

siunitx — A comprehensive (SI) units package*

Joseph Wright[†]

Released 2008/08/14

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	12
I Introduction	4	7.1 Literal units	12
		7.2 The unit interpreter . .	13
		7.3 Powers of units	13
		7.4 Units with no values . .	14
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	15
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 v1.0i, last revised 2008/08/14.

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

8.4	Astronomy (<code>astro</code>) . . .	22	14.8	Adding items after the last column of a tabular .	41
9	Font control	22	15	Reporting a problem	42
10	Package options	22	16	Feature requests	43
10.1	Font family and style .	23	17	Acknowledgements	43
10.2	Spacing and separators	24			
10.3	Number formatting . . .	25	III	Correct application of (SI) units	45
10.4	Angle formatting	27	18	Background	45
10.5	Tabular material	28	19	Units	45
10.6	Units	30	19.1	SI base units	45
10.7	Symbols	31	19.2	SI derived units	45
10.8	Colour	32	19.3	SI prefixes	46
10.9	International support .	32	19.4	Other units	46
10.10	Package control	33	20	Units and values in print	47
10.11	Back-compatibility options	33	20.1	Mathematical meaning .	47
10.12	Summary of all options	33	20.2	Unit multiplication and division	47
11	Emulation of other packages	37	20.3	Repeating units	48
12	Configuration files	37	20.4	Clarity in writing values of quantities	48
13	Common questions	38	20.5	Graphs and tables . . .	48
13.1	Why do I need <code>\per</code> more than once?	38	IV	Implementation	51
13.2	Why is the order of my units changed?	38	21	Main package	51
13.3	Why are compound units not recommended outside of <code>\SI/\si</code> ? . .	38	21.1	Setup code	51
13.4	How do I set superscripts to use lining numbers?	38	21.2	Logging	55
13.5	Why do most of the examples use $\text{J mol}^{-1} \text{K}^{-1}$? .	39	21.3	String comparison . . .	55
13.6	What can <code>numprint</code> do that <code>siunitx</code> cannot? . . .	39	21.4	Option handling	56
14	Tricks and known issues	39	21.5	Compatibility options .	74
14.1	Ensuring maths mode .	39	21.6	Constants	75
14.2	Using . and fixed spaces in units	39	21.7	Symbols	76
14.3	Passing unprocessed digits through an <code>S</code> column	40	21.8	Handling fractions . . .	77
14.4	Limitations of <code>\mathrm</code>	40	21.9	Font control	79
14.5	Entire document in sans serif font	41	21.10	Formatting numbers . .	84
14.6	Effects of emulation . . .	41	21.11	Formatting angles . . .	105
14.7	Centring columns on non-decimal input	41	21.12	Tabular material	111
			21.13	Units	119
			21.14	Locales	138
			21.15	Output routine	140
			21.16	Finalisation	141

22 Loadable modules	145	24.2 United States	154
22.1 Multiple prefixes	146	24.3 Germany	155
22.2 Derived units with specific names	146	24.4 South Africa	155
22.3 Units with prefixes	147	25 Emulation code	155
22.4 Abbreviated units	150	25.1 units	155
22.5 Additional (temporary) SI units	151	25.2 unitsdef	156
22.6 Units accepted for use with SI	152	25.3 Slstyle	162
22.7 Units based on physical measurements	152	25.4 Slunits	164
		25.5 hepunits	171
		25.6 fancynum	172
		25.7 fancyunits	173
23 Additional configurations	152	V Notes	176
23.1 Synthetic chemistry	152	26 Change History	176
23.2 High-energy physics	153	27 Index	176
23.3 Astronomy	153	28 References	195
23.4 Binary units	154		
24 Loadable locales	154		
24.1 United Kingdom	154		

Part I

Introduction

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

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

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

¹General Conference on Weights and Measures.

²International System of Units.

Part II

Using the siunitx package

1 For the impatient

siunitx provides the user macros:

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

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

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

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

10 g
 23.4 g cm³
 1 × 10³⁴
 1°2'3''
 16.7 m s⁻¹
 30 × 10³ Hz
 1.2 mm × 3.56 mm × 9.2 mm
 −4.5 cm
 J mol⁻¹ K⁻¹
 $\frac{\text{J}}{\text{mol K}}$
 1.2346
 9.8000

Heading
1.3
134.2
3.56
74.7

```
\SI{10}{\gram}\\
\SI{23.4}{g.cm^3}\\
\num{1e34}\\
\ang{1;2;3}\\
\emph{\SI{16,7}{\metre\per\second}}\\
\textbf{\SI{30e3}{\Hz}}\\
\SI{1.2 x 3.56 x 9.2}{\milli\metre}\\
\sisetup{obeyall}
\textbf{\SI{-4.5}{\cm}}\\
\si{\joule\per\mole\per\kelvin}\\
\si[per=frac]{\joule\per\mole\per\kelvin}\\
\num[dp=4]{1.23456}\\
\num[dp=4]{9.8}\\
\begin{tabular}{S[tabformat=3.2]}
\toprule
{Heading}\\
\midrule
1.3 \\
134.2 \\
3.56 \\
74.7 \\
\bottomrule
\end{tabular}
```

2 Requirements

siunitx requires a reasonably up to date \TeX system. The package requires $\varepsilon\text{-}\TeX$ -extensions, which should be available on most systems.³ The following packages are also needed:

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

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

To use the `fraction=sfrac` option, the `xfrac` package is needed. This needs various experimental \LaTeX 3 packages. As a result, siunitx does not load `xfrac`. If you want to use `fraction=sfrac`, *you* need to load `xfrac` in your preamble before siunitx.⁴ If the package is not loaded, `fraction=sfrac` falls back on a `nicefrac`-like method. The interested user should look at the `xfrac` documentation for reasons this might not be ideal.⁵

3 Loading the package

siunitx is loaded by the usual \LaTeX method.

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

As is shown in the example, the package can be loaded with one or more options, using the key–value system. The full range of package options are described in Section 10; some options are described in the along with the appropriate user macros. Most of the user macros accept the same key–value settings as an optional argument.

4 Numbers

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

1	123	1234	12	345	<code>\num{1}</code>	<code>\num{123}</code>	<code>\num{1234}</code>	<code>\num{12345}\</code>
0.1	0.123	0.1234	0.123	45	<code>\num{0.1}</code>	<code>\num{0.123}</code>	<code>\num{0,1234}</code>	<code>\num{.12345}\</code>
1×10^{10}	3.45×10^{-4}	-10^{10}			<code>\num{1e10}</code>	<code>\num{3.45d-4}</code>	<code>\num{-e10}</code>	

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

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

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

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

$1.234(5) = 1.234 \pm 0.005$
 $\$ \backslash \text{num}\{1.234(5)\} = \backslash \text{num}[\text{seperr}]\{1.234(5)\} \$ \backslash \backslash$
 $1.234(5) \times 10^6 = (1.234 \pm 0.005) \times 10^6$
 $\$ \backslash \text{num}\{1.234(5)\text{e}6\} = \backslash \text{num}[\text{seperr}]\{1.234(5)\text{e}6\} \$$

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

$m_e = 9.1093897(54) \times 10^{-31} \text{ kg}$
 $\$ \text{m}_{\{\backslash \text{mathrm}\{e\}\}} = \backslash \text{SI}\{9.1093897(54)\text{e-}31\}\{\backslash \text{kg}\} \$ \backslash \backslash$
 $m_e = (9.1093897 \pm 0.0000054) \times 10^{-31} \text{ kg}$
 $\$ \text{m}_{\{\backslash \text{mathrm}\{e\}\}} = \backslash \text{SI}[\text{seperr}]\{9.1093897(54)\text{e-}31\}\{\backslash \text{kg}\} \$$

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

$1\,234\,567\,890$
 $1\,234\,567\,890$
 $\backslash \text{num}\{1234567890\}$
 $\backslash \text{num}[\text{mode}=\text{text}]\{1234567890\}$

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

$1\,1.0$
 $\backslash \text{num}\{1.\}$
 $\backslash \text{num}[\text{padnumber=all}]\{1.\} \backslash \backslash$
 $2+2$
 $\backslash \text{num}\{2\}$
 $\backslash \text{num}[\text{addsign=all}]\{2\} \backslash \backslash$
 $3 \times 10^4 + 3 \times 10^4 + 3 \times 10^{+4}$
 $\backslash \text{num}\{3\text{e}4\}$
 $\backslash \text{num}[\text{addsign=mant}]\{3\text{e}4\}$
 $\backslash \text{num}[\text{addsign=all}]\{3\text{e}4\} \backslash \backslash$
 $0.5\,5$
 $\backslash \text{num}\{.5\}$
 $\backslash \text{num}[\text{padnumber=none}]\{.5\}$

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

$1234\,1234$
 $\backslash \text{num}\{1234\}$
 $\backslash \text{num}[\text{sepfour=true}]\{1234\} \backslash \backslash$
 $12\,345\,12,345$
 $\backslash \text{num}\{12345\}$
 $\backslash \text{num}[\text{digitsep=comma}]\{12345\} \backslash \backslash$
 12345
 $\backslash \text{num}[\text{digitsep=none}]\{12345\}$

The formatting of exponents is also customisable.

$1 \times 10^{10} \, 1 \cdot 10^{10}$
 $\backslash \text{num}\{1\text{e}10\}$
 $\backslash \text{num}[\text{expproduct=cdot}]\{1\text{e}10\} \backslash \backslash$
 $2 \times 10^{20} \, 2 \times 5^{20}$
 $\backslash \text{num}\{2\text{e}20\}$
 $\backslash \text{num}[\text{expbase=5}]\{2\text{e}20\} \backslash \backslash$
 $3 \times 10^{30} \, 3 \times 10^{30}$
 $\backslash \text{num}\{3\text{e}30\}$
 $\backslash \text{num}[\text{expproduct=tighttimes}]\{3\text{e}30\}$

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

-1
 $\backslash \text{num}\{-1\} \backslash \backslash$
 -2
 $\backslash \text{setup}\{\text{colourneg}\}$
 $\backslash \text{num}\{-2\} \backslash \backslash$
 3×10^{-3}
 $\backslash \text{num}\{3\text{e-}3\} \backslash \backslash$
 -4
 $\backslash \text{num}[\text{negcolour=blue}]\{-4\}$

⁶This is common in chemical crystallography, for example.

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

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

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

5 Tabular material

5.1 Aligning numbers

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

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

```
\begin{table}
  \caption{Behaviour of \texttt{S} column type}
  \label{tab:default}}
  \centering
  \begin{tabular}{%
    S%
    S[tabnumalign=centre,tabformat=2.4]%
    S[tabnumalign=right,tabformat=2.4]%
    S[tabnumalign=centre,tabformat=2.4,decimalsymbol=comma]}
    \toprule
    {Some Values} & {Some Values} & {Some Values} & {Some Values} \\
    \midrule
    2.3456 & 2.3456 & 2.3456 & 2.3456 \\
    34.2345 & 34.2345 & 34.2345 & 34.2345 \\
    56.7835 & 56.7835 & 56.7835 & 56.7835 \\
    90.473 & 90.473 & 90.473 & 90.473 \\
    \bottomrule
  \end{tabular}
\end{table}
```

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

Table 1: Behaviour of S column type

Some Values	Some Values	Some Values	Some Values
2.3456	2.3456	2.3456	2,3456
34.2345	34.2345	34.2345	34,2345
56.7835	56.7835	56.7835	56,7835
90.473	90.473	90.473	90,473

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

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

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

Table 2: Exponents in tables

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

Table 3: Number and units in tables

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

5.2 Columns of units

As a complement to the `S` column, `siunitx` also provides a second column type, `s`. This is intended for producing columns of units. The letters chosen are intended to be similar to `\SI` and `\si`, respectively. The alignment of material in `s` columns is governed by the `tabunitalign` option.

```
\begin{table}
\centering
\caption{Number and units in tables}
\label{tab:num-unit}
\begin{tabular}{%
  S[tabformat=1.2e-1,tabnumalign=centre]%
  s[tabunitalign=left]}
\toprule
{Value} & \multicolumn{1}{c}{Unit} \\
\midrule
2.16e-5 & \metre\squared\per\second \\
2.83e-6 & \metre\squared\per\second \\
7.39e3 & \pascal\cubic\metre\per\mole \\
1.0e5 & \pascal \\
\bottomrule
\end{tabular}
\end{table}
```

As the `\si` macro can take literal or macro input, the `s` column cannot validate the input. *Everything* in an `s` column is therefore passed to the `\si` macro for processing. To prevent this, you have to use `\multicolumn`, as is shown in [Table 4](#). Notice that the braces do not prevent processing and colouring of the cell contents.

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

Table 4: The `s` column processes everything

Unit	Unit
m^3	m^3
kg	kg

```

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

```

6 Angles

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

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

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

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

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

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

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

1.2° 1'2
 1°2.3' 1°2'3
 1°2'3.4'' 1°2'3''4

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

7 Units and values

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

1.23 J mol ⁻¹ K ⁻¹	<code>\SI[mode=text]{1.23}{J.mol^{-1}.K^{-1}}\</code>
0.23 × 10 ⁷ cd	<code>\SI{.23e7}{\candela}\</code>
£1.99/kg	<code>\SI[per=slash]{1.99}{\pounds\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), ° (degree) and Å (ring-A).⁹

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

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

7.2 The unit interpreter

The second operation mode for the `\SI` macro is based on the behaviour of Slunits. Here, each unit, SI multiple prefix and power is given a macro name. These are entered in a method very similar to the reading of the unit name in English.

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

On its own, this is very similar to Slunits, and is less convenient than the direct input method.¹⁰ However, the package allows you to define new unit macros; a large number of pre-defined abbreviations are also supplied. More importantly, by defining macros for units, instead of literal values, new functionality is made available. Units may be re-defined to give different output, and handling of reciprocal values can be altered.

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

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

7.3 Powers of units

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

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

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

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

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

`\raiseto`

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

used “in line” to produce a power. As follows from standard English usage, this comes after the unit. `\raiseto` achieves the same, but is used *before* a unit to add a power *after*.¹¹

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

7.4 Units with no values

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

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

7.5 Free-standing units

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

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

7.6 Pre-defined units, prefixes and powers

`\metre` The package always defines the seven base SI units, irrespective of any package options given (Table 5). The kilogram is notable as by default it is a *base* unit with a prefix. Thus, when the package is loaded with the option `load={}`, `\kilo` and `\gram` are not defined. As `metre` is often spelled as “meter” in the US, the macro `\meter` is provided in addition to the `\metre` macro.¹³

By default, a number of additional definitions are created by the package. These are controlled by the `load` and `noload` options. Unless specifically requested with the option `noload=prefix`, `siunitx` defines the standard prefixes for powers of ten (Table 6). This leads to the redefinition of `\kilogram` as `\kilo\gram`. The macro `\deka` is provided, as this is used as an alias for `\deca` in some places. The package also defines a number of derived SI units

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

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

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

Table 5: The seven base SI units

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

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

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

which have assigned names and symbols (Table 7). Note that `\Gray` is capitalised to avoid a name clash with the `pstricks` package.¹⁴

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

7.7 Prefixed and abbreviated units

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

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

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

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

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

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

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

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

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

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

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

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

Unit	Macro	Symbol	Abbreviation
<i>Masses</i>			
kilogram	\kilogram	kg	\kg
femtogram	\femtogram	fg	\fg
picogram	\picogram	pg	\pg
nanogram	\nanogram	ng	\nanog
microgram	\microgram	μg	\micg
milligram	\milligram	mg	\mg
atomic mass	\atomicmass	u	\amu
<i>Lengths</i>			
picometre	\picometre	pm	\picom
nanometre	\nanometre	nm	\nm
micrometre	\micrometre	μm	\micm
millimetre	\millimetre	mm	\mm
centimetre	\centimetre	cm	\cm
decimetre	\decimetre	dm	\dm
kilometre	\kilometre	km	\km
<i>Times</i>			
second	\second	s	\Sec
attosecond	\attosecond	as	\as
femtosecond	\femtosecond	fs	\fs
picosecond	\picosecond	ps	\ps
nanosecond	\nanosecond	ns	\ns
microsecond	\microsecond	μs	\mics
millisecond	\millisecond	ms	\ms

Continued on next page

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

Continued on next page

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

7.8 Defining new units

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

```

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

```

As with the \LaTeX commands `\newcommand`, *etc.*, the choice of `\newunit`, `\renewunit` or `\provideunit` depends on the presence of an existing definition. While `\newunit` should be used when a unit has not been previously defined, `\renewunit` will issue a warning if the named unit does not already exist. `\provideunit` defines the unit if it does not exist, and otherwise does nothing

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

at all. The same behaviour is seen with `\providepower` and `\provideprefix` (*vide infra*).

