si@tempdima{%
2267         \hss\unhbox\si@tempboxd\hss}%
2268     \else
2269       \setbox\si@tempboxa=\hbox to\si@tempdima{%
2270         \hss\unhbox\si@tempboxd\hss}%
2271     \fi
2272     \usebox\si@tempboxa%
2273     \ifdim\si@tempdima>\si@tempdima\relax\else
2274       \hskip\scriptspace
2275     \fi}%
2276 \renewcommand*{\si@num@decimalhook}{\expandafter\aftergroup
2277   \csname si@ang@kill#1\endcsname}}%

```

21.12 Tabular material

The automatic formatting and alignment of numerical data in columns is handled here. The various other packages that work in this area are basically ripped-off here. The letters D, N and R are already taken by the other packages for numerical alignment, and so S (= siunitx) is chosen for the alignment of numerical material. The package also provides a second column type, *s*, for units (the letter is taken from `\si`).

`\NC@list` The first part of the job is to create the basic apparatus for the columns using the array package. To prevent any issues with the content of optional arguments to the new columns, so rearrangement is needed. The siunitx columns have to come *before* any other column definitions. This is achieved by saving `\NC@list`, creating the columns then restoring the list with the appropriate extra parts.

```

2278 \edef\si@tempa{%
2279   \noexpand\NC@do S\noexpand\NC@do s\the\NC@list}
2280 \newcolumnntype{S}{}
2281 \newcolumnntype{s}{}
2282 \NC@list\expandafter{\si@tempa}

```

`\NC@rewrite@S` [#1]: keyval options

`\NC@rewrite@s` Following the numprint approach, the `\NC@rewrite@...` macros are rewritten to collect the cell contents. This means messing with the internal macros of

another package, but there is no other way to do this. As array is a standard package from the tools bundle, this should be reasonably safe. Here the begin and end code needed is added to the existing list if \@temptokena, with the start and end macros unexpanded. Argument #1 contains any user setup options for this column. Passing an argument at this stage will cause issues, so each column type needs its own begin and end macros.

```

2283 \renewcommand*{\NC@rewrite@S}[1][ ]{%
2284   \edef\si@tempa{\the\@temptokena
2285     >\noexpand\si@tab@begin@S[#1]}c%
2286     <\noexpand\si@tab@end@S}%
2287   \@temptokena\expandafter{\si@tempa}%
2288   \NC@find}
2289 \renewcommand*{\NC@rewrite@s}[1][ ]{%
2290   \edef\si@tempa{\the\@temptokena
2291     >\noexpand\si@tab@begin@s[#1]}c%
2292     <\noexpand\si@tab@end@s}%
2293   \@temptokena\expandafter{\si@tempa}%
2294   \NC@find}

```

\si@tab@begin@S [#1]: keyval options

\si@tab@begin@s At this stage, the appropriate token gathering macro is activated, and the common starting macro is called. For the S column, the seperr is turned off, and an error is set to be raised by any “x-part” input.

```

2295 \newcommand*{\si@tab@begin@S}[1][ ]{%
2296   \si@log@debug{Processing S column cell contents}%
2297   \let\si@tab@gettok\si@tab@gettok@S
2298   \si@seperrfalse
2299   \renewcommand*{\si@num@sepxpart}{%
2300     \si@log@err{Multiple numbers not allowed in
2301       tables\MessageBreak Only the first number used}
2302     \@ehb}%
2303   \si@tab@begin[#1]}
2304 \newcommand*{\si@tab@begin@s}[1][ ]{%
2305   \si@log@debug{Processing s column cell contents}%
2306   \let\si@tab@gettok\si@tab@gettok@s
2307   \si@tab@begin[#1]}

```

\si@tab@toks Some storage is needed for the data to build up. In common with rccol and \si@tab@pretoks numprint, token registers are used for this (thus leaving problematic input to be \si@tab@posttoks handled later).

```

2308 \newtoks\si@tab@toks
2309 \newtoks\si@tab@pretoks
2310 \newtoks\si@tab@posttoks

```

\si@tab@begin [#1]: keyval options

The macro for gathering up input is common to both column types. It uses the method for rccol; the cell contents are collected by a second macro, which then stores all of the data in an appropriate token store.

```

2311 \newcommand*{\si@tab@begin}[1][ ]{%
2312   \begingroup
2313   \sisetup{#1}%
2314   \si@tab@toks{ }%

```

```

2315 \si@tab@pretoks{}%
2316 \si@tab@posttoks{}%
2317 \si@switchfalse
2318 \si@tab@gettok}

```

\si@tab@next A macro is needed for recursion when collecting cell contents.

```

2319 \newcommand*{\si@tab@next}{}

```

\si@tab@gettok@S #1 : token

The cell-content collection system is inherited from rccol. First, the current input is compared with a fixed list of macros which need custom handling.

```

2320 \newcommand*{\si@tab@gettok@S}[1]{%
2321 \ifx\tabularnewline#1\relax
2322 \let\si@tab@next\si@tab@newline@S
2323 \else
2324 \ifx\end#1\relax
2325 \let\si@tab@next\end
2326 \else
2327 \ifx\si@tab@end@S#1\relax
2328 \let\si@tab@next\si@tab@end@S
2329 \else
2330 \ifx\endtabular#1\relax
2331 \let\si@tab@next\endtabular
2332 \else
2333 \ifx\csmame#1\relax
2334 \let\si@tab@next\csmame
2335 \else
2336 \ifx\relax#1\relax
2337 \let\si@tab@next\relax
2338 \else

```

If the code gets to here, then recursion should occur.

```

2339 \let\si@tab@next\si@tab@gettok@S

```

Cells always contain \ignorespaces and \unskip. These can both be filtered out here and are saved to either the pre-number or post-number token store, as appropriate. This is done without fixing the destination in case the user adds addition copies to the input.

```

2340 \ifx\ignorespaces#1\relax
2341 \si@tab@othertok{#1}%
2342 \else
2343 \ifx\unskip#1\relax
2344 \si@tab@othertok{#1}%
2345 \else

```

If the current token is a valid numerical character, then the flag is set and the input is saved into the “processing” token register.

```

2346 \si@str@ifchrstr{#1}{\si@numvalid}
2347 {\si@switchtrue
2348 \si@log@debug{Found numerical cell
2349 contents '#1'}%
2350 \si@tab@toks=\expandafter{%
2351 \the\si@tab@toks#1}}

```

For non-numerical content, some testing is needed. A check is made to see if the current “token” is a single item. If it was wrapped in one set of braces before being picked up here, they will have been removed. So the input may now have more than one character to it. If it is a single token, testing for expendability first requires a check for another set of braces. This is handed off using `\futurelet`.

```

2352             {\si@tab@ifonechar{#1}
2353             {\futurelet\si@tempa\si@tab@bracetest#1}%
2354             {\si@tab@othertok{#1}}}%
2355         \fi
2356     \fi
2357 \fi
2358 \fi
2359 \fi
2360 \fi
2361 \fi
2362 \fi
2363 \si@tab@next}

```

`\si@tab@othertok #1 : token`

Tokens which are not part of a number can occur before or after a number. This is all sorted out here, with some debugging information for good measure.

```

2364 \newcommand*{\si@tab@othertok}[1]{%
2365 \si@log@debug{Found other cell contents '\unexpanded{#1}'}%
2366 \ifsi@switch
2367     \si@log@debug{Adding to post-numeral part}%
2368     \si@tab@posttoks=\expandafter{\the\si@tab@posttoks#1}%
2369 \else
2370     \si@log@debug{Adding to pre-numeral part}%
2371     \si@tab@pretoks=\expandafter{\the\si@tab@pretoks#1}%
2372 \fi}

```

`\si@tab@ifonechar #1 : token`

`\si@tab@onechar` A quick check for a single character.

```

2373 \newcommand*{\si@tab@ifonechar}[1]{\si@tab@onechar#1\@empty}
2374 \def\si@tab@onechar#1#2\@empty{%
2375 \ifx\@empty#2\@empty
2376     \expandafter\@firstoftwo
2377 \else
2378     \expandafter\@secondoftwo
2379 \fi}

```

`\si@tab@bracetest` For the `\futurelet` test, a separate macro is needed. If the current token starts with a brace, then it is added to the “other” stack. If not, a test can be made on its definition.

```

2380 \newcommand*{\si@tab@bracetest}{%
2381 \ifx\si@tempa\bgroup
2382     \expandafter\si@tab@othertok
2383 \else
2384     \expandafter\si@tab@exptest
2385 \fi}

```

`\si@tab@exptest` #1 : token
`\si@tab@meaning` The meaning of the token is compared to that for a short macro. If the two are
`\si@tab@strip` of the same type, expansion is attempted. In this case, the current pre-number
tokens are also typeset.

```

2386 \newcommand*{\si@tab@exptest}[1]{%
2387   \edef\si@tempa{\si@tab@meaning{#1}}%
2388   \ifx\si@tempa\si@tab@macro
2389     \si@log@debug{Expanding cell contents '\unexpanded{#1}'}%
2390     \the\si@tab@pretoks
2391     \si@tab@pretoks{}%
2392     \protected@edef\si@tab@next{\noexpand\si@tab@gettok@s#1}%
2393   \else
2394     \si@tab@othertok{#1}%
2395   \fi}
2396 \newcommand*{\si@tab@meaning}[1]{%
2397   \expandafter\si@tab@strip\meaning#1:->:->\@empty}
2398 \def\si@tab@strip#1:->#2:->#3\@empty{#1}

```

`\si@tab@macro` A supporter macro for the above which always expands to “macro” with every-
thing with category code 12.

```

2399 \newcommand*{\si@tab@macro}{}
2400 \edef\si@tab@macro{\si@tab@meaning{\si@tab@macro}}

```

`\si@tab@gettok@s` #1 : token
The situation for the `s` column is less complex, as there is only one store and
expansion is not an issue.

```

2401 \newcommand*{\si@tab@gettok@s}[1]{%
2402   \ifx\tabularnewline#1\relax
2403     \let\si@tab@next\si@tab@newline@s
2404   \else
2405     \ifx\end#1\relax
2406       \let\si@tab@next\end
2407     \else
2408       \ifx\si@tab@end@s#1\relax
2409         \let\si@tab@next\si@tab@end@s
2410       \else
2411         \ifx\endtabular#1\relax
2412           \let\si@tab@next\endtabular
2413         \else
2414           \ifx\csize#1\relax
2415             \let\si@tab@next\csize
2416           \else
2417             \ifx\relax#1\relax
2418               \let\si@tab@next\relax
2419             \else
2420               \let\si@tab@next\si@tab@gettok@s
2421             \ifx\ignorespaces#1\relax\else
2422               \ifx\unskip#1\relax\else
2423                 \si@tab@toks=\expandafter{%
2424                   \the\si@tab@toks#1}%
2425                 \si@log@debug{Found cell contents '#1'}%
2426               \fi
2427             \fi

```

```

2428         \fi
2429     \fi
2430     \fi
2431     \fi
2432 \fi
2433 \fi
2434 \si@tab@next}

```

`\si@tab@end@S` The end macros are similar, but with some minor differences. In both cases, the appropriate filling is carried out. For the `S` column, this depends on the cell contents, whereas for the `s` column the fill is always the same. Output of a number in an `S` column is only attempted if one was found, otherwise the cell contents will all be in `\si@tab@pretoks`.

```

2435 \newcommand*{\si@tab@end@S}{%
2436     \ifsi@switch
2437         \let\si@tab@lfill\si@tab@lfill@S
2438         \let\si@tab@rfill\si@tab@rfill@S
2439     \else
2440         \let\si@tab@rfill\si@tab@rfill@t
2441         \let\si@tab@lfill\si@tab@lfill@t
2442     \fi
2443     \si@tab@lfill\relax
2444     \ifsi@switch
2445         \the\si@tab@pretoks
2446         \si@tab@numout
2447         \the\si@tab@posttoks
2448     \else
2449         \the\si@tab@pretoks
2450     \fi
2451     \si@tab@rfill\relax
2452 \endgroup}
2453 \newcommand*{\si@tab@end@s}{%
2454     \si@tab@lfill@s\relax
2455     \ignorespaces
2456     \expandafter\si\expandafter{\the\si@tab@toks}%
2457     \unskip
2458     \si@tab@rfill@s\relax
2459 \endgroup}

```

`\si@tab@newline@S` If the column is the final one read, then some work is needed with the `\tabularnewline` macro. Output has to happen *before* the new line, then the ending macro is made safe before calling the \LaTeX line end. If the user makes use of the primitive `\cr` then this problem does not arise as `\si@tab@end@...` is called correctly.

```

2460 \newcommand*{\si@tab@newline@S}{%
2461     \si@tab@end@S
2462     \hfil\relax
2463     \let\si@tab@end\si@tab@end@S
2464     \renewcommand*{\si@tab@end@S}{\let\si@tab@end@S\si@tab@end}%
2465     \tabularnewline}
2466 \newcommand*{\si@tab@newline@s}{%
2467     \si@tab@end@s
2468     \hfil\relax

```

```

2469 \let\si@tab@end\si@tab@end@s
2470 \renewcommand*{\si@tab@end@s}{\let\si@tab@end@s\si@tab@end}%
2471 \tabularnewline

\si@tempcnta The second part of the tabular code is concerned with typesetting numbers
\si@tempcntb in S columns with the appropriate alignment. Counters are needed for the
digit-counting system.
2472 \newcount\si@tempcnta
2473 \newcount\si@tempcntb

\si@tab@numout If a number is found, then some secondary processing is needed to format it
correctly.
2474 \newcommand*{\si@tab@numout}{%
2475 \si@num@intabtrue
2476 \ifsi@tab@fixed
2477 \ifsi@tabautofit
2478 \si@num@dp\si@tab@mantpostcnt\relax
2479 \expandafter\expandafter\expandafter\si@fixdptrue
2480 \fi
2481 \fi
2482 \expandafter\si@num\expandafter{\the\si@tab@toks}%
2483 \si@tab@format}

\si@tab@prebox The various boxes needed for the column centring are declared Unlike the
\si@tab@postbox dcolumn original, private boxes are used here. \si@tempboxa is used when a
\si@tab@midbox space to measure one of the constituents is needed; it is never used for output.
\si@tab@expbox2484 \newbox\si@tab@prebox
2485 \newbox\si@tab@midbox
2486 \newbox\si@tab@postbox
2487 \newbox\si@tab@expbox

\si@tab@format The formatting set up is taken from dcolumn, but with control of the output form
the stored information. The choice of a variable (decimal-centred) column or
fixed width boxes is made.
2488 \newcommand*{\si@tab@format}{%
2489 \ifsi@tab@fixed
2490 \expandafter\si@tab@fixed
2491 \else
2492 \expandafter\si@tab@unfixed
2493 \fi

A hack to get the correct colour everywhere without too much work.
2494 \ifsi@colourvalues
2495 \si@fam@colourcmd{\si@valuecolour}%
2496 \fi
2497 \box\si@tab@prebox\box\si@tab@midbox\box\si@tab@postbox%
2498 \box\si@tab@expbox}

\si@tab@unfixed When the width of the contents is not fixed, the system creates a block in which
the decimal marker is always at the centre. This is achieved by placing the pre-
and post-decimal parts of the number in boxes. The wider one is then used to set
up the column width, by resizing the other one.
2499 \newcommand*\si@tab@unfixed{%

```

```

2500 \si@log@debug{Using variable width S column}%
2501 \ifx\@empty\si@num@out\@empty
2502   \setbox\si@tab@midbox=\hbox
2503     {\phantom{\ensuremath{\{\si@decimalsymbol\}}}}%
2504 \else
2505   \setbox\si@tab@midbox=\hbox
2506     {\ensuremath{\{\si@decimalsymbol\}}}%
2507 \fi
2508 \protected@edef\si@num@out{\si@num@out\si@tab@expout}%
2509 \setbox\si@tab@prebox=\hbox
2510   {\expandafter\si@out@num\expandafter{\si@tab@out}}%
2511 \setbox\si@tab@postbox=\hbox
2512   {\expandafter\si@out@num\expandafter{\si@num@out}}%
2513 \ifdim\wd\si@tab@prebox>\wd\si@tab@postbox\relax
2514   \setbox\si@tab@postbox=\hbox to\wd\si@tab@prebox%
2515     {\unhbox\si@tab@postbox\hfill}%
2516 \else
2517   \setbox\si@tab@prebox=\hbox to\wd\si@tab@postbox%
2518     {\hfill\unhbox\si@tab@prebox}%
2519 \fi
2520 \setbox\si@tab@expbox=\hbox{}

```

`\si@tab@predim` Some storage dimensions are declared.

```

\si@tab@postdim2521 \newdimen\si@tab@predim
\si@tab@expdim2522 \newdimen\si@tab@postdim
\si@tempdima2523 \newdimen\si@tab@expdim
\si@tempdimb2524 \newdimen\si@tempdima
2525 \newdimen\si@tempdimb

```

`\si@tab@sp` A short macro to control superscript.

```
2526 \newcommand*{\si@tab@sp}{}

```

`\si@tab@fixed` The column is not centred on the decimal marker; the user specifies how many characters on each side are allowed for. First, the width of a character is measured, and stored.

```

2527 \newcommand*{\si@tab@fixed}{%
2528 \si@log@debug{Using fixed-width S column}%
2529 \let\si@tab@sp\relax
2530 \setbox\si@tab@midbox=\hbox{}%
2531 \setbox\si@tab@expbox=\hbox{}%
2532 \setbox\si@tempboxa=\hbox{\si@out@num{1}}%
2533 \si@tempdima\wd\si@tempboxa\relax

```

The width for the two output boxes is set up.

```

2534 \si@tab@predim\the\si@tab@mantprecnt\si@tempdima\relax
2535 \si@tab@sepcorr{mantpre}{pre}%
2536 \si@tab@postdim\si@tab@mantpostcnt\si@tempdima\relax
2537 \setbox\si@tempboxa=\hbox{\ensuremath{\{\si@decimalsymbol\}}}%
2538 \ifnum\si@tab@mantpostcnt=\z@\relax\else
2539   \advance\si@tab@postdim\wd\si@tempboxa\relax
2540 \fi
2541 \si@tab@sepcorr{mantpost}{post}%

```

If space is needed for an exponent, it needs to be allowed for in the exponent box dimension. First, the digits of the two parts are checked for; the width of a character is altered to be superscript.

```

2542 \setbox\si@tempboxa=\hbox{\si@out@num{^1}}}%
2543 \si@tempdima\wd\si@tempboxa\relax
2544 \ifnum\si@tab@expprecnt>\z@\relax
2545   \si@tab@expdim\si@tab@expprecnt\si@tempdima\relax
2546   \si@tab@sepcorr{exp}{exp}%
2547 \fi
2548 \let\si@tab@sp\sp
2549 \ifnum\si@tab@exppostcnt>\z@\relax
2550   \advance\si@tab@expdim\si@tab@exppostcnt\si@tempdima\relax
2551   \setbox\si@tempboxa=\hbox{%
2552     \ensuremath{^{\si@decimalsymbol}}}%
2553   \advance\si@tab@expdim\wd\si@tempboxa\relax
2554   \si@tab@sepcorr{exp}{exp}%
2555 \fi

```

Space is reserved for signs.

```

2556 \setbox\si@tempboxa=\hbox{\ensuremath{-}}%
2557 \ifsi@tab@mantsign
2558   \advance\si@tab@predim\wd\si@tempboxa\relax
2559 \fi
2560 \setbox\si@tempboxa=\hbox{\ensuremath{^{\{-}}}%
2561 \ifsi@tab@expsign
2562   \advance\si@tab@expdim\wd\si@tempboxa\relax
2563 \fi

```

Now, if there is space to be saved for an exponent under any circumstances, the space for the “ $\times 10$ ” part is needed. This is done by adding both counters together, then using this result for the logic.

```

2564 \si@tempcnta\si@tab@expprecnt\relax
2565 \advance\si@tempcnta\si@tab@exppostcnt\relax
2566 \ifnum\si@tempcnta>\z@\relax
2567   \setbox\si@tempboxa=\hbox{\ensuremath{%
2568     {} \si@expproduct{} \si@expbase}}}%
2569   \advance\si@tab@expdim\wd\si@tempboxa\relax
2570 \fi

```

Finally for the box dimensions, if the exponent is not aligned, the space reserved for it is added to the post box.

```

2571 \ifsi@tab@alignexp\else
2572   \advance\si@tab@postdim\si@tab@expdim\relax
2573 \fi

```

With the boxes set up, the contents can be sorted out. A bit of shuffling may be needed, depending on the treatment of exponents.

```

2574 \setbox\si@tab@prebox=\hbox to\si@tab@predim{\hss\hfill
2575   \expandafter\si@out@num\expandafter{\si@tab@out}}%
2576 \ifx\@empty\si@num@out\@empty
2577   \setbox\si@tab@postbox=\hbox to\si@tab@postdim
2578     {\expandafter\si@out@num\expandafter{\si@num@out}\hfil}%
2579 \else
2580   \ifsi@tab@alignexp\else
2581     \protected@edef\si@num@out{\si@num@out\si@tab@expout}%

```



```

2582 \fi
2583 \setbox\si@tab@postbox=\hbox to\si@tab@postdim
2584 {\ensuremath{\{\si@decimalsymbol\}}\expandafter\si@out@num
2585 \expandafter{\si@num@out}\hfil}%
2586 \fi
2587 \ifx\@empty\si@tab@expout\@empty
2588 \ifsi@tabalignexp
2589 \setbox\si@tab@expbox=\hbox to\si@tab@expdim{\hfil}%
2590 \fi
2591 \else
2592 \ifsi@tabalignexp
2593 \setbox\si@tab@expbox=\hbox to\si@tab@expdim
2594 {\expandafter\si@out@num\expandafter{\si@tab@expout}%
2595 \hfil}%
2596 \fi
2597 \fi}

```

`\si@tab@sepcorr` #1 : one of mantpre, mantpost, exppre or exppost

#2 : one of pre, post or exp

A spacing correction is needed *if* the number of digits to be allowed for will lead to the introduction of a separator. A counter and dimension are needed for the testing.

```

2598 \newcommand*\si@tab@sepcorr}[2]{%
2599 \expandafter\si@tempcnta\expandafter\the
2600 \csname si@tab@#1cnt\endcsname\relax

```

Calculate how many groups of three there are, then allow for not separating four characters if `\ifsi@sepfour` is false.

```

2601 \divide\si@tempcnta\thr@@\relax
2602 \ifsi@sepfour\else
2603 \expandafter\ifnum\expandafter\the
2604 \csname si@tab@#1cnt\endcsname=4\relax
2605 \si@tempcnta\z@\relax
2606 \fi
2607 \fi

```

The width of the separators is measured, and the correct number of separator widths are added to the box dimension.

```

2608 \setbox\si@tempboxa=\hbox{%
2609 \ensuremath{\si@tab@sp{\si@digitsep}}}%
2610 \expandafter\advance\csname si@tab@#2dim\endcsname
2611 \si@tempcnta\wd\si@tempboxa}

```

21.13 Units

`\SI` [#1]: keyval options

#2 : number

There are two types of user macros for the units system; those for defining new units, prefixes and powers, and those for using them. There are two macros for using units, `\SI` and `\si`, which work in very similar ways. argument to `\SI`

```

2612 \si@newrobustcmd*\SI}[2][]{%
2613 \@ifnextchar[%]
2614 {\si@SI[#1]{#2}}
2615 {\si@SI[#1]{#2}[]}}

```

```

\si [#1]: keyval options
#2 : unit
\si is just an alias for \SI with no number; everything is handed off into an
internal macro. The internal macro also handles the optional prefix
2616 \si@newrobustcmd*{\si}[2][\si@SI[#1]{}][\{#2}]

```

```

\newunit [#1]: keyval options
\renewunit #2 : unit
\provideunit #3 : symbol

```

The `\newunit` and `\renewunit` macros create the new unit macros. To allow a mechanism for checking an existing definition, these macros simply carry out the appropriate tests, before handing off to the internal macro. `\@ifdefinable` is not used here as a customised error is desirable. Other than that, the code here gives very similar results to `\newcommand` and `\renewcommand`. Finally, `\provideunit` adds the unit definition only if it does not already exist.

```

2617 \newcommand*{\newunit}[3][\{
2618   \si@ifdefinable{#2}
2619   {\si@unt@defunit[#1]{#2}{#3}}
2620   {\si@log@err{Unit \string#2 already defined!}\@eha}}
2621 \newcommand*{\renewunit}[3][\{
2622   \si@ifdefinable{#2}
2623   {\si@log@err{Unit \string#2 undefined}\@ehc
2624     \si@unt@defunit[#1]{#2}{#3}}
2625   {\si@log@inf{Redefining unit \string#2}%
2626     \si@unt@defunit[#1]{#2}{#3}}}
2627 \newcommand*{\provideunit}[3][\{
2628   \si@ifdefinable{#2}
2629   {\si@unt@defunit[#1]{#2}{#3}}
2630   {}}

```

```

\newprefix [#1]: binary
\renewprefix #2 : multiple
\provideprefix #3 : powers-ten
#4 : symbol

```

The multiples of units are defined here; very similar code is used to the `\newunit`, *etc.*, macros. The multiple prefixes cannot take an optional argument, and must represent some power. Hence the arguments required are different.

```

2631 \newcommand*{\newprefix}[4][\{
2632   \si@ifdefinable{#2}
2633   {\si@unt@defprefix[#1]{#2}{#3}{#4}}
2634   {\si@log@err{Prefix \string#2 already defined!}\@eha}}
2635 \newcommand*{\renewprefix}[4][\{
2636   \si@ifdefinable{#2}
2637   {\si@log@err{Prefix \string#2 undefined}\@ehc
2638     \si@unt@defprefix[#1]{#2}{#3}{#4}}
2639   {\si@log@inf{Redefining prefix \string#2}%
2640     \si@unt@defprefix[#1]{#2}{#3}{#4}}}
2641 \newcommand*{\provideprefix}[4][\{
2642   \si@ifdefinable{#2}
2643   {\si@unt@defprefix[#1]{#2}{#3}{#4}}
2644   {}}

```

```

\newpower [#1]: post
\renewpower #2 : number
\providepower #3 : power

```

Here power multiples for units are set up. As with units and multiples, a layered approach is used to keep things easy to maintain.

```

2645 \newcommand*{\newpower}[3][\%
2646   \si@ifdefinable{#2}
2647     {\si@unt@defpower[#1]{#2}{#3}}
2648     {\si@log@err{Power \string#2 already defined!}\@eha}}
2649 \newcommand*{\renewpower}[3][\%
2650   \si@ifdefinable{#2}
2651     {\si@log@err{Power \string#2 undefined}\@ehc
2652     \si@unt@defpower[#1]{#2}{#3}}
2653     {\si@log@inf{Redefining power \string#2}%
2654     \si@unt@defpower[#1]{#2}{#3}}}}
2655 \newcommand*{\providepower}[3][\%
2656   \si@ifdefinable{#2}
2657     {\si@unt@defpower[#1]{#2}{#3}}
2658     {}

```

```

\ifsi@unt@num A flag is needed to tell the processor whether there is a number, to get the correct
              spacing.

```

```

2659 \newif\ifsi@unt@num

```

```

\si@unt@unitarg A storage macro for the argument of the unit macro.

```

```

2660 \newcommand*{\si@unt@unitarg}{}

```

```

\si@SI [#1]: keyval options
\si@unt@SIopts #2 : unit
                [#3]: preunit
                #4 : unit

```

The internal processing starts with `\si@SI`, which processes the second optional argument to `\SI` (which is empty for `\si`). Everything is set up in a group, and processing begins by handling the options.

```

2661 \newcommand*{\si@unt@SIopts}{}
2662 \def\si@SI[#1]#2[#3]#4{%
2663   \begingroup
2664     \let\fg\SIfg
2665     \si@ifnotmtarg{#1}
2666     {\sisetup{#1}%
2667     \renewcommand*{\si@unt@SIopts}{#1}}%

```

The prefix unit is handled before any processing of the number; the flags are set to get spacing correct.

```

2668   \si@unt@numfalse
2669   \si@xspacefalse
2670   \si@ifnotmtarg{#3}
2671   {\si@log@debug{Prefix unit found}%
2672   \si@unt@printunit{#3}}%

```

The numerical argument may be empty, in which case no extra space should be produced.

```

2673   \si@ifnotmtarg{#4}

```

```

2674      {\renewcommand*{\si@unt@unitarg}{#4}}%
2675      \si@ifnotmtarg{#2}
2676      {\si@log@debug{Number found in \string\SI\space
2677        argument}%
2678        \ifsi@repeatunits\else
2679          \ifsi@trapambigerr
2680            \expandafter\expandafter\expandafter
2681              \si@num@ambigerrtrue
2682          \fi
2683        \fi
2684        \num{#2}%
2685        \si@unt@numtrue}%

```

If there is a unit, a check is needed in case the units need to have a power added.

