

siunitx — A comprehensive (SI) units package*

Joseph Wright[†]

Released 2008/06/23

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
	7	Units and values	11
I Introduction	4	7.1 Literal units	12
		7.2 The unit interpreter . .	12
		7.3 Powers of units	13
		7.4 Units with no values . .	13
II Using the siunitx package	5	7.5 Free-standing units . . .	14
1 For the impatient	5	7.6 Pre-defined units, prefixes and powers	14
2 Requirements	6	7.7 Prefixed and abbreviated units	16
3 Loading the package	6	7.8 Defining new units . . .	19
4 Numbers	6	8 Specialist units	20
5 Tabular material	8	8.1 Binary units (binary) .	20
5.1 Aligning numbers	8	8.2 Synthetic chemistry (synchem)	20
5.2 Columns of units	10	8.3 High-energy physics (hep)	21

*This file describes version v1.ob, last revised 2008/06/23.

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

8.4	Astronomy (<code>astro</code>) . . .	21	16	Feature requests	42
9	Font control	22	17	Acknowledgements	42
10	Package options	22	III	Correct application of (SI) units	44
10.1	Font family and style .	23	18	Background	44
10.2	Spacing and separators	24	19	Units	44
10.3	Number formatting . .	24	19.1	SI base units	44
10.4	Angle formatting	27	19.2	SI derived units	44
10.5	Tabular material	27	19.3	SI prefixes	45
10.6	Units	29	19.4	Other units	45
10.7	Symbols	31	20	Units and values in print	46
10.8	Colour	32	20.1	Mathematical meaning .	46
10.9	International support .	32	20.2	Unit multiplication and division	46
10.10	Package control	32	20.3	Repeating units	47
10.11	Back-compatibility options	33	20.4	Clarity in writing values of quantities	47
10.12	Summary of all options	33	20.5	Graphs and tables . . .	47
11	Emulation of other packages	36	IV	Implementation	50
12	Configuration files	37	21	Main package	50
13	Common questions	37	21.1	Setup code	50
13.1	Why do I need <code>\per</code> more than once?	37	21.2	Logging	53
13.2	Why is the order of my units changed?	38	21.3	String comparison . . .	54
13.3	Why are compound units not recommended outside of <code>\SI/\si</code> ? . .	38	21.4	Option handling	55
13.4	How do I set superscripts to use lining numbers?	38	21.5	Compatibility options .	72
13.5	Why do most of the examples use $\text{J mol}^{-1} \text{K}^{-1}$?	38	21.6	Constants	74
13.6	What can <code>numprint</code> do that <code>siunitx</code> cannot? . . .	39	21.7	Symbols	75
14	Tricks and known issues	39	21.8	Handling fractions . . .	75
14.1	Ensuring maths mode .	39	21.9	Font control	78
14.2	Using <code>.</code> in units	39	21.10	Formatting numbers . .	82
14.3	Limitations of <code>\mathrm</code>	39	21.11	Formatting angles . . .	103
14.4	Entire document in sans serif font	40	21.12	Tabular material	110
14.5	Effects of emulation . .	40	21.13	Units	118
14.6	Centring columns on non-decimal input . . .	40	21.14	Locales	135
14.7	Adding items after the last column of a tabular .	41	21.15	Output routine	137
15	Reporting a problem	42	21.16	Finalisation	138
			22	Loadable modules	142
			22.1	Multiple prefixes	142
			22.2	Derived units with specific names	143
			22.3	Units with prefixes . . .	144

22.4	Abbreviated units . . .	146	24.4	South Africa	151
22.5	Additional (temporary) SI units	147	25	Emulation code	151
22.6	Units accepted for use with SI	148	25.1	units	152
22.7	Units based on physical measurements	148	25.2	unitsdef	153
23	Additional configurations	149	25.3	Slstyle	158
23.1	Synthetic chemistry . .	149	25.4	Slunits	160
23.2	High-energy physics . .	149	25.5	hepunits	167
23.3	Astronomy	150	25.6	fancynum	168
23.4	Binary units	150	25.7	fancyunits	169
24	Loadable locales	150	V	Notes	172
24.1	United Kingdom	150	26	Change History	172
24.2	United States	151	27	Index	172
24.3	Germany	151	28	References	190

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.

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.

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 0.12345	<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>	

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]:

$m_e = 9.1093897(54) \times 10^{-31} \text{ kg}$	<code>\m_{\mathrm{e}}</code>
$m_e = (9.1093897 \pm 0.0000054) \times 10^{-31} \text{ kg}$	<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 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}</code> <code>\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 1 234	<code>\num{1234}</code>	<code>\num[sepfour=true]{1234}\</code>
12345 12,345	<code>\num{12345}</code>	<code>\num[digitsep=comma]{12345}\</code>
12345	<code>\num[digitsep=none]{12345}</code>	

The formatting of exponents is also customisable.

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

⁶This is common in chemical crystallography, for example.

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

-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>

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 \\
  \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

```

56.7835 & 56.7835 & 56.7835 & 56.7835 \\
90.473 & 90.473 & 90.473 & 90.473 \\
\bottomrule
\end{tabular}
\end{table}

```

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,tabsignexp=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

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

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.

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}
```

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.

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

$0^\circ 0' 1''$	<code>\ang{;;1} \ang[padding=none]{;;1}\</code>
$2^\circ 2' 0''$	<code>\ang{2;;} \ang[padding=all]{2;;}\</code>
$0^\circ 3' 0''$	<code>\ssetup{padding=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°	<code>\ang{1.05} \ang[decimalsymbol=comma]{1.05}\</code>
3.67890°	<code>\ang{3.67890} \ang[digitsep=comma]{3.67890}\</code>
$9^\circ 8' 7''$	<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°	<code>\ang{1.2} \ang[astroang]{1.2}\</code>
$1^\circ 2.3'$	<code>\ang{1;2.3;} \ang[astroang]{1;2.3;}\</code>
$1^\circ 2' 3.4''$	<code>\ang{1;2;3.4} \ang[astroang]{1;2;3.4}</code>

7 Units and values

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

$1.23 \text{ J mol}^{-1} \text{ K}^{-1}$	<code>\SI[mode=text]{1.23}{J.mol^{-1}.K^{-1}}\</code>
$0.23 \times 10^7 \text{ cd}$	<code>\SI{.23e7}{\candela}\</code>
$\pounds 1.99/\text{kg}$	<code>\SI[per=slash]{1.99}{\pounds\per\kilogram}\</code>
70 m s^{-1}	<code>\SI{70}{\metre\per\second}\</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), $^\circ$ (degree) and \AA (ring-A).⁹

$10 \mu\text{m}$	<code>\SI{10}{\mu m}\%</code>
$20 ^\circ\text{C}$	<code>\SI{20}{^\circ C}\%</code>
30\AA	<code>\SI{30}{\AA}\%</code>

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>

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

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

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

7.3 Powers of units

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

10 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>
$40/\text{s}^3$	<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 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>

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.

Table 4: 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

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.

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

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 4). 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 5). This leads to the redefinition of `\kilogram` as `\kilo\gram`. The macro `\deka` is provided, as this is used as an alias for `\deca` in some places. The package also defines a number of derived SI units which have assigned names and symbols (Table 6). 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 7), those considered “accepted” (Table 8), and those accepted temporarily (Table 9).¹⁵ The unit “litre” is often spelled “liter”

`\litre`
`\liter`

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

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 5: The SI prefixes (load=prefix)

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

Table 6: The derived SI units with defined names (load=derived)

Unit	Macro	Symbol	Unit	Macro	Symbol
becquerel	\becquerel	Bq	newton	\newton	N
celsius	\celsius	$^{\circ}\text{C}$	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 7: Units derived from experiments (load=physical)

Unit	Macro	Symbol
electron volt	<code>\electronvolt</code>	eV
unified atomic mass unit	<code>\atomicmassunit</code>	u
	<code>\atomicmass</code>	u

Table 8: Units accepted for use with SI (load=accepted)

Unit	Macro	Symbol
bel	<code>\bel</code>	B
day	<code>\Day</code>	d
	<code>\dday</code>	d
degree	<code>\degree</code>	°
hour	<code>\hour</code>	h
litre	<code>\litre</code>	l
	<code>\liter</code>	L
minute	<code>\minute</code>	min
minute (arc)	<code>\arcmin</code>	'
neper	<code>\neper</code>	Np
percent	<code>\percent</code>	%
second (arc)	<code>\arcsec</code>	"
tonne	<code>\tonne</code>	t

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=prefixed). Several of these also have obvious abbreviations (such as `\MHz` for `\megahertz`), which are made available by `load=abbr`. In common with the units discussed above, the prefixed and abbreviated unit definitions are loaded by default.

Table 10: Prefixed (load=prefixed) and abbreviated (load=abbr) units

Unit	Macro	Symbol	Abbreviation
<i>Masses</i>			
kilogram	<code>\kilogram</code>	kg	<code>\kg</code>
femtogram	<code>\femtogram</code>	fg	<code>\fg</code>
picogram	<code>\picogram</code>	pg	<code>\pg</code>
nanogram	<code>\nanogram</code>	ng	<code>\nanog</code>
microgram	<code>\microgram</code>	µg	<code>\micg</code>

Continued on next page

Unit	Macro	Symbol	Abbreviation
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
<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

Continued on next page

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

Table 9: Additional (temporary) SI units (load=addn)

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

7.8 Defining new units

`\newunit` New units are produced using the `\newunit` macro. This works as might be
`\renewunit` expected: `\newunit[⟨options⟩]{⟨unit⟩}{⟨symbol⟩}`, where `⟨symbol⟩` can contain
`\provideunit` literal values, other units, multiple prefixes, powers and `\per`. The `⟨options⟩`
argument can be any suitable options, and applies the specific unit macro only.
The most obvious example for using this macro is the `\degree` unit.¹⁶ The (first)
optional argument to `\SI` and `\si` can be used to override the settings for the
unit. The `\renewunit` and `\provideunit` macros take the same arguments.

```

3.1415°
12 345XXX 67 890 XXX
\SI{3.1415}{\degree}
\newunit[valuesep=none]{\oddunit}{XXX}
\SI{12345}{\oddunit}
\SI[valuesep=thick]{67890}{\oddunit}

```

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

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

```

10 m π⁻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`

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

¹⁷The `\mathrm` font used for this document has an “R” at the π position.

Table 11: 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

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.

$$\frac{\text{kg}^4}{\text{m}^4}$$

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

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

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

The binary prefixes, `\bit`, `\byte` (Table 11) are not formally part of the SI system. They are available by giving the `alsoload=binary` option.

100 MiB

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

8.2 Synthetic chemistry (synchem)

The `synchem` file adds the common chemistry units `\mmHg`, `\molar`, `\Molar`, `\torr` and `\dalton` to `siunitx`. The `\Molar` macro is somewhat awkward, as it can be given as either “m” or “M”. The later is obviously easily confused with the

Table 12: 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	

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 12). 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 second unit provided is \evperc, which is commonly-used and clear enough for a compound definition. Notice that the value of \evcorrb will need to be adjusted when using this unit.

4.657 MeV/c ²	\SI[per=slash,evcorrb=0.4ex]{4.657}{\mega\evperc\squared}
--------------------------	---

8.4 Astronomy (astro)

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

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

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

9 Font control

Following the lead of `Slstyle`, `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	<code>\num{10} \$\num{10}\$\\</code>
20° 20°	<code>\sffamily \ang{20} \$\ang{20}\$\\</code>
30 kg	<code>\textbf{\SI{30}{\kilo\gram}}\\</code>
40 kg	<code>\boldmath \$\SI{40}{\kilo\gram}\$</code>
50	<code>\[\num{50} \]</code>

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"	<code>\sisetup{obeyall}\\</code>
2°2'2" 2°2'2"	<code>\ang{1;1;1} \$\ang{1;1;1}\$\\</code>
3°3'3" 3°3'3"	<code>\sffamily \ang{2;2;2} \$\ang{2;2;2}\$\\</code>
4°4'4" 4°4'4"	<code>\textbf{\ang{3;3;3}} \boldmath \$\ang{3;3;3}\$\\</code>
5°5'5"	<code>\emph{\ang{4;4;4}} \emph{\$\ang{4;4;4}\$}</code>
	<code>\[\ang{5;5;5} \]</code>

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"	<code>\sisetup{obeyfamily}\\</code>
2°2'2" 2°2'2"	<code>\ang{1;1;1} \$\ang{1;1;1}\$\\</code>
3°3'3" 3°3'3"	<code>\sffamily \ang{2;2;2} \$\ang{2;2;2}\$\\</code>
4°4'4" 4°4'4"	<code>\sisetup{obeybold}</code>
5°5'5"	<code>\textbf{\ang{3;3;3}} \boldmath \$\ang{3;3;3}\$\\</code>
	<code>\emph{\ang{4;4;4}} \emph{\$\ang{4;4;4}\$}</code>
	<code>\[\ang{5;5;5} \]</code>

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.

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

The font commands used by the package to achieve the above are all available for user modification. The options `mathsrms`, `mathssfs` and `mathstts` hold the command sequences used in maths mode,²⁰ while `textrms`, `textsf` and `texttt` do the same for text mode. By default, these contain the obvious command names, for example `mathsrms=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 `mathssfs` 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

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.

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

name. For example, the `mathsrms` option may be split into `valuemathsrms` and `unitmathsrms`.

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

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

10.2 Spacing and separators

`unitsep` 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**.

`unitspace` 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.

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**.

1234	<code>\num{1234}\</code>
1 234	<code>\num[sepfour]{1234}</code>

<code>seperr</code>	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 <code>seperr</code> option, and requires that <code>numopenerr</code> and <code>numcloseerr</code> contain the left- and right-hand delimiters for the error [defaults <code>numopenerr=(</code> and <code>numcloseerr=)</code>].
<code>numopenerr</code>	
<code>numcloseerr</code>	

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

<code>trapambigerr</code>	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 <code>openerr</code> and <code>closeerr</code> 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 <code>trapambigerr</code> option, although for numbers with units the <code>repeatunits</code> option is also important. The spacing around the \pm sign is normally set by TeX. However, using the <code>tightpm</code> option will cause this to be reduced to a minimum.
<code>openerr</code>	
<code>closeerr</code>	
<code>tightpm</code>	

$1.234 \times 10^6 \pm 0.005 \times 10^6$	<code>\sisetup{seperr}\</code>
$1.234 \text{ m} \pm 0.005 \text{ m}$	<code>\num[trapambigerr=false]{1.234(5)e6}\</code>
$(1.234 \pm 0.005) \text{ m}$	<code>\SI{1.234(5)}{\metre}\</code>
$1.234 \pm 0.005 \text{ m}$	<code>\sisetup{repeatunits=false}</code>
1.234 ± 0.005	<code>\SI{1.234(5)}{\metre}\</code>
	<code>\SI[trapambigerr=false]{1.234(5)}{\metre}\</code>
	<code>\num[tightpm]{1.234(5)}</code>

<code>numprod</code>	The number processor can recognise products in the numerical input; the symbols used for products are stored in <code>numprod</code> , with the default value of “x”. By default, the <code>\SI</code> macro will repeat the units for a number given in this way. This behaviour is altered by the <code>repeatunits</code> option, which takes the values true , false and power . The later applies only when providing multiplied numbers with units, and converts the unit to the appropriate power rather than repeating the units. ²² Notice that when applied to errors, <code>repeatunits</code> takes priority over <code>trapambigerr</code> .
<code>repeatunits</code>	

²¹`fullstop` also has aliases `stop` and `period`.

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

$1 \times 2 \times 3$

$4\text{m} \times 5\text{m} \times 6\text{m}$

$1.2 \times 3.4 \times 5.6 \text{ mm}^3$

$(1.234 \pm 0.005) \text{ m}$

$9.109\,389\,7 \pm 0.000\,005\,4 \times 10^{-31} \text{ kg}$

$9.109\,389\,7 \times 10^{-31} \text{ kg} \pm 0.000\,005\,4 \times 10^{-31} \text{ kg}$

$1 \times 2 \times 3 \text{ m}^3$

```
\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 \times
`allowzeroexp` in 2×10^2), while the value of `expbase` sets the power used (the 10 in the exam-
 ple). 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.

0.1

2

3.0

4.0

0.5

6

7

.8

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

`fixdp` 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.

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`.

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 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>
$2^\circ 0' 0''$	<code>\ang[padangle=large]{2;;}\</code>
3°	<code>\ang[padangle=small]{3;}\</code>
$0^\circ 4' 0''$	<code>\ang[padangle=both]{;4;}\</code>
$5' 5''$	<code>\ang[padangle=none]{;5;5}\</code>
$1^\circ 2'$	<code>\ang[strictarc=false]{1;2}\</code>
$1^\circ 2' 3''$	<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 T_EX dimensions, and is therefore limited in accuracy. For this reason, the output is automatically rounded: output as a decimal angle is limited to three places, and that as an arc is given to a single decimal place for the seconds component.

$1^\circ 2' 3'' = 1.034^\circ$	<code>\ang{1;2;3} = \ang[angformat=dec]{1;2;3}\</code>
$4.56^\circ = 4^\circ 33' 36.0''$	<code>\ang{4.56} = \ang[angformat=arc]{4.56}\</code>

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

$1^\circ 2' 3.4''$	<code>\ang{1;2;3.4}\</code>
$5^\circ 6' 7''.8$	<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,

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

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

Table 13: The `tabalignexp` option

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

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 no line up (Table 13). 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}
```

`tabtextalign` Cells containing no numbers are handled by `siunitx` in a manner similar to
`tabunitalign` `\multicolumn`. The setting of `tabtextalign` is taken from the list **centre**,
`tabalign` `right` and `left`.²⁹ As would be expected, these settings `centre`, `right`- or `left`-

²⁸Aliases `centerdecimal = centredecimal`; `center = centre`.

²⁹Alias `centre = center`.

Table 14: The `tabautofit` option

Header	Header	Header
1.2	1.200	1.2
1.2345	1.235	1.2345

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 14).

```
\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}
```

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
`slash` applicable values here is **frac**, **nice**, **ugly** and **sfrac**. In each case, the unit

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

is typeset as a fraction, but the macro use to achieve this varies. `frac` uses the \TeX `\frac` macro, while `nice` makes use of a `\nicefrac`-like method. The `ugly` option uses a slash in text mode and `\frac` in maths mode.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>
$\frac{m}{s}$	<code>\si[fraction=frac]{\metre\per\second}\</code>
$\frac{m}{s}$	<code>\si[fraction=nice]{\metre\per\second}\</code>
$\frac{m}{s}$	<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

$\text{kg m}^{-1} \text{A}$	<code>\si{\kilogram\per\metre\ampere}\</code>
$\text{kg m}^{-1} \text{A}^{-1}$	<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.

$\text{m}/(\text{kg})$	<code>\sisetup{trapambigfrac,openfrac=(,closefrac=),per=slash}\</code>
m/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 `prefixbase` and `prefixproduct`, which work in the same way as `exbase` and `exproduct`.

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

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

³²`xfrac` is part of the experimental system for \LaTeX 3. 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, $\text{J mol}^{-1} \text{K}^{-1}$ is pronounced “joules per mole per kelvin”.

³⁴This is the behaviour in `Slunits`.

³⁵Provided things are not too complex!

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

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`.

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

³⁶This document uses `eVcorra=0.1ex`.

10.8 Colour

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

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

```

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

```

`colourneg` `siunitx` can automatically add a colour to negative numbers. This is turned on
`negcolour` 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>`.

```

6.022 × 1023 mol-1 \sisetup{loctolang={UK:UKenglish,DE:german}}\\
6,022 · 1023 mol-1 \SI{6.022e23}{\per\mole}\\
                    \selectlanguage{german}\\
                    \SI{6.022e23}{\per\mole}

```

10.10 Package control

`load` The package keeps most of the unit and abbreviations definitions in files separate
`noload` from `siunitx.sty`. To control what is loaded, three complementary options are
`alsoload` 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`
`debug`

`strict` 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.

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

Table 15: All package options

Option	Type	Description
<code>addsign</code>	List	Add sign to number
<code>allowzeroexp</code>	Boolean	Allow 10^0
<code>angformat</code>	List	Conversion of angle format
<code>anglesep</code>	List or literal	Space between angle components
<code>astroang</code>	Boolean	Astronomy-style angles
<code>closeerr</code>	Literal	Closes potential-ambiguous error
<code>closefrac</code>	Literal	Closes potential-ambiguous fraction
<code>color</code>	Literal	Colour used for units and values
<code>colour</code>	Literal	Colour used for units and values
<code>colorall</code>	Boolean	Switch for colouring all output
<code>colourall</code>	Boolean	Switch for colouring all output
<code>colorneg</code>	Boolean	Colour negative numbers
<code>colourneg</code>	Boolean	Colour negative numbers
<code>colorunits</code>	Boolean	Switch for colouring units

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
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
mathsminute	Literal	"'" symbol in maths mode
mathsmu	Literal	"μ" symbol in maths mode
mathsringA	Literal	"Å" symbol in maths mode
mathsrms	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

Continued on next page

Option	Type	Description
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/math)
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 “/”
stickyper	Boolean	Require \per only once
strict	Boolean	Obey the rules strictly
strictarc	Boolean	Require exactly zero or two semi-colons
tabalign	List	Positioning of all column data
tabalignexp	Boolean	Alignment of exponents in tables
tabautofit	Boolean	Switch for rounding numbers to length given by tabformat
tabformat	Number	Space reserved in table for numbers
tabnumalign	List	Alignment of S column numbers
tabtextalign	List	Positioning of text in S columns
tabunitalign	List	Positioning of units in s columns

Continued on next page

Option	Type	Description
<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 ±
<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

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

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 T_EX. 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.

`\requiresiconfigs`

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

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

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 kg⁻¹ 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`.

m/μs
m μs⁻¹
m μs⁻¹
kg m μs⁻¹
kgm μs⁻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.

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!

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

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 $\$. \dots \$$, 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 . in units

To use a literal `.` in a unit, it has to be protected from expansion. Using `\text` will not work here (in contrast to `SIstyle`). A robust command is needed:

10 V vs NHE	<code>\newunit{\myunit}{V~vs.~NHE}%</code>
20 V vs. NHE	<code>\SI{10}{\myunit}\</code>
	<code>\DeclareRobustCommand{\MyUnit}{V~vs.~NHE}</code>
	<code>\renewunit[unitspace=space]{\myunit}{\MyUnit}</code>
	<code>\SI{20}{\myunit}</code>

14.3 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.

$4\beta \times 10^{-7}$	<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 \LaTeX protection mechanism (`\DeclareRobustCommand`) may be sufficient; in some cases, this will fail and the $\epsilon\text{-}\TeX$ `\protected` system may be required. There are several potential pitfalls in this area; experimentation may well be needed.

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

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

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

14.4 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>

14.5 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.

14.6 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 16). 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}
```


Table 16: Non-standard S column

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

14.7 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 17).⁴¹ In most cases, this should not be a serious issue.

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

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 17: 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

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}
7 package.\\
8 Please put your demonstration here, and e-mail to the package
9 author.
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 `biblatex`. Thanks to the various contributors of ideas for the package: Donald Arseneau, Michele Dondi, Paul Gans, Ben Morrow, Lan Thuy Pham, Alan Ristow, Patrick Heinze, Andrea Blomenhofer, Morten Høgholm, Burkhard Moddemann and Patrick Steegstra.

Part III

Correct application of (SI) units

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 4. 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 6, Table 7 and Table 8)

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 7).

Some of these units (in Table 8) are regarded as only “temporarily” accepted, as the use of only the base and fully-consistent derived units in Table 6 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 5. 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 11) are the most obvious example. These are *not* part of the SI specifications, but the prefix names are derived from those in Table 5.

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

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.

chemists measure the pressure inside vacuum apparatus in mmHg, partly because the most common gauge for the task still uses a column of mercury metal. For these reasons, siunitx does define non-SI units.

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 18 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 18: 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 19](#). 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%
S[tabformat=1.3e1]<{\,\,\text{kilogram}}}%

```

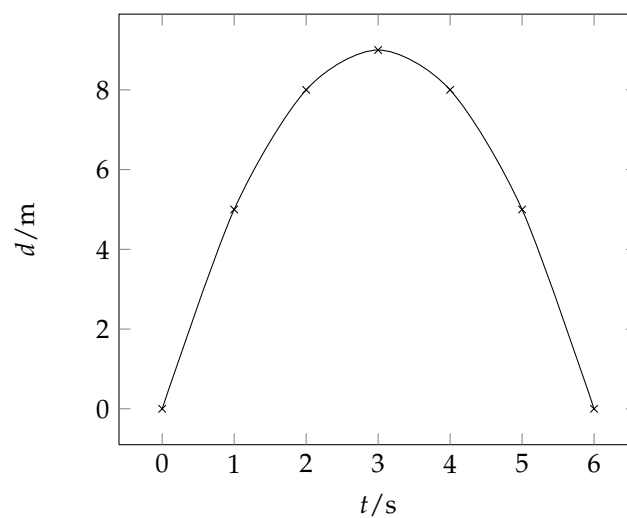



Figure 1: An example of graph labelling

Table 19: 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

```
\si@svn@version
\si@svn@id
\si@svn@ver
```

As always, the package starts with identification. This defines a couple of macros for use with *Subversion*, so that everything is nice and clear. This should also make it a bit easier to avoid messing up the revision data! For anyone reading the source, the revision number is also available as well as the release version number, of course.

```
1 \NeedsTeXFormat{LaTeX2e}
2 \newcommand*\si@svn@ver{v1.0b}
3 \def\si@svn@id$#1: #2.#3 #4 #5-#6-#7 #8 #9${%
4   \newcommand*\si@svn@version{%
5     #5/#6/#7\space\si@svn@ver\space}}
6 \si@svn@id $Id: siunitx.dtx 88 2008-06-23 08:25:52Z joseph $
7 \ProvidesPackage{siunitx}
8 [\si@svn@version A comprehensive (SI) units package]
```

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

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

⁴⁸Although extra optional arguments may be added.

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

`\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. L^AT_EX will have set `@` as a letter on loading siunitx.

```

19 \edef\si@catcodes{%
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'\string - \the\catcode\string'\-\relax
26   \catcode\string'\string + \the\catcode\string'\+\relax
27   \catcode\string'\string ; \the\catcode\string'\;\relax
28   \catcode\string'\string , \the\catcode\string'\,\relax
29   \catcode\string'\string . \the\catcode\string'\.\relax}
30 \catcode\string'\` 12\relax
31 \catcode'\= 12\relax
32 \catcode'\^ 7\relax
33 \catcode'\~ \active\relax
34 \@makeother{:}
35 \@makeother{-}
36 \@makeother{+}
37 \@makeother{;}
38 \@makeother{,}
39 \@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.

```

40 \RequirePackage{xkeyval}[2005/05/07]
41 \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 42 \newcommand*\si@tempa{}
43 \newcommand*\si@tempb{}
44 \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.

```

45 \newif\ifsi@switch

```

`\si@tempboxa` Some boxes are also needed.

```

\si@tempboxb 46 \newbox\si@tempboxa
\si@tempboxc 47 \newbox\si@tempboxb
\si@tempboxd 48 \newbox\si@tempboxc
49 \newbox\si@tempboxd

```

`\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.

```

\si@blockpkgs
\si@checkpkgs

```

```

50 \newcommand*{\si@blockpkgs}{SIunits,sistyle,siunits,SIstyle,%
51   unitsdef,fancyunits}
52 \newcommand*{\si@checkpkgs}{units,hepunits,fancynum}
53 \newcommand*{\si@packagecheck}{%
54   \begingroup
55   \@for\si@tempa:=\si@blockpkgs\do{
56     \ifpackageloaded{\si@tempa}
57       {\PackageError{siunitx}
58         {Package '\si@tempa' incompatible}
59         {The \si@tempa\space package and siunitx are
60           incompatible.\MessageBreak Use the
61           'emulate=\si@tempa' package option when loading
62           siunitx}}
63     {}

```

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

```

64   \@for\si@tempa:=\si@checkpkgs\do{%
65     \ifpackageloaded{\si@tempa}
66       {\PackageWarning{siunitx}
67         {Consider loading the siunitx package
68           with\MessageBreak option 'emulate=\si@tempa', rather
69           than\MessageBreak loading both \si@tempa\space and
70           siunitx}}
71     {}
72   \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.

```

73 \si@packagecheck
74 \AtBeginDocument{\si@packagecheck}

```

`\si@ifdefinable` 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.

```

\si@ifdefinable{<macro>}
75 \newcommand*{\si@ifdefinable}[1]{%
76   \expandafter\expandafter\expandafter\@ifundefined%
77   \expandafter\expandafter\expandafter%
78   {\expandafter\@gobble\string#1}}

```

`\si@addtolist` It is quite useful to be able to add to a comma-separated list of expandable items.

```

\si@addtolist{<macro>}{<items>}
79 \newcommand*{\si@addtolist}[2]{%
80   \ifx\@empty#1\@empty
81     \edef#1{#2}%
82   \else
83     \edef#1{#1,#2}%
84   \fi}

```

`\si@addtocname` A second item to add to a command sequence.

```

\si@temptoks \si@addtocname{<cname>}{<tokens>}
85 \newtoks{\si@temptoks}

```

```

86 \newcommand*{\si@addtocname}[2]{%
87   \@ifundefined{#1}
88     {\expandafter\gdef\csname #1\endcsname{#2}}
89     {\si@temptoks\expandafter\expandafter\expandafter{%
90       \csname #1\endcsname#2}%
91       \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
92 \begingroup
93   \catcode`\Q=3
94   \long\gdef\si@ifmtarg#1{%
95     \si@xifmtarg#1QQ\@secondoftwo\@firstoftwo\@nil}
96   \long\gdef\si@xifmtarg#1#2Q#3#4#5\@nil{#4}
97   \long\gdef\si@ifnotmtarg#1{%
98     \si@xifmtarg#1QQ\@firstofone\@gobble\@nil}
99 \endgroup

\si@newrobustcmd  Some more copying, this time from etoolbox. Various macros need to be really
\si@newcommand    robust. This is achieved using the  $\epsilon$ -TeX \protected primitive in various places.
\si@newcmd        However, it would be nice to have a \protected version of \newcommand.
\si@xargdef       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.
                  \si@newcommand{<macro>}
                  \si@newcmd{<macro>}[<num-args>]
                  \si@xargdef{<macro>}[<num-args>][<default>]{<def>}

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

21.2 Logging

```

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

```

121 \newif\ifsi@debug
122 \newif\ifsi@logmin
123 \newif\ifsi@lognone

\si@log@err Some handy re-usable macros are defined here. These all take names beginning
\si@log@warn These pop up in various places. First errors, warnings and information are
\si@log@inf handled. Package options are used to control how much output is given.
\si@log@err{<error>}{<explanation>}
\si@log@warn{<warning>}
\si@log@inf{<information>}

124 \newcommand*{\si@log@err}[2]{%
125   \ifsi@lognone\else
126     \ifsi@logmin
127       \PackageWarning{siunitx}{#1}%
128     \else
129       \PackageError{siunitx}{#1}{#2}%
130     \fi
131   \fi}
132 \newcommand*{\si@log@warn}[1]{%
133   \ifsi@lognone\else
134     \ifsi@logmin\else
135       \PackageWarning{siunitx}{#1}%
136     \fi
137   \fi}
138 \newcommand*{\si@log@inf}[1]{%
139   \ifsi@lognone\else
140     \ifsi@logmin\else
141       \PackageInfo{siunitx}{#1}%
142     \fi
143   \fi}

\si@log@debug The debug macro only gives output if the appropriate package option is set.
\si@log@debug{<debug-information>}

144 \newcommand*{\si@log@debug}[1]{%
145   \ifsi@lognone\else
146     \ifsi@debug
147       \PackageInfo{siunitx}{#1}%
148     \fi
149   \fi}

```

21.3 String comparison

```

\si@str@ifchrstr At various points, the package needs to compare two strings, to find if one occurs
\si@str@chrstr 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.
\si@str@ifchrstr{<char>}{<chars>}
\si@str@chrstr<char><chars>\@empty

150 \newcommand*{\si@str@ifchrstr}[2]{%
151   \begingroup
152     \si@switchfalse

```

```

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

```

`\si@str@ifonlychrs` 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.

```

\si@str@ifonlychrs{<string>}{<valid-chars>}
\si@str@onlychrs<char><chars>\@empty
171 \newcommand*{\si@str@ifonlychrs}[2]{%
172 \begingroup
173 \si@switchtrue
174 \protected@edef\si@tempa{#1}%
175 \renewcommand*{\si@tempb}{#2}%
176 \expandafter\si@str@onlychrs\si@tempa\@empty\@empty\@empty
177 \ifsi@switch
178 \aftergroup\@firstoftwo
179 \else
180 \aftergroup\@secondoftwo
181 \fi
182 \endgroup}
183 \def\si@str@onlychrs#1#2\@empty{%
184 \si@str@ifchrstr{#1}{\si@tempb}
185 {}{\si@switchfalse}%
186 \ifx\@empty#2\@empty\else
187 \si@str@onlychrs#2\@empty\@empty
188 \fi}

```

21.4 Option handling

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

```

189 \newcommand*{\sisetup}{\setkeys[si]{key}}

```

`\si@opt@key` 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.

```

\si@opt@key{<keyname>}{<code>}

```

```

190 \newcommand*\si@opt@key}[2]{%
191   \define@key[si]{key}{#1}
192   {#2\si@log@debug{Option #1 set to ##1}}}

```

`\si@opt@cmdkey` The command versions of the above.

`\si@opt@cmdkeys` `\si@opt@cmdkey[<default>]{<keyname>}{<function>}`
`\si@opt@cmdkeys[<default>]{<keynames>}`

```

193 \newcommand*\si@opt@cmdkey}[3][{}]{%
194   \define@cmdkey[si]{key}[si@]{#2}[#1]{#3}}
195 \newcommand*\si@opt@cmdkeys}[2][{}]{%
196   \define@cmdkeys[si]{key}[si@]{#2}[#1]}

```

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

```

\si@opt@boolkey[<optional-processing>]{<keyname>}

197 \newcommand*\si@opt@boolkey}[2][{}]{%
198   \define@boolkey[si]{key}[si@]{#2}[true]
199   {#1\si@log@debug{Option #2 set to ##1}}}

```

`\si@opt@choicekey` 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.

```

\si@opt@choicekey[<default>]{<keyname>}{<choices>}{<in-list>}

200 \newcommand*\si@opt@choicekey}[4][{}]{%
201   \define@choicekey*[si]{key}{#2}[\si@tempa]{#3}[#1]
202   {#4\si@log@debug{Option #2 set to ##1}}
203   {\si@log@warn{Unknown value ‘##1’ for option #2}}}

```

`\si@opt@xchoicekey` 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.

```

\si@opt@xchoicekey{<keyname>}{<choices>}{<initial>}

204 \newcommand*\si@opt@xchoicekey}[3]{%
205   \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.

```

206   {\si@log@debug{Option #1 set to ##1}}%
207   \expandafter\renewcommand\expandafter*\expandafter{%
208     \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.

```

209   {\si@log@debug{Option #1 set to ##1}}%
210   \expandafter\renewcommand\expandafter*\expandafter{%
211     \csname si@#1\endcsname}{##1}}

```

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

```

212   \expandafter\newcommand\expandafter*\expandafter{%
213     \csname si@#1\endcsname}%
214     {\@nameuse{si@fix@#3}}}

```


`\si@opt@compatkey` 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.

```
\si@opt@compatkey{<package>}{<keyname>}.
215 \newcommand*{\si@opt@compatkey}[2]{%
216   \define@boolkey[si]{key}[si@old@]{#2}[true]
217   {\si@log@debug{Emulating #1 package option\MessageBreak #2}%
218     \sisetup{emulate=#1}%
219     \si@log@debug{Option #2 set to ##1}}
220   \AtEndOfPackage{\si@opt@disablekey{#2}
221     {Compatibility option #2 only\MessageBreak
222       available when loading siunitx package}}}
```

`\si@opt@disablekey` The ability to disable a key with a meaningful message is a must; the warning will come from siunitx, and not from xkeyval

```
\si@opt@disablekey{<keyname>}{<warning>}
223 \newcommand*{\si@opt@disablekey}[2]{%
224   \key@ifundefined[si]{key}{#1}
225   {}
226   {\si@log@debug{Disabling key #1}%
227     \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.

```
228 \si@opt@choicelkey[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.

```
229 {\si@debugfalse
230   \si@logminfalse
231   \si@lognonefalse
232   \renewcommand*{\si@tempb}{none}%
233   \ifx\si@tempa\si@tempb
234     \si@lognonetrue
235   \fi
236   \renewcommand*{\si@tempb}{minimal}%
237   \ifx\si@tempa\si@tempb
238     \si@logmintrue
239   \fi
240   \renewcommand*{\si@tempb}{debug}%
241   \ifx\si@tempa\si@tempb
242     \si@debugtrue
243   \fi
244   \renewcommand*{\si@tempb}{verbose}%
245   \ifx\si@tempa\si@tempb
246     \si@debugtrue
247   \fi}
```

A quick method to set `log=debug`.

```
248 \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.

```

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

```

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

```

272 \newcommand*{\si@emulate}{}
273 \si@opt@key{emulate}{\si@addtolist{\si@emulate}{#1}}
274 \AtEndOfPackage{
275   \si@opt@disablekey{emulate}
276   {Option 'emulate' only available when\MessageBreak
277     loading package}}

```

`\si@unitsep` 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`).
`\si@unitspace` Known values here are `thin`, `med`, `medium`, `thick`, `cdot`, `tightcdot`⁵⁰ and
`\si@errspace` `none`;⁵¹ other entries will be treated as custom spaces.
`\si@valuesep`

```

278 \si@opt@xchoicekey{unitsep}
279   {thin,med,medium,thick,none,comma,stop,fullstop,period,
280     times,tighttimes,cdot,tightcdot}{thin}
281 \si@opt@xchoicekey{unitspace}{space,thin,med,medium,thick,
282   none}{thin}
283 \si@opt@xchoicekey{errspace}{space,thin,med,medium,thick,
284   none}{none}
285 \si@opt@xchoicekey{valuesep}
286   {thin,med,medium,thick,none,comma,stop,fullstop,period,
287     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

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

⁵¹Only valid for `valuesep`.

quite long, so that a range of options are handled automatically. Notice that `digitsep=none` will be used for no separation at all.

```
288 \si@opt@xchoicekey{digitsep}
289   {thin,med,medium,thick,none,comma,stop,fullstop,period,
290     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.

```
291 \si@opt@xchoicekey{decimalsymbol}{comma,stop,fullstop,period,
292   cdot,tightcdot}{fullstop}
```

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

```
293 \si@opt@xchoicekey{anglesep}
294   {thin,med,medium,thick,none,comma,stop,fullstop,period,
295     times,tighttimes,cdot,tightcdot}{none}
```

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

```
296 \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.

```
297 \newif\ifsi@numtextmode
298 \newif\ifsi@unittextmode
```

The `textmode` option has to set both flags.

```
299 \si@opt@choicekey[true]{textmode}{true,false}
300   {\si@numtextmodefalse
301     \si@unittextmodefalse
302     \renewcommand*{\si@tempb}{true}%
303     \ifx\si@tempa\si@tempb
304       \si@numtextmodetrue
305       \si@unittextmodetrue
306     \fi}
```

The `mode` option applies to numbers and units.

```
307 \si@opt@choicekey{mode}{math,maths,text}
308   {\si@numtextmodefalse
309     \si@unittextmodefalse
310     \renewcommand*{\si@tempb}{text}%
311     \ifx\si@tempa\si@tempb
312       \si@numtextmodetrue
313       \si@unittextmodetrue
314     \fi}
```

Now the two options for numbers or units alone.

```
315 \si@opt@choicekey{valuemode}{math,maths,text}
316   {\si@numtextmodefalse
317     \renewcommand*{\si@tempb}{text}%
318     \ifx\si@tempa\si@tempb
319       \si@numtextmodetrue
320     \fi}
```

```

321 \si@opt@choicekey{unitmode}{math,maths,text}
322 {\si@unittextmodefalse
323 \renewcommand*{\si@tempb}{text}%
324 \ifx\si@tempa\si@tempb
325 \si@unittextmodetrue
326 \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.

```

327 \si@opt@boolkey{obeyfamily}

```

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

```

328 \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`).

```

329 \newif\ifsi@inlinebtext
330 \si@opt@choicekey{inlinebold}{text,maths,math}
331 {\si@inlinebtextfalse
332 \renewcommand*{\si@tempb}{text}%
333 \ifx\si@tempa\si@tempb
334 \si@inlinebtexttrue
335 \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.

```

336 \si@opt@boolkey{obeyitalic}

```

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

```

337 \si@opt@boolkey{detectdisplay}

```

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

```

338 \si@opt@choicekey[true]{obeyall}{true,false}
339 {\si@obeyboldfalse
340 \si@obeyitalicfalse
341 \si@obeymodefalse
342 \si@obeyfamilyfalse
343 \renewcommand*{\si@tempb}{true}%
344 \ifx\si@tempa\si@tempb
345 \si@obeyboldtrue
346 \si@obeyitalictrue
347 \si@obeymodetrue
348 \si@obeyfamilytrue
349 \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@valuemathssf
\si@valuemathstt
\si@unitmathsrms 350 \si@opt@cmdkeys{valuemathsrms,valuemathssf,valuemathstt}
\si@unitmathssf 351 \si@opt@cmdkeys{unitmathsrms,unitmathssf,unitmathstt}
\si@unitmathstt

```

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

```
352 \si@opt@key{valuemathrm}{\sisetup{valuemathsrmsf=#1}}
353 \si@opt@key{valuemathsf}{\sisetup{valuemathsssf=#1}}
354 \si@opt@key{valuemathstt}{\sisetup{valuemathsttsf=#1}}
355 \si@opt@key{unitmathrm}{\sisetup{unitmathsrmsf=#1}}
356 \si@opt@key{unitmathsf}{\sisetup{unitmathsssf=#1}}
357 \si@opt@key{unitmathstt}{\sisetup{unitmathsttsf=#1}}
```

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

```
358 \si@opt@key{mathsrmsf}{\sisetup{valuemathsrmsf=#1,unitmathsrmsf=#1}}
359 \si@opt@key{mathsssf}{\sisetup{valuemathsssf=#1,unitmathsssf=#1}}
360 \si@opt@key{mathsttsf}{\sisetup{valuemathsttsf=#1,unitmathsttsf=#1}}
361 \si@opt@key{mathrm}{\sisetup{valuemathsrmsf=#1,unitmathsrmsf=#1}}
362 \si@opt@key{mathsf}{\sisetup{valuemathsssf=#1,unitmathsssf=#1}}
363 \si@opt@key{mathstt}{\sisetup{valuemathsttsf=#1,unitmathsttsf=#1}}
```

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

```
\si@valuetextsf
\si@valuetexttt 364 \si@opt@cmdkeys{valuetextrm,valuetextsf,valuetexttt}
\si@unittextrm 365 \si@opt@cmdkeys{unittextrm,unittextsf,unittexttt}
\si@unittextsf 366 \si@opt@key{textrm}{\sisetup{unittextrm=#1,valuetextrm=#1}}
\si@unittexttt 367 \si@opt@key{textsf}{\sisetup{unittextsf=#1,valuetextsf=#1}}
368 \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 369 \si@opt@cmdkeys{numdigits,numdecimal,numexp,numgobble,numsign,%
\si@numprod 370 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 371 \newcommand*{\si@numextra}{%
372 \si@numopenerr\si@numcloseerr\si@numaddn}
373 \newcommand*{\si@numvalid}{\si@numgobble\si@numexp\si@numsign
374 \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 375 \si@opt@boolkey{seperr}
\si@closeerr 376 \si@opt@boolkey{trapambigerr}
377 \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`.

```
378 \si@opt@boolkey{sepfour}
```

`\ifsi@retainplus` An option to keep an explicit positive sign.

```
379 \si@opt@boolkey{retainplus}
```

```

\si@expbase    The options for exponents are set up.
\si@expproduct 380 \si@opt@xchoicekey{expproduct}{times,tighttimes,
381   cdot,tightcdot}{times}
382 \si@opt@xchoicekey{expbase}{ten}{ten}

\ifsi@allowzeroexp  The package normally prevents 100.
383 \si@opt@boolkey{allowzeroexp}

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

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

\ifsi@prefixsymbolic  Unit prefixes can be given as either symbols or numerically.
387 \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”).
388 \newif\ifsi@num@padlead
389 \newif\ifsi@num@padtrail
390 \si@opt@choicekey[all]{padnumber}
391   {leading,lead,trailing,trail,all,both,true,none,false}
392   {\si@num@padleadfalse
393    \si@num@padtrailfalse
394    \renewcommand*{\si@tempb}{leading}%
395    \ifx\si@tempa\si@tempb
396      \si@num@padleadtrue
397    \fi
398    \renewcommand*{\si@tempb}{lead}%
399    \ifx\si@tempa\si@tempb
400      \si@num@padleadtrue
401    \fi
402    \renewcommand*{\si@tempb}{trailing}%
403    \ifx\si@tempa\si@tempb
404      \si@num@padtrailtrue
405    \fi
406    \renewcommand*{\si@tempb}{trail}%
407    \ifx\si@tempa\si@tempb
408      \si@num@padtrailtrue
409    \fi
410    \renewcommand*{\si@tempb}{all}%
411    \ifx\si@tempa\si@tempb
412      \si@num@padleadtrue
413      \si@num@padtrailtrue
414    \fi
415    \renewcommand*{\si@tempb}{true}%
416    \ifx\si@tempa\si@tempb
417      \si@num@padleadtrue
418      \si@num@padtrailtrue
419    \fi
420    \renewcommand*{\si@tempb}{both}%

```

```

421 \ifx\si@tempa\si@tempb
422 \si@num@padleadtrue
423 \si@num@padtrailtrue
424 \fi}

```

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

```

\ifsi@num@signmant 425 \newif\ifsi@num@signmant
\ifsi@num@signexp 426 \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.

```

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

```

The option is now processed.

```

430 {\si@num@signmantfalse
431 \si@num@signexpfalse
432 \renewcommand*{\si@tempb}{mantissa}%
433 \ifx\si@tempa\si@tempb
434 \si@num@signmanttrue
435 \fi
436 \renewcommand*{\si@tempb}{mant}%
437 \ifx\si@tempa\si@tempb
438 \si@num@signmanttrue
439 \fi
440 \renewcommand*{\si@tempb}{exponent}%
441 \ifx\si@tempa\si@tempb
442 \si@num@signexptrue
443 \fi
444 \renewcommand*{\si@tempb}{exp}%
445 \ifx\si@tempa\si@tempb
446 \si@num@signexptrue
447 \fi
448 \renewcommand*{\si@tempb}{all}%
449 \ifx\si@tempa\si@tempb
450 \si@num@signmanttrue
451 \si@num@signexptrue
452 \fi
453 \renewcommand*{\si@tempb}{true}%
454 \ifx\si@tempa\si@tempb
455 \si@num@signmanttrue
456 \si@num@signexptrue
457 \fi
458 \renewcommand*{\si@tempb}{both}%
459 \ifx\si@tempa\si@tempb
460 \si@num@signmanttrue
461 \si@num@signexptrue
462 \fi}

```

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

```

\si@pm 463 \si@opt@boolkey{tightpm}
464 \newcommand*{\si@pm}{%
465 \ifsi@tightpm

```

```

466     \si@fix@tightpm
467 \else
468     \si@fix@pm
469 \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.