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

$10\text{m}\pi^{-2}$ `\newunit{\SIpi}{\ensuremath{\mathnormal{\pi}}}`
`\SI{10}{\metre\per\SIpi\squared}`

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

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

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

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

8 Specialist units

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

8.1 Binary units (binary)

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

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

8.2 Synthetic chemistry (synchem)

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

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

Table 12: Binary prefixes (alsoload=binary)

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

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

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

can be given as either “m” or “M”. The later is obviously easily confused with the sign for the prefix mega. By default, siunitx uses the UK default of a small-caps symbol. The \dalton unit is defined here as this name is not recognised by the various international bodies: the symbol u is preferred.

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

8.3 High-energy physics (hep)

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

The hep option defines two units which are slightly unusual. \cflight gives c, which is recognised as a unit when used in the appropriate circumstances. The

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

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

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

8.4 Astronomy (astro)

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

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

9 Font control

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

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

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

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

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

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

10 Package options

`\sisetup` The “native” options for the package are all given using the key–value method.

Most of the package options can be given both when loading the package and at any point in the document. This is achieved using the `\sisetup` macro.

The package options take a number of different forms.

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

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

10.1 Font family and style

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

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

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

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

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

The font commands used by the package to achieve the above are all available for user modification. The options `mathsrn`, `mathssf` and `mathstt` hold the command sequences used in maths mode,²⁰ while `textrn`, `textsf` and `texttt` do the same for text mode. By default, these contain the obvious command names, for example `mathsrn=mathrm` and `texttt=ttfamily`. However, they can be set at will: the macro names indicate the nature of the surrounding text detected. For example, the value of `mathssf` is used in maths mode when the surrounding text is sans serif.

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

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

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

10.2 Spacing and separators

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

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

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

10.3 Number formatting

numdigits	There are two groups of options for formatting numbers. The first group all begin with “num”, and take literal values used by the package to parse numbers. numdigits contains the valid number symbols (0123456789), with numdecimal containing the decimal markers (. , ,). As in the numprint package, numexp (the list of exponent markers) recognises deDE as valid by default. numsign contains the sign markers for numbers (+-\pm\mp). numaddn and numgobble both control which other characters do not give an error when present in a number. numaddn contains valid characters which should be included in the final output “as is”, whereas numgobble lists the characters that are completely ignored. In all cases, the content of the options is a simple string, for example numdigits=1234567890.
decimalsymbol	The second group of number options control the output of numbers after parsing. The symbol used by siunitx as a decimal marker is set by the decimalsymbol option, which can take a list of choices or a literal. The valid choices here are fullstop , comma , cdot and tightcdot . ²¹ Notice that this does not have to agree with the input marker. The other separator for numerical output is the division of digits into groups of three. The result is dependant on two options. The previously-described digitsep option controls the spacing added between groups of three numbers. For numbers consisting of exactly four digits, the sepfour Boolean option controls whether separation occurs in these cases. The default is false .
digitsep	
sepfour	
1234	<code>\num{1234}\</code>
1 234	<code>\num[sepfour]{1234}</code>
seperr	For numbers given with an error [e.g. 1.23(4)], the package can separate out the error part, to give for example 1.23 ± 0.04. This behaviour is activated by the seperr option, and requires that numopenerr and numcloseerr contain the left- and right-hand delimiters for the error [defaults numopenerr=(and numcloseerr=)].
numopenerr	
numcloseerr	
1.234 ± 0.005	<code>\sisetup{seperr}\</code>
(1.234 ± 0.005) × 10 ⁶	<code>\num{1.234(5)}\</code> <code>\num{1.234(5)e6}</code>
trapambigerr	If the number has an exponent, or if units are not repeated, then the result can be considered ambiguous. By default, the package adds the markers stored in openerr and closeerr to remove the ambiguity; the options have the same default values as the input error markers. Detection of a potentially-ambiguous error is controlled by the trapambigerr option, although for numbers with units the repeatunits option is also important. The spacing around the ± sign is normally set by T _E X. However, using the tightpm option will cause this to be reduced to a minimum.
openerr	
closeerr	
tightpm	
1.234 × 10 ⁶ ± 0.005 × 10 ⁶	<code>\sisetup{seperr}\</code> <code>\num[trapambigerr=false]{1.234(5)e6}\</code>
1.234 m ± 0.005 m	<code>\SI{1.234(5)}{\metre}\</code>
(1.234 ± 0.005) m	<code>\sisetup{repeatunits=false}</code> <code>\SI{1.234(5)}{\metre}\</code>
1.234 ± 0.005 m	<code>\SI[trapambigerr=false]{1.234(5)}{\metre}\</code>
1.234±0.005	<code>\num[tightpm]{1.234(5)}</code>

²¹fullstop also has aliases stop and period.

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

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

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

`expproduct` The formatting of exponents is controlled by `expproduct` and `expbase`.
`expbase` `expproduct` sets the symbol used to indicate a product for exponents (e.g. the \times
`allowzeroexp` in 2×10^2), while the value of `expbase` sets the power used (the 10 in the example). Both options accept a very short list of options: **times**, **tighttimes**, **cdot** and **tightcdot** for the product, and **ten** and **two** for the power.²³ Other choices are used literally. Also relevant to exponent processing is the `allowzeroexp` option. By default, the package will suppress a zero exponent, but setting the flag will allow the output of 10^0 .

`addsign` Additions to the input can take the form of implicit signs and padded ze-
`sign` ros. The `addsign` option takes a list of potential sites to add a sign: **none**,
`retainplus` **mantissa**, **exponent** and **both**.²⁴ If no sign is given in the input, the setting here determines if one is added. The sign to add is stored in `sign`, which takes the list of choices **plus**, **minus**, **pm** and **mp**, or uses the input literally (in maths mode). For positive numbers, the `retainplus` option causes a $+$ sign explicitly in the input to be retained. By default, the package will remove such signs.

`padnumber` The `padnumber` option controls the addition of zeros to the input, to “pad” the result. The option takes a list of choices: **leading**, **trailing**, **both** and **none**.²⁵ No additional precision is added by this option; integer input will not add a decimal point.

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

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

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

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

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

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

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

10.4 Angle formatting

`padangle` The angle formatter uses `\num` to format numbers: any options for numbers are
`strictarc` therefore applicable here. The `padangle` option takes choices **small**, **large**,
all and **none**, and controls how angles are padded when given in degrees,
minutes and seconds.²⁶ When giving angles as arcs (in degrees, minutes and
seconds), the package can detect if the correct number of semi-colons have
been given. This is controlled by the `strictarc` option, which is a Boolean
switch with a **true** default. With `strictarc` set to **false**, an incomplete arc is
interpreted as degrees and minute, while an over-complete one will drop excess
input.

1°	<code>\ang[padangle=none]{1;}\</code>
2°0'0"	<code>\ang[padangle=large]{2;}\</code>
3°	<code>\ang[padangle=small]{3;}\</code>
0°4'0"	<code>\ang[padangle=both]{;4;}\</code>
5'5"	<code>\ang[padangle=none]{;5;5}\</code>
1°2'	<code>\ang[strictarc=false]{1;2}\</code>
1°2'3"	<code>\ang[strictarc=false]{1;2;3;4;}</code>

`angformat` The angle formatting system can convert between decimal angles and those
given as degrees, minutes and seconds. This is controlled by the `angformat`
option, which takes choices **unchanged**, **decimal** and **arc**.²⁷ When set to
unchanged, nothing is done to the input. The conversion is based on 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.

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

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

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

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

10.5 Tabular material

`tabnumalign` Material typeset in S columns is processed internally by the `\num` macro. Thus, as with angles, the number options also apply here. The positioning of tabular material is controlled by the two options `tabnumalign` and `tabformat`. `tabnumalign` takes values **centredecimal**, `centre`, `left` and `right`.²⁸ When using `centredecimal`, the package places the decimal marker of the mantissa at the centre of the column, which then grows to accommodate the widest number given. For equal numbers of digits before and after the decimal sign, this is the easiest option. The other choices use a fixed-width box to store the number; the box is then aligned with the edges of the column.

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

`tabalignexp` When `tabformat` contains exponents, two possibilities are available for alignment. The first method is to place the exponent parts so that the “ $\times 10$ ” parts form a column, with whitespace after shorter mantissa components. In the second method, no additional space is added after the mantissa, and the exponents do not line up (Table 14). This is controlled by the `tabexpalign` option, which can be set to `true` or `false`.

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

Table 14: The `tabalignexp` option

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

Table 15: The `tabautofit` option

Header	Header	Header
1.2	1.200	1.2
1.2345	1.235	1.2345

`tabtextalign` Cells containing no numbers are handled by `siunitx` in a manner similar to `\multicolumn`. The setting of `tabtextalign` is taken from the list **centre**, `tabunitalign` **right** and **left**.²⁹ As would be expected, these settings **centre**, **right**- or **left**-align the cell contents. In `s` columns, all content is treated as input to the `\si` macro. The alignment of the contents relative to the cell is controlled by the `tabunitalign` option, which takes options **left**, **right** and **centre**. The settings for `tabnumalign`, `tabtextalign` and `tabunitalign` can be set to the same value in one go with the `tabalign` option.

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

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

²⁹Alias `centre = center`.

10.6 Units

`per` Most of the unit options are concerned with the processing of named units. The processor for units given as macro names can be influenced to give a variety of output formats. The `per` option defines how the keyword macro `\per` is handled. This option takes a choice from the list **reciprocal**, `slash` and `fraction`.³⁰ The default option uses `\per` to indicate reciprocal powers, whereas `slash` causes the package to use `"/` to show division.

`fraction` The `fraction` option defines how `per=fraction` is interpreted. The list of applicable values here is **frac**, `nice`, `ugly` and `sfrac`. In each case, the unit is typeset as a fraction, but the macro used 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>
m/s	<code>\si[fraction=frac]{\metre\per\second}\\</code>
m_s	<code>\si[fraction=nice]{\metre\per\second}\\</code>
m/s	<code>\si[fraction=sfrac]{\metre\per\second}\\</code>
	<code>\si[per=slash]{\metre\per\second}</code>

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

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

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

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

`prefixsymbolic` The unit prefixes (`\kilo`, *etc.*) are normally given as letters. However, the package can convert these into numerical powers.³⁵ This is controlled by the `prefixsymbolic` Boolean option, which by default is **true**. If `prefixsymbolic` is set to **false**, the format of the prefix is controlled by

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

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

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

³⁴This is the behaviour in `Slunits`.

³⁵Provided things are not too complex!

prefixbase and prefixproduct, which work in the same way as expbase and expproduct.

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

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

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

10.7 Symbols

User access to control the symbols used for Ω , μ , $^\circ$, $'$, $''$, \AA and $^\circ\text{C}$ is provided here. These are all literal options, which are available in text and maths mode variants. For example, `textmicro` is the code used for the μ symbol in text mode. The text mode macros should be safe when forced into text, and the maths ones when forced into maths. The symbols defined in this way are:

- `textOmega;`
- `mathsOmega;`
- `textmu;`
- `mathsmu;`
- `textdegree;`
- `mathsdegree;`
- `textminute;`
- `mathsminute;`
- `textsecond;`
- `mathssecond;`
- `textringA;`
- `mathsringA;`
- `textcelsius;`
- `mathscelsius.`

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

`eVcorra` The `eV` symbol requires some fine-tuning, and so has two options of its own, both \TeX lengths. `eVcorra` is the correction applied to the gap between “e” and “V” of the unit: the default is `0.3ex`. `eVcorrb` is the correction applied to the gap between “V” of the unit and whatever follows; the default is `0ex`. The optimal value for these options will depend on the current font settings.³⁶

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

10.8 Colour

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

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

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

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

10.9 International support

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

`1.234 m` `\SI{1.234}{\metre}\`
`6,789 m` `\SI[locale=DE]{6.789}{\metre}`

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

³⁶This document uses `eVcorra=0.1ex`.

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

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

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

10.10 Package control

`load` The package keeps most of the unit and abbreviations definitions in files separate from `siunitx.sty`. To control what is loaded, three complementary options are provided, all of which take a list of one or more choices. `load` and `alsoload` define which support configuration files are loaded. The list in `load` recognises the value `default`, which is expanded to the normal list before loading. The difference between `load` and `alsoload` is that `load` specifies the *complete* list of files to load, whereas `alsoload` adds to the existing list. To use the `load` option successfully requires knowing everything that is needed. The `noload` option can be used to delete one or more items from the `load` list, without needing to know what is on it.³⁷

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

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

10.11 Back-compatibility options

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

10.12 Summary of all options

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

Table 16: All package options

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

Continued on next page

³⁷`noload` does not prevent the loading of a file needed by one which is loaded. Thus the package may internally override a `noload` value if needed.

Option	Type	Description
angformat	List	Conversion of angle format
anglesep	List or literal	Space between angle components
astroang	Boolean	Astronomy-style angles
closeerr	Literal	Closes potential-ambiguous error
closefrac	Literal	Closes potential-ambiguous fraction
color	Literal	Colour used for units and values
colour	Literal	Colour used for units and values
colorall	Boolean	Switch for colouring all output
colourall	Boolean	Switch for colouring all output
colorneg	Boolean	Colour negative numbers
colourneg	Boolean	Colour negative numbers
colorunits	Boolean	Switch for colouring units
colourunits	Boolean	Switch for colouring units
colorvalues	Boolean	Switch for colouring values
colourvalues	Boolean	Switch for colouring values
decimalsymbol	List or literal	Decimal symbol
debug	Boolean	Write debugging data to log
detectdisplay	Boolean	Treat display maths separately
digitsep	List	Separation of digits in large numbers
dp	Integer	Number of decimal places to output numbers to
emulate	Modules	Emulation modules to load
errspace	List	Spacing of bracketed error
eVcorra	Length	Spacing correction in eV
eVcorrb	Length	Spacing correction after eV
expbase	List or Literal	Base used for exponents
expproduct	List or literal	Product sign for exponents
fixdp	Boolean	Switch for fixing decimal places of numbers
fraction	List	Method used when setting <code>per=frac</code>
inlinebold	List	Select how inline bold is tested
load	Modules	Modules to load
locale	Modules	Locale to follow
loctolang	Special	Associate locale with babel language
log	List	Amount of data added to log
mathOmega	Literal	"Ω" symbol in maths mode
mathcelsius	Literal	"°C" symbol in maths mode
mathdegree	Literal	"°" symbol in maths mode
mathminute	Literal	"′" symbol in maths mode
mathmu	Literal	"μ" symbol in maths mode
mathringA	Literal	"Å" symbol in maths mode
mathrm	Csname	Roman maths font
mathsOmega	Literal	"Ω" symbol in maths mode
mathscelsius	Literal	"°C" symbol in maths mode
mathsdegree	Literal	"°" symbol in maths mode
mathsecond	Literal	"″" symbol in maths mode
mathsf	Csname	Sans serif maths font

Continued on next page

Option	Type	Description
mathsminute	Literal	"'" symbol in maths mode
mathsmu	Literal	"μ" symbol in maths mode
mathsringA	Literal	"Å" symbol in maths mode
mathsrm	Csname	Roman maths font
mathssecond	Literal	"'" symbol in maths mode
mathssf	Csname	Sans serif maths font
mathstt	Csname	Fixed-width maths font
mathtt	Csname	Fixed-width maths font
mode	List	Use text or maths mode for typesetting
negcolor	Literal	Colour used for negative numbers
negcolour	Literal	Colour used for negative numbers
noload	Modules	Modules not to load
numaddn	Literal	Additional input allowed in numbers
numcloseerr	Literal	Character indicating end of numerical error
numdecimal	Literal	Decimal symbols in numbers
numdigits	Literal	Digit characters in numbers
numexp	Literal	Exponent characters in numbers
numgobble	Literal	Characters to ignore in numbers
numopenerr	Literal	Character indicating start of numerical error
numprod	Literal	Characters used for a product
numsign	Literal	Sign characters in numbers
obeyall	Boolean	Combination of obeybold, obeyitalic and obeymode
obeybold	Boolean	Check local bold setting
obeyitalic	Boolean	Check local italic setting
obeymode	Boolean	Check local mode (text/maths)
openerr	Literal	Opens potentially-ambiguous error
openfrac	Literal	Opens potentially-ambiguous fraction
padangle	List	Add zeros to blank parts of angles
padnumber	List	Add zeros to blank parts of numbers
per	List	Behaviour of \per
prefixbase	List or literal	Base used when making prefixes numerical
prefixproduct	List or literal	Product sign for prefixes
prefixsymbolic	Boolean	Behaviour of unit prefixes
prespace	Boolean	Add space before units
redefsymbols	Boolean	Use better symbols if available
repeatunits	List	Repeat units with separated errors and products
retainplus	Boolean	Retain explicit plus sign
seper	Boolean	Separate number and error
sepfour	Boolean	Separate four-digit numbers
sign	List or literal	Sign to add to numbers
slash	List or literal	Symbol used for "/"

Continued on next page

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

11 Emulation of other packages

siunitx has been designed as a replacement for Slunits, Slstyle, unitsdef, units, hepunits, fancyunits and fancynum. It therefore provides options reproduce the functions of all of these packages. In this way, siunitx should be usable as a straight replacement for the older packages.³⁸ This means for example that the `\num` macro takes an optional star when emulating Slstyle. However, there are some points that should be remembered. In particular, siunitx validates numerical input, meaning that places where a number is expected in the older packages *require* a number when emulated by siunitx.