```

2686      \si@ifnotmtarg{#4}
2687      {\si@ifmtarg{#2}
2688        {\si@unt@printunit{#4}}
2689        {\si@tempcnta\z@\relax
2690          \ifsi@addunitpower
2691            \si@unt@countx{#2}%
2692          \fi
2693          \ifnum\si@tempcnta>\z@\relax
2694            \advance\si@tempcnta\@ne\relax
2695            \edef\si@tempa{\noexpand\tothe{\si@tempcnta}}%
2696            \renewcommand*{\si@tempb}{#4}%
2697            \expandafter\expandafter\expandafter
2698              \si@unt@printunit\expandafter\expandafter
2699                \expandafter{%
2700                  \expandafter\si@tempb\si@tempa}%
2701          \else
2702            \si@unt@printunit{#4}%
2703          \fi}}%
2704      \endgroup}

```

`\si@unt@countx` A short macro to count up any multiplication in numerical input.

```

2705 \newcommand*{\si@unt@countx}[1]{%
2706   \si@tempcnta\z@\relax
2707   \expandafter\si@unt@cntx#1\@empty\@empty}
2708 \def\si@unt@cntx#1#2\@empty{%
2709   \si@str@ifchrstr{#1}{\si@numprod}
2710   {\advance\si@tempcnta\@ne\relax}
2711   {}%
2712   \ifx\@empty#2\@empty\else
2713     \si@unt@cntx#2\@empty\@empty
2714   \fi}

```

`\si@unt@ifliteral` #1 : unit

`\ifsi@unt@littest` The next stage of the processor is to determine whether or not the argument of the unit macro is processable. For literal arguments, this is not the case, and the argument is typeset “as is”. On the other hand, any units, *etc.*, declared by the package will work with the processor, and so need to be executed before typesetting the result.

```

2715 \newif\ifsi@unt@littest
2716 \newcommand*{\si@unt@ifliteral}[1]{%

```

```

2717 \begingroup
2718 \si@unt@litesttrue

```

The test relies on any non-processable test having some width; hopefully, this should be the case.

```

2719 \setbox\si@tempboxa=\hbox{\si@unt@out{#1}}%
2720 \ifdim\wd\si@tempboxa>\z@ \relax
2721 \aftergroup\@firstoftwo
2722 \else
2723 \aftergroup\@secondoftwo
2724 \fi
2725 \endgroup}

```

```
\ifsi@unt@litout #1 : unit
```

`\si@unt@printunit` The printing macro uses the above test to determine how to act. It then carries out the appropriate action: either typesetting or executing. A flag is also provided so that any macro units inside a partially-literal argument will work (this is also needed to emulate `unitsdef`).

```

2726 \newif\ifsi@unt@litout
2727 \newcommand*{\si@unt@printunit}[1]{%
2728 \si@unt@ifliteral{#1}

```

The unit includes one or more literal items; typeset using the unit typesetting macro.

```

2729 {\si@log@debug{Literal items found in unit
2730 argument:\MessageBreak outputting without further
2731 processing}%
2732 \si@unt@litouttrue
2733 \si@unt@addvaluesep
2734 \si@unt@out{#1}}

```

For processable output, the argument is executed; the macros are all designed for this.

```

2735 {\si@log@debug{Macro unit found:\MessageBreak
2736 processing to format output}%
2737 \si@unt@init
2738 \advance\si@unt@depthcnt\@ne \relax
2739 #1%
2740 \si@unt@final}}

```

`\si@unt@addvaluesep` To ensure no problems pop up with expansion, adding the value–unit space is handled by a macro.

```

\si@unt@addvalsep \si@unt@litvalsep2741 \newcommand*{\si@unt@addvaluesep}{%
\si@unt@stackvalsep2742 \ifsi@unt@num
2743 \expandafter\si@unt@addvalsep
2744 \fi}
2745 \newcommand*{\si@unt@addvalsep}{%
2746 \ifsi@unt@litout
2747 \expandafter\si@unt@litvalsep
2748 \else
2749 \expandafter\si@unt@stackvalsep
2750 \fi}
2751 \newcommand*{\si@unt@stackvalsep}{%
2752 \protected@edef\si@unt@spstack{\si@valuesep}}

```

```

2753 \newcommand*{\si@unt@litvalsep}{%
2754   \nobreak\ensuremath{\si@valuesep}\nobreak}

\si@unt@spstack The initialisation macro sets up the various switches, and clears the storage areas
\si@unt@stacka for the formatted output. There are two stacks, as when typesetting as fractions,
\si@unt@stackb the two parts of the number have to be stored separately. The depth counter is
\si@unt@unitcnta used to tell when recursion ends in the processor. The “first” switch is needed as
\si@unt@unitcntb the depth counter will not be at one for items processed by \SI.
\si@unt@depthcnt2755 \newcommand*{\si@unt@spstack}{}
\ifsi@unt@first2756 \newcommand*{\si@unt@stacka}{}
\ifsi@unt@first2757 \newcommand*{\si@unt@stackb}{}
\si@unt@init2758 \newcount\si@unt@unitcnta
2759 \newcount\si@unt@unitcntb
2760 \newcount\si@unt@depthcnt
2761 \newif\ifsi@unt@first
2762 \si@unt@depthcnt@m@ne\relax
2763 \newcommand*{\si@unt@init}{%
2764   \begingroup
2765     \si@unt@litoutfalse
2766     \si@unt@litprefixfalse
2767     \si@unt@firsttrue
2768     \si@unt@perfalse
2769     \si@unt@perseenfalse
2770     \si@unt@prepowerfalse
2771     \si@unt@depthcnt\z@\relax
2772     \si@unt@powerdim\z@\relax
2773     \si@unt@unitcnta\z@\relax
2774     \si@unt@unitcntb\z@\relax
2775     \si@unt@prefixcnt\z@\relax
2776     \renewcommand*{\si@unt@spstack}{}%
2777     \renewcommand*{\si@unt@stacka}{}%
2778     \renewcommand*{\si@unt@stackb}{}%
2779     \renewcommand*{\si@unt@holdstacka}{}%
2780     \renewcommand*{\si@unt@holdstackb}{}%
2781     \renewcommand*{\si@unt@lastadda}{space}%
2782     \renewcommand*{\si@unt@lastaddb}{space}}

\si@unt@final The finalisation macro finishes off the output and resets the flags.
2783 \newcommand*{\si@unt@final}{%
2784   \si@unt@third
2785   \si@unt@stackout
2786   \endgroup
2787   \ifsi@xspace
2788     \expandafter\expandafter\expandafter\xspace
2789   \fi}

\si@unt@defunit [#1]: keyval options
#2 : unit
#3 : symbol
The internal macro for defining a unit does not check for redefinition; that is
done by the user macros.
2790 \newcommand*{\si@unt@defunit}[3][]{%
2791   \si@log@debug{Declaring unit \string#2 with \MessageBreak
2792     meaning \string#3}%

```

The optional argument needs to be saved. The macro name is reversed so that life is easier with the expansions here.

```
2793 \si@ifnotmtarg{#1}
2794     {\expandafter\@namedef\expandafter{%
2795       \expandafter\@gobble\string#2@opt@unt@si}{#1}}%
```

The unit macro itself is now defined. The definition simply selects the correct path for the rest of the processing to go down.

```
2796 \protected\def#2{%
2797     \ifsi@allowoptarg
2798     \expandafter\si@unt@withopt
2799     \else
2800     \expandafter\si@unt@noopt
2801     \fi
2802     {#2}{#3}}}
```

```
\si@unt@withopt #1 : unit
\si@unt@noopt   #2 : symbol
```

To allow the correct expansion, the potential optional argument to a unit macro has to come last. So `\@ifnextchar` is needed to detect it and pass data through. To keep variation down, when the argument is not allowed, the empty [] is supplied.

```
2803 \newcommand*{\si@unt@withopt}[2]{%
2804     \@ifnextchar[%]
2805     {\si@unt@opt{#1}{#2}}
2806     {\si@unt@opt{#1}{#2}[]}}
2807 \newcommand*{\si@unt@noopt}[2]{\si@unt@opt{#1}{#2}[]}
```

```
\si@unt@opt #1 : unit
            #2 : symbol
            [#3]: number
```

The optional argument to the unit macro (if present) is converted to a normal one for ease. The correct route for processing is then picked.

```
2808 \def\si@unt@opt#1#2[#3]{%
2809     \ifsi@unt@littest
2810     \expandafter\si@gobblethree
2811     \else
```

For literal output, the second argument is all that is needed.

```
2812     \ifsi@unt@litout
2813     \expandafter\expandafter\expandafter\@gobbletwo
2814     \else
2815     \expandafter\expandafter\expandafter\si@unt@unit
2816     \fi
2817     \fi
2818     {#3}{#1}{#2}}
```

`\si@gobblethree` \LaTeX does not have a `\@gobblethree` macro, but one is needed.

```
2819 \long\def\si@gobblethree #1#2#3{}
```

```
\si@unt@defprefix [#1]: binary
\ifsi@unt@litprefix #2 : multiple
                   #3 : powers-ten
```

#4 : symbol

As with units, multiples are defined by an internal macro.

```
2820 \newif\ifsi@unt@litprefix
2821 \si@unt@litprefixtrue
2822 \newcommand*{\si@unt@defprefix}[4][\%
2823 \si@log@debug{Declaring multiple \string#1 with\MessageBreak
2824 meaning \string#4}\%
```

The optional argument is saved, using `\def` as no check is made on an existing definition of the storage macro.

```
2825 \expandafter\expandafter\expandafter\def\expandafter
2826 \csname\expandafter\@gobble\string#2@opt@si@endcsname{#1}%
2827 \protected\def#2{%
2828 \ifsi@unt@littest
2829 \expandafter\si@gobblethree
2830 \else
2831 \ifsi@unt@litout
2832 \expandafter\expandafter\expandafter\@gobbletwo
2833 \else
2834 \ifsi@unt@litprefix
2835 \expandafter\expandafter\expandafter\expandafter
2836 \expandafter\expandafter\expandafter\@gobbletwo
2837 \else
2838 \expandafter\expandafter\expandafter\expandafter
2839 \expandafter\expandafter\expandafter\si@unt@prefix
2840 \fi
2841 \fi
2842 \fi
2843 {#2}{#3}{#4}}}
```

```
\si@unt@defpower [#1]: post
#2 : number
#3 : power
```

The definition of powers is complicated by the need to handle both those given before units (such as `\cubic`) and those given after (e.g. `\cubed`). This means that an optional argument is needed.

```
2844 \newcommand*{\si@unt@defpower}[3][\%
2845 \si@log@debug{Declaring power \string#2 with\MessageBreak
2846 meaning \string#3}\%
```

Once again the optional argument is saved.

```
2847 \expandafter\expandafter\expandafter\def\expandafter
2848 \csname\expandafter\@gobble\string#2@opt@si@endcsname{#1}%
2849 \protected\def#2{%
2850 \ifsi@unt@littest
2851 \expandafter\@gobbletwo
2852 \else
```

The literal output here does not need to gobble anything.

```
2853 \ifsi@unt@litout
2854 \expandafter\expandafter\expandafter\si@unt@litpower
2855 \else
2856 \expandafter\expandafter\expandafter\si@unt@power
2857 \fi
```



```

2858     \fi
2859     {\#2}{\#3}}

```

```

\si@unt@unithook #1 : number
\si@unt@unit     #2 : unit
                  #3 : symbol

```

The macro for units is actually a processor, rather than typesetting anything, which is handled elsewhere. The first argument to the macro is optional, but does not have square brackets to keep things simple with gobbling.

```

2860 \newcommand*{\si@unt@unithook}{}
2861 \newcommand*{\si@unt@unit}[3]{%

```

When the count is minus one at the start of the processor, then the unit is begun on its own: initialisation occurs.

```

2862   \ifnum\si@unt@depthcnt=\m@ne\relax
2863     \expandafter\si@unt@init
2864   \fi
2865   \advance\si@unt@depthcnt\@ne\relax
2866   \si@log@debug{Unit processing: level \the\si@unt@depthcnt,
2867     \MessageBreak unit \string\#2}%
2868   \si@unt@firstorsecond{\#1}{\#2}%

```

The core of the `\si@unt@unit` macro is testing if the symbol for the unit is a literal value or another macro. Depending on the result, the symbol is either used as a literal or executed.

```

2869   \si@unt@ifliteral{\#3}
2870     {\si@unt@addtostack{unit}{\#3}%
2871     \ifsi@unt@prepower
2872       \expandafter\si@unt@stkpower
2873     \fi}
2874   {\#3}%

```

The counter is now stepped down, before checking if this is the end of a compound unit.

```

2875   \advance\si@unt@depthcnt\m@ne\relax
2876   \ifnum\si@unt@depthcnt=\z@\relax
2877     \expandafter\si@unt@final
2878   \fi}

```

```

\si@unt@firstorsecond #1 : number
                      #2 : unit-macro

```

At this stage, the flag will be set for the first item to be processed whichever route the unit has been called by.

```

2879 \newcommand*{\si@unt@firstorsecond}[2]{%
2880   \ifsi@unt@first
2881     \expandafter\si@unt@first
2882   \else
2883     \expandafter\si@unt@second
2884   \fi
2885   {\#1}{\#2}}%

```

```

\si@unt@first #1 : number
              #2 : unit-macro

```

For the first unit in the input, some extra tasks are needed. First, the optional argument for the unit macro needs to be tested.

```
2886 \newcommand*{\si@unt@first}[2]{%
2887   \si@ifnotmtarg{#1}
2888     {\num{#1}%
2889     \si@unt@numtrue}%
2890   \si@unt@unithook
```

To avoid filling up the macro list with useless values, the ϵ -TeX primitive `\ifcsname` is employed here (it also avoids complex expansion issues). If some options exist, they are set.

```
2891   \ifcsname\expandafter\@gobble\string#2\opt@unt@si\endcsname
2892     \expandafter\si@unt@setopts
2893   \else
2894     \expandafter\@gobble
2895   \fi
2896   {#2}%
2897   \si@unt@addvaluesep
2898   \si@unt@firstfalse}
```

`\si@unt@setopts` #1 : unit

`\si@unt@setSIopts` A rather long set of `\expandafter` commands to get the options to set safely.

```
2899 \newcommand*{\si@unt@setopts}[1]{%
2900   \expandafter\expandafter\expandafter\expandafter\expandafter
2901     \expandafter\expandafter\si@temptoks\expandafter
2902     \expandafter\expandafter\expandafter\expandafter
2903     \expandafter\expandafter{\expandafter%
2904       \csname\expandafter\@gobble\string#1\opt@unt@si%
2905       \endcsname}%
2906   \expandafter\sisetup\expandafter{\the\si@temptoks}%
2907   \si@log@debug{Applying options '\the\si@temptoks'}
2908   for\MessageBreak unit \string#1}%
```

The user options are reloaded, if defined, to ensure that they still work as expected.

```
2909   \@ifundefined{si@unt@SIopts}{}
2910     {\ifx\@empty\si@unt@SIopts\@empty\else
2911       \expandafter\expandafter\si@unt@setSIopts
2912     \fi}}
2913 \newcommand*{\si@unt@setSIopts}{}%
2914   \expandafter\si@temptoks\expandafter{\si@unt@SIopts}%
2915   \expandafter\sisetup\expandafter{\the\si@temptoks}}
```

`\si@unt@second` #1 : number

`\si@unt@third` #2 : unit

For everything apart from the first item to be processed, spacing may need to be added to separated different units. The macro is divided into two, so that everything except the space can be added in finalisation.

```
2916 \newcommand*{\si@unt@second}[2]{%
2917   \si@ifnotmtarg{#1}
2918     {\si@log@warn{Optional argument to unit macro\MessageBreak
2919       allowed only for outer unit}}%
2920   \si@unt@third
2921   \si@unt@addtostack{space}{\ensuremath{\si@unitsep}}}
```

```

2922 \newcommand*{\si@unt@third}{%
2923   \ifsi@unt@prepower\else
2924     \expandafter\si@unt@stkpower
2925   \fi

```

A check is made to avoid adding -1 to prefixes. If `frac` is active, then the `b` stack will be in use, otherwise it will be `a`.

```

2926 \renewcommand*{\si@tempa}{prefix}%
2927 \expandafter\ifx
2928   \csname si@unt@lastadd\si@unt@checkstack\endcsname\si@tempa
2929 \else
2930   \expandafter\si@unt@spacecheck
2931 \fi
2932 \ifsi@unt@per
2933   \expandafter\si@unt@perseenttrue
2934 \fi}

```

`\si@unt@spacecheck` A check to prevent adding -1 at the very beginning of the unit, where there is a space on the stack.

```

2935 \newcommand*{\si@unt@spacecheck}{%
2936 \renewcommand*{\si@tempa}{space}%
2937 \expandafter\ifx
2938   \csname si@unt@lastadd\si@unt@checkstack\endcsname\si@tempa
2939 \else
2940   \expandafter\si@unt@reciptest
2941 \fi}

```

`\si@unt@prefix` #1 : multiple
 #2 : power-ten
 #3 : symbol

Actual output of the prefixes.

```

2942 \newcommand*{\si@unt@prefix}[3]{%
2943   \si@unt@firstorsecond{}\{#1}%
2944   \ifsi@prefixsymbolic
2945     \expandafter\si@unt@addprefix
2946   \else
2947     \expandafter\si@unt@countprefix
2948   \fi
2949   {#1}{#2}{#3}}

```

`\si@unt@addprefix` #1 : gobbled
 #2 : gobbled
 #3 : symbol

To add the prefix, a little translation is needed.

```

2950 \newcommand*{\si@unt@addprefix}[3]{%
2951   \si@unt@addtostack{prefix}{#3}}

```

`\si@unt@prefixcnt` A storage area is created.

```

2952 \newcount\si@unt@prefixcnt

```

`\si@unt@countprefix` #1 : multiple
`\si@unt@invprefix` #2 : powers-ten
 #3 : gobbled

To count the prefix numeral, the symbol is thrown away. First, a check is made for binary units.

```

2953 \newcommand*{\si@unt@countprefix}[3]{%
2954   \renewcommand*{\si@tempa}{binary}%
2955   \expandafter\expandafter\expandafter\ifx\expandafter
2956     \csname\expandafter\@gobble\string#1\opt@si\endcsname
2957     \si@tempa
2958     \expandafter\sisetup
2959   \else
2960     \expandafter\@gobble
2961   \fi
2962   {prefixbase=two}%
2963   \si@tempcnta#2\relax
2964   \ifsi@unt@per
2965     \expandafter\si@unt@invprefix
2966   \fi
2967   \advance\si@unt@prefixcnt\si@tempcnta\relax}
2968 \newcommand*{\si@unt@invprefix}{%
2969   \si@tempcntb\si@tempcnta\relax
2970   \si@tempcnta -\si@tempcntb\relax}

```

```

\si@unt@litpower #1 : gobbled
                  #2 : number

```

For literal power output, the number can't simply be dumped, so a macro is needed.

```

2971 \newcommand*{\si@unt@litpower}[2]{\textsuperscript{#2}}

```

\ifsi@unt@prepower If a power is seen before a unit, tracking is needed.

```

2972 \newif\ifsi@unt@prepower

```

```

\si@unt@power #1 : power
               #2 : number

```

The handling of powers starts by ensuring that “pre” powers follow \per cleanly. Then a check is needed for inversion.

```

2973 \newcommand*{\si@unt@power}[2]{%
2974   \renewcommand*{\si@tempa}{post}%
2975   \expandafter\expandafter\expandafter\ifx\expandafter
2976     \csname\expandafter\@gobble\string#1\opt@si\endcsname
2977     \si@tempa
2978     \expandafter\@gobbletwo
2979   \else
2980     \expandafter\si@unt@firstorsecond
2981   \fi
2982   {}{\power}%
2983   \si@unt@powerdim #2 pt\relax
2984   \ifsi@frac\else
2985     \ifsi@unt@per
2986       \expandafter\expandafter\expandafter\si@unt@invpower
2987     \fi
2988   \fi
2989   \renewcommand*{\si@tempa}{post}%
2990   \si@unt@prepowertrue
2991   \expandafter\expandafter\expandafter\ifx\expandafter

```

```

2992 \csname\expandafter\@gobble\string#1\opt@si\endcsname
2993 \si@tempa
2994 \expandafter\si@unt@stackpower
2995 \else
2996 \si@log@debug{Power \strip@pt\si@unt@powerdim\space saved
2997 to be added after\MessageBreak next unit}%
2998 \fi}

```

`\si@unt@powerdim` To do sign-inversion on the power, a dimension is used (this allows non-integer values to be handled).

```
2999 \newdimen\si@unt@powerdim
```

`\si@unt@stackpower` Adding powers to the stack should also clear the power list. If the number is already zero, then of course nothing happens.

```

\si@unt@stkpwrr3000 \newcommand*{\si@unt@stackpower}{%
3001 \si@unt@prepowerfalse

```

A trap is used for -1 added to the denominator of a fraction.

```

3002 \si@unt@stkpower
3003 \ifsi@stickyper\else
3004 \si@unt@perfalse
3005 \si@unt@perseenfalse
3006 \fi}

```

The `\si@unt@stkpower` macro needs to be called from a few places, so is spun out from the above.

```

3007 \newcommand*{\si@unt@stkpower}{%
3008 \ifdim\si@unt@powerdim=\m@ne pt\relax
3009 \ifsi@frac\else
3010 \expandafter\expandafter\expandafter\si@unt@stkpwrr
3011 \fi
3012 \else
3013 \expandafter\si@unt@stkpwrr
3014 \fi}

```

Finally, the actual adding (set up to avoid problems with the `\if` above).

```

3015 \newcommand*{\si@unt@stkpwrr}{%
3016 \ifdim\si@unt@powerdim=\z@ \relax\else
3017 \renewcommand*{\si@tempa}{unit}%
3018 \expandafter\ifx
3019 \csname si@unt@lastadd\si@unt@checkstack\endcsname
3020 \si@tempa
3021 \si@log@debug{Adding power
3022 \strip@pt\si@unt@powerdim\space to output stack}%
3023 \si@unt@addtostack{power}{^\num[fixdp=false]{%
3024 \strip@pt\si@unt@powerdim}}}%
3025 \fi
3026 \fi
3027 \si@unt@powerdim\z@\relax}

```

`\si@unt@invpower` A macro to change the sign of the current power.

```

3028 \newcommand*{\si@unt@invpower}{%
3029 \si@tempdima\si@unt@powerdim\relax
3030 \si@unt@powerdim -\si@tempdima\relax

```

The power might end up as “1”, which is not wanted. So it is chunked away.

```
3031 \ifdim\si@unt@powerdim=\p@\relax
3032 \si@unt@powerdim\z@\relax
3033 \fi}
```

\ifsi@unt@per The \per macro sets the correct flags; almost everything else is done elsewhere.
\ifsi@unt@perseen There is always the case of two \per instructions; so the flag is flipped rather
\per than set arbitrarily. The second flag is needed so that \per can give powers of
\si@per -1 properly.

```
\si@unt@per3034 \newif\ifsi@unt@per
3035 \newif\ifsi@unt@perseen
3036 \si@newrobustcmd*{\si@per}{%
3037 \ifsi@unt@littest\else
3038 \ifsi@unt@litout
3039 \expandafter\expandafter\expandafter /%
3040 \else
3041 \ifsi@unt@litprefix
3042 \expandafter\expandafter\expandafter\expandafter
3043 \expandafter\expandafter\expandafter /%
3044 \else
3045 \expandafter\expandafter\expandafter\expandafter
3046 \expandafter\expandafter\expandafter\si@unt@per
3047 \fi
3048 \fi
3049 \fi}
3050 \newcommand*{\si@unt@per}{%
3051 \si@unt@firstorsecond}{\per}%
3052 \ifsi@unt@per
3053 \ifsi@stickyper\else
3054 \expandafter\expandafter\expandafter\si@unt@perfalse
3055 \fi
3056 \else
3057 \expandafter\si@unt@pertrue
3058 \fi}
3059 \let\per\si@per
```

\si@unt@reciptest A test is needed for adding -1 when needed. The second macro is fired only if
\si@unt@recip the power should be reciprocal.

```
3060 \newcommand*{\si@unt@reciptest}{%
3061 \ifsi@unt@per
3062 \ifsi@unt@perseen
3063 \expandafter\expandafter\expandafter\si@unt@recip
3064 \fi
3065 \fi}
3066 \newcommand*{\si@unt@recip}{%
3067 \si@unt@powerdim\m@ne pt\relax
3068 \si@unt@stackpower}
```

\si@unt@lastadda Items cannot be added directly to the stacks (except the spacing stack, a) as the
\si@unt@lastaddb fractional handling may need to add the item to either storage area. First, a track
is kept of what has been added at each stage.

```
3069 \newcommand*{\si@unt@lastadda}{%
3070 \newcommand*{\si@unt@lastaddb}{%}
```

```
\si@unt@addtostack #1 : type
#2 : token
```

By indicating the type of data to be added to the stack, problems can be avoided.

```
3071 \newcommand*\si@unt@addtostack}[2]{%
3072 \renewcommand*\si@tempa}{#1}%
```

Two consecutive items cannot be of the same type; there must be spaces between units, units between prefixes, *etc.*

```
3073 \expandafter\ifx
3074 \csname si@unt@lastadd\si@unt@checkstack\endcsname\si@tempa
3075 \expandafter\@gobbletwo
3076 \else
3077 \expandafter\si@unt@preplussp
3078 \fi
3079 {#1}{#2}}
```

```
\si@unt@preplussp #1 : type
#2 : stack
#3 : token
#4 : gobbled
```

The space added after a prefix needs to be ignored.

```
3080 \newcommand*\si@unt@preplussp[2]{%
3081 \renewcommand*\si@tempa}{prefix+space}%
3082 \edef\si@tempb{%
3083 \csname si@unt@lastadd\si@unt@checkstack\endcsname+#1}%
3084 \ifx\si@tempa\si@tempb
3085 \expandafter\@gobbletwo
3086 \else
3087 \expandafter\si@unt@stack
3088 \fi
3089 {#1}{#2}}
```

```
\si@unt@stack #1 : type
#2 : token
```

The macro for actually doing the stacking up.

```
3090 \newcommand*\si@unt@stack[2]{%
3091 \expandafter\renewcommand\expandafter*\expandafter{%
3092 \csname si@unt@lastadd\si@unt@checkstack\endcsname}{#1}%
```

A count is kept of the number of *units* added to each stack.

```
3093 \renewcommand*\si@tempa}{#1}%
3094 \renewcommand*\si@tempb}{unit}%
3095 \ifx\si@tempa\si@tempb
3096 \expandafter\si@unt@incnt
3097 \fi
```

If a space is added, it is actually held until the next add.

```
3098 \renewcommand*\si@tempb}{space}%
3099 \ifx\si@tempa\si@tempb
3100 \expandafter\si@unt@holdspace
3101 \else
3102 \expandafter\si@unt@addstack
3103 \fi
3104 {#2}}
```

`\si@unt@incnt` The appropriate counter is added to.

```

3105 \newcommand*{\si@unt@incnt}{%
3106   \expandafter\advance
3107     \csname si@unt@unitcnt\si@unt@checkstack\endcsname
3108     \@ne\relax}

```

`\si@unt@holdstacka` The stacked material needs somewhere to live.

```

\si@unt@holdstackb3109 \newcommand*{\si@unt@holdstacka}{}
3110 \newcommand*{\si@unt@holdstackb}{}

```

`\si@unt@holdspace` #1 : token

`\si@unt@addstack` Depending on the nature of the addition, it is either held or added to the stack. For the “b” space stack, a check is made to ensure that a space cannot be added before the first item.

```

3111 \newcommand*{\si@unt@holdspace}[1]{%
3112   \renewcommand*{\si@tempa}{b}%
3113   \edef\si@tempb{\si@unt@checkstack}%
3114   \ifx\si@tempa\si@tempb
3115     \ifx\@empty\si@unt@stackb\@empty
3116     \else
3117       \expandafter\protected@edef
3118       \csname si@unt@holdstack\si@unt@checkstack\endcsname{#1}%
3119     \fi
3120   \else
3121     \expandafter\protected@edef
3122     \csname si@unt@holdstack\si@unt@checkstack\endcsname{#1}%
3123   \fi}
3124 \newcommand*{\si@unt@addstack}[1]{%
3125   \expandafter\protected@edef
3126     \csname si@unt@stack\si@unt@checkstack\endcsname
3127     {\csname si@unt@stack\si@unt@checkstack\endcsname
3128      \csname si@unt@holdstack\si@unt@checkstack\endcsname#1}%
3129   \expandafter\renewcommand\expandafter*\expandafter{%
3130     \csname si@unt@holdstack\si@unt@checkstack\endcsname}{}}

```

`\si@unt@stackout` The stack contents are actually typeset here. First the spacing between units and values is added.