```

470 \newif\ifsi@ang@padsmall
471 \newif\ifsi@ang@padlarge
472 \si@opt@choicakey[all]{padangle}
473 {small,large,all,both,true,none,false}
474 {\si@ang@padsmallfalse
475  \si@ang@padlargefalse
476  \renewcommand*\si@tempb}{small}%
477  \ifx\si@tempa\si@tempb
478    \si@ang@padsmalltrue
479  \fi
480  \renewcommand*\si@tempb}{large}%
481  \ifx\si@tempa\si@tempb
482    \si@ang@padlargetrue
483  \fi
484  \renewcommand*\si@tempb}{all}%
485  \ifx\si@tempa\si@tempb
486    \si@ang@padsmalltrue
487    \si@ang@padlargetrue
488  \fi
489  \renewcommand*\si@tempb}{true}%
490  \ifx\si@tempa\si@tempb
491    \si@ang@padsmalltrue
492    \si@ang@padlargetrue
493  \fi
494  \renewcommand*\si@tempb}{both}%
495  \ifx\si@tempa\si@tempb
496    \si@ang@padsmalltrue
497    \si@ang@padlargetrue
498  \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.

```

499 \newif\ifsi@ang@toarc
500 \newif\ifsi@ang@toddec
501 \si@opt@choicakey[all]{angformat}
502 {dec,decimal,arc,dms,unchanged,none}
503 {\si@ang@toarcfalse
504  \si@ang@toddecfalse
505  \renewcommand*\si@tempb}{dec}%
506  \ifx\si@tempa\si@tempb
507    \si@ang@todectrue
508  \fi
509  \renewcommand*\si@tempb}{decimal}%
510  \ifx\si@tempa\si@tempb
511    \si@ang@todectrue
512  \fi}

```



```

513 \renewcommand*{\si@tempb}{arc}%
514 \ifx\si@tempa\si@tempb
515 \si@ang@toarctrue
516 \fi
517 \renewcommand*{\si@tempb}{dms}%
518 \ifx\si@tempa\si@tempb
519 \si@ang@toarctrue
520 \fi}

```

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

```
521 \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.

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

```

`\ifsi@tabalignexp` A switch for aligning exponents.

```
547 \si@opt@boolkey{tabalignexp}
```

`\si@tab@mantprecnt` To process the format information, various internal number-processing macros are used. First, some storage areas are created.

```

\si@tab@expprecnt 548 \newcount\si@tab@mantprecnt
\si@tab@exppostcnt 549 \newcount\si@tab@mantpostcnt
\ifsi@tab@mantsign 550 \newcount\si@tab@expprecnt
\ifsi@tab@expsign 551 \newcount\si@tab@exppostcnt

```

```

552 \newif\ifsi@tab@mantsign
553 \newif\ifsi@tab@expsign

```

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

```

554 \si@opt@cmdkey{tabformat}
555   {\si@num@fixpm
556    \renewcommand*{\si@num@arg}{tabformat data}%
557    \renewcommand*{\si@num@exp}{}%
558    \renewcommand*{\si@num@mant}{}%
559    \si@tab@mantsignfalse
560    \si@tab@expsignfalse
561    \si@switchfalse
562    \si@num@sepmantexp{#1}%

```

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

```

563   \si@num@sepsign{mant}%
564   \ifx\@empty\si@num@mantsign\@empty
565     \ifsi@num@delplus
566       \expandafter\expandafter\expandafter\si@tab@mantsigntrue
567     \fi
568   \else
569     \expandafter\si@tab@mantsigntrue
570   \fi
571   \si@num@sepsign{exp}%
572   \ifx\@empty\si@num@expsign\@empty
573     \ifsi@num@delplus
574       \expandafter\expandafter\expandafter\si@tab@expsigntrue
575     \fi
576   \else
577     \expandafter\si@tab@expsigntrue
578   \fi
579   \si@opt@proctform{mant}%
580   \si@opt@proctform{exp}%

```

The tabformat input should really have both an integer and decimal part for the mantissa. If neither are present, an error is raised. If just the one part is missing, an warning seems more suitable.

```

581   \ifnum\si@tab@mantpostcnt=\z@\relax
582   \ifnum\si@tab@mantprecnt=\z@\relax
583     \si@log@err{Empty mantissa argument for tabformat}
584     {The argument '#1' contains no valid entry for
585      a mantissa\MessageBreak It should be of the
586      form 'm.n', where m and n are integers}%
587   \else
588     \si@log@warn{Argument of tabformat contains\MessageBreak
589      no decimal part}%
590   \fi
591 \else
592   \ifnum\si@tab@mantprecnt=\z@\relax
593     \si@log@warn{Argument of tabformat contains\MessageBreak
594      no integer part}%
595   \fi
596 \fi

```

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

```
597 \ifsi@tab@fixed\else
598 \sisetup{tabnumalign=centre}%
599 \fi
600 \let\pm\si@num@pm
601 \let\mp\si@num@mp}
```

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

```
\si@opt@proctform{<mant/exp>}
602 \newcommand*{\si@opt@proctform}[1]{%
603 \renewcommand*{\si@num@predec}{}%
604 \renewcommand*{\si@num@postdec}{}%
605 \si@switchfalse
606 \expandafter\si@ifnotmtarg\expandafter{%
607 \csname si@num@#1\endcsname}
608 {\expandafter\expandafter\expandafter\si@num@digits
609 \csname si@num@#1\endcsname\@empty\@empty}%
610 \csname si@tab@#1precnt\endcsname\z@\relax
611 \csname si@tab@#1postcnt\endcsname\z@\relax
612 \ifx\@empty\si@num@predec\@empty\else
613 \csname si@tab@#1precnt\endcsname\si@num@predec\relax
614 \fi
615 \ifx\@empty\si@num@postdec\@empty\else
616 \csname si@tab@#1postcnt\endcsname\si@num@postdec\relax
617 \fi}
```

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

```
618 \si@opt@choicekey{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 619 {\let\si@tab@rfill@t\hfill
620 \let\si@tab@lfill@t\hfill
621 \renewcommand*{\si@tempb}{right}%
622 \ifx\si@tempa\si@tempb
623 \let\si@tab@rfill@t\relax
624 \fi
625 \renewcommand*{\si@tempb}{left}%
626 \ifx\si@tempa\si@tempb
627 \let\si@tab@lfill@t\relax
628 \fi}
```

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

```
629 \si@opt@choicekey{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 630 {\let\si@tab@rfill@s\hfill
631 \let\si@tab@lfill@s\hfill
632 \renewcommand*{\si@tempb}{right}%
633 \ifx\si@tempa\si@tempb
634 \let\si@tab@rfill@s\relax
```

```

635 \fi
636 \renewcommand*{\si@tempb}{left}%
637 \ifx\si@tempa\si@tempb
638 \let\si@tab@lfill@s\relax
639 \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.

```

\si@num@dp
640 \si@opt@boolkey{fixdp}
641 \newcount\si@num@dp
642 \si@opt@key{dp}{%
643 \si@str@ifonlychrs{#1}{0123456789}
644 {}
645 {\si@log@err{Invalid input for dp option}
646 {The dp option must be given a positive integer}}%
647 \si@num@dp#1\relax
648 \si@fixdptrue}

```

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

```

649 \si@opt@boolkey{tabautofit}

```

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

```

650 \si@opt@boolkey{xspace}

```

`\ifsi@prespace`

```

651 \si@opt@boolkey
652 [\si@unt@numfalse
653 \ifsi@prespace
654 \si@unt@numtrue
655 \fi]
656 {prespace}

```

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

```

657 \si@opt@boolkey{allowoptarg}

```

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

```

\ifsi@slash 658 \newif\ifsi@slash
\ifsi@stickyper 659 \newif\ifsi@frac
660 \si@opt@boolkey{stickyper}
661 \si@opt@choickey[reciprocal]{per}
662 {reciprocal,rp,power,slash,frac,fraction}
663 {\si@slashfalse
664 \si@fracfalse
665 \renewcommand*{\si@tempb}{slash}%
666 \ifx\si@tempa\si@tempb
667 \si@fractrue
668 \si@slashtrue
669 \let\si@frac\si@frc@slash
670 \fi
671 \renewcommand*{\si@tempb}{frac}%
672 \ifx\si@tempa\si@tempb
673 \si@fractrue
674 \fi

```

```

675 \renewcommand*\si@tempb}{fraction}%
676 \ifx\si@tempa\si@tempb
677 \si@fractrue
678 \fi}

```

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

```

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

```

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

```

\ifsi@addunitpower 680 \newif\ifsi@repeatunits
681 \newif\ifsi@addunitpower
682 \si@opt@choicekey[true]{repeatunits}{true,false,power}
683 {\si@repeatunitsfalse
684 \si@addunitpowerfalse
685 \renewcommand*\si@tempb}{true}%
686 \ifx\si@tempa\si@tempb
687 \si@repeatunitstrue
688 \fi
689 \renewcommand*\si@tempb}{power}%
690 \ifx\si@tempa\si@tempb
691 \si@addunitpowertrue
692 \fi}

```

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

```

\si@closefrac
\si@openfrac 693 \si@opt@boolkey{trapambigfrac}
694 \si@opt@cmdkeys{closefrac,openfrac}

```

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

```

695 \si@opt@choicekey[frac]{fraction}
696 {frac,nicefrac,nice,sfrac,xfrac,uglyfrac,ugly}
697 {\let\si@frac\si@frc@frac
698 \renewcommand*\si@tempb}{nicefrac}%
699 \ifx\si@tempa\si@tempb
700 \let\si@frac\si@frc@nice
701 \fi
702 \renewcommand*\si@tempb}{uglyfrac}%
703 \ifx\si@tempa\si@tempb
704 \let\si@frac\si@frc@ugly
705 \fi
706 \renewcommand*\si@tempb}{nice}%
707 \ifx\si@tempa\si@tempb
708 \let\si@frac\si@frc@nice
709 \fi
710 \renewcommand*\si@tempb}{sfrac}%
711 \ifx\si@tempa\si@tempb
712 \let\si@frac\si@frc@sfrac
713 \fi
714 \renewcommand*\si@tempb}{xfrac}%
715 \ifx\si@tempa\si@tempb
716 \let\si@frac\si@frc@sfrac
717 \fi

```

```

718 \renewcommand*{\si@tempb}{ugly}%
719 \ifx\si@tempa\si@tempb
720 \let\si@frac\si@frc@ugly
721 \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.

```

722 \si@opt@cmdkeys{load,noload}
723 \si@opt@key{alsoload}{\si@addtolist{\si@load}{#1}}
724 \AtEndOfPackage{
725 \si@opt@disablekey{load}
726 {Configuration files can only be used\MessageBreak
727 when loading package}
728 \si@opt@disablekey{noload}
729 {Configuration files can only be used\MessageBreak
730 when loading package}}
731 \AtEndOfPackage{
732 \si@opt@key{alsoload}{%
733 \@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 734 \si@opt@boolkey{colourunits}
\si@valuecolour 735 \si@opt@boolkey{colourvalues}
736 \si@opt@choicekey[true]{colorunits}
737 {true,false}
738 {\si@colourunitsfalse
739 \renewcommand*{\si@tempb}{true}%
740 \ifx\si@tempa\si@tempb
741 \si@colourunitstrue
742 \fi}
743 \si@opt@choicekey[true]{colorvalues}
744 {true,false}
745 {\si@colourvaluesfalse
746 \renewcommand*{\si@tempb}{true}%
747 \ifx\si@tempa\si@tempb
748 \si@colourvaluestru
749 \fi}
750 \si@opt@choicekey[true]{colorall}
751 {true,false}
752 {\si@colourvaluesfalse
753 \si@colourunitsfalse
754 \renewcommand*{\si@tempb}{true}%
755 \ifx\si@tempa\si@tempb
756 \si@colourunitstrue
757 \si@colourvaluestru
758 \fi}
759 \si@opt@choicekey[true]{colourall}
760 {true,false}
761 {\si@colourvaluesfalse
762 \si@colourunitsfalse
763 \renewcommand*{\si@tempb}{true}%
764 \ifx\si@tempa\si@tempb

```

```

765 \si@colourunitstrue
766 \si@colourvaluesttrue
767 \fi}
768 \si@opt@cmdkeys{unitcolour,valuecolour}
769 \si@opt@key{unitcolor}{\sisetup{unitcolour=#1}}
770 \si@opt@key{valuecolor}{\sisetup{valuecolour=#1}}
771 \si@opt@key{colour}{\sisetup{unitcolour=#1,valuecolour=#1}}
772 \si@opt@key{color}{\sisetup{unitcolour=#1,valuecolour=#1}}

```

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

```

\si@negcolour 773 \si@opt@boolkey{colourneg}
774 \si@opt@choicekey[true]{colorneg}
775 {true,false}
776 {\si@colournegfalse
777 \renewcommand*{\si@tempb}{true}%
778 \ifx\si@tempa\si@tempb
779 \si@colournegtrue
780 \fi}
781 \si@opt@cmdkeys{negcolour}
782 \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.

```

783 \si@opt@cmdkeys{textOmega,mathsOmega}
784 \si@opt@key{mathOmega}{\sisetup{mathsOmega=#1}}
785 \newcommand*{\si@mathsOmega}{\text{\ensuremath{\Omega}}}
786 \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 787 \si@opt@cmdkeys{textmu,mathsmu}
788 \si@opt@key{mathmu}{\sisetup{mathsmu=#1}}
789 \DeclareFontEncoding{TS1}{}{}
790 \DeclareFontSubstitution{TS1}{cmr}{m}{n}
791 \DeclareTextSymbol{\si@textmu}{TS1}{181}
792 \DeclareTextSymbolDefault{\si@textmu}{TS1}
793 \DeclareFontFamily{OML}{eur}{\skewchar\font127}
794 \DeclareFontShape{OML}{eur}{m}{n}%
795 {<5> <6> <7> <8> <9> gen * eurm %
796 <10><10.95><12><14.4><17.28><20.74><24.88>eurm10}{}
797 \DeclareSymbolFont{greek}{OML}{eur}{m}{n}
798 \DeclareMathSymbol{\si@mathsmu}{\mathord}{greek}{"16}

```

`\si@textdegree` The angle signs.

```

\si@mathsdegree 799 \si@opt@cmdkeys{textdegree,mathsdegree,textminute,mathsminute,
\si@textminute 800 textsecond,mathssecond}
\si@mathsminute 801 \si@opt@key{mathdegree}{\sisetup{mathsdegree=#1}}
\si@textsecond 802 \si@opt@key{mathminute}{\sisetup{mathsminute=#1}}
\si@mathssecond 803 \si@opt@key{mathsecond}{\sisetup{mathssecond=#1}}
804 \newcommand*{\si@textdegree}{\ensuremath{{}^{\circ}}}
805 \newcommand*{\si@mathsdegree}{{}^{\circ}}
806 \newcommand*{\si@textminute}{\ensuremath{{}'}}
807 \newcommand*{\si@mathsminute}{{}'}

```

```

808 \newcommand*{\si@textsecond}{\ensuremath{{}^{\prime}\prime}}
809 \newcommand*{\si@mathssecond}{{}^{\prime}\prime}}

\si@textcelsius    Finally, degrees Celsius, which may need the degree symbol.
\si@mathscelsius 810 \si@opt@cmdkeys{textcelsius,mathscelsius}
811 \si@opt@key{mathcelsius}{\sisetup{mathscelsius=#1}}
812 \newcommand*{\si@textcelsius}{%
813   \si@textdegree\kern-\scriptspace C}
814 \newcommand*{\si@mathscelsius}{%
815   \si@mathsdegree\kern-\scriptspace\mathrm{C}}

\si@textringA    The Å sign.
\si@mathsringA 816 \si@opt@cmdkeys{textringA,mathsringA}
817 \si@opt@key{mathringA}{\sisetup{mathsringA=#1}}
818 \newcommand*{\si@textringA}{\AA}
819 \newcommand*{\si@mathsringA}{\text{\AA}}

\ifsi@redefsymbols  A flag for using textcomp and upgreek to provide better symbols.
820 \si@opt@boolkey{redefsymbols}
821 \AtBeginDocument{
822   \si@opt@disablekey{redefsymbols}
823   {Symbols can only be redefined\MessageBreak
824     when loading siunitx}}

\si@eVcorra
\si@eVcorrb 825 \newlength\si@eVcorra
826 \newlength\si@eVcorrb
827 \si@opt@key{eVcorra}{\setlength\si@eVcorra{#1}}
828 \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.
829 \si@opt@key{locale}{%
830   \si@loc@load{#1}%
831   \si@loc@set{#1}}%
832 \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.
833 \si@opt@compatkey{units}{ugly}
834 \si@opt@compatkey{units}{nice}
835 \si@opt@compatkey{units}{loose}
836 \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 837 \si@opt@compatkey{unitsdef}{OHM}
\ifsi@gensymb 838 \si@opt@compatkey{unitsdef}{ohm}
839 \si@opt@compatkey{unitsdef}{redef-gensymb}
840 \newif\ifsi@gensymb

```


`\ifsi@old@LITER` The second set are more general functionality.

```
\ifsi@old@liter 841 \si@opt@compatkey{unitsdef}{LITER}
\ifsi@old@noospace 842 \si@opt@compatkey{unitsdef}{liter}
\ifsi@old@noconfig 843 \si@opt@compatkey{unitsdef}{noospace}
844 \si@opt@compatkey{unitsdef}{noconfig}
```

`\ifsi@old@noabbr` The final set are for control of abbreviations, and are a good demonstration of why to use `xkeyval`!

```
\ifsi@old@noamperageabbr
\ifsi@old@nofrequncyabbr 845 \si@opt@compatkey{unitsdef}{noabbr}
\ifsi@old@nomolabbr 846 \si@opt@compatkey{unitsdef}{noampereageabbr}
\ifsi@old@novoltageabbr 847 \si@opt@compatkey{unitsdef}{nofrequncyabbr}
\ifsi@old@novolumeabbr 848 \si@opt@compatkey{unitsdef}{nomolabbr}
\ifsi@old@noweightabbr 849 \si@opt@compatkey{unitsdef}{novoltageabbr}
\ifsi@old@noenergyabbr 850 \si@opt@compatkey{unitsdef}{novolumeabbr}
\ifsi@old@nolengthabbr 851 \si@opt@compatkey{unitsdef}{noweightabbr}
\ifsi@old@notimeabbr 852 \si@opt@compatkey{unitsdef}{noenergyabbr}
853 \si@opt@compatkey{unitsdef}{nolengthabbr}
854 \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 855 \si@opt@compatkey{SIunits}{cdot}
\ifsi@old@mediumspace 856 \si@opt@compatkey{SIunits}{thickspace}
\ifsi@old@thinspace 857 \si@opt@compatkey{SIunits}{mediumspace}
\ifsi@old@thickqspace 858 \si@opt@compatkey{SIunits}{thinspace}
\ifsi@old@mediumqspace 859 \si@opt@compatkey{SIunits}{thickqspace}
\ifsi@old@thingqspace 860 \si@opt@compatkey{SIunits}{mediumqspace}
861 \si@opt@compatkey{SIunits}{thingqspace}
```

`\ifsi@old@amssymb` These options are used by Slunits to control clashes with other packages.

```
\ifsi@old@squaren 862 \si@opt@compatkey{SIunits}{amssymb}
\ifsi@old@pstricks 863 \si@opt@compatkey{SIunits}{squaren}
\ifsi@old@Gray 864 \si@opt@compatkey{SIunits}{pstricks}
\ifsi@old@italian 865 \si@opt@compatkey{SIunits}{Gray}
866 \si@opt@compatkey{SIunits}{italian}
```

`\ifsi@old@textstyle` The miscellaneous options.

```
\ifsi@old@binary 867 \si@opt@compatkey{SIunits}{textstyle}
\ifsi@old@noams 868 \si@opt@compatkey{SIunits}{binary}
\ifsi@old@derivedinbase 869 \si@opt@compatkey{SIunits}{noams}
\ifsi@old@derived 870 \si@opt@compatkey{SIunits}{derivedinbase}
871 \si@opt@compatkey{SIunits}{derived}
```

`\ifsi@old@noprefixcmds` The hepunits package only has one option.

```
872 \si@opt@compatkey{hepunits}{noprefixcmds}
```

`\ifsi@old@english` **fancynum** provides a few options. First the rather oddly named `english` and `french` ones.

```
873 \si@opt@compatkey{fancynum}{english}
874 \si@opt@compatkey{fancynum}{french}
```

`\ifsi@old@tight` A couple for spacing multiplication.

```
\ifsi@old@loose 875 \si@opt@compatkey{fancynum}{tight}
876 \si@opt@compatkey{fancynum}{loose}
```

`\ifsi@old@thinspaces` Three for digit separation.

```

\ifsi@old@commas 877 \si@opt@compatkey{fancynum}{thinspaces}
\ifsi@old@plain 878 \si@opt@compatkey{fancynum}{commas}
879 \si@opt@compatkey{fancynum}{plain}

\ifsi@old@spaceqspace The fancyunits package provides one option not available with Slunits.
880 \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 881 \newcommand*{\si@fix@thin}{\,}
\si@fix@thick 882 \newcommand*{\si@fix@med}{\:}
\si@fix@space 883 \newcommand*{\si@fix@medium}{\:}
884 \newcommand*{\si@fix@thick}{\;}
885 \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 886 \newcommand*{\si@fix@cdot}{{\cdot}}
\si@fix@fullstop 887 \newcommand*{\si@fix@comma}{{,}}
\si@fix@period 888 \newcommand*{\si@fix@stop}{{.}}
\si@fix@times 889 \newcommand*{\si@fix@fullstop}{{.}}
\si@fix@tighttimes 890 \newcommand*{\si@fix@period}{{.}}
\si@fix@tightcdot 891 \newcommand*{\si@fix@times}{\times}
892 \newcommand*{\si@fix@tighttimes}{\bgroup\times\egroup}
893 \newcommand*{\si@fix@tightcdot}{\bgroup\cdot\egroup}

\si@fix@plus Signs for numbers are needed.
\si@fix@minus 894 \newcommand*{\si@fix@plus}{+}
\si@fix@pm 895 \newcommand*{\si@fix@minus}{-}
\si@fix@tightpm 896 \newcommand*{\si@fix@pm}{\pm}
\si@fix@mp 897 \newcommand*{\si@fix@tightpm}{\bgroup\pm\egroup}
898 \newcommand*{\si@fix@mp}{\mp}

\si@fix@two The literals “2” and “10” are needed for exponents.
\si@fix@ten 899 \newcommand*{\si@fix@two}{2}
900 \newcommand*{\si@fix@ten}{10}

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

\si@fix@none Finally for spacing, there is the possibility of nothing at all
902 \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.

```
903 \newcommand*{\si@symbol}[1]{%
904   \expandafter\protected\expandafter\def
905     \csname si@sym@#1\endcsname{%
906     \ifmmode
907       \expandafter\csname si@maths#1\expandafter\endcsname
908     \else
909       \expandafter\csname si@text#1\expandafter\endcsname
910     \fi}}
```

`\si@sym@Omega` The various symbols are now declared.

```
\si@sym@ringA 911 \si@symbol{Omega}
\si@sym@mu     912 \si@symbol{ringA}
\si@sym@degree 913 \si@symbol{mu}
\si@sym@minute 914 \si@symbol{degree}
\si@sym@second 915 \si@symbol{minute}
\si@sym@celsius 916 \si@symbol{second}
\si@sym@celsius 917 \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.

```
918 \AtBeginDocument{%
919   \ifsi@redefsymbols
920     \ifpackageloaded{textcomp}
921       {\si@log@debug{Redefining symbols using textcomp}%
922       \renewcommand*{\si@textdegree}{\textdegree}%
923       \renewcommand*{\si@mathsdegree}{\text{\textdegree}}}%
924     \ifpackageloaded{mathptmx}{}
925     {\renewcommand*{\si@textmu}{\textmu}%
926     \renewcommand*{\si@textOmega}{\textohm}}%
```

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

```
924     \ifpackageloaded{mathptmx}{}
925     {\renewcommand*{\si@textmu}{\textmu}%
926     \renewcommand*{\si@textOmega}{\textohm}}%
927     \renewcommand*{\si@tempa}{OT1}%
928     \ifx\si@tempa\encodingdefault
929       \renewcommand*{\si@mathsringA}{%
930         \text{\capitalring{A}}}%
931       \renewcommand*{\si@textringA}{\capitalring{A}}
932     \fi}{}
933   \ifpackageloaded{upgreek}
934     {\si@log@debug{Redefining symbols using upgreek}%
935     \renewcommand*{\si@mathsmu}{\upmu}%
936     \renewcommand*{\si@mathsOmega}{\Upomega}}{}
937   \fi}
```

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.

21.8 Handling fractions

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

```
\si@frc@hook
\si@frc@frac
\si@frc@slash
\si@frc@nice
\si@frc@sfrac
```

```

938 \newcommand*{\si@frc@frac}[2]{%
939   \ensuremath{\si@frc@hook\frac{%
940     \expandafter\si@unt@out\expandafter{#1}}%
941     {\expandafter\si@unt@out\expandafter{#2}}}}
942 \let\si@frac\si@frc@frac
943 \newcommand*{\si@frc@hook}{}
944 \newcommand*{\si@frc@slash}[2]{%
945   \expandafter\si@unt@out\expandafter{#1}%
946   \si@out{\ensuremath{\si@slash}}}%
947   \expandafter\si@unt@out\expandafter{#2}}
948 \newcommand*{\si@frc@nice}[2]{%
949   \ensuremath{\si@frc@nicefrac{\expandafter\si@unt@out%
950     \expandafter{#1}}{\expandafter\si@unt@out\expandafter
951       {#2}}}}
952 \newcommand*{\si@frc@sfrac}[2]{%
953   \sfrac{\expandafter\si@unt@out\expandafter{#1}}%
954     {\expandafter\si@unt@out\expandafter{#2}}}
955 \AtBeginDocument{
956   \ifpackageloaded{xfrac}
957   {}
958   {\si@log@inf{xfrac package unavailable}\MessageBreak
959     using 'fraction=sfrac' will fall back on\MessageBreak
960     nicefrac-like method}%
961   \renewcommand*{\si@frc@sfrac}[2]{%
962     \si@log@warn{xfrac package unavailable}%
963     \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 964 \newlength\si@frc@displen
\si@frc@textlen 965 \newlength\si@frc@textlen
\si@frc@suplen 966 \newlength\si@frc@suplen
\si@frc@ssuplen 967 \newlength\si@frc@ssuplen
968 \newcommand*{\si@frc@nicefrac}{%
969   \ifmmode
970     \expandafter\si@frc@mathsnf
971   \else
972     \expandafter\si@frc@textnf
973   \fi}

```

`\si@frc@mathsnf` The maths mode system.

```

\si@frc@mathsnf{<numerator>}{<denominator>}
974 \newcommand*{\si@frc@mathsnf}[2]{%
975   \begingroup
976     \settoheight{\si@frc@displen}{\ensuremath{%
977       \displaystyle{M}}}%
978     \settoheight{\si@frc@textlen}{\ensuremath{%
979       \textstyle{M}}}%
980     \settoheight{\si@frc@suplen}{\ensuremath{%
981       \scriptstyle{M}}}%
982     \settoheight{\si@frc@ssuplen}{%

```

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

```

983     \ensuremath{\scriptscriptstyle{M}}}%
984     \addtolength{\si@frc@displen}{-\si@frc@ssuplen}%
985     \addtolength{\si@frc@textlen}{-\si@frc@ssuplen}%
986     \addtolength{\si@frc@suplen}{-\si@frc@ssuplen}%
987     \mathchoice
988       {\raisebox{\si@frc@displen}{\ensuremath{%
989         \scriptstyle{#1}}}}}%
990       {\raisebox{\si@frc@textlen}{\ensuremath{%
991         \scriptstyle{#1}}}}}%
992       {\raisebox{\si@frc@suplen}{%
993         \ensuremath{\scriptscriptstyle{#1}}}}}%
994       {\raisebox{\si@frc@ssuplen}{%
995         \ensuremath{\scriptscriptstyle{#1}}}}}%
996     \mkern-2mu\relax/\mkern-1mu\relax
997     \bgroup
998     \mathchoice
999       {\scriptstyle}%
1000       {\scriptstyle}%
1001       {\scriptscriptstyle}%
1002       {\scriptscriptstyle}%
1003     {#2}%
1004     \egroup
1005   \endgroup}

```

`\si@frc@textnf` A stripped down version of the nicefrac system for text mode.

```

\si@frc@textnf{\langle numerator\rangle}{\langle denominator\rangle}
1006 \newcommand*\si@frc@textnf[2]{%
1007   \begingroup
1008     \settoheight{\si@frc@textlen}{M}%
1009     \settoheight{\si@frc@ssuplen}{\fontsize\sf@size\z@\relax
1010       \selectfont{M}}}%
1011     \addtolength{\si@frc@textlen}{-\si@frc@ssuplen}%
1012     \raisebox{\si@frc@textlen}{\fontsize\sf@size\z@\relax
1013       \selectfont{#1}}}%
1014     \hspace{-0.25ex}/\hspace{-0.25ex}%
1015     \hbox{\fontsize\sf@size\z@\selectfont{#2}}}%
1016   \endgroup}

```

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

```

\si@frc@ugly{\langle numerator\rangle}
1017 \newcommand*\si@frc@ugly[1]{%
1018   \renewcommand*\si@tempa{#1}%
1019   \ifmmode
1020     \expandafter\si@frc@frac
1021   \else
1022     \renewcommand*\si@tempb{1}%
1023     \ifx\si@tempa\si@tempb

```

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

```

1024     \setbox\si@tempboxa=\hbox{\ensuremath{\si@valuesep}}}%
1025     \hskip-\wd\si@tempboxa\relax
1026     \renewcommand*\si@tempa{}%

```

```

1027     \fi
1028     \expandafter\si@frc@slash
1029     \fi
1030     {\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 `Slstyle`; 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`.

```

1031 \g@addto@macro{\document}{%
1032   \ifdefined\mathsf
1033     \setbox\si@tempboxa=\hbox{%
1034       $\mathsf{\global\chardef\si@fam@sf=\fam}$}%
1035   \else
1036     \si@log@inf{\string\mathsf not found}%
1037     \global\chardef\si@fam@sf=99\relax
1038   \fi
1039   \ifdefined\mathtt
1040     \setbox\si@tempboxa=\hbox{%
1041       $\mathtt{\global\chardef\si@fam@tt=\fam}$}%
1042   \else
1043     \si@log@inf{\string\mathtt not found}%
1044     \global\chardef\si@fam@tt=99\relax
1045   \fi}

```

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

```

\si@fam@ifbmaths \si@fam@ifbtext{\code}
\si@fam@ifbmaths \si@fam@ifbmaths{\code}

1046 \newcommand*\si@fam@ifbtext[1]{%
1047   \if b\expandafter\@car\@fseries\@nil
1048     #1\fi}
1049 \newcommand*\si@fam@ifbmaths[1]{%
1050   \renewcommand*\si@tempa{bold}%
1051   \ifx\math@version\si@tempa
1052     #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.

```

1053 \newcommand*\si@fam@ifbinline{%
1054   \ifsi@inlinebtext
1055     \expandafter\si@fam@ifbtext
1056   \else
1057     \expandafter\si@fam@ifbmaths
1058   \fi}

```

`\si@fam@ifitext` This test check for italic or slanted text in text mode, by negation (upright text is n).

```
\si@fam@ifitext{\code}
1059 \newcommand*{\si@fam@ifitext}[1]{%
1060   \if n\expandafter\@car\f@series\@nil\else
1061     #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.

```
1062 \newcommand*{\si@fam@mode}{%
1063   \ifsi@obeymode
1064     \ifmmode
1065       \sisetup{mode=maths}%
1066     \else
1067       \sisetup{mode=text}%
1068     \fi
1069   \fi}
```

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

```
1070 \AtBeginDocument{
1071   \@ifpackageloaded{color}
1072     {\let\si@fam@colourcmd\color}
1073     {\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.

```
1074 \newif\ifsi@fam@set
1075 \newif\ifsi@textmode
```

`\si@fam@set` Using the code from `Sstyle` 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.

```
1076 \newcommand*{\si@fam@set}{%
1077   \ifsi@out@num
1078     \ifsi@numtextmode
1079       \expandafter\expandafter\expandafter\si@textmodetrue
1080     \else
1081       \expandafter\expandafter\expandafter\si@textmodefalse
1082     \fi
1083   \else
1084     \ifsi@unittextmode
1085       \expandafter\expandafter\expandafter\si@textmodetrue
1086     \else
1087       \expandafter\expandafter\expandafter\si@textmodefalse
1088     \fi
1089   \fi}
```

`\si@mathsrms` The appropriate font macros are now established, if necessary.

```
\si@mathssfi1090 \ifsi@fam@set\else
\si@mathstt1091   \let\si@colourcmd\@gobble
\si@textrm1092   \ifsi@out@num
\si@textsf1093   \let\si@mathsrms\si@valuemathsrms
\si@texttt1094   \let\si@mathssf\si@valuemathssf
\si@colourcmd1095   \let\si@mathstt\si@valuemathstt
\si@colour
```

```

1096     \let\si@textrm\si@valuetextrm
1097     \let\si@textsf\si@valuetextsf
1098     \let\si@texttt\si@valuetexttt
1099     \ifsi@colourvalues
1100         \let\si@colourcmd\si@fam@colourcmd
1101     \fi
1102     \let\si@colour\si@valuecolour
1103 \else
1104     \let\si@mathsrms\si@unitmathsrms
1105     \let\si@mathssf\si@unitmathssf
1106     \let\si@mathstt\si@unitmathstt
1107     \let\si@textrm\si@unittextrm
1108     \let\si@textsf\si@unittextsf
1109     \let\si@texttt\si@unittexttt
1110     \ifsi@colourunits
1111         \let\si@colourcmd\si@fam@colourcmd
1112     \fi
1113     \let\si@colour\si@unitcolour
1114 \fi
1115 \fi
1116 \si@fam@settrue

```

The temporary macros are needed for the `\ifx` tests, which need to be expanded once.

```

1117 \edef\si@tempa{\sfdefault}%
1118 \edef\si@tempb{\ttdefault}%

```

`\si@fam@maths` The surrounding font family is only tested if matching is requested. First, the defaults are set up assuming no detection takes place.

```

\si@fam@text
1119 \expandafter\let\expandafter\si@fam@maths
1120     \csname\si@mathsrms\endcsname
1121 \expandafter\let\expandafter\si@fam@text
1122     \csname\si@textrm\endcsname
1123 \ifsi@obeyfamily
1124     \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.

```

1125 \ifmmode
1126     \ifinner
1127         \si@log@debug{Font detection: inline maths}%
1128         \si@fam@detttext
1129     \else
1130         \si@log@debug{Font detection: display maths}%
1131         \ifsi@detectdisplay
1132             \si@fam@detmaths
1133         \else
1134             \si@fam@detttext
1135         \fi
1136     \fi
1137 \else
1138     \si@log@debug{Font detection: text}%

```



```

1139     \si@fam@detttext
1140   \fi
1141 \else
1142   \si@log@debug{Font detection: inactive}%
1143 \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.

```

1144 \def\si@fam@bold{\unboldmath\mdseries}%
1145 \ifsi@obeybold
1146   \si@log@debug{Weight detection: checking weight}%
1147   \ifmmode
1148     \ifdim\displaywidth>0pt\relax
1149       \ifsi@detectdisplay
1150         \expandafter\si@fam@ifbmaths
1151       \else
1152         \expandafter\si@fam@ifbtext
1153       \fi
1154       \si@fam@setbold
1155     \else
1156       \si@fam@ifbinline\si@fam@setbold
1157     \fi
1158   \else
1159     \si@fam@ifbtext\si@fam@setbold
1160   \fi
1161 \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.

```

1162 \let\si@fam@italic\upshape
1163 \ifsi@obeyitalic
1164   \si@log@debug{Italic detection: checking italic}%
1165   \si@fam@ifitext
1166   {\let\si@fam@italic\relax
1167     \si@log@debug{Italic detection: italic}}%
1168 \fi}

```

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

```

\si@fam@detttext
1169 \newcommand*{\si@fam@detmaths}{%
1170   \ifnum\the\fam=\si@fam@sf
1171     \si@log@debug{Font detection: sf}%
1172     \expandafter\let\expandafter\si@fam@maths
1173       \csname\si@mathssf\endcsname
1174     \expandafter\let\expandafter\si@fam@text
1175       \csname\si@textsf\endcsname
1176   \else
1177     \ifnum\the\fam=\si@fam@tt
1178       \si@log@debug{Font detection: tt}%
1179       \expandafter\let\expandafter\si@fam@maths
1180         \csname\si@mathstt\endcsname

```

```

1181     \expandafter\let\expandafter\si@fam@text
1182     \csname\si@textttt\endcsname
1183   \else
1184     \si@log@debug{Font detection: rm}%
1185     \expandafter\let\expandafter\si@fam@maths
1186     \csname\si@mathsrms\endcsname
1187     \expandafter\let\expandafter\si@fam@text
1188     \csname\si@textrm\endcsname
1189   \fi
1190 \fi}
1191 \newcommand*{\si@fam@detttext}{%
1192   \ifx\f@family\si@tempa
1193     \si@log@debug{Font detection: sf}%
1194     \expandafter\let\expandafter\si@fam@maths
1195     \csname\si@mathssf\endcsname
1196     \expandafter\let\expandafter\si@fam@text
1197     \csname\si@textsf\endcsname
1198   \else
1199     \ifx\f@family\si@tempb
1200       \si@log@debug{Font detection: tt}%
1201       \expandafter\let\expandafter\si@fam@maths
1202       \csname\si@mathstt\endcsname
1203       \expandafter\let\expandafter\si@fam@text
1204       \csname\si@textttt\endcsname
1205     \else
1206       \si@log@debug{Font detection: rm}%
1207       \expandafter\let\expandafter\si@fam@maths
1208       \csname\si@mathsrms\endcsname
1209       \expandafter\let\expandafter\si@fam@text
1210       \csname\si@textrm\endcsname
1211     \fi
1212   \fi}

```

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

```

\si@fam@boldify1213 \newcommand*{\si@fam@setbold}{%
1214   \si@log@debug{Weight detection: bold weight}%
1215   \let\si@fam@bold\si@fam@boldify}
1216 \newcommand*{\si@fam@boldify}{\boldmath\bfseries}

```

21.10 Formatting numbers

`\num` 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.

```

\num[<options>]{<num>}
1217 \si@newrobustcmd*{\num}[2][]{%
1218   \beginngroup
1219     \sisetup{#1}%
1220     \si@fam@mode
1221     \si@num@intabfalse

```

```

1222 \si@log@debug{Processing \string\num\space input `#2'}%
1223 \expandafter\si@out@num\expandafter{\si@num{#2}}%
1224 \endgroup}

```

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

```

1225 \newif\ifsi@num@intab

```

`\si@num` 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.

`\si@num{<num>}`

```

1226 \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.

```

1227 \si@num@fixpm
1228 \begingroup
1229 \makeatletter
1230 \@makeother{,}%
1231 \@makeother{.}%
1232 \@makeother{+}%
1233 \@makeother{-}%
1234 \def~{}%
1235 \def\,{}%
1236 \catcode'\~= \active\relax
1237 \catcode'\^= \active\relax
1238 \everyeof{\noexpand}%
1239 \endlinechar\m@ne
1240 \protected@xdef\si@tempa{\scantokens{#1}}%
1241 \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.

```

1242 \si@ifnotmtarg{\si@tempa}
1243 {\si@num@ifvalid{\si@tempa}
1244 {\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.

```

1245 {\si@log@err{Invalid character `#1' in numerical input}%
1246 {Only characters from the list
1247 '\si@numvalid'\MessageBreak should be present in the
1248 argument of the \string\num\space macro\MessageBreak
1249 (or derivative such as an 's' column)}}%
1250 {#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.

```

\si@num@pm
\si@num@mp
\pm
\mp

```

```

1251 \newcommand*{\si@num@fixpm}{%
1252   \let\si@num@pm\pm
1253   \let\si@num@mp\mp
1254   \protected\def\pm{\si@num@pm}%
1255   \protected\def\mp{\si@num@mp}}

```

`\si@num@ifvalid` 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.

```

\si@num@valid
\si@num@ifvalid{<chars>}
\si@num@valid<char><chars>\@empty

1256 \newcommand*{\si@num@ifvalid}[1]{%
1257   \begingroup
1258     \si@switchtrue
1259     \expandafter\si@num@valid#1\@empty\@empty
1260     \ifsi@switch
1261       \aftergroup\@firstoftwo
1262     \else
1263       \aftergroup\@secondoftwo
1264     \fi
1265   \endgroup}
1266 \def\si@num@valid#1#2\@empty{%
1267   \si@str@ifchrstr{#1}{\si@num@valid}
1268   {\ifx\@empty#2\@empty\else
1269     \si@num@valid#2\@empty\@empty\@empty
1270   \fi}
1271   {\si@switchfalse}}

```

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

```

\si@num@out1272 \newcommand*{\si@num@in}{}
\si@num@exp1273 \newcommand*{\si@num@out}{}
\si@num@expsign1274 \newcommand*{\si@num@exp}{}
\si@num@mant1275 \newcommand*{\si@num@expsign}{}
\si@num@mantsign1276 \newcommand*{\si@num@mant}{}
\si@num@err1277 \newcommand*{\si@num@mantsign}{}
\si@num@xpart1278 \newcommand*{\si@num@err}{}
\si@num@ambig1279 \newcommand*{\si@num@xpart}{}
\si@tab@out1280 \newcommand*{\si@num@ambig}{}
\si@tab@expout1281 \newcommand*{\si@tab@out}{}
\si@tab@expout1282 \newcommand*{\si@tab@expout}{}

```

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

```

1283 \newif\ifsi@num@erropen

```

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

```

\si@num@arg
\si@num@format{<num>}

1284 \newcommand*{\si@num@arg}{}
1285 \newcommand*{\si@num@format}[1]{%
1286   \protected@edef\si@num@arg{#1}%
1287   \si@log@debug{Formatting number '\si@num@arg'}%

```

The storage areas are emptied.

```

1288 \renewcommand*{\si@num@in}{}%
1289 \renewcommand*{\si@num@exp}{}%
1290 \renewcommand*{\si@num@expsign}{}%
1291 \renewcommand*{\si@num@mant}{}%
1292 \renewcommand*{\si@num@mantsign}{}%
1293 \renewcommand*{\si@num@err}{}%
1294 \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`.

```

1295 \si@switchfalse
1296 \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`.

```

1297 \si@switchfalse
1298 \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}).

```

1299 \si@num@sepsign{exp}%
1300 \si@num@sepsign{mant}%
1301 \ifx\@empty\si@num@exp\@empty
1302   \ifx\@empty\si@num@expsign\@empty\else
1303     \si@log@warn{Sign but no number for '\si@num@arg'}%
1304   \fi
1305   \let\si@num@expsign\@empty
1306 \fi
1307 \ifx\@empty\si@num@mant\@empty
1308   \ifx\@empty\si@num@mantsign\@empty\else
1309     \ifx\@empty\si@num@exp\@empty
1310       \si@log@warn{Sign but no number for '\si@num@arg'}%
1311       \let\si@num@mantsign\@empty
1312     \fi
1313   \fi
1314 \fi

```

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

```

1315 \renewcommand*{\si@tempa}{\{-}}%
1316 \ifx\si@num@mantsign\si@tempa
1317   \ifsi@colourneg
1318     \expandafter\expandafter\expandafter\si@fam@colourcmd
1319   \else
1320     \expandafter\expandafter\expandafter\@gobble
1321   \fi
1322 \else
1323   \expandafter\@gobble
1324 \fi
1325 {\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

```
1326 \si@num@procnum{exp}%
1327 \si@num@procnum{mant}%
```

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

```
1328 \si@str@ifonlychrs{\si@num@exp}{0\si@numdecimal}
1329 {\ifsi@allowzeroexp\else
1330 \renewcommand*{\si@num@exp}{}%
1331 \ifx\@empty\si@num@mant\@empty
1332 \renewcommand*{\si@num@mant}{1}%
1333 \fi
1334 \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.