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

```

-123 456
-123 456 N/mm2

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

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

12 Configuration files

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

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

```

Some text as filler
TEST This example is rather trivial!

\addtolocale{DE}{TEST}
Some text as filler \SI[obeymode]{-123456}{N/mm^2}
\sisetup{locale=DE}
This example is rather trivial!
```

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

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

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

13 Common questions

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

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

13.2 Why is the order of my units changed?

Then using `per=reciprocal`, the units are typeset as given. However, when using `per=slash` or `per=fraction`, the package needs to find which ones are in the denominator. It then prints the numerator and denominator separately. So if you give a unit in the denominator *before* one in the numerator, they have to be re-ordered.³⁹

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

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

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

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

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

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

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

13.4 How do I set superscripts to use lining numbers?

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

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

1234 m s⁻¹
1234 m s⁻¹

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

13.5 Why do most of the examples use J mol⁻¹ K⁻¹?

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

13.6 What can `numprint` do that `siunitx` cannot?

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

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

14 Tricks and known issues

14.1 Ensuring maths mode

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

14.2 Using `.` and fixed spaces in units

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

10 V vs. NHE

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

Table 17: Passing single digit characters

Heading
1-2
1
1
-

14.3 Passing unprocessed digits through an S column

The method used to detect numbers in an S column will pick up material wrapped inside braces if there is more than a single character inside the braces. If you want to pass a *single* numerical character without processing it, you need two sets of braces (Table 17).

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

14.4 Limitations of \mathrm

The package uses the \mathrm font family by default to typeset output in maths mode. This however has a few side-effects. For example, the Greek alphabet can give odd results.⁴⁰ The use of the \mathnormal font *may* get around this issue.

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

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

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

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

14.5 Entire document in sans serif font

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

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

14.6 Effects of emulation

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

14.7 Centring columns on non-decimal input

The dcolumn manual suggests using that package to align a column on a \pm sign. The same type of output is possible using siunitx, but some care is needed (Table 18). Odd things may happen: use with care!

```

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

14.8 Adding items after the last column of a tabular

If you use an S or s column as the last one in a tabular, and you use the array “<” construction to add items after it, the spacing may be wrong. This will occur if the column contents are of differing widths. Changing the \LaTeX `\\` line ending

Table 18: Non-standard S column

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

for the plain \TeX `\cr` will give the correct spacing, but does not allow adjustment of inter-row distance (Table 19).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}
```

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

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 19: 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

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 (`Sstyle`), 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 5. Apart from the kilogram, these are defined in terms of a measurable physical quantity needing the definition alone.⁴² The base units have been chosen such that all physical quantities can be expressed using an appropriate combination of these units, needing no others and with no redundancy. The kilogram is slightly different from the other base units as it is still defined in terms of a “prototype” held in Paris.⁴³

19.2 SI derived units

All other units within the SI system are regarded as “derived” from the seven base units. At the most basic, all other SI units can be expressed as combinations of the base units. However, many units (listed in Table 7, Table 8 and Table 9)

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

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

have a special name and symbol.⁴⁴ Most of these units are simple combinations of one or more base units (raised to powers as appropriate). A small number of units derived from experimental data are allowed as SI units (Table 8).

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

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

19.3 SI prefixes

A series of SI prefixes for decimal multiples and submultiples are provided, and can be used as modifiers for any SI unit (either base or derived units) with the exception of the kilogram. The prefixes are listed in Table 6. No space should be used between a prefix and the unit, and only a single prefix should be used. Even the degree Celsius can be given a prefix, for example $1 \text{ m}^\circ\text{C}$. The only exception to this rule is for degrees, minutes and seconds of an arc: $1^\circ 2' 3''$.

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

19.4 Other units

The application of SI units is meant to provide a single set of units which ensure consistency and clarity across all areas. However, other units are common in many areas, and are not without merit. The units provided by `siunitx` by default do not include any of these; only units which are part of the SI set or are accepted for use with SI units are defined. However, several other sets of units can be loaded as optional modules. The binary prefixes and units (Section 8.1 and Table 12) are the most obvious example. These are *not* part of the SI specifications, but the prefix names are derived from those in Table 6.

Other units (such as those provided by the modules `synchem`, `hep` and `astro`) are normally to be avoided where possible. SI units should, in the main, be preferred due to the advantages of clear definition and self-consistency this brings. However, there will probably always be a place for specialist or non-standard units. This is particularly true of units derived from basic physical constants; for example reason, the `hep` module defines the speed of light, c , as a unit. For work in basic science, a small number of physical constants are recognised as units provided the results for comparison with experiment are given in SI units.

There are also many areas where non-standard units are used so commonly that to do otherwise is difficult or impossible. For example, most synthetic

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**: 54 F
But perhaps in a running block,
it might look better: 54 F

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

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 20 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 20: 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 21**. The use of `\multicolumn` is needed here due to the “<”; without `\multicolumn`, the titles are followed by “kg”!

```

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

```

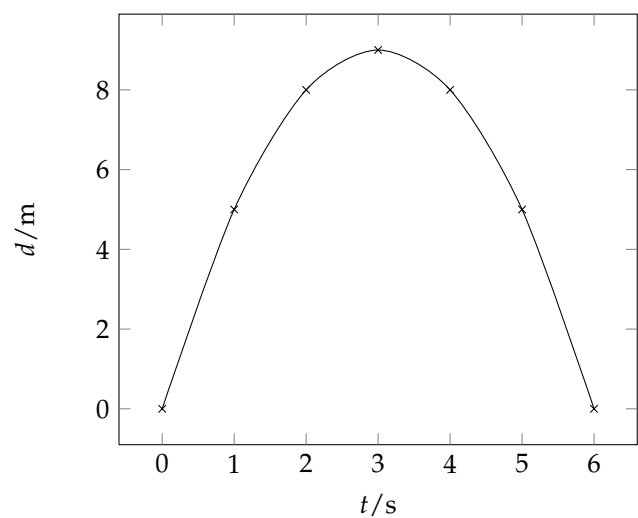


Figure 1: An example of graph labelling

Table 21: 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{v1.0i}
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 121 2008-08-14 14:06:03Z 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\string'\= 12\relax
32 \catcode\string'\^ 7\relax
33 \catcode\string'\~ \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 $\mathcal{A}\mathcal{M}\mathcal{S}$ 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
\si@tempboxc
\si@tempboxd
46 \newbox\si@tempboxa
47 \newbox\si@tempboxb
48 \newbox\si@tempboxc
49 \newbox\si@tempboxd

```

`\si@temptoks` A token register is also handy.

```

50 \newtoks\si@temptoks

```

`\si@packagecheck` As siunitx is intended to replace the other unit-management packages, these are tested for before any further processing. If any are loaded, the package halts

`\si@blockpkgs`

`\si@checkpkgs`

compilation; name clashes or unexpected results could occur if this is not tested. Notice that `SIunits` and `SIstyle` could be loaded with variable capitalisation (at least on Windows); both possibilities are tested.

```

51 \newcommand*{\si@blockpkgs}{SIunits,sistyle,siunits,SIstyle,%
52  unitsdef,fancyunits}
53 \newcommand*{\si@checkpkgs}{units,hepunits,fancynum}
54 \newcommand*{\si@packagecheck}{%
55  \begingroup
56  \@for\si@tempa:=\si@blockpkgs\do{
57    \ifpackageloaded{\si@tempa}
58      {\PackageError{siunitx}
59        {Package '\si@tempa' incompatible}
60        {The \si@tempa\space package and siunitx are
61          incompatible.\MessageBreak Use the
62          'emulate=\si@tempa' package option when loading
63          siunitx}}
64      {}}

```

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

```

65  \@for\si@tempa:=\si@checkpkgs\do{%
66    \ifpackageloaded{\si@tempa}
67      {\PackageWarning{siunitx}
68        {Consider loading the siunitx package
69          with\MessageBreak option 'emulate=\si@tempa', rather
70          than\MessageBreak loading both \si@tempa\space and
71          siunitx}}
72      {}}
73  \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.

```

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

```

`\si@ifdefinable` #1 : macro

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

```

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

```

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

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

```

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

```

```

\si@addtocname #1 : csname
#2 : tokens
A second item to add to a command sequence.
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 `ifmtarg` is copied here with an internal name.

```

\si@xifmtarg
\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 #1 : macro
\si@newcommand [#2]: num-args
\si@newcmd [#3]: default
\si@xargdef #4 : def

```

Some more copying, this time from `etoolbox`. Various macros need to be *really* robust. This is achieved using the ϵ -TeX `\protected` primitive in various places. However, it would be nice to have a `\protected` version of `\newcommand`. `etoolbox` has code for that, but to avoid needing to load it, the necessary stuff is copied here. The only changes from the original are names, and the use of `\newcommand*` for the `\si@newcommand` macro.

```

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

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

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}

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

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 #1 : debug-info
              The debug macro only gives output if the appropriate package option is set.

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 #1 : char
\si@str@chrstr  #2 : string
                At various points, the package needs to compare two strings, to find if one occurs
                in the other. The first test is if a single character is part of a second string; this is
                used, for example, to check that a character is valid as input. The first argument
```

is not expanded further, but the second is two allow division into individual units.

```

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

```

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

```

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 #1 : options
```


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

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

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

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

```
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 [#1]: default
                #2 : keyname
                #3 : function
```

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

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

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

Whereas multiple definitions do not take a function.

```
195 \newcommand*{\si@opt@cmdkeys}[2][[]]{%
196   \define@cmdkeys[si]{key}[si@]{#2}[#1]}
```

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

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

```
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 [#1]: default
                   #2 : keyname
                   #3 : choices
                   #4 : function
```

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

```
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 #1 : keyname
                   #2 : choices
                   #3 : initial-value
```

Several of the package options can take either a choice from a list of known options, or a value to be interpreted literally. To aid maintenance, the necessary code can be set up here. These keys all define a new macro, which must exist.

The `\si@opt@xchoicekey` macro therefore ensures that this is defined, as well as setting up the `xkeyval` key.

```
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 #1 : package
                  #2 : keyname
```

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

```
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 #1 : keyname
                   #2 : text
```

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

```
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@choicekey[normal]{log}{debug,verbose,normal,errors,none}
```

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

```
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 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}

```

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

⁵¹Only valid for `valuesep`.

```

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@numtextmodetrue
309  \si@unittextmodetrue
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@numtextmodetrue
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@unittextmodetrue
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 `\bffamily`).

```

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@valuemathsssf
\si@valuemathstt
\si@unitmathsrms 350 \si@opt@cmdkeys{valuemathsrms,valuemathsssf,valuemathstt}
\si@unitmathsssf 351 \si@opt@cmdkeys{unitmathsrms,unitmathsssf,unitmathstt}
\si@unitmathstt

```

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

```

352 \si@opt@key{valuemathrm}{\sisetup{valuemathsrms=#1}}
353 \si@opt@key{valuemathssf}{\sisetup{valuemathsssf=#1}}
354 \si@opt@key{valuemathhtt}{\sisetup{valuemathstt=#1}}
355 \si@opt@key{unitmathrm}{\sisetup{unitmathsrms=#1}}
356 \si@opt@key{unitmathssf}{\sisetup{unitmathsssf=#1}}
357 \si@opt@key{unitmathhtt}{\sisetup{unitmathstt=#1}}

```

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

```

358 \si@opt@key{mathsrms}{\sisetup{valuemathsrms=#1,unitmathsrms=#1}}
359 \si@opt@key{mathsssf}{\sisetup{valuemathsssf=#1,unitmathsssf=#1}}
360 \si@opt@key{mathstt}{\sisetup{valuemathstt=#1,unitmathstt=#1}}
361 \si@opt@key{mathrm}{\sisetup{valuemathsrms=#1,unitmathsrms=#1}}
362 \si@opt@key{mathssf}{\sisetup{valuemathsssf=#1,unitmathsssf=#1}}
363 \si@opt@key{mathtt}{\sisetup{valuemathstt=#1,unitmathstt=#1}}

```

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

```

\si@valuetextsf
\si@valuetexttt 364 \si@opt@cmdkeys{valuetextrms,valuetextsf,valuetexttt}
\si@unittextrms 365 \si@opt@cmdkeys{unittextrms,unittextsf,unittexttt}
\si@unittextsf 366 \si@opt@key{textrms}{\sisetup{unittextrms=#1,valuetextrms=#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
\si@numprod 369 \si@opt@cmdkeys{numdigits,numdecimal,numexp,numgobble,numsign,%
\si@numgobble 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@numvalid
\si@numcloseerr
\si@numextra
\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
\ifsi@trapambigerr from the number.
    \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 are 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” \pm 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@choicekey[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, 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@choicekey[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@fixed
\si@tabnumalign
\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}%

```

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

```

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 \fi
588 \fi

```

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

```

589 \ifsi@tab@fixed\else
590 \sisetup{tabnumalign=centre}%
591 \fi
592 \let\pm\si@num@pm
593 \let\mp\si@num@mp}

```

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

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

```

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

```

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

```

610 \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 611 {\let\si@tab@rfill@t\hfill
612 \let\si@tab@lfill@t\hfill
613 \renewcommand*{\si@tempb}{right}%
614 \ifx\si@tempa\si@tempb
615 \let\si@tab@rfill@t\relax
616 \fi
617 \renewcommand*{\si@tempb}{left}%
618 \ifx\si@tempa\si@tempb
619 \let\si@tab@lfill@t\relax
620 \fi}

```

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