```

3131 \newcommand*{\si@unt@stackout}{%
3132   \si@unt@litouttrue
3133   \ifsi@frac
3134     \expandafter\si@unt@fracout
3135   \else
3136     \expandafter\si@unt@normout
3137   \fi}

```

`\si@unt@checkstack` Which stack is in use needs to be tested.

```

3138 \newcommand*{\si@unt@checkstack}{%
3139   \ifsi@frac
3140     \ifsi@unt@per
3141       \expandafter\expandafter\expandafter b%
3142     \else
3143       \expandafter\expandafter\expandafter a%
3144     \fi

```



```

3145 \else
3146   \expandafter a%
3147 \fi}

```

`\si@unt@spaceout` The space before a unit might not be needed, so it crops up a few times in the output routine.

```

3148 \newcommand*{\si@unt@spaceout}{%
3149   \ensuremath{\si@unt@spstack}}

```

`\si@unt@prefixout` If the prefix counter is not zero, then there is something to typeset.

```

3150 \newcommand*{\si@unt@prefixout}{%
3151   \ifnum\si@unt@prefixcnt=\z@\relax\else
3152     \ifsi@unt@num
3153       \si@out{\ensuremath{\{\}\si@prefixproduct{\}}}%
3154       \fi
3155       \si@unt@stackvalsep
3156       \let\si@expbase\si@prefixbase
3157       \num[fixdp=false]{e\the\si@unt@prefixcnt}%
3158     \fi}

```

`\si@unt@normout` The normal output mode is set up here; nothing much needs to be done as there is no need for complex checks.

```

3159 \newcommand*{\si@unt@normout}{%
3160   \si@unt@prefixout
3161   \si@unt@spaceout
3162   \expandafter\si@unt@out\expandafter{\si@unt@stacka}}

```

`\si@unt@fracout` For fractions, some checks are needed.

```

3163 \newcommand*{\si@unt@fracout}{%
3164   \si@unt@notambig
3165   \ifx\@empty\si@unt@stacka\@empty
3166     \ifx\@empty\si@unt@stackb\@empty
3167       \ifsi@unt@litout\else
3168         \si@log@err{Empty fractional unit}{The unit
3169           argument\MessageBreak given does not contain any
3170           symbols}%
3171       \fi
3172     \else

```

With an empty numerator, no space is added

```

3173       \ifsi@slash
3174         \si@unt@prefixout
3175         \si@frac{\}\si@unt@stackb}%
3176       \else
3177         \si@unt@prefixout
3178         \si@unt@spaceout
3179         \si@frac{1}\si@unt@stackb}%
3180       \fi
3181     \fi
3182   \else

```

If the denominator is empty, then the usual output system can be used.

```

3183     \ifx\@empty\si@unt@stackb\@empty
3184       \si@unt@normout

```

```

3185     \else
3186         \si@unt@prefixout
3187         \si@unt@spaceout
3188         \si@frac{\si@unt@stacka}{\si@unt@stackb}%
3189     \fi
3190 \fi}

```

`\si@unt@notambig` A trap is set for adding brackets to units using a slash, when more than one unit is in the denominator.

```

3191 \newcommand*{\si@unt@notambig}{%
3192     \ifnum\si@unt@unitcntb>\@ne\relax
3193         \ifsi@slash
3194             \ifsi@trapambigfrac
3195                 \expandafter\expandafter\expandafter\expandafter
3196                 \expandafter\expandafter\expandafter\si@unt@notabg
3197             \fi
3198         \fi
3199 \fi}
3200 \newcommand*{\si@unt@notabg}{%
3201     \protected@edef\si@unt@stackb{\si@openfrac\si@unt@stackb
3202         \si@closefrac}}

```

`\si@unt@out` #1 : unit

The final part of the units system is the output routine. This has to cope with units not only as macros but also as direct input (S_Istyle-type input). Non-Latin characters are also handled cleanly. The `\scantokens` system deals with everything except full stops; these are left out so that a single level system can be used *via* a token register.

```

3203 \begingroup
3204 \catcode`\~=\active
3205 \catcode`\.=\active
3206 \gdef\si@unt@out#1{%
3207     \si@temptoks{#1}%
3208     \si@unt@fullstop
3209     \def.\{\ensuremath{\si@unitsep}}%
3210     \def~{\ensuremath{\si@unitspace}}%
3211     \expandafter\protected@edef\expandafter\si@tempa
3212     \expandafter{\the\si@temptoks}%
3213     \begingroup
3214         \si@unt@nonlatin
3215         \makeatletter
3216         \endlinechar\m@ne
3217         \expandafter\si@out\expandafter{%
3218             \expandafter\scantokens\expandafter{\si@tempa}}%
3219     \endgroup
3220 \endgroup

```

`\si@unt@fullstop` Two macros modified from `kvsetkeys` to deal with a single level of active full stops only.

```

\si@unt@stp
3221 \begingroup
3222 \catcode`\.=\active
3223 \catcode`\&=12\relax
3224 \begingroup

```

```

3225 \lccode`\.=`\.\relax
3226 \lccode`\&=`\.\relax
3227 \lowercase{\endgroup
3228 \gdef\si@unt@fullstop{%
3229 \si@temptoks\expandafter{\expandafter}\expandafter
3230 \si@unt@stp\the\si@temptoks&\@nil}
3231 \gdef\si@unt@stp#1&#2\@nil{%
3232 \edef\si@tempa{\the\si@temptoks}%
3233 \ifx\si@tempa\empty
3234 \expandafter\@firstoftwo
3235 \else
3236 \expandafter\@secondoftwo
3237 \fi
3238 {\si@temptoks{#1}}
3239 {\si@temptoks\expandafter{\the\si@temptoks.#1}}%
3240 \si@ifmtarg{#2}
3241 {}
3242 {\si@unt@stp#2\@nil}}
3243 \endgroup

```

`\si@unt@nonlatin` To handle non-Latin symbols in the input, a single macro is provided. If XeTeX is in use, this can be detected immediately.

```

3244 \newcommand*{\si@unt@nonlatin}{}
3245 \ifdefined\XeTeXrevision
3246 \renewcommand*{\si@unt@nonlatin}{%
3247 \catcode176=\active
3248 \catcode181=\active
3249 \catcode197=\active
3250 \si@unt@sym{176}{\si@sym@degree}%
3251 \si@unt@sym{181}{\si@sym@mu}%
3252 \si@unt@sym{197}{\si@sym@ringA}}%
3253 \fi

```

If `inputenc` has been loaded, then a check is made that the encoding is correct. If all is well, the non-Latin symbols are handled. The degree symbol is character 176, the micro symbol is character 181 and ring capital A is character 197 in `latin1`.

```

3254 \AtBeginDocument{
3255 \@ifpackageloaded{inputenc}
3256 {\@for\si@tempa:=latin1,latin5,latin9\do{
3257 \ifx\inputencodingname\si@tempa
3258 \renewcommand*{\si@unt@nonlatin}{%
3259 \catcode176=\active
3260 \catcode181=\active
3261 \catcode197=\active
3262 \si@unt@sym{176}{\si@sym@degree}%
3263 \si@unt@sym{181}{\si@sym@mu}%
3264 \si@unt@sym{197}{\si@sym@ringA}}%
3265 \fi}}
3266 {}}

```

`\si@unt@sym` #1 : charcode

A macro for declaring symbols: a copy of `\DeclareInputMath` from `inputenc`.

```

3267 \newcommand*{\si@unt@sym}[1]{%
3268 \bgroup

```

```

3269      \uccode'\~#1%
3270      \uppercase{%
3271        \egroup
3272        \def~}}

```

`\kilogram` With the system set up, the basic unit macros are implemented. The only units defined whatever options are given are the base SI units.

```

\metre
\meter3273 \newunit{\kilogram}{kg}
\mole3274 \newunit{\metre}{m}
\kelvin3275 \newunit{\meter}{\metre}
\candela3276 \newunit{\mole}{mol}
\second3277 \newunit{\second}{s}
\ampere3278 \newunit{\ampere}{A}
3279 \newunit{\kelvin}{K}
3280 \newunit{\candela}{cd}

```

`\Square` Unlike multiples (which can be skipped if needed), the basic powers are also always defined.

```

\ssquare
\squared3281 \newpower{\Square}{2}
\cubic3282 \newpower{\ssquare}{2}
\cubed3283 \newpower[post]{\squared}{2}
3284 \newpower{\cubic}{3}
3285 \newpower[post]{\cubed}{3}

```

`\tothe` The user macros for arbitrary powers are simple calling macros to a common internal macro.

```

\raiseto
3286 \newcommand*{\tothe}{\si@tothe{\tothe}}
3287 \newcommand*{\raiseto}{\si@tothe{\raiseto}}

```

`\si@tothe` #1 : either `\tothe` or `\raiseto`

`\tothe@opt@si` #2 : number

`\raiseto@opt@si` A macro for arbitrary powers, which comes after the unit and so needs to be marked as such. Two fake option-storage macros are created so that everything works correctly.

```

3288 \newcommand*{\si@tothe}[2]{%
3289   \ifsi@unt@littest
3290     \expandafter\@gobbletwo
3291   \else
3292     \ifsi@unt@litout
3293       \expandafter\expandafter\expandafter\si@unt@litpower
3294     \else
3295       \expandafter\expandafter\expandafter\si@unt@power
3296     \fi
3297   \fi
3298   {#1}{#2}}
3299 \newcommand*{\tothe@opt@si}{post}
3300 \newcommand*{\raiseto@opt@si}{}

```

21.14 Locales

`\si@loc@load` #1 : locale

`\si@loc@ssetup`

When loading a locale, the setup is saved rather than applied. Anything other than simple settings should be inside `\addtolocale`, which is already defined.

```

3301 \newcommand*{\si@loc@load}[1]{%
3302   \let\si@loc@ssetup\sisetup
3303   \renewcommand*{\sisetup}[1]{%
3304     \expandafter\gdef\csname si@loc@#1\endcsname{##1}}%
3305   \si@loadfile{#1}%
3306   \let\sisetup\si@loc@ssetup}

```

`\si@loc@set` #1 : locale

Setting the locale transfers the settings to `\sisetup`, and runs any extra macros.

```

3307 \newcommand*{\si@loc@set}[1]{%
3308   \ifcsname si@loc@#1\endcsname
3309     \si@log@inf{Setting locale to '#1'}%
3310     \expandafter\expandafter\expandafter\expandafter
3311       \expandafter\expandafter\expandafter\si@temptoks
3312       \expandafter\expandafter\expandafter\expandafter
3313       \expandafter\expandafter\expandafter{%
3314         \expandafter\csname si@loc@#1\endcsname}%
3315     \expandafter\sisetup\expandafter{\the\si@temptoks}%
3316     \ifcsname si@loc@#1@extra\endcsname
3317       \csname si@loc@#1@extra\endcsname
3318     \fi
3319   \else
3320     \ifcsname si@loc@#1@extra\endcsname
3321       \si@log@inf{Setting locale to '#1'}%
3322       \csname si@loc@#1@extra\endcsname
3323     \else
3324       \si@log@warn{Unknown locale '#1'}%
3325     \fi
3326   \fi}

```

`\si@loc@ltol` #1 : list of locales and languages

The necessary loading for language modules occurs.

```

3327 \newcommand*{\si@loc@ltol}[1]{%
3328   \def\si@tempa##1:##2\@nil{\si@loc@load{##1}}
3329   \@for\si@tempb:=#1\do{%
3330     \expandafter\si@tempa\si@tempb:\@nil}
3331   \AtBeginDocument{
3332     \ifpackageloaded{babel}
3333       {\def\si@tempa##1:##2:##3\@nil{%
3334         \expandafter\addto\expandafter{%
3335           \csname extras##2\endcsname}%
3336         {\si@loc@set{##1}}}%
3337       \@for\si@tempb:=#1\do{%
3338         \expandafter\si@tempa\si@tempb:\@nil}%
3339       \expandafter\selectlanguage\expandafter{\language}
3340       {\si@log@warn{babel not loaded \MessageBreak
3341         loctolang option ignored}}}%
3342   \AtBeginDocument{
3343     \ifpackageloaded{babel}
3344       {\renewcommand*{\si@loc@ltol}[1]{%
3345         \def\si@tempa##1:##2\@nil{\si@loc@load{##1}}%

```

```

3346      \@for\si@tempb:=#1\do{%
3347          \expandafter\si@tempa\si@tempb:\@nil}%
3348      \def\si@tempa##1:##2:##3\@nil{%
3349          \expandafter\addto\expandafter{%
3350              \csname extras##2\endcsname}%
3351              {\si@loc@set{##1}}}%
3352      \@for\si@tempb:=#1\do{%
3353          \expandafter\si@tempa\si@tempb::\@nil}}}
3354      {\renewcommand*\si@loc@ltol}[1]{%
3355          \si@log@warn{babel not loaded \MessageBreak
3356              loctolang option ignored}}}%

```

\addtolocale #1 : locale

#2 : commands

Arbitrary macros may need to be added to the locale.

```

3357 \newcommand*\addtolocale[2]{%
3358     \si@addtocname{si@loc@#1@extra}{#2}}

```

21.15 Output routine

\si@out #1 : text

With all of the setup done, the text can finally be typeset. This is done inside a `\text` block, which takes care of `\ensuremath`, *etc.* First of all, the various catcode settings needed to get maths-in-text mode are made. `\makeatletter` is needed so that `\scantokens` still allows internal macros to work.

```

3359 \begingroup
3360     \catcode'\^=\active
3361     \catcode'\-=\active
3362     \gdef\si@out#1{%
3363         \begingroup
3364             \catcode'\^=\active
3365             \makeatletter
3366             \endlinechar\m@ne

```

The various font families can now be set up. First a check is made in case there are nested calls to `\si@out@text`.

```

3367     \ifsi@fam@set\else
3368         \expandafter\si@fam@set
3369         \fi
3370     \text{%
3371         \si@colourcmd{\si@colour}%
3372         \si@fam@italic\si@fam@bold\si@fam@text

```

The correct mode is selected, and the input is handed off for typesetting.

```

3373     \ifsi@textmode
3374         \expandafter\si@out@text
3375     \else
3376         \expandafter\si@out@maths
3377     \fi
3378     {\scantokens{#1}}}%
3379 \endgroup
3380 \check@mathfonts}

```

```

\si@out@text #1 : text
\si@out@maths Output occurs with the correct changes to superscript behaviour.
3381 \gdef\si@out@text#1{%
3382     \let^\si@out@sp
3383     \let\textsuperscript\si@out@sp
3384     \catcode`\-=\active\relax
3385     \let-\si@out@minus
3386     #1}
3387 \gdef\si@out@maths#1{%
3388     \let^\sp
3389     \let\textsuperscript\sp
3390     $\si@fam@maths{#1}$}
3391 \endgroup

\si@out@sp #1 : text
\textsuperscript gives slightly different alignment of numbers to using ^
in text mode. To avoid this, a slightly different definition is used. Elsewhere
\textsuperscript is used, as the code above sorts out the text/math mode
issues.
3392 \newcommand*{\si@out@sp}[1]{\ensuremath{^{\text{#1}}}}

\si@out@minus An active minus sign is needed.
3393 \newcommand*{\si@out@minus}{\ensuremath{-}}

\ifsi@out@num A flag is needed to control output settings. This will be false unless inside
\si@out@num.
3394 \newif\ifsi@out@num

\si@out@num #1 : number
For numerical output, the default fonts are controlled slightly differently to text
output.
3395 \newcommand*{\si@out@num}[1]{%
3396     \begingroup
3397     \si@out@numtrue
3398     \si@out{#1}%
3399     \endgroup}

```

21.16 Finalisation

```

\si@extension To keep the code easy to maintain, the reusable filename components are macros
\si@fileprefix rather than literals.
3400 \newcommand*{\si@extension}{cfg}
3401 \newcommand*{\si@fileprefix}{si-}

\si@ifl@aded #1 : package
\si@ifloaded A bit of borrowing from the LATEX kernel. A copy of \@ifl@aded is needed as
things aren't always done in the preamble by siunitx.
3402 \newcommand*{\si@ifl@aded}{}
3403 \let\si@ifl@aded\@ifl@aded
3404 \newcommand*{\si@ifloaded}[1]{%
3405     \si@ifl@aded\si@extension{\si@fileprefix#1}}

```

`\si@loadfile #1 : file`

Loading configuration files is handled here.

```
3406 \newcommand*{\si@loadfile}[1]{%
3407   \si@ifloaded{#1}{}
3408   {\si@InputIfFileExists{\si@fileprefix#1.\si@extension}
3409     {}
3410     {\si@log@err{Failed to load file
3411       \si@fileprefix#1.\si@extension}
3412       {The configuration file requested could not be
3413         found}}}}
```

`\ifsi@outerinput` At various points, the package needs to know if the currently-read configuration file is the outer one.

```
3414 \newif\ifsi@outerinput\si@outerinputtrue
```

`\si@InputIfFileExists` To allow reading configuration files after the start of the document, a private copy of `\InputIfFileExists` is needed. The main macro here simply checks if the modified version is needed.

```
3415 \let\si@InputIfFileExists\InputIfFileExists
3416 \AtBeginDocument{
3417   \renewcommand*{\si@InputIfFileExists}{%
3418     \ifsi@outerinput
3419       \expandafter\si@outerinput
3420     \else
3421       \expandafter\InputIfFileExists
3422     \fi}}
```

`\si@outerinput #1 : file`

`#2 : true-code`

`#3 : false-code`

The outer call to load a file needs to set @ to a letter and turn off printing using `\nullfont`.

```
3423 \newcommand*{\si@outerinput}[3]{%
3424   \makeatletter
3425   \nullfont
3426   \si@outerinputfalse
3427   \IfFileExists{#1}%
3428     {#2\@addtofilelist{#1}\@input \@filef@und\normalsize}
3429     {#3}%
3430   \normalsize
3431   \makeatother}
```

`\requiresiconfigs #1 : config-file`

The configuration files depend on each other.

```
3432 \newcommand*{\requiresiconfigs}[1]{%
3433   \@for\si@tempb:=#1\do{\si@loadfile{\si@tempb}}}
```

`\si@loademfile #1 : file`

For emulation files, an additional check is made.

```
3434 \newcommand*{\si@loademfile}[1]{%
3435   \@ifpackageloaded{#1}
3436   {\si@log@err{Emulation clash for package '#1'}}
```



```

3437     {You have asked for emulation of package
3438     `#1'\MessageBreak
3439     (perhaps by giving siunitx a back-compatibility
3440     option)\MessageBreak but the package is already
3441     loaded!}}
3442     {\si@loadfile{#1}}}}

\si@emclash #1 : package
            #2 : package
            A macro for emulation clashes.

3443 \newcommand*{\si@emclash}[2]{%
3444   \si@log@err{Emulation clash: `#1' and `#2'}
3445   {You have asked for emulation of package `#1'\MessageBreak
3446   but have already loaded emulation of `#2'}}}

\si@emulating #1 : package
              #2 : version
              For packages that are emulated, the LATEX mechanism to prevent re-loading is
              used. The list of packages to check at the start of the document also has to be
              altered.

3447 \newcommand*{\si@emulating}[2]{%
3448   \@namedef{ver@#1.sty}{#2 siunitx emulation of #1}%
3449   \let\si@tempa\si@blockpkgs
3450   \renewcommand*{\si@blockpkgs}{}%
3451   \@for\si@tempb:=\si@tempa\do{%
3452     \renewcommand*{\si@tempa}{#1}%
3453     \ifx\si@tempa\si@tempb\else
3454       \lowercase{\edef\si@tempa{#1}}%
3455       \lowercase{\edef\si@tempc{\si@tempb}}%
3456       \ifx\si@tempa\si@tempc
3457         \@namedef{ver@\si@tempc.sty}{#2 siunitx emulation of
3458         #1}%
3459       \else
3460         \si@addtolist{\si@blockpkgs}{\si@tempb}%
3461       \fi
3462     \fi}%
3463   \let\si@tempa\si@checkpkgs
3464   \renewcommand*{\si@checkpkgs}{}%
3465   \renewcommand*{\si@tempb}{#1}%
3466   \@for\si@tempc:=\si@tempa\do{%
3467     \ifx\si@tempb\si@tempc\else
3468       \si@addtolist{\si@checkpkgs}{\si@tempc}%
3469     \fi}}

With the siunitx kernel macros defined, the package can now run through finalisa-
tion. First, the default key values are set. The user options are then processed.

3470 \sisetup{
3471   addsign=none,
3472   allowzeroexp=false,
3473   angformat=unchanged,
3474   astroang=false,%(
3475   closeerr=),%(
3476   closefrac=),

```

```

3477 colour=black,
3478 colourall=false,
3479 colourneg=false,
3480 decimalsymbol=fullstop,
3481 detectdisplay=true,
3482 digitsep=thin,
3483 dp=3,
3484 eVcorra=0.3ex,
3485 eVcorrb=0ex,
3486 errspace=none,
3487 fixdp=false,
3488 inlinebold=text,
3489 load=default,
3490 mathsrn=mathrm,
3491 mathssf=mathsf,
3492 mathstt=mathtt,
3493 mode=maths,
3494 negcolour=red,
3495 noload={},
3496 numaddn={},%(
3497 numcloseerr=),%
3498 numdecimal={.,},
3499 numdigits=0123456789,
3500 numexp=eEdD,
3501 numgobble={},
3502 numopenerr=(,% )
3503 numprod=x,
3504 numsign=+-\pm\mp,
3505 obeybold=false,
3506 obeyitalic=false,
3507 obeymode=false,
3508 openerr=(,% )
3509 openfrac=(,% )
3510 padangle=small,
3511 padnumber=lead,
3512 per=reciprocal,
3513 prefixbase=ten,
3514 prefixproduct=times,
3515 prefixsymbolic=true,
3516 prespace=false,
3517 redefsymbols=true,
3518 repeatunits=true,
3519 retainplus=false,
3520 seperr=false,
3521 sepfour=false,
3522 sign=plus,
3523 slash=slash,
3524 stickyper=false,
3525 strictarc=true,
3526 tabalignexp=true,
3527 tabautofit=false,
3528 tabformat=3.2,
3529 tabnumalign=centredecimal,
3530 tabtextalign=centre,

```

```

3531 tabunitalign=left,
3532 textrm=rmfamily,
3533 textsf=sffamily,
3534 texttt=ttfamily,
3535 tightpm=false,
3536 trapambigerr=true,
3537 trapambigfrac=true,
3538 unitsep=thin,
3539 valuesep=thin,
3540 xspace=false}
3541 \ProcessOptionsX[si]<key>

```

A check is now made so that emulation takes place one file at a time, and that each file is loaded only once.

```

3542 \ifx\@empty\si@emulate\@empty\else
3543   \@for\si@tempa:=\si@emulate\do{%
3544     \si@loademfile{\si@tempa}}
3545 \fi

```

`\si@expanddefault` For turning the list of default choices into a loadable list.

```

3546 \newcommand*{\si@expanddefault}[2]{%
3547   \expandafter\ifx\expandafter\@empty\csname si@#1\endcsname
3548     \@empty
3549   \else
3550     \renewcommand*{\si@tempb}{default}%
3551     \renewcommand*{\si@tempc}{}%
3552     \expandafter\@for\expandafter\si@tempa\expandafter
3553       :\expandafter=\csname si@#1\endcsname\do{%
3554       \ifx\si@tempa\si@tempb
3555         \si@addtolist{\si@tempc}{#2}%
3556       \else
3557         \si@addtolist{\si@tempc}{\si@tempa}%
3558       \fi}
3559     \expandafter\edef\csname si@#1\endcsname{\si@tempc}%
3560     \expandafter\si@addtolist\expandafter{%
3561       \csname si@no#1\endcsname}%
3562     {default}%
3563     \renewcommand*{\si@tempc}{}%
3564     \expandafter\@for\expandafter\si@tempa\expandafter
3565       :\expandafter=\csname si@#1\endcsname\do{%
3566       \si@switchfalse
3567       \expandafter\@for\expandafter\si@tempb\expandafter
3568         :\expandafter=\csname si@no#1\endcsname\do{%
3569         \ifx\si@tempa\si@tempb
3570           \si@switchtrue
3571         \fi
3572         \ifsi@switch\else
3573           \si@addtolist{\si@tempc}{\si@tempa}%
3574         \fi}}
3575     \@for\si@tempa:=\si@tempc\do{%
3576       \si@loadfile{\si@tempa}}%
3577   \fi}

```

The configuration and abbreviation files are loaded.

```

3578 \si@expanddefault{load}{prefix,named,addn,prefixed,accepted,%
3579   physical,abbr}

```

The very last job is to load a local configuration file, if one exists, and restore catcodes.

```

3580 \IfFileExists{siunitx.cfg}
3581   {\si@log@inf{Local configuration file found}%
3582     \InputIfFileExists{siunitx.cfg}{}{}}
3583 {}
3584 \si@catcodes

```

22 Loadable modules

To keep the package relatively clear, and to make maintenance easier, the only units defined in the package itself are the base units. Everything else is a loadable module (similar to the approach in `unitsdef`).

22.1 Multiple prefixes

`\yocto` The various SI multiple prefixes are defined here. First the small powers.

```

\zepto3585 \ProvidesFile{si-prefix.cfg}
\atto3586   [2008/09/03 v1.01 siunitx: SI Multiple prefixes]
\femto3587 \newprefix{\yocto}{-24}{y}
\pico3588 \newprefix{\zepto}{-21}{z}
\nano3589 \newprefix{\atto}{-18}{a}
\micro3590 \newprefix{\femto}{-15}{f}
\Micro3591 \newprefix{\pico}{-12}{p}
\milli3592 \newprefix{\nano}{-9}{n}
\centi

```

Some testing is needed for `unitsdef` compatibility.

```

\deci3593 \ifsi@old@OHM
3594   \newprefix{\Micro}{-6}{\si@sym@mu}
3595 \else
3596   \ifsi@gensymb\else
3597     \newprefix{\micro}{-6}{\si@sym@mu}
3598   \fi
3599 \fi
3600 \newprefix{\milli}{-3}{m}
3601 \newprefix{\centi}{-2}{c}
3602 \newprefix{\deci}{-1}{d}

```

`\deca` Now the large ones.

```

\hecto3603 \newprefix{\deca}{1}{da}
\kilo3604 \newprefix{\hecto}{2}{h}
\mega3605 \newprefix{\kilo}{3}{k}
\giga3606 \newprefix{\mega}{6}{M}
\tera3607 \newprefix{\giga}{9}{G}
\peta3608 \newprefix{\tera}{12}{T}
\exa3609 \newprefix{\peta}{15}{P}
\zetta3610 \newprefix{\exa}{18}{E}
\yotta3611 \newprefix{\zetta}{21}{Z}
3612 \newprefix{\yotta}{24}{Y}

```

`\deka` Apparently, “deka” is common in the US for deca.

```
3613 \newprefix{\deka}{1}{da}
```

`\gram` As the base unit of mass is the kilogram, rather than the gram, a bit of extra work is needed; by default the package only defines `\kilogram`, but with the prefixes available, this is altered to be `\kilo\gram`. For that, the `\gram` must be defined first.

```
3614 \newunit{\gram}{g}
```

```
3615 \renewunit{\kilogram}{\kilo\gram}
```

22.2 Derived units with specific names

`\becquerel` Derived units with specific names and symbols are defined. Litre is an awkward one, but here the UK standard is used.

```
\farad3616 \ProvidesFile{si-named.cfg}
```

```
\Gray3617 [2008/09/03 v1.01 siunitx: SI Named units]
```

```
\ggray3618 \newunit{\becquerel}{Bq}
```

```
\hertz3619 \newunit{\coulomb}{C}
```

```
\henry3620 \newunit{\farad}{F}
```

```
\joule3621 \newunit{\Gray}{Gy}
```

```
\katal3622 \newunit{\ggray}{Gy}
```

```
\lumen3623 \newunit{\hertz}{Hz}
```

```
\lux3624 \newunit{\henry}{H}
```

```
\lux3625 \newunit{\joule}{J}
```

```
\newton3626 \newunit{\katal}{kat}
```

```
3627 \newunit{\lumen}{lm}
```

```
3628 \newunit{\lux}{lx}
```

```
3629 \newunit{\newton}{N}
```

`\ohm` Some testing is needed for `unitsdef` compatibility.

```
\Ohm3630 \ifsi@old@OHM
```

```
\pascal3631 \newunit{\Ohm}{\si@sym@Omega}
```

```
\siemens3632 \else
```

```
\sievert3633 \ifsi@gensymb\else
```

`\tesla` To be on the safe side, use `\provideunit`.

```
\volt3634 \provideunit{\ohm}{\si@sym@Omega}
```

```
\watt3635 \fi
```

```
\weber3636 \fi
```

```
3637 \newunit{\pascal}{Pa}
```

```
3638 \newunit{\siemens}{S}
```

```
3639 \newunit{\sievert}{Sv}
```

```
3640 \newunit{\tesla}{T}
```

```
3641 \newunit{\volt}{V}
```

```
3642 \newunit{\watt}{W}
```

```
3643 \newunit{\weber}{Wb}
```

`\celsius` The degree celsius is a named unit.