```
1335 \ifx\@empty\si@num@exp\@empty\else
1336 \ifsi@trapambigerr
1337 \expandafter\expandafter\expandafter\si@num@ambigertrue
1338 \fi
1339 \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 \i@num@out. \si@tab@out On the other hand, under normal circumstances the entire mantissa should be copied to \si@num@out.

```
1340 \protected@edef\si@num@out{%
1341 \ensuremath{\si@num@mantsign}\si@num@mant}%
1342 \renewcommand*{\si@tempa}{num}%
1343 \ifsi@num@intab
1344 \protected@edef\si@tab@out{%
1345 \ensuremath{\si@num@mantsign}\si@num@predec}%
1346 \protected@edef\si@num@out{\si@num@postdec}%
1347 \renewcommand*{\si@tempa}{tab}%
1348 \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.

```
1349 \ifx\@empty\si@num@postdec\@empty\else
1350 \renewcommand*{\si@tempa}{num}%
1351 \fi
1352 \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.

```
1353 \ifsi@seperr
1354 \ifsi@num@ambigerr
1355 \protected@edef\si@num@out{%
```

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

```

1356         \ensuremath{\si@openerr}\si@num@out}%
1357         \si@repeatunitsfalse
1358         \expandafter\si@num@erropentrue
1359     \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.

```

1360         \ifsi@trapambigerr\else
1361         \ifx\@empty\si@num@exp\@empty\else
1362             \protected@edef\si@num@err{%
1363                 \si@num@err\expandafter\car\si@num@exp\@nil
1364                 \si@num@expsign\si@num@exp}%
1365             \ifsi@repeatunits\else
1366                 \renewcommand*\si@num@exp{}}%
1367                 \renewcommand*\si@num@expsign{}}%
1368             \fi
1369         \fi
1370     \fi
1371 \fi

```

If *sepperr* 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.

```

1372 \else
1373     \expandafter\protected@edef\csname
1374         si@\si@tempa @out\endcsname{%
1375         \si@num@out\ensuremath{\si@errspace}\ensuremath
1376         {\si@openerr}\si@num@err\ensuremath{\si@closeerr}}%
1377     \renewcommand*\si@num@err{}}%
1378 \fi
1379 \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.

```

1380 \renewcommand*\si@tempa{num@out}%
1381 \ifsi@num@erropen
1382     \renewcommand*\si@tempa{num@ambig}%
1383 \fi
1384 \ifsi@num@intab
1385     \renewcommand*\si@tempa{tab@expout}%
1386 \fi
1387 \ifx\@empty\si@num@exp\@empty
1388     \ifx\@empty\si@num@mant\@empty
1389         \si@log@err{Invalid number format '\si@num@arg'}
1390         {Something is wrong with the number format; does it
1391         contain \MessageBreak any numbers (from the list
1392         '\si@numdigits')?}%
1393     \renewcommand*\si@num@out{}}%
1394 \fi
1395 \else
1396     \ifx\@empty\si@num@mant\@empty\else
1397         \expandafter\protected@edef\csname
1398             si@\si@tempa\endcsname{%
1399             \csname si@\si@tempa\endcsname\ensuremath{}}%
1400         \si@expproduct{}}}%

```

```

1401 \fi
1402 \expandafter\protected@edef\csname
1403   si@\si@tempa\endcsname{%
1404     \csname si@\si@tempa\endcsname\si@exbase
1405     \textsuperscript{\ensuremath{\si@num@expsign}}%
1406     \si@num@exp}}%
1407 \fi

```

With everything done, the result is output.

```

1408 \ifsi@num@intab\else
1409   \expandafter\si@num@out
1410 \fi

```

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

```

1411 \ifx\@empty\si@num@err\@empty\else
1412   \expandafter\si@num@procerr
1413 \fi
1414 \ifsi@num@erropen
1415   \expandafter\si@out@num\expandafter{%
1416     \ensuremath{\si@closeerr}}%
1417   \ifx\@empty\si@num@ambig\@empty\else
1418     \expandafter\si@out@num\expandafter{\si@num@ambig}%
1419     \renewcommand*{\si@num@ambig}{}%
1420   \fi
1421 \fi
1422 \si@num@erropenfalse
1423 \ifx\@empty\si@num@xpart\@empty\else
1424   \expandafter\si@num@sepxpart
1425 \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.

```

1426 \def\si@num@findxpart#1#2\@empty{%
1427   \si@str@ifchrstr{#1}{\si@numprod}
1428   {\si@switchtrue\si@seperrfalse}}%
1429 \ifsi@switch
1430   \protected@edef\si@num@xpart{\si@num@xpart#1}%
1431 \else
1432   \protected@edef\si@num@in{\si@num@in#1}%
1433 \fi
1434 \ifx\@empty#2\@empty\else
1435   \si@num@findxpart#2\@empty
1436 \fi}

```

`\si@num@sepmantexp` 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.

```

\si@num@sepmantexp{<num>}
\si@num@mantexp{<char>{<chars>}\@empty
1437 \newcommand*{\si@num@sepmantexp}[1]{%

```



```

1438 \expandafter\si@num@mantexp#1\@empty\@empty}
1439 \def\si@num@mantexp#1#2\@empty{%
1440 \si@str@ifchrstr{#1}{\si@numgobble}
1441 {\si@log@debug{Gobbling '#1' in \si@num@arg}}
1442 {\si@str@ifchrstr{#1}{\si@numexp}
1443 {\ifsi@switch
1444 \si@log@err{Duplicate exponent marker found}
1445 {Only a single exponent character \MessageBreak
1446 (from the list '\si@numexp')\MessageBreak may
1447 occur in a numerical argument}%
1448 \else
1449 \si@log@debug{Exponent marker '#1' found in
1450 '\si@num@arg'}%
1451 \fi
1452 \si@switchtrue}%
1453 {\ifsi@switch
1454 \expandafter\si@num@addexp
1455 \else
1456 \expandafter\si@num@addmnt
1457 \fi
1458 {#1}}}%

```

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

```

1459 \ifx\@empty#2\@empty
1460 \expandafter\@gobble
1461 \else
1462 \expandafter\si@num@sepmantexp
1463 \fi
1464 {#2}}

```

\si@num@addmnt To allow \expandafter use in the above, the actual addition to the appropriate macro is handled here.

```

\si@num@addexp \si@num@addmnt {\char}
\si@num@addmntexp \si@num@addexp {\char}
\si@num@addmntexp {\char} {\mant/exp} {\text}
1465 \newcommand*\si@num@addmnt [1]{%
1466 \si@num@addmntexp{#1}{mant}{mantissa}}
1467 \newcommand*\si@num@addexp [1]{%
1468 \si@num@addmntexp{#1}{exp}{exponent}}
1469 \newcommand*\si@num@addmntexp [3]{%
1470 \si@log@debug{Adding '#1' to #3 for '\si@num@arg'}%
1471 \expandafter\protected@edef\csname si@num@#2\endcsname{%
1472 \csname si@num@#2\endcsname#1}}

```

\si@num@sepsign 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.

```

\si@num@sepsign {\mant/exp}
1473 \newcommand*\si@num@sepsign [1]{%
1474 \expandafter\ifx\expandafter\@empty
1475 \csname si@num@#1\endcsname\@empty
1476 \expandafter\@gobble
1477 \else

```

```

1478 \expandafter\si@num@gensign
1479 \fi
1480 {#1}}

```

\si@num@gensign 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.

```

\si@num@gensign{<mant/exp>}
1481 \newcommand*{\si@num@gensign}[1]{%
1482 \expandafter\expandafter\expandafter\si@num@findsign
1483 \csname si@num@#1\endcsname\@empty\@empty

```

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

```

1484 \ifx\@empty\si@num@sign\@empty
1485 \ifx\@empty\si@num@value\@empty
1486 \expandafter\expandafter\expandafter\@gobble
1487 \else
1488 \expandafter\expandafter\expandafter\si@num@addsign
1489 \fi
1490 \else
1491 \expandafter\@gobble
1492 \fi
1493 {#1}%

```

The appropriate storage areas are now assigned.

```

1494 \expandafter\let\csname si@num@#1sign\endcsname\si@num@sign
1495 \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.

```

\si@num@findsign<char><char><chars>\@empty
1496 \newcommand*{\si@num@sign}{}
1497 \def\si@num@findsign#1#2#3\@empty{%
1498 \si@num@delplusfalse
1499 \si@str@ifchrstr{#1}{\si@numsign}{%
1500 \si@str@ifchrstr{#2}{\si@numsign}{%
1501 \if +#1%
1502 \if -#2%
1503 \si@log@debug{Found sign combination +- for
1504 '\si@num@arg'}%
1505 \renewcommand*{\si@num@sign}{\si@pm}}%
1506 \else
1507 \si@log@inf{Unknown sign combination '#1#2'}%
1508 \renewcommand*{\si@num@sign}{\si@#1#2}}%
1509 \fi
1510 \else
1511 \if -#1%
1512 \if +#2%
1513 \si@log@debug{Found sign combination -+ for
1514 '\si@num@arg'}%
1515 \renewcommand*{\si@num@sign}{\si@mp}}%

```

```

1516         \else
1517         \si@log@inf{Unknown sign combination `#1#2'}%
1518         \renewcommand*{\si@num@sign}{{#1#2}}%
1519         \fi
1520     \else
1521     \si@log@inf{Unknown sign combination `#1#2'}%
1522     \renewcommand*{\si@num@sign}{{#1#2}}%
1523     \fi
1524 \fi
1525 \protected@edef\si@num@value{#3}}%

```

Only one valid sign character.

```

1526     {\si@log@debug{Found single sign character `#1' for
1527     '\si@num@arg'}%
1528     \renewcommand*{\si@num@sign}{{#1}}%
1529     \if +#1%
1530     \ifsi@retainplus\else
1531     \expandafter\expandafter\expandafter\si@num@killsign
1532     \fi
1533     \fi
1534     \protected@edef\si@num@value{#2#3}}}%

```

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

```

1535     {\si@log@debug{No sign found for '\si@num@arg'}%
1536     \renewcommand*{\si@num@sign}{}%
1537     \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.

```

1538 \newif\ifsi@num@delplus
1539 \newcommand*{\si@num@killsign}{%
1540   \si@num@delplustrue
1541   \renewcommand*{\si@num@sign}{}%

```

`\si@num@addsign` 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.

`\si@num@assign`

```

\si@num@addsign{<mant/exp>}
\si@num@assign{<mantissa/exponent>}
1542 \newcommand*{\si@num@addsign}[1]{%
1543   \begingroup
1544     \renewcommand*{\si@tempa}{#1}%
1545     \renewcommand*{\si@tempb}{mant}%
1546     \ifx\si@tempa\si@tempb
1547       \aftergroup\@firstoftwo
1548     \else
1549       \aftergroup\@secondoftwo
1550     \fi
1551   \endgroup
1552   {\ifsi@num@signmant
1553     \expandafter\si@num@assign
1554   \else
1555     \expandafter\@gobble
1556   \fi

```

```

1557 {mantissa}}
1558 {\ifsi@num@signexp
1559   \expandafter\si@num@assign
1560   \else
1561     \expandafter\@gobble
1562   \fi
1563 {exponent}}}}
1564 \newcommand*{\si@num@assign}[1]{%
1565   \let\si@num@sign\si@sign
1566   \si@log@debug{Adding sign \si@sign\space to #1 for
1567     '\si@num@arg' }}

```

`\si@num@procnum` The control macro for processing the number (plus any extra characters).
`\si@num@finddigits{<mant/exp>}`

```

1568 \newcommand*{\si@num@procnum}[1]{%
1569   \expandafter\ifx\expandafter\@empty
1570     \csname si@num@#1\endcsname\@empty
1571     \expandafter\@gobble
1572   \else
1573     \expandafter\si@num@finddigits
1574   \fi
1575 {#1}}

```

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

```

\si@num@postdec 1576 \newcommand*{\si@num@predec}{}
1577 \newcommand*{\si@num@postdec}{}

```

`\si@num@finddigits` 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.

```

\si@num@finddigits{<mant/exp>}
1578 \newcommand*{\si@num@finddigits}[1]{%
1579   \renewcommand*{\si@num@predec}{}%
1580   \renewcommand*{\si@num@postdec}{}%
1581   \si@switchfalse
1582   \expandafter\expandafter\expandafter\si@num@digits
1583   \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.

```

1584   \ifx\@empty\si@num@predec\@empty
1585     \ifsi@num@padlead
1586       \expandafter\expandafter\expandafter\si@num@addprezero
1587     \fi
1588   \fi
1589   \ifx\@empty\si@num@postdec\@empty
1590     \ifsi@num@padtrail
1591       \ifsi@switch
1592         \expandafter\expandafter\expandafter\expandafter
1593         \expandafter\expandafter\expandafter
1594         \si@num@addpostzero
1595       \fi
1596     \fi
1597   \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.

```

1598 \si@num@unsign{#1}%
1599 \ifx\@empty\si@num@predec\@empty
1600 \else
1601 \expandafter\si@num@nozero
1602 \fi

```

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

```

1603 \ifx\@empty\si@num@predec\@empty
1604 \ifx\@empty\si@num@postdec\@empty
1605 \expandafter\expandafter\expandafter\@gobble
1606 \else
1607 \expandafter\expandafter\expandafter\si@num@sepdigits
1608 \fi
1609 \else
1610 \expandafter\si@num@sepdigits
1611 \fi
1612 {#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. \si@num@digits⟨char⟩⟨chars⟩\@empty

\si@num@pre \si@num@post \si@num@prepost

```

\si@num@pre{⟨num⟩}
\si@num@pre{⟨num⟩}
\si@num@prepost{⟨num⟩}{⟨pre/post⟩}{⟨text⟩}

```

```

1613 \def\si@num@digits#1#2\@empty{%
1614 \si@str@ifchrstr{#1}{\si@numdecimal}
1615 {\ifsi@switch
1616 \si@log@err{Duplicate decimal marker in '\si@num@arg'}
1617 {Only a single decimal marker (from the list
1618 '\si@numdecimal')\MessageBreak may occur in a
1619 numerical argument}%
1620 \else
1621 \si@log@debug{Found decimal marker '#1' in
1622 '\si@num@arg'}%
1623 \expandafter\si@switchtrue
1624 \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.

```

1625 {\si@str@ifchrstr{#1}{\si@numsign}
1626 {\si@log@err{Misplaced sign character
1627 '#1' in '\si@num@arg'}
1628 {Sign characters '\si@numsign' can only
1629 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.

```

1630 {\ifsi@switch
1631 \expandafter\si@num@post

```

```

1632         \else
1633         \expandafter\si@num@pre
1634         \fi
1635         {#1}}}%
1636 \ifx\@empty#2\@empty\else
1637     \si@num@digits#2\@empty\@empty
1638 \fi}
1639 \newcommand*{\si@num@pre}[1]{%
1640     \si@num@prepost{#1}{pre}{integer}}
1641 \newcommand*{\si@num@post}[1]{%
1642     \si@num@prepost{#1}{post}{decimal}}
1643 \newcommand*{\si@num@prepost}[3]{%
1644     \expandafter\protected@edef\csname si@num@#2dec\endcsname{%
1645         \csname si@num@#2dec\endcsname#1}%
1646     \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@addpostzero \si@num@addpzero{<pre/post>}{<text>}
\si@num@addpzero1647 \newcommand*{\si@num@addprezero}{%
1648     \si@num@addpzero{pre}{leading}}
1649 \newcommand*{\si@num@addpostzero}{%
1650     \si@num@addpzero{post}{trailing}}
1651 \newcommand*{\si@num@addpzero}[2]{%
1652     \si@log@debug{Adding #2 zero for '\si@num@arg'}}%
1653 \@namedef{si@num@#1dec}{0}}

```

`\si@num@unsign` 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.

```

\si@num@unsign{<mant/exp>}
\si@num@nosign{<mant/exp>}
1654 \newcommand*{\si@num@unsign}[1]{%
1655     \expandafter\ifx\expandafter\@empty
1656         \csname si@num@#1sign\endcsname\@empty
1657         \expandafter\@gobble
1658     \else
1659         \expandafter\si@num@nosign
1660     \fi
1661     {#1}}
1662 \newcommand*{\si@num@nosign}[1]{%
1663     \begingroup
1664         \si@switchtrue
1665         \si@str@ifonlychrs{\si@num@predec\si@num@postdec}{0}
1666         {\si@switchfalse}}}%
1667 \ifsi@switch
1668     \aftergroup\@gobble
1669 \else
1670     \aftergroup\@firstofone
1671 \fi
1672 \endgroup
1673 {\si@log@debug{Zero value: removing any sign}}%
1674 \ifsi@ang@sign\else
1675     \@namedef{si@num@#1sign}{}%
1676 \fi}}

```

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

```

1677 \newcommand*{\si@num@nozero}{%
1678   \si@str@ifonlychrs{\si@num@predec}{0}
1679   {\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.

```

1680 \newcommand*{\si@num@decimalhook}{}

```

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

```

\si@num@sepdigits{<mant/exp>}
1681 \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.

```

1682   \begingroup
1683     \let\si@numextra\si@numaddn
1684     \protected@edef\si@tempa{\si@num@predec\si@num@postdec}%
1685     \si@num@ifextra{\si@tempa}
1686       {\aftergroup\@gobble}
1687       {\aftergroup\@firstofone}%
1688   \endgroup

```

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

```

1689   {\renewcommand*{\si@tempb}{mant}}%
1690   \renewcommand*{\si@tempc}{#1}%
1691   \ifx\si@tempb\si@tempc
1692     \expandafter\si@num@checkerr
1693     \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.

```

1694   \protected@edef\si@tempa{\si@num@predec\si@num@postdec}%
1695   \expandafter\si@str@ifonlychrs\expandafter{\si@tempa}
1696   {0123456789}
1697   {\ifx\@empty\si@num@err\@empty
1698     \ifsi@fixdp
1699       \expandafter\expandafter\expandafter\si@num@fixdp
1700       \fi
1701     \fi}{}}%

```

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

```

1702   \si@num@ifextra{\si@num@predec}{}
1703   {\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.

```

1704   \renewcommand*{\si@tempc}{}%
1705   \ifx\@empty\si@num@postdec\@empty\else
1706     \si@num@decimalhook
1707     \renewcommand*{\si@tempc}{%
1708       \ensuremath{{\si@decimalsymbol}}}%

```

```

1709     \si@num@ifextra{\si@num@postdec}{}
1710     {\expandafter\si@num@dec\expandafter{\si@num@postdec}}%
1711 \fi

```

The construction is finalised by re-combining the number.

```

1712 \expandafter\protected@edef\csname si@num#1\endcsname
1713 {\si@num@predec\si@tempc\si@num@postdec}

```

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

```

\si@num@extra
\si@num@ifextra{<integer>}
\si@num@extra<char><chars>\@empty
1714 \newcommand*{\si@num@ifextra}[1]{%
1715 \begingroup
1716 \si@switchfalse
1717 \expandafter\si@num@extra#1\@empty\@empty
1718 \ifsi@switch
1719 \si@log@debug{Found ‘extra’ characters in ‘#1’}%
1720 \aftergroup\@firstoftwo
1721 \else
1722 \aftergroup\@secondoftwo
1723 \fi
1724 \endgroup}
1725 \def\si@num@extra#1#2\@empty{%
1726 \ifx\@empty#1\@empty\else
1727 \si@str@ifchrstr{#1}{\si@num@extra}{\si@switchtrue}}%
1728 \ifx\@empty#2\@empty\else
1729 \si@num@extra#2\@empty\@empty
1730 \fi
1731 \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.

`\si@num@checkerr`

```

1732 \newif\ifsi@num@ambigerr
1733 \newcommand*{\si@num@checkerr}{%
1734 \ifx\@empty\si@num@postdec\@empty
1735 \expandafter\si@num@preerr
1736 \else
1737 \expandafter\si@num@posterr
1738 \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`.

```

1739 \newcommand*{\si@num@preerr}{%
1740 \si@num@seperr{pre}%
1741 \ifx\@empty\si@tempb\@empty\else
1742 \expandafter\renewcommand\expandafter*\expandafter
1743 \si@num@err\expandafter{\si@tempb}%
1744 \fi}

```


`\si@num@posterr` Life is more complex for an error in the decimal part. This is found, then it may
`\si@num@psterr` 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.

```

1745 \newcommand*{\si@num@posterr}{%
1746   \si@num@seperr{post}%
1747   \ifx\@empty\si@tempb\@empty\else
1748     \ifsi@seperr
1749       \expandafter\expandafter\expandafter\si@num@psterr
1750     \else
1751       \let\si@num@err\si@tempb
1752     \fi
1753   \fi}
1754 \newcommand*{\si@num@psterr}{%
1755   \si@num@cndigits{\si@tempb}%
1756   \si@tempcntb\si@tempcnta\relax
1757   \si@num@cndigits{\si@num@postdec}%
1758   \ifnum\si@tempcnta<\si@tempcntb\relax
1759     \expandafter\si@num@largeerr
1760   \else
1761     \expandafter\si@num@smallerr
1762   \fi}

```

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

```

\si@num@finderr \si@num@seperr{<pre/post>}
\si@num@finderr <char><chars>\@empty
1763 \newcommand*{\si@num@seperr}[1]{%
1764   \si@switchfalse
1765   \renewcommand*{\si@tempa}{}%
1766   \renewcommand*{\si@tempb}{}%
1767   \expandafter\expandafter\expandafter\si@num@finderr
1768   \csname si@num@#1dec\endcsname\@empty\@empty
1769   \ifx\@empty\si@tempb\@empty\else
1770     \expandafter\let\csname si@num@#1dec\endcsname\si@tempa
1771   \fi}
1772 \def\si@num@finderr#1#2\@empty{%

```

First a check for the opening character of an error.

```

1773 \si@str@ifchrstr{#1}{\si@numopenerr}

```

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

```

1774   {\ifsi@switch
1775     \si@log@err{Invalid error in number}
1776     {The numerical argument \si@num@arg\space has two (or
1777       more)\MessageBreak error-opening characters}%
1778   \else
1779     \expandafter\si@switchtrue
1780   \fi}

```

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

```

1781   {\si@str@ifchrstr{#1}{\si@numcloseerr}
1782   {\ifsi@switch
1783     \ifx\@empty#2\@empty\else
1784     \si@log@err{Invalid error in number}

```

```

1785             {The numerical argument \si@num@arg\space has an
1786             error-closing before the last character}%
1787         \fi
1788     \else
1789         \si@log@err{Invalid error in number}
1790         {The numerical argument \si@num@arg\space has an
1791         error-closing character\MessageBreak but no
1792         error-opening one}%
1793     \fi}

```

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

```

1794     {\ifsi@switch
1795     \expandafter\si@num@addtmpb
1796     \else
1797     \expandafter\si@num@addtmpa
1798     \fi
1799     {#1}}}%
1800 \ifx\@empty#2\@empty\else
1801     \si@num@finderr#2\@empty
1802 \fi}

```

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

```

\si@num@addtmpb \si@num@addtmpa{<num>}
\si@num@addtmp \si@num@addtmpb{<num>}
\si@num@addtmp \si@num@addtmp{<a/b>}{<num>}
1803 \newcommand*{\si@num@addtmpa}[1]{\si@num@addtmp{a}{#1}}
1804 \newcommand*{\si@num@addtmpb}[1]{\si@num@addtmp{b}{#1}}
1805 \newcommand*{\si@num@addtmp}[2]{%
1806     \expandafter\protected@edef\csname si@temp#1\endcsname{%
1807         \csname si@temp#1\endcsname#2}}

```

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

```

\si@num@cntdgt \si@num@cntdigits{<num>}
\si@num@cntdgt<char><chars>\@empty
1808 \newcommand*{\si@num@cntdigits}[1]{%
1809     \si@tempcnta\z@ \relax
1810     \expandafter\si@num@cntdgt#1\@empty\@empty}
1811 \def\si@num@cntdgt#1#2\@empty{%
1812     \ifx\@empty#1\@empty\else
1813         \advance\si@tempcnta\@ne \relax
1814     \fi
1815     \ifx\@empty#2\@empty\else
1816         \expandafter\si@num@cntdgt#2\@empty
1817     \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.

```

\si@num@serr
1818 \newcommand*{\si@num@smallerr}{%
1819     \si@tempcntb\si@tempcnta \relax
1820     \si@num@serr
1821     \protected@edef\si@num@err{%

```

```

1822 \ifsi@num@padlead0\fi\expandafter\@car\si@numdecimal\@nil
1823 \si@tempb}}
1824 \newcommand*{\si@num@serr}{%
1825 \si@num@cntdigits{\si@tempb}%
1826 \ifnum\si@tempcnta=\si@tempcntb\relax\else
1827 \protected@edef\si@tempb{0\si@tempb}%
1828 \expandafter\si@num@serr
1829 \fi}

```

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

```

\si@num@lerr \si@num@movedigit<char><chars>\@empty
\si@num@movedigit1830 \newcommand*{\si@num@largeerr}{%
1831 \renewcommand*{\si@tempa}{}%
1832 \si@tempcntb\si@tempcnta\relax
1833 \si@num@lerr
1834 \protected@edef\si@num@err{%
1835 \si@tempa\ensuremath{\si@decimalsymbol}\si@tempb}}
1836 \newcommand*{\si@num@lerr}{%
1837 \si@num@cntdigits{\si@tempb}%
1838 \ifnum\si@tempcnta=\si@tempcntb\relax\else
1839 \expandafter\si@num@movedigit\si@tempb\@empty\@empty
1840 \si@num@lerr
1841 \fi}
1842 \def\si@num@movedigit#1#2\@empty{%
1843 \protected@edef\si@tempa{\si@tempa#1}%
1844 \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.

```

1845 \newcommand*{\si@num@fixdp}{%
1846 \si@num@cntdigits{\si@num@postdec}%
1847 \ifx\@empty\si@num@postdec\@empty
1848 \si@tempcnta\z@\relax
1849 \fi
1850 \ifnum\si@tempcnta>\si@num@dp\relax
1851 \expandafter\si@num@round
1852 \else
1853 \ifnum\si@tempcnta<\si@num@dp\relax
1854 \expandafter\expandafter\expandafter\si@num@pad
1855 \fi
1856 \fi}

```

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

```

\si@num@pd
1857 \newcommand*{\si@num@pad}{%
1858 \si@log@debug{Padding to \the\si@num@dp\space digits}%
1859 \loop\ifnum\si@tempcnta<\si@num@dp\si@num@pd\repeat}
1860 \newcommand*{\si@num@pd}{%
1861 \advance\si@tempcnta\@ne\relax
1862 \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. The initial stage is to reverse the entire number, to make life easier. This

\si@num@prernd

\si@num@postrnd

\si@num@reverse

\si@num@rev

is then undone by the other macros as the output is constructed.

```

\si@num@reverse{<macro>}
\si@num@rev<char><chars>\@empty
1863 \newcommand*{\si@num@prernd}{}
1864 \newcommand*{\si@num@postrnd}{}
1865 \newcommand*{\si@num@round}{%
1866   \si@log@debug{Rounding to \the\si@num@dp\space digits}%
1867   \si@num@reverse{\si@num@postdec}%
1868   \si@num@reverse{\si@num@predec}%
1869   \let\si@num@prernd\si@num@predec
1870   \let\si@num@postrnd\si@num@postdec
1871   \renewcommand*{\si@num@predec}{}%
1872   \renewcommand*{\si@num@postdec}{}%
1873   \si@switchfalse
1874   \si@num@rnd}
1875 \newcommand*{\si@num@reverse}[1]{%
1876   \renewcommand*{\si@tempa}{}%
1877   \expandafter\si@num@rev#1\@empty\@empty
1878   \let#1\si@tempa}
1879 \def\si@num@rev#1#2\@empty{%
1880   \edef\si@tempa{#1\si@tempa}%
1881   \ifx\@empty#2\@empty\else
1882     \si@num@rev#2\@empty\@empty
1883   \fi}

```

`\si@num@rnd` The core looping macro of the system simply divides the flow between rounding
`\si@num@rndpre` before and after the decimal. The two routines are quite similar, but have
`\si@num@rndpost` 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.

```

1884 \newcommand*{\si@num@rnd}{%
1885   \ifnum\si@tempcnta>z@\relax
1886     \expandafter\si@num@rndpost
1887   \else
1888     \expandafter\si@num@rndpre
1889   \fi}
1890 \newcommand*{\si@num@rndpre}{%
1891   \expandafter\edef\expandafter\si@tempa\expandafter{%
1892     \expandafter\@car\si@num@prernd\@nil}%
1893   \expandafter\edef\expandafter\si@num@prernd\expandafter{%
1894     \expandafter\@cdr\si@num@prernd\@nil}%
1895   \si@tempcntb\si@tempa\relax
1896   \ifsi@switch
1897     \advance\si@tempcntb\@ne\relax
1898   \fi
1899   \si@switchfalse
1900   \ifnum\si@tempcntb=10\relax
1901     \si@tempcntb\z@\relax
1902     \expandafter\expandafter\expandafter\si@switchtrue
1903   \fi
1904   \edef\si@num@predec{\the\si@tempcntb\si@num@predec}%
1905   \ifx\@empty\si@num@prernd\@empty
1906     \ifsi@switch
1907       \edef\si@num@predec{1\si@num@predec}%

```

```

1908 \fi
1909 \else
1910 \expandafter\si@num@rnd
1911 \fi}
1912 \newcommand*{\si@num@rndpost}{%
1913 \expandafter\edef\expandafter\si@tempa\expandafter{%
1914 \expandafter\@car\si@num@postrnd\@nil}%
1915 \expandafter\edef\expandafter\si@num@postrnd\expandafter{%
1916 \expandafter\@cdr\si@num@postrnd\@nil}%
1917 \si@tempcntb\si@tempa\relax
1918 \ifsi@switch
1919 \advance\si@tempcntb\@ne\relax
1920 \fi
1921 \si@switchfalse
1922 \advance\si@num@dp\@ne\relax
1923 \ifnum\si@tempcnta>\si@num@dp\relax
1924 \advance\si@num@dp\m@ne\relax
1925 \else
1926 \advance\si@num@dp\m@ne\relax
1927 \ifnum\si@tempcnta>\si@num@dp\relax
1928 \ifnum\si@tempcntb>4\relax
1929 \expandafter\expandafter\expandafter\si@switchtrue
1930 \fi
1931 \else
1932 \ifnum\si@tempcntb=10\relax
1933 \si@tempcntb\z@\relax
1934 \expandafter\expandafter\expandafter\si@switchtrue
1935 \fi
1936 \edef\si@num@postdec{\the\si@tempcntb\si@num@postdec}%
1937 \fi
1938 \fi
1939 \advance\si@tempcnta\m@ne\relax
1940 \si@num@rnd}

```

`\si@num@int` 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.

`\si@num@int` {*integer-part*}

```

1941 \newcommand*{\si@num@int}[1]{%
1942 \renewcommand*{\si@num@predec}{}%
1943 \ifsi@sepfour
1944 \si@num@intfmt{ }#1\@empty\@empty\@empty
1945 \else
1946 \si@num@iffive{#1}
1947 {\si@num@intfmt{ }#1\@empty\@empty\@empty}
1948 {\renewcommand*{\si@num@predec}{#1}}%
1949 \fi}

```

`\si@num@iffive` A test is needed for the presence of more than four characters.

```

\si@num@five \si@num@iffive{<num>}
\si@num@five<char><char><char><char><chars>\end
1950 \newcommand*{\si@num@iffive}[1]{%

```

```

1951 \si@num@five#1\@empty\@empty\@empty\@empty\@empty\end}
1952 \def\si@num@five#1#2#3#4#5\end{%
1953 \ifx\@empty#5\@empty
1954 \expandafter\@secondoftwo
1955 \else
1956 \expandafter\@firstoftwo
1957 \fi}

\si@num@intfmt The business end of the integer formatter.
\si@num@fiint \si@num@intfmt{\langle char\rangle}{\langle char\rangle}{\langle char\rangle}{\langle char\rangle}
\si@num@fiint\langle chars\rangle\fi\fi\fi
1958 \newcommand*\si@num@intfmt[4]{%
1959 \ifx\@empty#2\@empty
1960 \si@num@intsep#1\relax
1961 \else
1962 \ifx\@empty#3\@empty
1963 \si@num@intsep\@empty\@empty#1#2\relax
1964 \else
1965 \ifx\@empty#4\@empty
1966 \si@num@intsep\@empty#1#2#3\relax
1967 \else
1968 \si@num@fiint{#1#2#3#4}%
1969 \fi
1970 \fi
1971 \fi}
1972 \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.
\si@num@intsep{\langle char\rangle}{\langle char\rangle}{\langle char\rangle}{\langle char\rangle}
1973 \newcommand*\si@num@intsep[4]{%
1974 \protected@edef\si@num@predec{\si@num@predec#1#2#3}%
1975 \if\relax#4\relax\else
1976 \protected@edef\si@num@predec{%
1977 \si@num@predec\ensuremath{\noexpand\si@digitsep}}%
1978 \expandafter\si@num@intsep\expandafter#4%
1979 \fi}

\si@num@dec Formatting a decimal uses a similar mechanism, but with a few alterations
\si@num@decfmt needed.
\si@num@dec{\langle decimal-part\rangle}
\si@num@decfmt{\langle char\rangle}{\langle char\rangle}{\langle char\rangle}{\langle char\rangle}
1980 \newcommand*\si@num@dec[1]{%
1981 \renewcommand*\si@num@postdec{}}%
1982 \ifsi@sepfour
1983 \si@num@decfmt#1\@empty\@empty\@empty\@empty
1984 \else
1985 \si@num@iffive{#1}
1986 {\si@num@decfmt#1\@empty\@empty\@empty\@empty}
1987 {\protected@edef\si@num@postdec{\si@num@postdec#1}}%
1988 \fi}
1989 \newcommand*\si@num@decfmt[4]{%
1990 \protected@edef\si@num@postdec{\si@num@postdec#1#2#3}%
1991 \ifx\@empty#4\@empty%

```

```

1992 \else
1993 \protected@edef\si@num@postdec{%
1994 \si@num@postdec\ensuremath{\noexpand\si@digitsep}}%
1995 \expandafter\si@num@decfmt\expandafter#4%
1996 \fi}

```

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

```

1997 \newcommand*{\si@num@procerr}{%
1998 \si@num@addunit
1999 \ensuremath{\si@pm}%
2000 \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.

```

2001 \newcommand*{\si@num@sepxpart}{%
2002 \si@num@addunit
2003 \ensuremath{\times}%
2004 \expandafter\expandafter\expandafter\si@num\expandafter
2005 \expandafter\expandafter{%
2006 \expandafter\@cdr\si@num@xpart\@nil}}

```

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

```

2007 \newcommand*{\si@num@addunit}{%
2008 \si@unt@numtrue
2009 \ifx\@empty\si@unt@unitarg\@empty\else
2010 \ifsi@repeatunits
2011 \si@unt@printunit{\si@unt@unitarg}%
2012 \fi
2013 \fi}

```

21.11 Formatting angles

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

```

\ang[\options]{\decimal-angle}
\ang[\options]{\deg;\min;\sec}
2014 \si@newrobustcmd*{\ang}[2][\]{%
2015 \begingroup
2016 \sisetup{#1}%
2017 \si@fam@mode
2018 \si@log@debug{Processing \string\ang\space input `#2'}%
2019 \@makeother\;}%
2020 \makeatletter
2021 \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.

```

\si@ang@parse\<num>;\<num>;\<num>;\<chars>\@nil
2022 \def\si@ang@parse#1;#2;#3;#4\@nil{%
2023 \let\ifsi@ang@fixdp\ifsi@fixdp
2024 \si@fixdpfalse

```

```

2025 \si@ifmtarg{#4}
2026   {\si@log@debug{Angle argument contains no
2027     semi-colons:\MessageBreak decimal angle}%
2028     \si@ang@dec{#1}{}}{}
2029   {\si@log@debug{Angle argument contains
2030     semi-colons:\MessageBreak degree-minute-second angle}%
2031     \renewcommand*{\si@tempa}{#4}%
2032     \renewcommand*{\si@tempb}{; ;}%
2033     \ifx\si@tempa\si@tempb\else
2034       \ifsi@strictarc
2035         \renewcommand*{\si@tempb}{; ;}%
2036         \ifx\si@tempa\si@tempb
2037           \si@log@err{Insufficient semi-colons in argument
2038             of \string\ang}{The argument of
2039             \string\ang\space must contain either no
2040             semi-colons or exactly two}%
2041         \else
2042           \si@log@err{Excess semi-colons in argument of
2043             \string\ang}{The argument of \string\ang\space
2044             must contain either no semi-colons or exactly
2045             two}%
2046         \fi
2047       \fi
2048     \fi
2049     \si@ang@arc{#1}{#2}{#3}}

```

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

```

\si@ang@arc2050 \newcommand*{\si@ang@dec}{%
2051   \let\si@ang@fix\@gobble
2052   \ifsi@ang@toarc
2053     \expandafter\si@ang@dectoarc
2054   \else
2055     \sisetup{padangle=none}\expandafter\si@ang@typeset
2056   \fi}
2057 \newcommand*{\si@ang@arc}{%
2058   \let\si@ang@fix\si@ang@arcfix
2059   \ifsi@ang@todec
2060     \expandafter\si@ang@arctodec
2061   \else
2062     \expandafter\si@ang@typeset
2063   \fi}

```

\ifsi@ang@fixdp A check is needed so that rounding and zero filling are only applied to the
\si@ang@fix seconds of an arc angle. \si@ang@fix{<degree/minute/second>}
\si@ang@arcfix \si@ang@arcfix{<degree/minute/second>}

```

2064 \newif\ifsi@ang@fixdp
2065 \newcommand*{\si@ang@fix}[1]{}
2066 \newcommand*{\si@ang@arcfix}[1]{%
2067   \renewcommand*{\si@tempa}{second}%
2068   \renewcommand*{\si@tempb}{#1}%
2069   \ifx\si@tempa\si@tempb
2070     \ifsi@ang@fixdp
2071       \expandafter\expandafter\expandafter\si@fixdptrue
2072     \else

```



```

2073     \expandafter\expandafter\expandafter\si@fixdpfalse
2074     \fi
2075   \else
2076     \expandafter\si@fixdpfalse
2077   \fi}

```

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

```

\si@ang@ifnum{<num>}
2078 \newcommand*{\si@ang@ifnum}[1]{%
2079   \begin{group}
2080     \renewcommand*{\si@numvalid}{0123456789,.-+}%
2081     \ifx\@empty#1\@empty
2082       \aftergroup\@firstoftwo
2083     \else
2084       \si@num@ifvalid{#1}
2085       {\aftergroup\@firstoftwo}
2086       {\aftergroup\@secondoftwo}%
2087     \fi
2088   \end{group}}

```

`\si@ang@arctodec` 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.

```

\si@ang@arctodec{<dec>}{<min>}{<sec>}
2089 \newcommand*{\si@ang@arctodec}[3]{%
2090   \let\si@ang@fix\@gobble
2091   \ifnum\si@num@dp>\thr@@\relax
2092     \si@num@dp\thr@@\relax
2093   \fi
2094   \si@fixdptrue
2095   \si@ang@ifnum{#1}
2096   {\si@ang@ifnum{#2}
2097     {\si@ang@ifnum{#3}
2098       {\si@tempdima\z@\relax
2099         \renewcommand*{\si@tempa}{+}%
2100         \ifx\@empty#1\@empty\else
2101           \si@tempdima #1pt\relax
2102         \fi
2103         \ifdim\si@tempdima<\z@\relax
2104           \renewcommand*{\si@tempa}{-}%
2105         \fi
2106         \ifx\@empty#2\@empty\else
2107           \si@tempdima\dimexpr\si@tempdima\si@tempa
2108             #2pt/60\relax
2109         \fi
2110         \ifdim\si@tempdima<\z@\relax
2111           \renewcommand*{\si@tempa}{-}%
2112         \else
2113         \fi
2114       \ifx\@empty#3\@empty\else

```

```

2115         \si@tempdima\dimexpr\si@tempdima\si@tempa
2116         #3pt/3600\relax
2117     \fi
2118     \sisetup{numdecimal=.%}
2119     \expandafter\si@ang@typeset\expandafter{%
2120         \strip@pt\si@tempdima}{\{ \} \} \}
2121     {\si@ang@notnum{#1}{#2}{#3}}
2122     {\si@ang@notnum{#1}{#2}{#3}}
2123     {\si@ang@notnum{#1}{#2}{#3}}

```

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

\si@ang@arcdeg

\si@ang@arcmin

\si@ang@arcsec \si@ang@dectoarc{*angle*}

```

2124 \newcommand*{\si@ang@dectoarc}[1]{%
2125     \let\si@ang@fix\si@ang@arcfix
2126     \si@ang@fixdptrue
2127     \ifnum\si@num@dp>\@ne\relax
2128         \si@num@dp\@ne\relax
2129     \fi
2130     \si@ang@ifnum{#1}
2131     {\si@tempdima\z@\relax
2132     \ifx\@empty#1\@empty\else
2133         \si@tempdima #1pt\relax
2134     \fi
2135     \si@ang@sepint{deg}%
2136     \si@tempdima\dimexpr\si@tempdima *60\relax
2137     \si@ang@sepint{min}%
2138     \edef\si@tempa{\the\dimexpr\si@tempdima *60\relax}%
2139     \expandafter\newcommand\expandafter*\expandafter{%
2140         \expandafter\si@ang@arcsec\expandafter}\expandafter{%
2141         \expandafter\si@ang@strippt\si@tempa}%

```

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

```

2142     \si@tempdima\z@\relax
2143     \edef\si@tempa{\the\si@tempdima}%
2144     \expandafter\renewcommand\expandafter*\expandafter{%
2145         \expandafter\si@tempa\expandafter}\expandafter{%
2146         \expandafter\si@ang@strippt\si@tempa}%
2147     \ifx\si@tempa\si@ang@arcsec
2148         \renewcommand*{\si@ang@arcsec}{0}%
2149     \fi

```

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

```

2150     \renewcommand*{\si@tempa}{0}%
2151     \ifx\si@ang@arcdeg\si@tempa
2152         \si@temptoks{\} \} \}%
2153     \else
2154         \si@temptoks{\si@ang@arcdeg} \} \}%
2155     \fi
2156     \ifx\si@ang@arcmin\si@tempa
2157         \si@temptoks\expandafter{\the\si@temptoks} \} \}%
2158     \else

```

```

2159      \si@temptoks\expandafter{\the\si@temptoks{%
2160      \si@ang@arcmin}}}%
2161      \fi
2162      \ifx\si@ang@arcsec\si@tempa
2163      \si@temptoks\expandafter{\the\si@temptoks{}}}%
2164      \else
2165      \si@temptoks\expandafter{\the\si@temptoks{%
2166      \si@ang@arcsec}}}%
2167      \fi
2168      \expandafter\si@ang@typeset\the\si@temptoks}
2169      {\si@ang@notnum{#1}}{}}{}}

\si@ang@sepint  Support macros for the conversion from decimal to arc angles.
\si@ang@sint    \si@ang@sepint{\deg/min}
\si@ang@stript \si@ang@sint<chars>.<chars>\@empty
               \si@ang@stript<chars>pt
2170 \newcommand*{\si@ang@sepint}[1]{%
2171   \expandafter\si@ang@sint\the\si@tempdima\@empty
2172   \expandafter\let\csname si@ang@arc#1\endcsname\si@tempa}
2173 \def\si@ang@sint#1.#2\@empty{%
2174   \renewcommand*{\si@tempa}{#1}%
2175   \si@tempdima 0.#2\relax}
2176 \begingroup
2177   \catcode`P=12
2178   \catcode`T=12
2179   \lowercase{
2180     \renewcommand*{\si@tempa}{%
2181       \def\si@ang@stript##1PT{##1}}
2182   \expandafter\endgroup
2183   \si@tempa

\si@ang@notnum  Not a TeX number: complain.
               \si@ang@notnum{\dec}{\min}{\sec}
2184 \newcommand*{\si@ang@notnum}[3]{%
2185   \si@log@warn{Angle `#1;#2;#3' is not a pure
2186     number:\MessageBreak output will be as given}%
2187   \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.
2188 \newif\ifsi@ang@sign

\si@ang@typeset  The \si@ang@set macro does the work of assigning the degrees, minutes and
               seconds, and actually typesetting the result.
               \si@ang@typeset{\num}{\num}{\num}
2189 \newcommand*{\si@ang@typeset}[3]{%

\si@ang@deg     First, the three macros that will contain the measures must exist.
\si@ang@mins2190 \ifsi@ang@padlarge
\si@ang@secs2191   \newcommand*{\si@ang@deg}{0\si@sym@degree}%
2192   \newcommand*{\si@ang@mins}{0\si@sym@minute}%
2193   \newcommand*{\si@ang@secs}{0\si@sym@second}%
2194   \else

```

```

2195 \newcommand*{\si@ang@deg}{\si@ang@deg}%
2196 \newcommand*{\si@ang@min}{\si@ang@min}%
2197 \newcommand*{\si@ang@sec}{\si@ang@sec}%
2198 \fi
2199 \protected@edef\si@ang@decimalsymbol{\si@decimalsymbol}%

```

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

```

2200 \ifsi@astroang
2201 \let\si@ang@movesign\si@ang@astroang
2202 \else
2203 \let\si@ang@movesign\@gobble
2204 \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.

```

2205 \newcommand*{\si@ang@secnum}{\si@ang@num{second}}%
2206 \newcommand*{\si@ang@minnum}{\si@ang@num{minute}}%
2207 \si@ifnotmtarg{#3}
2208 {\si@log@debug{Found seconds `#3' }%
2209 \si@ang@ifnum{#3}
2210 {\ifdim #3 pt=\z@ \relax \else
2211 \si@ang@signtrue
2212 \fi}}}%
2213 \renewcommand*{\si@ang@sec}{
2214 {\si@ang@secnum{#3}\si@sym@second}}%
2215 \renewcommand*{\si@ang@min}{
2216 {\si@ang@pad{0}\si@sym@minute}}%
2217 \renewcommand*{\si@ang@deg}{
2218 {\si@ang@pad{0}\si@sym@degree}}}%
2219 \si@ifnotmtarg{#2}
2220 {\si@log@debug{Found minutes `#2' }%
2221 \si@ang@ifnum{#2}
2222 {\ifdim #2 pt=\z@ \relax \else
2223 \si@ang@signtrue
2224 \fi}}}%
2225 \renewcommand*{\si@ang@secnum}{%
2226 \si@ang@signlessnum{second}}%
2227 \renewcommand*{\si@ang@min}{
2228 {\si@ang@minnum{#2}\si@sym@minute}}%
2229 \renewcommand*{\si@ang@deg}{
2230 {\si@ang@pad{0}\si@sym@degree}}}%
2231 \si@ifnotmtarg{#1}
2232 {\si@log@debug{Found degrees `#1' }%
2233 \renewcommand*{\si@ang@secnum}{%
2234 \si@ang@signlessnum{second}}%
2235 \renewcommand*{\si@ang@minnum}{%
2236 \si@ang@signlessnum{minute}}%
2237 \renewcommand*{\si@ang@deg}{

```

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

```

2238 {\si@ang@num{degree}{#1}}%
2239 \si@sym@degree}}%

```

```

2240 \si@out@num
2241   {\si@ang@deg\si@anglesep\si@ang@min\si@anglesep
2242     \si@ang@sec}%

```

The group opened by `\ang` is closed.

```

2243 \endgroup}

```

`\si@ang@pad` Padding is only added if requested; the zero is a literal.

```

\si@ang@pad{\num}

```

```

2244 \newcommand*{\si@ang@pad}[1]{\ifsi@ang@pads�all #1\fi}

```

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

`\si@ang@signlessnum`

```

\si@ang@num{\degree/minute/second}{\num}
\si@ang@signlessnum{\degree/minute/second}{\num}

```

```

2245 \newcommand*{\si@ang@num}[2]{%
2246   \begingroup
2247     \si@ang@fix{#1}%
2248     \si@ang@movesign{#1}%
2249     \si@num{#2}%
2250   \endgroup}
2251 \newcommand*{\si@ang@signlessnum}[2]{%
2252   \begingroup
2253     \si@ang@fix{#1}%
2254     \si@ang@movesign{#1}%
2255     \sisetup{addsign=none}%
2256     \si@num{#2}%
2257   \endgroup}

```

`\si@ang@killdegree` A mechanism is needed to handle moving the angle unit signs for the `astroang` option. This requires 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.