```
621 \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 622 {\let\si@tab@rfill@s\hfill
623 \let\si@tab@lfill@s\hfill
624 \renewcommand*{\si@tempb}{right}%
625 \ifx\si@tempa\si@tempb
626 \let\si@tab@rfill@s\relax
627 \fi
628 \renewcommand*{\si@tempb}{left}%
629 \ifx\si@tempa\si@tempb
630 \let\si@tab@lfill@s\relax
631 \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
632 \si@opt@boolkey{fixdp}
633 \newcount\si@num@dp
634 \si@opt@key{dp}{%
635 \si@str@ifonlychrs{#1}{0123456789}
636 {}
637 {\si@log@err{Invalid input for dp option}
638 {The dp option must be given a positive integer}}%
639 \si@num@dp#1\relax
640 \si@fixdptrue}
```

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

```
641 \si@opt@boolkey{tabautofit}
```

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

```
642 \si@opt@boolkey{xspace}
```

`\ifsi@prespace`

```
643 \si@opt@boolkey
644 [\si@unt@numfalse
645 \ifsi@prespace
646 \si@unt@numtrue
647 \fi]
648 {prespace}
```

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

```
649 \si@opt@boolkey{allowoptarg}
```

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

```
\ifsi@slash 650 \newif\ifsi@slash
\ifsi@stickyper 651 \newif\ifsi@frac
652 \si@opt@boolkey{stickyper}
653 \si@opt@choicekey[reciprocal]{per}
654 {reciprocal,rp,power,slash,frac,fraction}
655 {\si@slashfalse
656 \si@fracfalse
657 \renewcommand*{\si@tempb}{slash}%
```

```

658 \ifx\si@tempa\si@tempb
659 \si@fractrue
660 \si@slashtrue
661 \let\si@frac\si@frc@slash
662 \fi
663 \renewcommand*{\si@tempb}{frac}%
664 \ifx\si@tempa\si@tempb
665 \si@fractrue
666 \fi
667 \renewcommand*{\si@tempb}{fraction}%
668 \ifx\si@tempa\si@tempb
669 \si@fractrue
670 \fi}

```

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

```

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

```

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

```

\ifsi@addunitpower 672 \newif\ifsi@repeatunits
673 \newif\ifsi@addunitpower
674 \si@opt@choicekey[true]{repeatunits}{true,false,power}
675 {\si@repeatunitsfalse
676 \si@addunitpowerfalse
677 \renewcommand*{\si@tempb}{true}%
678 \ifx\si@tempa\si@tempb
679 \si@repeatunitstrue
680 \fi
681 \renewcommand*{\si@tempb}{power}%
682 \ifx\si@tempa\si@tempb
683 \si@addunitpowertrue
684 \fi}

```

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

```

\si@closefrac
\si@openfrac 685 \si@opt@boolkey{trapambigfrac}
686 \si@opt@cmdkeys{closefrac,openfrac}

```

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

```

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

```

```

701 \fi
702 \renewcommand*{\si@tempb}{sfrac}%
703 \ifx\si@tempa\si@tempb
704   \let\si@frac\si@frc@sfrac
705 \fi
706 \renewcommand*{\si@tempb}{xfrac}%
707 \ifx\si@tempa\si@tempb
708   \let\si@frac\si@frc@sfrac
709 \fi
710 \renewcommand*{\si@tempb}{ugly}%
711 \ifx\si@tempa\si@tempb
712   \let\si@frac\si@frc@ugly
713 \fi}

```

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

```

714 \si@opt@cmdkeys{load,noload}
715 \si@opt@key{alsoload}{\si@addtolist{\si@load}{#1}}
716 \AtEndOfPackage{
717   \si@opt@disablekey{load}
718   {Configuration files can only be used\MessageBreak
719     when loading package}
720   \si@opt@disablekey{noload}
721   {Configuration files can only be used\MessageBreak
722     when loading package}}
723 \AtEndOfPackage{
724   \si@opt@key{alsoload}{%
725     \@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
\ifsi@colourvalues spellings are also provided.

```

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

```

```

748     \si@colourunitstrue
749     \si@colourvaluestrue
750   \fi}
751 \si@opt@choicekey[true]{colourall}
752 {true,false}
753 {\si@colourvaluesfalse
754  \si@colourunitsfalse
755  \renewcommand*{\si@tempb}{true}%
756  \ifx\si@tempa\si@tempb
757    \si@colourunitstrue
758    \si@colourvaluestrue
759  \fi}
760 \si@opt@cmdkeys{unitcolour,valuecolour}
761 \si@opt@key{unitcolor}{\sisetup{unitcolour=#1}}
762 \si@opt@key{valuecolor}{\sisetup{valuecolour=#1}}
763 \si@opt@key{colour}{\sisetup{unitcolour=#1,valuecolour=#1}}
764 \si@opt@key{color}{\sisetup{unitcolour=#1,valuecolour=#1}}

```

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

```

\si@negcolour 765 \si@opt@boolkey{colourneg}
766 \si@opt@choicekey[true]{colorneg}
767 {true,false}
768 {\si@colournegfalse
769  \renewcommand*{\si@tempb}{true}%
770  \ifx\si@tempa\si@tempb
771    \si@colournegtrue
772  \fi}
773 \si@opt@cmdkeys{negcolour}
774 \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.

```

775 \si@opt@cmdkeys{textOmega,mathsOmega}
776 \si@opt@key{mathOmega}{\sisetup{mathsOmega=#1}}
777 \newcommand*{\si@mathsOmega}{\text{\ensuremath{\Omega}}}
778 \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 779 \si@opt@cmdkeys{textmu,mathsmu}
780 \si@opt@key{mathmu}{\sisetup{mathsmu=#1}}
781 \DeclareFontEncoding{TS1}{}{}
782 \DeclareFontSubstitution{TS1}{cmr}{m}{n}
783 \DeclareTextSymbol{\si@textmu}{TS1}{181}
784 \DeclareTextSymbolDefault{\si@textmu}{TS1}
785 \@ifpackageloaded{upgreek}
786 {}
787 {\DeclareFontFamily{OML}{eur}{\skewchar\font'177}
788  \DeclareFontShape{OML}{eur}{m}{n}{%
789    <-6> eurm5 <6-8> eurm7 <8-> eurm10}{}}
790 \AtBeginDocument{
791   \@ifpackageloaded{upgreek}
792     {\let\si@mathsmu\upmu}

```



```

793     {\DeclareSymbolFont{si@greek}{OML}{eur}{m}{n}}
794     \DeclareMathSymbol{\si@mathsmu}{\mathord}{si@greek}{"16}}

\si@textdegree   The angle signs.
\si@mathsdegree 795 \si@opt@cmdkeys{textdegree,mathsdegree,textminute,mathsminute,
\si@textminute 796   textsecond,mathssecond}
\si@mathsminute 797 \si@opt@key{mathdegree}{\sisetup{mathsdegree=#1}}
\si@textsecond 798 \si@opt@key{mathminute}{\sisetup{mathsminute=#1}}
\si@mathssecond 799 \si@opt@key{mathsecond}{\sisetup{mathssecond=#1}}
800 \newcommand*{\si@textdegree}{\ensuremath{{}^{\circ}}}
801 \newcommand*{\si@mathsdegree}{\mathring{}}
802 \newcommand*{\si@textminute}{\ensuremath{{}'}}
803 \newcommand*{\si@mathsminute}{\mathring{}}
804 \newcommand*{\si@textsecond}{\ensuremath{{}''}}
805 \newcommand*{\si@mathssecond}{\mathring{}}

\si@textcelsius   Finally, degrees Celsius, which may need the degree symbol.
\si@mathscelsius 806 \si@opt@cmdkeys{textcelsius,mathscelsius}
807 \si@opt@key{mathcelsius}{\sisetup{mathscelsius=#1}}
808 \newcommand*{\si@textcelsius}{\si@textdegree}
809 \si@textdegree\kern-\scriptspace C}
810 \newcommand*{\si@mathscelsius}{\mathring{C}}
811 \si@mathsdegree\kern-\scriptspace\mathrm{C}}

\si@textringA     The Å sign.
\si@mathsringA 812 \si@opt@cmdkeys{textringA,mathsringA}
813 \si@opt@key{mathringA}{\sisetup{mathsringA=#1}}
814 \newcommand*{\si@textringA}{\AA}
815 \newcommand*{\si@mathsringA}{\text{\AA}}

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

\si@eVcorra
\si@eVcorrb 821 \newlength\si@eVcorra
822 \newlength\si@eVcorrb
823 \si@opt@key{eVcorra}{\setlength\si@eVcorra{#1}}
824 \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.
825 \si@opt@key{locale}{%
826   \si@loc@load{#1}%
827   \si@loc@set{#1}%
828 \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 users of the older packages. These all set up switches, but do not do anything.
`\ifsi@old@nice` That is left to the emulation files, loaded at the end of the package. First of all,
`\ifsi@old@loose` the units options are dealt with; there are not many.
`\ifsi@old@tight`

```
829 \si@opt@compatkey{units}{ugly}
830 \si@opt@compatkey{units}{nice}
831 \si@opt@compatkey{units}{loose}
832 \si@opt@compatkey{units}{tight}
```

`\ifsi@old@OHM` The `unitsdef` package is unfortunately much more profligate with options. The first set are to do with support for `gensymb`.
`\ifsi@old@ohm`

```
\ifsi@old@redef-gensymb 833 \si@opt@compatkey{unitsdef}{OHM}
\ifsi@gensymb 834 \si@opt@compatkey{unitsdef}{ohm}
835 \si@opt@compatkey{unitsdef}{redef-gensymb}
836 \newif\ifsi@gensymb
```

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

```
\ifsi@old@liter 837 \si@opt@compatkey{unitsdef}{LITER}
\ifsi@old@noospace 838 \si@opt@compatkey{unitsdef}{liter}
\ifsi@old@noconfig 839 \si@opt@compatkey{unitsdef}{noospace}
840 \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 841 \si@opt@compatkey{unitsdef}{noabbr}
\ifsi@old@nomolabbr 842 \si@opt@compatkey{unitsdef}{noampereageabbr}
\ifsi@old@novoltageabbr 843 \si@opt@compatkey{unitsdef}{nofrequncyabbr}
\ifsi@old@novolumeabbr 844 \si@opt@compatkey{unitsdef}{nomolabbr}
\ifsi@old@noweightabbr 845 \si@opt@compatkey{unitsdef}{novoltageabbr}
\ifsi@old@noenergyabbr 846 \si@opt@compatkey{unitsdef}{novolumeabbr}
\ifsi@old@nolengthabbr 847 \si@opt@compatkey{unitsdef}{noweightabbr}
\ifsi@old@notimeabbr 848 \si@opt@compatkey{unitsdef}{noenergyabbr}
849 \si@opt@compatkey{unitsdef}{nolengthabbr}
850 \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 851 \si@opt@compatkey{SIunits}{cdot}
\ifsi@old@mediumspace 852 \si@opt@compatkey{SIunits}{thickspace}
\ifsi@old@thinspace 853 \si@opt@compatkey{SIunits}{mediumspace}
\ifsi@old@thickqspace 854 \si@opt@compatkey{SIunits}{thinspace}
\ifsi@old@mediumqspace 855 \si@opt@compatkey{SIunits}{thickqspace}
\ifsi@old@thinqspace 856 \si@opt@compatkey{SIunits}{mediumqspace}
857 \si@opt@compatkey{SIunits}{thinqspace}
```

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

```
\ifsi@old@squaren 858 \si@opt@compatkey{SIunits}{amssymb}
\ifsi@old@pstricks 859 \si@opt@compatkey{SIunits}{squaren}
\ifsi@old@Gray 860 \si@opt@compatkey{SIunits}{pstricks}
\ifsi@old@italian 861 \si@opt@compatkey{SIunits}{Gray}
862 \si@opt@compatkey{SIunits}{italian}
```

```

\ifsi@old@textstyle The miscellaneous options.
\ifsi@old@binary 863 \si@opt@compatkey{SIunits}{textstyle}
\ifsi@old@noams 864 \si@opt@compatkey{SIunits}{binary}
\ifsi@old@derivedinbase 865 \si@opt@compatkey{SIunits}{noams}
\ifsi@old@derived 866 \si@opt@compatkey{SIunits}{derivedinbase}
867 \si@opt@compatkey{SIunits}{derived}

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

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

\ifsi@old@tight A couple for spacing multiplication.
\ifsi@old@loose 871 \si@opt@compatkey{fancynum}{tight}
872 \si@opt@compatkey{fancynum}{loose}

\ifsi@old@thinspaces Three for digit separation.
\ifsi@old@commas 873 \si@opt@compatkey{fancynum}{thinspaces}
\ifsi@old@plain 874 \si@opt@compatkey{fancynum}{commas}
875 \si@opt@compatkey{fancynum}{plain}

\ifsi@old@spaceqspace The fancyunits package provides one option not available with SIunits.
876 \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 877 \newcommand*{\si@fix@thin}{\,}
\si@fix@thick 878 \newcommand*{\si@fix@med}{\:}
\si@fix@space 879 \newcommand*{\si@fix@medium}{\:}
880 \newcommand*{\si@fix@thick}{\;}
881 \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 882 \newcommand*{\si@fix@cdot}{{}\cdot{}}
\si@fix@fullstop 883 \newcommand*{\si@fix@comma}{{,}}
\si@fix@period 884 \newcommand*{\si@fix@stop}{{.}}
\si@fix@times 885 \newcommand*{\si@fix@fullstop}{{.}}
\si@fix@tighttimes 886 \newcommand*{\si@fix@period}{{.}}
\si@fix@tightcdot 887 \newcommand*{\si@fix@times}{\times}
888 \newcommand*{\si@fix@tighttimes}{\bgroup\times\egroup}
889 \newcommand*{\si@fix@tightcdot}{\bgroup\cdot\egroup}

```

```

\si@fix@plus   Signs for numbers are needed.
\si@fix@minus 890 \newcommand*{\si@fix@plus}{+}
\si@fix@pm 891 \newcommand*{\si@fix@minus}{-}
\si@fix@tightpm 892 \newcommand*{\si@fix@pm}{\pm}
\si@fix@mp 893 \newcommand*{\si@fix@tightpm}{\bgroup\pm\egroup}
894 \newcommand*{\si@fix@mp}{\mp}

\si@fix@two    The literals “2” and “10” are needed for exponents.
\si@fix@ten 895 \newcommand*{\si@fix@two}{2}
896 \newcommand*{\si@fix@ten}{10}

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

\si@fix@none   Finally for spacing, there is the possibility of nothing at all
898 \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.
899 \newcommand*{\si@symbol}[1]{%
900   \expandafter\protected\expandafter\def
901     \csname si@sym@#1\endcsname{%
902       \ifmmode
903         \expandafter\csname si@maths#1\expandafter\endcsname
904       \else
905         \expandafter\csname si@text#1\expandafter\endcsname
906       \fi}}

\si@sym@Omega  The various symbols are now declared.
\si@sym@ringA 907 \si@symbol{Omega}
\si@sym@mu 908 \si@symbol{ringA}
\si@sym@degree 909 \si@symbol{mu}
\si@sym@minute 910 \si@symbol{degree}
\si@sym@second 911 \si@symbol{minute}
\si@sym@celsius 912 \si@symbol{second}
913 \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.

```

914 \AtBeginDocument{%
915   \ifsi@redefsymbols
916     \@ifpackageloaded{textcomp}
917       {\si@log@debug{Redefining symbols using textcomp}%
918         \renewcommand*{\si@textdegree}{\textdegree}%
919         \renewcommand*{\si@mathsdegree}{\text{\textdegree}}}%
920     \@ifpackageloaded{mathptmx}{}
921     {\renewcommand*{\si@textmu}{\textmu}%
922       \renewcommand*{\si@textOmega}{\textohm}}%

```

mathptmx will give issues with textcomp and the Ω sign.

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.