```
\Celsius3644 \ifsi@old@OHM
```

```
3645 \newunit{\Celsius}{\si@sym@celsius}
```

```
3646 \else
```

```
3647 \ifsi@gensymb\else
```

```
3648 \newunit{\celsius}{\si@sym@celsius}
```

```

3649 \fi
3650 \fi

```

```

\radian The radian and steradian are officially derived units.
\steradian3651 \newunit{\radian}{rad}
3652 \newunit{\steradian}{sr}

```

22.3 Units with prefixes

As in `unitsdef`, some commonly used prefixed units are set up. This requires `si-prefix.cfg` and `si-named.cfg`.

```

3653 \ProvidesFile{si-prefixed.cfg}
3654 [2008/09/03 v1.01 siunitx: SI Prefixed units]
3655 \requiresiconfigs{prefix,named,accepted,physical}

```

`\picometre` Extra distances.

```

\nanometre3656 \newunit{\picometre}{\pico\metre}
\micrometre3657 \newunit{\nanometre}{\nano\metre}
\millimetre3658 \newunit{\micrometre}{\micro\metre}
\centimetre3659 \newunit{\millimetre}{\milli\metre}
\decimetre3660 \newunit{\centimetre}{\centi\metre}
\kilometre3661 \newunit{\decimetre}{\deci\metre}
3662 \newunit{\kilometre}{\kilo\metre}

```

`\femtogram` Extra masses.

```

\picogram3663 \newunit{\femtogram}{\femto\gram}
\nanogram3664 \newunit{\picogram}{\pico\gram}
\microgram3665 \newunit{\nanogram}{\nano\gram}
\milligram3666 \newunit{\microgram}{\micro\gram}
3667 \newunit{\milligram}{\milli\gram}

```

`\femtomole` Now some moles.

```

\picomole3668 \newunit{\femtomole}{\femto\mole}
\nanomole3669 \newunit{\picomole}{\pico\mole}
\micromole3670 \newunit{\nanomole}{\nano\mole}
\millimole3671 \newunit{\micromole}{\micro\mole}
3672 \newunit{\millimole}{\milli\mole}

```

`\attosecond` Prefixed seconds.

```

\femtosecond3673 \newunit{\attosecond}{\atto\second}
\picosecond3674 \newunit{\femtosecond}{\femto\second}
\nanosecond3675 \newunit{\picosecond}{\pico\second}
\microsecond3676 \newunit{\nanosecond}{\nano\second}
\millisecond3677 \newunit{\microsecond}{\micro\second}
3678 \newunit{\millisecond}{\milli\second}

```

`\picoampere` The last prefixed base units are amperes.

```

\nanoampere3679 \newunit{\picoampere}{\pico\ampere}
\microampere3680 \newunit{\nanoampere}{\nano\ampere}
\milliampere3681 \newunit{\microampere}{\micro\ampere}
\kiloampere3682 \newunit{\milliampere}{\milli\ampere}
3683 \newunit{\kiloampere}{\kilo\ampere}

```

```

\millivolt    More electricity-related units.
\kilovolt3684 \newunit{\millivolt}{\milli\volt}
\milliwatt3685 \newunit{\kilovolt}{\kilo\volt}
\kilowatt3686 \newunit{\milliwatt}{\milli\watt}
\megawatt3687 \newunit{\kilowatt}{\kilo\watt}
\femtofarad3688 \newunit{\megawatt}{\mega\watt}
\picofarad3689 \newunit{\femtofarad}{\femto\farad}
\nanofarad3690 \newunit{\picofarad}{\pico\farad}
\microfarad3691 \newunit{\nanofarad}{\nano\farad}
\millifarad3692 \newunit{\microfarad}{\micro\farad}
\millifarad3693 \newunit{\millifarad}{\milli\farad}
\millisiemens3694 \newunit{\millisiemens}{\milli\siemens}

\kiloohm    For resistance, checks are needed again for the definition of \ohm.
\megaohm3695 \ifsi@old@OHM
\gigaohm3696 \newunit{\kiloohm}{\kilo\Ohm}
3697 \newunit{\megaohm}{\mega\Ohm}
3698 \newunit{\gigaohm}{\giga\Ohm}
3699 \else
3700 \ifsi@gensymb\else
3701 \newunit{\kiloohm}{\kilo\ohm}
3702 \newunit{\megaohm}{\mega\ohm}
3703 \newunit{\gigaohm}{\giga\ohm}
3704 \fi
3705 \fi

\microlitre    Volumes (unlike unitsdef, with litre and metre spelled correctly). \millilitre
\millilitre    and \microlitre are defined as they are the two officially-sanctioned prefixes
\cubicmetre    for the litre.
\cubiccentimetre3706 \newunit{\microlitre}{\micro\litre}
\centimetrecubed3707 \newunit{\millilitre}{\milli\litre}
\cubicmicrometre3708 \newunit{\cubicmetre}{\metre\cubed}
\cubicmillimetre3709 \newunit{\cubiccentimetre}{\centi\metre\cubed}
\cubicdecimetre3710 \newunit{\centimetrecubed}{\centi\metre\cubed}
3711 \newunit{\cubicmicrometre}{\micro\metre\cubed}
3712 \newunit{\cubicmillimetre}{\milli\metre\cubed}
3713 \newunit{\cubicdecimetre}{\cubic\deci\metre}

\squaremetre    Areas, with metre spelled corrected; \are and \hectare are in the “temporarily
\squarecentimetre    accepted” file.
\centimetresquared3714 \newunit{\squaremetre}{\Square\metre}
\squarekilometre3715 \newunit{\squarecentimetre}{\Square\centi\metre}
3716 \newunit{\centimetresquared}{\centi\metre\squared}
3717 \newunit{\squarekilometre}{\Square\kilo\metre}

\millijoule    Some energy is needed by now! Notice the definition of \kilowatthour
\kilojoule3718 \newunit{\millijoule}{\milli\joule}
\megajoule3719 \newunit{\kilojoule}{\kilo\joule}
\millielelectronvolt3720 \newunit{\megajoule}{\mega\joule}
\kiloelectronvolt3721 \newunit{\millielelectronvolt}{\milli\electronvolt}
\megaelectronvolt3722 \newunit{\kiloelectronvolt}{\kilo\electronvolt}
\gigaelectronvolt3723 \newunit{\megaelectronvolt}{\mega\electronvolt}
\teraelectronvolt3724 \newunit{\gigaelectronvolt}{\giga\electronvolt}
\kilowatthour

```

```

3725 \newunit{\teraelectronvolt}{\tera\electronvolt}
3726 \newunit[unitsep=none]{\kilowatthour}{\kilo\watt\hour}

\millihertz   Frequencies.
\kilohertz3727 \newunit{\millihertz}{\milli\hertz}
\megahertz3728 \newunit{\kilohertz}{\kilo\hertz}
\gigahertz3729 \newunit{\megahertz}{\mega\hertz}
\terahertz3730 \newunit{\gigahertz}{\giga\hertz}
3731 \newunit{\terahertz}{\tera\hertz}

\millinewton  A few more from various areas.
\kilonewton3732 \newunit{\millinewton}{\milli\newton}
\hectopascal3733 \newunit{\kilonewton}{\kilo\newton}
\megabecquerel3734 \newunit{\hectopascal}{\hecto\pascal}
\millisievert3735 \newunit{\megabecquerel}{\mega\becquerel}
3736 \newunit{\millisievert}{\milli\sievert}

```

22.4 Abbreviated units

```

\pA   The abbreviated units are sorted in one file. To allow back-compatibility with
\nA   unitsdef, each one is inside an \if block for the appropriate option. First currents.
\micA3737 \ProvidesFile{si-abbr.cfg}
\mA3738   [2008/09/03 v1.01 siunitx: Abbreviated units]
\kA3739 \requiresiconfigs{prefix,named,accepted,physical}
3740 \newunit{\pA}{\pico\ampere}
3741 \newunit{\nA}{\nano\ampere}
3742 \newunit{\micA}{\micro\ampere}
3743 \newunit{\mA}{\milli\ampere}
3744 \newunit{\kA}{\kilo\ampere}

\Hz   Then frequencies.
\mHz3745 \newunit{\Hz}{\hertz}
\kHz3746 \newunit{\mHz}{\milli\hertz}
\MHz3747 \newunit{\kHz}{\kilo\hertz}
\GHz3748 \newunit{\MHz}{\mega\hertz}
\THz3749 \newunit{\GHz}{\giga\hertz}
3750 \newunit{\THz}{\tera\hertz}

\fmol  Amounts of substance.
\pmol3751 \newunit{\fmol}{\femto\mole}
\nmol3752 \newunit{\pmol}{\pico\mole}
\micmol3753 \newunit{\nmol}{\nano\mole}
\mmol3754 \newunit{\micmol}{\micro\mole}
3755 \newunit{\mmol}{\milli\mole}

\kV   Potentials.
\mV3756 \newunit{\kV}{\kilo\volt}
3757 \newunit{\mV}{\milli\volt}

\ml   Volumes and areas
\micl3758 \provideunit{\ml}{\milli\litre}
\cmcl3759 \provideunit{\micl}{\micro\litre}
\dmc
\cms

```



```

3760 \newunit{\cmc}{\centi\metre\cubed}
3761 \newunit{\dmc}{\deci\metre\cubed}
3762 \newunit{\cms}{\centi\metre\squared}

\fg  Masses.
\SIfg3763 \newunit{\kg}{\kilo\gram}
\kg  There is a name clash with babel here in French; hopefully there will not be too
\fg  many complaints.
\pg 3764 \AtBeginDocument{\provideunit{\fg}{\femto\gram}}
\nanog3765 \newunit{\SIfg}{\femto\gram}
\micg3766 \newunit{\pg}{\pico\gram}
\mg3767 \newunit{\nanog}{\nano\gram}
\amu3768 \newunit{\micg}{\micro\gram}
3769 \newunit{\mg}{\milli\gram}
3770 \newunit{\amu}{\atomicmass}

\kJ  Energies.
\eV3771 \newunit{\kJ}{\kilo\joule}
\meV3772 \newunit{\eV}{\electronvolt}
\keV3773 \newunit{\meV}{\milli\electronvolt}
\MeV3774 \newunit{\keV}{\kilo\electronvolt}
\GeV3775 \newunit{\MeV}{\mega\electronvolt}
\TeV3776 \newunit{\GeV}{\giga\electronvolt}
\kWh3777 \newunit{\TeV}{\tera\electronvolt}
3778 \newunit[unitsep=none]{\kWh}{\kilo\watt\hour}

\picom Lengths.
\nm3779 \newunit{\picom}{\pico\metre}
\micm3780 \newunit{\nm}{\nano\metre}
\mm3781 \newunit{\micm}{\micro\metre}
\cm3782 \newunit{\mm}{\milli\metre}
\dm3783 \newunit{\cm}{\centi\metre}
\km3784 \newunit{\dm}{\deci\metre}
3785 \newunit{\km}{\kilo\metre}

\Sec  Finally, times.
\as3786 \newunit{\Sec}{\second}
\fs3787 \newunit{\as}{\atto\second}
\ps3788 \newunit{\fs}{\femto\second}
\ns  The letter class (and others) define \ps for postscripts, so \provideunit is best
\mic  here.
\ms3789 \provideunit{\ps}{\pico\second}
3790 \newunit{\ns}{\nano\second}
3791 \newunit{\mics}{\micro\second}
3792 \newunit{\ms}{\milli\second}

```

22.5 Additional (temporary) SI units

\angstrom Some units are “temporarily” acceptable for use in the SI system. These are defined here, although some are in very general use.

```

\hectare3793 \ProvidesFile{si-addn.cfg}
\barn
\BAR
\bbar
\millibar
\gal
\curie
\roentgen
\rad
\rem

```

```

3794 [2008/09/03 v1.01 siunitx: SI Additional units]
3795 \newunit{\angstrom}{\si@sym@ringA}
3796 \newunit{\are}{a}
3797 \newunit{\hectare}{\hecto\are}
3798 \newunit{\barn}{b}
3799 \newunit{\BAR}{bar}
3800 \newunit{\bbar}{bar}
3801 \newunit{\millibar}{\milli\BAR}
3802 \newunit{\gal}{Gal}
3803 \newunit{\curie}{Ci}
3804 \newunit{\roentgen}{R}
3805 \newunit{\rad}{rad}
3806 \newunit{\rem}{rem}

```

22.6 Units accepted for use with SI

The units which are accepted but do not fit in the above, plus \percent which seems to fit into this category.

```

\minute
\hour3807 \ProvidesFile{si-accepted.cfg}
\Day3808 [2008/09/03 v1.01 siunitx: SI Accepted units]
\dday3809 \newunit{\minute}{min}
\degree3810 \newunit{\hour}{h}
\Degree3811 \newunit{\Day}{d}
\arcmin3812 \newunit{\dday}{d}
\arcsec3813 \ifsi@old@OHM
3814 \newunit[valuesep=none]{\Degree}{\si@sym@degree}
\litre3815 \else
\liter3816 \ifsi@gensymb\else
\tonne3817 \newunit[valuesep=none]{\degree}{\si@sym@degree}
\neper3818 \fi
\bel3819 \fi
\percent3820 \newunit[valuesep=none]{\arcmin}{\si@sym@minute}
3821 \newunit[valuesep=none]{\arcsec}{\si@sym@second}
3822 \newunit{\litre}{l}
3823 \newunit{\liter}{L}
3824 \newunit{\tonne}{t}
3825 \newunit{\neper}{Np}
3826 \newunit{\bel}{B}
3827 \newunit{\percent}{\%}

```

22.7 Units based on physical measurements

`\si@eVspacea` A few units based on physical measurements exist. For `\eV`, the need for a negative kern does make things a bit complicated.

```

\si@eVspaceb
\electronvolt3828 \ProvidesFile{si-physical.cfg}
\atomicmassunit3829 [2008/09/03 v1.01 siunitx: SI Physically-measured units]
\atomicmass3830 \newcommand*{\si@eVspacea}{\text{\kern-\si@eVcorra}}%
3831 \newcommand*{\si@eVspaceb}{\text{\kern-\si@eVcorrb}}%
3832 \newunit{\electronvolt}{\protect\si@eVspacea V\protect%
3833 \si@eVspaceb}
3834 \newunit{\atomicmass}{u}

```

```
3835 \newunit{\atomicmassunit}{u}
```

23 Additional configurations

To provide flexibility for people in specific areas, specialised units can be set up. These are then stored separately to ease use.

23.1 Synthetic chemistry

```
\mmHg  Some useful units for synthetic chemists; although \mmHg and \Molar are out-
\molar  side of the SI system, they are used a lot. These are set up using \provideunit
\Molar  as people may have their own definitions.
\torr3836 \ProvidesFile{si-synchem.cfg}
\dalton3837 [2008/09/03 v1.01 siunitx: Units for synthetic chemists]
3838 \requiresiconfigs{prefix}
3839 \newunit{\mmHg}{mmHg}
3840 \newunit{\molar}{\mole\per\cubic\deci\metre}
3841 \newunit{\Molar}{\textsc{m}}
3842 \newunit{\torr}{Torr}
3843 \newunit{\dalton}{Da}
```

23.2 High-energy physics

Some units inspired by hepunits.

```
3844 \ProvidesFile{si-hep.cfg}
3845 [2008/09/03 v1.01 siunitx: Units for high-energy physics]
3846 \requiresiconfigs{prefix,named}

\micron  These specific to high-energy physics, but are not defined elsewhere in siunitx.54
\mrad3847 \provideunit{\micron}{\micro\metre}
\gauss3848 \newunit{\mrad}{\milli\rad}
3849 \newunit{\gauss}{G}

\cflight  The speed of light is used in units for the area, although of course it is not strictly
\evperc  a unit. However, this makes it easier to use in other units.
3850 \newunit{\cflight}{\ensuremath{\mathnormal{c}}}
3851 \newunit{\evperc}{\eV\per\cflight}

\nanobarn  Various prefixed barns.
\picobarn3852 \newunit{\nanobarn}{\nano\barn}
\femtobarn3853 \newunit{\picobarn}{\pico\barn}
\attobarn3854 \newunit{\femtobarn}{\femto\barn}
\zeptobarn3855 \newunit{\attobarn}{\atto\barn}
\yoctobarn3856 \newunit{\zeptobarn}{\zepto\barn}
3857 \newunit{\yoctobarn}{\yocto\barn}
```

⁵⁴It is not clear from hepunits, but the assumption is that `\mrad` represents millirad and not milliradian.

`\nb` Abbreviations for the same, but using `\provideunit` to avoid any clashes.

```
\pb3858 \provideunit{\nb}{\nano\barn}
\fb3859 \provideunit{\pb}{\pico\barn}
\ab3860 \provideunit{\fb}{\femto\barn}
\zb3861 \provideunit{\ab}{\atto\barn}
\yb3862 \provideunit{\zb}{\zepto\barn}
3863 \provideunit{\yb}{\yocto\barn}
```

23.3 Astronomy

`\parsec` For astronomy, the `\parsec` unit is needed.

```
\lightyear3864 \ProvidesFile{si-astro.cfg}
3865 [2008/09/03 v1.01 siunitx: Units for astronomy]
3866 \newunit{\parsec}{pc}
3867 \newunit{\lightyear}{ly}
```

23.4 Binary units

`\kibi` The binary units, as specified by the IEC and made available by Slunits. First, the
`\mebi` binary prefixes.

```
\gibi3868 \ProvidesFile{si-binary.cfg}
\tebi3869 [2008/09/03 v1.01 siunitx: Binary units]
\pebi3870 \newprefix[binary]{\kibi}{10}{Ki}
\exbi3871 \newprefix[binary]{\mebi}{20}{Mi}
\zebi3872 \newprefix[binary]{\gibi}{30}{Gi}
\yobi3873 \newprefix[binary]{\tebi}{40}{Ti}
3874 \newprefix[binary]{\pebi}{50}{Pi}
3875 \newprefix[binary]{\exbi}{60}{Ei}
3876 \newprefix[binary]{\zebi}{70}{Zi}
3877 \newprefix[binary]{\yobi}{80}{Yi}
```

`\bit` Now the units.

```
\byte3878 \newunit{\bit}{bit}
3879 \newunit{\byte}{B}
```

24 Loadable locales

Some short files to provide the correct settings for various places.

24.1 United Kingdom

This is identical to the USA locale, and contains the package default values.⁵⁵

```
3880 \ProvidesFile{si-UK.cfg}
3881 [2008/09/03 v1.01 siunitx: UK locale]
3882 \sisetup{
3883   unitsep=thin,
3884   expproduct=times,
3885   valuesep=thin,
3886   decimalsymbol=fullstop,
```

⁵⁵The package author is in the UK.

```

3887 digitsep=thin,
3888 sepfour=false}

```

24.2 United States

The same as for the UK.

```

3889 \ProvidesFile{si-USA.cfg}
3890 [2008/09/03 v1.01 siunitx: USA locale]
3891 \sisetup{
3892   unitsep=thin,
3893   expproduct=times,
3894   valuesep=thin,
3895   decimalsymbol=fullstop,
3896   digitsep=thin,
3897   sepfour=false}

```

24.3 Germany

Germany, hopefully.

```

3898 \ProvidesFile{si-DE.cfg}
3899 [2008/09/03 v1.01 siunitx: Germany locale]
3900 \sisetup{
3901   unitsep=cdot,
3902   valuesep=thin,
3903   decimalsymbol=comma,
3904   expproduct=cdot,
3905   digitsep=thin,
3906   sepfour=false}

```

24.4 South Africa

Design decisions taken from the same section of `Slstyle`.

```

3907 \ProvidesFile{si-ZA.cfg}
3908 [2008/09/03 v1.01 siunitx: South Africa locale]
3909 \sisetup{
3910   unitsep=cdot,
3911   valuesep=thin,
3912   expproduct=times,
3913   decimalsymbol=comma,
3914   digitsep=thin,
3915   sepfour=false}

```

25 Emulation code

Each emulation mode loads an appropriate definition file. This then alters the package defaults, and defines new macros provided by the emulated package.

25.1 units

The very first thing to do here is a reload check, as things could go wrong with `unitsdef` emulation.

```
3916 \si@ifloaded{units}\endinput{}
```

The `units` package is quite easy to emulate, as it only has a few options and macros. There is also no error checking in `units` for conflicting options, so users probably expect none.

```
3917 \ProvidesFile{si-units.cfg}
3918 [2008/09/03 v1.01 siunitx: Emulation of units]
3919 \si@emulating{units}{1998/08/04 v0.9b}
3920 \si@ifloaded{SIunits}
3921 {\si@emclash{units}{SIunits}\endinput{}}
3922 \si@ifloaded{sistyle}
3923 {\si@emclash{units}{sistyle}\endinput{}}
```

To emulate `units`, `\per` must give fractions.

```
3924 \sisetup{per=fraction, fraction=nice, obeybold, inlinebold=maths,
3925 , obeymode}
3926 \ifsi@old@tight
3927 \sisetup{valuesep=thin}
3928 \fi
3929 \ifsi@old@loose
3930 \sisetup{valuesep=space}
3931 \fi
3932 \ifsi@old@ugly
3933 \sisetup{fraction=ugly}
3934 \fi
```

`\unit` [**#1**]: **number**

#2 : **unit**

The `units` version of `\unit` is similar to `\SI`. Here and in `\unitfrac` the `\num` macro is used; thus the number given really has to be a number. However, if users are using `siunitx` rather than `units` they should expect more checking of input. As the `units` package uses the current mode, this has to be detected.

```
3935 \si@newrobustcmd*{\unit}[2][]{%
3936 \ifmmode
3937 \SI{#1}{#2}%
3938 \else
3939 \SI[obeyfamily,obeyitalic]{#1}{#2}%
3940 \fi}
```

`\unitfrac` [**#1**]: **number**

#2 : **numerator**

#3 : **denominator**

`\unitfrac` is a bit more of a hack.

```
3941 \si@newrobustcmd*{\unitfrac}[3][]{%
3942 \begingroup
3943 \si@fam@mode%
3944 \ifmmode\else
3945 \sisetup{obeyfamily,obeyitalic}%
3946 \fi
3947 \si@ifnotmtarg{#1}
3948 {\num{#1}\ensuremath{\si@valuesep}}%
3949 \si@frac{#2}{#3}
3950 \endgroup}
```

25.2 unitsdef

The package begins with the usual identification of what is happening. Although `si-units.cfg` makes the same checks, the error will make more sense if it comes here, in the event of a clash.

```
3951 \ProvidesFile{si-unitsdef.cfg}
3952 [2008/09/03 v1.01 siunitx: Emulation of unitsdef]
3953 \si@emulating{unitsdef}{2005/01/04 v0.2}
3954 \si@ifloaded{SIunits}
3955 {\si@emclash{unitsdef}{SIunits}\endinput}{}
3956 \si@ifloaded{sistyle}
3957 {\si@emclash{unitsdef}{sistyle}\endinput}{}

```

Emulation of units is needed for unitsdef to work.

```
3958 \requiresiconfigs{units}

```

The `unitsdef` package loads some packages that `siunitx` does not. In particular, it loads `textcomp` and `fontenc`. This could be important for output, and so the same is done here.

```
3959 \RequirePackage{textcomp}
3960 \RequirePackage[T1]{fontenc}

```

The multitude of package options for `unitsdef` need to be handled.

```
3961 \sisetup{mode=text, allowoptarg, prespace}
3962 \ifsi@old@noxspace
3963 \sisetup{xspace=false}
3964 \fi

```

The various options for loading unit abbreviations have to be handled. Here, any request to avoid abbreviations prevents any loading.

```
3965 \ifsi@old@noabbr
3966 \sisetup{noload=abbr}
3967 \fi
3968 \ifsi@old@nofrequncyabbr
3969 \sisetup{noload=abbr}
3970 \fi
3971 \ifsi@old@nomolabbr
3972 \sisetup{noload=abbr}
3973 \fi
3974 \ifsi@old@novoltageabbr
3975 \sisetup{noload=abbr}
3976 \fi
3977 \ifsi@old@novolumeabbr
3978 \sisetup{noload=abbr}
3979 \fi
3980 \ifsi@old@noweightabbr
3981 \sisetup{noload=abbr}
3982 \fi
3983 \ifsi@old@noenergyabbr
3984 \sisetup{noload=abbr}
3985 \fi
3986 \ifsi@old@nolengthabbr
3987 \sisetup{noload=abbr}
3988 \fi
3989 \ifsi@old@notimeabbr

```

```

3990 \sisetup{noload=abbr}
3991 \fi

```

`\unitvaluesep` To emulate the `\unitvaluesep` macro, a hack is needed of the original `xkeyval` macro for `valuesep`, as well of course as a definition of the macro itself.

```

3992 \newcommand*{\unitvaluesep}{\, }
3993 \renewcommand*{\si@valuesep}{\text{\unitvaluesep}}
3994 \define@choicekey*+[si]{key}{valuesep}[\si@tempa]
3995 {space,thin,med,medium,thick,none}
3996 {\renewcommand*\unitvaluesep\@nameuse{si@fix@##1}%
3997 \si@log@debug{Option valuesep set to ##1}}
3998 {\si@log@debug{Option valuesep set to ##1}%
3999 \renewcommand*\unitvaluesep{##1}}

```

`\unitsignonly` Some rather straight-forward definitions, with just a bit of fun to get the spacing correct.

```

\ilu
\arc4000 \let\unitsignonly\si
4001 \si@newrobustcmd*{\ilu}[2][]{%
4002 \beginngroup
4003 #1\unitvaluesep%
4004 \unit{#2}%
4005 \endgroup}
4006 \let\arc\ang

```

`\unitSIdef` The `unitsdef` package uses a different approach to setting the font inside its version of `\SI`. The problem is the same as for `\unitvaluesep`, but with the added problem that `siunitx` uses `\csname ... \endcsname`.

```

4007 \newcommand*{\unitSIdef}{\upshape}
4008 \newcommand*{\si@unitSIdef}{\unitSIdef\selectfont}
4009 \sisetup{textrm=si@unitSIdef}

```

`\per` Rather awkwardly, `unitsdef` uses `\per` in a different way to `siunitx`.

```

4010 \let\per\relax
4011 \si@newrobustcmd*{\per}[2][{%
4012 \beginngroup
4013 \si@xspacefalse
4014 \renewcommand*{\unitvaluesep}{}%
4015 \unitfrac{#1}{#2}%
4016 \endgroup}

```

`\unittimes` Some pretty straight-forward stuff again; notice that the automatic analyser for units has to be turned off for this to work.

```

\unitsep
\unitsuperscript4017 \newcommand*{\unittimes}{\ensuremath{\cdot}}
4018 \newcommand*{\unitsep}{\, }
4019 \renewcommand*{\si@unt@unithook}{\si@unt@litouttrue}
4020 \sisetup{unitsep=none}
4021 \newcommand*{\unitsuperscript}{\tothe}

```

`\newnosepunit` Simple aliases.

```

\renewnosepunit4022 \newcommand*{\newnosepunit}{\newunit[valuesep=none]}
4023 \newcommand*{\renewnosepunit}{\renewunit[valuesep=none]}

```


`\setTextOmega` Controlling symbols is a simple translation job; as only one setting is used by
`\setMathOmega` siunitx in text mode, a bit of extra work is needed.

```

4024 \newcommand*\setTextOmega[2]{%
4025 \renewcommand*\si@textOmega{%
4026 \begingroup
4027 \edef\si@tempa{\sfdefault}%
4028 \ifx\f@family\si@tempa
4029 \expandafter#2%
4030 \else
4031 \expandafter#1%
4032 \fi
4033 \endgroup}}
4034 \newcommand*\setMathOmega[1]{\sisetup{mathsOmega=#1}}
4035 \newcommand*\setTextmu[2]{%
4036 \renewcommand*\si@textmu{%
4037 \begingroup
4038 \edef\si@tempa{\sfdefault}%
4039 \ifx\f@family\si@tempa
4040 \expandafter#2%
4041 \else
4042 \expandafter#1%
4043 \fi
4044 \endgroup}}
4045 \newcommand*\setMathmu[1]{\sisetup{mathsmu=#1}}
4046 \newcommand*\setTextCelsius[2]{%
4047 \renewcommand*\si@textcelsius{%
4048 \begingroup
4049 \edef\si@tempa{\sfdefault}%
4050 \ifx\f@family\si@tempa
4051 \expandafter#2%
4052 \else
4053 \expandafter#1%
4054 \fi
4055 \endgroup}}
4056 \newcommand*\setMathCelsius[1]{\sisetup{mathscelsius=#1}}
4057 \newcommand*\setMathDegree[2]{%
4058 \renewcommand*\si@textdegree{%
4059 \begingroup%
4060 \edef\si@tempa{\sfdefault}%
4061 \ifx\f@family\si@tempa
4062 \expandafter#2%
4063 \else
4064 \expandafter#1%
4065 \fi
4066 \endgroup}}
4067 \newcommand*\setTextDegree[1]{\sisetup{textdegree=#1}}

```

The `ohm` and `OHM` options are checked, and some sanity is ensured. This needs to happen before loading the configuration files.