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

`\si@ang@killsecond`

`\si@ang@astrosign`

`\si@ang@decimalsymbol`

```

\si@ang@astrosign{\degree/minute/second}

```

```

2258 \newcommand*{\si@ang@killdegree}{\let\si@sym@degree\relax}
2259 \newcommand*{\si@ang@killminute}{\let\si@sym@minute\relax}
2260 \newcommand*{\si@ang@killsecond}{\let\si@sym@second\relax}
2261 \newcommand*{\si@ang@astrosign}[1]{%
2262   \renewcommand*{\si@decimalsymbol}{%
2263     \setbox\si@tempboxa=\hbox{%
2264       \ensuremath{\si@ang@decimalsymbol}}}%
2265     \si@tempdima\wd\si@tempboxa\relax
2266     \setbox\si@tempboxb=\hbox to\z@{%
2267       \hss\unhbox\si@tempboxa\hss}%
2268     \setbox\si@tempboxa=\hbox{%
2269       \csname si@sym@#1\endcsname\hskip-\scriptspace}%
2270     \si@tempdimb\wd\si@tempboxa\relax
2271     \setbox\si@tempboxc=\hbox to\z@{%
2272       \hss\unhbox\si@tempboxa\hss}%
2273     \setbox\si@tempboxd=\hbox{%
2274       \usebox\si@tempboxb\usebox\si@tempboxc}%

```

```

2275 \ifdim\si@tempdima>\si@tempdimb\relax
2276 \setbox\si@tempboxa=\hbox to\si@tempdima{%
2277 \hss\unhbox\si@tempboxd\hss}%
2278 \else
2279 \setbox\si@tempboxa=\hbox to\si@tempdimb{%
2280 \hss\unhbox\si@tempboxd\hss}%
2281 \fi
2282 \usebox\si@tempboxa%
2283 \ifdim\si@tempdima>\si@tempdimb\relax\else
2284 \hskip\scriptspace
2285 \fi}%
2286 \renewcommand*{\si@num@decimalhook}{\expandafter\aftergroup
2287 \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.

```

2288 \edef\si@tempa{%
2289 \noexpand\NC@do S\noexpand\NC@do s\the\NC@list}
2290 \newcolumnntype{S}{}
2291 \newcolumnntype{s}{}
2292 \NC@list\expandafter{\si@tempa}

```

\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.

```

\NC@rewrite@s \NC@rewrite@s[\options]
2293 \renewcommand*{\NC@rewrite@S}[1][{}]{%
2294 \edef\si@tempa{\the\@temptokena
2295 >\noexpand\si@tab@begin@S[#1]}c%
2296 <\noexpand\si@tab@end@S}%
2297 \@temptokena\expandafter{\si@tempa}%
2298 \NC@find}
2299 \renewcommand*{\NC@rewrite@s}[1][{}]{%
2300 \edef\si@tempa{\the\@temptokena
2301 >\noexpand\si@tab@begin@s[#1]}c%

```

```

2302     <{\noexpand\si@tab@end@s}}%
2303     \@temptokena\expandafter{\si@tempa}%
2304     \NC@find}

\si@tab@begin@s At this stage, the appropriate token gathering macro is activated, and the common
\si@tab@begin@s starting macro is called. For the S column, the seperr is turned off, and an error
\si@tab@gettok is set to be raised by any “x-part” input.
    \si@tab@begin@s[options]
    \si@tab@begin@s[options]
2305 \newcommand*{\si@tab@begin@s}[1][{}]{%
2306     \si@log@debug{Processing S column cell contents}%
2307     \let\si@tab@gettok\si@tab@gettok@s
2308     \si@seperrfalse
2309     \renewcommand*{\si@num@sepxpart}{%
2310         \si@log@err{Multiple numbers not allowed in
2311             tables\MessageBreak Only the first number used}
2312         \@ehb}%
2313     \si@tab@begin[#1]}
2314 \newcommand*{\si@tab@begin@s}[1][{}]{%
2315     \si@log@debug{Processing s column cell contents}%
2316     \let\si@tab@gettok\si@tab@gettok@s
2317     \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).
2318 \newtoks\si@tab@toks
2319 \newtoks\si@tab@pretoks
2320 \newtoks\si@tab@posttoks

\si@tab@begin 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.
    \si@tab@begin[options]
2321 \newcommand*{\si@tab@begin}[1][{}]{%
2322     \beginngroup
2323     \sisetup{#1}%
2324     \si@tab@toks{}%
2325     \si@tab@pretoks{}%
2326     \si@tab@posttoks{}%
2327     \si@switchfalse
2328     \si@tab@gettok}

\si@tab@gettok@s The two collection macros are very similar. Both compare the current input with
\si@tab@othertok a list of possible fixed values; this is pretty much a direct copy of numprint. If the
\si@tab@gettok@s input is not on the list of choices, it is processed as data for siunitx to handle. The
\si@tab@next S column does an additional check, to allow division of a number from any text.
For the s column, everything gets added to \si@tab@toks for later processing.
Two separate macros are needed here as the fixed values are dependant on the
column type, and awkward errors pop up if a combined approach is tried.
\si@tab@gettok@s{cell-contents}

```

```

\si@tab@othertok{<chars>}
\si@tab@gettok@s{<cell-contents>}
2329 \newcommand*{\si@tab@gettok@S}[1]{%
2330   \ifx\tabularnewline#1\relax
2331     \let\si@tab@next\si@tab@newline@S
2332   \else
2333     \ifx\end#1\relax
2334       \let\si@tab@next\end
2335     \else
2336       \ifx\si@tab@end@S#1\relax
2337         \let\si@tab@next\si@tab@end@S
2338       \else
2339         \ifx\endtabular#1\relax
2340           \let\si@tab@next\endtabular
2341         \else
2342           \ifx\csname#1\relax
2343             \let\si@tab@next\csname
2344           \else
2345             \ifx\relax#1\relax
2346               \let\si@tab@next\relax
2347             \else
2348               \let\si@tab@next\si@tab@gettok@S
2349               \si@str@ifchrstr{#1}{\si@numvalid}
2350               {\si@switchtrue
2351                 \si@log@debug{Found numerical cell
2352                   contents `#1'}%
2353                 \si@tab@toks=\expandafter{%
2354                   \the\si@tab@toks#1}}
2355               {\si@log@debug{Found other cell contents
2356                 \string#1}%
2357                 \si@tab@othertok{#1}}}%
2358             \fi
2359           \fi
2360         \fi
2361       \fi
2362     \fi
2363   \si@tab@next}
2364 \newcommand*{\si@tab@othertok}[1]{%
2365   \ifsi@switch
2366     \si@tab@posttoks=\expandafter{\the\si@tab@posttoks#1}%
2367   \else
2368     \si@tab@pretoks=\expandafter{\the\si@tab@pretoks#1}%
2369   \fi}
2370 \newcommand*{\si@tab@gettok@s}[1]{%
2371   \ifx\tabularnewline#1\relax
2372     \let\si@tab@next\si@tab@newline@s
2373   \else
2374     \ifx\end#1\relax
2375       \let\si@tab@next\end
2376     \else
2377       \ifx\si@tab@end@s#1\relax
2378         \let\si@tab@next\si@tab@end@s
2379       \else
2380

```



```

2381         \ifx\endtabular#1\relax
2382         \let\si@tab@next\endtabular
2383     \else
2384         \ifx\csize#1\relax
2385         \let\si@tab@next\csize
2386     \else
2387         \ifx\relax#1\relax
2388         \let\si@tab@next\relax
2389     \else
2390         \let\si@tab@next\si@tab@gettok@s
2391         \ifx\ignorespaces#1\relax\else
2392         \ifx\unskip#1\relax\else
2393             \si@tab@toks=\expandafter{%
2394                 \the\si@tab@toks#1}%
2395             \si@log@debug{Found cell contents '#1'}%
2396         \fi
2397     \fi
2398 \fi
2399 \fi
2400 \fi
2401 \fi
2402 \fi
2403 \fi
2404 \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`.

```

2405 \newcommand*{\si@tab@end@S}{%
2406     \ifsi@switch
2407         \let\si@tab@lfill\si@tab@lfill@S
2408         \let\si@tab@rfill\si@tab@rfill@S
2409     \else
2410         \let\si@tab@rfill\si@tab@rfill@t
2411         \let\si@tab@lfill\si@tab@lfill@t
2412     \fi
2413     \si@tab@lfill\relax
2414     \ifsi@switch
2415         \llap{\the\si@tab@pretoks}%
2416         \expandafter\si@tab@numout
2417         \rlap{\the\si@tab@posttoks}%
2418     \else
2419         \the\si@tab@pretoks
2420     \fi
2421     \si@tab@rfill\relax
2422 \endgroup}
2423 \newcommand*{\si@tab@end@s}{%
2424     \si@tab@lfill@s\relax
2425     \ignorespaces
2426     \expandafter\si\expandafter{\the\si@tab@toks}%
2427     \unskip
2428     \si@tab@rfill@s\relax

```

```

2429 \endgroup}

\si@tab@newline@S If the column is the final one read, then some work is needed with the
\si@tab@newline@s \tabularnewline macro. Output has to happen before the new line, then
\si@tab@end 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.
2430 \newcommand*{\si@tab@newline@S}{%
2431 \si@tab@end@S
2432 \hfil\relax
2433 \let\si@tab@end\si@tab@end@S
2434 \renewcommand*{\si@tab@end@S}{\let\si@tab@end@S\si@tab@end}%
2435 \tabularnewline}
2436 \newcommand*{\si@tab@newline@s}{%
2437 \si@tab@end@s
2438 \hfil\relax
2439 \let\si@tab@end\si@tab@end@s
2440 \renewcommand*{\si@tab@end@s}{\let\si@tab@end@s\si@tab@end}%
2441 \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.
2442 \newcount\si@tempcnta
2443 \newcount\si@tempcntb

\si@tab@numout If a number is found, then some secondary processing is needed to format it
correctly.
2444 \newcommand*{\si@tab@numout}{%
2445 \si@num@intabtrue
2446 \ifsi@tab@fixed
2447 \ifsi@tabautofit
2448 \si@num@dp\si@tab@mantpostcnt\relax
2449 \expandafter\expandafter\expandafter\si@fixdptrue
2450 \fi
2451 \fi
2452 \expandafter\si@num\expandafter{\the\si@tab@toks}%
2453 \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@expbox2454 \newbox\si@tab@prebox
2455 \newbox\si@tab@midbox
2456 \newbox\si@tab@postbox
2457 \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.
2458 \newcommand*{\si@tab@format}{%
2459 \ifsi@tab@fixed

```

```

2460 \expandafter\si@tab@fixed
2461 \else
2462 \expandafter\si@tab@unfixed
2463 \fi

A hack to get the correct colour everywhere without too much work.

2464 \ifsi@colourvalues
2465 \si@fam@colourcmd{\si@valuecolour}%
2466 \fi
2467 \box\si@tab@prebox\box\si@tab@midbox\box\si@tab@postbox%
2468 \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.

```

2469 \newcommand*\si@tab@unfixed{%
2470 \si@log@debug{Using variable width S column}%
2471 \protected@edef\si@num@out{\si@num@out\si@tab@expout}%
2472 \setbox\si@tab@prebox=\hbox
2473 {\expandafter\si@out@num\expandafter{\si@tab@out}}%
2474 \ifx\@empty\si@num@out\@empty
2475 \setbox\si@tab@midbox=\hbox
2476 {\phantom{\ensuremath{\si@decimalsymbol}}}%
2477 \else
2478 \setbox\si@tab@midbox=\hbox
2479 {\ensuremath{\si@decimalsymbol}}}%
2480 \fi
2481 \setbox\si@tab@postbox=\hbox
2482 {\expandafter\si@out@num\expandafter{\si@num@out}}%
2483 \ifdim\wd\si@tab@prebox>\wd\si@tab@postbox\relax
2484 \setbox\si@tab@postbox=\hbox to\wd\si@tab@prebox%
2485 {\unhbox\si@tab@postbox\hfill}%
2486 \else
2487 \setbox\si@tab@prebox=\hbox to\wd\si@tab@postbox%
2488 {\hfill\unhbox\si@tab@prebox}%
2489 \fi
2490 \setbox\si@tab@expbox=\hbox{}}

```

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

```

\si@tab@postdim2491 \newdimen\si@tab@predim
\si@tab@expdim2492 \newdimen\si@tab@postdim
\si@tempdima2493 \newdimen\si@tab@expdim
\si@tempdimb2494 \newdimen\si@tempdima
2495 \newdimen\si@tempdimb

```

`\si@tab@sp` A short macro to control superscript.

```

2496 \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.

```

2497 \newcommand*\si@tab@fixed{%
2498 \si@log@debug{Using fixed-width S column}%

```

```

2499 \let\si@tab@sp\relax
2500 \setbox\si@tab@midbox=\hbox{ }%
2501 \setbox\si@tab@expbox=\hbox{ }%
2502 \setbox\si@tempboxa=\hbox{\si@out@num{1}}%
2503 \si@tempdima\wd\si@tempboxa\relax

```

The width for the two output boxes is set up.

```

2504 \si@tab@predim\the\si@tab@mantprecnt\si@tempdima\relax
2505 \si@tab@sepcorr{mantpre}{pre}%
2506 \si@tab@postdim\si@tab@mantpostcnt\si@tempdima\relax
2507 \setbox\si@tempboxa=\hbox{\ensuremath{\{\si@decimalsymbol\}}}%
2508 \advance\si@tab@postdim\wd\si@tempboxa\relax
2509 \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.

```

2510 \setbox\si@tempboxa=\hbox{\si@out@num^{1}}%
2511 \si@tempdima\wd\si@tempboxa\relax
2512 \ifnum\si@tab@expprecnt>z@\relax
2513   \si@tab@expdim\si@tab@expprecnt\si@tempdima\relax
2514   \si@tab@sepcorr{exppe}{exp}%
2515 \fi
2516 \let\si@tab@sp\sp
2517 \ifnum\si@tab@exppostcnt>z@\relax
2518   \advance\si@tab@expdim\si@tab@exppostcnt\si@tempdima\relax
2519   \setbox\si@tempboxa=\hbox{%
2520     \ensuremath{\^{ \si@decimalsymbol}}}%
2521   \advance\si@tab@expdim\wd\si@tempboxa\relax
2522   \si@tab@sepcorr{exp post}{exp}%
2523 \fi

```

Space is reserved for signs.

```

2524 \setbox\si@tempboxa=\hbox{\ensuremath{-}}%
2525 \ifsi@tab@mantsign
2526   \advance\si@tab@predim\wd\si@tempboxa\relax
2527 \fi
2528 \setbox\si@tempboxa=\hbox{\ensuremath{\{-}}}%
2529 \ifsi@tab@expsign
2530   \advance\si@tab@expdim\wd\si@tempboxa\relax
2531 \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.

```

2532 \si@tempcnta\si@tab@expprecnt\relax
2533 \advance\si@tempcnta\si@tab@exppostcnt\relax
2534 \ifnum\si@tempcnta>z@\relax
2535   \setbox\si@tempboxa=\hbox{\ensuremath{%
2536     }\si@expproduct}\si@expbase}}%
2537   \advance\si@tab@expdim\wd\si@tempboxa\relax
2538 \fi

```

Finally for the box dimensions, if the exponent is not aligned, the space reserved for it is added to the post box.

```

2539 \ifsi@tab@alignexp\else

```

```

2540 \advance\si@tab@postdim\si@tab@expdim\relax
2541 \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.

```

2542 \setbox\si@tab@prebox=\hbox to\si@tab@predim{\hss\hfill
2543 \expandafter\si@out@num\expandafter{\si@tab@out}}%
2544 \ifx\@empty\si@num@out\@empty
2545 \setbox\si@tab@postbox=\hbox to\si@tab@postdim
2546 {\expandafter\si@out@num\expandafter{\si@num@out}\hfil}%
2547 \else
2548 \ifsi@tabalignexp\else
2549 \protected@edef\si@num@out{\si@num@out\si@tab@expout}%
2550 \fi
2551 \setbox\si@tab@postbox=\hbox to\si@tab@postdim
2552 {\ensuremath{\{\si@decimalsymbol\}}\expandafter\si@out@num
2553 \expandafter{\si@num@out}\hfil}%
2554 \fi
2555 \ifx\@empty\si@tab@expout\@empty
2556 \ifsi@tabalignexp
2557 \setbox\si@tab@expbox=\hbox to\si@tab@expdim{\hfil}%
2558 \fi
2559 \else
2560 \ifsi@tabalignexp
2561 \setbox\si@tab@expbox=\hbox to\si@tab@expdim
2562 {\expandafter\si@out@num\expandafter{\si@tab@expout}%
2563 \hfil}%
2564 \fi
2565 \fi}

```

`\si@tab@sepcorr` 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.

```

\si@tab@sepcorr{\count}{\dimen}
2566 \newcommand*\si@tab@sepcorr[2]{%
2567 \expandafter\si@tempcnta\expandafter\the
2568 \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.

```

2569 \divide\si@tempcnta\thr@@\relax
2570 \ifsi@sepfour\else
2571 \expandafter\ifnum\expandafter\the
2572 \csname si@tab@#1cnt\endcsname=4\relax
2573 \si@tempcnta\z@\relax
2574 \fi
2575 \fi

```

The width of the separators is measured, and the correct number of separator widths are added to the box dimension.

```

2576 \setbox\si@tempboxa=\hbox{%
2577 \ensuremath{\si@tab@sp{\si@digitsep}}}%
2578 \expandafter\advance\csname si@tab@#2dim\endcsname
2579 \si@tempcnta\wd\si@tempboxa}

```

21.13 Units

`\SI` 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. `\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 argument to `\SI`

`\SI[⟨options⟩]{⟨num⟩} \si[⟨options⟩]{⟨unit⟩}`

```
2580 \si@newrobustcmd*{\SI}[2][]{%
2581   \@ifnextchar[%
2582     {\si@SI[#1]{#2}}
2583     {\si@SI[#1]{#2}[]}}
2584 \si@newrobustcmd*{\si}[2][]{\si@SI[#1]{}[]{#2}}
```

`\newunit` 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.

`\newunit[⟨options⟩]{⟨unit⟩}{⟨symbol⟩}`
`\renewunit[⟨options⟩]{⟨unit⟩}{⟨symbol⟩}`
`\provideunit[⟨options⟩]{⟨unit⟩}{⟨symbol⟩}`

```
2585 \newcommand*{\newunit}[3][]{%
2586   \si@ifdefinable{#2}
2587     {\si@unt@defunit[#1]{#2}{#3}}
2588     {\si@log@err{Unit \string#2 already defined!}\@eha}}
2589 \newcommand*{\renewunit}[3][]{%
2590   \si@ifdefinable{#2}
2591     {\si@log@err{Unit \string#2 undefined}\@ehc
2592     \si@unt@defunit[#1]{#2}{#3}}
2593     {\si@log@inf{Redefining unit \string#2}%
2594     \si@unt@defunit[#1]{#2}{#3}}}
2595 \newcommand*{\provideunit}[3][]{%
2596   \si@ifdefinable{#2}
2597     {\si@unt@defunit[#1]{#2}{#3}}
2598   {}}
```

`\newprefix` 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. `\newprefix[⟨binary⟩]{⟨multiple⟩}{⟨powers-ten⟩}{⟨symbol⟩}`

`\renewprefix[⟨binary⟩]{⟨multiple⟩}{⟨powers-ten⟩}{⟨symbol⟩}`
`\provideprefix[⟨binary⟩]{⟨multiple⟩}{⟨powers-ten⟩}{⟨symbol⟩}`

```
2599 \newcommand*{\newprefix}[4][]{%
2600   \si@ifdefinable{#2}
2601     {\si@unt@defprefix[#1]{#2}{#3}{#4}}
2602     {\si@log@err{Prefix \string#2 already defined!}\@eha}}
2603 \newcommand*{\renewprefix}[4][]{%
2604   \si@ifdefinable{#2}
2605     {\si@log@err{Prefix \string#2 undefined}\@ehc
2606     \si@unt@defprefix[#1]{#2}{#3}{#4}}}
```

```

2607     {\si@log@inf{Redefining prefix \string#2}%
2608       \si@unt@defprefix[#1]{#2}{#3}{#4}}
2609 \newcommand*{\provideprefix}[4][[]]{%
2610   \si@ifdefinable{#2}
2611     {\si@unt@defprefix[#1]{#2}{#3}{#4}}
2612     {}}

```

`\newpower` Here power multiples for units are set up. As with units and multiples, a layered approach is used to keep things easy to maintain. The optional argument here is *not* a **keyval** one: only `post` is a valid value.

```

\renewpower[\langle post \rangle]{\langle num \rangle}{\langle power \rangle}
\renewpower[\langle post \rangle]{\langle num \rangle}{\langle power \rangle}
\providepower[\langle post \rangle]{\langle num \rangle}{\langle power \rangle}

```

```

2613 \newcommand*{\newpower}[3][[]]{%
2614   \si@ifdefinable{#2}
2615     {\si@unt@defpower[#1]{#2}{#3}}
2616     {\si@log@err{Power \string#2 already defined!}\@eha}}
2617 \newcommand*{\renewpower}[3][[]]{%
2618   \si@ifdefinable{#2}
2619     {\si@log@err{Power \string#2 undefined}\@ehc
2620     \si@unt@defpower[#1]{#2}{#3}}
2621     {\si@log@inf{Redefining power \string#2}%
2622     \si@unt@defpower[#1]{#2}{#3}}}
2623 \newcommand*{\providepower}[3][[]]{%
2624   \si@ifdefinable{#2}
2625     {\si@unt@defpower[#1]{#2}{#3}}
2626     {}}

```

`\ifsi@unt@num` A flag is needed to tell the processor whether there is a number, to get the correct spacing.

```

2627 \newif\ifsi@unt@num

```

`\si@unt@unitarg` A storage macro for the argument of the unit macro.

```

2628 \newcommand*{\si@unt@unitarg}{}

```

`\si@SI` 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.

```

\si@SI[\langle options \rangle]{\langle unit \rangle}[\langle preunit \rangle]{\langle unit \rangle}

```

```

2629 \newcommand*{\si@unt@SIopts}{}
2630 \def\si@SI[#1]#2[#3]#4{%
2631   \begingroup
2632     \si@ifnotmtarg{#1}
2633     {\sisetup{#1}%
2634     \renewcommand*{\si@unt@SIopts}{#1}}%

```

The prefix unit is handled before any processing of the number; the flags are set to get spacing correct.

```

2635   \si@unt@numfalse
2636   \si@xspacefalse
2637   \si@ifnotmtarg{#3}
2638     {\si@log@debug{Prefix unit found}%
2639     \si@unt@printunit{#3}}%

```

The numerical argument may be empty, in which case no extra space should be produced.

```

2640 \si@ifnotmtarg{#4}
2641 {\renewcommand*\si@unt@unitarg}{#4}}%
2642 \si@ifnotmtarg{#2}
2643 {\si@log@debug{Number found in \string\SI\space
2644 argument}%
2645 \ifsi@repeatunits\else
2646 \ifsi@trapambigerr
2647 \expandafter\expandafter\expandafter
2648 \si@num@ambigertrue
2649 \fi
2650 \fi
2651 \num{#2}%
2652 \si@unt@numtrue}%

```

If there is a unit, a check is needed in case the units need to have a power added.

```

2653 \si@ifnotmtarg{#4}
2654 {\si@ifmtarg{#2}
2655 {\si@unt@printunit{#4}}
2656 {\si@tempcnta\z@ \relax
2657 \ifsi@addunitpower
2658 \si@unt@countx{#2}%
2659 \fi
2660 \ifnum\si@tempcnta>\z@ \relax
2661 \advance\si@tempcnta\@ne \relax
2662 \edef\si@tempa{\noexpand\tothe{\si@tempcnta}}%
2663 \renewcommand*\si@tempb{#4}%
2664 \expandafter\expandafter\expandafter
2665 \si@unt@printunit\expandafter\expandafter
2666 \expandafter{%
2667 \expandafter\si@tempb\si@tempa}%
2668 \else
2669 \si@unt@printunit{#4}%
2670 \fi}}%
2671 \endgroup}

```

`\si@unt@countx` A short macro to count up any multiplication in numerical input.

```

2672 \newcommand*\si@unt@countx[1]{%
2673 \si@tempcnta\z@ \relax
2674 \expandafter\si@unt@cntx#1\@empty\@empty}
2675 \def\si@unt@cntx#1#2\@empty{%
2676 \si@str@ifchrstr{#1}{\si@numprod}
2677 {\advance\si@tempcnta\@ne \relax}
2678 {}}%
2679 \ifx\@empty#2\@empty\else
2680 \si@unt@cntx#2\@empty\@empty
2681 \fi}

```

`\si@unt@ifliteral` 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.

```
\si@unt@ifliteral{text}
```

```
2682 \newif\ifsi@unt@littest
2683 \newcommand*{\si@unt@ifliteral}[1]{%
2684   \begingroup
2685     \si@unt@littesttrue
```

The test relies on any non-processable test having some width; hopefully, this should be the case.

```
2686     \setbox\si@tempboxa=\hbox{\si@unt@out{#1}}%
2687     \ifdim\wd\si@tempboxa>\z@ \relax
2688       \aftergroup\@firstoftwo
2689     \else
2690       \aftergroup\@secondoftwo
2691     \fi
2692   \endgroup}
```

`\ifsi@unt@litout` 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`).

```
\si@unt@printunit{unit}
```

```
2693 \newif\ifsi@unt@litout
2694 \newcommand*{\si@unt@printunit}[1]{%
2695   \si@unt@ifliteral{#1}
```

The unit includes one or more literal items; typeset using the unit typesetting macro.

```
2696   {\si@log@debug{Literal items found in unit
2697     argument:\MessageBreak outputting without further
2698     processing}%
2699     \si@unt@litouttrue
2700     \si@unt@addvaluesep
2701     \si@unt@out{#1}}
```

For processable output, the argument is executed; the macros are all designed for this.

```
2702   {\si@log@debug{Macro unit found:\MessageBreak
2703     processing to format output}%
2704     \si@unt@init
2705     \advance\si@unt@depthcnt\@ne \relax
2706     #1%
2707     \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@litvalsep2708 \newcommand*{\si@unt@addvaluesep}{%
\si@unt@stackvalsep2709 \ifsi@unt@num
2710   \expandafter\si@unt@addvalsep
2711   \fi}
2712 \newcommand*{\si@unt@addvalsep}{%
2713   \ifsi@unt@litout
2714     \expandafter\si@unt@litvalsep
2715   \else
```

```

2716 \expandafter\si@unt@stackvalsep
2717 \fi}
2718 \newcommand*{\si@unt@stackvalsep}{%
2719 \protected@edef\si@unt@spstack{\si@valuesep}}
2720 \newcommand*{\si@unt@litvalsep}{%
2721 \nobreak\ensuremath{\si@valuesep}\nobreak}

```

`\si@unt@spstack` The initialisation macro sets up the various switches, and clears the storage areas for the formatted output. There are two stacks, as when typesetting as fractions, the two parts of the number have to be stored separately. The depth counter is used to tell when recursion ends in the processor. The “first” switch is needed as the depth counter will not be at one for items processed by `\SI`.

```

\si@unt@depthcnt2722 \newcommand*{\si@unt@spstack}{}
\ifsi@unt@first2723 \newcommand*{\si@unt@stacka}{}
\ifsi@unt@first2724 \newcommand*{\si@unt@stackb}{}
\si@unt@init2725 \newcount\si@unt@unitcnta
2726 \newcount\si@unt@unitcntb
2727 \newcount\si@unt@depthcnt
2728 \newif\ifsi@unt@first
2729 \si@unt@depthcnt\m@ne\relax
2730 \newcommand*{\si@unt@init}{%
2731 \begingroup
2732 \si@unt@litoutfalse
2733 \si@unt@litprefixfalse
2734 \si@unt@firsttrue
2735 \si@unt@perfalse
2736 \si@unt@perseenfalse
2737 \si@unt@prepowerfalse
2738 \si@unt@depthcnt\z@\relax
2739 \si@unt@powerdim\z@\relax
2740 \si@unt@unitcnta\z@\relax
2741 \si@unt@unitcntb\z@\relax
2742 \si@unt@prefixcnt\z@\relax
2743 \renewcommand*{\si@unt@spstack}{}%
2744 \renewcommand*{\si@unt@stacka}{}%
2745 \renewcommand*{\si@unt@stackb}{}%
2746 \renewcommand*{\si@unt@holdstacka}{}%
2747 \renewcommand*{\si@unt@holdstackb}{}%
2748 \renewcommand*{\si@unt@lastadda}{space}%
2749 \renewcommand*{\si@unt@lastaddb}{space}}

```

`\si@unt@final` The finalisation macro finishes off the output and resets the flags.

```

2750 \newcommand*{\si@unt@final}{%
2751 \si@unt@third
2752 \si@unt@stackout
2753 \endgroup
2754 \ifsi@xspace
2755 \expandafter\expandafter\expandafter\xspace
2756 \fi}

```

`\si@unt@defunit` The internal macro for defining a unit does not check for redefinition; that is done by the user macros.

```
\si@unt@defunit[\langle options \rangle]{\langle unit \rangle}{\langle symbol \rangle}
```

```

2757 \newcommand*{\si@unt@defunit}[3][\%
2758 \si@log@debug{Declaring unit \string#2 with \MessageBreak
2759 meaning \string#3}%

```

The optional argument needs to be saved. The macro name is reversed so that life is easier with the expansions here.

```

2760 \si@ifnotmtarg{#1}
2761 {\expandafter\@namedef\expandafter{\%
2762 \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.

```

2763 \protected\def#2{\%
2764 \ifsi@allowoptarg
2765 \expandafter\si@unt@withopt
2766 \else
2767 \expandafter\si@unt@noopt
2768 \fi
2769 {#2}{#3}}

```

\si@unt@withopt 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.

```

\si@unt@withopt{\langle unit \rangle}{\langle symbol \rangle}
\si@unt@noopt{\langle unit \rangle}{\langle symbol \rangle}

2770 \newcommand*{\si@unt@withopt}[2]{\%
2771 \@ifnextchar[\%
2772 {\si@unt@opt{#1}{#2}}
2773 {\si@unt@opt{#1}{#2}[]}}
2774 \newcommand*{\si@unt@noopt}[2]{\si@unt@opt{#1}{#2}[]}

```

\si@unt@opt 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.

```

\si@unt@opt{\langle unit \rangle}{\langle symbol \rangle}[\langle num \rangle]

2775 \def\si@unt@opt#1#2[#3]{\%
2776 \ifsi@unt@littest
2777 \expandafter\si@gobblethree
2778 \else

```

For literal output, the second argument is all that is needed.

```

2779 \ifsi@unt@litout
2780 \expandafter\expandafter\expandafter\@gobbletwo
2781 \else
2782 \expandafter\expandafter\expandafter\si@unt@unit
2783 \fi
2784 \fi
2785 {#3}{#1}{#2}}

```

\si@gobblethree L^AT_EX does not have a \@gobblethree macro, but one is needed.

```

2786 \long\def\si@gobblethree #1#2#3{}

```

\si@unt@defprefix As with units, multiples are defined by an internal macro.
\ifsi@unt@litprefix \si@unt@defprefix[\langle binary \rangle]{\langle multiple \rangle}{\langle powers-ten \rangle}{\langle symbol \rangle}

```

2787 \newif\ifsi@unt@litprefix
2788 \si@unt@litprefixtrue
2789 \newcommand*{\si@unt@defprefix}[4][\%
2790 \si@log@debug{Declaring multiple \string#1 with\MessageBreak
2791 meaning \string#4}\%

```

The optional argument is saved, using `\def` as no check is made on an existing definition of the storage macro.

```

2792 \expandafter\expandafter\expandafter\def\expandafter
2793 \csname\expandafter\@gobble\string#2@opt@si\endcsname{#1}%
2794 \protected\def#2{%
2795 \ifsi@unt@littest
2796 \expandafter\si@gobblethree
2797 \else
2798 \ifsi@unt@litout
2799 \expandafter\expandafter\expandafter\@gobbletwo
2800 \else
2801 \ifsi@unt@litprefix
2802 \expandafter\expandafter\expandafter\expandafter
2803 \expandafter\expandafter\expandafter\@gobbletwo
2804 \else
2805 \expandafter\expandafter\expandafter\expandafter
2806 \expandafter\expandafter\expandafter\si@unt@prefix
2807 \fi
2808 \fi
2809 \fi
2810 {#2}{#3}{#4}}

```

`\si@unt@defpower` 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.

```

\si@unt@defpower[\post]{\power}{\num}
2811 \newcommand*{\si@unt@defpower}[3][\%
2812 \si@log@debug{Declaring power \string#2 with\MessageBreak
2813 meaning \string#3}\%

```

Once again the optional argument is saved.

```

2814 \expandafter\expandafter\expandafter\def\expandafter
2815 \csname\expandafter\@gobble\string#2@opt@si\endcsname{#1}%
2816 \protected\def#2{%
2817 \ifsi@unt@littest
2818 \expandafter\@gobbletwo
2819 \else

```

The literal output here does not need to gobble anything.

```

2820 \ifsi@unt@litout
2821 \expandafter\expandafter\expandafter\si@unt@litpower
2822 \else
2823 \expandafter\expandafter\expandafter\si@unt@power
2824 \fi
2825 \fi
2826 {#2}{#3}}

```

`\si@unt@unithook` The macro for units is actually a processor, rather than typesetting anything,
`\si@unt@unit` which is handled elsewhere. The first argument to the macro is optional, but

does not have square brackets to keep things simple with gobbling.

```
\si@unt@unit{<num>}{<unit>}{<symbol>}
```

```
2827 \newcommand*{\si@unt@unithook}{}
2828 \newcommand*{\si@unt@unit}[3]{%
```

When the count is minus one at the start of the processor, then the unit is begin used on its own: initialisation occurs.

```
2829 \ifnum\si@unt@depthcnt=\m@ne\relax
2830 \expandafter\si@unt@init
2831 \fi
2832 \advance\si@unt@depthcnt\@ne\relax
2833 \si@log@debug{Unit processing: level \the\si@unt@depthcnt,
2834 \MessageBreak unit \string#2}%
2835 \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.

```
2836 \si@unt@ifliteral{#3}
2837 {\si@unt@addtostack{unit}{#3}%
2838 \ifsi@unt@prepower
2839 \expandafter\si@unt@stkpowers
2840 \fi}
2841 {#3}%
```

The counter is now stepped down, before checking if this is the end of a compound unit.

```
2842 \advance\si@unt@depthcnt\m@ne\relax
2843 \ifnum\si@unt@depthcnt=\z@\relax
2844 \expandafter\si@unt@final
2845 \fi}
```

`\si@unt@firstorsecond` At this stage, the flag will be set for the first item to be processed whichever route the unit has been called by.

```
\si@unt@firstorsecond{<num>}{<macro>}
2846 \newcommand*{\si@unt@firstorsecond}[2]{%
2847 \ifsi@unt@first
2848 \expandafter\si@unt@first
2849 \else
2850 \expandafter\si@unt@second
2851 \fi
2852 {#1}{#2}}%
```

`\si@unt@first` For the first unit in the input, some extra tasks are needed. First, the optional argument for the unit macro needs to be tested.

```
\si@unt@first{<num>}{<unit>}
2853 \newcommand*{\si@unt@first}[2]{%
2854 \si@ifnotmtarg{#1}
2855 {\num{#1}%
2856 \si@unt@numtrue}%
2857 \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.

```

2858 \ifcsname\expandafter\@gobble\string#2@opt@unt@si\endcsname
2859 \expandafter\si@unt@setopts
2860 \else
2861 \expandafter\@gobble
2862 \fi
2863 {#2}%
2864 \si@unt@addvaluesep
2865 \si@unt@firstfalse}

```

\si@unt@setopts A rather long set of \expandafter commands to get the options to set safely.
\si@unt@setSIopts \si@unt@setopts{*unit*}

```

2866 \newcommand*{\si@unt@setopts}[1]{%
2867 \expandafter\expandafter\expandafter\expandafter\expandafter
2868 \expandafter\expandafter\si@temptoks\expandafter
2869 \expandafter\expandafter\expandafter\expandafter
2870 \expandafter\expandafter{\expandafter%
2871 \csname\expandafter\@gobble\string#1@opt@unt@si%
2872 \endcsname}%
2873 \expandafter\sisetup\expandafter{\the\si@temptoks}%
2874 \si@log@debug{Applying options '\the\si@temptoks'
2875 for\MessageBreak unit \string#1}%

```

The user options are reloaded, if defined, to ensure that they still work as expected.

```

2876 \@ifundefined{si@unt@SIopts}{}
2877 {\ifx\@empty\si@unt@SIopts\@empty\else
2878 \expandafter\expandafter\si@unt@setSIopts
2879 \fi}}
2880 \newcommand*{\si@unt@setSIopts}{%
2881 \expandafter\si@temptoks\expandafter{\si@unt@SIopts}%
2882 \expandafter\sisetup\expandafter{\the\si@temptoks}}

```

\si@unt@second For everything apart from the first item to be processed, spacing may need to
\si@unt@third be added to separated different units. The macro is divided into two, so that
everything except the space can be added in finalisation.

```

\si@unt@second{<num>}{<unit>}
2883 \newcommand*{\si@unt@second}[2]{%
2884 \si@ifnotmtarg{#1}
2885 {\si@log@warn{Optional argument to unit macro\MessageBreak
2886 allowed only for outer unit}}%
2887 \si@unt@third
2888 \si@unt@addtostack{space}{\ensuremath{\si@unitsep}}}
2889 \newcommand*{\si@unt@third}{%
2890 \ifsi@unt@prepower\else
2891 \expandafter\si@unt@stkpower
2892 \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`.

```

2893 \renewcommand*{\si@tempa}{prefix}%
2894 \expandafter\ifx
2895 \csname si@unt@lastadd\si@unt@checkstack\endcsname\si@tempa
2896 \else
2897 \expandafter\si@unt@spacecheck

```

```

2898 \fi
2899 \ifsi@unt@per
2900 \expandafter\si@unt@perseenttrue
2901 \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.

```

2902 \newcommand*{\si@unt@spacecheck}{%
2903 \renewcommand*{\si@tempa}{space}%
2904 \expandafter\ifx
2905 \csname si@unt@lastadd\si@unt@checkstack\endcsname\si@tempa
2906 \else
2907 \expandafter\si@unt@reciptest
2908 \fi}

```

`\si@unt@prefix` Actual output of the prefixes.

```

\si@unt@prefix{\langle multiple \rangle}{\langle powers-ten \rangle}{\langle symbol \rangle}
2909 \newcommand*{\si@unt@prefix}[3]{%
2910 \si@unt@firstorsecond{}{#1}%
2911 \ifsi@prefixsymbolic
2912 \expandafter\si@unt@addprefix
2913 \else
2914 \expandafter\si@unt@countprefix
2915 \fi
2916 {#1}{#2}{#3}}

```

`\si@unt@addprefix` To add the prefix, a little translation is needed.

```

\si@unt@countprefix{\langle gobble \rangle}{\langle gobble \rangle}{\langle symbol \rangle}
2917 \newcommand*{\si@unt@addprefix}[3]{%
2918 \si@unt@addtostack{prefix}{#3}}

```

`\si@unt@prefixcnt` On the other hand, to count the prefix numeral, the symbol is thrown away.

`\si@unt@countprefix` `\si@unt@countprefix{\langle multiple \rangle}{\langle powers-ten \rangle}{\langle gobble \rangle}`

```

\si@unt@invprefix2919 \newcount\si@unt@prefixcnt
2920 \newcommand*{\si@unt@countprefix}[3]{%

```

A check is made for binary units.

```

2921 \renewcommand*{\si@tempa}{binary}%
2922 \expandafter\expandafter\expandafter\ifx\expandafter
2923 \csname\expandafter\@gobble\string#1\opt@si\endcsname
2924 \si@tempa
2925 \expandafter\sisetup
2926 \else
2927 \expandafter\@gobble
2928 \fi
2929 {prefixbase=two}%
2930 \si@tempcnta#2\relax
2931 \ifsi@unt@per
2932 \expandafter\si@unt@invprefix
2933 \fi
2934 \advance\si@unt@prefixcnt\si@tempcnta\relax}
2935 \newcommand*{\si@unt@invprefix}{%
2936 \si@tempcntb\si@tempcnta\relax
2937 \si@tempcnta-\si@tempcntb\relax}

```

`\si@unt@litpower` For literal power output, the number can't simply be dumped, so a macro is needed.

```

\si@unt@litpower{\gobble}{\num}
2938 \newcommand*{\si@unt@litpower}[2]{\textsuperscript{#2}}

```

`\ifsi@unt@prepower` The handling of powers starts by ensuring that “pre” powers follow `\per` cleanly.

`\si@unt@power` Then a check is needed for inversion.

```

\si@unt@power{\power}{\num}
2939 \newif\ifsi@unt@prepower
2940 \newcommand*{\si@unt@power}[2]{%
2941   \renewcommand*{\si@tempa}{post}%
2942   \expandafter\expandafter\expandafter\ifx\expandafter
2943     \csname\expandafter\@gobble\string#1\opt@si\endcsname
2944     \si@tempa
2945     \expandafter\@gobbletwo
2946   \else
2947     \expandafter\si@unt@firstorsecond
2948   \fi
2949   }{\power}%
2950   \si@unt@powerdim #2 pt\relax
2951   \ifsi@frac\else
2952     \ifsi@unt@per
2953       \expandafter\expandafter\expandafter\si@unt@invpower
2954     \fi
2955   \fi
2956   \renewcommand*{\si@tempa}{post}%
2957   \si@unt@prepowertrue
2958   \expandafter\expandafter\expandafter\ifx\expandafter
2959     \csname\expandafter\@gobble\string#1\opt@si\endcsname
2960     \si@tempa
2961     \expandafter\si@unt@stackpower
2962   \else
2963     \si@log@debug{Power \strip@pt\si@unt@powerdim\space saved
2964       to be added after\MessageBreak next unit}%
2965   \fi}

```

`\si@unt@powerdim` To do sign-inversion on the power, a dimension is used (this allows non-integer values to be handled).