```

923      \renewcommand*\si@tempa\{OT1}%
924      \ifx\si@tempa\encodingdefault
925        \renewcommand*\si@mathsringA\{%
926          \text{\capitalring{A}}}%
927        \renewcommand*\si@textringA\{\capitalring{A}}
928      \fi}%
929    \@ifpackageloaded{upgreek}
930      {\si@log@debug{Redefining symbols using upgreek}%
931        \renewcommand*\si@mathsOmega\{\Upomega}}{}
932    \fi}

```

21.8 Handling fractions

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

```

\si@frc@hook 933 \newcommand*\si@frc@frac[2]{%
\si@frc@frac 934   \ensuremath{\si@frc@hook\frac{%
\si@frc@slash 935     \expandafter\si@unt@out\expandafter{#1}}}%
\si@frc@nice 936     {\expandafter\si@unt@out\expandafter{#2}}}%
\si@frc@sfrac 937 \let\si@frac\si@frc@frac
938 \newcommand*\si@frc@hook{}
939 \newcommand*\si@frc@slash[2]{%
940   \expandafter\si@unt@out\expandafter{#1}%
941   \si@out{\ensuremath{\si@slash}}}%
942   \expandafter\si@unt@out\expandafter{#2}}
943 \newcommand*\si@frc@nice[2]{%
944   \ensuremath{\si@frc@nicefrac{\expandafter\si@unt@out%
945     \expandafter{#1}}{\expandafter\si@unt@out\expandafter
946       {#2}}}%
947 \newcommand*\si@frc@sfrac[2]{%
948   \sfrac{\expandafter\si@unt@out\expandafter{#1}}%
949     {\expandafter\si@unt@out\expandafter{#2}}}%
950 \AtBeginDocument{
951   \@ifpackageloaded{xfrac}
952     {}
953     {\si@log@inf{xfrac package unavailable\MessageBreak
954       using 'fraction=sfrac' will fall back on\MessageBreak
955       nicefrac-like method}%
956     \renewcommand*\si@frc@sfrac[2]{%
957       \si@log@warn{xfrac package unavailable}%
958       \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 959 \newlength\si@frc@displen
\si@frc@suplen 960 \newlength\si@frc@textlen
\si@frc@ssuplen 961 \newlength\si@frc@suplen
962 \newlength\si@frc@ssuplen

```

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

```

963 \newcommand*{\si@frc@nicefrac}{%
964   \ifmmode
965     \expandafter\si@frc@mathsnf
966   \else
967     \expandafter\si@frc@textnf
968   \fi}

\si@frc@mathsnf #1 : numerator
                  #2 : denominator
                  The maths mode system.
969 \newcommand*{\si@frc@mathsnf}[2]{%
970   \begingroup
971     \settoheight{\si@frc@displen}{\ensuremath{%
972       \displaystyle{M}}}%
973     \settoheight{\si@frc@textlen}{\ensuremath{%
974       \textstyle{M}}}%
975     \settoheight{\si@frc@suplen}{\ensuremath{%
976       \scriptstyle{M}}}%
977     \settoheight{\si@frc@ssuplen}{%
978       \ensuremath{\scriptscriptstyle{M}}}%
979     \addtolength{\si@frc@displen}{-\si@frc@ssuplen}%
980     \addtolength{\si@frc@textlen}{-\si@frc@ssuplen}%
981     \addtolength{\si@frc@suplen}{-\si@frc@ssuplen}%
982     \mathchoice
983       {\raisebox{\si@frc@displen}{\ensuremath{%
984         \scriptstyle{#1}}}}%
985       {\raisebox{\si@frc@textlen}{\ensuremath{%
986         \scriptstyle{#1}}}}%
987       {\raisebox{\si@frc@suplen}{%
988         \ensuremath{\scriptscriptstyle{#1}}}}%
989       {\raisebox{\si@frc@ssuplen}{%
990         \ensuremath{\scriptscriptstyle{#1}}}}%
991     \mkern-2mu\relax/\mkern-1mu\relax
992   \bgroup
993     \mathchoice
994       {\scriptstyle}%
995       {\scriptstyle}%
996       {\scriptscriptstyle}%
997       {\scriptscriptstyle}%
998     {#2}%
999   \egroup
1000 \endgroup}

\si@frc@textnf #1 : numerator
                 #2 : denominator
                 A stripped down version of the nicefrac system for text mode.
1001 \newcommand*{\si@frc@textnf}[2]{%
1002   \begingroup
1003     \settoheight{\si@frc@textlen}{M}%
1004     \settoheight{\si@frc@ssuplen}{\fontsize\sf@size\z@\relax
1005       \selectfont{M}}%
1006     \addtolength{\si@frc@textlen}{-\si@frc@ssuplen}%
1007     \raisebox{\si@frc@textlen}{\fontsize\sf@size\z@\relax
1008       \selectfont{#1}}%

```

```

1009     \hspace{-0.25ex}/\hspace{-0.25ex}%
1010     \hbox{\fontsize\sf@size\z@\selectfont{#2}}%
1011 \endgroup}

```

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

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

```

1012 \newcommand*\si@frc@ugly}[1]{%
1013   \renewcommand*\si@tempa{#1}%
1014   \ifmmode
1015     \expandafter\si@frc@frac
1016   \else
1017     \renewcommand*\si@tempb{1}%
1018     \ifx\si@tempa\si@tempb

```

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

```

1019       \setbox\si@tempboxa=\hbox{\ensuremath{\si@valuesep}}}%
1020       \hskip-\wd\si@tempboxa\relax
1021       \renewcommand*\si@tempa{}%
1022     \fi
1023     \expandafter\si@frc@slash
1024   \fi
1025   {\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, `\si@fam@tt` `fontspec`.

```

1026 \g@addto@macro{\document}{%
1027   \ifdefined\mathsf
1028     \setbox\si@tempboxa=\hbox{%
1029       $\mathsf{\global\chardef\si@fam@sf=\fam}$}%
1030   \else
1031     \si@log@inf{\string\mathsf not found}%
1032     \global\chardef\si@fam@sf=99\relax
1033   \fi
1034   \ifdefined\mathtt
1035     \setbox\si@tempboxa=\hbox{%
1036       $\mathtt{\global\chardef\si@fam@tt=\fam}$}%
1037   \else
1038     \si@log@inf{\string\mathtt not found}%
1039     \global\chardef\si@fam@tt=99\relax
1040   \fi}

```

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

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

```

1041 \newcommand*{\si@fam@ifbtext}[1]{%
1042   \if b\expandafter\@car\@f@series\@nil
1043     #1\fi}
1044 \newcommand*{\si@fam@ifbmaths}[1]{%
1045   \renewcommand*{\si@tempa}{bold}%
1046   \ifx\math@version\si@tempa
1047     #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.

```

1048 \newcommand*{\si@fam@ifbinline}{%
1049   \ifsi@inlinebtext
1050     \expandafter\si@fam@ifbtext
1051   \else
1052     \expandafter\si@fam@ifbmaths
1053   \fi}

```

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

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

```

1054 \newcommand*{\si@fam@ifitext}[1]{%
1055   \if n\expandafter\@car\@f@series\@nil\else
1056     #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.

```

1057 \newcommand*{\si@fam@mode}{%
1058   \ifsi@obeymode
1059     \ifmmode
1060       \sisetup{mode=maths}%
1061     \else
1062       \sisetup{mode=text}%
1063     \fi
1064   \fi}

```

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

```

1065 \AtBeginDocument{
1066   \@ifpackageloaded{color}
1067     {\let\si@fam@colourcmd\color}
1068     {\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.

```

1069 \newif\ifsi@fam@set
1070 \newif\ifsi@textmode

```

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

```

1071 \newcommand*{\si@fam@set}{%
1072   \ifsi@out@num
1073     \ifsi@numtextmode
1074       \expandafter\expandafter\expandafter\si@textmodetrue
1075     \else

```



```

1076     \expandafter\expandafter\expandafter\si@textmodefalse
1077     \fi
1078   \else
1079     \ifsi@unittextmode
1080       \expandafter\expandafter\expandafter\si@textmodetrue
1081     \else
1082       \expandafter\expandafter\expandafter\si@textmodefalse
1083     \fi
1084   \fi

\si@mathsrms The appropriate font macros are now established, if necessary.
\si@mathssfi1085 \ifsi@fam@set\else
\si@mathstt1086 \let\si@colourcmd\@gobble
\si@textrm1087 \ifsi@out@num
\si@textsf1088 \let\si@mathsrms\si@valuemathsrms
\si@textstt1089 \let\si@mathssf\si@valuemathssf
\si@texttt1090 \let\si@mathstt\si@valuemathstt
\si@colourcmd1091 \let\si@textrm\si@valuetextrm
\si@colour1092 \let\si@textsf\si@valuetextsf
1093 \let\si@texttt\si@valuetexttt
1094 \ifsi@colourvalues
1095   \let\si@colourcmd\si@fam@colourcmd
1096 \fi
1097 \let\si@colour\si@valuecolour
1098 \else
1099   \let\si@mathsrms\si@unitmathsrms
1100   \let\si@mathssf\si@unitmathssf
1101   \let\si@mathstt\si@unitmathstt
1102   \let\si@textrm\si@unittextrm
1103   \let\si@textsf\si@unittextsf
1104   \let\si@texttt\si@unittexttt
1105   \ifsi@colourunits
1106     \let\si@colourcmd\si@fam@colourcmd
1107   \fi
1108   \let\si@colour\si@unitcolour
1109 \fi
1110 \fi
1111 \si@fam@settrue

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

1112 \edef\si@tempa{\sfdefault}%
1113 \edef\si@tempb{\ttdefault}%

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

1114 \expandafter\let\expandafter\si@fam@maths
1115   \csname\si@mathsrms\endcsname
1116 \expandafter\let\expandafter\si@fam@text
1117   \csname\si@textrm\endcsname
1118 \ifsi@obeyfamily
1119   \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.

```

1120 \ifmmode
1121 \ifinner
1122 \si@log@debug{Font detection: inline maths}%
1123 \si@fam@detttext
1124 \else
1125 \si@log@debug{Font detection: display maths}%
1126 \ifsi@detectdisplay
1127 \si@fam@detmaths
1128 \else
1129 \si@fam@detttext
1130 \fi
1131 \fi
1132 \else
1133 \si@log@debug{Font detection: text}%
1134 \si@fam@detttext
1135 \fi
1136 \else
1137 \si@log@debug{Font detection: inactive}%
1138 \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 `Sistyle` 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.

```

1139 \def\si@fam@bold{\unboldmath\mdseries}%
1140 \ifsi@obeybold
1141 \si@log@debug{Weight detection: checking weight}%
1142 \ifmmode
1143 \ifdim\displaywidth>0pt\relax
1144 \ifsi@detectdisplay
1145 \expandafter\si@fam@ifbmaths
1146 \else
1147 \expandafter\si@fam@ifbtext
1148 \fi
1149 \si@fam@setbold
1150 \else
1151 \si@fam@ifbinline\si@fam@setbold
1152 \fi
1153 \else
1154 \si@fam@ifbtext\si@fam@setbold
1155 \fi
1156 \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.

```

1157 \let\si@fam@italic\upshape
1158 \ifsi@obeyitalic
1159 \si@log@debug{Italic detection: checking italic}%
1160 \si@fam@ifitext
1161 {\let\si@fam@italic\relax

```

```

1162         \si@log@debug{Italic detection: italic}}%
1163     \fi}

```

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

```

1164 \newcommand*{\si@fam@detmaths}{%
1165     \ifnum\the\fam=\si@fam@sf
1166         \si@log@debug{Font detection: sf}%
1167         \expandafter\let\expandafter\si@fam@maths
1168             \csname\si@mathssf\endcsname
1169         \expandafter\let\expandafter\si@fam@text
1170             \csname\si@textsf\endcsname
1171     \else
1172         \ifnum\the\fam=\si@fam@tt
1173             \si@log@debug{Font detection: tt}%
1174             \expandafter\let\expandafter\si@fam@maths
1175                 \csname\si@mathstt\endcsname
1176             \expandafter\let\expandafter\si@fam@text
1177                 \csname\si@texttt\endcsname
1178         \else
1179             \si@log@debug{Font detection: rm}%
1180             \expandafter\let\expandafter\si@fam@maths
1181                 \csname\si@mathsr\endcsname
1182             \expandafter\let\expandafter\si@fam@text
1183                 \csname\si@textrm\endcsname
1184         \fi
1185     \fi}
1186 \newcommand*{\si@fam@detttext}{%
1187     \ifx\f@family\si@tempa
1188         \si@log@debug{Font detection: sf}%
1189         \expandafter\let\expandafter\si@fam@maths
1190             \csname\si@mathssf\endcsname
1191         \expandafter\let\expandafter\si@fam@text
1192             \csname\si@textsf\endcsname
1193     \else
1194         \ifx\f@family\si@tempb
1195             \si@log@debug{Font detection: tt}%
1196             \expandafter\let\expandafter\si@fam@maths
1197                 \csname\si@mathstt\endcsname
1198             \expandafter\let\expandafter\si@fam@text
1199                 \csname\si@texttt\endcsname
1200         \else
1201             \si@log@debug{Font detection: rm}%
1202             \expandafter\let\expandafter\si@fam@maths
1203                 \csname\si@mathsr\endcsname
1204             \expandafter\let\expandafter\si@fam@text
1205                 \csname\si@textrm\endcsname
1206         \fi
1207     \fi}

```

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

```

\si@fam@boldify1208 \newcommand*{\si@fam@setbold}{%
1209     \si@log@debug{Weight detection: bold weight}%
1210     \let\si@fam@bold\si@fam@boldify}

```

```
1211 \newcommand*{\si@fam@boldify}{\boldmath\bfseries}
```

21.10 Formatting numbers

`\num` [#1]: keyval options

#2 : number

The system used here is modelled on that in `numprint`; the input is broken down into single tokens, each one is examined and the result is re-assembled into an output number. However, various changes have been made to the system used, and so the macros here are not simply renamed copies of those in `numprint`. The user macro `\num` sets any local keys, then calls the number formatting macro on the processed number.

```
1212 \si@newrobustcmd*{\num}[2][]{%
1213   \begingroup
1214     \sisetup{#1}%
1215     \si@fam@mode
1216     \si@num@intabfalse
1217     \si@log@debug{Processing \string\num\space input `#2'}%
1218     \expandafter\si@out@num\expandafter{\si@num{#2}}%
1219   \endgroup}
```

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

```
1220 \newif\ifsi@num@intab
```

`\si@num` #1 : number

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

```
1221 \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.

```
1222   \si@num@fixpm
1223   \begingroup
1224     \makeatletter
1225     \@makeother{\,%
1226     \@makeother{.}%
1227     \@makeother{+}%
1228     \@makeother{-}%
1229     \def~{}%
1230     \def\,%
1231     \catcode`\~= \active\relax
1232     \catcode`\^= \active\relax
1233     \everyeof{\noexpand}%
1234     \endlinechar\m@ne
1235     \protected@xdef\si@tempa{\scantokens{#1}}%
1236   \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.

```
1237 \si@ifnotmtarg{\si@tempa}
1238   {\si@num@ifvalid{\si@tempa}
1239    {\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.