```

4068 \ifsi@old@OHM
4069 \ifsi@old@ohm
4070 \si@log@inf{Both 'ohm' and 'OHM' options given\MessageBreak
4071 Using default behaviour for unitsdef}
4072 \expandafter\expandafter\expandafter\si@old@OHMfalse

```

```

4073 \fi
4074 \fi

\liter Tonne is spelled as “ton” by unitsdef, which is wrong in the UK at least (1 ton =
\ton 40 cwt = 2240 lb!).
\days4075 \ifsi@old@liter
4076 \ifsi@old@LITER
4077 \si@log@inf{Both ‘liter’ and ‘LITER’ options
4078 given\MessageBreak Using default behaviour for unitsdef}
4079 \else
4080 \renewunit{\liter}{l}
4081 \fi
4082 \fi
4083 \newunit{\ton}{t}
4084 \newunit{\days}{d}

\picometer Extra distances.
\nanometer4085 \newunit{\picometer}{\pico\meter}
\micrometer4086 \newunit{\nanometer}{\nano\meter}
\millimeter4087 \newunit{\micrometer}{\micro\meter}
\centimeter4088 \newunit{\millimeter}{\milli\meter}
\decimeter4089 \newunit{\centimeter}{\centi\meter}
\kilometer4090 \newunit{\decimeter}{\deci\meter}
4091 \newunit{\kilometer}{\kilo\meter}

\femtoliter Volumes with US spellings.
\picoliter4092 \newunit{\femtoliter}{\femto\liter}
\nanoliter4093 \newunit{\picoliter}{\pico\liter}
\microliter4094 \newunit{\nanoliter}{\nano\liter}
\milliliter4095 \newunit{\microliter}{\micro\liter}
\centiliter4096 \newunit{\milliliter}{\milli\liter}
\deciliter4097 \newunit{\centiliter}{\centi\liter}
4098 \newunit{\deciliter}{\deci\liter}
\hectoliter4099 \newunit{\hectoliter}{\hecto\liter}
\cubicmeter4100 \newunit{\cubicmeter}{\meter\cubed}
\cubicmicrometer4101 \newunit{\cubicmicrometer}{\micro\meter\cubed}
\cubicmillimeter4102 \newunit{\cubicmillimeter}{\milli\meter\cubed}

\squaremeter Areas, including the mis-spellings for \are and \hectare.
\squarecentimeter4103 \newunit{\squaremeter}{\Square\meter}
\squarekilometer4104 \newunit{\squarecentimeter}{\Square\centi\meter}
\ar4105 \newunit{\squarekilometer}{\Square\kilo\meter}
\hectar4106 \newunit{\ar}{a}
4107 \newunit{\hectar}{\hecto\ar}

\kv The code for unitsdef has the capitalisation wrong for \kV and \mV.
\mv4108 \ifsi@old@noabbr
4109 \else
4110 \ifsi@old@novoltageabbr\else
4111 \newunit{\kv}{\kilo\volt}
4112 \newunit{\mv}{\milli\volt}
4113 \fi
4114 \fi

```

`\sek` There are some slightly different abbreviations, plus some which are not officially allowed.

```

\fl4115 \ifsi@old@noabbr\else
\pl4116 \ifsi@old@notimeabbr\else
\nl4117 \newunit{\sek}{\second}
\micl4118 \fi
\ml4119 \ifsi@old@noweightabbr\else
\cl4120 \newunit{\fg}{\femto\gram}
\dl4121 \fi
\hl4122 \ifsi@old@novolumeabbr\else
4123 \newunit{\fl}{\femto\liter}
4124 \newunit{\pl}{\pico\liter}
4125 \newunit{\nl}{\nano\liter}
4126 \newunit{\micl}{\micro\liter}
4127 \newunit{\ml}{\milli\liter}
4128 \newunit{\cl}{\centi\liter}
4129 \newunit{\dl}{\deci\liter}
4130 \newunit{\hl}{\hecto\liter}
4131 \fi
4132 \fi

```

`\calory` `unitsdef` spells calorie incorrectly, and it is also not an SI unit.

```

\kilocalory4133 \newunit{\calory}{cal}
4134 \newunit{\kilocalory}{\kilo\calory}

```

`\uBar` `unitsdef` uses `\ubar` for bar.

```

4135 \newunit{\uBar}{ba}

```

`\gensymbohm` If the options relating to `gensymb` are given, then the package *has* to be loaded.

`\gensymbcelsius` The definitions are then renamed; a slight awkward feature is that the hyphen character needs to be a letter. To avoid needing to worry about this again, a second switch is set up.

```

\gensymbmicro
\gensymbdegree
4136 \catcode'\-=11\relax
4137 \ifsi@old@redef-gensymb
4138 \expandafter\si@gensymbtrue
4139 \fi
4140 \catcode'\-=12\relax
4141 \ifsi@gensymb
4142 \RequirePackage{gensymb}
4143 \AtBeginDocument{
4144 \let\gensymbohm\ohm
4145 \let\gensymbcelsius\celsius
4146 \let\gensymbmicro\micro
4147 \let\gensymbdegree\degree
4148 \let\ohm\@undefined
4149 \let\celsius\@undefined
4150 \let\micro\@undefined
4151 \let\degree\@undefined
4152 \ifsi@old@OHM\else
4153 \newunit{\ohm}{\si@sym@Omega}
4154 \newunit{\celsius}{\si@sym@celsius}
4155 \newprefix{\micro}{\si@sym@mu}{-6}
4156 \newunit{\degree}{\si@sym@degree}

```

```

4157     \fi}
4158 \fi

```

The configuration files can now be loaded.

```

4159 \requiresiconfigs{prefix,named,addn,accepted}

```

The `noconfig` option could be ignored, but it costs little to let it be used.

```

4160 \ifsi@old@noconfig\else
4161   \InputIfFileExists{unitsdef.cfg}
4162   {\si@log@inf{unitsdef config file loaded}}
4163   {\si@log@inf{unitsdef config file not found}}
4164 \fi

```

25.3 Sstyle

After setting the necessary defaults, the emulation code defines the macros in `Sstyle` as given in the manual for that package.

```

4165 \ProvidesFile{si-sstyle.cfg}
4166 [2008/09/03 v1.01 siunitx: Emulation of Sstyle]
4167 \si@emulating{sstyle}{2006/12/20 v2.3}
4168 \sisetup{%
4169   sepfour=true,
4170   obeyfamily,
4171   obeyitalic=true,
4172   numsign=+-,
4173   strictarc=false,
4174   unitsep=cdot}

```

`\SIobeyboldtrue` Some simple switches, but not using `\newif`.

```

\SIobeyboldfalse4175 \newcommand*{\SIobeyboldtrue}{\sisetup{obeybold=true}}
4176 \newcommand*{\SIobeyboldfalse}{\sisetup{obeybold=false}}

```

`\num` To get the correct behaviour for `\num`, some redefinitions are needed to handle to optional `*`.

```

\si@sis@num
\si@sis@numstar4177 \let\num\relax
4178 \si@newrobustcmd*{\num}{%
4179   \@ifstar
4180     {\si@sis@numstar}
4181     {\si@sis@num}}
4182 \newcommand*{\si@sis@num}[2][{}]{%
4183   \begingroup%
4184     \sisetup{#1}%
4185     \expandafter\si@out@num\expandafter{\si@num{#2}}%
4186   \endgroup}
4187 \newcommand*{\si@sis@numstar}[2][{}]{%
4188   \begingroup%
4189     \sisetup{mode=text,obeybold}%
4190     \sisetup{#1}%
4191     \expandafter\si@out@num\expandafter{\si@num{#2}}%
4192   \endgroup}

```

`\pnt` The `\pnt` macro is needed as `.` is active inside `\SI`. The name is exactly the same as in `Sstyle`, but the implementation is different. This is not defined by the

main package as there are better ways of including numbers in the output than this.

```
4193 \newcommand*\pnt{\ensuremath{\si@decimalsymbol}}
```

`\SIgroupfourtrue` Switches for grouping four characters.

```
\SIgroupfourfalse4194 \newcommand*\SIgroupfourtrue{\sisetup{sepfour=true}}
4195 \newcommand*\SIgroupfourfalse{\sisetup{sepfour=false}}
```

`\SIunitsep` Whatever is given here is passed through to `\sisetup`.

```
\SIunitsep4196 \newcommand*\SIunitsep[1]{\sisetup{valuesep={#1}}}
\SIunitdot4197 \newcommand*\SIunitsep[1]{\sisetup{unitsep={#1}}}
4198 \newcommand*\SIunitdot[1]{\sisetup{unitsep={#1}}}
```

`\SIdecimalsymbol` The same is true here, with the appropriate translation.

```
\SIthousandsep4199 \newcommand*\SIdecimalsymbol[1]{\sisetup{decimalsymbol={#1}}}
\SIproductsign4200 \newcommand*\SIthousandsep[1]{\sisetup{digitsep={#1}}}
\SIdecimalsign4201 \newcommand*\SIproductsign[1]{\sisetup{expproduct={#1}}}
4202 \newcommand*\SIdecimalsign[1]{\sisetup{decimalsymbol={#1}}}
```

`\si@sis@savefont` #1 : setting
#2 : argument

The font definitions need a bit of extra work doing. As both settings here have @ as a letter, all should be fine.

```
4203 \newcommand*\si@sis@savefont[2]{%
4204 \@namedef{si@sis@#1}{#2}%
4205 \sisetup{#1=si@sis@#1}}
```

`\SIMathrm` The font control macros have to ensure that a macro name is passed to `\sisetup`.

```
\SIMathsf4206 \newcommand*\SIMathrm[1]{\si@sis@savefont{mathrm}{#1}}
\SIMathtt4207 \newcommand*\SIMathsf[1]{\si@sis@savefont{mathsf}{#1}}
4208 \newcommand*\SIMathtt[1]{\si@sis@savefont{mathtt}{#1}}
```

`\SIdefaultMfam` The same for the default keys.

```
\SIdefaultNfam4209 \newcommand*\SIdefaultMfam[1]{\si@sis@savefont{mathrm}{#1}}
\SIdefaultTfam4210 \newcommand*\SIdefaultNfam[1]{\si@sis@savefont{mathnumrm}{#1}}
4211 \newcommand*\SIdefaultTfam[1]{\si@sis@savefont{textrm}{#1}}
```

`\ensureupmath` The `\ensureupmath` command guarantees processing by the font-matching system. The argument cannot be processed here, so care is needed.

```
4212 \si@newrobustcmd*\ensureupmath[1]{%
4213 \begingroup
4214 \sisetup{mode=maths,obeyitalic=false}%
4215 \si@out{#1}%
4216 \endgroup}
```

`\degC` A few extra symbol names are needed.

```
\degF4217 \newcommand*\degC{\si@sym@celsius}
\arcdeg4218 \newcommand*\arcdeg{\si@sym@degree}
4219 \newcommand*\degF{\si@sym@degree F}
```

`\AddToSIstyle` Finally, the locale control.

```

\SIstyle4220 \newcommand*{\SIstyle}[1]{\sisetup{locale=#1}}
\SIstyleToLang4221 \newcommand*{\SIstyleToLang}[2]{\sisetup{loctolang=#1:#2}}
\si@sis@addtolocale4222 \newcommand*{\AddToSIstyle}{%
4223   \si@switchfalse
4224   \@ifstar
4225     {\si@switchtrue
4226       \si@sis@addtolocale}
4227     {\si@sis@addtolocale}}
4228 \newcommand*{\si@sis@addtolocale}[2]{%
4229   \ifsi@switch
4230     \expandafter\let\csname si@loc@#1@extra\endcsname\relax
4231   \fi
4232   \addtolocale{#1}{#2}}

```

25.4 Slunits

Slunits emulation starts in much the same way.

```

4233 \ProvidesFile{si-SIunits.cfg}
4234 [2008/09/03 v1.01 siunitx: Emulation of SIunits]
4235 \si@emulating{SIunits}{2007/12/02 v1.36}
4236 \sisetup{
4237   unitsep=thick,
4238   valuesep=thick,
4239   prefixproduct=\si@valuesep,
4240   trapambigfrac=false,
4241   stickyper}
4242 \requiresiconfigs{prefix,named,accepted,physical}

```

`\reciprocal` A few very simple translations, using the internal version of `\per` to allow changes of output style.

```

\per4243 \newcommand*{\reciprocal}{\sisetup{per=reciprocal}\si@per}
\usk4244 \let\rp\reciprocal
\power4245 \renewcommand*{\per}{\sisetup{per=slash}\si@per}
\rpsquare4246 \newcommand*{\usk}{}
\rpcubic4247 \newcommand*{\power}[1]{#1\tothe}
\fourth4248 \newcommand*{\rpsquare}{\sisetup{per=reciprocal}\si@per\Square}
4249 \newcommand*{\rpcubic}{\sisetup{per=reciprocal}\si@per\cubic}
\rfourth4250 \newpower{\fourth}{4}
4251 \newcommand*{\rfourth}{\sisetup{per=reciprocal}\si@per\fourth}

```

`\rpsquared` Here, some low-level switch changing is needed.

```

\rpcubed4252 \newcommand*{\rpsquared}{%
4253   \sisetup{per=reciprocal}\si@unt@pertrue\si@unt@perseenttrue%
4254   \squared}
4255 \newcommand*{\rpcubed}{%
4256   \sisetup{per=reciprocal}\si@unt@pertrue\cubed}

```

`\SIsetup` The various package spacing options are processed. They also have to be correctly handled by the `\SIsetup` macro.

```

\si@siu@setup
4257 \newcommand*{\SIsetup}[1]{%
4258   \@for\si@tempa:=#1\do{%
4259     \ifundefined{ifsi@old@#1}

```

```

4260      {\si@log@warn{Unknown SIunits option '#1'}}
4261      {\csname si@old@#1true\endcsname}}
4262 \si@siu@setup
4263 \newcommand*{\si@siu@setup}{%
4264 \ifsi@old@cdot
4265   \sisetup{unitsep=cdot}%
4266 \fi
4267 \ifsi@old@thickspace
4268   \sisetup{unitsep=thick}%
4269 \fi
4270 \ifsi@old@mediumspace
4271   \sisetup{unitsep=medium}%
4272 \fi
4273 \ifsi@old@thinspace
4274   \sisetup{unitsep=thin}%
4275 \fi
4276 \ifsi@old@thickqspace
4277   \sisetup{valuesep=thick}%
4278 \fi
4279 \ifsi@old@mediumqspace
4280   \sisetup{valuesep=medium}%
4281 \fi
4282 \ifsi@old@thingspace
4283   \sisetup{valuesep=thin}%
4284 \fi}
4285 \si@siu@setup

```

`\square` **SIunits** does slightly different things about the clash with `\square`, and either
`\square` **redefines this macro or provides `\square`.**

```

4286 \ifsi@old@squaren
4287 \newpower{\square}{2}
4288 \fi
4289 \AtBeginDocument{%
4290 \ifundefined{square}
4291   {\newpower{\square}{2}}
4292   {\ifsi@old@amssymb
4293     \renewpower{\square}{2}
4294   \else
4295     \ifsi@old@squaren\else
4296       \si@log@warn{\string\square\space already
4297         defined\MessageBreak SIunits mode may cause
4298         errors}%
4299     \fi
4300   \fi}}

```

`\gray` The potential clash with **PStricks** is also handled differently; here, `\Gray` will
 already be defined by the **siunitx** kernel.

```

4301 \AtBeginDocument{
4302 \ifundefined{gray}
4303   {\newunit{\gray}{Gy}}
4304   {\ifsi@old@pstricks
4305     \renewunit{\gray}{Gy}
4306   \else
4307     \ifsi@old@Gray\else

```

```

4308      \si@log@warn{\string\gray\space already
4309      defined\MessageBreak SIunits mode may cause
4310      errors}%
4311      \fi
4312  \fi}}

```

`\unit` The `\unit` macro is defined.

```

\unita4313 \ifsi@old@italian
4314   \let\unita\SI
4315 \else
4316   \let\unit\SI
4317 \fi

```

The miscellaneous options are moped up.

```

4318 \ifsi@old@textstyle
4319   \sisetup{mode=text}
4320 \fi
4321 \ifsi@old@binary
4322   \sisetup{also load= binary}
4323 \fi
4324 \ifsi@old@noams
4325   \AtBeginDocument{%
4326     \renewcommand*{\si@textmu}{\ensuremath\si@mathsmu}}
4327 \fi

```

`\arcminute` The unit macros defined by SIunits that are not defined by siunitx (by default).

```

\arcsecond4328 \newunit[valuesep=none]{\arcminute}{\si@sym@minute}
\rperminute4329 \newunit[valuesep=none]{\arcsecond}{\si@sym@second}
\ton4330 \newunit{\rperminute}{r/min}
\degreecelsius4331 \newunit{\ton}{t}
4332 \newunit{\degreecelsius}{\celsius}

```

`\addunit` This is an alias for `\newunit`.

```

4333 \let\addunit\newunit

```

`\addprefix` A little more work for `\addprefix`.

```

4334 \newcommand*{\addprefix}[2]{\newprefix{#1}{#2}}

```

`\si@siu@newunit` [#1]: power

`\si@siu@power` #2 : numerator

`\si@siu@newunithook` #3 : denominator

To save some code, making new units which need a . . .np variant is handled by a dedicated macro.

```

4335 \newcommand*{\si@siu@newunit}[3][[]]{%

```

A test is needed to sort out `\square`.

```

4336   \renewcommand*{\si@tempa}{#1}%
4337   \renewcommand*{\si@tempb}{square}%
4338   \renewcommand*{\si@siu@power}{}%
4339   \ifx\@empty\si@tempa\@empty\else
4340     \ifx\si@tempa\si@tempb
4341       \renewcommand*{\si@siu@power}{\ssquare}%
4342     \else

```



```

4343     \edef\si@siu@power{%
4344         \expandafter\noexpand\csname #1\endcsname}%
4345     \fi
4346 \fi

```

The necessary information is now stored in temporary macros.

```

4347 \edef\si@tempa{%
4348     \expandafter\noexpand\csname #2per#1#3\endcsname}%
4349 \edef\si@tempb{%
4350     \expandafter\noexpand\csname #2\endcsname\noexpand\per
4351     \expandafter\noexpand\si@siu@power
4352     \expandafter\noexpand\csname #3\endcsname}%
4353 \expandafter\expandafter\expandafter\newunit\expandafter%
4354     \expandafter\expandafter{\expandafter\si@tempa\expandafter}%
4355     \expandafter{\si@tempb}%
4356 \edef\si@tempa{%
4357     \expandafter\noexpand\csname #2per#1#3np\endcsname}%
4358 \edef\si@tempb{%
4359     \expandafter\noexpand\csname #2\endcsname\noexpand
4360     \reciprocal\expandafter\noexpand\si@siu@power
4361     \expandafter\noexpand\csname #3\endcsname}%
4362 \expandafter\expandafter\expandafter\newunit\expandafter
4363     \expandafter\expandafter{\expandafter\si@tempa\expandafter}%
4364     \expandafter{\si@tempb}%
4365 \si@siu@newunithook[#1]{#2}{#3}}
4366 \providecommand*\si@siu@newunithook}[3][{}{}

```

The basic units are now defined; these only have a single csname in each of the numerator and denominator.

```

4367 \si@siu@newunit{gray}{second}
4368 \si@siu@newunit[square]{metre}{second}
4369 \si@siu@newunit{joule}{mole}
4370 \si@siu@newunit[cubic]{mole}{metre}
4371 \si@siu@newunit[square]{radian}{second}
4372 \si@siu@newunit{radian}{second}
4373 \si@siu@newunit[cubic]{squaremetre}{metre}
4374 \si@siu@newunit[cubic]{katal}{metre}
4375 \si@siu@newunit{coulomb}{mol}
4376 \si@siu@newunit[square]{ampere}{metre}
4377 \si@siu@newunit[cubic]{kilogram}{metre}
4378 \si@siu@newunit[cubic]{coulomb}{metre}
4379 \si@siu@newunit{volt}{metre}
4380 \si@siu@newunit[square]{coulomb}{squaremetre}
4381 \si@siu@newunit{farad}{metre}
4382 \si@siu@newunit[square]{watt}{metre}
4383 \si@siu@newunit[square]{joule}{metre}
4384 \si@siu@newunit[cubic]{newton}{metre}
4385 \si@siu@newunit{newton}{kilogram}
4386 \si@siu@newunit{joule}{kelvin}
4387 \si@siu@newunit{joule}{kilogram}
4388 \si@siu@newunit{coulomb}{kilogram}
4389 \si@siu@newunit{squaremetre}{second}
4390 \si@siu@newunit[square]{squaremetre}{second}
4391 \si@siu@newunit[square]{candela}{metre}

```

```

4392 \si@siu@newunit{ampere}{metre}
4393 \si@siu@newunit{joule}{tesla}
4394 \si@siu@newunit{henry}{metre}
4395 \si@siu@newunit{kilogram}{second}
4396 \si@siu@newunit[square]{kilogram}{metre}
4397 \si@siu@newunit{kilogram}{metre}
4398 \si@siu@newunit[square]{newton}{metre}
4399 \si@siu@newunit{watt}{kilogram}
4400 \si@siu@newunit[cubic]{watt}{metre}
4401 \si@siu@newunit{squaremetre}{kilogram}
4402 \si@siu@newunit{cubicmetre}{kilogram}
4403 \si@siu@newunit{newton}{metre}
4404 \si@siu@newunit[cubic]{squaremetre}{second}
4405 \si@siu@newunit{metre}{second}
4406 \si@siu@newunit[cubic]{joule}{metre}
4407 \si@siu@newunit{cubicmetre}{second}

```

\si@siu@newunitx For the more complex units, a slightly different approach is used; four arguments are required, and have to cover everything.

```

4408 \newcommand*{\si@siu@newunitx}[4]{%
4409   \expandafter\newunit\expandafter{\csname #1per#2\endcsname}
4410   {#3\per#4}%
4411   \expandafter\newunit\expandafter{\csname #1per#2np\endcsname}
4412   {#3\reciprocal#4}
4413   \si@siu@newunitxhook{#1}{#2}{#3}{#4}}
4414 \providecommand*{\si@siu@newunitxhook}[4]{}

```

The units are defined.

```

4415 \si@siu@newunitx{kilogramsquaremetre}{second}
4416   {\kilogram\squaremetre}{\second}
4417 \si@siu@newunitx{squaremetre}{newtonsecond}{\squaremetre}
4418   {\newton\second}
4419 \si@siu@newunitx{kilogrammetre}{second}{\kilogram\metre}
4420   {\second}
4421 \si@siu@newunitx{kilogram}{squaremetresecond}{\kilogram}
4422   {\squaremetre\second}
4423 \si@siu@newunitx{joule}{molekelvin}{\joule}{\mole\kelvin}
4424 \si@siu@newunitx{kilogram}{kilomole}{\kilogram}{\kilo\mole}
4425 \si@siu@newunitx{kilogrammetre}{squaresecond}{\kilogram\metre}
4426   {\second\squared}
4427 \si@siu@newunitx{watt}{squaremetresteradian}{\watt}
4428   {\squaremetre\steradian}
4429 \si@siu@newunitx{joule}{kilogramkelvin}{\joule}
4430   {\kilogram\kelvin}
4431 \si@siu@newunitx{watt}{metrekkelvin}{\watt}{\metre\kelvin}
4432 \si@siu@newunitx{kilogram}{cubicmetrecoulomb}{\kilogram}
4433   {\cubic\metre\coulomb}
4434 \si@siu@newunitx{kilogram}{secondcubicmetre}{\kilogram}
4435   {\second\cubicmetre}

```

\si@siu@unity A bit of cleverness to get the “1” correct; to avoid any clash, the unit is given an internal name.

```

4436 \newunit{\si@siu@unity}{1}
4437 \si@siu@newunitx{{squaremetresecond}{\si@siu@unity}

```

```
4438 {\squaremetre\second}
```

A few compound units that are best defined directly.

```
4439 \newunit{\pascalsecond}{\pascal\second}
4440 \newunit{\amperemetresecond}{\ampere\metre\second}
4441 \newunit{\ohmmetre}{\ohm\metre}
4442 \newunit{\newtonmetre}{\newton\metre}
4443 \let\newtonmetrenp\newtonmetre
4444 \newunit{\kilogramsquaremetre}{\kilogram\squaremetre}
4445 \let\kilogramsquaremetrenp\kilogramsquaremetre
```

\si@siu@newprefix #1 : prefix

To generate the prefixes correctly, a small saving in repetition.

```
4446 \newcommand*{\si@siu@newprefix}[1]{%
4447 \edef\si@tempa{\expandafter\noexpand\csname #1d\endcsname}%
4448 \edef\si@tempb{\expandafter\noexpand\csname #1\endcsname}%
4449 \expandafter\expandafter\expandafter\newcommand\expandafter
4450 \expandafter\expandafter*\expandafter\expandafter
4451 \expandafter{\expandafter\si@tempa\expandafter}\expandafter
4452 {\expandafter\si@prefixsymbolicfalse\si@tempb}}
```

This is now implemented.

```
4453 \si@siu@newprefix{yocto}
4454 \si@siu@newprefix{zepto}
4455 \si@siu@newprefix{atto}
4456 \si@siu@newprefix{femto}
4457 \si@siu@newprefix{pico}
4458 \si@siu@newprefix{nano}
4459 \si@siu@newprefix{micro}
4460 \si@siu@newprefix{milli}
4461 \si@siu@newprefix{centi}
4462 \si@siu@newprefix{deca}
4463 \si@siu@newprefix{deka}
4464 \si@siu@newprefix{hecto}
4465 \si@siu@newprefix{kilo}
4466 \si@siu@newprefix{mega}
4467 \si@siu@newprefix{giga}
4468 \si@siu@newprefix{tera}
4469 \si@siu@newprefix{peta}
4470 \si@siu@newprefix{exa}
4471 \si@siu@newprefix{zetta}
4472 \si@siu@newprefix{yotta}
4473 \ifsi@old@binary
4474 \si@siu@newprefix{kibi}
4475 \si@siu@newprefix{mebi}
4476 \si@siu@newprefix{gibi}
4477 \si@siu@newprefix{tebi}
4478 \si@siu@newprefix{pebi}
4479 \si@siu@newprefix{exbi}
4480 \fi
```

\derradian The derived units may need to be defined.

```
\dersteradian4481 \ifsi@old@derived
\derhertz4482 \newunit{\derradian}{\metre\reciprocal\metre}
\dernewton
\derpascal
\derjoule
\derwatt
\dercoulomb
\dervolt
\derfarad
\derohm
```

```

4483 \newunit{\dersteradian}{\squaremetre\rpsquare\metre}
4484 \newunit{\derhertz}{\reciprocal\second}
4485 \newunit{\dernewton}{\metre\kilogram\second\rpsquared}
4486 \newunit{\derpascal}{\newton\rpsquare\metre}
4487 \newunit{\derjoule}{\newton\metre}
4488 \newunit{\derwatt}{\joule\reciprocal\second}
4489 \newunit{\dercoulomb}{\ampere\second}
4490 \newunit{\dervolt}{\watt\reciprocal\ampere}
4491 \newunit{\derfarad}{\coulomb\reciprocal\volt}
4492 \newunit{\derohm}{\volt\reciprocal\ampere}

```

\dersiemens In two blocks!

```

\derweber4493 \newunit{\dersiemens}{\ampere\reciprocal\volt}
\dertesla4494 \newunit{\derweber}
\derhenry4495 {\squaremetre\kilogram\second\rpsquared\reciprocal\ampere}
\dercelsius4496 \newunit{\dertesla}{\weber\rpsquare\metre}
\derlumen4497 \newunit{\derhenry}{\weber\reciprocal\ampere}
\derlux4498 \newunit{\dercelsius}{\kelvin}
\derbecquerel4499 \newunit{\derlumen}{\candela\steradian}
4500 \newunit{\derlux}{\lumen\rpsquare\metre}
\dergray4501 \newunit{\derbecquerel}{\derhertz}
\dersievert4502 \newunit{\dergray}{\joule\reciprocal\kilogram}
\derkatal4503 \newunit{\dersievert}{\dergray}
4504 \newunit{\derkatal}{\rp\second\usk\mole}
4505 \fi

```

\radianbase Also the “derived-in-base”.