```

2966 \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@stkpower`

```

\si@unt@stkpower2967 \newcommand*{\si@unt@stackpower}{%
2968   \si@unt@prepowerfalse

```

A trap is used for -1 added to the denominator of a fraction.

```

2969   \si@unt@stkpower
2970   \ifsi@stickyper\else
2971     \si@unt@perfalse
2972     \si@unt@perseenfalse
2973   \fi}

```

The `\si@unt@stkpower` macro needs to be called from a few places, so is spun out from the above.


```

2974 \newcommand*{\si@unt@stkpower}{%
2975   \ifdim\si@unt@powerdim=\m@ne pt\relax
2976     \ifsi@frac\else
2977       \expandafter\expandafter\expandafter\si@unt@stkpwr
2978     \fi
2979   \else
2980     \expandafter\si@unt@stkpwr
2981   \fi}

```

Finally, the actual adding (set up to avoid problems with the \if above).

```

2982 \newcommand*{\si@unt@stkpwr}{%
2983   \ifdim\si@unt@powerdim=\z@\relax\else
2984     \renewcommand*{\si@tempa}{unit}%
2985     \expandafter\ifx
2986       \csname si@unt@lastadd\si@unt@checkstack\endcsname
2987       \si@tempa
2988       \si@log@debug{Adding power
2989         \strip@pt\si@unt@powerdim\space to output stack}%
2990       \si@unt@addtostack{power}{^{\num[fixdp=false]}{%
2991         \strip@pt\si@unt@powerdim}}}%
2992     \fi
2993   \fi
2994   \si@unt@powerdim\z@\relax}

```

`\si@unt@invpower` A macro to change the sign of the current power.

```

2995 \newcommand*{\si@unt@invpower}{%
2996   \si@tempdima\si@unt@powerdim\relax
2997   \si@unt@powerdim -\si@tempdima\relax

```

The power might end up as “1”, which is not wanted. So it is chucked away.

```

2998   \ifdim\si@unt@powerdim=\p@\relax
2999     \si@unt@powerdim\z@\relax
3000   \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@per3001 \newif\ifsi@unt@per
3002 \newif\ifsi@unt@perseen
3003 \si@newrobustcmd*{\si@per}{%
3004   \ifsi@unt@littest\else
3005     \ifsi@unt@litout
3006       \expandafter\expandafter\expandafter /%
3007     \else
3008       \ifsi@unt@litprefix
3009         \expandafter\expandafter\expandafter\expandafter
3010         \expandafter\expandafter\expandafter /%
3011       \else
3012         \expandafter\expandafter\expandafter\expandafter
3013         \expandafter\expandafter\expandafter\si@unt@per
3014       \fi
3015     \fi
3016   \fi}

```

```

3017 \newcommand*{\si@unt@per}{%
3018   \si@unt@firstorsecond{}{\per}%
3019   \ifsi@unt@per
3020     \ifsi@stickyper\else
3021       \expandafter\expandafter\expandafter\si@unt@perfalse
3022     \fi
3023   \else
3024     \expandafter\si@unt@pertrue
3025   \fi}
3026 \let\per\si@per

```

`\si@unt@reciptest` A test is needed for adding -1 when needed. The second macro is fired only if the power should be reciprocal.

```

\si@unt@recip
3027 \newcommand*{\si@unt@reciptest}{%
3028   \ifsi@unt@per
3029     \ifsi@unt@perseen
3030       \expandafter\expandafter\expandafter\si@unt@recip
3031     \fi
3032   \fi}
3033 \newcommand*{\si@unt@recip}{%
3034   \si@unt@powerdim\m@ne pt\relax
3035   \si@unt@stackpower}

```

`\si@unt@lastadda` Items cannot be added directly to the stacks (except the spacing stack, a) as the fractional handling may need to add the item to either storage area. By indicating the type of data to be added to the stack, problems can be avoided.

`\si@unt@addtostack{⟨type⟩}{⟨token⟩}`

```

3036 \newcommand*{\si@unt@lastadda}{}
3037 \newcommand*{\si@unt@lastaddb}{}
3038 \newcommand*{\si@unt@addtostack}[2]{%
3039   \renewcommand*{\si@tempa}{#1}%

```

Two consecutive items cannot be of the same type; there must be spaces between units, units between prefixes, *etc.*

```

3040   \expandafter\ifx
3041     \csname si@unt@lastadd\si@unt@checkstack\endcsname\si@tempa
3042     \expandafter\@gobbletwo
3043   \else
3044     \expandafter\si@unt@preplussp
3045   \fi
3046   {#1}{#2}}

```

`\si@unt@preplussp` The space added after a prefix needs to be ignored.

`\si@unt@preplussp{⟨type⟩}{⟨stack⟩}{⟨token⟩}{⟨gobble⟩}`

```

3047 \newcommand*{\si@unt@preplussp}[2]{%
3048   \renewcommand*{\si@tempa}{prefix+space}%
3049   \edef\si@tempb{%
3050     \csname si@unt@lastadd\si@unt@checkstack\endcsname+#1}%
3051   \ifx\si@tempa\si@tempb
3052     \expandafter\@gobbletwo
3053   \else
3054     \expandafter\si@unt@stack
3055   \fi
3056   {#1}{#2}}

```

`\si@unt@stack` The macro for actually doing the stacking up.

```
\si@unt@stack{<type>}{<tokens>}
3057 \newcommand*{\si@unt@stack}[2]{%
3058   \expandafter\renewcommand\expandafter*\expandafter{%
3059     \csname si@unt@lastadd\si@unt@checkstack\endcsname}{#1}%
```

A count is kept of the number of *units* added to each stack.

```
3060   \renewcommand*{\si@tempa}{#1}%
3061   \renewcommand*{\si@tempb}{unit}%
3062   \ifx\si@tempa\si@tempb
3063     \expandafter\si@unt@inccnt
3064   \fi
```

If a space is added, it is actually held until the next add.

```
3065   \renewcommand*{\si@tempb}{space}%
3066   \ifx\si@tempa\si@tempb
3067     \expandafter\si@unt@holdspace
3068   \else
3069     \expandafter\si@unt@addstack
3070   \fi
3071   {#2}}
```

`\si@unt@inccnt` The appropriate counter is added to.

```
3072 \newcommand*{\si@unt@inccnt}{%
3073   \expandafter\advance
3074     \csname si@unt@unitcnt\si@unt@checkstack\endcsname
3075     \@ne\relax}
```

`\si@unt@holdspace` Depending on the nature of the addition, it is either held or added to the stack.

`\si@unt@addstack` For the “b” space stack, a check is made to ensure that a space cannot be added before the first item.

`\si@unt@holdstacka` `\si@unt@holdspace{<tokens>}`

`\si@unt@holdstackb` `\si@unt@addstack{<tokens>}`

```
3076 \newcommand*{\si@unt@holdstacka}{}
3077 \newcommand*{\si@unt@holdstackb}{}
3078 \newcommand*{\si@unt@holdspace}[1]{%
3079   \renewcommand*{\si@tempa}{b}%
3080   \edef\si@tempb{\si@unt@checkstack}%
3081   \ifx\si@tempa\si@tempb
3082     \ifx\@empty\si@unt@stackb\@empty
3083     \else
3084       \expandafter\protected@edef
3085         \csname si@unt@holdstack\si@unt@checkstack\endcsname{#1}%
3086     \fi
3087   \else
3088     \expandafter\protected@edef
3089       \csname si@unt@holdstack\si@unt@checkstack\endcsname{#1}%
3090   \fi}
3091 \newcommand*{\si@unt@addstack}[1]{%
3092   \expandafter\protected@edef
3093     \csname si@unt@stack\si@unt@checkstack\endcsname
3094     {\csname si@unt@stack\si@unt@checkstack\endcsname
3095      \csname si@unt@holdstack\si@unt@checkstack\endcsname#1}%
3096   \expandafter\renewcommand\expandafter*\expandafter{%
3097     \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.

```

3098 \newcommand*{\si@unt@stackout}{%
3099   \si@unt@litouttrue
3100   \ifsi@frac
3101     \expandafter\si@unt@fracout
3102   \else
3103     \expandafter\si@unt@normout
3104   \fi}

```

`\si@unt@checkstack` Which stack is in use needs to be tested.

```

3105 \newcommand*{\si@unt@checkstack}{%
3106   \ifsi@frac
3107     \ifsi@unt@per
3108       \expandafter\expandafter\expandafter b%
3109     \else
3110       \expandafter\expandafter\expandafter a%
3111     \fi
3112   \else
3113     \expandafter a%
3114   \fi}

```

`\si@unt@spaceout` The space before a unit might not be needed, so it crops up a few times in the output routine.

```

3115 \newcommand*{\si@unt@spaceout}{%
3116   \ensuremath{\si@unt@spstack}}

```

`\si@unt@prefixout` If the prefix counter is not zero, then there is something to typeset.

```

3117 \newcommand*{\si@unt@prefixout}{%
3118   \ifnum\si@unt@prefixcnt=\z@\relax\else
3119     \ifsi@unt@num
3120       \si@out{\ensuremath{\{\}\si@prefixproduct{\}}}%
3121     \fi
3122     \si@unt@stackvalsep
3123     \let\si@expbase\si@prefixbase
3124     \num[fixdp=false]{e\the\si@unt@prefixcnt}%
3125   \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.

```

3126 \newcommand*{\si@unt@normout}{%
3127   \si@unt@prefixout
3128   \si@unt@spaceout
3129   \expandafter\si@unt@out\expandafter{\si@unt@stacka}}

```

`\si@unt@fracout` For fractions, some checks are needed.

```

3130 \newcommand*{\si@unt@fracout}{%
3131   \si@unt@notambig
3132   \ifx\@empty\si@unt@stacka\@empty
3133     \ifx\@empty\si@unt@stackb\@empty
3134       \ifsi@unt@litout\else
3135         \si@log@err{Empty fractional unit}{The unit
3136           argument\MessageBreak given does not contain any

```

```

3137         symbols}%
3138     \fi
3139 \else

```

With an empty numerator, no space is added

```

3140     \ifsi@slash
3141     \si@unt@prefixout
3142     \si@frac{}{\si@unt@stackb}%
3143 \else
3144     \si@unt@prefixout
3145     \si@unt@spaceout
3146     \si@frac{1}{\si@unt@stackb}%
3147 \fi
3148 \fi
3149 \else

```

If the denominator is empty, then the usual output system can be used.

```

3150     \ifx\@empty\si@unt@stackb\@empty
3151     \si@unt@normout
3152 \else
3153     \si@unt@prefixout
3154     \si@unt@spaceout
3155     \si@frac{\si@unt@stacka}{\si@unt@stackb}%
3156 \fi
3157 \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.

```

3158 \newcommand*{\si@unt@notambig}{%
3159 \ifnum\si@unt@unitcntb>\@ne\relax
3160     \ifsi@slash
3161         \ifsi@trapambigfrac
3162             \expandafter\expandafter\expandafter\expandafter
3163             \expandafter\expandafter\expandafter\si@unt@notabg
3164         \fi
3165     \fi
3166 \fi}
3167 \newcommand*{\si@unt@notabg}{%
3168 \protected@edef\si@unt@stackb{\si@openfrac\si@unt@stackb
3169     \si@closefrac}}

```

`\si@unt@out` 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 (`\SI`-type input). Non-Latin characters are also handled cleanly. As usual, `\scantokens` is used to make life easier.

```

\si@unt@out{\unit}

3170 \begingroup
3171 \catcode'\~=\active
3172 \catcode'\.=\active
3173 \gdef\si@unt@out#1{%
3174     \begingroup
3175     \si@unt@nonlatin
3176     \makeatletter
3177     \catcode'\~=\active

```

```

3178      \catcode`\.=\active
3179      \def~{\ensuremath{\si@unitspace}}%
3180      \def.\{\ensuremath{\si@unitsep}}%
3181      \endlinechar\m@ne
3182      \si@out{\scantokens{#1}}%
3183  \endgroup}
3184 \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.

```

3185 \newcommand*{\si@unt@nonlatin}{}
3186 \ifdefined\XeTeXrevision
3187   \renewcommand*{\si@unt@nonlatin}{%
3188     \catcode176=\active
3189     \catcode181=\active
3190     \catcode197=\active
3191     \si@unt@sym{176}{\si@sym@degree}%
3192     \si@unt@sym{181}{\si@sym@mu}%
3193     \si@unt@sym{197}{\si@sym@ringA}}%
3194 \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.

```

3195 \AtBeginDocument{
3196   \@ifpackageloaded{inputenc}
3197   {\@for\si@tempa:=latin1,latin5,latin9\do{
3198     \ifx\inputencodingname\si@tempa
3199       \renewcommand*{\si@unt@nonlatin}{%
3200         \catcode176=\active
3201         \catcode181=\active
3202         \catcode197=\active
3203         \si@unt@sym{176}{\si@sym@degree}%
3204         \si@unt@sym{181}{\si@sym@mu}%
3205         \si@unt@sym{197}{\si@sym@ringA}}%
3206       \fi}}
3207   {}}

```

`\si@unt@sym` A macro for declaring symbols: a copy of `\DeclareInputMath` from inputenc.
`\si@unt@sym{⟨charcode⟩}`

```

3208 \newcommand*{\si@unt@sym}[1]{%
3209   \bgroup
3210   \uccode`\~#1%
3211   \uppercase{%
3212     \egroup
3213     \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
\meter3214 \newunit{\kilogram}{kg}
\mole3215 \newunit{\metre}{m}
\kelvin3216 \newunit{\meter}{\metre}
\candela3217 \newunit{\mole}{mol}
\second3218 \newunit{\second}{s}
\ampere

```

```

3219 \newunit{\ampere}{A}
3220 \newunit{\kelvin}{K}
3221 \newunit{\candela}{cd}

```

`\Square` Unlike multiples (which can be skipped if needed), the basic powers are also always defined.

```

\ssquare
\squared3222 \newpower{\Square}{2}
\cubic3223 \newpower{\ssquare}{2}
\cubed3224 \newpower[post]{\squared}{2}
3225 \newpower{\cubic}{3}
3226 \newpower[post]{\cubed}{3}

```

`\tothe` A macro for arbitrary powers, which comes after the unit and so needs to be marked as such.

```

\si@tothe \tothe{<num>}
\raiseto \tothe{<num>}
\tothe@opt@si \raiseto{<num>}
\raiseto@opt@si3227 \newcommand*{\tothe}{\si@tothe{\tothe}}
3228 \newcommand*{\raiseto}{\si@tothe{\raiseto}}
3229 \newcommand*{\si@tothe}[2]{%
3230   \ifsi@unt@littest
3231     \expandafter\@gobbletwo
3232   \else
3233     \ifsi@unt@litout
3234       \expandafter\expandafter\expandafter\si@unt@litpower
3235     \else
3236       \expandafter\expandafter\expandafter\si@unt@power
3237     \fi
3238   \fi
3239   {#1}{#2}}
3240 \newcommand*{\tothe@opt@si}[post]
3241 \newcommand*{\raiseto@opt@si}{}

```

21.14 Locales

`\si@loc@load` When loading a locale, the setup is saved rather than applied. Anything other than simple settings should be inside `\addtolocale`, which is already defined.

```

\si@loc@load{<locale>}
3242 \newcommand*{\si@loc@load}[1]{%
3243   \let\si@loc@sisetup\sisetup
3244   \renewcommand*{\sisetup}[1]{%
3245     \expandafter\gdef\csname si@loc@#1\endcsname{##1}}
3246   \si@loadfile{#1}%
3247   \let\sisetup\si@loc@sisetup}

```

`\si@loc@set` Setting the locale transfers the settings to `\sisetup`, and runs any extra macros.

```

\si@loc@set{<locale>}
3248 \newcommand*{\si@loc@set}[1]{%
3249   \ifcsname si@loc@#1\endcsname
3250     \si@log@inf{Setting locale to '#1'}%
3251     \expandafter\expandafter\expandafter\expandafter
3252     \expandafter\expandafter\expandafter\si@temptoks
3253     \expandafter\expandafter\expandafter\expandafter

```

```

3254     \expandafter\expandafter\expandafter{%
3255     \expandafter\csname si@loc@#1\endcsname}%
3256     \expandafter\sisetup\expandafter{\the\si@temptoks}%
3257     \ifcsname si@loc@#1@extra\endcsname
3258     \csname si@loc@#1@extra\endcsname
3259     \fi
3260 \else
3261     \ifcsname si@loc@#1@extra\endcsname
3262     \si@log@inf{Setting locale to '#1'}%
3263     \csname si@loc@#1@extra\endcsname
3264     \else
3265     \si@log@warn{Unknown locale '#1'}%
3266     \fi
3267 \fi}

```

\si@loc@ltol The necessary loading for language modules occurs.

```

\si@loc@ltol{<list>}
3268 \newcommand*{\si@loc@ltol}[1]{%
3269 \def\si@tempa##1:##2\@nil{\si@loc@load{##1}}
3270 \@for\si@tempb:=#1\do{%
3271 \expandafter\si@tempa\si@tempb:\@nil}
3272 \AtBeginDocument{
3273 \ifpackageloaded{babel}
3274 {\def\si@tempa##1:##2:##3\@nil{%
3275 \expandafter\addto\expandafter{%
3276 \csname extras##2\endcsname}%
3277 {\si@loc@set{##1}}}%
3278 \@for\si@tempb:=#1\do{%
3279 \expandafter\si@tempa\si@tempb:\@nil}%
3280 \expandafter\selectlanguage\expandafter{\language}
3281 {\si@log@warn{babel not loaded \MessageBreak
3282 loctolang option ignored}}}}
3283 \AtBeginDocument{
3284 \ifpackageloaded{babel}
3285 {\renewcommand*{\si@loc@ltol}[1]{%
3286 \def\si@tempa##1:##2\@nil{\si@loc@load{##1}}%
3287 \@for\si@tempb:=#1\do{%
3288 \expandafter\si@tempa\si@tempb:\@nil}%
3289 \def\si@tempa##1:##2:##3\@nil{%
3290 \expandafter\addto\expandafter{%
3291 \csname extras##2\endcsname}%
3292 {\si@loc@set{##1}}}%
3293 \@for\si@tempb:=#1\do{%
3294 \expandafter\si@tempa\si@tempb:\@nil}}}}
3295 {\renewcommand*{\si@loc@ltol}[1]{%
3296 \si@log@warn{babel not loaded \MessageBreak
3297 loctolang option ignored}}}}

```

\addtolocale Arbitrary macros may need to be added to the locale.

```

\addtolocale{<locale>}{<commands>}
3298 \newcommand*{\addtolocale}[2]{%
3299 \si@addtocname{si@loc@#1@extra}{#2}}

```


21.15 Output routine

`\si@out` 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.

`\si@out{⟨text⟩}`

```
3300 \begingroup
3301   \catcode`\^=\active\relax
3302   \catcode`\-=\active\relax
3303   \gdef\si@out#1{%
3304     \begingroup
3305       \catcode`\^=\active\relax
3306       \makeatletter
3307       \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`.

```
3308     \ifsi@fam@set\else
3309       \expandafter\si@fam@set
3310     \fi
3311     \si@colourcmd{\si@colour}%
3312     \text{\si@fam@italic\si@fam@bold\si@fam@text
```

The correct mode is selected, and the input is handed off for typesetting.

```
3313     \ifsi@textmode
3314       \expandafter\si@out@text
3315     \else
3316       \expandafter\si@out@maths
3317     \fi
3318     {\scantokens{#1}}}%
3319   \endgroup
3320   \check@mathfonts}
```

`\si@out@text` Output occurs with the correct changes to superscript behaviour.

`\si@out@maths` `\si@out@text{⟨text⟩}`
`\si@out@maths{⟨text⟩}`

```
3321   \gdef\si@out@text#1{%
3322     \let^\si@out@sp
3323     \let\textsuperscript\si@out@sp
3324     \catcode`\-=\active\relax
3325     \let-\si@out@minus
3326     #1}
3327   \gdef\si@out@maths#1{%
3328     \let^\sp
3329     \let\textsuperscript\sp
3330     $\si@fam@maths{#1}$}
3331 \endgroup
```

`\si@out@sp` `\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/maths mode issues.

`\si@out@sp{⟨text⟩}`

```

3332 \newcommand*{\si@out@sp}[1]{\ensuremath{\text{\#1}}}}

\si@out@minus  An active minus sign is needed.

3333 \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.

3334 \newif\ifsi@out@num

\si@out@num  For numerical output, the default fonts are controlled slightly differently to text
output.
\si@out@num{\num}

3335 \newcommand*{\si@out@num}[1]{%
3336   \begingroup
3337     \si@out@numtrue
3338     \si@out{\#1}%
3339   \endgroup}

```

21.16 Finalisation

```

\si@extension      To keep the code easy to maintain, the reusable filename components are macros
\si@fileprefix    rather than literals.

3340 \newcommand*{\si@extension}{cfg}
3341 \newcommand*{\si@fileprefix}{si-}

\si@ifl@aded      A bit of borrowing from the LATEX kernel. A copy of \@ifl@aded is needed as
\si@ifloaded      things aren't always done in the preamble by siunitx.
\si@ifloaded{\package}

3342 \newcommand*{\si@ifl@aded}{}
3343 \let\si@ifl@aded\@ifl@aded
3344 \newcommand*{\si@ifloaded}[1]{%
3345   \si@ifl@aded\si@extension\si@fileprefix#1}}

\si@loadfile      Loading configuration files is handled here.
\si@loadfile{\file}

3346 \newcommand*{\si@loadfile}[1]{%
3347   \si@ifloaded{#1}}
3348   {\InputIfFileExists{\si@fileprefix#1.\si@extension}
3349     {}
3350     {\si@log@err{Failed to load file
3351       \si@fileprefix#1.\si@extension}
3352       {The configuration file requested could not be
3353         found}}}}

\requiresiconfigs The configuration files depend on each other.
\requiresiconfigs{\cfile}

3354 \newcommand*{\requiresiconfigs}[1]{%
3355   \@for\si@tempb:=#1\do{\si@loadfile{\si@tempb}}}}

\si@loademfile    For emulation files, an additional check is made.
\si@loademfile{\file}

3356 \newcommand*{\si@loademfile}[1]{%

```

```

3357 \@ifpackageloaded{#1}
3358   {\si@log@err{Emulation clash for package '#1'}
3359    {You have asked for emulation of package
3360     '#1'\MessageBreak
3361     (perhaps by giving siunitx a back-compatibility
3362     option)\MessageBreak but the package is already
3363     loaded!}}
3364   {\si@loadfile{#1}}}
```

`\si@emclash` A macro for emulation clashes.

```

\si@emclash{<package>}{<package>}
3365 \newcommand*{\si@emclash}[2]{%
3366   \si@log@err{Emulation clash: '#1' and '#2'}
3367   {You have asked for emulation of package '#1'\MessageBreak
3368    but have already loaded emulation of '#2'}}
```

`\si@emulating` For packages that are emulated, the L^AT_EX mechanism to prevent re-loading is used. The list of packages to check at the start of the document also has to be altered.

```

\si@emulating{<package>}{<version>}
3369 \newcommand*{\si@emulating}[2]{%
3370   \@namedef{ver@#1.sty}{#2 siunitx emulation of #1}%
3371   \let\si@tempa\si@blockpkgs
3372   \renewcommand*{\si@blockpkgs}{}%
3373   \@for\si@tempb:=\si@tempa\do{%
3374     \renewcommand*{\si@tempa}{#1}%
3375     \ifx\si@tempa\si@tempb\else
3376       \lowercase{\edef\si@tempa{#1}}%
3377       \lowercase{\edef\si@tempc{\si@tempb}}%
3378       \ifx\si@tempa\si@tempc
3379         \@namedef{ver@\si@tempc.sty}{#2 siunitx emulation of
3380          #1}%
3381       \else
3382         \si@addtolist{\si@blockpkgs}{\si@tempb}%
3383       \fi
3384     \fi}%
3385   \let\si@tempa\si@checkpkgs
3386   \renewcommand*{\si@checkpkgs}{}%
3387   \renewcommand*{\si@tempb}{#1}%
3388   \@for\si@tempc:=\si@tempa\do{%
3389     \ifx\si@tempb\si@tempc\else
3390       \si@addtolist{\si@checkpkgs}{\si@tempc}%
3391     \fi}}
```

With the siunitx kernel macros defined, the package can now run through finalisation. First, the default key values are set. The user options are then processed.

```

3392 \sisetup{
3393   addsign=none,
3394   allowzeroexp=false,
3395   angformat=unchanged,
3396   astroang=false,% (
3397   closeerr=),%(
3398   closefrac=),
```

```

3399 colour=black,
3400 colourall=false,
3401 colourneg=false,
3402 decimalsymbol=fullstop,
3403 detectdisplay=true,
3404 digitsep=thin,
3405 dp=3,
3406 eVcorra=0.3ex,
3407 eVcorrb=0ex,
3408 errspace=none,
3409 fixdp=false,
3410 inlinebold=text,
3411 load=default,
3412 mathsrn=mathrm,
3413 mathssf=mathsf,
3414 mathstt=mathtt,
3415 mode=maths,
3416 negcolour=red,
3417 noload={},
3418 numaddn={},%(
3419 numcloseerr=),%
3420 numdecimal={.,},
3421 numdigits=0123456789,
3422 numexp=eEdD,
3423 numgobble={},
3424 numopenerr=(,% )
3425 numprod=x,
3426 numsign=+-\pm\mp,
3427 obeybold=false,
3428 obeyitalic=false,
3429 obeymode=false,
3430 openerr=(,% )
3431 openfrac=(,% )
3432 padangle=small,
3433 padnumber=lead,
3434 per=reciprocal,
3435 prefixbase=ten,
3436 prefixproduct=times,
3437 prefixsymbolic=true,
3438 prespace=false,
3439 redefsymbols=true,
3440 repeatunits=true,
3441 retainplus=false,
3442 seperr=false,
3443 sepfour=false,
3444 sign=plus,
3445 slash=slash,
3446 stickyper=false,
3447 strictarc=true,
3448 tabalignexp=true,
3449 tabautofit=false,
3450 tabformat=3.2,
3451 tabnumalign=centredecimal,
3452 tabtextalign=centre,

```

```

3453 tabunitalign=left,
3454 textrm=rmfamily,
3455 textsf=sffamily,
3456 texttt=ttfamily,
3457 tightpm=false,
3458 trapambigerr=true,
3459 trapambigfrac=true,
3460 unitsep=thin,
3461 valuesep=thin,
3462 xspace=false}
3463 \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.

```

3464 \ifx\@empty\si@emulate\@empty\else
3465   \@for\si@tempa:=\si@emulate\do{%
3466     \si@loademfile{\si@tempa}}
3467 \fi

```

`\si@expanddefault` For turning the list of default choices into a loadable list.

```

3468 \newcommand*{\si@expanddefault}[2]{%
3469   \expandafter\ifx\expandafter\@empty\csname si@#1\endcsname
3470     \@empty
3471   \else
3472     \renewcommand*{\si@tempb}{default}%
3473     \renewcommand*{\si@tempc}{}%
3474     \expandafter\@for\expandafter\si@tempa\expandafter
3475       :\expandafter=\csname si@#1\endcsname\do{%
3476       \ifx\si@tempa\si@tempb
3477         \si@addtolist{\si@tempc}{#2}%
3478       \else
3479         \si@addtolist{\si@tempc}{\si@tempa}%
3480       \fi}
3481     \expandafter\edef\csname si@#1\endcsname{\si@tempc}%
3482     \expandafter\si@addtolist\expandafter{%
3483       \csname si@no#1\endcsname}%
3484     {default}%
3485     \renewcommand*{\si@tempc}{}%
3486     \expandafter\@for\expandafter\si@tempa\expandafter
3487       :\expandafter=\csname si@#1\endcsname\do{%
3488       \si@switchfalse
3489       \expandafter\@for\expandafter\si@tempb\expandafter
3490         :\expandafter=\csname si@no#1\endcsname\do{%
3491         \ifx\si@tempa\si@tempb
3492           \si@switchtrue
3493         \fi
3494         \ifsi@switch\else
3495           \si@addtolist{\si@tempc}{\si@tempa}%
3496         \fi}}
3497     \@for\si@tempa:=\si@tempc\do{%
3498       \si@loadfile{\si@tempa}}%
3499   \fi}

```

The configuration and abbreviation files are loaded.

```

3500 \si@expanddefault{load}{prefix,named,addn,prefixed,accepted,%
3501   physical,abbr}

```

The very last job is to load a local configuration file, if one exists, and restore catcodes.

```

3502 \IfFileExists{siunitx.cfg}
3503   {\si@log@inf{Local configuration file found}%
3504     \InputIfFileExists{siunitx.cfg}{}}{}
3505   {}
3506 \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.

```

\zepto3507 \ProvidesFile{si-prefix.cfg}
\atto3508   [\csname si@svn@version\endcsname siunitx:
\femto3509   SI Multiple prefixes]
\pico3510 \newprefix{\yocto}{-24}{y}
\nano3511 \newprefix{\zepto}{-21}{z}
\micro3512 \newprefix{\atto}{-18}{a}
\Micro3513 \newprefix{\femto}{-15}{f}
\milli3514 \newprefix{\pico}{-12}{p}
\centi3515 \newprefix{\nano}{-9}{n}
\deci
3516 \ifsi@old@OHM
3517   \newprefix{\Micro}{-6}{\si@sym@mu}
3518 \else
3519   \ifsi@gensymb\else
3520     \newprefix{\micro}{-6}{\si@sym@mu}
3521   \fi
3522 \fi
3523 \newprefix{\milli}{-3}{m}
3524 \newprefix{\centi}{-2}{c}
3525 \newprefix{\deci}{-1}{d}

```

`\deca` Now the large ones.

```

\hecto3526 \newprefix{\deca}{1}{da}
\kilo3527 \newprefix{\hecto}{2}{h}
\mega3528 \newprefix{\kilo}{3}{k}
\giga3529 \newprefix{\mega}{6}{M}
\tera3530 \newprefix{\giga}{9}{G}
\peta3531 \newprefix{\tera}{12}{T}
\exa3532 \newprefix{\peta}{15}{P}
\zetta3533 \newprefix{\exa}{18}{E}
\yotta3534 \newprefix{\zetta}{21}{Z}
3535 \newprefix{\yotta}{24}{Y}

```

`\deka` Apparently, “deka” is common in the US for deca.

```
3536 \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.

```
3537 \newunit{\gram}{g}
```

```
3538 \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.

```
\coulomb \farad3539 \ProvidesFile{si-named.cfg}
```

```
\Gray3540 [\csname si@svn@version\endcsname siunitx: SI Named units]
```

```
\ggray3541 \newunit{\becquerel}{Bq}
```

```
\hertz3542 \newunit{\coulomb}{C}
```

```
\henry3543 \newunit{\farad}{F}
```

```
\joule3544 \newunit{\Gray}{Gy}
```

```
\katal3545 \newunit{\ggray}{Gy}
```

```
\lumen3546 \newunit{\hertz}{Hz}
```

```
\lux3547 \newunit{\henry}{H}
```

```
\lux3548 \newunit{\joule}{J}
```

```
\newton3549 \newunit{\katal}{kat}
```

```
3550 \newunit{\lumen}{lm}
```

```
3551 \newunit{\lux}{lx}
```

```
3552 \newunit{\newton}{N}
```

`\ohm` Some testing is needed for `unitsdef` compatibility.

```
\Ohm3553 \ifsi@old@OHM
```

```
\pascal3554 \newunit{\Ohm}{\si@sym@Omega}
```

```
\siemens3555 \else
```

```
\sievert3556 \ifsi@gensymb\else
```

`\tesla` To be on the safe side, use `\provideunit`.

```
\volt3557 \provideunit{\ohm}{\si@sym@Omega}
```

```
\watt3558 \fi
```

```
\weber3559 \fi
```

```
3560 \newunit{\pascal}{Pa}
```

```
3561 \newunit{\siemens}{S}
```

```
3562 \newunit{\sievert}{Sv}
```

```
3563 \newunit{\tesla}{T}
```

```
3564 \newunit{\volt}{V}
```

```
3565 \newunit{\watt}{W}
```

```
3566 \newunit{\weber}{Wb}
```

`\celsius` The degree celsius is a named unit.

```
\Celsius3567 \ifsi@old@OHM
```

```
3568 \newunit{\Celsius}{\si@sym@celsius}
```

```
3569 \else
```

```
3570 \ifsi@gensymb\else
```

```
3571 \newunit{\celsius}{\si@sym@celsius}
```

```

3572 \fi
3573 \fi

```

```

\radian The radian and steradian are officially derived units.
\steradian3574 \newunit{\radian}{rad}
3575 \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`.

```

3576 \ProvidesFile{si-prefixed.cfg}
3577 [\csname si@svn@version\endcsname siunitx:
3578 SI Prefixed units]
3579 \requiresiconfigs{prefix,named,accepted,physical}

```

`\picometre` Extra distances.

```

\nanometre3580 \newunit{\picometre}{\pico\metre}
\micrometre3581 \newunit{\nanometre}{\nano\metre}
\millimetre3582 \newunit{\micrometre}{\micro\metre}
\centimetre3583 \newunit{\millimetre}{\milli\metre}
\decimetre3584 \newunit{\centimetre}{\centi\metre}
\kilometre3585 \newunit{\decimetre}{\deci\metre}
3586 \newunit{\kilometre}{\kilo\metre}

```

`\femtogram` Extra masses.

```

\picogram3587 \newunit{\femtogram}{\femto\gram}
\nanogram3588 \newunit{\picogram}{\pico\gram}
\microgram3589 \newunit{\nanogram}{\nano\gram}
\milligram3590 \newunit{\microgram}{\micro\gram}
3591 \newunit{\milligram}{\milli\gram}

```

`\femtomole` Now some moles.

```

\picomole3592 \newunit{\femtomole}{\femto\mole}
\nanomole3593 \newunit{\picomole}{\pico\mole}
\micromole3594 \newunit{\nanomole}{\nano\mole}
\millimole3595 \newunit{\micromole}{\micro\mole}
3596 \newunit{\millimole}{\milli\mole}

```

`\attosecond` Prefixed seconds.

```

\femtosecond3597 \newunit{\attosecond}{\atto\second}
\picosecond3598 \newunit{\femtosecond}{\femto\second}
\nanosecond3599 \newunit{\picosecond}{\pico\second}
\microsecond3600 \newunit{\nanosecond}{\nano\second}
\millisecond3601 \newunit{\microsecond}{\micro\second}
3602 \newunit{\millisecond}{\milli\second}

```

`\picoampere` The last prefixed base units are amperes.

```

\nanoampere3603 \newunit{\picoampere}{\pico\ampere}
\microampere3604 \newunit{\nanoampere}{\nano\ampere}
\milliampere3605 \newunit{\microampere}{\micro\ampere}
\kiloampere3606 \newunit{\milliampere}{\milli\ampere}
3607 \newunit{\kiloampere}{\kilo\ampere}

```



```

\millivolt    More electricity-related units.
\kilovolt3608 \newunit{\millivolt}{\milli\volt}
\milliwatt3609 \newunit{\kilovolt}{\kilo\volt}
\kilowatt3610 \newunit{\milliwatt}{\milli\watt}
\megawatt3611 \newunit{\kilowatt}{\kilo\watt}
\femtofarad3612 \newunit{\megawatt}{\mega\watt}
\picofarad3613 \newunit{\femtofarad}{\femto\farad}
\nanofarad3614 \newunit{\picofarad}{\pico\farad}
\microfarad3615 \newunit{\nanofarad}{\nano\farad}
\millifarad3616 \newunit{\microfarad}{\micro\farad}
\millisiemens3617 \newunit{\millifarad}{\milli\farad}
\millisiemens3618 \newunit{\millisiemens}{\milli\siemens}

\kiloohm    For resistance, checks are needed again for the definition of \ohm.
\megaohm3619 \ifsi@old@OHM
\gigaohm3620 \newunit{\kiloohm}{\kilo\Ohm}
3621 \newunit{\megaohm}{\mega\Ohm}
3622 \newunit{\gigaohm}{\giga\Ohm}
3623 \else
3624 \ifsi@gensymb\else
3625 \newunit{\kiloohm}{\kilo\ohm}
3626 \newunit{\megaohm}{\mega\ohm}
3627 \newunit{\gigaohm}{\giga\ohm}
3628 \fi
3629 \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.
\cubiccentimetre3630 \newunit{\microlitre}{\micro\litre}
\centimetrecubed3631 \newunit{\millilitre}{\milli\litre}
\cubicmicrometre3632 \newunit{\cubicmetre}{\metre\cubed}
\cubicmillimetre3633 \newunit{\cubiccentimetre}{\centi\metre\cubed}
\cubicdecimetre3634 \newunit{\centimetrecubed}{\centi\metre\cubed}
3635 \newunit{\cubicmicrometre}{\micro\metre\cubed}
3636 \newunit{\cubicmillimetre}{\milli\metre\cubed}
3637 \newunit{\cubicdecimetre}{\cubic\deci\metre}

\squaremetre    Areas, with metre spelled corrected; \are and \hectare are in the “temporarily
\squarecentimetre    accepted” file.
\centimetresquared3638 \newunit{\squaremetre}{\Square\metre}
\squarekilometre3639 \newunit{\squarecentimetre}{\Square\centi\metre}
3640 \newunit{\centimetresquared}{\centi\metre\squared}
3641 \newunit{\squarekilometre}{\Square\kilo\metre}

\millijoule    Some energy is needed by now! Notice the definition of \kilowatthour
\kilojoule3642 \newunit{\millijoule}{\milli\joule}
\megajoule3643 \newunit{\kilojoule}{\kilo\joule}
\millielelectronvolt3644 \newunit{\megajoule}{\mega\joule}
\kiloelectronvolt3645 \newunit{\millielelectronvolt}{\milli\electronvolt}
\megaelectronvolt3646 \newunit{\kiloelectronvolt}{\kilo\electronvolt}
\gigaelectronvolt3647 \newunit{\megaelectronvolt}{\mega\electronvolt}
\teraelectronvolt3648 \newunit{\gigaelectronvolt}{\giga\electronvolt}
\kilowatthour

```

```

3649 \newunit{\teraelectronvolt}{\tera\electronvolt}
3650 \newunit[unitsep=none]{\kilowatthour}{\kilo\watt\hour}

\millihertz   Frequencies.
\kilohertz3651 \newunit{\millihertz}{\milli\hertz}
\megahertz3652 \newunit{\kilohertz}{\kilo\hertz}
\gigahertz3653 \newunit{\megahertz}{\mega\hertz}
\terahertz3654 \newunit{\gigahertz}{\giga\hertz}
3655 \newunit{\terahertz}{\tera\hertz}

\millinewton  A few more from various areas.
\kilonewton3656 \newunit{\millinewton}{\milli\newton}
\hectopascal3657 \newunit{\kilonewton}{\kilo\newton}
\megabecquerel3658 \newunit{\hectopascal}{\hecto\pascal}
\millisievert3659 \newunit{\megabecquerel}{\mega\becquerel}
3660 \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.

```

\micA3661 \ProvidesFile{si-abbr.cfg}
\mA3662 [\csname si@svn@version\endcsname siunitx: Abbreviated units]
\kA3663 \requiresiconfigs{prefix,named,accepted,physical}
3664 \newunit{\pA}{\pico\ampere}
3665 \newunit{\nA}{\nano\ampere}
3666 \newunit{\micA}{\micro\ampere}
3667 \newunit{\mA}{\milli\ampere}
3668 \newunit{\kA}{\kilo\ampere}

```

\Hz Then frequencies.

```

\mHz3669 \newunit{\Hz}{\hertz}
\kHz3670 \newunit{\mHz}{\milli\hertz}
\MHz3671 \newunit{\kHz}{\kilo\hertz}
\GHz3672 \newunit{\MHz}{\mega\hertz}
\THz3673 \newunit{\GHz}{\giga\hertz}
3674 \newunit{\THz}{\tera\hertz}

```

\fmol Amounts of substance.

```

\pmol3675 \newunit{\fmol}{\femto\mole}
\nmol3676 \newunit{\pmol}{\pico\mole}
\micmol3677 \newunit{\nmol}{\nano\mole}
\mmol3678 \newunit{\micmol}{\micro\mole}
3679 \newunit{\mmol}{\milli\mole}

```

\kV Potentials.

```

\mV3680 \newunit{\kV}{\kilo\volt}
3681 \newunit{\mV}{\milli\volt}

```

\ml Volumes and areas

```

\micl3682 \provideunit{\ml}{\milli\litre}
\cmc3683 \provideunit{\micl}{\micro\litre}
\dmc
\cms

```

```

3684 \newunit{\cmc}{\centi\metre\cubed}
3685 \newunit{\dmc}{\deci\metre\cubed}
3686 \newunit{\cms}{\centi\metre\squared}

\kg  Masses.
\fg3687 \newunit{\kg}{\kilo\gram}
\pg  There is a name clash with babel here in French; hopefully there will not be too
\nanog many complaints.
\micg3688 \provideunit{\fg}{\femto\gram}
\mg3689 \newunit{\pg}{\pico\gram}
\amu3690 \newunit{\nanog}{\nano\gram}
3691 \newunit{\micg}{\micro\gram}
3692 \newunit{\mg}{\milli\gram}
3693 \newunit{\amu}{\atomicmass}

\kJ  Energies.
\eV3694 \newunit{\kJ}{\kilo\joule}
\meV3695 \newunit{\eV}{\electronvolt}
\keV3696 \newunit{\meV}{\milli\electronvolt}
\MeV3697 \newunit{\keV}{\kilo\electronvolt}
\GeV3698 \newunit{\MeV}{\mega\electronvolt}
\TeV3699 \newunit{\GeV}{\giga\electronvolt}
\kWh3700 \newunit{\TeV}{\tera\electronvolt}
3701 \newunit[unitsep=none]{\kWh}{\kilo\watt\hour}

\picom Lengths.
\nm3702 \newunit{\picom}{\pico\metre}
\micm3703 \newunit{\nm}{\nano\metre}
\mm3704 \newunit{\micm}{\micro\metre}
\cm3705 \newunit{\mm}{\milli\metre}
\dm3706 \newunit{\cm}{\centi\metre}
\km3707 \newunit{\dm}{\deci\metre}
3708 \newunit{\km}{\kilo\metre}

\Sec  Finally, times.
\as3709 \newunit{\Sec}{\second}
\fs3710 \newunit{\as}{\atto\second}
\ps3711 \newunit{\fs}{\femto\second}
\ns  The letter class (and others) define \ps for postscripts, so \provideunit is best
\mic  here.
\ms3712 \provideunit{\ps}{\pico\second}
3713 \newunit{\ns}{\nano\second}
3714 \newunit{\mics}{\micro\second}
3715 \newunit{\ms}{\milli\second}

```

22.5 Additional (temporary) SI units

```

\angstrom  Some units are “temporarily” acceptable for use in the SI system. These are
\are  defined here, although some are in very general use.
\hectare3716 \ProvidesFile{si-addn.cfg}
\bar3717  [\csname si@svn@version\endcsname siunitx:
\BAR
\bbar
\millibar
\gal
\curie
\roentgen
\rad
\rem

```

```

3718 SI Additional units]
3719 \newunit{\angstrom}{\si@sym@ringA}
3720 \newunit{\are}{a}
3721 \newunit{\hectare}{\hecto\are}
3722 \newunit{\barn}{b}
3723 \newunit{\BAR}{bar}
3724 \newunit{\bbar}{bar}
3725 \newunit{\millibar}{\milli\BAR}
3726 \newunit{\gal}{Gal}
3727 \newunit{\curie}{Ci}
3728 \newunit{\roentgen}{R}
3729 \newunit{\rad}{rad}
3730 \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
\hour3731 \ProvidesFile{si-accepted.cfg}
\Day3732 [\csname si@svn@version\endcsname siunitx: SI Accepted units]
\dday3733 \newunit{\minute}{min}
\degree3734 \newunit{\hour}{h}
\Degree3735 \newunit{\Day}{d}
\arcmin3736 \newunit{\dday}{d}
\arcsec3737 \ifsi@old@OHM
\litre3738 \newunit[valuesep=none]{\Degree}{\si@sym@degree}
\liter3739 \else
\tonne3740 \ifsi@gensymb\else
\neper3741 \newunit[valuesep=none]{\degree}{\si@sym@degree}
\bel3742 \fi
\bel3743 \fi
\percent3744 \newunit[valuesep=none]{\arcmin}{\si@sym@minute}
3745 \newunit[valuesep=none]{\arcsec}{\si@sym@second}
3746 \newunit{\litre}{l}
3747 \newunit{\liter}{L}
3748 \newunit{\tonne}{t}
3749 \newunit{\neper}{Np}
3750 \newunit{\bel}{B}
3751 \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.

```

\electronvolt3752 \ProvidesFile{si-physical.cfg}
\atomicmassunit3753 [\csname si@svn@version\endcsname siunitx:
\atomicmass3754 SI Physically-measured units]
3755 \newcommand*{\si@eVspacea}{\text{\kern-\si@eVcorra}}%
3756 \newcommand*{\si@eVspaceb}{\text{\kern-\si@eVcorrb}}%
3757 \newunit{\electronvolt}{\protect\si@eVspacea V\protect%
3758 \si@eVspaceb}

```

```

3759 \newunit{\atomicmass}{u}
3760 \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 outside of the SI system, they are used a lot. These are set up using `\provideunit`
`\molar` as people may have their own definitions.
`\Molar`

```

\torr3761 \ProvidesFile{si-synchem.cfg}
\dalton3762 [\csname si@svn@version\endcsname siunitx: Units for
3763     synthetic chemists]
3764 \requiresiconfigs{prefix}
3765 \newunit{\mmHg}{mmHg}
3766 \newunit{\molar}{\mole\per\cubic\deci\metre}
3767 \newunit{\Molar}{\textsc{m}}
3768 \newunit{\torr}{Torr}
3769 \newunit{\dalton}{Da}

```

23.2 High-energy physics

Some units inspired by hepunits.

```

3770 \ProvidesFile{si-hep.cfg}
3771 [\csname si@svn@version\endcsname siunitx: Units for
3772     high-energy physics]
3773 \requiresiconfigs{prefix,named}

```

`\micron` These specific to high-energy physics, but are not defined elsewhere in siunitx.⁵⁴

```

\mrad3774 \provideunit{\micron}{\micro\metre}
\gauss3775 \newunit{\mrad}{\milli\rad}
3776 \newunit{\gauss}{G}

```

`\clight` 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.

```

3777 \newunit{\clight}{\ensuremath{\mathnormal{c}}}
3778 \newunit{\eVperc}{\eV\per\clight}

```

`\nanobarn` Various prefixed barns.

```

\picobarn3779 \newunit{\nanobarn}{\nano\barn}
\femtobarn3780 \newunit{\picobarn}{\pico\barn}
\attobarn3781 \newunit{\femtobarn}{\femto\barn}
\zeptobarn3782 \newunit{\attobarn}{\atto\barn}
\yoctobarn3783 \newunit{\zeptobarn}{\zepto\barn}
3784 \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.