```
1240   {\si@log@err{Invalid character '#1' in numerical input}%
1241    {Only characters from the list
1242     '\si@num@valid'\MessageBreak should be present in the
1243     argument of the \string\num\space macro\MessageBreak
1244     (or derivative such as an 's' column)}}%
1245    {#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.

```
\pm1246 \newcommand*{\si@num@fixpm}{%
\mp1247   \let\si@num@pm\pm
1248   \let\si@num@mp\mp
1249   \protected\def\pm{\si@num@pm}%
1250   \protected\def\mp{\si@num@mp}}
```

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

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

```
1251 \newcommand*{\si@num@ifvalid}[1]{%
1252   \begingroup
1253   \si@switchtrue
1254   \expandafter\si@num@valid#1\@empty\@empty
1255   \ifsi@switch
1256     \aftergroup\@firstoftwo
1257   \else
1258     \aftergroup\@secondoftwo
1259   \fi
1260   \endgroup}
1261 \def\si@num@valid#1#2\@empty{%
1262   \si@str@ifchrstr{#1}{\si@num@valid}
1263   {\ifx\@empty#2\@empty\else
1264    \si@num@valid#2\@empty\@empty\@empty
1265    \fi}
1266   {\si@switchfalse}}
```

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

```
\si@num@out1267 \newcommand*{\si@num@in}{}
\si@num@exp1268 \newcommand*{\si@num@out}{}
\si@num@expsign1269 \newcommand*{\si@num@exp}{}
\si@num@mant1270 \newcommand*{\si@num@expsign}{}
\si@num@mantsign1271 \newcommand*{\si@num@mant}{}
\si@num@err
\si@num@xpart
\si@num@ambig
\si@tab@out
\si@tab@expout
```

```

1272 \newcommand*{\si@num@mantsign}{}
1273 \newcommand*{\si@num@err}{}
1274 \newcommand*{\si@num@xpart}{}
1275 \newcommand*{\si@num@ambig}{}
1276 \newcommand*{\si@tab@out}{}
1277 \newcommand*{\si@tab@expout}{}

```

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

```

1278 \newif\ifsi@num@erropen

```

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

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

```

1279 \newcommand*{\si@num@arg}{}
1280 \newcommand*{\si@num@format}[1]{%
1281   \protected@edef\si@num@arg{#1}%
1282   \si@log@debug{Formatting number '\si@num@arg'}%

```

The storage areas are emptied.

```

1283 \renewcommand*{\si@num@in}{}%
1284 \renewcommand*{\si@num@exp}{}%
1285 \renewcommand*{\si@num@expsign}{}%
1286 \renewcommand*{\si@num@mant}{}%
1287 \renewcommand*{\si@num@mantsign}{}%
1288 \renewcommand*{\si@num@err}{}%
1289 \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`.

```

1290 \si@switchfalse
1291 \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`.

```

1292 \si@switchfalse
1293 \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}).

```

1294 \si@num@sepsign{exp}%
1295 \si@num@sepsign{mant}%
1296 \ifx\@empty\si@num@exp\@empty
1297   \ifx\@empty\si@num@expsign\@empty\else
1298     \si@log@warn{Sign but no number for '\si@num@arg'}%
1299   \fi
1300   \let\si@num@expsign\@empty
1301 \fi
1302 \ifx\@empty\si@num@mant\@empty
1303   \ifx\@empty\si@num@mantsign\@empty\else
1304     \ifx\@empty\si@num@exp\@empty
1305       \si@log@warn{Sign but no number for '\si@num@arg'}%
1306       \let\si@num@mantsign\@empty

```

```

1307     \fi
1308   \fi
1309 \fi

```

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

```

1310 \renewcommand*{\si@tempa}{\{-}\}%
1311 \ifx\si@num@mantsign\si@tempa
1312   \ifsi@colourneg
1313     \expandafter\expandafter\expandafter\si@fam@colourcmd
1314   \else
1315     \expandafter\expandafter\expandafter\@gobble
1316   \fi
1317 \else
1318   \expandafter\@gobble
1319 \fi
1320 {\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

```

1321 \si@num@procnum{exp}%
1322 \si@num@procnum{mant}%

```

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

```

1323 \si@str@ifonlychrs{\si@num@exp}{0\si@num@decimal}
1324 {\ifsi@allowzeroexp\else
1325   \renewcommand*{\si@num@exp}{}}%
1326 \ifx\@empty\si@num@mant\@empty
1327   \renewcommand*{\si@num@mant}{1}%
1328 \fi
1329 \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.

```

1330 \ifx\@empty\si@num@exp\@empty\else
1331   \ifsi@trapambigerr
1332     \expandafter\expandafter\expandafter\si@num@ambigertrue
1333   \fi
1334 \fi

```

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`. On the other hand, under normal circumstances the entire mantissa should be copied to `\si@num@out`.

```

1335 \protected@edef\si@num@out{%
1336   \ensuremath{\{\si@num@mantsign\}\si@num@mant}%
1337 \renewcommand*{\si@tempa}{num}%
1338 \ifsi@num@intab
1339   \protected@edef\si@tab@out{%

```

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

```

1340     \ensuremath{{\si@num@mantsign}}\si@num@predec}%
1341     \protected@edef\si@num@out{\si@num@postdec}%
1342     \renewcommand*{\si@tempa}{tab}%
1343 \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.

```

1344 \ifx\@empty\si@num@postdec\@empty\else
1345 \renewcommand*{\si@tempa}{num}%
1346 \fi
1347 \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.

```

1348 \ifsi@seperr
1349 \ifsi@num@ambigerr
1350 \protected@edef\si@num@out{%
1351 \ensuremath{\si@openerr}\si@num@out}%
1352 \si@repeatunitsfalse
1353 \expandafter\si@num@erropentrue
1354 \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.

```

1355 \ifsi@trapambigerr\else
1356 \ifx\@empty\si@num@exp\@empty\else
1357 \protected@edef\si@num@err{%
1358 \si@num@err\expandafter\car\si@num@exp\@nil
1359 \si@num@expsign\si@num@exp}%
1360 \ifsi@repeatunits\else
1361 \renewcommand*{\si@num@exp}{}%
1362 \renewcommand*{\si@num@expsign}{}%
1363 \fi
1364 \fi
1365 \fi
1366 \fi

```

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

```

1367 \else
1368 \expandafter\protected@edef\csname
1369 si@\si@tempa @out\endcsname{%
1370 \si@num@out\ensuremath{\si@errspace}\ensuremath
1371 {\si@openerr}\si@num@err\ensuremath{\si@closeerr}}%
1372 \renewcommand*{\si@num@err}{}%
1373 \fi
1374 \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.

```

1375 \renewcommand*{\si@tempa}{num@out}%
1376 \ifsi@num@erropen

```



```

1377 \renewcommand*{\si@tempa}{num@ambig}%
1378 \fi
1379 \ifsi@num@intab
1380 \renewcommand*{\si@tempa}{tab@expout}%
1381 \fi
1382 \ifx\@empty\si@num@exp\@empty
1383 \ifx\@empty\si@num@mant\@empty
1384 \si@log@err{Invalid number format '\si@num@arg'}
1385 {Something is wrong with the number format; does it
1386 contain \MessageBreak any numbers (from the list
1387 '\si@numdigits')}?}%
1388 \renewcommand*\si@num@out{}}%
1389 \fi
1390 \else
1391 \ifx\@empty\si@num@mant\@empty\else
1392 \expandafter\protected@edef\csname
1393 si@\si@tempa\endcsname{%
1394 \csname si@\si@tempa\endcsname\ensuremath{}}%
1395 \si@expproduct{}}}%
1396 \fi
1397 \expandafter\protected@edef\csname
1398 si@\si@tempa\endcsname{%
1399 \csname si@\si@tempa\endcsname\si@expbase
1400 \textsuperscript{\ensuremath{\si@num@expsign}}%
1401 \si@num@exp}}}%
1402 \fi

```

With everything done, the result is output.

```

1403 \ifsi@num@intab\else
1404 \expandafter\si@num@out
1405 \fi

```

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

```

1406 \ifx\@empty\si@num@err\@empty\else
1407 \expandafter\si@num@procerr
1408 \fi
1409 \ifsi@num@erropen
1410 \expandafter\si@out@num\expandafter{%
1411 \ensuremath{\si@closeerr}}%
1412 \ifx\@empty\si@num@ambig\@empty\else
1413 \expandafter\si@out@num\expandafter{\si@num@ambig}%
1414 \renewcommand*{\si@num@ambig}{}}%
1415 \fi
1416 \fi
1417 \si@num@erropenfalse
1418 \ifx\@empty\si@num@xpart\@empty\else
1419 \expandafter\si@num@sepxpart
1420 \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.

```

1421 \def\si@num@findxpart#1#2\@empty{%
1422 \si@str@ifchrstr{#1}{\si@numprod}

```

```

1423     {\si@switchtrue\si@seperrfalse}}}%
1424 \ifsi@switch
1425     \protected@edef\si@num@xpart{\si@num@xpart#1}%
1426 \else
1427     \protected@edef\si@num@in{\si@num@in#1}%
1428 \fi
1429 \ifx\@empty#2\@empty\else
1430     \si@num@findxpart#2\@empty
1431 \fi}

```

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

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

```

1432 \newcommand*{\si@num@sepmantexp}[1]{%
1433     \expandafter\si@num@mantexp#1\@empty\@empty}
1434 \def\si@num@mantexp#1#2\@empty{%
1435     \si@str@ifchrstr{#1}{\si@num@gobble}
1436     {\si@log@debug{Gobbling '#1' in \si@num@arg}}
1437     {\si@str@ifchrstr{#1}{\si@num@exp}
1438     {\ifsi@switch
1439         \si@log@err{Duplicate exponent marker found}
1440         {Only a single exponent character \MessageBreak
1441         (from the list '\si@num@exp')\MessageBreak may
1442         occur in a numerical argument}%
1443     \else
1444         \si@log@debug{Exponent marker '#1' found in
1445         '\si@num@arg'}%
1446     \fi
1447     \si@switchtrue}%
1448     {\ifsi@switch
1449         \expandafter\si@num@addexp
1450     \else
1451         \expandafter\si@num@addmnt
1452     \fi
1453     {#1}}}%

```

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

```

1454 \ifx\@empty#2\@empty
1455     \expandafter\@gobble
1456 \else
1457     \expandafter\si@num@sepmantexp
1458 \fi
1459 {#2}}

```

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

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

```

1460 \newcommand*{\si@num@addmnt}[1]{%
1461     \si@num@addmntexp{#1}{mant}{mantissa}}
1462 \newcommand*{\si@num@addexp}[1]{%
1463     \si@num@addmntexp{#1}{exp}{exponent}}

```

```

\si@num@addmntexp #1 : char
                  #2 : either mant or exp
                  #3 : info-text
                  Then the business end.
1464 \newcommand*{\si@num@addmntexp}[3]{%
1465   \si@log@debug{Adding '#1' to #3 for '\si@num@arg'}%
1466   \expandafter\protected@edef\csname si@num@#2\endcsname{%
1467     \csname si@num@#2\endcsname#1}}

\si@num@sepsign #1 : either mant or exp
                The input is now tested for a sign. If one exists, it is transferred into
                the \si@num@#1sign storage macro, with the remained of the number in
                \si@num@#1. The only check made directly here is that there is something
                to process.
1468 \newcommand*{\si@num@sepsign}[1]{%
1469   \expandafter\ifx\expandafter\@empty
1470     \csname si@num@#1\endcsname\@empty
1471     \expandafter\@gobble
1472   \else
1473     \expandafter\si@num@gensign
1474   \fi
1475   {#1}}

\si@num@gensign #1 : either mant or exp
                The sign generator starts by calling the procedure to check if the input contains
                a valid one- or two-digit sign. The results are returned as \si@num@sign and
                \si@num@value.
1476 \newcommand*{\si@num@gensign}[1]{%
1477   \expandafter\expandafter\expandafter\si@num@findsign
1478   \csname si@num@#1\endcsname\@empty\@empty

                If no sign has been found, then there may be a need to add one anyway.
1479   \ifx\@empty\si@num@sign\@empty
1480     \ifx\@empty\si@num@value\@empty
1481       \expandafter\expandafter\expandafter\@gobble
1482     \else
1483       \expandafter\expandafter\expandafter\si@num@addsign
1484     \fi
1485   \else
1486     \expandafter\@gobble
1487   \fi
1488   {#1}}%

                The appropriate storage areas are now assigned.
1489   \expandafter\let\csname si@num@#1sign\endcsname\si@num@sign
1490   \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.
1491 \newcommand*{\si@num@sign}{}

```

```

1492 \def\si@num@findsign#1#2#3\@empty{%
1493   \si@num@delplusfalse
1494   \si@str@ifchrstr{#1}{\si@numsign}{%
1495     \si@str@ifchrstr{#2}{\si@numsign}{%
1496       \if +#1%
1497         \if -#2%
1498           \si@log@debug{Found sign combination +- for
1499             '\si@num@arg'}%
1500           \renewcommand*\si@num@sign{{\si@pm}}%
1501         \else
1502           \si@log@inf{Unknown sign combination '#1#2'}%
1503           \renewcommand*\si@num@sign{{#1#2}}%
1504         \fi
1505       \else
1506         \if -#1%
1507           \if +#2%
1508             \si@log@debug{Found sign combination -+ for
1509               '\si@num@arg'}%
1510             \renewcommand*\si@num@sign{{\mp}}%
1511           \else
1512             \si@log@inf{Unknown sign combination '#1#2'}%
1513             \renewcommand*\si@num@sign{{#1#2}}%
1514           \fi
1515         \else
1516           \si@log@inf{Unknown sign combination '#1#2'}%
1517           \renewcommand*\si@num@sign{{#1#2}}%
1518         \fi
1519       \fi
1520     \protected@edef\si@num@value{#3}}%

```

Only one valid sign character.

```

1521   {\si@log@debug{Found single sign character '#1' for
1522     '\si@num@arg'}%
1523   \renewcommand*\si@num@sign{{#1}}%
1524   \if +#1%
1525     \ifsi@retainplus\else
1526       \expandafter\expandafter\expandafter\si@num@killsign
1527     \fi
1528   \fi
1529   \protected@edef\si@num@value{#2#3}}%

```

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

```

1530   {\si@log@debug{No sign found for '\si@num@arg'}%
1531   \renewcommand*\si@num@sign{}}%
1532   \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.

```

1533 \newif\ifsi@num@delplus
1534 \newcommand*\si@num@killsign{%
1535   \si@num@delplustrue
1536   \renewcommand*\si@num@sign{}}

```

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

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

```

1537 \newcommand*{\si@num@addsign}[1]{%
1538   \begingroup
1539     \renewcommand*{\si@tempa}{#1}%
1540     \renewcommand*{\si@tempb}{mant}%
1541     \ifx\si@tempa\si@tempb
1542       \aftergroup\@firstoftwo
1543     \else
1544       \aftergroup\@secondoftwo
1545     \fi
1546   \endgroup
1547   {\ifsi@num@signmant
1548     \expandafter\si@num@asign
1549   \else
1550     \expandafter\@gobble
1551   \fi
1552   {mantissa}}
1553   {\ifsi@num@signexp
1554     \expandafter\si@num@asign
1555   \else
1556     \expandafter\@gobble
1557   \fi
1558   {exponent}}
1559 \newcommand*{\si@num@asign}[1]{%
1560   \let\si@num@sign\si@sign
1561   \si@log@debug{Adding sign \si@sign\space to #1 for
1562     '\si@num@arg'}}
```

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

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

```

1563 \newcommand*{\si@num@procnun}[1]{%
1564   \expandafter\ifx\expandafter\@empty
1565     \csname si@num@#1\endcsname\@empty
1566   \expandafter\@gobble
1567   \else
1568     \expandafter\si@num@finddigits
1569   \fi
1570   {#1}}
```

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

```

\si@num@postdec1571 \newcommand*{\si@num@predec}{}
1572 \newcommand*{\si@num@postdec}{}

```

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

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

```

1573 \newcommand*{\si@num@finddigits}[1]{%
1574   \renewcommand*{\si@num@predec}{}%
1575   \renewcommand*{\si@num@postdec}{}%
1576   \si@switchfalse
```

```

1577 \expandafter\expandafter\expandafter\si@num@digits
1578 \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.

```

1579 \ifx\@empty\si@num@predec\@empty
1580 \ifsi@num@padlead
1581 \expandafter\expandafter\expandafter\si@num@addprezero
1582 \fi
1583 \fi
1584 \ifx\@empty\si@num@postdec\@empty
1585 \ifsi@num@padtrail
1586 \ifsi@switch
1587 \expandafter\expandafter\expandafter\expandafter
1588 \expandafter\expandafter\expandafter
1589 \si@num@addpostzero
1590 \fi
1591 \fi
1592 \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.