```

\steradianbase4506 \ifsi@old@derivedinbase
\hertzbase4507 \newunit{\radianbase}{\metre\reciprocal\metre}
\newtonbase4508 \newunit{\steradianbase}{\squaremetre\rpsquare\metre}
\pascalbase4509 \newunit{\hertzbase}{\reciprocal\second}
\joulebase4510 \newunit{\newtonbase}{\metre\kilogram\second\rpsquared}
\wattbase4511 \newunit{\pascalbase}{\reciprocal\metre\kilogram\second%
4512 \rpsquared}
\coulombbase4513 \newunit{\joulebase}{\squaremetre\kilogram\second\rpsquared}
\voltbase4514 \newunit{\wattbase}{\squaremetre\kilogram\rpcubic\second}
\faradbase4515 \newunit{\coulombbase}{\ampere\second}
\ohmbase4516 \newunit{\voltbase}
4517 {\squaremetre\kilogram\rpcubic\second\reciprocal\ampere}
4518 \newunit{\faradbase}
4519 {\rpsquare\metre\reciprocal\kilogram\fourth\second\ampere%
4520 \squared}
4521 \newunit{\ohmbase}
4522 {\squaremetre\kilogram\rpcubic\second\rpsquare\ampere}

```

\siemensbase Also in two blocks.

```

\weberbase4523 \newunit{\siemensbase}
\teslabase4524 {\rpsquare\metre\reciprocal\kilogram\cubic\second\ampere%
4525 \squared}
\celsiusbase4526 \newunit{\weberbase}
\lumenbase4527 {\squaremetre\kilogram\second\rpsquared\reciprocal\ampere}
\luxbase4528 \newunit{\teslabase}{\kilogram\second\rpsquared\reciprocal%
4529 \ampere}
\becquerelbase4530 \newunit{\henrybase}
\graybase
\sievertbase
\katalbase

```

```

4531     {\squaremetre\kilogram\second\rpsquared\rpsquare\ampere}
4532 \newunit{\celsiusbase}{\kelvin}
4533 \newunit{\lumenbase}{\candela\squaremetre\rpsquare\metre}
4534 \newunit{\luxbase}{\candela\squaremetre\rfourth\metre}
4535 \newunit{\becquerelbase}{\hertzbase}
4536 \newunit{\graybase}{\squaremetre\second\rpsquared}
4537 \newunit{\sievertbase}{\graybase}
4538 \newunit{\katalbase}{\rp\second\mole}
4539 \fi

```

Any configuration file is used if found.

```

4540 \InputIfFileExists{SIunits.cfg}
4541 {\si@log@inf{SIunits config file loaded}}
4542 {\si@log@inf{SIunits config file not found}}

```

25.5 hepunits

The hepunits package provides some rather odd unit names, which are not really to be encouraged.

```

4543 \ProvidesFile{si-hepunits.cfg}
4544 [2008/09/03 v1.01 siunitx: Emulation of hepunits]
4545 \si@emulating{hepunits}{2007/09/27}
4546 \requiresiconfigs{SIunits,accepted,prefix,hep}

```

`\invbarn` Inverses barn units.

```

\invnanobarn4547 \ifsi@old@noprefixcmds\else
\invpicobarn4548 \newunit{\invbarn}{\per\barn}
\invfemtobarn4549 \newunit{\invnanobarn}{\per\nano\barn}
\invattobarn4550 \newunit{\invpicobarn}{\per\pico\barn}
\invzeptobarn4551 \newunit{\invfemtobarn}{\per\femto\barn}
\invyoctobarn4552 \newunit{\invattobarn}{\per\atto\barn}
4553 \newunit{\invzeptobarn}{\per\zepto\barn}
4554 \newunit{\invyoctobarn}{\per\yocto\barn}

```

`\invnb` Also available abbreviated.

```

\invpb4555 \newunit{\invnb}{\per\nano\barn}
\invfb4556 \newunit{\invpb}{\per\pico\barn}
\invab4557 \newunit{\invfb}{\per\femto\barn}
\invzb4558 \newunit{\invab}{\per\atto\barn}
\invyb4559 \newunit{\invzb}{\per\zepto\barn}
4560 \newunit{\invyb}{\per\yocto\barn}
4561 \fi

```

`\invcmsqpersecond` Luminosity.

```

\invcmsqpersec4562 \newunit{\invcmsqpersecond}{\per\Square\centi\metre\per\second}
\lumiunits4563 \newunit{\invcmsqpersec}{\per\Square\centi\metre\per\second}
4564 \newunit{\lumiunits}{\per\Square\centi\metre\per\second}

```

`\inveV` The inverse of an electron-volt, plus prefixes.

```

\minveV4565 \newunit{\inveV}{\per\electronvolt}
\minveV4566 \newunit{\minveV}{\per\milli\electronvolt}
\kinveV4567 \newunit{\kinveV}{\per\kilo\electronvolt}
\MinveV4568 \newunit{\MinveV}{\per\mega\electronvolt}
\GinveV
\TinveV

```

```

4569 \newunit{\GinveV}{\per\giga\electronvolt}
4570 \newunit{\TinveV}{\per\tera\electronvolt}

\evoverc Some combinations of electron-volts and the speed of light. As these are called
\evovercsq over, they are set with a slash. The eVcorrb values have been set for Computer
Modern.

4571 \newunit[per=slash,eVcorrb=0.6ex]{\evoverc}
4572 {\electronvolt\per\clight}
4573 \newunit[per=slash,eVcorrb=0.6ex]{\evovercsq}
4574 {\electronvolt\per\Square\clight}

\meVoverc Prefixed combinations, first of the speed of light.
\keVoverc4575 \newunit[per=slash,eVcorrb=0.6ex]{\meVoverc}
\MeVoverc4576 {\milli\electronvolt\per\clight}
\GeVoverc4577 \newunit[per=slash,eVcorrb=0.6ex]{\keVoverc}
\TeVoverc4578 {\kilo\electronvolt\per\clight}
4579 \newunit[per=slash,eVcorrb=0.6ex]{\MeVoverc}
4580 {\mega\electronvolt\per\clight}
4581 \newunit[per=slash,eVcorrb=0.6ex]{\GeVoverc}
4582 {\giga\electronvolt\per\clight}
4583 \newunit[per=slash,eVcorrb=0.6ex]{\TeVoverc}
4584 {\tera\electronvolt\per\clight}

\meVovercsq Then of the square.
\keVovercsq4585 \newunit[per=slash,eVcorrb=0.6ex]{\meVovercsq}
\MeVovercsq4586 {\milli\electronvolt\per\Square\clight}
\GeVovercsq4587 \newunit[per=slash,eVcorrb=0.6ex]{\keVovercsq}
\TeVovercsq4588 {\kilo\electronvolt\per\Square\clight}
4589 \newunit[per=slash,eVcorrb=0.6ex]{\MeVovercsq}
4590 {\mega\electronvolt\per\Square\clight}
4591 \newunit[per=slash,eVcorrb=0.6ex]{\GeVovercsq}
4592 {\giga\electronvolt\per\Square\clight}
4593 \newunit[per=slash,eVcorrb=0.6ex]{\TeVovercsq}
4594 {\tera\electronvolt\per\Square\clight}

```

25.6 fancynum

`fancynum` only does things with numbers, so there is only a little emulation and a few macros needed.

```

4595 \ProvidesFile{si-fancynum.cfg}
4596 [2008/09/03 v1.01 siunitx: Emulation of fancynum]
4597 \si@emulating{fancynum}{2000/08/08 0.92}
4598 \sisetup{decimalsymbol=cdot,digitsep=comma}

\fnm The \fnm macro is rather restricted, but this is not reproduced. Instead, it is
\let as an alias to the \num macro.

4599 \let\fnm\num

```

`\setfnumdsym` The control macros are defined.

```

\setfnumgsym4600 \newcommand*\setfnumdsym[1]{\sisetup{decimalsymbol={#1}}}
\setfnummsym4601 \newcommand*\setfnumgsym[1]{\sisetup{digitsep={#1}}}
4602 \newcommand*\setfnummsym[1]{\sisetup{expproduct={#1}}}

```

The various package options are now processed if necessary.

```

4603 \ifsi@old@english
4604 \sisetup{decimalsymbol=cdot,digitsep=comma}
4605 \fi
4606 \ifsi@old@french
4607 \sisetup{decimalsymbol=comma,digitsep=fullstop}
4608 \fi
4609 \ifsi@old@tight
4610 \sisetup{expproduct=tighttimes}
4611 \fi
4612 \ifsi@old@loose
4613 \sisetup{expproduct=times}
4614 \fi
4615 \ifsi@old@thinspace
4616 \sisetup{digitsep=thin}
4617 \fi
4618 \ifsi@old@commas
4619 \sisetup{digitsep=comma}
4620 \fi
4621 \ifsi@old@plain
4622 \sisetup{digitsep=none}
4623 \fi

```

25.7 fancyunits

The fancyunits package is not available on CTAN, but is available from its authors homepage [7]. It is similar to Slunits, and so most of the code is shared here. However, a few bits of set up occur first, and an emulation-clash test is needed.

```

4624 \ProvidesFile{si-fancyunits.cfg}
4625 [2008/09/03 v1.01 siunitx: Emulation of fancyunits]
4626 \si@emulating{fancyunits}{2007/02/01 v1.0.1}
4627 \si@ifloaded{SIunits}
4628 {\si@log@err{SIunits emulation loaded\MessageBreak before
4629 fancyunits emulation}{You need to load the fancyunits
4630 emulation\MessageBreak code before that for
4631 SIunits.\MessageBreak Try emulate=fancyunits as the first
4632 option when\MessageBreak loading siunitx}}{}

```

\si@siu@newunithook To create the extra macros provided by fancyunits, the Slunits emulation code is changed to add the “uf” variants.

```

\si@siu@newunitxhook
4633 \newcommand*{\si@siu@newunithook}[3][{}]{%
4634 \edef\si@tempa{%
4635 \expandafter\noexpand\csname #2per#1#3uf\endcsname}%
4636 \renewcommand*{\si@tempb}{stickyper,per=fraction,
4637 fraction=nice}%
4638 \edef\si@tempc{%
4639 \noexpand\sisetup{\si@tempb}%
4640 \expandafter\noexpand\csname #2\endcsname\noexpand\si@per%
4641 \expandafter\noexpand\si@siu@power%
4642 \expandafter\noexpand\csname #3\endcsname}%
4643 \expandafter\expandafter\expandafter\newunit\expandafter
4644 \expandafter\expandafter\expandafter\si@tempa\expandafter}%
4645 \expandafter\si@tempc}%

```

```

4646 \edef\si@tempa{%
4647   \expandafter\noexpand\csname #2per#1#3Uf\endcsname}%
4648 \renewcommand*{\si@tempb}{stickyper,per=fraction,
4649   fraction=frac}%
4650 \edef\si@tempc{%
4651   \noexpand\sisetup{\si@tempb}%
4652   \noexpand\def\noexpand\si@frc@hook{\noexpand\textstyle}%
4653   \expandafter\noexpand\csname #2\endcsname\noexpand\si@per%
4654   \expandafter\noexpand\si@siu@power%
4655   \expandafter\noexpand\csname #3\endcsname}%
4656 \expandafter\expandafter\expandafter\newunit\expandafter
4657   \expandafter\expandafter{\expandafter\si@tempa\expandafter}%
4658   \expandafter{\si@tempc}%
4659 \edef\si@tempa{%
4660   \expandafter\noexpand\csname #2per#1#3UF\endcsname}%
4661 \edef\si@tempc{%
4662   \noexpand\sisetup{\si@tempb}%
4663   \noexpand\def\noexpand\si@frc@hook{\noexpand\displaystyle}%
4664   \expandafter\noexpand\csname #2\endcsname\noexpand\si@per%
4665   \expandafter\noexpand\si@siu@power%
4666   \expandafter\noexpand\csname #3\endcsname}%
4667 \expandafter\expandafter\expandafter\newunit\expandafter
4668   \expandafter\expandafter{\expandafter\si@tempa\expandafter}%
4669   \expandafter{\si@tempc}}
4670 \newcommand*{\si@siu@newunitxhook}[4]{%
4671   \expandafter\newunit\expandafter{\csname #1per#2uf\endcsname}
4672   {\sisetup{stickyper,per=fraction,fraction=nice}%
4673     #3\si@per#4}%
4674   \expandafter\newunit\expandafter{\csname #1per#2Uf\endcsname}
4675   {\sisetup{stickyper,per=fraction,fraction=frac}%
4676     \renewcommand*{\si@frc@hook}{\textstyle}%
4677     #3\si@per#4}%
4678   \expandafter\newunit\expandafter{\csname #1per#2UF\endcsname}
4679   {\sisetup{stickyper,per=fraction,fraction=frac}%
4680     \renewcommand*{\si@frc@hook}{\displaystyle}%
4681     #3\si@per#4}}

```

With that done, the emulation modules can be loaded.

```

4682 \requiresinconfigs{SIunits,addn,astro}
4683 \sisetup{obeyall}

```

There is one fancyunits-specific option to handle. The other options all get sent through to the Slunits system.⁵⁶

```

4684 \ifsi@old@spaceqspace
4685   \sisetup{valuesep=space}
4686 \fi

```

```

\pamminute fancyunits provides some extra units, plus tonne spelled incorrectly (again).
\parsecond4687 \newunit{\pamminute}{'}
\AstroE4688 \newunit{\parsecond}{''}
\oersted4689 \newunit{\AstroE}{AE}
\ton4690 \newunit{\oersted}{OE}

```

⁵⁶As Slunits is rather more widely known than fancyunits, any other options which could be for either are assumed to be for Slunits.


```

4691 \provideunit{\ton}{t}

\decaD  An additional prefix.
4692 \let\decaD\decad

\ufrac  The fractional unit macros need to be reproduced.
\Ufrac4693 \newcommand*{\ufrac}[2]{%
\Ufrac4694   \si[stickyper,per=fraction,fraction=nice]{#1\si@per#2}}
4695 \newcommand*{\Ufrac}[2]{%
4696   \ensuremath{\textstyle{%
4697     \si[stickyper,per=fraction,fraction=frac]{#1\si@per#2}}}}
4698 \newcommand*{\Ufrac}[2]{%
4699   \ensuremath{\displaystyle{%
4700     \si[stickyper,per=fraction,fraction=frac]{#1\si@per#2}}}}