```
\pb3785 \provideunit{\nb}{\nano\barn}
\fb3786 \provideunit{\pb}{\pico\barn}
\ab3787 \provideunit{\fb}{\femto\barn}
\zb3788 \provideunit{\ab}{\atto\barn}
\yb3789 \provideunit{\zb}{\zepto\barn}
3790 \provideunit{\yb}{\yocto\barn}
```

23.3 Astronomy

`\parsec` For astronomy, the `\parsec` unit is needed.

```
\lightyear3791 \ProvidesFile{si-astro.cfg}
3792 [\csname si@svn@version\endcsname siunitx:
3793 Units for astronomy]
3794 \newunit{\parsec}{pc}
3795 \newunit{\lightyear}{ly}
```

23.4 Binary units

`\kibi` The binary units, as specified by the IEC and made available by Slunits. First, the binary prefixes.

```
\gibi3796 \ProvidesFile{si-binary.cfg}
\tebi3797 [\csname si@svn@version\endcsname siunitx: Binary units]
\pebi3798 \newprefix[binary]{\kibi}{10}{Ki}
\exbi3799 \newprefix[binary]{\mebi}{20}{Mi}
\zebi3800 \newprefix[binary]{\gibi}{30}{Gi}
\yobi3801 \newprefix[binary]{\tebi}{40}{Ti}
3802 \newprefix[binary]{\pebi}{50}{Pi}
3803 \newprefix[binary]{\exbi}{60}{Ei}
3804 \newprefix[binary]{\zebi}{70}{Zi}
3805 \newprefix[binary]{\yobi}{80}{Yi}
```

`\bit` Now the units.

```
\byte3806 \newunit{\bit}{bit}
3807 \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

```
3808 \ProvidesFile{si-UK.cfg}
3809 [\csname si@svn@version\endcsname siunitx: UK locale]
3810 \sisetup{
3811 unitsep=thin,
3812 expproduct=times,
3813 valuesep=thin,
```

⁵⁵The package author is in the UK.

```

3814 decimalsymbol=fullstop,
3815 digitsep=thin,
3816 sepfour=false}

```

24.2 United States

The same as for the UK.

```

3817 \ProvidesFile{si-USA.cfg}
3818 [\csname si@svn@version\endcsname siunitx: USA locale]
3819 \sisetup{
3820   unitsep=thin,
3821   expproduct=times,
3822   valuesep=thin,
3823   decimalsymbol=fullstop,
3824   digitsep=thin,
3825   sepfour=false}

```

24.3 Germany

Germany, hopefully.

```

3826 \ProvidesFile{si-DE.cfg}
3827 [\csname si@svn@version\endcsname siunitx: Germany locale]
3828 \sisetup{
3829   unitsep=cdot,
3830   valuesep=thin,
3831   decimalsymbol=comma,
3832   expproduct=cdot,
3833   digitsep=thin,
3834   sepfour=false}

```

24.4 South Africa

Design decisions taken from the same section of Slstyle.

```

3835 \ProvidesFile{si-ZA.cfg}
3836 [\csname si@svn@version\endcsname siunitx:
3837   South Africa locale]
3838 \sisetup{
3839   unitsep=cdot,
3840   valuesep=thin,
3841   expproduct=times,
3842   decimalsymbol=comma,
3843   digitsep=thin,
3844   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.

```
3845 \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.

```
3846 \ProvidesFile{si-units.cfg}
3847 [\csname si@svn@version\endcsname siunitx: Emulation of units]
3848 \si@emulating{units}{1998/08/04 v0.9b}
3849 \si@ifloaded{SIunits}
3850 {\si@emclash{units}{SIunits}\endinput}{}
3851 \si@ifloaded{sistyle}
3852 {\si@emclash{units}{sistyle}\endinput}{}

```

To emulate `units`, `\per` must give fractions.

```
3853 \sisetup{per=fraction, fraction=nice, obeybold, inlinebold=maths,
3854 , obeymode}
3855 \ifsi@old@tight
3856 \sisetup{valuesep=thin}
3857 \fi
3858 \ifsi@old@loose
3859 \sisetup{valuesep=space}
3860 \fi
3861 \ifsi@old@ugly
3862 \sisetup{fraction=ugly}
3863 \fi

```

`\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.

`\unit[⟨num⟩]{⟨unit⟩}`

```
3864 \si@newrobustcmd*{\unit}[2][]{%
3865 \ifmmode
3866 \SI{#1}{#2}%
3867 \else
3868 \SI[obeyfamily,obeyitalic]{#1}{#2}%
3869 \fi}

```

`\unitfrac` `\unitfrac` is a bit more of a hack.

`\unitfrac[⟨num⟩]{⟨numerator⟩}{⟨denominator⟩}`

```
3870 \si@newrobustcmd*{\unitfrac}[3][]{%
3871 \begingroup
3872 \si@fam@mode%
3873 \ifmmode\else
3874 \sisetup{obeyfamily,obeyitalic}%
3875 \fi
3876 \si@ifnotmtarg{#1}
3877 {\num{#1}\ensuremath{\si@valuesep}}%
3878 \si@frac{#2}{#3}
3879 \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.

```
3880 \ProvidesFile{si-unitsdef.cfg}
3881 [\csname si@svn@version\endcsname siunitx:
3882   Emulation of unitsdef]
3883 \si@emulating{unitsdef}{2005/01/04 v0.2}
3884 \si@ifloaded{SIunits}
3885 {\si@emclash{unitsdef}{SIunits}\endinput}{}
3886 \si@ifloaded{sistyle}
3887 {\si@emclash{unitsdef}{sistyle}\endinput}{}

```

Emulation of units is needed for unitsdef to work.

```
3888 \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.

```
3889 \RequirePackage{textcomp}
3890 \RequirePackage[T1]{fontenc}

```

The multitude of package options for `unitsdef` need to be handled.

```
3891 \sisetup{mode=text,allowoptarg,prespace}
3892 \ifsi@old@noospace
3893   \sisetup{xspace=false}
3894 \fi

```

The various options for loading unit abbreviations have to be handled. Here, any request to avoid abbreviations prevents any loading.

```
3895 \ifsi@old@noabbr
3896   \sisetup{noload=abbr}
3897 \fi
3898 \ifsi@old@nofrequencyabbr
3899   \sisetup{noload=abbr}
3900 \fi
3901 \ifsi@old@nomolabbr
3902   \sisetup{noload=abbr}
3903 \fi
3904 \ifsi@old@novoltageabbr
3905   \sisetup{noload=abbr}
3906 \fi
3907 \ifsi@old@novolumeabbr
3908   \sisetup{noload=abbr}
3909 \fi
3910 \ifsi@old@noweightabbr
3911   \sisetup{noload=abbr}
3912 \fi
3913 \ifsi@old@noenergyabbr
3914   \sisetup{noload=abbr}
3915 \fi
3916 \ifsi@old@nolengthabbr
3917   \sisetup{noload=abbr}
3918 \fi

```

```

3919 \ifsi@old@notimeabbr
3920 \sisetup{noload=abbr}
3921 \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.

```

3922 \newcommand*{\unitvaluesep}{\,}
3923 \renewcommand*{\si@valuesep}{\text{\unitvaluesep}}
3924 \define@choicekey*+[si]{key}{valuesep}[\si@tempa]
3925 {space,thin,med,medium,thick,none}
3926 {\renewcommand*\unitvaluesep\@nameuse{si@fix@##1}}
3927 \si@log@debug{Option valuesep set to ##1}
3928 {\si@log@debug{Option valuesep set to ##1}}
3929 \renewcommand*\unitvaluesep{##1}

```

`\unitsignonly` Some rather straight-forward definitions, with just a bit of fun to get the spacing correct.

```

\ilu
\arc3930 \let\unitsignonly\si
3931 \si@newrobustcmd*{\ilu}[2][]{%
3932   \begingroup
3933     #1\unitvaluesep%
3934     \unit{#2}%
3935   \endgroup}
3936 \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`.

```

3937 \newcommand*{\unitSIdef}{\upshape}
3938 \newcommand*{\si@unitSIdef}{\unitSIdef\selectfont}
3939 \sisetup{textrm=si@unitSIdef}

```

`\per` Rather awkwardly, `unitsdef` uses `\per` in a different way to `siunitx`.

```

3940 \let\per\relax
3941 \si@newrobustcmd*{\per}[2]{%
3942   \begingroup
3943     \si@xspacefalse
3944     \renewcommand*{\unitvaluesep}{}%
3945     \unitfrac{#1}{#2}%
3946   \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`

```

\unitsuperscript3947 \newcommand*{\unittimes}{\ensuremath{\cdot}}
3948 \newcommand*{\unitsep}{\,}
3949 \renewcommand*{\si@unt@unithook}{\si@unt@litouttrue}
3950 \sisetup{unitsep=none}
3951 \newcommand*{\unitsuperscript}{\tothe}

```

`\newnosepunit` Simple aliases.

```

\renewnosepunit3952 \newcommand*{\newnosepunit}{\newunit[valuesep=none]}
3953 \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.

```

\setTextmu3954 \newcommand*{\setTextOmega}[2]{%
\setMathmu3955 \renewcommand*{\si@textOmega}{%
\setTextCelsius3956 \begingroup
\setMathCelsius3957 \edef\si@tempa{\sfdefault}%
\setMathDegree3958 \ifx\f@family\si@tempa
\setTextDegree3959 \expandafter#2%
3960 \else
3961 \expandafter#1%
3962 \fi
3963 \endgroup}}
3964 \newcommand*{\setMathOmega}[1]{\sisetup{mathsOmega=#1}}
3965 \newcommand*{\setTextmu}[2]{%
3966 \renewcommand*{\si@textmu}{%
3967 \begingroup
3968 \edef\si@tempa{\sfdefault}%
3969 \ifx\f@family\si@tempa
3970 \expandafter#2%
3971 \else
3972 \expandafter#1%
3973 \fi
3974 \endgroup}}
3975 \newcommand*{\setMathmu}[1]{\sisetup{mathsmu=#1}}
3976 \newcommand*{\setTextCelsius}[2]{%
3977 \renewcommand*{\si@textcelsius}{%
3978 \begingroup
3979 \edef\si@tempa{\sfdefault}%
3980 \ifx\f@family\si@tempa
3981 \expandafter#2%
3982 \else
3983 \expandafter#1%
3984 \fi
3985 \endgroup}}
3986 \newcommand*{\setMathCelsius}[1]{\sisetup{mathscelsius=#1}}
3987 \newcommand*{\setMathDegree}[2]{%
3988 \renewcommand*{\si@textdegree}{%
3989 \begingroup%
3990 \edef\si@tempa{\sfdefault}%
3991 \ifx\f@family\si@tempa
3992 \expandafter#2%
3993 \else
3994 \expandafter#1%
3995 \fi
3996 \endgroup}}
3997 \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.

```

3998 \ifsi@old@OHM
3999 \ifsi@old@ohm
4000 \si@log@inf{Both 'ohm' and 'OHM' options given\MessageBreak
4001 Using default behaviour for unitsdef}
4002 \expandafter\expandafter\expandafter\si@old@OHMfalse

```

```

4003 \fi
4004 \fi

\liter Tonne is spelled as “ton” by unitsdef, which is wrong in the UK at least (1 ton =
\ton 40 cwt = 2240 lb!).
\days4005 \ifsi@old@liter
4006 \ifsi@old@LITER
4007 \si@log@inf{Both ‘liter’ and ‘LITER’ options
4008 given\MessageBreak Using default behaviour for unitsdef}
4009 \else
4010 \renewunit{\liter}{l}
4011 \fi
4012 \fi
4013 \newunit{\ton}{t}
4014 \newunit{\days}{d}

\picometer Extra distances.
\nanometer4015 \newunit{\picometer}{\pico\meter}
\micrometer4016 \newunit{\nanometer}{\nano\meter}
\millimeter4017 \newunit{\micrometer}{\micro\meter}
\centimeter4018 \newunit{\millimeter}{\milli\meter}
\decimeter4019 \newunit{\centimeter}{\centi\meter}
\kilometer4020 \newunit{\decimeter}{\deci\meter}
4021 \newunit{\kilometer}{\kilo\meter}

\femtoliter Volumes with US spellings.
\picoliter4022 \newunit{\femtoliter}{\femto\liter}
\nanoliter4023 \newunit{\picoliter}{\pico\liter}
\microliter4024 \newunit{\nanoliter}{\nano\liter}
\milliliter4025 \newunit{\microliter}{\micro\liter}
\centiliter4026 \newunit{\milliliter}{\milli\liter}
\deciliter4027 \newunit{\centiliter}{\centi\liter}
4028 \newunit{\deciliter}{\deci\liter}
\hectoliter4029 \newunit{\hectoliter}{\hecto\liter}
\cubicmeter4030 \newunit{\cubicmeter}{\meter\cubed}
\cubicmicrometer4031 \newunit{\cubicmicrometer}{\micro\meter\cubed}
\cubicmillimeter4032 \newunit{\cubicmillimeter}{\milli\meter\cubed}

\squaremeter Areas, including the mis-spellings for \are and \hectare.
\squarecentimeter4033 \newunit{\squaremeter}{\Square\meter}
\squarekilometer4034 \newunit{\squarecentimeter}{\Square\centi\meter}
\ar4035 \newunit{\squarekilometer}{\Square\kilo\meter}
\hectar4036 \newunit{\ar}{a}
4037 \newunit{\hectar}{\hecto\ar}

\kv The code for unitsdef has the capitalisation wrong for \kV and \mV.
\mv4038 \ifsi@old@noabbr
4039 \else
4040 \ifsi@old@novoltageabbr\else
4041 \newunit{\kv}{\kilo\volt}
4042 \newunit{\mv}{\milli\volt}
4043 \fi
4044 \fi

```

```

\sek  There are some slightly different abbreviations, plus some which are not officially
\fg   allowed.
\fl4045 \ifsi@old@noabbr\else
\pl4046 \ifsi@old@notimeabbr\else
\nl4047 \newunit{\sek}{\second}
\micl4048 \fi
\ml4049 \ifsi@old@noweightabbr\else
\cl4050 \newunit{\fg}{\femto\gram}
\dl4051 \fi
\hl4052 \ifsi@old@novolumeabbr\else
4053 \newunit{\fl}{\femto\liter}
4054 \newunit{\pl}{\pico\liter}
4055 \newunit{\nl}{\nano\liter}
4056 \newunit{\micl}{\micro\liter}
4057 \newunit{\ml}{\milli\liter}
4058 \newunit{\cl}{\centi\liter}
4059 \newunit{\dl}{\deci\liter}
4060 \newunit{\hl}{\hecto\liter}
4061 \fi
4062 \fi

```

`\calory` `unitsdef` spells calorie incorrectly, and it is also not an SI unit.

```

\kilocalory4063 \newunit{\calory}{cal}
4064 \newunit{\kilocalory}{\kilo\calory}

```

`\uBar` `unitsdef` uses `\ubar` for bar.

```

4065 \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
`\gensymbmicro` character needs to be a letter. To avoid needing to worry about this again, a
`\gensymbdegree` second switch is set up.

```

4066 \catcode'\-=11\relax
4067 \ifsi@old@redef-gensymb
4068 \expandafter\si@gensymbtrue
4069 \fi
4070 \catcode'\-=12\relax
4071 \ifsi@gensymb
4072 \RequirePackage{gensymb}
4073 \AtBeginDocument{
4074 \let\gensymbohm\ohm
4075 \let\gensymbcelsius\celsius
4076 \let\gensymbmicro\micro
4077 \let\gensymbdegree\degree
4078 \let\ohm\@undefined
4079 \let\celsius\@undefined
4080 \let\micro\@undefined
4081 \let\degree\@undefined
4082 \ifsi@old@OHM\else
4083 \newunit{\ohm}{\si@sym@Omega}
4084 \newunit{\celsius}{\si@sym@celsius}
4085 \newprefix{\micro}{\si@sym@mu}{-6}
4086 \newunit{\degree}{\si@sym@degree}

```

```

4087 \fi}
4088 \fi

```

The configuration files can now be loaded.

```

4089 \requiresiconfigs{prefix,named,addn,accepted}

```

The `noconfig` option could be ignored, but it costs little to let it be used.

```

4090 \ifsi@old@noconfig\else
4091 \InputIfFileExists{unitsdef.cfg}
4092 {\si@log@inf{unitsdef config file loaded}}
4093 {\si@log@inf{unitsdef config file not found}}
4094 \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.

```

4095 \ProvidesFile{si-sstyle.cfg}
4096 [\csname si@svn@version\endcsname siunitx: Emulation of
4097 Sstyle]
4098 \si@emulating{sstyle}{2006/12/20 v2.3}
4099 \sisetup{%
4100 sepfour=true,
4101 obeyfamily,
4102 obeyitalic=true,
4103 numsign=+-,
4104 strictarc=false,
4105 unitsep=cdot}

```

`\SIobeyboldtrue` Some simple switches, but not using `\newif`.

```

\SIobeyboldfalse4106 \newcommand*{\SIobeyboldtrue}{\sisetup{obeybold=true}}
4107 \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@numstar4108 \let\num\relax
4109 \si@newrobustcmd*{\num}{%
4110 \@ifstar
4111 {\si@sis@numstar}
4112 {\si@sis@num}}
4113 \newcommand*{\si@sis@num}[2][{}]{%
4114 \begingroup%
4115 \sisetup{#1}%
4116 \expandafter\si@out@num\expandafter{\si@num{#2}}%
4117 \endgroup}
4118 \newcommand*{\si@sis@numstar}[2][{}]{%
4119 \begingroup%
4120 \sisetup{mode=text,obeybold}%
4121 \sisetup{#1}%
4122 \expandafter\si@out@num\expandafter{\si@num{#2}}%
4123 \endgroup}

```

`\pnt` The `\pnt` macro is needed as `.` is active inside `\SI`. The name is exactly the same as in `Sistyle`, 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.

```
4124 \newcommand*{\pnt}{\ensuremath{\si@decimalsymbol}}
```

`\SIgroupfourtrue` Switches for grouping four characters.

```
\SIgroupfourfalse4125 \newcommand*{\SIgroupfourtrue}{\sisetup{sepfour=true}}
4126 \newcommand*{\SIgroupfourfalse}{\sisetup{sepfour=false}}
```

`\SIunitsep` Whatever is given here is passed through to `\sisetup`.

```
\SIunitsep4127 \newcommand*{\SIunitsep}[1]{\sisetup{valuesep={#1}}}
\SIunitdot4128 \newcommand*{\SIunitdot}[1]{\sisetup{unitsep={#1}}}
4129 \newcommand*{\SIunitdot}[1]{\sisetup{unitsep={#1}}}
```

`\SIdecimalsymbol` The same is true here, with the appropriate translation.

```
\SIthousandsep4130 \newcommand*{\SIdecimalsymbol}[1]{\sisetup{decimalsymbol={#1}}}
\SIproductsign4131 \newcommand*{\SIthousandsep}[1]{\sisetup{digitsep={#1}}}
\SIdecimalsign4132 \newcommand*{\SIproductsign}[1]{\sisetup{expproduct={#1}}}
4133 \newcommand*{\SIdecimalsign}[1]{\sisetup{decimalsymbol={#1}}}
```

`\si@sis@savefont` The font definitions need a bit of extra work doing. As both settings here have `@` as a letter, all should be fine.

```
\si@sis@savefont{<setting>}{<argument>}
4134 \newcommand*{\si@sis@savefont}[2]{%
4135   \@namedef{si@sis@#1}{#2}%
4136   \sisetup{#1=si@sis@#1}}
```

`\SIMathrm` The font control macros have to ensure that a macro name is passed to `\sisetup`.

```
\SIMathsf4137 \newcommand*{\SIMathrm}[1]{\si@sis@savefont{mathrm}{#1}}
\SIMathtt4138 \newcommand*{\SIMathsf}[1]{\si@sis@savefont{mathsf}{#1}}
4139 \newcommand*{\SIMathtt}[1]{\si@sis@savefont{mathtt}{#1}}
```

`\SIdefaultMfam` The same for the default keys.

```
\SIdefaultNfam4140 \newcommand*{\SIdefaultMfam}[1]{\si@sis@savefont{mathrm}{#1}}
\SIdefaultTfam4141 \newcommand*{\SIdefaultNfam}[1]{\si@sis@savefont{mathnumrm}{#1}}
4142 \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.

```
4143 \si@newrobustcmd*{\ensureupmath}[1]{%
4144   \begingroup
4145     \sisetup{mode=maths, obeyitalic=false}%
4146     \si@out{#1}%
4147   \endgroup}
```

`\degC` A few extra symbol names are needed.

```
\degF4148 \newcommand*{\degC}{\si@sym@celsius}
\arcdeg4149 \newcommand*{\arcdeg}{\si@sym@degree}
4150 \newcommand*{\degF}{\si@sym@degree F}
```

`\AddToSIstyle` Finally, the locale control.

```

\SIstyle4151 \newcommand*{\SIstyle}[1]{\sisetup{locale=#1}}
\SIstyleToLang4152 \newcommand*{\SIstyleToLang}[2]{\sisetup{loctolang=#1:#2}}
\si@sis@addtolocale4153 \newcommand*{\AddToSIstyle}{%
4154   \si@switchfalse
4155   \@ifstar
4156     {\si@switchtrue
4157       \si@sis@addtolocale}
4158     {\si@sis@addtolocale}}
4159 \newcommand*{\si@sis@addtolocale}[2]{%
4160   \ifsi@switch
4161     \expandafter\let\csname si@loc@#1@extra\endcsname\relax
4162   \fi
4163   \addtolocale{#1}{#2}}

```

25.4 Slunits

Slunits emulation starts in much the same way.

```

4164 \ProvidesFile{si-SIunits.cfg}
4165 [\csname si@svn@version\endcsname siunitx: Emulation of
4166   SIunits]
4167 \si@emulating{SIunits}{2007/12/02 v1.36}
4168 \sisetup{
4169   unitsep=thick,
4170   valuesep=thick,
4171   prefixproduct=\si@valuesep,
4172   trapambigfrac=false,
4173   stickyper}
4174 \requiresiconfigs{prefix,named,accepted,physical}

```

`\reciprocal` A few very simple translations, using the internal version of `\per` to allow changes of output style.

```

\per4175 \newcommand*{\reciprocal}{\sisetup{per=reciprocal}\si@per}
\usk4176 \let\rp\reciprocal
\power4177 \renewcommand*{\per}{\sisetup{per=slash}\si@per}
\rpsquare4178 \newcommand*{\usk}{}
\rpcubic4179 \newcommand*{\power}[1]{#1\tothe}
\fourth4180 \newcommand*{\rpsquare}{\sisetup{per=reciprocal}\si@per\Square}
\rpfourth4181 \newcommand*{\rpcubic}{\sisetup{per=reciprocal}\si@per\cubic}
4182 \newpower{\fourth}{4}
4183 \newcommand*{\rpfourth}{\sisetup{per=reciprocal}\si@per\fourth}

```

`\rpsquared` Here, some low-level switch changing is needed.

```

\rpcubed4184 \newcommand*{\rpsquared}{%
4185   \sisetup{per=reciprocal}\si@unt@pertrue\si@unt@perseenttrue%
4186   \squared}
4187 \newcommand*{\rpcubed}{%
4188   \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
4189 \newcommand*{\SIsetup}[1]{%
4190   \@for\si@tempa:=#1\do{%

```



```

4191 \ifundefined{ifsi@old@#1}
4192 {\si@log@warn{Unknown SIunits option `#1'}}}
4193 {\csname si@old@#1true\endcsname}}
4194 \si@siu@setup}
4195 \newcommand*{\si@siu@setup}{%
4196 \ifsi@old@cdot
4197 \sisetup{unitsep=cdot}%
4198 \fi
4199 \ifsi@old@thickspace
4200 \sisetup{unitsep=thick}%
4201 \fi
4202 \ifsi@old@mediumspace
4203 \sisetup{unitsep=medium}%
4204 \fi
4205 \ifsi@old@thinspace
4206 \sisetup{unitsep=thin}%
4207 \fi
4208 \ifsi@old@thickqspace
4209 \sisetup{valuesep=thick}%
4210 \fi
4211 \ifsi@old@mediumqspace
4212 \sisetup{valuesep=medium}%
4213 \fi
4214 \ifsi@old@thingspace
4215 \sisetup{valuesep=thin}%
4216 \fi}
4217 \si@siu@setup

```

`\square` **SIunits** does slightly different things about the clash with `\square`, and either
`\square` **redefines this macro or provides `\square`.**

```

4218 \ifsi@old@squaren
4219 \newpower{\square}{2}
4220 \fi
4221 \AtBeginDocument{%
4222 \ifundefined{square}
4223 {\newpower{\square}{2}}
4224 {\ifsi@old@amssymb
4225 \renewpower{\square}{2}
4226 \else
4227 \ifsi@old@squaren\else
4228 \si@log@warn{\string\square\space already
4229 defined\MessageBreak SIunits mode may cause
4230 errors}%
4231 \fi
4232 \fi}}

```

`\gray` The potential clash with **PStricks** is also handled differently; here, `\Gray` will
 already be defined by the **siunitx** kernel.

```

4233 \AtBeginDocument{
4234 \ifundefined{gray}
4235 {\newunit{\gray}{Gy}}
4236 {\ifsi@old@pstricks
4237 \renewunit{\gray}{Gy}
4238 \else

```

```

4239         \ifsi@old@Gray\else
4240         \si@log@warn{\string\gray\space already
4241             defined\MessageBreak SIunits mode may cause
4242             errors}%
4243         \fi
4244     \fi}}

```

\unit The **\unit** macro is defined.

```

\unita4245 \ifsi@old@italian
4246     \let\unita\SI
4247 \else
4248     \let\unit\SI
4249 \fi

```

The miscellaneous options are moped up.

```

4250 \ifsi@old@textstyle
4251     \sisetup{mode=text}
4252 \fi
4253 \ifsi@old@binary
4254     \sisetup{also load= binary}
4255 \fi
4256 \ifsi@old@noams
4257     \AtBeginDocument{%
4258         \renewcommand*{\si@textmu}{\ensuremath\si@mathsmu}}
4259 \fi

```

\arcminute The unit macros defined by Slunits that are not defined by siunitx (by default).

```

\arcsecond4260 \newunit[valuesep=none]{\arcminute}{\si@sym@minute}
\rperminute4261 \newunit[valuesep=none]{\arcsecond}{\si@sym@second}
\ton4262 \newunit{\rperminute}{r/min}
\degreecelsius4263 \newunit{\ton}{t}
4264 \newunit{\degreecelsius}{\celsius}

```

\addunit This is an alias for **\newunit**.

```

4265 \let\addunit\newunit

```

\addprefix A little more work for **\addprefix**.

```

4266 \newcommand*{\addprefix}[2]{\newprefix{#1}{#2}}

```

\si@siu@newunit To save some code, making new units which need a . . . np variant is handled by a dedicated macro.

```

\si@siu@power \si@siu@addunit[\langle power \rangle]{\langle pre \rangle}{\langle post \rangle}
\si@siu@newunithook \si@siu@addunit[\langle power \rangle]{\langle pre \rangle}{\langle post \rangle}
4267 \newcommand*{\si@siu@newunit}[3][[]]{%

```

A test is needed to sort out **\square**.

```

4268 \renewcommand*{\si@tempa}{#1}%
4269 \renewcommand*{\si@tempb}{square}%
4270 \renewcommand*{\si@siu@power}{}%
4271 \ifx\@empty\si@tempa\@empty\else
4272     \ifx\si@tempa\si@tempb
4273         \renewcommand*{\si@siu@power}{\ssquare}%
4274     \else
4275         \edef\si@siu@power{%

```

```

4276         \expandafter\noexpand\csname #1\endcsname}%
4277     \fi
4278 \fi

```

The necessary information is now stored in temporary macros.

```

4279 \edef\si@tempa{%
4280     \expandafter\noexpand\csname #2per#1#3\endcsname}%
4281 \edef\si@tempb{%
4282     \expandafter\noexpand\csname #2\endcsname\noexpand\per
4283     \expandafter\noexpand\si@siu@power
4284     \expandafter\noexpand\csname #3\endcsname}%
4285 \expandafter\expandafter\expandafter\newunit\expandafter%
4286     \expandafter\expandafter\expandafter\si@tempa\expandafter}%
4287     \expandafter\si@tempb}%
4288 \edef\si@tempa{%
4289     \expandafter\noexpand\csname #2per#1#3np\endcsname}%
4290 \edef\si@tempb{%
4291     \expandafter\noexpand\csname #2\endcsname\noexpand
4292     \reciprocal\expandafter\noexpand\si@siu@power
4293     \expandafter\noexpand\csname #3\endcsname}%
4294 \expandafter\expandafter\expandafter\newunit\expandafter
4295     \expandafter\expandafter\expandafter\si@tempa\expandafter}%
4296     \expandafter\si@tempb}%
4297 \si@siu@newunithook[#1]{#2}{#3}}
4298 \providecommand*\si@siu@newunithook}[3][{}{}

```

The basic units are now defined; these only have a single csname in each of the numerator and denominator.

```

4299 \si@siu@newunit{gray}{second}
4300 \si@siu@newunit[square]{metre}{second}
4301 \si@siu@newunit{joule}{mole}
4302 \si@siu@newunit[cubic]{mole}{metre}
4303 \si@siu@newunit[square]{radian}{second}
4304 \si@siu@newunit{radian}{second}
4305 \si@siu@newunit[cubic]{squaremetre}{metre}
4306 \si@siu@newunit[cubic]{katal}{metre}
4307 \si@siu@newunit{coulomb}{mol}
4308 \si@siu@newunit[square]{ampere}{metre}
4309 \si@siu@newunit[cubic]{kilogram}{metre}
4310 \si@siu@newunit[cubic]{coulomb}{metre}
4311 \si@siu@newunit{volt}{metre}
4312 \si@siu@newunit[square]{coulomb}{squaremetre}
4313 \si@siu@newunit{farad}{metre}
4314 \si@siu@newunit[square]{watt}{metre}
4315 \si@siu@newunit[square]{joule}{metre}
4316 \si@siu@newunit[cubic]{newton}{metre}
4317 \si@siu@newunit{newton}{kilogram}
4318 \si@siu@newunit{joule}{kelvin}
4319 \si@siu@newunit{joule}{kilogram}
4320 \si@siu@newunit{coulomb}{kilogram}
4321 \si@siu@newunit{squaremetre}{second}
4322 \si@siu@newunit[square]{squaremetre}{second}
4323 \si@siu@newunit[square]{candela}{metre}
4324 \si@siu@newunit{ampere}{metre}

```

```

4325 \si@siu@newunit{joule}{tesla}
4326 \si@siu@newunit{henry}{metre}
4327 \si@siu@newunit{kilogram}{second}
4328 \si@siu@newunit[square]{kilogram}{metre}
4329 \si@siu@newunit{kilogram}{metre}
4330 \si@siu@newunit[square]{newton}{metre}
4331 \si@siu@newunit{watt}{kilogram}
4332 \si@siu@newunit[cubic]{watt}{metre}
4333 \si@siu@newunit{squaremetre}{kilogram}
4334 \si@siu@newunit{cubicmetre}{kilogram}
4335 \si@siu@newunit{newton}{metre}
4336 \si@siu@newunit[cubic]{squaremetre}{second}
4337 \si@siu@newunit{metre}{second}
4338 \si@siu@newunit[cubic]{joule}{metre}
4339 \si@siu@newunit{cubicmetre}{second}

```

\si@siu@newunitx For the more complex units, a slightly different approach is used; four arguments
\si@siu@newunitxhook are required, and have to cover everything.

```

4340 \newcommand*{\si@siu@newunitx}[4]{%
4341   \expandafter\newunit\expandafter{\csname #1per#2\endcsname}
4342   {#3\per#4}%
4343   \expandafter\newunit\expandafter{\csname #1per#2np\endcsname}
4344   {#3\reciprocal#4}
4345   \si@siu@newunitxhook{#1}{#2}{#3}{#4}}
4346 \providecommand*{\si@siu@newunitxhook}[4]{}

```

The units are defined.

```

4347 \si@siu@newunitx{kilogramsquaremetre}{second}
4348   {\kilogram\squaremetre}{\second}
4349 \si@siu@newunitx{squaremetre}{newtonsecond}{\squaremetre}
4350   {\newton\second}
4351 \si@siu@newunitx{kilogrammetre}{second}{\kilogram\metre}
4352   {\second}
4353 \si@siu@newunitx{kilogram}{squaremetresecond}{\kilogram}
4354   {\squaremetre\second}
4355 \si@siu@newunitx{joule}{molekelvin}{\joule}{\mole\kelvin}
4356 \si@siu@newunitx{kilogram}{kilomole}{\kilogram}{\kilo\mole}
4357 \si@siu@newunitx{kilogrammetre}{squaresecond}{\kilogram\metre}
4358   {\second\squared}
4359 \si@siu@newunitx{watt}{squaremetresteradian}{\watt}
4360   {\squaremetre\steradian}
4361 \si@siu@newunitx{joule}{kilogramkelvin}{\joule}
4362   {\kilogram\kelvin}
4363 \si@siu@newunitx{watt}{metrekelvin}{\watt}{\metre\kelvin}
4364 \si@siu@newunitx{kilogram}{cubicmetrecoulomb}{\kilogram}
4365   {\cubic\metre\coulomb}
4366 \si@siu@newunitx{kilogram}{secondcubicmetre}{\kilogram}
4367   {\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.

```

4368 \newunit{\si@siu@unity}{1}
4369 \si@siu@newunitx{}{squaremetresecond}{\si@siu@unity}
4370   {\squaremetre\second}

```

A few compound units that are best defined directly.

```

4371 \newunit{\pascalsecond}{\pascal\second}
4372 \newunit{\amperemetresecond}{\ampere\metre\second}
4373 \newunit{\ohmmetre}{\ohm\metre}
4374 \newunit{\newtonmetre}{\newton\metre}
4375 \let\newtonmetrenp\newtonmetre
4376 \newunit{\kilogramsquaremetre}{\kilogram\squaremetre}
4377 \let\kilogramsquaremetrenp\kilogramsquaremetre

```

\si@siu@newprefix To generate the prefixes correctly, a small saving in repetition.

```

\si@siu@newprefix{\prefix}
4378 \newcommand*{\si@siu@newprefix}[1]{%
4379   \edef\si@tempa{\expandafter\noexpand\csname #1d\endcsname}%
4380   \edef\si@tempb{\expandafter\noexpand\csname #1\endcsname}%
4381   \expandafter\expandafter\expandafter\newcommand\expandafter
4382     \expandafter\expandafter*\expandafter\expandafter
4383     \expandafter{\expandafter\si@tempa\expandafter}\expandafter
4384     {\expandafter\si@prefixsymbolicfalse\si@tempb}}

```

This is now implemented.

```

4385 \si@siu@newprefix{yocto}
4386 \si@siu@newprefix{zepto}
4387 \si@siu@newprefix{atto}
4388 \si@siu@newprefix{femto}
4389 \si@siu@newprefix{pico}
4390 \si@siu@newprefix{nano}
4391 \si@siu@newprefix{micro}
4392 \si@siu@newprefix{milli}
4393 \si@siu@newprefix{centi}
4394 \si@siu@newprefix{deca}
4395 \si@siu@newprefix{deka}
4396 \si@siu@newprefix{hecto}
4397 \si@siu@newprefix{kilo}
4398 \si@siu@newprefix{mega}
4399 \si@siu@newprefix{giga}
4400 \si@siu@newprefix{tera}
4401 \si@siu@newprefix{peta}
4402 \si@siu@newprefix{exa}
4403 \si@siu@newprefix{zetta}
4404 \si@siu@newprefix{yotta}
4405 \ifsi@old@binary
4406   \si@siu@newprefix{kibi}
4407   \si@siu@newprefix{mebi}
4408   \si@siu@newprefix{gibi}
4409   \si@siu@newprefix{tebi}
4410   \si@siu@newprefix{pebi}
4411   \si@siu@newprefix{exbi}
4412 \fi

```

\derradian The derived units may need to be defined.

```

\dersteradian4413 \ifsi@old@derived
  \derhertz4414 \newunit{\derradian}{\metre\reciprocal\metre}
  \dernewton4415 \newunit{\dersteradian}{\squaremetre\rpsquare\metre}
  \derpascal
  \derjoule
  \derwatt
  \dercoulomb
  \dervolt
  \derfarad
  \derohm

```

```

4416 \newunit{\derhertz}{\reciprocal\second}
4417 \newunit{\dernewton}{\metre\kilogram\second\rpsquared}
4418 \newunit{\derpascal}{\newton\rpsquare\metre}
4419 \newunit{\derjoule}{\newton\metre}
4420 \newunit{\derwatt}{\joule\reciprocal\second}
4421 \newunit{\dercoulomb}{\ampere\second}
4422 \newunit{\dervolt}{\watt\reciprocal\ampere}
4423 \newunit{\derfarad}{\coulomb\reciprocal\volt}
4424 \newunit{\derohm}{\volt\reciprocal\ampere}

\dersiemens In two blocks!
\derweber4425 \newunit{\dersiemens}{\ampere\reciprocal\volt}
\deresla4426 \newunit{\derweber}
\derhenry4427 {\squaremetre\kilogram\second\rpsquared\reciprocal\ampere}
\dercelsius4428 \newunit{\deresla}{\weber\rpsquare\metre}
\derlumen4429 \newunit{\derhenry}{\weber\reciprocal\ampere}
\derlux4430 \newunit{\dercelsius}{\kelvin}
\derbecquerel4431 \newunit{\derlumen}{\candela\steradian}
4432 \newunit{\derlux}{\lumen\rpsquare\metre}
\dergray4433 \newunit{\derbecquerel}{\derhertz}
\dersievert4434 \newunit{\dergray}{\joule\reciprocal\kilogram}
\derkatal4435 \newunit{\dersievert}{\dergray}
4436 \newunit{\derkatal}{\rp\second\usk\mole}
4437 \fi

\radianbase Also the “derived-in-base”.
\steradianbase4438 \ifsi@old@derivedinbase
\hertzbase4439 \newunit{\radianbase}{\metre\reciprocal\metre}
\newtonbase4440 \newunit{\steradianbase}{\squaremetre\rpsquare\metre}
\pascalbase4441 \newunit{\hertzbase}{\reciprocal\second}
\joulebase4442 \newunit{\newtonbase}{\metre\kilogram\second\rpsquared}
\wattbase4443 \newunit{\pascalbase}{\reciprocal\metre\kilogram\second%
\squared}
\coulombbase4444 \newunit{\joulebase}{\squaremetre\kilogram\second\rpsquared}
\voltbase4445 \newunit{\wattbase}{\squaremetre\kilogram\rpcubic\second}
\faradbase4446 \newunit{\coulombbase}{\ampere\second}
\ohmbase4447 \newunit{\voltbase}
4448 {\squaremetre\kilogram\rpcubic\second\reciprocal\ampere}
4449 \newunit{\faradbase}
4450 {\rpsquare\metre\reciprocal\kilogram\fourth\second\ampere%
\squared}
4451 \newunit{\ohmbase}
4452 {\squaremetre\kilogram\rpcubic\second\rpsquare\ampere}
4453
4454

\siemensbase Also in two blocks.
\weberbase4455 \newunit{\siemensbase}
\teslabase4456 {\rpsquare\metre\reciprocal\kilogram\cubic\second\ampere%
\squared}
\henrybase4457 \newunit{\weberbase}
\celsiusbase4458 {\squaremetre\kilogram\second\rpsquared\reciprocal\ampere}
\lumenbase4459 \newunit{\teslabase}{\kilogram\second\rpsquared\reciprocal%
\ampere}
\luxbase4460 \newunit{\henrybase}
\becquerelbase4461 {\squaremetre\kilogram\second\rpsquared\rpsquare\ampere}
\graybase4462
\sievertbase4463
\katalbase

```

```

4464 \newunit{\celsiusbase}{\kelvin}
4465 \newunit{\lumenbase}{\candela\squaremetre\rpsquare\metre}
4466 \newunit{\luxbase}{\candela\squaremetre\rfourth\metre}
4467 \newunit{\becquerelbase}{\hertzbase}
4468 \newunit{\graybase}{\squaremetre\second\rpsquared}
4469 \newunit{\sievertbase}{\graybase}
4470 \newunit{\katalbase}{\rp\second\mole}
4471 \fi

```

Any configuration file is used if found.

```

4472 \InputIfFileExists{SIunits.cfg}
4473 {\si@log@inf{SIunits config file loaded}}
4474 {\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.

```

4475 \ProvidesFile{si-hepunits.cfg}
4476 [\csname si@svn@version\endcsname Emulation of siunitx:
4477   hepunits]
4478 \si@emulating{hepunits}{2007/09/27}
4479 \requiresiconfigs{SIunits,accepted,prefix,hep}

```

`\invbarn` Inverses barn units.

```

\invnanobarn4480 \ifsi@old@noprefixcmds\else
\invpicobarn4481 \newunit{\invbarn}{\per\barn}
\invfemtobarn4482 \newunit{\invnanobarn}{\per\nano\barn}
\invattobarn4483 \newunit{\invpicobarn}{\per\pico\barn}
\invzeptobarn4484 \newunit{\invfemtobarn}{\per\femto\barn}
\invyoctobarn4485 \newunit{\invattobarn}{\per\atto\barn}
4486 \newunit{\invzeptobarn}{\per\zepto\barn}
4487 \newunit{\invyoctobarn}{\per\yocto\barn}

```

`\invnb` Also available abbreviated.

```

\invpb4488 \newunit{\invnb}{\per\nano\barn}
\invfb4489 \newunit{\invpb}{\per\pico\barn}
\invab4490 \newunit{\invfb}{\per\femto\barn}
\invzb4491 \newunit{\invab}{\per\atto\barn}
\invyb4492 \newunit{\invzb}{\per\zepto\barn}
4493 \newunit{\invyb}{\per\yocto\barn}
4494 \fi

```

`\invcmsqpersecond` Luminosity.

```

\invcmsqpersec4495 \newunit{\invcmsqpersecond}{\per\Square\centi\metre\per\second}
\lumiunits4496 \newunit{\invcmsqpersec}{\per\Square\centi\metre\per\second}
4497 \newunit{\lumiunits}{\per\Square\centi\metre\per\second}

```

`\inveV` The inverse of an electron-volt, plus prefixes.

```

\minveV4498 \newunit{\inveV}{\per\electronvolt}
\minveV4499 \newunit{\minveV}{\per\milli\electronvolt}
\kinveV4500 \newunit{\kinveV}{\per\kilo\electronvolt}
\MinveV4501 \newunit{\MinveV}{\per\mega\electronvolt}
\GinveV
\TinveV

```

```

4502 \newunit{\GinveV}{\per\giga\electronvolt}
4503 \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.

4504 \newunit[per=slash,eVcorrb=0.6ex]{\evoverc}
4505 {\electronvolt\per\clight}
4506 \newunit[per=slash,eVcorrb=0.6ex]{\evovercsq}
4507 {\electronvolt\per\Square\clight}

\meVoverc Prefixed combinations, first of the speed of light.
\keVoverc4508 \newunit[per=slash,eVcorrb=0.6ex]{\meVoverc}
\MeVoverc4509 {\milli\electronvolt\per\clight}
\GeVoverc4510 \newunit[per=slash,eVcorrb=0.6ex]{\keVoverc}
\TeVoverc4511 {\kilo\electronvolt\per\clight}
4512 \newunit[per=slash,eVcorrb=0.6ex]{\MeVoverc}
4513 {\mega\electronvolt\per\clight}
4514 \newunit[per=slash,eVcorrb=0.6ex]{\GeVoverc}
4515 {\giga\electronvolt\per\clight}
4516 \newunit[per=slash,eVcorrb=0.6ex]{\TeVoverc}
4517 {\tera\electronvolt\per\clight}

\meVovercsq Then of the square.
\keVovercsq4518 \newunit[per=slash,eVcorrb=0.6ex]{\meVovercsq}
\MeVovercsq4519 {\milli\electronvolt\per\Square\clight}
\GeVovercsq4520 \newunit[per=slash,eVcorrb=0.6ex]{\keVovercsq}
\TeVovercsq4521 {\kilo\electronvolt\per\Square\clight}
4522 \newunit[per=slash,eVcorrb=0.6ex]{\MeVovercsq}
4523 {\mega\electronvolt\per\Square\clight}
4524 \newunit[per=slash,eVcorrb=0.6ex]{\GeVovercsq}
4525 {\giga\electronvolt\per\Square\clight}
4526 \newunit[per=slash,eVcorrb=0.6ex]{\TeVovercsq}
4527 {\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.

```

4528 \ProvidesFile{si-fancynum.cfg}
4529 [\csname si@svn@version\endcsname Emulation of siunitx:
4530   fancynum]
4531 \si@emulating{fancynum}{2000/08/08 0.92}
4532 \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.

4533 \let\fnm\num

```

\setfnumdsym The control macros are defined.

```

\setfnumgsym4534 \newcommand*{\setfnumdsym}[1]{\sisetup{decimalsymbol={#1}}}
\setfnummsym4535 \newcommand*{\setfnumgsym}[1]{\sisetup{digitsep={#1}}}
4536 \newcommand*{\setfnummsym}[1]{\sisetup{expproduct={#1}}}

```


The various package options are now processed if necessary.