```

1593 \si@num@unsign{#1}%
1594 \ifx\@empty\si@num@predec\@empty
1595 \else
1596 \expandafter\si@num@nozero
1597 \fi

```

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

```

1598 \ifx\@empty\si@num@predec\@empty
1599 \ifx\@empty\si@num@postdec\@empty
1600 \expandafter\expandafter\expandafter\@gobble
1601 \else
1602 \expandafter\expandafter\expandafter\si@num@sepdigits
1603 \fi
1604 \else
1605 \expandafter\si@num@sepdigits
1606 \fi
1607 {#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.

```

1608 \def\si@num@digits#1#2\@empty{%
1609 \si@str@ifchrstr{#1}{\si@numdecimal}
1610 {\ifsi@switch
1611 \si@log@err{Duplicate decimal marker in '\si@num@arg'}
1612 {Only a single decimal marker (from the list
1613 '\si@numdecimal')\MessageBreak may occur in a
1614 numerical argument}%
1615 \else
1616 \si@log@debug{Found decimal marker '#1' in

```

```

1617         '\si@num@arg' }%
1618         \expandafter\si@switchtrue
1619     \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.

```

1620     {\si@str@ifchrstr{#1}{\si@numsign}
1621     {\si@log@err{Misplaced sign character
1622     '#1' in '\si@num@arg'}
1623     {Sign characters '\si@numsign' can only
1624     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.

```

1625     {\ifsi@switch
1626     \expandafter\si@num@post
1627     \else
1628     \expandafter\si@num@pre
1629     \fi
1630     {#1}}}%
1631 \ifx\@empty#2\@empty\else
1632 \si@num@digits#2\@empty\@empty
1633 \fi}

```

```

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

```

```

1634 \newcommand*{\si@num@pre}[1]{%
1635 \si@num@prepost{#1}{pre}{integer}}
1636 \newcommand*{\si@num@post}[1]{%
1637 \si@num@prepost{#1}{post}{decimal}}

```

```

\si@num@prepost #1 : char
#2 : either pre or post
#3 : info-text

```

Then actually stored here.

```

1638 \newcommand*{\si@num@prepost}[3]{%
1639 \expandafter\protected@edef\csname si@num@#2dec\endcsname{%
1640 \csname si@num@#2dec\endcsname#1}%
1641 \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@addpostzero1642 \newcommand*{\si@num@addprezero}{%
1643 \si@num@addpzero{pre}{leading}}
1644 \newcommand*{\si@num@addpostzero}{%
1645 \si@num@addpzero{post}{trailing}}

```

```

\si@num@addpzero #1 : either pre or post

```

In this case, there is no argument to be passed along.

```

1646 \newcommand*{\si@num@addpzero}[2]{%
1647 \si@log@debug{Adding #2 zero for '\si@num@arg'}%
1648 \@namedef{si@num@#1dec}{0}}

```

`\si@num@unsign` #1 : either mant or exp
`\si@num@nosign` The trap for a sign with zero numerical input is made. First, a check is made to see if there is a sign to worry about. The pre- and post-decimal parts are then examined, to see if they contain something other than “o” or an extra character.

```

1649 \newcommand*{\si@num@unsign}[1]{%
1650   \expandafter\ifx\expandafter\@empty
1651     \csname si@num@#1sign\endcsname\@empty
1652     \expandafter\@gobble
1653   \else
1654     \expandafter\si@num@nosign
1655   \fi
1656   {#1}}
1657 \newcommand*{\si@num@nosign}[1]{%
1658   \begingroup
1659     \si@switchtrue
1660     \si@str@ifonlychrs{\si@num@predec\si@num@postdec}{0}
1661     {\si@switchfalse}{}%
1662     \ifsi@switch
1663       \aftergroup\@gobble
1664     \else
1665       \aftergroup\@firstofone
1666     \fi
1667   \endgroup
1668   {\si@log@debug{Zero value: removing any sign}%
1669   \ifsi@ang@sign\else
1670     \@namedef{si@num@#1sign}{}%
1671   \fi}}

```

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

```

1672 \newcommand*{\si@num@nozero}{%
1673   \si@str@ifonlychrs{\si@num@predec}{0}
1674   {\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.

```

1675 \newcommand*{\si@num@decimalhook}{}

```

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

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

```

1676 \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.

```

1677   \begingroup
1678     \let\si@numextra\si@numaddn
1679     \protected@edef\si@tempa{\si@num@predec\si@num@postdec}%
1680     \si@num@ifextra{\si@tempa}
1681     {\aftergroup\@gobble}
1682     {\aftergroup\@firstofone}%
1683   \endgroup

```

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


```

1684 {\renewcommand*{\si@tempb}{mant}}%
1685 \renewcommand*{\si@tempc}{#1}%
1686 \ifx\si@tempb\si@tempc
1687 \expandafter\si@num@checkerr
1688 \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.

```

1689 \protected@edef\si@tempa{\si@num@predec\si@num@postdec}%
1690 \expandafter\si@str@ifonlychrs\expandafter{\si@tempa}
1691 {0123456789}
1692 {\ifx\@empty\si@num@err\@empty
1693 \ifsi@fixdp
1694 \expandafter\expandafter\expandafter\si@num@fixdp
1695 \fi
1696 \fi}}}%

```

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

```

1697 \si@num@ifextra{\si@num@predec}{}
1698 {\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.

```

1699 \renewcommand*{\si@tempc}{}%
1700 \ifx\@empty\si@num@postdec\@empty\else
1701 \si@num@decimalhook
1702 \renewcommand*{\si@tempc}{}%
1703 \ensuremath{{\si@decimalsymbol}}}%
1704 \si@num@ifextra{\si@num@postdec}{}
1705 {\expandafter\si@num@dec\expandafter{\si@num@postdec}}%
1706 \fi

```

The construction is finalised by re-combining the number.

```

1707 \expandafter\protected@edef\csname si@num#1\endcsname
1708 {\si@num@predec\si@tempc\si@num@postdec}}

```

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

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

```

1709 \newcommand*{\si@num@ifextra}[1]{%
1710 \begingroup
1711 \si@switchfalse
1712 \expandafter\si@num@extra#1\@empty\@empty
1713 \ifsi@switch
1714 \si@log@debug{Found ‘extra’ characters in ‘#1’}%
1715 \aftergroup\@firstoftwo
1716 \else
1717 \aftergroup\@secondoftwo
1718 \fi
1719 \endgroup}
1720 \def\si@num@extra#1#2\@empty{%
1721 \ifx\@empty#1\@empty\else
1722 \si@str@ifchrstr{#1}{\si@numextra}{\si@switchtrue}}%
1723 \ifx\@empty#2\@empty\else

```

```

1724     \si@num@extra#2\@empty\@empty
1725     \fi
1726     \fi}

\ifsi@num@ambigerr When separating out an error from a number, the first step is to see if there is a
\si@num@checkerr 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.

1727 \newif\ifsi@num@ambigerr
1728 \newcommand*{\si@num@checkerr}{%
1729     \ifx\@empty\si@num@postdec\@empty
1730     \expandafter\si@num@preerr
1731     \else
1732     \expandafter\si@num@posterr
1733     \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.

1734 \newcommand*{\si@num@preerr}{%
1735     \si@num@seperr{pre}%
1736     \ifx\@empty\si@tempb\@empty\else
1737     \expandafter\renewcommand\expandafter*\expandafter
1738     \si@num@err\expandafter{\si@tempb}%
1739     \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.

1740 \newcommand*{\si@num@posterr}{%
1741     \si@num@seperr{post}%
1742     \ifx\@empty\si@tempb\@empty\else
1743     \ifsi@seperr
1744     \expandafter\expandafter\expandafter\si@num@psterr
1745     \else
1746     \let\si@num@err\si@tempb
1747     \fi
1748     \fi}
1749 \newcommand*{\si@num@psterr}{%
1750     \si@num@cndigits{\si@tempb}%
1751     \si@tempcntb\si@tempcnta\relax
1752     \si@num@cndigits{\si@num@postdec}%
1753     \ifnum\si@tempcnta<\si@tempcntb\relax
1754     \expandafter\si@num@largeerr
1755     \else
1756     \expandafter\si@num@smallerr
1757     \fi}

\si@num@seperr #1 : either pre or post
\si@num@finderr The usual hand-off is made for searching for an error.

1758 \newcommand*{\si@num@seperr}[1]{%
1759     \si@switchfalse
1760     \renewcommand*{\si@tempa}{}%
1761     \renewcommand*{\si@tempb}{}%

```

```

1762 \expandafter\expandafter\expandafter\si@num@finderr
1763 \csname si@num@#1dec\endcsname\@empty\@empty
1764 \ifx\@empty\si@tempb\@empty\else
1765 \expandafter\let\csname si@num@#1dec\endcsname\si@tempa
1766 \fi}
1767 \def\si@num@finderr#1#2\@empty{%

```

First a check for the opening character of an error.

```

1768 \si@str@ifchrstr{#1}{\si@numopenerr}

```

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

```

1769 {\ifsi@switch
1770 \si@log@err{Invalid error in number}
1771 {The numerical argument \si@num@arg\space has two (or
1772 more)\MessageBreak error-opening characters}%
1773 \else
1774 \expandafter\si@switchtrue
1775 \fi}

```

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

```

1776 {\si@str@ifchrstr{#1}{\si@numcloseerr}
1777 {\ifsi@switch
1778 \ifx\@empty#2\@empty\else
1779 \si@log@err{Invalid error in number}
1780 {The numerical argument \si@num@arg\space has an
1781 error-closing before the last character}%
1782 \fi
1783 \else
1784 \si@log@err{Invalid error in number}
1785 {The numerical argument \si@num@arg\space has an
1786 error-closing character\MessageBreak but no
1787 error-opening one}%
1788 \fi}

```

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

```

1789 {\ifsi@switch
1790 \expandafter\si@num@addtmpb
1791 \else
1792 \expandafter\si@num@addtmpa
1793 \fi
1794 {#1}}}%
1795 \ifx\@empty#2\@empty\else
1796 \si@num@finderr#2\@empty
1797 \fi}

```

\si@num@addtmpa #1 : number

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

```

1798 \newcommand*{\si@num@addtmpa}[1]{\si@num@addtmp{a}{#1}}
1799 \newcommand*{\si@num@addtmpb}[1]{\si@num@addtmp{b}{#1}}

```

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

#2 : number

The addition is committed.

```

1800 \newcommand*{\si@num@addtmp}[2]{%

```

```

1801 \expandafter\protected@edef\csname si@temp#1\endcsname{%
1802 \csname si@temp#1\endcsname#2}}

\si@num@cntdigits #1 : number
\si@num@cntdgt A recursive system for counting the number of characters in a given input; only
used here to count digits in a number.

1803 \newcommand*{\si@num@cntdigits}[1]{%
1804 \si@tempcnta\z@ \relax
1805 \expandafter\si@num@cntdgt#1\@empty\@empty}
1806 \def\si@num@cntdgt#1#2\@empty{%
1807 \ifx\@empty#1\@empty\else
1808 \advance\si@tempcnta\@ne \relax
1809 \fi
1810 \ifx\@empty#2\@empty\else
1811 \expandafter\si@num@cntdgt#2\@empty
1812 \fi}

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

1813 \newcommand*{\si@num@smallerr}{%
1814 \si@tempcntb\si@tempcnta \relax
1815 \si@num@serr
1816 \protected@edef\si@num@err{%
1817 \ifsi@num@padlead0\fi \expandafter\@car\si@numdecimal\@nil
1818 \si@tempb}}
1819 \newcommand*{\si@num@serr}{%
1820 \si@num@cntdigits{\si@tempb}%
1821 \ifnum\si@tempcnta=\si@tempcntb \relax \else
1822 \protected@edef\si@tempb{0\si@tempb}%
1823 \expandafter\si@num@serr
1824 \fi}

\si@num@largeerr When the error has more digits than the decimal part, some shuffling is needed.
\si@num@lerr1825 \newcommand*{\si@num@largeerr}{%
\si@num@movedigit1826 \renewcommand*{\si@tempa}{}%
1827 \si@tempcntb\si@tempcnta \relax
1828 \si@num@lerr
1829 \protected@edef\si@num@err{%
1830 \si@tempa\ensuremath{\si@decimalsymbol}\si@tempb}}
1831 \newcommand*{\si@num@lerr}{%
1832 \si@num@cntdigits{\si@tempb}%
1833 \ifnum\si@tempcnta=\si@tempcntb \relax \else
1834 \expandafter\si@num@movedigit\si@tempb\@empty\@empty
1835 \si@num@lerr
1836 \fi}
1837 \def\si@num@movedigit#1#2\@empty{%
1838 \protected@edef\si@tempa{\si@tempa#1}%
1839 \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.

1840 \newcommand*{\si@num@fixdp}{%

```

```

1841 \si@num@cntdigits{\si@num@postdec}%
1842 \ifx\@empty\si@num@postdec\@empty
1843 \si@tempcnta\z@\relax
1844 \fi
1845 \ifnum\si@tempcnta>\si@num@dp\relax
1846 \expandafter\si@num@round
1847 \else
1848 \ifnum\si@tempcnta<\si@num@dp\relax
1849 \expandafter\expandafter\expandafter\si@num@pad
1850 \fi
1851 \fi}

```

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

```

1852 \newcommand*{\si@num@pad}{%
1853 \si@log@debug{Padding to \the\si@num@dp\space digits}%
1854 \loop\ifnum\si@tempcnta<\si@num@dp\si@num@pd\repeat}
1855 \newcommand*{\si@num@pd}{%
1856 \advance\si@tempcnta\@ne\relax
1857 \protected@edef\si@num@postdec{\si@num@postdec0}}

```

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

```

1858 \newcommand*{\si@num@prernd}{%
1859 \newcommand*{\si@num@postrnd}{%
1860 \newcommand*{\si@num@round}{%
1861 \si@log@debug{Rounding to \the\si@num@dp\space digits}%
1862 \si@num@reverse{\si@num@postdec}%
1863 \si@num@reverse{\si@num@predec}%
1864 \let\si@num@prernd\si@num@predec
1865 \let\si@num@postrnd\si@num@postdec
1866 \renewcommand*{\si@num@predec}{}%
1867 \renewcommand*{\si@num@postdec}{}%
1868 \si@switchfalse
1869 \si@num@rnd}

```

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

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

```

1870 \newcommand*{\si@num@reverse}[1]{%
1871 \renewcommand*{\si@tempa}{}%
1872 \expandafter\si@num@rev#1\@empty\@empty
1873 \let#1\si@tempa}
1874 \def\si@num@rev#1#2\@empty{%
1875 \edef\si@tempa{#1\si@tempa}%
1876 \ifx\@empty#2\@empty\else
1877 \si@num@rev#2\@empty\@empty
1878 \fi}

```

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

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

This is very similar to the same routines used in other macro packages to achieve the same thing.

```

1879 \newcommand*{\si@num@rnd}{%
1880   \ifnum\si@tempcnta>\z@\relax
1881     \expandafter\si@num@rndpost
1882   \else
1883     \expandafter\si@num@rndpre
1884   \fi}
1885 \newcommand*{\si@num@rndpre}{%
1886   \expandafter\edef\expandafter\si@tempa\expandafter{%
1887     \expandafter\@car\si@num@prernd\@nil}%
1888   \expandafter\edef\expandafter\si@num@prernd\expandafter{%
1889     \expandafter\@cdr\si@num@prernd\@nil}%
1890   \si@tempcntb\si@tempa\relax
1891   \ifsi@switch
1892     \advance\si@tempcntb\@ne\relax
1893   \fi
1894   \si@switchfalse
1895   \ifnum\si@tempcntb=10\relax
1896     \si@tempcntb\z@\relax
1897     \expandafter\expandafter\expandafter\si@switchtrue
1898   \fi
1899   \edef\si@num@predec{\the\si@tempcntb\si@num@predec}%
1900   \ifx\@empty\si@num@prernd\@empty
1901     \ifsi@switch
1902       \edef\si@num@predec{1\si@num@predec}%
1903     \fi
1904   \else
1905     \expandafter\si@num@rnd
1906   \fi}
1907 \newcommand*{\si@num@rndpost}{%
1908   \expandafter\edef\expandafter\si@tempa\expandafter{%
1909     \expandafter\@car\si@num@postrnd\@nil}%
1910   \expandafter\edef\expandafter\si@num@postrnd\expandafter{%
1911     \expandafter\@cdr\si@num@postrnd\@nil}%
1912   \si@tempcntb\si@tempa\relax
1913   \ifsi@switch
1914     \advance\si@tempcntb\@ne\relax
1915   \fi
1916   \si@switchfalse
1917   \advance\si@num@dp\@ne\relax
1918   \ifnum\si@tempcnta>\si@num@dp\relax
1919     \advance\si@num@dp\m@ne\relax
1920   \else
1921     \advance\si@num@dp\m@ne\relax
1922     \ifnum\si@tempcnta>\si@num@dp\relax
1923       \ifnum\si@tempcntb>4\relax
1924         \expandafter\expandafter\expandafter\si@switchtrue
1925       \fi
1926     \else
1927       \ifnum\si@tempcntb=10\relax
1928         \si@tempcntb\z@\relax
1929         \expandafter\expandafter\expandafter\si@switchtrue
1930       \fi