\pow  A slightly-shortened named \power.
4701 \let\pow\power

\Squaremetre  An alias for \squaremetre.
4702 \let\Squaremetre\squaremetre

    As with Slunits, there is now a list of compound units to add. Only a few are not
    covered by the Slunits emulation.
4703 \si@siu@newunit{Gray}{second}
4704 \si@siu@newunit[square]{Squaremetre}{metre}
4705 \si@siu@newunitx{Squaremetre}{newtonsecond}{\Square\metre}
4706   {\newton\second}
4707 \si@siu@newunit{Squaremetre}{second}
4708 \si@siu@newunit[square]{Squaremetre}{squaresecond}
4709 \si@siu@newunit{Squaremetre}{kilogram}
4710 \si@siu@newunit[cubic]{Squaremetre}{second}

```

Part V

Notes

26 Change History

1.of	\si@out: Fixed bug with colour and spacing 141	v1.ob	\si@num@rndpost: Corrected bug in rounding code 101
1.0j	\si@tab@gettok@S: Token gathering code now checks for expandable content 113	v1.0c	General: Fixed excess loading of maths fonts 72
v1.0	General: First official release 1	v1.od	\si@unt@out: Sanitise one level of unit input only 137
v1.0a	\si@fix@space: Fixed problem with space option 75	v1.0h	General: Simplified warnings for tabformat 67
	\si@loc@ltol: babel support fixed for default document language 140	v1.ok	\si@tab@end@S: Fixed problem with coloured cells 116
	\si@tab@end@S: Fixed mixed number/text column alignment 116	v1.ol	\si@tab@fixed: Fixed issue with spacing when no digits present after decimal marker 118
	\si@tab@gettok@s: Fixed issues with alignment of s column contents 115		\si@tab@unfixed: Fixed issue with decimal sign and integers 117

27 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.

Symbols			
\% 3827	\addunit <u>4333</u>
\& 3223, 3226	allowoptarg (option) 30
\@@input 3428	allowzeroexp (option) 25
\@addtofilelist 3428	alsoload (option) 32
\@filef@und 3428	\ampere	. 14 , <u>3273</u> , 3679–3683, 3740–3744, 4440, 4489, 4490, 4492, 4493, 4495, 4497, 4515, 4517, 4519, 4522, 4524, 4527, 4529, 4531
\^ 17, 27, 1227, 3360, 3364	\amu 16 , <u>3763</u>
\` 15, 25	\ang	... 10 , <u>2004</u> , 2028, 2029, 2033, 4006
\~ 18, 28, 1226, 3204, 3269	angformat (option) 26
A		anglesep (option) 23
\ab 20 , <u>3858</u>	\angstrom 16 , <u>3793</u>
\addprefix <u>4334</u>	\ar <u>4103</u>
addsign (option) 25	\arc <u>4000</u>
\addtolocale 36 , <u>3357</u> , 4232		
\AddToSistyle <u>4220</u>		

<code>\arcdeg</code>	4217	<code>colorall (option)</code>	31
<code>\arcmin</code>	15 , 3807	<code>colorneg (option)</code>	31
<code>\arcminute</code>	4328	<code>colorunit (option)</code>	31
<code>\arcsec</code>	15 , 3807	<code>colorvalue (option)</code>	31
<code>\arcsecond</code>	4328	<code>colour (option)</code>	31
<code>\are</code>	16 , 3793	<code>colourall (option)</code>	31
<code>\as</code>	16 , 3786	<code>colourneg (option)</code>	31
<code>astroang (option)</code>	27	<code>colourunits (option)</code>	31
<code>\AstroE</code>	4687	<code>colourvalues (option)</code>	31
<code>\atomicmass</code>	15 , 16 , 3770 , 3828	<code>\coulomb</code>	15 , 3616 , 4433 , 4491
<code>\atomicmassunit</code>	15 , 3828	<code>\coulombbase</code>	4506
<code>\atto</code>	14 , 3585 , 3673 , 3787 , 3855 , 3861 , 4552 , 4558	<code>\cubed</code>	12 , 3281 , 3708 – 3712 , 3760 , 3761 , 4100 – 4102 , 4256
<code>\attobarn</code>	20 , 3852	<code>\cubic</code>	12 , 3281 , 3713 , 3840 , 4249 , 4433 , 4524
<code>\attosecond</code>	16 , 3673	<code>\cubiccentimetre</code>	17 , 3706
B		<code>\cubicdecimetre</code>	17 , 3706
<code>\BAR</code>	16 , 3793	<code>\cubicmeter</code>	4092
<code>\barn</code> ..	16 , 3793 , 3852 – 3863 , 4548 – 4560	<code>\cubicmetre</code>	3706 , 4435
<code>\bbar</code>	16 , 3793	<code>\cubicmicrometer</code>	4092
<code>\becquerel</code>	15 , 3616 , 3735	<code>\cubicmicrometre</code>	3706
<code>\becquerelbase</code>	4523	<code>\cubicmillimeter</code>	4092
<code>\bel</code>	15 , 3807	<code>\cubicmillimetre</code>	3706
<code>\bit</code>	19 , 3878	<code>\curie</code>	16 , 3793
<code>\byte</code>	19 , 3878	D	
C		<code>\dalton</code>	19 , 3836
<code>\calory</code>	4133	<code>\Day</code>	15 , 3807
<code>\candela</code> ...	14 , 3273 , 4499 , 4533 , 4534	<code>\days</code>	4075
<code>\Celsius</code>	3644	<code>\dday</code>	15 , 3807
<code>\celsius</code> 15 , 3644 , 4145 , 4149 , 4154 , 4332		<code>debug (option)</code>	32
<code>\celsiusbase</code>	4523	<code>\deca</code>	14 , 3603
<code>\centi</code> ..	14 , 3585 , 3660 , 3709 , 3710 , 3715 , 3716 , 3760 , 3762 , 3783 , 4089 , 4097 , 4104 , 4128 , 4562 – 4564	<code>\decaD</code>	4692
<code>\centiliter</code>	4092	<code>\decad</code>	4692
<code>\centimeter</code>	4085	<code>\deci</code>	14 , 3585 , 3661 , 3713 , 3761 , 3784 , 3840 , 4090 , 4098 , 4129
<code>\centimetre</code>	16 , 3656	<code>\deciliter</code>	4092
<code>\centimetrecubed</code>	17 , 3706	<code>decimalsymbol (option)</code>	24
<code>\centimetresquared</code>	17 , 3714	<code>\decimeter</code>	4085
<code>\cl</code>	4115	<code>\decimetre</code>	16 , 3656
<code>\clight</code>	20 , 3850 , 4572 , 4574 , 4576 , 4578 , 4580 , 4582 , 4584 , 4586 , 4588 , 4590 , 4592 , 4594	<code>\degC</code>	4217
<code>closeerr (option)</code>	24	<code>\degF</code>	4217
<code>closefrac (option)</code>	29	<code>\Degree</code>	3807
<code>\cm</code>	16 , 3779	<code>\degree</code> ...	15 , 3807 , 4147 , 4151 , 4156
<code>\cmc</code>	17 , 3758	<code>\degreecelsius</code>	4328
<code>\cms</code>	17 , 3758	<code>\deka</code>	3613
<code>color (option)</code>	31	<code>\derbecquerel</code>	4493
		<code>\dercelsius</code>	4493
		<code>\dercoulomb</code>	4481
		<code>\derfarad</code>	4481

[illegible]

\hectoliter	<u>4092</u>	\ifsi@old@binary ...	<u>858</u> , 4321, 4473
\hectopascal	<u>18</u> , <u>3732</u>	\ifsi@old@cdot	<u>846</u> , 4264
\henry	<u>15</u> , <u>3616</u>	\ifsi@old@commas	<u>868</u> , 4618
\henrybase	<u>4523</u>	\ifsi@old@derived	<u>858</u> , 4481
\hertz <u>15</u> , <u>17</u> , <u>3616</u> , 3727–3731, 3745–3750		\ifsi@old@derivedinbase	<u>858</u> , 4506
\hertzbase	<u>4506</u> , 4535	\ifsi@old@english	<u>864</u> , 4603
\hl	<u>4115</u>	\ifsi@old@french	<u>864</u> , 4606
\hour	<u>15</u> , 3726, 3778, <u>3807</u>	\ifsi@old@Gray	<u>853</u> , 4307
\Hz	<u>17</u> , <u>3745</u>	\ifsi@old@italian	<u>853</u> , 4313
I			
\ifdefined	1022, 1029, 3245	\ifsi@old@LITER	832, 4076
\ifsi@addunitpower	667, 2690	\ifsi@old@liter	832, 4075
\ifsi@allowoptarg	644, 2797	\ifsi@old@loose .	<u>824</u> , <u>866</u> , 3929, 4612
\ifsi@allowzeroexp	<u>378</u> , 1319	\ifsi@old@mediumqspace .	<u>846</u> , 4279
\ifsi@ang@fixdp	2013, <u>2054</u>	\ifsi@old@mediumspace ..	<u>846</u> , 4270
\ifsi@ang@padlarge	<u>465</u> , 2180	\ifsi@old@nice	<u>824</u>
\ifsi@ang@padsmall	<u>465</u> , 2234	\ifsi@old@noabbr <u>836</u> , 3965, 4108, 4115	
\ifsi@ang@sign	1664, <u>2178</u>	\ifsi@old@noamperageabbr ...	<u>836</u>
\ifsi@ang@toarc	<u>494</u> , 2042	\ifsi@old@noams	<u>858</u> , 4324
\ifsi@ang@todec	<u>494</u> , 2049	\ifsi@old@noconfig	832, 4160
\ifsi@astroang	<u>516</u> , 2190	\ifsi@old@noenergyabbr .	<u>836</u> , 3983
\ifsi@colourneg	<u>760</u> , 1307	\ifsi@old@nofrequncyabbr	<u>836</u> , 3968
\ifsi@colourunits	<u>721</u> , 1100	\ifsi@old@nolengthabbr .	<u>836</u> , 3986
\ifsi@colourvalues .	<u>721</u> , 1089, 2494	\ifsi@old@nomolabbr	<u>836</u> , 3971
\ifsi@debug	<u>116</u> , 141	\ifsi@old@noprefixcmds .	<u>863</u> , 4547
\ifsi@detectdisplay	<u>332</u> , 1121, 1139	\ifsi@old@notimeabbr <u>836</u> , 3989, 4116	
\ifsi@fam@set	<u>1064</u> , 1080, 3367	\ifsi@old@novoltageabbr	<u>836</u> , 3974, 4110
\ifsi@fixdp	<u>627</u> , 1688, 2013	\ifsi@old@novolumeabbr	<u>836</u> , 3977, 4122
\ifsi@frac .	<u>645</u> , 2984, 3009, 3133, 3139	\ifsi@old@noweightabbr	<u>836</u> , 3980, 4119
\ifsi@gensymb	<u>828</u> , 3596, 3633, 3647, 3700, 3816, 4141	\ifsi@old@noxspace	<u>832</u> , 3962
\ifsi@inlinebtext	<u>324</u> , 1044	\ifsi@old@OHM	<u>828</u> , 3593, 3630, 3644, 3695, 3813, 4068, 4152
\ifsi@logmin	<u>116</u> , 121, 129, 135	\ifsi@old@ohm	<u>828</u> , 4069
\ifsi@lognone .	<u>116</u> , 120, 128, 134, 140	\ifsi@old@plain	<u>868</u> , 4621
\ifsi@num@ambigerr	1344, <u>1722</u>	\ifsi@old@pstricks	<u>853</u> , 4304
\ifsi@num@delplus ...	560, 568, <u>1528</u>	\ifsi@old@redef	4137
\ifsi@num@erropen .	<u>1273</u> , 1371, 1404	\ifsi@old@redef-gensymb	<u>828</u>
\ifsi@num@intab <u>1215</u> , 1333, 1374, 1398		\ifsi@old@spaceqspace ..	<u>871</u> , 4684
\ifsi@num@padlead ..	<u>383</u> , 1575, 1812	\ifsi@old@squaren ..	<u>853</u> , 4286, 4295
\ifsi@num@padtrail	<u>383</u> , 1580	\ifsi@old@textstyle	<u>858</u> , 4318
\ifsi@num@signexp	420, 1548	\ifsi@old@thickqspace ..	<u>846</u> , 4276
\ifsi@num@signmant	420, 1542	\ifsi@old@thickspace ...	<u>846</u> , 4267
\ifsi@numtextmode	<u>292</u> , 1068	\ifsi@old@thingqspace ...	<u>846</u> , 4282
\ifsi@obeybold	<u>323</u> , 1135	\ifsi@old@thinspace	<u>846</u> , 4273
\ifsi@obeyfamily	<u>322</u> , 1113	\ifsi@old@thinspacees ...	<u>868</u> , 4615
\ifsi@obeyitalic	<u>331</u> , 1153	\ifsi@old@tight .	<u>824</u> , <u>866</u> , 3926, 4609
\ifsi@obeymode	<u>291</u> , 1053	\ifsi@old@ugly	<u>824</u> , 3932
\ifsi@old@amssymb	<u>853</u> , 4292		

\kilowatt	18, 3684	\megaohm	18, 3695
\kilowatthour	17, 3718	\megawatt	18, 3684
\kinveV	4565	\meter	13, 3273, 4085–4091, 4100–4105
\kJ	17, 3771	\metre	13, 14, 3273, 3656–3662, 3708–3717, 3760–3762, 3779– 3785, 3840, 3847, 4419, 4425, 4431, 4433, 4440–4442, 4482, 4483, 4485–4487, 4496, 4500, 4507, 4508, 4510, 4511, 4519, 4524, 4533, 4534, 4562–4564, 4705
\km	16, 3779	\MeV	17, 3771
\kV	17, 3756	\meV	17, 3771
\kv	4108	\MeVoverc	4575
\kWh	3771	\meVoverc	4575
L			
\language	3339	\MeVovercsq	4585
\lccode	3225, 3226	\meVovercsq	4585
\lightyear	21, 3864	\mg	16, 3763
\liter	14, 15, 3807, 4075, 4092–4099, 4123–4130	\MHz	17, 3745
\litre	14, 15, 3706, 3707, 3758, 3759, 3807	\mHz	17, 3745
load (option)	32	\micA	17, 3737
locale (option)	31	\micg	16, 3763
loctolang (option)	31	\micl	17, 3758, 4115
log (option)	32	\micm	16, 3779
\lumen	15, 3616, 4500	\micmol	17, 3751
\lumenbase	4523	\Micro	3585
\lumiunits	4562	\micro	14, 3585, 3658, 3666, 3671, 3677, 3681, 3692, 3706, 3711, 3742, 3754, 3759, 3768, 3781, 3791, 3847, 4087, 4095, 4101, 4126, 4146, 4150, 4155
\lux	15, 3616	\microampere	17, 3679
\luxbase	4523	\microfarad	18, 3684
M			
\mA	17, 3737	\microgram	16, 3663
\makeatother	3431	\microliter	4092
\mathrm (option)	23	\microlitre	17, 3706
\mathscelsius (option)	30	\micrometer	4085
\mathsdegree (option)	30	\micrometre	16, 3656
\mathsf (option)	23	\micromole	17, 3668
\mathsminute (option)	30	\micron	20, 3847
\mathsmu (option)	30	\microsecond	16, 3673
\mathsOmega (option)	30	\mics	16, 3786
\mathsringA (option)	30	\milli	14, 3585, 3659, 3667, 3672, 3678, 3682, 3684, 3686, 3693, 3694, 3707, 3712, 3718, 3721, 3727, 3732, 3736, 3743, 3746, 3755, 3757, 3758, 3769, 3773, 3782, 3792, 3801, 3848, 4088, 4096, 4102, 4112, 4127, 4566, 4576, 4586
\mathsrm (option)	23	\milliampere	17, 3679
\mathssecond (option)	30	\millibar	16, 3793
\mathssf (option)	23		
\mathstt (option)	23		
\mathtt (option)	23		
\meaning	2397		
\mebi	20, 3868		
\mega	14, 3603, 3688, 3697, 3702, 3720, 3723, 3729, 3735, 3748, 3775, 4568, 4580, 4590		
\megabecquerel	18, 3732		
\megaelectronvolt	17, 3718		
\megahertz	17, 3727		
\megajoule	3718		

\millielectronvolt	17, 3718	\nanosecond	16, 3673
\millifarad	18, 3684	\nb	20, 3858
\milligram	16, 3663	\NC@list	2278
\millihertz	17, 3727	\NC@rewrite@S	2283
\millijoule	3718	\NC@rewrite@s	2283
\milliliter	4092	negcolor (option)	31
\millilitre	17, 3706	negcolour (option)	31
\millimeter	4085	\neper	15, 3807
\millimetre	16, 3656	\newnosepunit	4022
\millimole	17, 3668	\newpower	19,
\millinewton	18, 3732	2645, 3281–3285, 4250, 4287, 4291	
\millisecond	16, 3673	\newprefix	
\millisiemens	18, 3684	19, 2631, 3587–3592, 3594, 3597,	
\millisievert	18, 3732	3600–3613, 3870–3877, 4155, 4334	
\millivolt	17, 3684	\newton	15, 3616, 3732,
\milliwatt	18, 3684	3733, 4418, 4442, 4486, 4487, 4706	
\minute	15, 3807	\newtonbase	4506
\MinveV	4565	\newunit	18, 2617,
\minveV	4565	3273–3280, 3614, 3618–3629,	
\ml	17, 3758, 4115	3631, 3637–3643, 3645, 3648,	
\mm	16, 3779	3651, 3652, 3656–3694, 3696–	
\mmHg	19, 3836	3698, 3701–3703, 3706–3736,	
\mmol	17, 3751	3740–3757, 3760–3763, 3765–	
mode (option)	22	3788, 3790–3792, 3795–3806,	
\Molar	19, 3836	3809–3812, 3814, 3817, 3820–	
\molar	19, 3836	3827, 3832, 3834, 3835, 3839–	
\mole	14, 3273, 3668–3672, 3751–	3843, 3848–3857, 3866, 3867,	
3755, 3840, 4423, 4424, 4504, 4538		3878, 3879, 4022, 4083–4107,	
\mp	1241	4111, 4112, 4117, 4120, 4123–	
\mrad	20, 3847	4130, 4133–4135, 4153, 4154,	
\ms	16, 3786	4156, 4303, 4328–4333, 4353,	
\mV	17, 3756	4362, 4409, 4411, 4436, 4439–	
\mv	4108	4442, 4444, 4482–4494, 4496–	
		4504, 4507–4511, 4513–4516,	
		4518, 4521, 4523, 4526, 4528,	
		4530, 4532–4538, 4548–4560,	
		4562–4571, 4573, 4575, 4577,	
		4579, 4581, 4583, 4585, 4587,	
		4589, 4591, 4593, 4643, 4656,	
		4667, 4671, 4674, 4678, 4687–4690	
		\nl	4115
		\nm	16, 3779
		\nmol	17, 3751
		noload (option)	32
		\ns	16, 3786
		\num	5, 1207, 1238, 2684,
		2888, 3023, 3157, 3948, 4177, 4599	
		numaddn (option)	24
		numcloseerr (option)	24
		numdecimal (option)	24
		numdigits (option)	24
		numexp (option)	24
N			
\nA	17, 3737		
\nano	14, 3585, 3657, 3665,		
3670, 3676, 3680, 3691, 3741,			
3753, 3767, 3780, 3790, 3852,			
3858, 4086, 4094, 4125, 4549, 4555			
\nanoampere	17, 3679		
\nanobarn	20, 3852		
\nanofarad	18, 3684		
\nanog	16, 3763		
\nanogram	16, 3663		
\nanoliter	4092		
\nanometer	4085		
\nanometre	16, 3656		
\nanomole	17, 3668		

numgobble (option)	24	loctolang	31
numopenerr (option)	24	log	32
numprod (option)	25	mathrm	23
numsign (option)	24	mathscelsius	30
O			
obeyall (option)	22	mathsdegree	30
obeybold (option)	22	mathsf	23
obeyfamily (option)	22	mathsminute	30
obeyitalic (option)	22	mathsmu	30
obeymode (option)	22	mathsOmega	30
\oersted	4687	mathsringA	30
\Ohm	3630, 3696–3698	mathsrn	23
\ohm	15, 3630,	mathssecond	30
3701–3703, 4144, 4148, 4153, 4441		mathssf	23
\ohmbase	4506	mathstt	23
\Omega	772, 773	mathtt	23
openerr (option)	24	mode	22
openfrac (option)	29	negcolor	31
options:		negcolour	31
addsign	25	noload	32
allowoptarg	30	numaddn	24
allowzeroexp	25	numcloseerr	24
alsoload	32	numdecimal	24
angformat	26	numdigits	24
anglesep	23	numexp	24
astroang	27	numgobble	24
closeerr	24	numopenerr	24
closefrac	29	numprod	25
color	31	numsign	24
colorall	31	obeyall	22
colorneg	31	obeybold	22
colorunit	31	obeyfamily	22
colorvalue	31	obeyitalic	22
colour	31	obeymode	22
colourall	31	openerr	24
colourneg	31	openfrac	29
colourunits	31	padangle	26
colourvalues	31	padnumber	25
debug	32	per	29
decimalsymbol	24	prefixbase	29
detectdisplay	23	prefixproduct	29
digitsep	23, 24	prefixsymbolic	29
dp	26	prespace	30
emulate	32	redefsymbols	31
errspace	23	repeatunits	25
eVcorra	31	retainplus	25
eVcorrb	31	seperr	24
expbase	25	sepfour	24
expproduct	25	sign	25
fixdp	26	slash	29
fraction	29	stickyper	29
load	32	strict	32
locale	31	strictarc	26
		tabalign	28
		tabalignexp	27

tabautofit	28	\pascalbase	4506
tabformat	27	\pb	20, 3858
tabnumalign	27	\pebi	20, 3868
tabtextalign	28	\per	12, 3034, 3840, 3851, 4010, 4243, 4350, 4410, 4548–4560, 4562–4570, 4572, 4574, 4576, 4578, 4580, 4582, 4584, 4586, 4588, 4590, 4592, 4594
tabunitalign	28	per (option)	29
textcelsius	30	\percent	15, 3807
textdegree	30	\peta	14, 3603
textminute	30	\pg	16, 3763
textmode	22	\pico	14, 3585, 3656, 3664, 3669, 3675, 3679, 3690, 3740, 3752, 3766, 3779, 3789, 3853, 3859, 4085, 4093, 4124, 4550, 4556
textmu	30	\picoampere	17, 3679
textOmega	30	\picobarn	20, 3852
textringA	30	\picofarad	18, 3684
textrm	23	\picogram	16, 3663
textsecond	30	\picoliter	4092
textsfc	23	\picom	16, 3779
texttt	23	\picometer	4085
tightpm	24	\picometre	16, 3656
trapambigerr	24	\picomole	17, 3668
trapambigfrac	29	\picosecond	16, 3673
unitcolor	31	\pl	4115
unitcolour	31	\pm	587, 887, 888, 1241, 3504
unitmathsrn	23	\pmol	17, 3751
unitmathssf	23	\pnt	4193
unitmathstt	23	\pow	4701
unitmode	22	\power	2982, 4243, 4701
unitsep	23	prefixbase (option)	29
unitspace	23	prefixproduct (option)	29
unittextrm	23	prefixsymbolic (option)	29
unittextsf	23	prespace (option)	30
unittexttt	23	\prime	797–800
valuecolor	31	\protected@xdef	1230
valuecolour	31	\providecommand	4366, 4414
valuemathrm	23	\providepower	19, 2645
valuemathsf	23	\provideprefix	19, 2631
valuemathsrn	23	\provideunit 18, 2617, 3634, 3758, 3759, 3764, 3789, 3847, 3858–3863, 4691
valuemathssf	23	\ps	16, 3786
valuemathstt	23		
valuemathtt	23		
valuemode	22		
valuesep	23		
valuetextrm	23		
valuetextsf	23		
valuetexttt	23		
xspace	30		

P			
\pA	17, 3737		
padangle (option)	26		
padnumber (option)	25		
\pamminute	4687		
\parsec	21, 3864		
\parsecond	4687		
\pascal	15, 3630, 3734, 4439		

R	
\rad	16, 3793, 3848
\radian	15, 3651
\radianbase	4506

<code>\raiseto</code>	12 , 3286	<code>\setMathOmega</code>	4024
<code>\raiseto@opt@si</code>	3288	<code>\setTextCelsius</code>	4024
<code>\reciprocal</code>	4243 , 4360 , 4412 , 4482 , 4484 , 4488 , 4490 – 4493 , 4495 , 4497 , 4502 , 4507 , 4509 , 4511 , 4517 , 4519 , 4524 , 4527 , 4528	<code>\setTextDegree</code>	4024
redefsymbols (option)	31	<code>\setTextmu</code>	4024
<code>\rem</code>	16 , 3793	<code>\setTextOmega</code>	4024
<code>\renewnosepunit</code>	4022	<code>\SI</code> 11 , 2612 , 2676 , 3937 , 3939 , 4314 , 4316	
<code>\renewpower</code>	19 , 2645 , 4293	<code>\si</code> 13 , 2456 , 2616 , 4000 , 4694 , 4697 , 4700	
<code>\renewprefix</code>	19 , 2631	<code>\si@addtocsnam</code>	81 , 3358
<code>\renewunit</code>		<code>\si@addtolist</code>	75 , 268 , 710 , 3460 , 3468 , 3555 , 3557 , 3560 , 3573
... 18 , 2617 , 3615 , 4023 , 4080 , 4305		<code>\si@addunitpowerfalse</code>	671
<code>\repeat</code>	1849	<code>\si@addunitpowertrue</code>	678
repeatunits (option)	25	<code>\si@ang@arc</code>	2039 , 2040
<code>\requiresiconfigs</code>		<code>\si@ang@arcdeg</code>	2114
.... 36 , 3432 , 3655 , 3739 , 3838 , 3846 , 3958 , 4159 , 4242 , 4546 , 4682		<code>\si@ang@arcfix</code>	2048 , 2054 , 2115
retainplus (option)	25	<code>\si@ang@arcmin</code>	2114
<code>\roentgen</code>	16 , 3793	<code>\si@ang@arcsec</code>	2114
<code>\rp</code>	4243 , 4504 , 4538	<code>\si@ang@arctodec</code>	2050 , 2079
<code>\rpcubed</code>	4252	<code>\si@ang@astrosign</code>	2191 , 2251
<code>\rpcubic</code>	4243 , 4514 , 4517 , 4522	<code>\si@ang@dec</code>	2018 , 2040
<code>\rperminute</code>	4328	<code>\si@ang@decimalsymbol</code> .	2189 , 2254
<code>\rpfourth</code>	4243 , 4534	<code>\si@ang@dectoarc</code>	2043 , 2114
<code>\rpsquare</code>		<code>\si@ang@degs</code>	2180
.. 4243 , 4483 , 4486 , 4496 , 4500 , 4508 , 4519 , 4522 , 4524 , 4531 , 4533		<code>\si@ang@fix</code>	2041 , 2048 , 2054 , 2080 , 2115 , 2237 , 2243
<code>\rpsquared</code> .	4252 , 4485 , 4495 , 4510 , 4512 , 4513 , 4527 , 4528 , 4531 , 4536	<code>\si@ang@fixdptrue</code>	2116
S			
<code>\Sec</code>	16 , 3786	<code>\si@ang@ifnum</code>	
<code>\second</code>	14 , 16 , 3273 , 3673 – 3678 , 3786 – 3792 , 4117 , 4416 , 4418 , 4420 , 4422 , 4426 , 4435 , 4438 – 4440 , 4484 , 4485 , 4488 , 4489 , 4495 , 4504 , 4509 – 4511 , 4513 – 4515 , 4517 , 4519 , 4522 , 4524 , 4527 , 4528 , 4531 , 4536 , 4538 , 4562 – 4564 , 4706	2068 , 2085 – 2087 , 2120 , 2199 , 2211	
<code>\sek</code>	4115	<code>\si@ang@killdegree</code>	2248
<code>\selectlanguage</code>	3339	<code>\si@ang@killminute</code>	2248
seperr (option)	24	<code>\si@ang@killsecond</code>	2248
sepfour (option)	24	<code>\si@ang@minnum</code>	2195
<code>\setfnumdsym</code>	4600	<code>\si@ang@mins</code>	2180
<code>\setfnumgsym</code>	4600	<code>\si@ang@movesign</code> ..	2190 , 2238 , 2244
<code>\setfnummsym</code>	4600	<code>\si@ang@notnum</code> ..	2111 – 2113 , 2159 , 2174
<code>\setMathCelsius</code>	4024	<code>\si@ang@num</code> ...	2195 , 2196 , 2228 , 2235
<code>\setMathDegree</code>	4024	<code>\si@ang@pad</code> ...	2206 , 2208 , 2220 , 2234
<code>\setMathmu</code>	4024	<code>\si@ang@padlargefalse</code>	470
		<code>\si@ang@padlargetrue</code>	
		477 , 482 , 487 , 492
		<code>\si@ang@padsmaillfalse</code>	469
		<code>\si@ang@padsmailltrue</code>	
		473 , 481 , 486 , 491
		<code>\si@ang@parse</code>	2011 , 2012
		<code>\si@ang@secnum</code>	2195
		<code>\si@ang@secs</code>	2180
		<code>\si@ang@sepint</code>	2125 , 2127 , 2160
		<code>\si@ang@signlessnum</code>	
		2216 , 2224 , 2226 , 2235
		<code>\si@ang@signtrue</code>	2201 , 2213

<code>\si@ang@sint</code>	<u>2160</u>	<code>\si@fam@italic</code>	<u>1152</u> , 3372
<code>\si@ang@strippt</code> ...	<u>2131</u> , <u>2136</u> , <u>2160</u>	<code>\si@fam@maths</code>	<u>1109</u> , 1162,
<code>\si@ang@toarcfalse</code>	498		1169, 1175, 1184, 1191, 1197, 3390
<code>\si@ang@toarctrue</code>	510, 514	<code>\si@fam@mode</code> ..	<u>1052</u> , 1210, 2007, 3943
<code>\si@ang@todectfalse</code>	499	<code>\si@fam@set</code>	<u>1066</u> , 3368
<code>\si@ang@todecttrue</code>	502, 506	<code>\si@fam@setbold</code>	1144, 1146, 1149, <u>1203</u>
<code>\si@ang@typeset</code>		<code>\si@fam@settrue</code>	1106
	2045, 2052, 2109, 2158, 2177, 2179	<code>\si@fam@sf</code>	<u>1021</u> , 1160
<code>\si@anglesep</code>	<u>288</u> , 2231	<code>\si@fam@text</code>	1109, 1164,
<code>\si@blockpkgs</code> ...	<u>46</u> , 3449, 3450, 3460		1171, 1177, 1186, 1193, 1199, 3372
<code>\si@catcodes</code>	14, 3584	<code>\si@fam@tt</code>	<u>1021</u> , 1167
<code>\si@checkpkgs</code> ...	<u>46</u> , 3463, 3464, 3468	<code>\si@fileprefix</code>	<u>3400</u> , 3405, 3408, 3411
<code>\si@closeerr</code>	<u>370</u> , 1366, 1406	<code>\si@fix@cdot</code>	<u>877</u>
<code>\si@closefrac</code>	<u>680</u> , 3202	<code>\si@fix@comma</code>	<u>877</u>
<code>\si@colour</code>	<u>1080</u> , 3371	<code>\si@fix@fullstop</code>	<u>877</u>
<code>\si@colourcmd</code>	<u>1080</u> , 3371	<code>\si@fix@med</code>	<u>872</u>
<code>\si@colournegfalse</code>	763	<code>\si@fix@medium</code>	<u>872</u>
<code>\si@colournegtrue</code>	766	<code>\si@fix@minus</code>	<u>885</u>
<code>\si@colourunitsfalse</code> .	725, 740, 749	<code>\si@fix@mp</code>	<u>885</u>
<code>\si@colourunitstrue</code> ..	728, 743, 752	<code>\si@fix@none</code>	<u>893</u>
<code>\si@colourvaluesfalse</code>	732, 739, 748	<code>\si@fix@period</code>	<u>877</u>
<code>\si@colourvaluetrue</code> .	735, 744, 753	<code>\si@fix@plus</code>	<u>885</u>
<code>\si@debugfalse</code>	224	<code>\si@fix@pm</code>	463, <u>885</u>
<code>\si@debugtrue</code>	237, 241	<code>\si@fix@slash</code>	<u>892</u>
<code>\si@decimalsymbol</code>		<code>\si@fix@space</code>	<u>872</u>
	286, 1698, 1825, 2189, 2252,	<code>\si@fix@stop</code>	<u>877</u>
	2503, 2506, 2537, 2552, 2584, 4193	<code>\si@fix@ten</code>	<u>890</u>
<code>\si@digitsep</code> ...	<u>283</u> , 1967, 1984, 2609	<code>\si@fix@thick</code>	<u>872</u>
<code>\si@emclash</code>	<u>3443</u> , 3921, 3923, 3955, 3957	<code>\si@fix@thin</code>	<u>872</u>
<code>\si@emulate</code>	<u>267</u> , 3542, 3543	<code>\si@fix@tightcdot</code>	<u>877</u>
<code>\si@emulating</code>	<u>3447</u> , 3919,	<code>\si@fix@tightpm</code>	461, <u>885</u>
	3953, 4167, 4235, 4545, 4597, 4626	<code>\si@fix@tighttimes</code>	<u>877</u>
<code>\si@errspace</code>	<u>273</u> , 1365	<code>\si@fix@times</code>	<u>877</u>
<code>\si@eVcorra</code>	<u>816</u> , 3830	<code>\si@fix@two</code>	<u>890</u>
<code>\si@eVcorrb</code>	<u>816</u> , 3831	<code>\si@fixdpfalse</code>	2014, 2063, 2066
<code>\si@eVspacea</code>	<u>3828</u>	<code>\si@fixdptrue</code> ..	635, 2061, 2084, 2479
<code>\si@eVspaceb</code>	<u>3828</u>	<code>\si@frac</code>	656, 684, 687, 691, 695, 699,
<code>\si@expanddefault</code>	<u>3546</u> , 3578		703, 707, <u>928</u> , 3175, 3179, 3188, 3949
<code>\si@expbase</code>	<u>375</u> , 1394, 2568, 3156	<code>\si@fracfalse</code>	651
<code>\si@expproduct</code>	<u>375</u> , 1390, 2568	<code>\si@fractrue</code>	654, 660, 664
<code>\si@extension</code> .	<u>3400</u> , 3405, 3408, 3411	<code>\si@frc@displen</code> ..	<u>954</u> , 966, 974, 978
<code>\si@fam@bold</code>	<u>1134</u> , 1205, 3372	<code>\si@frc@frac</code>	684, <u>928</u> , 1010
<code>\si@fam@boldify</code>	<u>1203</u>	<code>\si@frc@hook</code>	<u>928</u> , 4652, 4663, 4676, 4680
<code>\si@fam@colourcmd</code>		<code>\si@frc@mathsnf</code>	960, 964
	1060, 1090, 1101, 1308, 2495	<code>\si@frc@nice</code>	687, 695, <u>928</u>
<code>\si@fam@detmaths</code>	1122, <u>1159</u>	<code>\si@frc@nicefrac</code>	939, <u>954</u>
<code>\si@fam@detttext</code>	1118, 1124, 1129, <u>1159</u>	<code>\si@frc@sfrac</code>	699, 703, <u>928</u>
<code>\si@fam@ifbinline</code>	<u>1043</u> , 1146	<code>\si@frc@slash</code>	656, <u>928</u> , 1018
<code>\si@fam@ifbmaths</code> ..	<u>1036</u> , 1047, 1140		
<code>\si@fam@ifbtext</code>	<u>1036</u> , 1045, 1142, 1149		