```

4537 \ifsi@old@english
4538 \sisetup{decimalsymbol=cdot,digitsep=comma}
4539 \fi
4540 \ifsi@old@french
4541 \sisetup{decimalsymbol=comma,digitsep=fullstop}
4542 \fi
4543 \ifsi@old@tight
4544 \sisetup{expproduct=tighttimes}
4545 \fi
4546 \ifsi@old@loose
4547 \sisetup{expproduct=times}
4548 \fi
4549 \ifsi@old@thinspace
4550 \sisetup{digitsep=thin}
4551 \fi
4552 \ifsi@old@commas
4553 \sisetup{digitsep=comma}
4554 \fi
4555 \ifsi@old@plain
4556 \sisetup{digitsep=none}
4557 \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.

```

4558 \ProvidesFile{si-fancyunits.cfg}
4559 [\csname si@svn@version\endcsname Emulation of siunitx:
4560   fancyunits]
4561 \si@emulating{fancyunits}{2007/02/01 v1.0.1}
4562 \si@ifloaded{SIunits}
4563 {\si@log@err{SIunits emulation loaded\MessageBreak before
4564   fancyunits emulation}{You need to load the fancyunits
4565   emulation\MessageBreak code before that for
4566   SIunits.\MessageBreak Try emulate=fancyunits as the first
4567   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

```

4568 \newcommand*{\si@siu@newunithook}[3][{}]{%
4569   \edef\si@tempa{%
4570     \expandafter\noexpand\csname #2per#1#3uf\endcsname}%
4571   \renewcommand*{\si@tempb}{stickyper,per=fraction,
4572     fraction=nice}%
4573   \edef\si@tempc{%
4574     \noexpand\sisetup{\si@tempb}%
4575     \expandafter\noexpand\csname #2\endcsname\noexpand\si@per%
4576     \expandafter\noexpand\si@siu@power%
4577     \expandafter\noexpand\csname #3\endcsname}%
4578   \expandafter\expandafter\expandafter\newunit\expandafter
4579     \expandafter\expandafter{\expandafter\si@tempa\expandafter}%

```

```

4580 \expandafter{\si@tempc}%
4581 \edef\si@tempa{%
4582 \expandafter\noexpand\csname #2per#1#3Uf\endcsname}%
4583 \renewcommand*{\si@tempb}{stickyper,per=fraction,
4584 fraction=frac}%
4585 \edef\si@tempc{%
4586 \noexpand\sisetup{\si@tempb}%
4587 \noexpand\def\noexpand\si@frc@hook{\noexpand\textstyle}%
4588 \expandafter\noexpand\csname #2\endcsname\noexpand\si@per%
4589 \expandafter\noexpand\si@siu@power%
4590 \expandafter\noexpand\csname #3\endcsname}%
4591 \expandafter\expandafter\expandafter\newunit\expandafter
4592 \expandafter\expandafter{\expandafter\si@tempa\expandafter}%
4593 \expandafter{\si@tempc}%
4594 \edef\si@tempa{%
4595 \expandafter\noexpand\csname #2per#1#3UF\endcsname}%
4596 \edef\si@tempc{%
4597 \noexpand\sisetup{\si@tempb}%
4598 \noexpand\def\noexpand\si@frc@hook{\noexpand\displaystyle}%
4599 \expandafter\noexpand\csname #2\endcsname\noexpand\si@per%
4600 \expandafter\noexpand\si@siu@power%
4601 \expandafter\noexpand\csname #3\endcsname}%
4602 \expandafter\expandafter\expandafter\newunit\expandafter
4603 \expandafter\expandafter{\expandafter\si@tempa\expandafter}%
4604 \expandafter{\si@tempc}}
4605 \newcommand*{\si@siu@newunitxhook}[4]{%
4606 \expandafter\newunit\expandafter{\csname #1per#2uf\endcsname}
4607 {\sisetup{stickyper,per=fraction,fraction=nice}%
4608 #3\si@per#4}%
4609 \expandafter\newunit\expandafter{\csname #1per#2Uf\endcsname}
4610 {\sisetup{stickyper,per=fraction,fraction=frac}%
4611 \renewcommand*{\si@frc@hook}{\textstyle}%
4612 #3\si@per#4}%
4613 \expandafter\newunit\expandafter{\csname #1per#2UF\endcsname}
4614 {\sisetup{stickyper,per=fraction,fraction=frac}%
4615 \renewcommand*{\si@frc@hook}{\displaystyle}%
4616 #3\si@per#4}}

```

With that done, the emulation modules can be loaded.

```

4617 \requiresiconfigs{SIunits,addn,astro}
4618 \sisetup{obeyall}

```

There is one fancyunits-specific option to handle. The other options all get sent through to the Slunits system.⁵⁶

```

4619 \ifsi@old@spaceqspace
4620 \sisetup{valuesep=space}
4621 \fi

```

```

\pamminute fancyunits provides some extra units, plus tonne spelled incorrectly (again).
\parsecond4622 \newunit{\pamminute}{'}
\AstroE4623 \newunit{\parsecond}{''}
\oersted4624 \newunit{\AstroE}{AE}
\ton

```

⁵⁶As Slunits is rather more widely known than fancyunits, any other options which could be for either are assumed to be for Slunits.

```

4625 \newunit{\oersted}{OE}
4626 \provideunit{\ton}{t}

\decaD   An additional prefix.
4627 \let\decaD\decad

\frac    The fractional unit macros need to be reproduced.
\Ufrac4628 \newcommand*{\Ufrac}[2]{%
\Ufrac4629   \si[stickyper,per=fraction,fraction=nice]{#1\si@per#2}}
4630 \newcommand*{\Ufrac}[2]{%
4631   \ensuremath{\textstyle{%
4632     \si[stickyper,per=fraction,fraction=frac]{#1\si@per#2}}}}
4633 \newcommand*{\Ufrac}[2]{%
4634   \ensuremath{\displaystyle{%
4635     \si[stickyper,per=fraction,fraction=frac]{#1\si@per#2}}}}

\pow     A slightly-shortened named \power.
4636 \let\pow\power

\Squaremetre   An alias for \squaremetre.
4637 \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.
4638 \si@siu@newunit{Gray}{second}
4639 \si@siu@newunit[square]{Squaremetre}{metre}
4640 \si@siu@newunitx{Squaremetre}{newtonsecond}{\Square\metre}
4641   {\newton\second}
4642 \si@siu@newunit{Squaremetre}{second}
4643 \si@siu@newunit[square]{Squaremetre}{squaresecond}
4644 \si@siu@newunit{Squaremetre}{kilogram}
4645 \si@siu@newunit[cubic]{Squaremetre}{second}

```

Part V

Notes

26 Change History

1.0a		\si@tab@gettok@s: Fixed issues with alignment of s column con- tents 110
\si@fix@space: Fixed problem with space option 73		
\si@loc@ltol: babel support fixed for default document lan- guage 135	v1.0	General: First official release 1
\si@tab@end@S: Fixed mixed number/text column alignment 112	v1.ob	\si@num@rndpost: Corrected bug in rounding code 99

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	
\% 3751
\^ 22, 32, 1237, 3301, 3305
\' 20, 30
\~ 23, 33, 1236, 3171, 3177, 3210
A	
\ab 20 , <u>3785</u>
\addprefix <u>4266</u>
addsign (option) 25
\addtolocale 36 , <u>3298</u> , 4163
\AddToSistyle <u>4151</u>
\addunit <u>4265</u>
allowoptarg (option) 29
allowzeroexp (option) 25
alsoload (option) 31
\ampere	. 13 , <u>3214</u> , 3603–3607, 3664– 3668, 4372, 4421, 4422, 4424, 4425, 4427, 4429, 4447, 4449, 4451, 4454, 4456, 4459, 4461, 4463
\amu 16 , <u>3687</u>
\ang	... 10 , <u>2014</u> , 2038, 2039, 2043, 3936
angformat (option) 26
anglesep (option) 23
\angstrom 18 , <u>3716</u>
\ar <u>4033</u>
\arc <u>3930</u>
\arcdeg <u>4148</u>
\arcmin 15 , <u>3731</u>
\arcminute <u>4260</u>
\arcsec 15 , <u>3731</u>
\arcsecond <u>4260</u>
\are 18 , <u>3716</u>
\as 16 , <u>3709</u>
astroang (option) 26
\AstroE <u>4622</u>
\atomicmass 15 , 16 , 3693, <u>3752</u>
\atomicmassunit 15 , <u>3752</u>
\atto 14 , <u>3507</u> , 3597, 3710, 3782, 3788, 4485, 4491
\attobarn 20 , <u>3779</u>
\attosecond 16 , <u>3597</u>
B	
\BAR 18 , <u>3716</u>
\barn	.. 18 , <u>3716</u> , 3779–3790, 4481–4493
\bbar 18 , <u>3716</u>
\becquerel 14 , <u>3539</u> , 3659
\becquerelbase <u>4455</u>
\bel 15 , <u>3731</u>
\bit 19 , <u>3806</u>
\byte 19 , <u>3806</u>

C	
\calory	4063
\candela	13, 3214, 4431, 4465, 4466
\Celsius	3567
\celsius	14, 3567, 4075, 4079, 4084, 4264
\celsiusbase	4455
\centi	14, 3507, 3584, 3633, 3634, 3639, 3640, 3684, 3686, 3706, 4019, 4027, 4034, 4058, 4495–4497
\centiliter	4022
\centimeter	4015
\centimetre	16, 3580
\centimetrecubed	16, 3630
\centimetresquared	16, 3638
\cl	4045
\clight	20, 3777, 4505, 4507, 4509, 4511, 4513, 4515, 4517, 4519, 4521, 4523, 4525, 4527
closeerr (option)	24
closefrac (option)	29
\cm	16, 3702
\cmc	16, 3682
\cms	16, 3682
color (option)	31
colorall (option)	31
colorneg (option)	31
colorunit (option)	31
colorvalue (option)	31
colour (option)	31
colourall (option)	31
colourneg (option)	31
colourunits (option)	31
colourvalues (option)	31
\coulomb	14, 3539, 4365, 4423
\coulombbase	4438
\cubed	12, 3222, 3632–3636, 3684, 3685, 4030–4032, 4188
\cubic	12, 3222, 3637, 3766, 4181, 4365, 4456
\cubiccentimetre	16, 3630
\cubicdecimetre	16, 3630
\cubicmeter	4022
\cubicmetre	3630, 4367
\cubicmicrometer	4022
\cubicmicrometre	3630
\cubicmillimeter	4022
\cubicmillimetre	3630
\curie	18, 3716
D	
\dalton	19, 3761
\Day	15, 3731
\days	4005
\dday	15, 3731
debug (option)	32
\deca	14, 3526
\decaD	4627
\decad	4627
\deci	14, 3507, 3585, 3637, 3685, 3707, 3766, 4020, 4028, 4059
\deciliter	4022
decimalsymbol (option)	23
\decimeter	4015
\decimetre	16, 3580
\degC	4148
\degF	4148
\Degree	3731
\degree	15, 3731, 4077, 4081, 4086
\degreecelsius	4260
\deka	3536
\derbecquerel	4425
\dercelsius	4425
\dercoulomb	4413
\derfarad	4413
\dergray	4425
\derhenry	4425
\derhertz	4413, 4433
\derjoule	4413
\derkatal	4425
\derlumen	4425
\derlux	4425
\dernewton	4413
\derohm	4413
\derpascal	4413
\derradian	4413
\dersiemens	4425
\dersievert	4425
\dersteradian	4413
\dertesla	4425
\dervolt	4413
\derwatt	4413
\derweber	4425
detectdisplay (option)	23
digitsep (option)	23
\dimexpr	2107, 2115, 2136, 2138
\dl	4045
\dm	16, 3702
\dmc	16, 3682
\document	1031
dp (option)	25

E	
\electronvolt	17, 3645–3649, 3695–3700, 3752, 4498–4503, 4505, 4507, 4509, 4511, 4513, 4515, 4517, 4519, 4521, 4523, 4525, 4527
emulate (option)	32
\ensureupmath	4143
errspace (option)	23
\eV	17, 3694, 3778
eVcorra (option)	30
eVcorrb (option)	30
\everyeof	1238
\eVoverc	4504
\eVovercsq	4504
\eVperc	20, 3777
\exa	14, 3526
\exbi	19, 3796
expbase (option)	25
expproduct (option)	25
F	
\farad	14, 3539, 3613–3617
\faradbase	4438
\fb	20, 3785
\femto ..	14, 3507, 3587, 3592, 3598, 3613, 3675, 3688, 3711, 3781, 3787, 4022, 4050, 4053, 4484, 4490
\femtobarn	20, 3779
\femtofarad	17, 3608
\femtogram	15, 3587
\femtoliter	4022
\femtomole	16, 3592
\femtosecond	16, 3597
\fg	15, 3687, 4045
fixdp (option)	25
\fl	4045
\fmol	16, 3675
\fnum	4533
\fourth	4175, 4451
fraction (option)	28
\fs	16, 3709
G	
\gal	18, 3716
\gauss	20, 3774
\gensymbcelsius	4066
\gensymbdegree	4066
\gensymbmicro	4066
\gensymbohbm	4066
\GeV	17, 3694
\GeVoverc	4508
\GeVovercsq	4518
\ggray	14, 3539
\GHz	17, 3669
\gibi	19, 3796
\giga ...	14, 3526, 3622, 3627, 3648, 3654, 3673, 3699, 4502, 4515, 4525
\gigaelectronvolt	17, 3642
\gigahertz	17, 3651
\gigaohm	17, 3619
\GinveV	4498
\gram .	3537, 3587–3591, 3687–3692, 4050
\Gray	14, 3539
\gray	4233
\graybase	4455
H	
\hectar	4033
\hectare	18, 3716
\hecto	14, 3526, 3658, 3721, 4029, 4037, 4060
\hectoliter	4022
\hectopascal	17, 3656
\henry	14, 3539
\henrybase	4455
\hertz	14, 17, 3539, 3651–3655, 3669–3674
\hertzbase	4438, 4467
\hl	4045
\hour	15, 3650, 3701, 3731
\Hz	17, 3669
I	
\ifdefined	1032, 1039, 3186
\ifsi@addunitpower	680, 2657
\ifsi@allowoptarg	657, 2764
\ifsi@allowzeroexp	383, 1329
\ifsi@ang@fixdp	2023, 2064
\ifsi@ang@padlarge	470, 2190
\ifsi@ang@padsmlal	470, 2244
\ifsi@ang@sign	1674, 2188
\ifsi@ang@toarc	499, 2052
\ifsi@ang@todec	499, 2059
\ifsi@astroang	521, 2200
\ifsi@colourneg	773, 1317
\ifsi@colourunits	734, 1110
\ifsi@colourvalues .	734, 1099, 2464
\ifsi@debug	121, 146
\ifsi@detectdisplay	337, 1131, 1149

<code>\ifsi@fam@set</code>	<u>1074</u> , 1090, 3308	<code>\ifsi@old@novolumeabbr</code> <u>845</u> , 3907, 4052
<code>\ifsi@fixdp</code>	<u>640</u> , 1698, 2023	<code>\ifsi@old@noweightabbr</code> <u>845</u> , 3910, 4049
<code>\ifsi@frac</code> .	<u>658</u> , 2951, 2976, 3100, 3106	<code>\ifsi@old@noxspace</code>	<u>841</u> , 3892
<code>\ifsi@gensymb</code>	<u>837</u> ,	<code>\ifsi@old@OHM</code>	<u>837</u> , 3516,
3519, 3556, 3570, 3624, 3740, 4071		3553, 3567, 3619, 3737, 3998, 4082	
<code>\ifsi@inlinebtext</code>	329, 1054	<code>\ifsi@old@ohm</code>	<u>837</u> , 3999
<code>\ifsi@logmin</code>	<u>121</u> , 126, 134, 140	<code>\ifsi@old@plain</code>	<u>877</u> , 4555
<code>\ifsi@lognone</code> .	<u>121</u> , 125, 133, 139, 145	<code>\ifsi@old@pstricks</code>	<u>862</u> , 4236
<code>\ifsi@num@ambigerr</code>	1354, <u>1732</u>	<code>\ifsi@old@redef</code>	4067
<code>\ifsi@num@delplus</code> ...	565, 573, <u>1538</u>	<code>\ifsi@old@redef-gensymb</code>	<u>837</u>
<code>\ifsi@num@erropen</code> .	<u>1283</u> , 1381, 1414	<code>\ifsi@old@spaceqspace</code> ..	<u>880</u> , 4619
<code>\ifsi@num@intab</code> <u>1225</u> ,	1343, 1384, 1408	<code>\ifsi@old@squaren</code> ..	<u>862</u> , 4218, 4227
<code>\ifsi@num@padlead</code> ..	<u>388</u> , 1585, 1822	<code>\ifsi@old@textstyle</code>	<u>867</u> , 4250
<code>\ifsi@num@padtrail</code>	<u>388</u> , 1590	<code>\ifsi@old@thickqspace</code> ..	<u>855</u> , 4208
<code>\ifsi@num@signexp</code>	<u>425</u> , 1558	<code>\ifsi@old@thickspace</code> ...	<u>855</u> , 4199
<code>\ifsi@num@signmant</code>	<u>425</u> , 1552	<code>\ifsi@old@thingqspace</code> ...	<u>855</u> , 4214
<code>\ifsi@num@textmode</code>	<u>297</u> , 1078	<code>\ifsi@old@thinspace</code>	<u>855</u> , 4205
<code>\ifsi@obeybold</code>	<u>328</u> , 1145	<code>\ifsi@old@thinspace</code> ...	<u>877</u> , 4549
<code>\ifsi@obeyfamily</code>	<u>327</u> , 1123	<code>\ifsi@old@tight</code> .	<u>833</u> , <u>875</u> , 3855, 4543
<code>\ifsi@obeyitalic</code>	<u>336</u> , 1163	<code>\ifsi@old@ugly</code>	<u>833</u> , 3861
<code>\ifsi@obeymode</code>	<u>296</u> , 1063	<code>\ifsi@out@num</code>	1077, 1092, <u>3334</u>
<code>\ifsi@old@amssymb</code>	<u>862</u> , 4224	<code>\ifsi@prefixsymbolic</code> ...	<u>387</u> , 2911
<code>\ifsi@old@binary</code> ...	<u>867</u> , 4253, 4405	<code>\ifsi@prespace</code>	<u>651</u>
<code>\ifsi@old@cdot</code>	<u>855</u> , 4196	<code>\ifsi@redefsymbols</code>	<u>820</u> , 919
<code>\ifsi@old@commas</code>	<u>877</u> , 4552	<code>\ifsi@repeatunits</code> <u>680</u> , 1365, 2010, 2645
<code>\ifsi@old@derived</code>	<u>867</u> , 4413	<code>\ifsi@retainplus</code>	<u>379</u> , 1530
<code>\ifsi@old@derivedinbase</code>	<u>867</u> , 4438	<code>\ifsi@seperr</code>	<u>375</u> , 1353, 1748
<code>\ifsi@old@english</code>	<u>873</u> , 4537	<code>\ifsi@sepfour</code> ..	<u>378</u> , 1943, 1982, 2570
<code>\ifsi@old@french</code>	<u>873</u> , 4540	<code>\ifsi@slash</code>	<u>658</u> , 3140, 3160
<code>\ifsi@old@Gray</code>	<u>862</u> , 4239	<code>\ifsi@stickyper</code>	<u>658</u> , 2970, 3020
<code>\ifsi@old@italian</code>	<u>862</u> , 4245	<code>\ifsi@strict</code>	<u>249</u>
<code>\ifsi@old@LITER</code>	<u>841</u> , 4006	<code>\ifsi@strictarc</code>	<u>522</u> , 2034
<code>\ifsi@old@liter</code>	<u>841</u> , 4005	<code>\ifsi@switch</code>	<u>45</u> ,
<code>\ifsi@old@loose</code> .	<u>833</u> , <u>875</u> , 3858, 4546	156, 177, 1260, 1429, 1443, 1453,	
<code>\ifsi@old@mediumqspace</code> .	<u>855</u> , 4211	1591, 1615, 1630, 1667, 1718,	
<code>\ifsi@old@mediumspace</code> ..	<u>855</u> , 4202	1774, 1782, 1794, 1896, 1906,	
<code>\ifsi@old@nice</code>	<u>833</u>	1918, 2366, 2406, 2414, 3494, 4160	
<code>\ifsi@old@noabbr</code> <u>845</u> ,	3895, 4038, 4045	<code>\ifsi@tab@expsign</code>	<u>548</u> , 2529
<code>\ifsi@old@noamperageabbr</code> ...	<u>845</u>	<code>\ifsi@tab@fixed</code> .	<u>523</u> , 597, 2446, 2459
<code>\ifsi@old@noams</code>	<u>867</u> , 4256	<code>\ifsi@tab@mantsign</code>	<u>548</u> , 2525
<code>\ifsi@old@noconfig</code>	<u>841</u> , 4090	<code>\ifsi@tabalignexp</code> <u>547</u> , 2539, 2548, 2556, 2560
<code>\ifsi@old@noenergyabbr</code> .	<u>845</u> , 3913	<code>\ifsi@tabautofit</code>	<u>649</u> , 2447
<code>\ifsi@old@nofrequncyabbr</code>	<u>845</u> , 3898	<code>\ifsi@textmode</code>	<u>1074</u> , 3313
<code>\ifsi@old@nolengthabbr</code> .	<u>845</u> , 3916	<code>\ifsi@tightpm</code>	<u>463</u>
<code>\ifsi@old@nomolabbr</code>	<u>845</u> , 3901	<code>\ifsi@trapambigerr</code> <u>375</u> , 1336, 1360, 2646
<code>\ifsi@old@noprefixcmds</code> .	<u>872</u> , 4480		
<code>\ifsi@old@notimeabbr</code> <u>845</u> ,	3919, 4046		
<code>\ifsi@old@novoltageabbr</code> <u>845</u> , 3904, 4040		

\ifsi@trapambigfrac	<u>693</u> , <u>3161</u>	\kilo	... <u>14</u> , <u>3526</u> , <u>3538</u> , <u>3586</u> , <u>3607</u> ,
\ifsi@unittextmode	<u>297</u> , <u>1084</u>		3609, 3611, 3620, 3625, 3641,
\ifsi@unt@first	<u>2722</u> , <u>2847</u>		3643, 3646, 3650, 3652, 3657,
\ifsi@unt@litout	...	<u>2693</u> , <u>2713</u> ,		3668, 3671, 3680, 3687, 3694,
		<u>2779</u> , <u>2798</u> , <u>2820</u> , <u>3005</u> , <u>3134</u> , <u>3233</u>		3697, 3701, 3708, 4021, 4035,
\ifsi@unt@litprefix	...	<u>2787</u> , <u>3008</u>		4041, 4064, 4356, 4500, 4511, 4521
\ifsi@unt@littest		\kiloampere <u>16</u> , <u>3603</u>
		<u>2682</u> , <u>2776</u> , <u>2795</u> , <u>2817</u> , <u>3004</u> , <u>3230</u>	\kilocalory <u>4063</u>
\ifsi@unt@num	<u>2627</u> , <u>2709</u> , <u>3119</u>	\kiloelectronvolt <u>17</u> , <u>3642</u>
\ifsi@unt@per		\kilogram
		<u>2899</u> , <u>2931</u> , <u>2952</u> , <u>3001</u> , <u>3028</u> , <u>3107</u>		<u>13</u> , <u>15</u> , <u>3214</u> , <u>3537</u> , <u>4348</u> , <u>4351</u> ,
\ifsi@unt@perseen	<u>3001</u> , <u>3029</u>		<u>4353</u> , <u>4356</u> , <u>4357</u> , <u>4362</u> , <u>4364</u> ,
\ifsi@unt@prepower		<u>2838</u> , <u>2890</u> , <u>2939</u>		<u>4366</u> , <u>4376</u> , <u>4417</u> , <u>4427</u> , <u>4434</u> ,
\ifsi@xspace	<u>650</u> , <u>2754</u>		<u>4442</u> , <u>4443</u> , <u>4445</u> , <u>4446</u> , <u>4449</u> ,
\ilu	<u>3930</u>		<u>4451</u> , <u>4454</u> , <u>4456</u> , <u>4459</u> , <u>4460</u> , <u>4463</u>
\invab	<u>4488</u>	\kilohertz <u>17</u> , <u>3651</u>
\invattobarn	<u>4480</u>	\kilojoule <u>17</u> , <u>3642</u>
\invbarn	<u>4480</u>	\kilometer <u>4015</u>
\invcmsqpersec	<u>4495</u>	\kilometre <u>16</u> , <u>3580</u>
\invcmsqpersecond	<u>4495</u>	\kilonewton <u>17</u> , <u>3656</u>
\inveV	<u>4498</u>	\kiloohm <u>17</u> , <u>3619</u>
\invfb	<u>4488</u>	\kilovolt <u>17</u> , <u>3608</u>
\invfemtobarn	<u>4480</u>	\kilowatt <u>17</u> , <u>3608</u>
\invnanobarn	<u>4480</u>	\kilowatthour <u>17</u> , <u>3642</u>
\invnb	<u>4488</u>	\kinveV <u>4498</u>
\invpb	<u>4488</u>	\kJ <u>17</u> , <u>3694</u>
\invpicobarn	<u>4480</u>	\km <u>16</u> , <u>3702</u>
\invyb	<u>4488</u>	\kV <u>17</u> , <u>3680</u>
\invyoctobarn	<u>4480</u>	\kv <u>4038</u>
\invzb	<u>4488</u>	\kWh <u>3694</u>
\invzeptobarn	<u>4480</u>		
			L	
J			\language	name <u>3280</u>
\joule	<u>14</u> , <u>3539</u> , <u>3642</u> - <u>3644</u> , <u>3694</u> , <u>4355</u> , <u>4361</u> , <u>4420</u> , <u>4434</u>	\lightyear <u>20</u> , <u>3791</u>
\joulebase	<u>4438</u>	\liter <u>13</u> ,
				<u>15</u> , <u>3731</u> , <u>4005</u> , <u>4022</u> - <u>4029</u> , <u>4053</u> - <u>4060</u>
K			\litre	<u>13</u> , <u>15</u> , <u>3630</u> , <u>3631</u> , <u>3682</u> , <u>3683</u> , <u>3731</u>
\kA	<u>16</u> , <u>3661</u>	\llap <u>2415</u>
\katal	<u>14</u> , <u>3539</u>	load (option) <u>31</u>
\katalbase	<u>4455</u>	locale (option) <u>31</u>
\kelvin	<u>13</u> ,	loctolang (option) <u>31</u>
		<u>3214</u> , <u>4355</u> , <u>4362</u> , <u>4363</u> , <u>4430</u> , <u>4464</u>	log (option) <u>32</u>
\kern	<u>813</u> , <u>815</u> , <u>3755</u> , <u>3756</u>	\lumen <u>14</u> , <u>3539</u> , <u>4432</u>
\keV	<u>17</u> , <u>3694</u>	\lumenbase <u>4455</u>
\keVoverc	<u>4508</u>	\lumiunits <u>4495</u>
\keVovercsq	<u>4518</u>	\lux <u>14</u> , <u>3539</u>
\kg	<u>15</u> , <u>3687</u>	\luxbase <u>4455</u>
\kHz	<u>17</u> , <u>3669</u>	M	
\kibi	<u>19</u> , <u>3796</u>	\mA <u>16</u> , <u>3661</u>
			mathrm (option) <u>2</u>

<code>\mathscelsius (option)</code>	30	<code>\microfarad</code>	17, 3608
<code>\mathsdegree (option)</code>	30	<code>\microgram</code>	15, 3587
<code>\mathsf (option)</code>	22	<code>\microliter</code>	4022
<code>\mathsminute (option)</code>	30	<code>\microlitre</code>	16, 3630
<code>\mathsmu (option)</code>	30	<code>\micrometer</code>	4015
<code>\mathsOmega (option)</code>	30	<code>\micrometre</code>	16, 3580
<code>\mathsringA (option)</code>	30	<code>\micromole</code>	16, 3592
<code>\mathsrm (option)</code>	22	<code>\micron</code>	20, 3774
<code>\mathsssecond (option)</code>	30	<code>\microsecond</code>	16, 3597
<code>\mathssf (option)</code>	22	<code>\mics</code>	16, 3709
<code>\mathstt (option)</code>	22	<code>\milli</code> 14, 3507, 3583, 3591, 3596, 3602,	
<code>\mathstt (option)</code>	22	3606, 3608, 3610, 3617, 3618,	
<code>\mebi</code>	19, 3796	3631, 3636, 3642, 3645, 3651,	
<code>\mega</code>	14, 3526, 3612,	3656, 3660, 3667, 3670, 3679,	
3621, 3626, 3644, 3647, 3653,		3681, 3682, 3692, 3696, 3705,	
3659, 3672, 3698, 4501, 4513, 4523		3715, 3725, 3775, 4018, 4026,	
<code>\megabecquerel</code>	17, 3656	4032, 4042, 4057, 4499, 4509, 4519	
<code>\megaelectronvolt</code>	17, 3642	<code>\milliampere</code>	16, 3603
<code>\megahertz</code>	17, 3651	<code>\millibar</code>	18, 3716
<code>\megajoule</code>	3642	<code>\millielectronvolt</code>	17, 3642
<code>\megaohm</code>	17, 3619	<code>\millifarad</code>	17, 3608
<code>\megawatt</code>	17, 3608	<code>\milligram</code>	16, 3587
<code>\meter</code> . 13, 3214, 4015–4021, 4030–4035		<code>\millihertz</code>	17, 3651
<code>\metre</code> 13, 13, 3214, 3580–3586,		<code>\millijoule</code>	3642
3632–3641, 3684–3686, 3702–		<code>\milliliter</code>	4022
3708, 3766, 3774, 4351, 4357,		<code>\millilitre</code>	16, 3630
4363, 4365, 4372–4374, 4414,		<code>\millimeter</code>	4015
4415, 4417–4419, 4428, 4432,		<code>\millimetre</code>	16, 3580
4439, 4440, 4442, 4443, 4451,		<code>\millimole</code>	16, 3592
4456, 4465, 4466, 4495–4497, 4640		<code>\millinewton</code>	17, 3656
<code>\MeV</code>	17, 3694	<code>\millisecond</code>	16, 3597
<code>\mev</code>	17, 3694	<code>\millisiemens</code>	17, 3608
<code>\MeVoverc</code>	4508	<code>\millisievert</code>	17, 3656
<code>\meVoverc</code>	4508	<code>\millivolt</code>	17, 3608
<code>\MeVovercsq</code>	4518	<code>\milliwatt</code>	17, 3608
<code>\meVovercsq</code>	4518	<code>\minute</code>	15, 3731
<code>\mg</code>	16, 3687	<code>\MinveV</code>	4498
<code>\MHz</code>	17, 3669	<code>\minveV</code>	4498
<code>\mHz</code>	17, 3669	<code>\ml</code>	16, 3682, 4045
<code>\micA</code>	16, 3661	<code>\mm</code>	16, 3702
<code>\micg</code>	15, 3687	<code>\mmHg</code>	19, 3761
<code>\micl</code>	16, 3682, 4045	<code>\mmol</code>	16, 3675
<code>\micm</code>	16, 3702	<code>mode (option)</code>	22
<code>\micmol</code>	16, 3675	<code>\Molar</code>	19, 3761
<code>\Micro</code>	3507	<code>\molar</code>	19, 3761
<code>\micro</code>	14, 3507, 3582,	<code>\mole</code> ... 13, 3214, 3592–3596, 3675–	
3590, 3595, 3601, 3605, 3616,		3679, 3766, 4355, 4356, 4436, 4470	
3630, 3635, 3666, 3678, 3683,		<code>\mp</code>	1251
3691, 3704, 3714, 3774, 4017,		<code>\mrad</code>	20, 3774
4025, 4031, 4056, 4076, 4080, 4085			
<code>\microampere</code>	16, 3603		

\ms	16 , 3709	4294, 4341, 4343, 4368, 4371–
\mV	17 , 3680	4374, 4376, 4414–4426, 4428–
\mv	4038	4436, 4439–4443, 4445–4448,
		4450, 4453, 4455, 4458, 4460,
		4462, 4464–4470, 4481–4493,
		4495–4504, 4506, 4508, 4510,
		4512, 4514, 4516, 4518, 4520,
		4522, 4524, 4526, 4578, 4591,
		4602, 4606, 4609, 4613, 4622–4625
N		
\nA	16 , 3661	
\nano	14 , 3507 , 3581, 3589,	
	3594, 3600, 3604, 3615, 3665,	
	3677, 3690, 3703, 3713, 3779,	
	3785, 4016, 4024, 4055, 4482, 4488	
\nanoampere	16 , 3603	\nl
\nanobarn	20 , 3779	4045
\nanofarad	17 , 3608	\nm
\nanog	15 , 3687	16 , 3702
\nanogram	15 , 3587	\nmol
\nanoliter	4022	16 , 3675
\nanometer	4015	noload (option)
\nanometre	16 , 3580	31
\nanomole	16 , 3592	\ns
\nanosecond	16 , 3597	16 , 3709
\nb	20 , 3785	\num
\NC@list	2288	5 , 1217, 1248, 2651,
\NC@rewrite@S	2293	2855, 2990, 3124, 3877, 4108 , 4533
\NC@rewrite@s	2293	numaddn (option)
negcolor (option)	31	23
negcolour (option)	31	numcloseerr (option)
\neper	15 , 3731	24
\newnosepunit	3952	numdecimal (option)
\newpower	18 ,	23
	2613 , 3222–3226, 4182, 4219, 4223	numdigits (option)
\newprefix	19 , 2599, 3510–3515, 3517, 3520,	23
	3523–3536, 3798–3805, 4085, 4266	numexp (option)
\newton	14 , 3539 , 3656,	23
	3657, 4350, 4374, 4418, 4419, 4641	numgobble (option)
\newtonbase	4438	23
\newunit	18 , 2585,	numopenerr (option)
	3214–3221, 3537, 3541–3552,	24
	3554, 3560–3566, 3568, 3571,	numprod (option)
	3574, 3575, 3580–3618, 3620–	24
	3622, 3625–3627, 3630–3660,	numsign (option)
	3664–3681, 3684–3687, 3689–	23
	3711, 3713–3715, 3719–3730,	
	3733–3736, 3738, 3741, 3744–	O
	3751, 3757, 3759, 3760, 3765–	obeyall (option)
	3769, 3775–3784, 3794, 3795,	22
	3806, 3807, 3952, 4013–4037,	obeybold (option)
	4041, 4042, 4047, 4050, 4053–	22
	4060, 4063–4065, 4083, 4084,	obeyfamily (option)
	4086, 4235, 4260–4265, 4285,	22
		obeyitalic (option)
		22
		obeymode (option)
		22
		\oersted
		4622
		\Ohm
		3553 , 3620–3622
		\ohm
		14 , 3553 ,
		3625–3627, 4074, 4078, 4083, 4373
		\ohmbase
		4438
		\Omega
		785, 786
		openerr (option)
		24
		openfrac (option)
		29
		options:
		addsign
		25
		allowoptarg
		29
		allowzeroexp
		25
		alsoload
		31
		angformat
		26
		anglesep
		23
		astroang
		26
		closeerr
		24
		closefrac
		29
		color
		31
		colorall
		31
		colorneg
		31

colorunit	31	obeymode	22
colorvalue	31	openerr	24
colour	31	openfrac	29
colourall	31	padangle	26
colourneg	31	padnumber	25
colourunits	31	per	28
colourvalues	31	prefixbase	29
debug	32	prefixproduct	29
decimalsymbol	23	prefixsymbolic	29
detectdisplay	23	prespace	29
digitsep	23	redefsymbols	30
dp	25	repeatunits	24
emulate	32	retainplus	25
errspace	23	seperr	24
eVcorra	30	sepfour	23
eVcorrb	30	sign	25
expbase	25	slash	28
expproduct	25	stickyper	29
fixdp	25	strict	32
fraction	28	strictarc	26
load	31	tabalign	27
locale	31	tabalignexp	27
loctolang	31	tabautofit	28
log	32	tabformat	27
mathrm	22	tabnumalign	26
mathscelsius	30	tabtextalign	27
mathsdegree	30	tabunitalign	27
mathssf	22	textcelsius	30
mathsminute	30	textdegree	30
mathsmu	30	textminute	30
mathsOmega	30	textmode	22
mathsringA	30	textmu	30
mathsrm	22	textOmega	30
mathssecond	30	textringA	30
mathssf	22	textrm	22
mathstt	22	textsecond	30
mathtt	22	textsf	22
mode	22	texttt	22
negcolor	31	tightpm	24
negcolour	31	trapambigerr	24
noload	31	trapambigfrac	29
numaddn	23	unitcolor	31
numcloseerr	24	unitcolour	31
numdecimal	23	unitmathsrm	22
numdigits	23	unitmathssf	22
numexp	23	unitmathstt	22
numgobble	23	unitmode	22
numopenerr	24	unitsep	23
numprod	24	unitspace	23
numsign	23	unittextrm	22
obeyall	22	unittextsf	22
obeybold	22	unittexttt	22
obeyfamily	22	valuecolor	31
obeyitalic	22	valuecolour	31

valuemathrm	22	\pow	4636
valuemathsf	22	\power	2949, 4175, 4636
valuemathsrn	22	prefixbase (option)	29
valuemathssf	22	prefixproduct (option)	29
valuemathstt	22	prefixsymbolic (option)	29
valuemathtt	22	prespace (option)	29
valuemode	22	\prime	806–809
valuesep	23	\protected@xdef	1240
valuetextrm	22	\providecommand	4298, 4346
valuetextsf	22	\providepower	18, 2613
valuetexttt	22	\provideprefix	19, 2599
xspace	29	\provideunit
		 18, 2585, 3557, 3682, 3683,
			3688, 3712, 3774, 3785–3790, 4626
		\ps	16, 3709
P			
\pA	16, 3661		
padangle (option)	26		
padnumber (option)	25		
\pamminute	4622		
\parsec	20, 3791		
\parsecond	4622		
\pascal	14, 3553, 3658, 4371		
\pascalbase	4438		
\pb	20, 3785		
\pebi	19, 3796		
\per	12, 3001, 3766,		
	3778, 3940, 4175, 4282, 4342,		
	4481–4493, 4495–4503, 4505,		
	4507, 4509, 4511, 4513, 4515,		
	4517, 4519, 4521, 4523, 4525, 4527		
per (option)	28		
\percent	15, 3731		
\peta	14, 3526		
\pg	15, 3687		
\pico	14, 3507, 3580, 3588,		
	3593, 3599, 3603, 3614, 3664,		
	3676, 3689, 3702, 3712, 3780,		
	3786, 4015, 4023, 4054, 4483, 4489		
\picoampere	16, 3603		
\picobarn	20, 3779		
\picofarad	17, 3608		
\picogram	15, 3587		
\picoliter	4022		
\picom	16, 3702		
\picometer	4015		
\picometre	16, 3580		
\picomole	16, 3592		
\picosecond	16, 3597		
\pl	4045		
\pm	600, 896, 897, 1251, 3426		
\pmol	16, 3675		
\pnt	4124		
		R	
		\rad	18, 3716, 3775
		\radian	14, 3574
		\radianbase	4438
		\raiseto	12, 3227
		\raiseto@opt@si	3227
		\reciprocal 4175, 4292, 4344,
			4414, 4416, 4420, 4422–4425,
			4427, 4429, 4434, 4439, 4441,
			4443, 4449, 4451, 4456, 4459, 4460
		redefsymbols (option)	30
		\rem	18, 3716
		\renewnosepunit	3952
		\renewpower	18, 2613, 4225
		\renewprefix	19, 2599
		\renewunit
			... 18, 2585, 3538, 3953, 4010, 4237
		\repeat	1859
		repeatunits (option)	24
		\requiresiconfigs
		 36, 3354, 3579, 3663, 3764,
			3773, 3888, 4089, 4174, 4479, 4617
		retainplus (option)	25
		\roentgen	18, 3716
		\rp	4175, 4436, 4470
		\rpcubed	4184
		\rpcubic	4175, 4446, 4449, 4454
		\rperminute	4260
		\rpfourth	4175, 4466
		\rpsquare
			.. 4175, 4415, 4418, 4428, 4432,
			4440, 4451, 4454, 4456, 4463, 4465
		\rpsquared	4184, 4417, 4427, 4442,
			4444, 4445, 4459, 4460, 4463, 4468

S	
\Sec	16 , 3709
\second	13 , 16 , 3214 , 3597–3602 , 3709–3715 , 4047 , 4348 , 4350 , 4352 , 4354 , 4358 , 4367 , 4370–4372 , 4416 , 4417 , 4420 , 4421 , 4427 , 4436 , 4441–4443 , 4445–4447 , 4449 , 4451 , 4454 , 4456 , 4459 , 4460 , 4463 , 4468 , 4470 , 4495–4497 , 4641
\sek	4045
\selectlanguage	3280
seperr (option)	24
sepfour (option)	23
\setfnumdsym	4534
\setfnumgsym	4534
\setfnummsym	4534
\setMathCelsius	3954
\setMathDegree	3954
\setMathmu	3954
\setMathOmega	3954
\setTextCelsius	3954
\setTextDegree	3954
\setTextmu	3954
\setTextOmega	3954
\SI 10 , 2580 , 2643 , 3866 , 3868 , 4246 , 4248	
\si 12 , 2426 , 2580 , 3930 , 4629 , 4632 , 4635	
\si@addtocname	85 , 3299
\si@addtolist	79 , 273 , 723 , 3382 , 3390 , 3477 , 3479 , 3482 , 3495
\si@addunitpowerfalse	684
\si@addunitpowertrue	691
\si@ang@arc	2049 , 2050
\si@ang@arcdeg	2124
\si@ang@arcfix	2058 , 2064 , 2125
\si@ang@arcmin	2124
\si@ang@arcsec	2124
\si@ang@arctodec	2060 , 2089
\si@ang@astrosign	2201 , 2258
\si@ang@dec	2028 , 2050
\si@ang@decimalsymbol .	2199 , 2258
\si@ang@dectoarc	2053 , 2124
\si@ang@degsg	2190
\si@ang@fix	2051 , 2058 , 2064 , 2090 , 2125 , 2247 , 2253
\si@ang@fixdptrue	2126
\si@ang@ifnum	2078 , 2095–2097 , 2130 , 2209 , 2221
\si@ang@killdegree	2258
\si@ang@killminute	2258
\si@ang@killsecond	2258
\si@ang@minnum	2205
\si@ang@mins	2190
\si@ang@movesign ..	2200 , 2248 , 2254
\si@ang@notnum ..	2121–2123 , 2169 , 2184
\si@ang@num ...	2205 , 2206 , 2238 , 2245
\si@ang@pad ...	2216 , 2218 , 2230 , 2244
\si@ang@padlargefalse	475
\si@ang@padlargetrue	482 , 487 , 492 , 497
\si@ang@padsmallfalse	474
\si@ang@padsmalltrue	478 , 486 , 491 , 496
\si@ang@parse	2021 , 2022
\si@ang@secnum	2205
\si@ang@secs	2190
\si@ang@sepint	2135 , 2137 , 2170
\si@ang@signlessnum	2226 , 2234 , 2236 , 2245
\si@ang@signtrue	2211 , 2223
\si@ang@sint	2170
\si@ang@strippt ...	2141 , 2146 , 2170
\si@ang@toarcfalse	503
\si@ang@toarctrue	515 , 519
\si@ang@todectfalse	504
\si@ang@todecttrue	507 , 511
\si@ang@typeset	2055 , 2062 , 2119 , 2168 , 2187 , 2189
\si@anglesep	293 , 2241
\si@blockpkgs ...	50 , 3371 , 3372 , 3382
\si@catcodes	19 , 3506
\si@checkpkgs ...	50 , 3385 , 3386 , 3390
\si@closeerr	375 , 1376 , 1416
\si@closefrac	693 , 3169
\si@colour	1090 , 3311
\si@colourcmd	1090 , 3311
\si@colournegfalse	776
\si@colournegtrue	779
\si@colourunitsfalse .	738 , 753 , 762
\si@colourunitstrue ..	741 , 756 , 765
\si@colourvaluesfalse	745 , 752 , 761
\si@colourvaluetrue .	748 , 757 , 766
\si@debugfalse	229
\si@debugtrue	242 , 246
\si@decimalsymbol	291 , 1708 , 1835 , 2199 , 2262 , 2476 , 2479 , 2507 , 2520 , 2552 , 4124
\si@digitsep ...	288 , 1977 , 1994 , 2577
\si@emclash	3365 , 3850 , 3852 , 3885 , 3887
\si@emulate	272 , 3464 , 3465

<code>\si@emulating</code>	<u>3369</u> , <u>3848</u> ,	<code>\si@fix@tightcdot</code>	<u>886</u>
	<u>3883</u> , <u>4098</u> , <u>4167</u> , <u>4478</u> , <u>4531</u> , <u>4561</u>	<code>\si@fix@tightpm</code>	<u>466</u> , <u>894</u>
<code>\si@errspace</code>	<u>278</u> , <u>1375</u>	<code>\si@fix@tighttimes</code>	<u>886</u>
<code>\si@eVcorra</code>	<u>825</u> , <u>3755</u>	<code>\si@fix@times</code>	<u>886</u>
<code>\si@eVcorrb</code>	<u>825</u> , <u>3756</u>	<code>\si@fix@two</code>	<u>899</u>
<code>\si@eVspacea</code>	<u>3752</u>	<code>\si@fixdpfalse</code>	<u>2024</u> , <u>2073</u> , <u>2076</u>
<code>\si@eVspaceb</code>	<u>3752</u>	<code>\si@fixdptrue</code> ..	<u>648</u> , <u>2071</u> , <u>2094</u> , <u>2449</u>
<code>\si@expanddefault</code>	<u>3468</u> , <u>3500</u>	<code>\si@frac</code> <u>669</u> , <u>697</u> , <u>700</u> , <u>704</u> , <u>708</u> , <u>712</u> ,	
<code>\si@expbase</code>	<u>380</u> , <u>1404</u> , <u>2536</u> , <u>3123</u>		<u>716</u> , <u>720</u> , <u>938</u> , <u>3142</u> , <u>3146</u> , <u>3155</u> , <u>3878</u>
<code>\si@expproduct</code>	<u>380</u> , <u>1400</u> , <u>2536</u>	<code>\si@fracfalse</code>	<u>664</u>
<code>\si@extension</code> .	<u>3340</u> , <u>3345</u> , <u>3348</u> , <u>3351</u>	<code>\si@fracttrue</code>	<u>667</u> , <u>673</u> , <u>677</u>
<code>\si@fam@bold</code>	<u>1144</u> , <u>1215</u> , <u>3312</u>	<code>\si@frc@displen</code> ..	<u>964</u> , <u>976</u> , <u>984</u> , <u>988</u>
<code>\si@fam@boldify</code>	<u>1213</u>	<code>\si@frc@frac</code>	<u>697</u> , <u>938</u> , <u>1020</u>
<code>\si@fam@colourcmd</code>		<code>\si@frc@hook</code> <u>938</u> , <u>4587</u> , <u>4598</u> , <u>4611</u> , <u>4615</u>	
 <u>1070</u> , <u>1100</u> , <u>1111</u> , <u>1318</u> , <u>2465</u>	<code>\si@frc@mathsnf</code>	<u>970</u> , <u>974</u>
<code>\si@fam@detmaths</code>	<u>1132</u> , <u>1169</u>	<code>\si@frc@nice</code>	<u>700</u> , <u>708</u> , <u>938</u>
<code>\si@fam@detttext</code> <u>1128</u> , <u>1134</u> , <u>1139</u> , <u>1169</u>		<code>\si@frc@nicefrac</code>	<u>949</u> , <u>964</u>
<code>\si@fam@ifbinline</code>	<u>1053</u> , <u>1156</u>	<code>\si@frc@sfrac</code>	<u>712</u> , <u>716</u> , <u>938</u>
<code>\si@fam@ifbmaths</code> ..	<u>1046</u> , <u>1057</u> , <u>1150</u>	<code>\si@frc@slash</code>	<u>669</u> , <u>938</u> , <u>1028</u>
<code>\si@fam@ifbtext</code> <u>1046</u> , <u>1055</u> , <u>1152</u> , <u>1159</u>		<code>\si@frc@ssuplen</code>	
<code>\si@fam@ifitext</code>	<u>1059</u> , <u>1165</u>		. <u>964</u> , <u>982</u> , <u>984</u> – <u>986</u> , <u>994</u> , <u>1009</u> , <u>1011</u>
<code>\si@fam@italic</code>	<u>1162</u> , <u>3312</u>	<code>\si@frc@suplen</code> ...	<u>964</u> , <u>980</u> , <u>986</u> , <u>992</u>
<code>\si@fam@maths</code>	<u>1119</u> , <u>1172</u> ,	<code>\si@frc@textlen</code>	
	<u>1179</u> , <u>1185</u> , <u>1194</u> , <u>1201</u> , <u>1207</u> , <u>3330</u>		<u>964</u> , <u>978</u> , <u>985</u> , <u>990</u> , <u>1008</u> , <u>1011</u> , <u>1012</u>
<code>\si@fam@mode</code> ..	<u>1062</u> , <u>1220</u> , <u>2017</u> , <u>3872</u>	<code>\si@frc@textnf</code>	<u>972</u> , <u>1006</u>
<code>\si@fam@set</code>	<u>1076</u> , <u>3309</u>	<code>\si@frc@ugly</code>	<u>704</u> , <u>720</u> , <u>1017</u>
<code>\si@fam@setbold</code> <u>1154</u> , <u>1156</u> , <u>1159</u> , <u>1213</u>		<code>\si@gensymbtrue</code>	<u>4068</u>
<code>\si@fam@settrue</code>	<u>1116</u>	<code>\si@gobblethree</code> ...	<u>2777</u> , <u>2786</u> , <u>2796</u>
<code>\si@fam@sf</code>	<u>1031</u> , <u>1170</u>	<code>\si@ifdefinable</code>	
<code>\si@fam@text</code>	<u>1119</u> , <u>1174</u> ,	 <u>75</u> , <u>2586</u> , <u>2590</u> , <u>2596</u> ,
	<u>1181</u> , <u>1187</u> , <u>1196</u> , <u>1203</u> , <u>1209</u> , <u>3312</u>		<u>2600</u> , <u>2604</u> , <u>2610</u> , <u>2614</u> , <u>2618</u> , <u>2624</u>
<code>\si@fam@tt</code>	<u>1031</u> , <u>1177</u>	<code>\si@ifl@aded</code>	<u>3342</u>
<code>\si@fileprefix</code> <u>3340</u> , <u>3345</u> , <u>3348</u> , <u>3351</u>		<code>\si@ifloaded</code>	<u>3342</u> , <u>3347</u> ,
<code>\si@fix@cdot</code>	<u>886</u>		<u>3845</u> , <u>3849</u> , <u>3851</u> , <u>3884</u> , <u>3886</u> , <u>4562</u>
<code>\si@fix@comma</code>	<u>886</u>	<code>\si@ifmtarg</code>	<u>92</u> , <u>2025</u> , <u>2654</u>
<code>\si@fix@fullstop</code>	<u>886</u>	<code>\si@ifnotmtarg</code> <u>92</u> , <u>606</u> , <u>1242</u> , <u>2207</u> ,	
<code>\si@fix@med</code>	<u>881</u>		<u>2219</u> , <u>2231</u> , <u>2632</u> , <u>2637</u> , <u>2640</u> ,
<code>\si@fix@medium</code>	<u>881</u>		<u>2642</u> , <u>2653</u> , <u>2760</u> , <u>2854</u> , <u>2884</u> , <u>3876</u>
<code>\si@fix@minus</code>	<u>894</u>	<code>\si@inlinebtextfalse</code>	<u>331</u>
<code>\si@fix@mp</code>	<u>894</u>	<code>\si@inlinebtexttrue</code>	<u>334</u>
<code>\si@fix@none</code>	<u>902</u>	<code>\si@load</code>	<u>722</u>
<code>\si@fix@period</code>	<u>886</u>	<code>\si@loademfile</code>	<u>3356</u> , <u>3466</u>
<code>\si@fix@plus</code>	<u>894</u>	<code>\si@loadfile</code>	
<code>\si@fix@pm</code>	<u>468</u> , <u>894</u>		.. <u>733</u> , <u>3246</u> , <u>3346</u> , <u>3355</u> , <u>3364</u> , <u>3498</u>
<code>\si@fix@slash</code>	<u>901</u>	<code>\si@loc@load</code> ...	<u>830</u> , <u>3242</u> , <u>3269</u> , <u>3286</u>
<code>\si@fix@space</code>	<u>881</u>	<code>\si@loc@ltol</code>	<u>832</u> , <u>3268</u>
<code>\si@fix@stop</code>	<u>886</u>	<code>\si@loc@set</code>	<u>831</u> , <u>3248</u> , <u>3277</u> , <u>3292</u>
<code>\si@fix@ten</code>	<u>899</u>	<code>\si@loc@sisetup</code>	<u>3242</u>
<code>\si@fix@thick</code>	<u>881</u>	<code>\si@locale</code>	<u>829</u>
<code>\si@fix@thin</code>	<u>881</u>	<code>\si@loctolang</code>	<u>829</u>

\si@log@debug	\si@num@addpostzero ...
144, 192, 199, 202, 206, 209, 217,	1594, <u>1647</u>
219, 226, 921, 934, 1124, 1127,	\si@num@addprezero
1130, 1138, 1142, 1146, 1164,	1586, <u>1647</u>
1167, 1171, 1178, 1184, 1193,	\si@num@addpzero
1200, 1206, 1214, 1222, 1287,	<u>1647</u>
1441, 1449, 1470, 1503, 1513,	\si@num@addsign
1526, 1535, 1566, 1621, 1646,	1488, <u>1542</u>
1652, 1673, 1719, 1858, 1866,	\si@num@addtmp
2018, 2026, 2029, 2208, 2220,	<u>1803</u>
2232, 2306, 2315, 2351, 2355,	\si@num@addtmpa
2395, 2470, 2498, 2638, 2643,	1797, <u>1803</u>
2696, 2702, 2758, 2790, 2812,	\si@num@addtmpb
2833, 2874, 2963, 2988, 3927, 3928	1795, <u>1803</u>
\si@log@err ..	\si@num@addunit ...
124, 583, 645, 1245,	1998, 2002, <u>2007</u>
1389, 1444, 1616, 1626, 1775,	\si@num@ambig
1784, 1789, 2037, 2042, 2310,	<u>1272</u> , 1417–1419
2588, 2591, 2602, 2605, 2616,	\si@num@ambigertrue ..
2619, 3135, 3350, 3358, 3366, 4563	1337, <u>2648</u>
\si@log@inf	\si@num@arg
124, 958, 1036,	556, <u>1284</u> , 1296, 1303, 1310, 1389,
1043, 1507, 1517, 1521, 2593,	1441, 1450, 1470, 1504, 1514,
2607, 2621, 3250, 3262, 3503,	1527, 1535, 1567, 1616, 1622,
4000, 4007, 4092, 4093, 4473, 4474	1627, <u>1646</u> , 1652, 1776, 1785, 1790
\si@log@warn ..	\si@num@assign
124, 203, 227, 588,	<u>1542</u>
593, 962, 1303, 1310, 2185, 2885,	\si@num@checkerr
3265, 3281, 3296, 4192, 4228, 4240	1692, <u>1732</u>
\si@logminfalse	\si@num@cntdgt
230	<u>1808</u>
\si@logmintrue	\si@num@cntdigits
238	. 1755, 1757, <u>1808</u> , 1825, 1837, 1846
\si@lognonefalse	\si@num@dec
231	1710, <u>1980</u>
\si@lognonetrue	\si@num@decfmt
234	<u>1980</u>
\si@mathscelsius	\si@num@decimalhook <u>1680</u> , 1706, 2286
810	\si@num@delplusfalse
\si@mathsdegree	1498
<u>799</u> , 815, 923	\si@num@delplustrue
\si@mathsminute	1540
<u>799</u>	\si@num@digits
\si@mathsmu	608, <u>1582</u> , <u>1613</u>
<u>787</u> , 935, 4258	\si@num@dp ..
\si@mathsOmega	<u>640</u> , 1850, 1853, 1858,
<u>783</u> , 936	1859, 1866, 1922–1924, 1926,
\si@mathsringA	1927, 2091, 2092, 2127, 2128, 2448
<u>816</u> , 929	\si@num@err
\si@mathsrm ...	<u>1272</u> , 1293, 1352,
<u>1090</u> , 1120, 1186, 1208	1362, 1363, 1376, 1377, 1411,
\si@mathssecond	1697, 1743, 1751, 1821, 1834, 2000
<u>799</u>	\si@num@erropenfalse
\si@mathssf	1422
<u>1090</u> , 1173, 1195	\si@num@erropentru
\si@mathstt	1358
<u>1090</u> , 1180, 1202	\si@num@exp
\si@negcolour	557, <u>1272</u> ,
<u>773</u> , 1325	1289, 1301, 1309, 1328, 1330,
\si@newcmd	1335, 1361, 1364, 1366, 1387, 1406
<u>100</u>	\si@num@expsign
\si@newcommand	572, <u>1272</u> ,
<u>100</u>	1290, 1302, 1305, 1364, 1367, 1405
\si@newrobustcmd	\si@num@extra
<u>100</u> , 1217, 2014, 2580, 2584, 3003,	<u>1714</u>
3864, 3870, 3931, 3941, 4109, 4143	\si@num@fiint
\si@noload	<u>1958</u>
<u>722</u>	\si@num@finddigits
\si@num	1573, <u>1578</u>
1223, <u>1226</u> , 2000,	\si@num@finderr
2004, 2249, 2256, 2452, 4116, 4122	<u>1763</u>
\si@num@addexp	\si@num@findsign
1454, <u>1465</u>	1482, <u>1496</u>
\si@num@addmnt	\si@num@findxpart
1456, <u>1465</u>	1296, <u>1426</u>
\si@num@addmntexp	\si@num@five
<u>1465</u>	<u>1950</u>
	\si@num@fixdp
	1699, <u>1845</u>
	\si@num@fixpm
	555, 1227, <u>1251</u>
	\si@num@format
	1244, <u>1284</u>
	\si@num@gensign
	1478, <u>1481</u>

\si@num@ifextra	1685, 1702, 1709, 1714	\si@num@procerr	1412, 1997
\si@num@iffive	1946, 1950, 1985	\si@num@procnun	1326, 1327, 1568
\si@num@ifvalid	1243, 1256, 2084	\si@num@psterr	1745
\si@num@in	1272, 1288, 1298, 1432	\si@num@rev	1863
\si@num@int	1703, 1941	\si@num@reverse	1863
\si@num@intabfalse	1221	\si@num@rnd	1874, 1884
\si@num@intabtrue	2445	\si@num@rndpost	1884
\si@num@intfmt	1944, 1947, 1958	\si@num@rndpre	1884
\si@num@intsep	1960, 1963, 1966, 1973	\si@num@round	1851, 1863
\si@num@killsign	1531, 1538	\si@num@sepdigits	1607, 1610, 1681
\si@num@largeerr	1759, 1830	\si@num@seperr	1740, 1746, 1763
\si@num@lerr	1830	\si@num@sepmantexp	562, 1298, 1437
\si@num@mant	558, 1272, 1291,	\si@num@sepsign	563, 571, 1299, 1300, 1473
1307, 1331, 1332, 1341, 1388, 1396		\si@num@sepxpart	1424, 2001, 2309
\si@num@mantexp	1437	\si@num@serr	1818
\si@num@mantsign	564, 1272,	\si@num@sign	1484, 1494, 1496, 1541, 1565
1292, 1308, 1311, 1316, 1341, 1345		\si@num@signexpfalse	431
\si@num@movedigit	1830	\si@num@signexptrue	442, 446, 451, 456, 461
\si@num@mp	601, 1251	\si@num@signmantfalse	430
\si@num@nosign	1654	\si@num@signmanttrue	434, 438, 450, 455, 460
\si@num@nozero	1601, 1677	\si@num@smallerr	1761, 1818
\si@num@out	1272, 1340, 1409, 2471,	\si@num@unsign	1598, 1654
2474, 2482, 2544, 2546, 2549, 2553		\si@num@valid	1256
\si@num@pad	1854, 1857	\si@num@value	1485, 1495, 1496
\si@num@padleadfalse	392	\si@num@xpart	1272, 1294, 1423, 1430, 2006
\si@num@padleadtrue	396, 400, 412, 417, 422	\si@numaddn	369, 372, 1683
\si@num@padtrailfalse	393	\si@numcloseerr	369, 372, 1781
\si@num@padtrailtrue	404, 408, 413, 418, 423	\si@numdecimal	369, 374, 1328, 1614, 1618, 1822
\si@num@pd	1857	\si@numdigits	369, 374, 1392
\si@num@pm	600, 1251	\si@numexp	369, 373, 1363, 1442, 1446
\si@num@post	1613	\si@numextra	371, 1683, 1727
\si@num@postdec	604, 615, 616, 1346,	\si@numgobble	369, 373, 1440
1349, 1576, 1580, 1589, 1604,		\si@numopenerr	369, 372, 1773
1665, 1684, 1694, 1705, 1709,		\si@numprod	369, 374, 1427, 2676
1710, 1713, 1734, 1757, 1846,		\si@numsign	369, 373, 1499, 1500, 1625, 1628
1847, 1862, 1867, 1870, 1872,		\si@numtextmodefalse	300, 308, 316
1936, 1981, 1987, 1990, 1993, 1994		\si@numtextmodetrue	304, 312, 319
\si@num@posterr	1737, 1745	\si@numvalid	371, 1247, 1267, 2080, 2349
\si@num@postrnd	1863, 1914–1916	\si@obeyboldfalse	339
\si@num@pre	1613	\si@obeyboldtrue	345
\si@num@predec	603, 612, 613, 1345, 1576, 1579,	\si@obeyfamilyfalse	342
1584, 1599, 1603, 1665, 1678,		\si@obeyfamilytrue	348
1679, 1684, 1694, 1702, 1703,		\si@obeyitalicfalse	340
1713, 1868, 1869, 1871, 1904,		\si@obeyitalictrue	346
1907, 1942, 1948, 1974, 1976, 1977			
\si@num@preerr	1735, 1739		
\si@num@prepost	1613		
\si@num@prernd	1863, 1892–1894, 1905		

<code>\si@obeymodefalse</code>	341	<code>\si@sis@addtolocale</code>	<u>4151</u>
<code>\si@obeymodetrue</code>	347	<code>\si@sis@num</code>	<u>4108</u>
<code>\si@old@OHMfalse</code>	4002	<code>\si@sis@numstar</code>	<u>4108</u>
<code>\si@openerr</code>	<u>375</u> , 1356, 1376	<code>\si@sis@savefont</code> ..	<u>4134</u> , 4137–4142
<code>\si@openfrac</code>	<u>693</u> , 3168	<code>\si@siu@newprefix</code>	
<code>\si@opt@boolkey</code>	<u>4378</u> , 4385–4404, 4406–4411
... 197, 248, 249, 296, 327, 328,		<code>\si@siu@newunit</code>	4267,
336, 337, 375, 376, 378, 379, 383,		4299–4339, 4638, 4639, 4642–4645	
387, 463, 521, 522, 547, 640, 649–		<code>\si@siu@newunithook</code> ...	<u>4267</u> , <u>4568</u>
651, 657, 660, 693, 734, 735, 773, 820		<code>\si@siu@newunitx</code>	4340, 4347, 4349,
<code>\si@opt@choicekey</code>	<u>200</u> ,	4351, 4353, 4355–4357, 4359,	
228, 299, 307, 315, 321, 330, 338,		4361, 4363, 4364, 4366, 4369, 4640	
390, 428, 472, 501, 524, 618, 629,		<code>\si@siu@newunitxhook</code> ..	<u>4340</u> , <u>4568</u>
661, 682, 695, 736, 743, 750, 759, 774		<code>\si@siu@power</code> ..	<u>4267</u> , 4576, 4589, 4600
<code>\si@opt@cmdkey</code>	<u>193</u> , 554	<code>\si@siu@setup</code>	<u>4189</u>
<code>\si@opt@cmdkeys</code>	<u>193</u> ,	<code>\si@siu@unity</code>	<u>4368</u>
350, 351, 364, 365, 369, 377, 694,		<code>\si@slash</code>	<u>679</u> , 946
722, 768, 781, 783, 787, 799, 810, 816		<code>\si@slashfalse</code>	663
<code>\si@opt@compatkey</code>		<code>\si@slashtrue</code>	668
.....	<u>215</u> , 833–839, 841–880	<code>\si@str@chrstr</code>	<u>150</u>
<code>\si@opt@disablekey</code>	220,	<code>\si@str@ifchrstr</code>	
<u>223</u> , 264, 269, 275, 725, 728, 822		<u>150</u> , 184, 1267, 1427,
<code>\si@opt@key</code>	<u>190</u> ,	1440, 1442, 1499, 1500, 1614,	
227, 273, 352–363, 366–368, 545,		1625, 1727, 1773, 1781, 2349, 2676	
642, 723, 732, 769–772, 782, 784,		<code>\si@str@ifonlychrs</code>	
788, 801–803, 811, 817, 827–829, 832		... <u>171</u> , 643, 1328, 1665, 1678, 1695	
<code>\si@opt@proctform</code>	579, 580, <u>602</u>	<code>\si@str@onlychrs</code>	<u>171</u>
<code>\si@opt@xchoicekey</code>		<code>\si@svn@id</code>	<u>1</u>
... <u>204</u> , 278, 281, 283, 285, 288,		<code>\si@svn@ver</code>	<u>1</u>
291, 293, 380, 382, 384, 386, 427, 679		<code>\si@svn@version</code>	<u>1</u>
<code>\si@out</code> 946, 3120, 3182, <u>3300</u> , 3338, 4146		<code>\si@switchfalse</code>	
<code>\si@out@maths</code>	3316, <u>3321</u>	.. 152, 185, 561, 605, 1271, 1295,	
<code>\si@out@minus</code>	3325, <u>3333</u>	1297, 1581, 1666, 1716, 1764,	
<code>\si@out@num</code> 1223, 1415, 1418, 2240,		1873, 1899, 1921, 2327, 3488, 4154	
2473, 2482, 2502, 2510, 2543,		<code>\si@switchtrue</code> 165, 173, 1258, 1428,	
2546, 2552, 2562, <u>3335</u> , 4116, 4122		1452, 1623, 1664, 1727, 1779,	
<code>\si@out@numtrue</code>	3337	1902, 1929, 1934, 2350, 3492, 4156	
<code>\si@out@sp</code>	3322, 3323, <u>3332</u>	<code>\si@sym@celsius</code>	
<code>\si@out@text</code>	3314, <u>3321</u>	<u>911</u> , 3568, 3571, 4084, 4148
<code>\si@packagecheck</code>	<u>50</u>	<code>\si@sym@degree</code>	<u>911</u> , 2191,
<code>\si@per</code> <u>3001</u> , 4175, 4177, 4180,		2218, 2230, 2239, 2258, 3191,	
4181, 4183, 4575, 4588, 4599,		3203, 3738, 3741, 4086, 4149, 4150	
4608, 4612, 4616, 4629, 4632, 4635		<code>\si@sym@minute</code>	<u>911</u> ,
<code>\si@pm</code>	<u>463</u> , 1505, 1999	2192, 2216, 2228, 2259, 3744, 4260	
<code>\si@prefixbase</code>	386, 3123	<code>\si@sym@mu</code>	
<code>\si@prefixproduct</code>	<u>384</u> , 3120	.. <u>911</u> , 3192, 3204, 3517, 3520, 4085	
<code>\si@prefixsymbolicfalse</code>	4384	<code>\si@sym@Omega</code> ..	<u>911</u> , 3554, 3557, 4083
<code>\si@repeatunitsfalse</code> ...	683, 1357	<code>\si@sym@ringA</code> ..	<u>911</u> , 3193, 3205, 3719
<code>\si@repeatunitstrue</code>	687	<code>\si@sym@second</code>	
<code>\si@seperrfalse</code>	1428, 2308	.. <u>911</u> , 2193, 2214, 2260, 3745, 4261	
<code>\si@SI</code>	2582–2584, <u>2629</u>	<code>\si@symbol</code>	<u>903</u> , 911–917
<code>\si@sign</code>	<u>425</u> , 1565, 1566	<code>\si@tab@begin</code>	2313, 2317, <u>2321</u>

<code>\si@tab@begin@S</code>	2295, <u>2305</u>	<code>\si@tab@predim</code> <u>2491</u> , 2504, 2526, 2542
<code>\si@tab@begin@s</code>	2301, <u>2305</u>	<code>\si@tab@pretoks</code>
<code>\si@tab@end</code>	<u>2430</u> <u>2318</u> , 2325, 2369, 2415, 2419
<code>\si@tab@end@S</code>	2296,	<code>\si@tab@rfill</code>
2336, 2337, <u>2405</u> , 2431, 2433, 2434		<u>2405</u>
<code>\si@tab@end@s</code>	2302,	<code>\si@tab@rfill@S</code>
2378, 2379, <u>2405</u> , 2437, 2439, 2440		<u>523</u> , 2408
<code>\si@tab@expbox</code>		<code>\si@tab@rfill@s</code>
. <u>2454</u> , 2468, 2490, 2501, 2557, 2561		<u>630</u> , 2428
<code>\si@tab@expdim</code> .	<u>2491</u> , 2513, 2518,	<code>\si@tab@rfill@t</code>
2521, 2530, 2537, 2540, 2557, 2561		<u>619</u> , 2410
<code>\si@tab@expout</code>		<code>\si@tab@sepcorr</code>
. <u>1272</u> , <u>1340</u> , 2471, 2549, 2555, 2562	 2505, 2509, 2514, 2522, 2566
<code>\si@tab@exppostcnt</code>		<code>\si@tab@sp</code>
..... <u>548</u> , 2517, 2518, 2533		<u>2496</u> , 2499, 2516, 2577
<code>\si@tab@expprecnt</code>		<code>\si@tab@toks</code>
..... <u>548</u> , 2512, 2513, 2532		<u>2318</u> , 2324,
<code>\si@tab@expsignfalse</code>	560	2353, 2354, 2393, 2394, 2426, 2452
<code>\si@tab@expsigntrue</code>	574, 577	<code>\si@tab@unfixed</code>
<code>\si@tab@fixed</code>	2460, <u>2497</u>	2462, <u>2469</u>
<code>\si@tab@fixedfalse</code>	539, 543	<code>\si@tab@numalign</code>
<code>\si@tab@fixedtrue</code>	526	<u>523</u>
<code>\si@tab@format</code>	2453, 2458	<code>\si@tempa</code> ...
<code>\si@tab@gettok</code>	<u>2305</u> , 2328	<u>42</u> , 55, 56, 58, 59, 61,
<code>\si@tab@gettok@S</code>	2307, <u>2329</u>	64, 65, 68, 69, 153, 164, 174, 176,
<code>\si@tab@gettok@s</code>	2316, <u>2329</u>	201, 205, 233, 237, 241, 245, 259,
<code>\si@tab@lfill</code>	<u>2405</u>	264, 265, 303, 311, 318, 324, 333,
<code>\si@tab@lfill@S</code>	<u>523</u> , 2407	344, 395, 399, 403, 407, 411, 416,
<code>\si@tab@lfill@s</code>	<u>630</u> , 2424	421, 433, 437, 441, 445, 449, 454,
<code>\si@tab@lfill@t</code>	<u>619</u> , 2411	459, 477, 481, 485, 490, 495, 506,
<code>\si@tab@mantpostcnt</code> <u>548</u> , 581, 2448, 2506	510, 514, 518, 530, 534, 538, 542,
<code>\si@tab@mantprecnt</code> <u>548</u> , 582, 592, 2504		622, 626, 633, 637, 666, 672, 676,
<code>\si@tab@mantsignfalse</code>	559	686, 690, 699, 703, 707, 711, 715,
<code>\si@tab@mantsigntrue</code>	566, 569	719, 733, 740, 747, 755, 764, 778,
<code>\si@tab@midbox</code>		927, 928, 1018, 1023, 1026, 1030,
..... <u>2454</u> , 2467, 2475, 2478, 2500		1050, 1051, 1117, 1192, 1240,
<code>\si@tab@newline@S</code>	2331, 2430	1242–1244, 1315, 1316, 1342,
<code>\si@tab@newline@s</code>	2373, <u>2430</u>	1347, 1350, 1374, 1380, 1382,
<code>\si@tab@next</code>	<u>2329</u>	1385, 1398, 1399, 1403, 1404,
<code>\si@tab@numout</code>	2416, <u>2444</u>	1544, 1546, 1684, 1685, 1694,
<code>\si@tab@othertok</code>	<u>2329</u>	1695, 1765, 1770, 1831, 1835,
<code>\si@tab@out</code> ...	<u>1272</u> , <u>1340</u> , 2473, 2543	1843, 1876, 1878, 1880, 1891,
<code>\si@tab@postbox</code>	<u>2454</u> , 2467,	1895, 1913, 1917, 2031, 2033,
2481, 2483–2485, 2487, 2545, 2551		2036, 2067, 2069, 2099, 2104,
<code>\si@tab@postdim</code>		2107, 2111, 2115, 2138, 2141,
. <u>2491</u> , 2506, 2508, 2540, 2545, 2551		2143, 2145–2147, 2150, 2151,
<code>\si@tab@posttoks</code>		2156, 2162, 2172, 2174, 2180,
..... <u>2318</u> , 2326, 2367, 2417		2183, 2288, 2292, 2294, 2297,
<code>\si@tab@prebox</code>	<u>2454</u> , 2467,	2300, 2303, 2662, 2667, 2893,
2472, 2483, 2484, 2487, 2488, 2542		2895, 2903, 2905, 2921, 2924,
		2941, 2944, 2956, 2960, 2984,
		2987, 3039, 3041, 3048, 3051,
		3060, 3062, 3066, 3079, 3081,
		3197, 3198, 3269, 3271, 3274,
		3279, 3286, 3288, 3289, 3294,
		3371, 3373–3376, 3378, 3385,
		3388, 3465, 3466, 3474, 3476,
		3479, 3486, 3491, 3495, 3497,
		3498, 3924, 3957, 3958, 3968,
		3969, 3979, 3980, 3990, 3991,
		4190, 4268, 4271, 4272, 4279,
		4286, 4288, 4295, 4379, 4383,

4569, 4579, 4581, 4592, 4594, 4603
 \si@tempb
 42, 154, 155, 175, 184, 232, 233,
 236, 237, 240, 241, 244, 245, 263,
 266, 302, 303, 310, 311, 317, 318,
 323, 324, 332, 333, 343, 344, 394,
 395, 398, 399, 402, 403, 406, 407,
 410, 411, 415, 416, 420, 421, 432,
 433, 436, 437, 440, 441, 444, 445,
 448, 449, 453, 454, 458, 459, 476,
 477, 480, 481, 484, 485, 489, 490,
 494, 495, 505, 506, 509, 510, 513,
 514, 517, 518, 529, 530, 533, 534,
 537, 538, 541, 542, 621, 622, 625,
 626, 632, 633, 636, 637, 665, 666,
 671, 672, 675, 676, 685, 686, 689,
 690, 698, 699, 702, 703, 706, 707,
 710, 711, 714, 715, 718, 719, 739,
 740, 746, 747, 754, 755, 763, 764,
 777, 778, 1022, 1023, 1118, 1199,
 1545, 1546, 1689, 1691, 1741,
 1743, 1747, 1751, 1755, 1766,
 1769, 1823, 1825, 1827, 1835,
 1837, 1839, 1844, 2032, 2033,
 2035, 2036, 2068, 2069, 2663,
 2667, 3049, 3051, 3061, 3062,
 3065, 3066, 3080, 3081, 3270,
 3271, 3278, 3279, 3287, 3288,
 3293, 3294, 3355, 3373, 3375,
 3377, 3382, 3387, 3389, 3472,
 3476, 3489, 3491, 4269, 4272,
 4281, 4287, 4290, 4296, 4380,
 4384, 4571, 4574, 4583, 4586, 4597
 \si@tempboxa
 46, 1024, 1025, 1033, 1040, 2263,
 2265, 2267, 2268, 2270, 2272,
 2276, 2279, 2282, 2502, 2503,
 2507, 2508, 2510, 2511, 2519,
 2521, 2524, 2526, 2528, 2530,
 2535, 2537, 2576, 2579, 2686, 2687
 \si@tempboxb 46, 2266, 2274
 \si@tempboxc 46, 2271, 2274
 \si@tempboxd 46, 2273, 2277, 2280
 \si@tempc 42, 163, 164, 1690,
 1691, 1704, 1707, 1713, 3377–
 3379, 3388–3390, 3473, 3477,
 3479, 3481, 3485, 3495, 3497,
 4573, 4580, 4585, 4593, 4596, 4604
 \si@tempcnta 1756, 1758,
 1809, 1813, 1819, 1826, 1832,
 1838, 1848, 1850, 1853, 1859,
 1861, 1885, 1923, 1927, 1939,
 2442, 2532–2534, 2567, 2569,
 2573, 2579, 2656, 2660–2662,
 2673, 2677, 2930, 2934, 2936, 2937
 \si@tempcntb 1756, 1758, 1819, 1826,
 1832, 1838, 1895, 1897, 1900,
 1901, 1904, 1917, 1919, 1928,
 1932, 1933, 1936, 2442, 2936, 2937
 \si@tempdima 2098, 2101,
 2103, 2107, 2110, 2115, 2120,
 2131, 2133, 2136, 2138, 2142,
 2143, 2171, 2175, 2265, 2275,
 2276, 2283, 2491, 2503, 2504,
 2506, 2511, 2513, 2518, 2996, 2997
 \si@tempdimb
 2270, 2275, 2279, 2283, 2491
 \si@temptoks . 85, 2152, 2154, 2157,
 2159, 2163, 2165, 2168, 2868,
 2873, 2874, 2881, 2882, 3252, 3256
 \si@textcelsius 810, 3977
 \si@textdegree .. 799, 813, 922, 3988
 \si@textminute 799
 \si@textmodefalse 1081, 1087
 \si@textmodetrue 1079, 1085
 \si@textmu 787, 925, 3966, 4258
 \si@textOmega 783, 926, 3955
 \si@textstringA 816, 931
 \si@textrm 1090, 1122, 1188, 1210
 \si@textsecond 799
 \si@textsf 1090, 1175, 1197
 \si@texttt 1090, 1182, 1204
 \si@tothe 3227
 \si@unitcolour 734, 1113
 \si@unitmathsrms 350, 1104
 \si@unitmathssf 350, 1105
 \si@unitmathstt 350, 1106
 \si@unitsep 278, 2888, 3180
 \si@unitSIdef 3937
 \si@unitspace 278, 3179
 \si@unittextmodefalse 301, 309, 322
 \si@unittextmodetrue . 305, 313, 325
 \si@unittextrm 364, 1107
 \si@unittextsf 364, 1108
 \si@unittexttt 364, 1109
 \si@unt@addprefix 2912, 2917
 \si@unt@addstack 3069, 3076
 \si@unt@addtostack
 2837, 2888, 2918, 2990, 3036
 \si@unt@addvalsep 2708
 \si@unt@addvaluesep 2700, 2708, 2864
 \si@unt@checkstack
 2895, 2905, 2986,
 3041, 3050, 3059, 3074, 3080,
 3085, 3089, 3093–3095, 3097, 3105

\si@unt@cntx	2674, 2675, 2680	\si@unt@power	2823, <u>2939</u> , 3236
\si@unt@countprefix ...	2914, <u>2919</u>	\si@unt@powerdim	2739,
\si@unt@countx	2658, <u>2672</u>		2950, 2963, 2966, 2975, 2983,
\si@unt@defpower			2989, 2991, 2994, 2996–2999, 3034
 2615, 2620, 2622, 2625, <u>2811</u>	\si@unt@prefix	<u>2806</u> , <u>2909</u>
\si@unt@defprefix		\si@unt@prefixcnt	
 2601, 2606, 2608, 2611, <u>2787</u>	 2742, <u>2919</u> , 3118, 3124
\si@unt@defunit		\si@unt@prefixout	
 2587, 2592, 2594, 2597, <u>2757</u>	 3117, 3127, 3141, 3144, 3153
\si@unt@depthcnt	2705,	\si@unt@preplussp	3044, <u>3047</u>
	<u>2722</u> , 2829, 2832, 2833, 2842, 2843	\si@unt@prepowerfalse .	2737, <u>2968</u>
\si@unt@final	2707, <u>2750</u> , 2844	\si@unt@prepowertrue	2957
\si@unt@first	2848, <u>2853</u>	\si@unt@printunit	
\si@unt@firstfalse	2865		. 2011, 2639, 2655, 2665, 2669, <u>2693</u>
\si@unt@firstorsecond		\si@unt@recip	<u>3027</u>
 2835, <u>2846</u> , 2910, 2947, 3018	\si@unt@reciptest	2907, <u>3027</u>
\si@unt@firstttrue	2734	\si@unt@second	2850, <u>2883</u>
\si@unt@fracout	3101, <u>3130</u>	\si@unt@setopts	2859, <u>2866</u>
\si@unt@holdspace	3067, <u>3076</u>	\si@unt@setSIopts	<u>2866</u>
\si@unt@holdstacka	2746, <u>3076</u>	\si@unt@SIopts	<u>2629</u> , 2877, 2881
\si@unt@holdstackb	2747, <u>3076</u>	\si@unt@spacecheck	2897, <u>2902</u>
\si@unt@ifliteral .	<u>2682</u> , 2695, 2836	\si@unt@spaceout	
\si@unt@incnt	3063, <u>3072</u>	 3115, 3128, 3145, 3154
\si@unt@init	2704, <u>2722</u> , 2830	\si@unt@spstack ...	2719, <u>2722</u> , 3116
\si@unt@invpower	2953, <u>2995</u>	\si@unt@stack	3054, <u>3057</u>
\si@unt@invprefix	<u>2919</u>	\si@unt@stacka <u>2722</u> , 3129, 3132, 3155	
\si@unt@lastadda	2748, <u>3036</u>	\si@unt@stackb	<u>2722</u> , 3082,
\si@unt@lastaddb	2749, <u>3036</u>		3133, 3142, 3146, 3150, 3155, 3168
\si@unt@litoutfalse	2732	\si@unt@stackout	2752, <u>3098</u>
\si@unt@litouttrue 2699, 3099, 3949		\si@unt@stackpower 2961, <u>2967</u> , 3035	
\si@unt@litpower ..	2821, <u>2938</u> , 3234	\si@unt@stackvalsep ...	<u>2708</u> , 3122
\si@unt@litprefixfalse	2733	\si@unt@stkpower ..	2839, 2891, <u>2967</u>
\si@unt@litprefixtrue	2788	\si@unt@stkpw	<u>2967</u>
\si@unt@littesttrue	2685	\si@unt@sym 3191–3193, 3203–3205, <u>3208</u>	
\si@unt@litvalsep	<u>2708</u>	\si@unt@third	2751, <u>2883</u>
\si@unt@nonlatin	3175, <u>3185</u>	\si@unt@unit	2782, <u>2827</u>
\si@unt@noopt	2767, <u>2770</u>	\si@unt@unitarg 2009, 2011, <u>2628</u> , 2641	
\si@unt@normout ...	3103, <u>3126</u> , 3151	\si@unt@unitcnta	2722
\si@unt@notabg	<u>3158</u>	\si@unt@unitcntb	2722, 3159
\si@unt@notambig	3131, <u>3158</u>	\si@unt@unithook ..	<u>2827</u> , 2857, 3949
\si@unt@numfalse	652, 2635	\si@unt@withopt	2765, <u>2770</u>
\si@unt@numtrue 654, 2008, 2652, 2856		\si@valuecolour	<u>734</u> , 1102, 2465
\si@unt@opt	2772–2774, <u>2775</u>	\si@valuemathsrn	<u>350</u> , 1093
\si@unt@out 940, 941, 945, 947, 949,		\si@valuemathssf	<u>350</u> , 1094
	950, 953, 954, 2686, 2701, 3129, <u>3170</u>	\si@valuemathstt	<u>350</u> , 1095
\si@unt@per	<u>3001</u>	\si@valuesep	<u>278</u> ,
\si@unt@perfalse ..	2735, 2971, 3021		1024, 2719, 2721, 3877, 3923, 4171
\si@unt@perseentfalse ..	2736, 2972	\si@valuetextrm	<u>364</u> , 1096
\si@unt@perseenttrue ...	2900, 4185	\si@valuetextsf	<u>364</u> , 1097
\si@unt@pertrue ...	3024, 4185, 4188	\si@valuetexttt	<u>364</u> , 1098

<code>\si@xargdef</code>	100	<code>\Square</code>	
<code>\si@xifmtarg</code>	92		12 , 3222 , 3638 , 3639 , 3641 , 4033 –
<code>\si@xspacefalse</code>	2636 , 3943		4035 , 4180 , 4495 – 4497 , 4507 ,
<code>\SIdecimalsign</code>	4130		4519 , 4521 , 4523 , 4525 , 4527 , 4640
<code>\SIdecimalsymbol</code>	4130	<code>\square</code>	4218
<code>\SIdefaultMfam</code>	4140	<code>\squarecentimeter</code>	4033
<code>\SIdefaultNfam</code>	4140	<code>\squarecentimetre</code>	16 , 3638
<code>\SIdefaultTfam</code>	4140	<code>\squared</code>	12 , 3222 ,
<code>\siemens</code>	14 , 3553 , 3618		3640 , 3686 , 4186 , 4358 , 4452 , 4457
<code>\siemensbase</code>	4455	<code>\squarekilometer</code>	4033
<code>\sievert</code>	14 , 3553 , 3660	<code>\squarekilometre</code>	16 , 3638
<code>\sievertbase</code>	4455	<code>\squaremeter</code>	4033
<code>sign (option)</code>	25	<code>\Squaremetre</code>	4637
<code>\SIgroupfourfalse</code>	4125	<code>\squaremetre</code>	
<code>\SIgroupfourtrue</code>	4125	 16 , 3638 , 4348 , 4349 , 4354 ,
<code>\SImathrm</code>	4137		4360 , 4370 , 4376 , 4415 , 4427 ,
<code>\SImathsf</code>	4137		4440 , 4445 , 4446 , 4449 , 4454 ,
<code>\SImathtt</code>	4137		4459 , 4463 , 4465 , 4466 , 4468 , 4637
<code>\SIobeyboldfalse</code>	4106	<code>\squaren</code>	4218
<code>\SIobeyboldtrue</code>	4106	<code>\ssquare</code>	12 , 3222 , 4273
<code>\SIproductsign</code>	4130	<code>\steradian</code>	14 , 3574 , 4360 , 4431
<code>\SIsetup</code>	4189	<code>\steradianbase</code>	4438
<code>\sisetup</code>	4 , 21 , 189 , 218 ,	<code>stickyper (option)</code>	29
	251 , 352 – 363 , 366 – 368 , 545 , 598 ,	<code>strict (option)</code>	32
	769 – 772 , 782 , 784 , 788 , 801 – 803 ,	<code>strictarc (option)</code>	26
	811 , 817 , 1065 , 1067 , 1219 , 2016 ,		
	2055 , 2118 , 2255 , 2323 , 2633 ,		
	2873 , 2882 , 2925 , 3243 , 3244 ,		
	3247 , 3256 , 3392 , 3810 , 3819 ,		
	3828 , 3838 , 3853 , 3856 , 3859 ,		
	3862 , 3874 , 3891 , 3893 , 3896 ,		
	3899 , 3902 , 3905 , 3908 , 3911 ,		
	3914 , 3917 , 3920 , 3939 , 3950 ,		
	3964 , 3975 , 3986 , 3997 , 4099 ,		
	4106 , 4107 , 4115 , 4120 , 4121 ,		
	4125 – 4133 , 4136 , 4145 , 4151 ,		
	4152 , 4168 , 4175 , 4177 , 4180 ,		
	4181 , 4183 , 4185 , 4188 , 4197 ,		
	4200 , 4203 , 4206 , 4209 , 4212 ,		
	4215 , 4251 , 4254 , 4532 , 4534 –		
	4536 , 4538 , 4541 , 4544 , 4547 ,		
	4550 , 4553 , 4556 , 4574 , 4586 ,		
	4597 , 4607 , 4610 , 4614 , 4618 , 4620		
<code>\SIstyle</code>	4151		
<code>\SIstyleToLang</code>	4151		
<code>\SIthousandsep</code>	4130		
<code>\SIunitdot</code>	4127		
<code>\SIunitsep</code>	4127		
<code>\SIunitspace</code>	4127		
<code>slash (option)</code>	28		

T

<code>tabalign (option)</code>	27
<code>tabalignexp (option)</code>	27
<code>tabautofit (option)</code>	28
<code>tabformat (option)</code>	27
<code>tabnumalign (option)</code>	26
<code>tabtextalign (option)</code>	27
<code>tabunitalign (option)</code>	27
<code>\tebi</code>	19 , 3796
<code>\tera</code>	14 , 3526 , 3649 ,
	3655 , 3674 , 3700 , 4503 , 4517 , 4527
<code>\teraelectronvolt</code>	17 , 3642
<code>\terahertz</code>	17 , 3651
<code>\tesla</code>	14 , 3553
<code>\teslabase</code>	4455
<code>\TeV</code>	17 , 3694
<code>\TeVoverc</code>	4508
<code>\TeVovercsq</code>	4518
<code>textcelsius (option)</code>	30
<code>textdegree (option)</code>	30
<code>textminute (option)</code>	30
<code>textmode (option)</code>	22
<code>textmu (option)</code>	30
<code>textOmega (option)</code>	30
<code>textringA (option)</code>	30
<code>textrm (option)</code>	22

textsecond (option)	30		
textsf (option)	22		
texttt (option)	22		
\THz	17, 3669		
tightpm (option)	24		
\TinveV	4498		
\ton	4005, 4260, 4622		
\tonne	15, 3731		
\torr	19, 3761		
\tothe	12, 2662, 3227, 3951, 4179		
\tothe@opt@si	3227		
trapambigerr (option)	24		
trapambigfrac (option)	29		
U			
\uBar	4065		
\UFrac	4628		
\Ufrac	4628		
\ufrac	4628		
\unit	3864, 3934, 4245		
\unita	4245		
unitcolor (option)	31		
unitcolour (option)	31		
\unitfrac	3870, 3945		
unitmathsrms (option)	22		
unitmathssf (option)	22		
unitmathstt (option)	22		
unitmode (option)	22		
\unitsep	3947		
unitsep (option)	23		
\unitSIDef	3937		
\unitsignonly	3930		
unitspace (option)	23		
\unitsuperscript	3947		
unittextrm (option)	22		
unittextsf (option)	22		
unittexttt (option)	22		
\unittimes	3947		
\unitvaluesep	3922, 3933, 3944		
\unskip	2392, 2427		
\usk	4175, 4436		
V			
valuecolor (option)	31		
valuecolour (option)	31		
valuemathrm (option)	22		
valuemathsf (option)	22		
valuemathsrms (option)	22		
valuemathssf (option)	22		
valuemathstt (option)	22		
valuemathtt (option)	22		
valuemode (option)	22		
valuesep (option)	23		
valuetextrm (option)	22		
valuetextsf (option)	22		
valuetexttt (option)	22		
\volt	14, 3553, 3608, 3609, 3680, 3681, 4041, 4042, 4423–4425		
\voltbase	4438		
W			
\watt	14, 3553, 3610–3612, 3650, 3701, 4359, 4363, 4422		
\wattbase	4438		
\weber	14, 3553, 4428, 4429		
\weberbase	4455		
X			
\XeTeXrevision	3186		
xspace (option)	29		
Y			
\yb	20, 3785		
\yobi	3796		
\yocto .	14, 3507, 3784, 3790, 4487, 4493		
\yoctobarn	20, 3779		
\yotta	14, 3526		
Z			
\zb	20, 3785		
\zebi	3796		
\zepto .	14, 3507, 3783, 3789, 4486, 4492		
\zeptobarn	20, 3779		
\zetta	14, 3526		

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.