<code>\si@fam@ifitext</code>	<u>1049</u> , 1155		

<code>\si@frc@ssuplen</code>	<code>\si@log@inf</code>
.. 954, 972, 974–976, 984, 999, 1001	127, 948, 1026,
<code>\si@frc@suplen</code> ... 954, 970, 976, 982	1033, 1497, 1507, 1511, 2625,
<code>\si@frc@textlen</code>	2639, 2653, 3309, 3321, 3581,
. 954, 968, 975, 980, 998, 1001, 1002	4070, 4077, 4162, 4163, 4541, 4542
<code>\si@frc@textnf</code>	<code>\si@log@warn</code>
962, 996	127, 198,
<code>\si@frc@ugly</code>	222, 952, 1293, 1300, 2175, 2918,
691, 707, 1007	3324, 3340, 3355, 4260, 4296, 4308
<code>\si@gensymbtrue</code>	<code>\si@logminfalse</code>
4138	225
<code>\si@gobblethree</code> ... 2810, 2819, 2829	<code>\si@logmintrue</code>
<code>\si@ifdefinable</code>	233
..... 71, 2618, 2622, 2628,	<code>\si@lognonefalse</code>
2632, 2636, 2642, 2646, 2650, 2656	226
<code>\si@ifl@aded</code>	<code>\si@lognonetrue</code>
3402	229
<code>\si@ifloaded</code>	<code>\si@mathscelsius</code>
3402, 3407,	801
3916, 3920, 3922, 3954, 3956, 4627	<code>\si@mathsdegree</code>
<code>\si@ifmtarg</code>	790, 806, 914
87, 2015, 2687, 3240	<code>\si@mathsminute</code>
<code>\si@ifnotmtarg</code> 87, 593, 1232, 2197,	790
2209, 2221, 2665, 2670, 2673,	<code>\si@mathsmu</code>
2675, 2686, 2793, 2887, 2917, 3947	774, 4326
<code>\si@inlinebtextfalse</code>	<code>\si@mathsOmega</code>
326	770, 926
<code>\si@inlinebtexttrue</code>	<code>\si@mathsringA</code>
329	807, 920
<code>\si@InputIfFileExists</code> . 3408, 3415	<code>\si@mathsrm</code> ... 1080, 1110, 1176, 1198
<code>\si@load</code>	<code>\si@mathssecond</code>
709	790
<code>\si@loademfile</code>	<code>\si@mathssf</code>
3434, 3544	1080, 1163, 1185
<code>\si@loadfile</code>	<code>\si@mathstt</code>
.. 720, 3305, 3406, 3433, 3442, 3576	1080, 1170, 1192
<code>\si@loc@load</code> ... 821, 3301, 3328, 3345	<code>\si@negcolour</code>
<code>\si@loc@ltol</code>	760, 1315
823, 3327	<code>\si@newcmd</code>
<code>\si@loc@set</code> 822, 3307, 3336, 3351	95
<code>\si@loc@sisetup</code>	<code>\si@newcommand</code>
3301	95
<code>\si@locale</code>	<code>\si@newrobustcmd</code>
820	95, 1207, 2004, 2612, 2616, 3036,
<code>\si@loctolang</code>	3935, 3941, 4001, 4011, 4178, 4212
820	<code>\si@noload</code>
<code>\si@log@debug</code>	709
139, 187, 194,	<code>\si@num</code>
197, 201, 204, 212, 214, 221, 912,	1213, 1216, 1990,
925, 1114, 1117, 1120, 1128, 1132,	1994, 2239, 2246, 2482, 4185, 4191
1136, 1154, 1157, 1161, 1168,	<code>\si@num@addexp</code>
1174, 1183, 1190, 1196, 1204,	1444, 1455
1212, 1277, 1431, 1439, 1460,	<code>\si@num@addmnt</code>
1493, 1503, 1516, 1525, 1556,	1446, 1455
1611, 1636, 1642, 1663, 1709,	<code>\si@num@addmntexp</code> . 1456, 1458, 1459
1848, 1856, 2008, 2016, 2019,	<code>\si@num@addpostzero</code> ... 1584, 1637
2198, 2210, 2222, 2296, 2305,	<code>\si@num@addprezero</code> 1576, 1637
2348, 2365, 2367, 2370, 2389,	<code>\si@num@addpzero</code> .. 1638, 1640, 1641
2425, 2500, 2528, 2671, 2676,	<code>\si@num@addsign</code>
2729, 2735, 2791, 2823, 2845,	1478, 1532
2866, 2907, 2996, 3021, 3997, 3998	<code>\si@num@addtmp</code> 1793, 1794, 1795
<code>\si@log@err</code> .. 119, 578, 632, 1235,	<code>\si@num@addtmpa</code>
1379, 1434, 1606, 1616, 1765,	1787, 1793
1774, 1779, 2027, 2032, 2300,	<code>\si@num@addtmpb</code>
2620, 2623, 2634, 2637, 2648,	1785, 1793
2651, 3168, 3410, 3436, 3444, 4628	<code>\si@num@addunit</code> ... 1988, 1992, 1997
	<code>\si@num@ambig</code> 1262, 1407–1409
	<code>\si@num@ambigertrue</code> .. 1327, 2681
	<code>\si@num@arg</code>
	551, 1274, 1286, 1293, 1300, 1379,
	1431, 1440, 1460, 1494, 1504,
	1517, 1525, 1557, 1606, 1612,
	1617, 1636, 1642, 1766, 1775, 1780
	<code>\si@num@assign</code>
	1532
	<code>\si@num@checkerr</code>
	1682, 1722
	<code>\si@num@cntdgt</code>
	1798

\si@num@cntdigits	\si@num@nosign	1644
. 1745, 1747, 1798, 1815, 1827, 1836	\si@num@nozero	1591, 1667
\si@num@dec	\si@num@out 1262, 1330, 1399, 2501,	
\si@num@decfmt	2508, 2512, 2576, 2578, 2581, 2585	
\si@num@decimalhook 1670, 1696, 2276	\si@num@pad	1844, 1847
\si@num@delplusfalse	\si@num@padleadfalse	387
\si@num@delplustrue	\si@num@padleadtrue	
\si@num@digits	391, 395, 407, 412, 417	
\si@num@dp .. 627, 1840, 1843, 1848,	\si@num@padtrailfalse	388
1849, 1856, 1912–1914, 1916,	\si@num@padtrailtrue	
1917, 2081, 2082, 2117, 2118, 2478	399, 403, 408, 413, 418	
\si@num@errr 1262, 1283, 1342,	\si@num@pd	1847
1352, 1353, 1366, 1367, 1401,	\si@num@pm	587, 1241
1687, 1733, 1741, 1811, 1824, 1990	\si@num@post	1621, 1629
\si@num@erropenfalse	\si@num@postdec 591, 602, 603, 1336,	
\si@num@erropenttrue	1339, 1566, 1570, 1579, 1594,	
\si@num@exp	1655, 1674, 1684, 1695, 1699,	
1279, 1291, 1299, 1318, 1320,	1700, 1703, 1724, 1747, 1836,	
1325, 1351, 1354, 1356, 1377, 1396	1837, 1852, 1857, 1860, 1862,	
\si@num@expsign	1926, 1971, 1977, 1980, 1983, 1984	
1280, 1292, 1295, 1354, 1357, 1395	\si@num@posterr	1727, 1735
\si@num@extra	\si@num@postrnd ... 1853, 1904–1906	
\si@num@fiint	\si@num@pre	1623, 1629
\si@num@finddigits 1563, 1568	\si@num@predec	
\si@num@finderr	590, 599, 600, 1335, 1566, 1569,	
\si@num@findsign	1574, 1589, 1593, 1655, 1668,	
\si@num@findxpart 1286, 1416	1669, 1674, 1684, 1692, 1693,	
\si@num@five	1703, 1858, 1859, 1861, 1894,	
\si@num@fixdp	1897, 1932, 1938, 1964, 1966, 1967	
\si@num@fixpm	\si@num@preerr	1725, 1729
\si@num@format	\si@num@prepost ... 1630, 1632, 1633	
\si@num@gensign	\si@num@prernd 1853, 1882–1884, 1895	
\si@num@ifextra 1675, 1692, 1699, 1704	\si@num@procerr	1402, 1987
\si@num@iffive 1936, 1940, 1975	\si@num@procnun ... 1316, 1317, 1558	
\si@num@ifvalid ... 1233, 1246, 2074	\si@num@psterr	1735
\si@num@in 1262, 1278, 1288, 1422	\si@num@rev	1865
\si@num@int	\si@num@reverse ... 1857, 1858, 1865	
\si@num@intabfalse	\si@num@rnd	1864, 1874
\si@num@intabtrue	\si@num@rndpost	1874
\si@num@intfmt 1934, 1937, 1948	\si@num@rndpre	1874
\si@num@intsep 1950, 1953, 1956, 1963	\si@num@round	1841, 1853
\si@num@killsign	\si@num@sepdigits . 1597, 1600, 1671	
\si@num@largeerr	\si@num@seperr 1730, 1736, 1753	
\si@num@lerr	\si@num@sepmantexp . 557, 1288, 1427	
\si@num@mant 553, 1262, 1281,	\si@num@sepsign	
1297, 1321, 1322, 1331, 1378, 1386	558, 566, 1289, 1290, 1463	
\si@num@mantexp	\si@num@sepxpart .. 1414, 1991, 2299	
\si@num@mantsign 559, 1262,	\si@num@serr	1808
1282, 1298, 1301, 1306, 1331, 1335	\si@num@sign	
\si@num@movedigit	1474, 1484, 1486, 1531, 1555	
\si@num@mp		

\si@num@signexpfalse	426	\si@opt@compatkey	
\si@num@signexptrue	210, 824–830, 832–871
.....	437, 441, 446, 451, 456	\si@opt@disablekey	215,
\si@num@signmantfalse	425	218, 259, 264, 270, 712, 715, 813
\si@num@signmanttrue		\si@opt@key	185,
.....	429, 433, 445, 450, 455	222, 268, 347–358, 361–363, 540,
\si@num@smallerr	1751, 1808	629, 710, 719, 756–759, 769, 771,
\si@num@unsign	1588, 1644	775, 792–794, 802, 808, 818–820, 823
\si@num@valid	1246	\si@opt@proctform	574, 575, 589
\si@num@value	1475, 1485, 1486	\si@opt@xchoicekey	
\si@num@xpart	199, 273, 276, 278, 280, 283,
.....	1262, 1284, 1413, 1420, 1996	286, 288, 375, 377, 379, 381, 422, 666
\si@numaddn	364, 367, 1673	\si@out	936, 3153, 3217, 3359, 3398, 4215
\si@numcloseerr	364, 367, 1771	\si@out@maths	3376, 3381
\si@numdecimal		\si@out@minus	3385, 3393
...	364, 369, 1318, 1604, 1608, 1812	\si@out@num	1213, 1405, 1408, 2230,
\si@numdigits	364, 369, 1382	2510, 2512, 2532, 2542, 2575,
\si@numexp	364, 368, 1353, 1432, 1436	2578, 2584, 2594, 3395, 4185, 4191
\si@numextra	366, 1673, 1717	\si@out@numtrue	3397
\si@numgobble	364, 368, 1430	\si@out@sp	3382, 3383, 3392
\si@numopenerr	364, 367, 1763	\si@out@text	3374, 3381
\si@numprod	364, 369, 1417, 2709	\si@outerinput	3419, 3423
\si@numsign		\si@outerinputfalse	3426
...	364, 368, 1489, 1490, 1615, 1618	\si@outerinputtrue	3414
\si@numtextmodefalse ..	295, 303, 311	\si@packagecheck	46
\si@numtextmodetrue ..	299, 307, 314	\si@per	3034, 4243, 4245, 4248,
\si@numvalid	366, 1237, 1257, 2070, 2346	4249, 4251, 4640, 4653, 4664,
\si@obeyboldfalse	334	4673, 4677, 4681, 4694, 4697, 4700
\si@obeyboldtrue	340	\si@pm	458, 1495, 1989
\si@obeyfamilyfalse	337	\si@prefixbase	381, 3156
\si@obeyfamilytrue	343	\si@prefixproduct	379, 3153
\si@obeyitalicfalse	335	\si@prefixsymbolicfalse ..	4452
\si@obeyitalictrue	341	\si@repeatunitsfalse ...	670, 1347
\si@obeymodefalse	336	\si@repeatunitstrue	674
\si@obeymodetrue	342	\si@seperrfalse	1418, 2298
\si@old@OHMfalse	4072	\si@SI	2614–2616, 2661
\si@openerr	370, 1346, 1366	\si@sign	420, 1555, 1556
\si@openfrac	680, 3201	\si@sis@addtolocale	4220
\si@opt@boolkey		\si@sis@num	4177
...	192, 243, 244, 291, 322, 323,	\si@sis@numstar	4177
.....	331, 332, 370, 371, 373, 374, 378,	\si@sis@savefont ..	4203, 4206–4211
.....	382, 458, 516, 517, 542, 627, 636–	\si@siu@newprefix	
.....	638, 644, 647, 680, 721, 722, 760, 811	4446, 4453–4472, 4474–4479
\si@opt@choicekey	195,	\si@siu@newunit	4335,
.....	223, 294, 302, 310, 316, 325, 333,	4367–4407, 4703, 4704, 4707–4710
.....	385, 423, 467, 496, 519, 605, 616,	\si@siu@newunithook ...	4335, 4633
.....	648, 669, 682, 723, 730, 737, 746, 761	\si@siu@newunitx	4408, 4415, 4417,
\si@opt@cmdkey	188, 549	4419, 4421, 4423–4425, 4427,
\si@opt@cmdkeys	190,	4429, 4431, 4432, 4434, 4437, 4705
.....	345, 346, 359, 360, 364, 372, 681,	\si@siu@newunitxhook ..	4408, 4633
.....	709, 755, 768, 770, 774, 790, 801, 807	\si@siu@power ..	4335, 4641, 4654, 4665

<code>\si@siu@setup</code>	<u>4257</u>	<code>\si@tab@expprecnt</code>	
<code>\si@siu@unity</code>	<u>4436</u>	<u>543</u> , 2544, 2545, 2564
<code>\si@slash</code>	<u>666</u> , 936	<code>\si@tab@expsignfalse</code>	555
<code>\si@slashfalse</code>	650	<code>\si@tab@expsigntrue</code>	569, 572
<code>\si@slashtrue</code>	655	<code>\si@tab@exptest</code>	<u>2384</u> , <u>2386</u>
<code>\si@str@chrstr</code>	<u>145</u>	<code>\si@tab@fixed</code>	2490, <u>2527</u>
<code>\si@str@ifchrstr</code>		<code>\si@tab@fixedfalse</code>	534, 538
.....	<u>145</u> , 179, 1257, 1417,	<code>\si@tab@fixedtrue</code>	521
	1430, 1432, 1489, 1490, 1604,	<code>\si@tab@format</code>	<u>2483</u> , <u>2488</u>
	1615, 1717, 1763, 1771, 2346, 2709	<code>\si@tab@gettok</code>	<u>2295</u> , 2318
<code>\si@str@ifonlychrs</code>		<code>\si@tab@gettok@S</code> ..	2297, <u>2320</u> , 2392
...	<u>166</u> , 630, 1318, 1655, 1668, 1685	<code>\si@tab@gettok@s</code>	2306, <u>2401</u>
<code>\si@str@onlychrs</code>	<u>166</u>	<code>\si@tab@ifonechar</code>	2352, <u>2373</u>
<code>\si@switchfalse</code>		<code>\si@tab@lfill</code>	<u>2435</u>
.	147, 180, 556, 592, 1261, 1285,	<code>\si@tab@lfill@S</code>	518, 2437
	1287, 1571, 1656, 1706, 1754,	<code>\si@tab@lfill@s</code>	<u>617</u> , 2454
	1863, 1889, 1911, 2317, 3566, 4223	<code>\si@tab@lfill@t</code>	<u>606</u> , 2441
<code>\si@switchtrue</code> 160, 168, 1248, 1418,		<code>\si@tab@macro</code>	<u>2388</u> , <u>2399</u>
	1442, 1613, 1654, 1717, 1769,	<code>\si@tab@mantpostcnt</code>	
	1892, 1919, 1924, 2347, 3570, 4225	<u>543</u> , 576, 2478, 2536, 2538
<code>\si@sym@celsius</code>		<code>\si@tab@mantprecnt</code> ..	<u>543</u> , 577, 2534
.....	<u>902</u> , 3645, 3648, 4154, 4217	<code>\si@tab@mantsignfalse</code>	554
<code>\si@sym@degree</code>	<u>902</u> , 2181,	<code>\si@tab@mantsigntrue</code>	561, 564
	2208, 2220, 2229, 2248, 3250,	<code>\si@tab@meaning</code>	<u>2386</u> , 2400
	3262, 3814, 3817, 4156, 4218, 4219	<code>\si@tab@midbox</code>	
<code>\si@sym@minute</code>	<u>902</u> ,	<u>2484</u> , 2497, 2502, 2505, 2530
	2182, 2206, 2218, 2249, 3820, 4328	<code>\si@tab@newline@S</code>	2322, <u>2460</u>
<code>\si@sym@mu</code>		<code>\si@tab@newline@s</code>	2403, <u>2460</u>
..	<u>902</u> , 3251, 3263, 3594, 3597, 4155	<code>\si@tab@next</code>	<u>2319</u> , 2322,
<code>\si@sym@Omega</code> ..	<u>902</u> , 3631, 3634, 4153		2325, 2328, 2331, 2334, 2337,
<code>\si@sym@ringA</code> ..	<u>902</u> , 3252, 3264, 3795		2339, 2363, 2392, 2403, 2406,
<code>\si@sym@second</code>			2409, 2412, 2415, 2418, 2420, 2434
..	<u>902</u> , 2183, 2204, 2250, 3821, 4329	<code>\si@tab@numout</code>	<u>2446</u> , <u>2474</u>
<code>\si@symbol</code>	894, 902–908	<code>\si@tab@onechar</code>	<u>2373</u>
<code>\si@tab@begin</code>	2303, 2307, <u>2311</u>	<code>\si@tab@othertok</code>	
<code>\si@tab@begin@S</code>	2285, <u>2295</u>	.	2341, 2344, 2354, <u>2364</u> , 2382, 2394
<code>\si@tab@begin@s</code>	2291, <u>2295</u>	<code>\si@tab@out</code> ...	<u>1262</u> , <u>1330</u> , 2510, 2575
<code>\si@tab@bracetest</code>	2353, <u>2380</u>	<code>\si@tab@postbox</code>	<u>2484</u> , 2497,
<code>\si@tab@end</code>	<u>2460</u>		2511, 2513–2515, 2517, 2577, 2583
<code>\si@tab@end@S</code>	2286,	<code>\si@tab@postdim</code>	
	2327, 2328, <u>2435</u> , 2461, 2463, 2464	.	2521, 2536, 2539, 2572, 2577, 2583
<code>\si@tab@end@s</code>	2292,	<code>\si@tab@posttoks</code>	
	2408, 2409, <u>2435</u> , 2467, 2469, 2470	<u>2308</u> , 2316, 2368, 2447
<code>\si@tab@expbox</code>		<code>\si@tab@prebox</code>	2484, 2497,
.	<u>2484</u> , 2498, 2520, 2531, 2589, 2593		2509, 2513, 2514, 2517, 2518, 2574
<code>\si@tab@expdim</code> .	<u>2521</u> , 2545, 2550,	<code>\si@tab@predim</code> <u>2521</u> , 2534, 2558, 2574	
	2553, 2562, 2569, 2572, 2589, 2593	<code>\si@tab@pretoks</code>	<u>2308</u> ,
<code>\si@tab@expout</code>			2315, 2371, 2390, 2391, 2445, 2449
.	<u>1262</u> , <u>1330</u> , 2508, 2581, 2587, 2594	<code>\si@tab@rfill</code>	<u>2435</u>
<code>\si@tab@exppostcnt</code>		<code>\si@tab@rfill@S</code>	<u>518</u> , 2438
.....	<u>543</u> , 2549, 2550, 2565	<code>\si@tab@rfill@s</code>	<u>617</u> , 2458

\si@tab@rfillt	606, 2440	231, 232, 235, 236, 239, 240, 258,
\si@tab@sepcorr		261, 297, 298, 305, 306, 312, 313,
.....	2535, 2541, 2546, 2554, 2598	318, 319, 327, 328, 338, 339, 389,
\si@tab@sp	2526, 2529, 2548, 2609	390, 393, 394, 397, 398, 401, 402,
\si@tab@strip	2386	405, 406, 410, 411, 415, 416, 427,
\si@tab@toks	2308, 2314,	428, 431, 432, 435, 436, 439, 440,
2350, 2351, 2423, 2424, 2456, 2482		443, 444, 448, 449, 453, 454, 471,
\si@tab@unfixed	2492, 2499	472, 475, 476, 479, 480, 484, 485,
\si@tabnumalign	518	489, 490, 500, 501, 504, 505, 508,
\si@tempa	37, 51, 52,	509, 512, 513, 524, 525, 528, 529,
54, 55, 57, 60, 61, 64, 65, 148, 159,		532, 533, 536, 537, 608, 609, 612,
169, 171, 196, 200, 228, 232, 236,		613, 619, 620, 623, 624, 652, 653,
240, 254, 259, 260, 298, 306, 313,		658, 659, 662, 663, 672, 673, 676,
319, 328, 339, 390, 394, 398, 402,		677, 685, 686, 689, 690, 693, 694,
406, 411, 416, 428, 432, 436, 440,		697, 698, 701, 702, 705, 706, 726,
444, 449, 454, 472, 476, 480, 485,		727, 733, 734, 741, 742, 750, 751,
490, 501, 505, 509, 513, 525, 529,		764, 765, 1012, 1013, 1108, 1189,
533, 537, 609, 613, 620, 624, 653,		1535, 1536, 1679, 1681, 1731,
659, 663, 673, 677, 686, 690, 694,		1733, 1737, 1741, 1745, 1756,
698, 702, 706, 720, 727, 734, 742,		1759, 1813, 1815, 1817, 1825,
751, 765, 918, 919, 1008, 1013,		1827, 1829, 1834, 2022, 2023,
1016, 1020, 1040, 1041, 1107,		2025, 2026, 2058, 2059, 2696,
1182, 1230, 1232–1234, 1305,		2700, 3082, 3084, 3094, 3095,
1306, 1332, 1337, 1340, 1364,		3098, 3099, 3113, 3114, 3329,
1370, 1372, 1375, 1388, 1389,		3330, 3337, 3338, 3346, 3347,
1393, 1394, 1534, 1536, 1674,		3352, 3353, 3433, 3451, 3453,
1675, 1684, 1685, 1755, 1760,		3455, 3460, 3465, 3467, 3550,
1821, 1825, 1833, 1866, 1868,		3554, 3567, 3569, 4337, 4340,
1870, 1881, 1885, 1903, 1907,		4349, 4355, 4358, 4364, 4448,
2021, 2023, 2026, 2057, 2059,		4452, 4636, 4639, 4648, 4651, 4662
2089, 2094, 2097, 2101, 2105,	\si@tempboxa	
2128, 2131, 2133, 2135–2137,	41, 1014, 1015, 1023, 1030, 2253,	
2140, 2141, 2146, 2152, 2162,	2255, 2257, 2258, 2260, 2262,	
2164, 2170, 2173, 2278, 2282,	2266, 2269, 2272, 2532, 2533,	
2284, 2287, 2290, 2293, 2353,	2537, 2539, 2542, 2543, 2551,	
2381, 2387, 2388, 2695, 2700,	2553, 2556, 2558, 2560, 2562,	
2926, 2928, 2936, 2938, 2954,	2567, 2569, 2608, 2611, 2719, 2720	
2957, 2974, 2977, 2989, 2993,	\si@tempboxb	41, 2256, 2264
3017, 3020, 3072, 3074, 3081,	\si@tempboxc	41, 2261, 2264
3084, 3093, 3095, 3099, 3112,	\si@tempboxd	41, 2263, 2267, 2270
3114, 3211, 3218, 3232, 3233,	\si@tempc	37, 158, 159, 1680,
3256, 3257, 3328, 3330, 3333,	1681, 1694, 1697, 1703, 3455–	
3338, 3345, 3347, 3348, 3353,	3457, 3466–3468, 3551, 3555,	
3449, 3451–3454, 3456, 3463,	3557, 3559, 3563, 3573, 3575,	
3466, 3543, 3544, 3552, 3554,	4638, 4645, 4650, 4658, 4661, 4669	
3557, 3564, 3569, 3573, 3575,	\si@tempcnta	1746, 1748,
3576, 3994, 4027, 4028, 4038,	1799, 1803, 1809, 1816, 1822,	
4039, 4049, 4050, 4060, 4061,	1828, 1838, 1840, 1843, 1849,	
4258, 4336, 4339, 4340, 4347,	1851, 1875, 1913, 1917, 1929,	
4354, 4356, 4363, 4447, 4451,	2472, 2564–2566, 2599, 2601,	
4634, 4644, 4646, 4657, 4659, 4668	2605, 2611, 2689, 2693–2695,	
\si@tempb	2706, 2710, 2963, 2967, 2969, 2970	
37, 149, 150, 170, 179, 227, 228,	\si@tempcntb	1746, 1748, 1809, 1816,
		1822, 1828, 1885, 1887, 1890,

1891, 1894, 1907, 1909, 1918,	\si@unt@countprefix ... 2947, <u>2953</u>
1922, 1923, 1926, <u>2472</u> , 2969, 2970	\si@unt@countx 2691, <u>2705</u>
\si@tempdima 2088, 2091,	\si@unt@defpower 2647, 2652, 2654, 2657, <u>2844</u>
2093, 2097, 2100, 2105, 2110,	\si@unt@defprefix 2633, 2638, 2640, 2643, <u>2820</u>
2121, 2123, 2126, 2128, 2132,	\si@unt@defunit 2619, 2624, 2626, 2629, <u>2790</u>
2133, 2161, 2165, 2255, 2265,	\si@unt@depthcnt 2738,
2266, 2273, <u>2521</u> , 2533, 2534,	2755, 2862, 2865, 2866, 2875, 2876
2536, 2543, 2545, 2550, 3029, 3030	\si@unt@final 2740, <u>2783</u> , 2877
\si@tempdimb 2260, 2265, 2269, 2273, <u>2521</u>	\si@unt@first 2881, <u>2886</u>
\si@temptoks 45, 84,	\si@unt@firstfalse 2898
86, 2142, 2144, 2147, 2149, 2153,	\si@unt@firstorsecond 2868, <u>2879</u> , 2943, 2980, 3051
2155, 2158, 2901, 2906, 2907,	\si@unt@firsttrue 2767
2914, 2915, 3207, 3212, 3229,	\si@unt@fracout 3134, <u>3163</u>
3230, 3232, 3238, 3239, 3311, 3315	\si@unt@fullstop 3208, <u>3221</u>
\si@textcelsius 801, 4047	\si@unt@holdspace 3100, <u>3111</u>
\si@textdegree .. <u>790</u> , 804, 913, 4058	\si@unt@holdstacka 2779, <u>3109</u>
\si@textminute <u>790</u>	\si@unt@holdstackb 2780, <u>3109</u>
\si@textmodefalse 1071, 1077	\si@unt@ifliteral . <u>2715</u> , 2728, 2869
\si@textmodetrue 1069, 1075	\si@unt@incnt 3096, <u>3105</u>
\si@textmu <u>774</u> , 916, 4036, 4326	\si@unt@init 2737, <u>2755</u> , 2863
\si@textOmega <u>770</u> , 917, 4025	\si@unt@invpower 2986, <u>3028</u>
\si@textringA 807, 922	\si@unt@invprefix <u>2953</u>
\si@textrm <u>1080</u> , 1112, 1178, 1200	\si@unt@lastadda 2781, <u>3069</u>
\si@textsecond <u>790</u>	\si@unt@lastaddb 2782, <u>3069</u>
\si@textsf <u>1080</u> , 1165, 1187	\si@unt@litoutfalse 2765
\si@texttt <u>1080</u> , 1172, 1194	\si@unt@litouttrue 2732, 3132, 4019
\si@tothe 3286, 3287, <u>3288</u>	\si@unt@litpower .. 2854, <u>2971</u> , 3293
\si@unitcolour <u>721</u> , 1103	\si@unt@litprefixfalse 2766
\si@unitmathsrm <u>345</u> , 1094	\si@unt@litprefixtrue 2821
\si@unitmathssf <u>345</u> , 1095	\si@unt@littesttrue 2718
\si@unitmathstt <u>345</u> , 1096	\si@unt@litvalsep <u>2741</u>
\si@unitsep <u>273</u> , 2921, 3209	\si@unt@nonlatin 3214, <u>3244</u>
\si@unitSIdef <u>4007</u>	\si@unt@noopt 2800, <u>2803</u>
\si@unitspace <u>273</u> , 3210	\si@unt@normout ... 3136, <u>3159</u> , 3184
\si@unittextmodefalse 296, 304, 317	\si@unt@notabg <u>3191</u>
\si@unittextmodetrue . 300, 308, 320	\si@unt@notambig 3164, <u>3191</u>
\si@unittextrm 359, 1097	\si@unt@numfalse 639, 2668
\si@unittextsf 359, 1098	\si@unt@numtrue 641, 1998, 2685, 2889
\si@unittexttt 359, 1099	\si@unt@opt 2805–2807, <u>2808</u>
\si@unt@addprefix 2945, <u>2950</u>	\si@unt@out 930, 931, 935, 937, 939,
\si@unt@addstack 3102, <u>3111</u>	940, 943, 944, 2719, 2734, 3162, <u>3203</u>
\si@unt@addtostack 2870, 2921, 2951, 3023, <u>3071</u>	\si@unt@per <u>3034</u>
\si@unt@addvalsep <u>2741</u>	\si@unt@perfalse .. 2768, 3004, 3054
\si@unt@addvaluesep 2733, <u>2741</u> , 2897	\si@unt@perseenfalse .. 2769, 3005
\si@unt@checkstack 2928, 2938, 3019,	\si@unt@perseenttrue ... 2933, <u>4253</u>
3074, 3083, 3092, 3107, 3113,	\si@unt@pertrue ... 3057, <u>4253</u> , <u>4256</u>
3118, 3122, 3126–3128, 3130, <u>3138</u>	
\si@unt@cntx 2707, 2708, 2713	

<code>\si@unt@power</code>	2856, 2973, 3295	<code>\si@valuetexttt</code>	359, 1088
<code>\si@unt@powerdim</code>	2772,	<code>\si@xargdef</code>	95
	2983, 2996, 2999, 3008, 3016,	<code>\si@xifmtarg</code>	87
	3022, 3024, 3027, 3029–3032, 3067	<code>\si@xspacefalse</code>	2669, 4013
<code>\si@unt@prefix</code>	2839, 2942	<code>\SIdecimalsign</code>	4199
<code>\si@unt@prefixcnt</code>		<code>\SIdecimalsymbol</code>	4199
 2775, 2952, 2967, 3151, 3157	<code>\SIdefaultMfam</code>	4209
<code>\si@unt@prefixout</code>		<code>\SIdefaultNfam</code>	4209
 3150, 3160, 3174, 3177, 3186	<code>\SIdefaultTfam</code>	4209
<code>\si@unt@preplussp</code>	3077, 3080	<code>\siemens</code>	15, 3630, 3694
<code>\si@unt@prepowerfalse</code> .	2770, 3001	<code>\siemensbase</code>	4523
<code>\si@unt@prepowertrue</code>	2990	<code>\sievert</code>	15, 3630, 3736
<code>\si@unt@printunit</code>		<code>\sievertbase</code>	4523
	. 2001, 2672, 2688, 2698, 2702, 2726	<code>\SIfig</code>	2664, 3763
<code>\si@unt@recip</code>	3060	<code>sign (option)</code>	25
<code>\si@unt@reciptest</code>	2940, 3060	<code>\SIGroupfourfalse</code>	4194
<code>\si@unt@second</code>	2883, 2916	<code>\SIGroupfourtrue</code>	4194
<code>\si@unt@setopts</code>	2892, 2899	<code>\SImathrm</code>	4206
<code>\si@unt@setSIopts</code>	2899	<code>\SImathsf</code>	4206
<code>\si@unt@SIopts</code>	2661, 2910, 2914	<code>\SImathhtt</code>	4206
<code>\si@unt@spacecheck</code>	2930, 2935	<code>\SIobeyboldfalse</code>	4175
<code>\si@unt@spaceout</code>		<code>\SIobeyboldtrue</code>	4175
 3148, 3161, 3178, 3187	<code>\SIproductsign</code>	4199
<code>\si@unt@spstack</code> ...	2752, 2755, 3149	<code>\SIsetup</code>	4257
<code>\si@unt@stack</code>	3087, 3090	<code>\sisetup</code>	4, 21, 184, 213,
<code>\si@unt@stacka</code> 2755,	3162, 3165, 3188		246, 347–358, 361–363, 540, 585,
<code>\si@unt@stackb</code>	2755, 3115,		756–759, 769, 771, 775, 792–794,
	3166, 3175, 3179, 3183, 3188, 3201		802, 808, 1055, 1057, 1209, 2006,
<code>\si@unt@stackout</code>	2785, 3131		2045, 2108, 2245, 2313, 2666,
<code>\si@unt@stackpower</code>	2994, 3000, 3068		2906, 2915, 2958, 3302, 3303,
<code>\si@unt@stackvalsep</code> ...	2741, 3155		3306, 3315, 3470, 3882, 3891,
<code>\si@unt@stkpower</code> ..	2872, 2924, 3000		3900, 3909, 3924, 3927, 3930,
<code>\si@unt@stkpwr</code>	3000		3933, 3945, 3961, 3963, 3966,
<code>\si@unt@stp</code>	3221		3969, 3972, 3975, 3978, 3981,
<code>\si@unt@sym</code> 3250–3252, 3262–3264,	3267		3984, 3987, 3990, 4009, 4020,
<code>\si@unt@third</code>	2784, 2916		4034, 4045, 4056, 4067, 4168,
<code>\si@unt@unit</code>	2815, 2860		4175, 4176, 4184, 4189, 4190,
<code>\si@unt@unitarg</code> 1999, 2001, 2660,	2674		4194–4202, 4205, 4214, 4220,
<code>\si@unt@unitcnta</code>	2755		4221, 4236, 4243, 4245, 4248,
<code>\si@unt@unitcntb</code>	2755, 3192		4249, 4251, 4253, 4256, 4265,
<code>\si@unt@unithook</code> ..	2860, 2890, 4019		4268, 4271, 4274, 4277, 4280,
<code>\si@unt@withopt</code>	2798, 2803		4283, 4319, 4322, 4598, 4600–
<code>\si@valuecolour</code>	721, 1092, 2495		4602, 4604, 4607, 4610, 4613,
<code>\si@valuemathsrms</code>	345, 1083		4616, 4619, 4622, 4639, 4651,
<code>\si@valuemathssf</code>	345, 1084		4662, 4672, 4675, 4679, 4683, 4685
<code>\si@valuemathstt</code>	345, 1085	<code>\SIstyle</code>	4220
<code>\si@valuesep</code>	273,	<code>\SIstyleToLang</code>	4220
	1014, 2752, 2754, 3948, 3993, 4239	<code>\SIthousandsep</code>	4199
<code>\si@valuetextrm</code>	359, 1086	<code>\SIunitdot</code>	4196
<code>\si@valuetextsf</code>	359, 1087	<code>\SIunitsep</code>	4196

valuemathsrms (option)	23		
valuemathssf (option)	23		
valuemathstt (option)	23		
valuemathhtt (option)	23		
valuemode (option)	22		
valuesep (option)	23		
valuetextrm (option)	23		
valuetextsf (option)	23		
valuetexttt (option)	23		
\volt	15, 3630, 3684, 3685,		
	3756, 3757, 4111, 4112, 4491–4493		
\voltbase	4506		
W			
\watt	15, 3630, 3686–		
	3688, 3726, 3778, 4427, 4431, 4490		
\wattbase	4506		
\weber	15, 3630, 4496, 4497		
\weberbase	4523		
X			
\XeTeXrevision	3245		
xspace (option)	30		
Y			
\yb	20, 3858		
\yobi	3868		
\yocto .	14, 3585, 3857, 3863, 4554, 4560		
\yoctobarn	20, 3852		
\yotta	14, 3603		
Z			
\zb	20, 3858		
\zebi	3868		
\zepto .	14, 3585, 3856, 3862, 4553, 4559		
\zeptobarn	20, 3852		
\zetta	14, 3603		

28 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/.
- [7] Heiko Bauke. fancyunits, 2007. http://www.mpi-hd.mpg.de/personalhomes/bauke/LaTeX/Tips_und_Tricks/fancyunits/index.php.