```

```

1931      \edef\si@num@postdec{\the\si@tempcntb\si@num@postdec}%
1932      \fi
1933  \fi
1934  \advance\si@tempcnta\m@ne\relax
1935  \si@num@rnd}

```

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

The formatting code for separating thousands is taken more-or-less directly from `Slstyle`. 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.

```

1936 \newcommand*{\si@num@int}[1]{%
1937   \renewcommand*{\si@num@predec}{}%
1938   \ifsi@sepfour
1939     \si@num@intfmt{ }#1\@empty\@empty\@empty
1940   \else
1941     \si@num@iffive{#1}
1942     {\si@num@intfmt{ }#1\@empty\@empty\@empty}
1943     {\renewcommand*{\si@num@predec}{#1}}%
1944   \fi}

```

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

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

```

1945 \newcommand*{\si@num@iffive}[1]{%
1946   \si@num@five#1\@empty\@empty\@empty\@empty\@empty\end}
1947 \def\si@num@five#1#2#3#4#5\end{%
1948   \ifx\@empty#5\@empty
1949     \expandafter\@secondoftwo
1950   \else
1951     \expandafter\@firstoftwo
1952   \fi}

```

`\si@num@intfmt` The business end of the integer formatter.

```

\si@num@fiint1953 \newcommand*{\si@num@intfmt}[4]{%
1954   \ifx\@empty#2\@empty
1955     \si@num@intsep#1\relax
1956   \else
1957     \ifx\@empty#3\@empty
1958       \si@num@intsep\@empty\@empty#1#2\relax
1959     \else
1960       \ifx\@empty#4\@empty
1961         \si@num@intsep\@empty#1#2#3\relax
1962       \else
1963         \si@num@fiint{#1#2#3#4}%
1964       \fi
1965     \fi
1966   \fi}
1967 \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.

```

1968 \newcommand*{\si@num@intsep}[4]{%
1969   \protected@edef\si@num@predec{\si@num@predec#1#2#3}%
1970   \if\relax#4\relax\else

```

```

1971 \protected@edef\si@num@predec{%
1972 \si@num@predec\ensuremath{\noexpand\si@digitsep}}%
1973 \expandafter\si@num@intsep\expandafter#4%
1974 \fi}

```

`\si@num@dec` #1 : decimal-part
`\si@num@decfmt` Formatting a decimal uses a similar mechanism, but with a few alterations needed.

```

1975 \newcommand*{\si@num@dec}[1]{%
1976 \renewcommand*{\si@num@postdec}{}%
1977 \ifsi@sepfour
1978 \si@num@decfmt#1\@empty\@empty\@empty\@empty
1979 \else
1980 \si@num@iffive{#1}
1981 {\si@num@decfmt#1\@empty\@empty\@empty\@empty}
1982 {\protected@edef\si@num@postdec{\si@num@postdec#1}}%
1983 \fi}
1984 \newcommand*{\si@num@decfmt}[4]{%
1985 \protected@edef\si@num@postdec{\si@num@postdec#1#2#3}%
1986 \ifx\@empty#4\@empty%
1987 \else
1988 \protected@edef\si@num@postdec{%
1989 \si@num@postdec\ensuremath{\noexpand\si@digitsep}}%
1990 \expandafter\si@num@decfmt\expandafter#4%
1991 \fi}

```

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

```

1992 \newcommand*{\si@num@procerr}{%
1993 \si@num@addunit
1994 \ensuremath{\si@pm}%
1995 \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.

```

1996 \newcommand*{\si@num@sepxpart}{%
1997 \si@num@addunit
1998 \ensuremath{\times}%
1999 \expandafter\expandafter\expandafter\si@num\expandafter
2000 \expandafter\expandafter{%
2001 \expandafter\@cdr\si@num@xpart\@nil}}

```

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

```

2002 \newcommand*{\si@num@addunit}{%
2003 \si@unt@numtrue
2004 \ifx\@empty\si@unt@unitarg\@empty\else
2005 \ifsi@repeatunits
2006 \si@unt@printunit{\si@unt@unitarg}%
2007 \fi
2008 \fi}

```


21.11 Formatting angles

`\ang` [#1]: keyval options

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

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

```

2009 \si@newrobustcmd*{\ang}[2][]{%
2010   \begingroup
2011     \sisetup{#1}%
2012     \si@fam@mode
2013     \si@log@debug{Processing \string\ang\space input `#2'}%
2014     \@makeother{\;}%
2015     \makeatletter
2016     \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.

```

2017 \def\si@ang@parse#1;#2;#3;#4\@nil{%
2018   \let\ifsi@ang@fixdp\ifsi@fixdp
2019   \si@fixdpfalse
2020   \si@ifmtarg{#4}
2021   {\si@log@debug{Angle argument contains no
2022     semi-colons:\MessageBreak decimal angle}%
2023     \si@ang@dec{#1}}{}}
2024   {\si@log@debug{Angle argument contains
2025     semi-colons:\MessageBreak degree-minute-second angle}%
2026     \renewcommand*{\si@tempa}{#4}%
2027     \renewcommand*{\si@tempb}{;}%
2028     \ifx\si@tempa\si@tempb\else
2029       \ifsi@strictarc
2030         \renewcommand*{\si@tempb}{;}%
2031         \ifx\si@tempa\si@tempb
2032           \si@log@err{Insufficient semi-colons in argument
2033             of \string\ang}{The argument of
2034             \string\ang\space must contain either no
2035             semi-colons or exactly two}%
2036         \else
2037           \si@log@err{Excess semi-colons in argument of
2038             \string\ang}{The argument of \string\ang\space
2039             must contain either no semi-colons or exactly
2040             two}%
2041         \fi
2042       \fi
2043     \fi
2044     \si@ang@arc{#1}{#2}{#3}}

```

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

```

\si@ang@arc2045 \newcommand*{\si@ang@dec}{%
2046   \let\si@ang@fix\@gobble
2047   \ifsi@ang@toarc
2048     \expandafter\si@ang@dectoarc
2049   \else
2050     \sisetup{padangle=none}\expandafter\si@ang@typeset

```

```

2051 \fi}
2052 \newcommand*{\si@ang@arc}{%
2053 \let\si@ang@fix\si@ang@arcfix
2054 \ifsi@ang@todec
2055 \expandafter\si@ang@arctodec
2056 \else
2057 \expandafter\si@ang@typeset
2058 \fi}

\ifsi@ang@fixdp #1 : number
\si@ang@fix A check is needed so that rounding and zero filling are only applied to the
\si@ang@arcfix seconds of an arc angle.

2059 \newif\ifsi@ang@fixdp
2060 \newcommand*{\si@ang@fix}[1]{%
2061 \newcommand*{\si@ang@arcfix}[1]{%
2062 \renewcommand*{\si@tempa}{second}%
2063 \renewcommand*{\si@tempb}{#1}%
2064 \ifx\si@tempa\si@tempb
2065 \ifsi@ang@fixdp
2066 \expandafter\expandafter\expandafter\si@fixdptrue
2067 \else
2068 \expandafter\expandafter\expandafter\si@fixdpfalse
2069 \fi
2070 \else
2071 \expandafter\si@fixdpfalse
2072 \fi}

\si@ang@ifnum #1 : number
A test is required to check that the provided data consists of numbers which can
actually be processed by TeX. This is achieved by using \si@num@ifvalid with
a fixed list of valid characters.

2073 \newcommand*{\si@ang@ifnum}[1]{%
2074 \begingroup
2075 \renewcommand*{\si@num@valid}{0123456789,.-+}%
2076 \ifx\@empty#1\@empty
2077 \aftergroup\@firstoftwo
2078 \else
2079 \si@num@ifvalid{#1}
2080 {\aftergroup\@firstoftwo}
2081 {\aftergroup\@secondoftwo}%
2082 \fi
2083 \endgroup}

\si@ang@arctodec #1 : degrees
#2 : minutes
#3 : seconds
The business end of converting arcs to decimal angles is relatively straight-
forward, 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.

2084 \newcommand*{\si@ang@arctodec}[3]{%
2085 \let\si@ang@fix\@gobble
2086 \ifnum\si@num@dp>\thr@@\relax

```

```

2087 \si@num@dp\thr@@\relax
2088 \fi
2089 \si@fixdptrue
2090 \si@ang@ifnum{#1}
2091   {\si@ang@ifnum{#2}
2092    {\si@ang@ifnum{#3}
2093     {\si@tempdima\z@\relax
2094      \renewcommand*{\si@tempa}{+}%
2095      \ifx\@empty#1\@empty\else
2096       \si@tempdima #1pt\relax
2097     \fi
2098     \ifdim\si@tempdima<\z@\relax
2099      \renewcommand*{\si@tempa}{-}%
2100     \fi
2101     \ifx\@empty#2\@empty\else
2102      \si@tempdima\dimexpr\si@tempdima\si@tempa
2103       #2pt/60\relax
2104     \fi
2105     \ifdim\si@tempdima<\z@\relax
2106      \renewcommand*{\si@tempa}{-}%
2107     \else
2108     \fi
2109     \ifx\@empty#3\@empty\else
2110      \si@tempdima\dimexpr\si@tempdima\si@tempa
2111       #3pt/3600\relax
2112     \fi
2113     \sisetup{numdecimal=.%}
2114     \expandafter\si@ang@typeset\expandafter{%
2115       \strip@pt\si@tempdima}{\{ \} \} \}
2116     {\si@ang@notnum{#1}{#2}{#3}}
2117     {\si@ang@notnum{#1}{#2}{#3}}
2118     {\si@ang@notnum{#1}{#2}{#3}}

```

\si@ang@dectoarc #1 : number

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

```

2119 \newcommand*{\si@ang@dectoarc}[1]{%
2120   \let\si@ang@fix\si@ang@arcfix
2121   \si@ang@fixdptrue
2122   \ifnum\si@num@dp>\@ne\relax
2123     \si@num@dp\@ne\relax
2124   \fi
2125   \si@ang@ifnum{#1}
2126     {\si@tempdima\z@\relax
2127      \ifx\@empty#1\@empty\else
2128       \si@tempdima #1pt\relax
2129      \fi
2130      \si@ang@sepint{deg}%
2131      \si@tempdima\dimexpr\si@tempdima *60\relax
2132      \si@ang@sepint{min}%
2133      \edef\si@tempa{\the\dimexpr\si@tempdima *60\relax}%
2134      \expandafter\newcommand\expandafter*\expandafter{%
2135        \expandafter\si@ang@arcsec\expandafter}\expandafter{%

```

```
2136 \expandafter\si@ang@strippt\si@tempa}%
```

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

```
2137 \si@tempdima\z@\relax
2138 \edef\si@tempa{\the\si@tempdima}%
2139 \expandafter\renewcommand\expandafter*\expandafter{%
2140 \expandafter\si@tempa\expandafter}\expandafter{%
2141 \expandafter\si@ang@strippt\si@tempa}%
2142 \ifx\si@tempa\si@ang@arcsec
2143 \renewcommand*\si@ang@arcsec{0}%
2144 \fi
```

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

```
2145 \renewcommand*\si@tempa{0}%
2146 \ifx\si@ang@arcdeg\si@tempa
2147 \si@temptoks{}}}%
2148 \else
2149 \si@temptoks{{\si@ang@arcdeg}}}%
2150 \fi
2151 \ifx\si@ang@arcmin\si@tempa
2152 \si@temptoks\expandafter{\the\si@temptoks{}}}%
2153 \else
2154 \si@temptoks\expandafter{\the\si@temptoks{
2155 \si@ang@arcmin}}}%
2156 \fi
2157 \ifx\si@ang@arcsec\si@tempa
2158 \si@temptoks\expandafter{\the\si@temptoks{}}}%
2159 \else
2160 \si@temptoks\expandafter{\the\si@temptoks{
2161 \si@ang@arcsec}}}%
2162 \fi
2163 \expandafter\si@ang@typeset\the\si@temptoks}
2164 {\si@ang@notnum{#1}}{}}{}}}
```

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

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

```
\si@ang@strippt2165 \newcommand*\si@ang@sepint}[1]{%
2166 \expandafter\si@ang@sint\the\si@tempdima\@empty
2167 \expandafter\let\csname si@ang@arc#1\endcsname\si@tempa}
2168 \def\si@ang@sint#1.#2\@empty{%
2169 \renewcommand*\si@tempa{#1}%
2170 \si@tempdima 0.#2\relax}
2171 \begingroup
2172 \catcode`P=12
2173 \catcode`T=12
2174 \lowercase{
2175 \renewcommand*\si@tempa{%
2176 \def\si@ang@strippt##1PT{##1}}}%
2177 \expandafter\endgroup
2178 \si@tempa
```

`\si@ang@notnum` #1 : degrees
#2 : minutes
#3 : seconds

Not a TeX number: complain.

```
2179 \newcommand*{\si@ang@notnum}[3]{%
2180   \si@log@warn{Angle '#1;#2;#3' is not a pure
2181     number:\MessageBreak output will be as given}%
2182   \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.

```
2183 \newif\ifsi@ang@sign
```

```
\si@ang@typeset #1 : degrees
                #2 : minutes
                #3 : seconds
```

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

```
2184 \newcommand*{\si@ang@typeset}[3]{%
```

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

```
\si@ang@mins2185 \ifsi@ang@padlarge
\si@ang@secs2186   \newcommand*{\si@ang@deg}{0\si@sym@degree}%
2187   \newcommand*{\si@ang@min}{0\si@sym@minute}%
2188   \newcommand*{\si@ang@sec}{0\si@sym@second}%
2189   \else
2190   \newcommand*{\si@ang@deg}{}%
2191   \newcommand*{\si@ang@min}{}%
2192   \newcommand*{\si@ang@sec}{}%
2193   \fi
```

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

```
2194 \protected@edef\si@ang@decimalsymbol{\si@decimalsymbol}%
```

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

```
2195 \ifsi@astroang
2196   \let\si@ang@movesign\si@ang@astrosign
2197 \else
2198   \let\si@ang@movesign\@gobble
2199 \fi
```

`\si@ang@secnum` The arguments are now examined in reverse order. If they are empty, then 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.

`\si@ang@minnum`

```
2200 \newcommand*{\si@ang@secnum}{\si@ang@num{second}}%
2201 \newcommand*{\si@ang@minnum}{\si@ang@num{minute}}%
2202 \si@ifnotmtarg{#3}
2203   {\si@log@debug{Found seconds '#3'}}%
2204   \si@ang@ifnum{#3}
2205     {\ifdim #3 pt=\z@\relax\else
2206       \si@ang@signtrue
2207     \fi}}}%
2208 \renewcommand*{\si@ang@secs}
2209   {\si@ang@secnum{#3}\si@sym@second}%
2210 \renewcommand*{\si@ang@mins}
```

```

2211      {\si@ang@pad{0\si@sym@minute}}%
2212      \renewcommand*{\si@ang@deg}
2213      {\si@ang@pad{0\si@sym@degree}}}%
2214      \si@ifnotmtarg{#2}
2215      {\si@log@debug{Found minutes `#2'}%
2216      \si@ang@ifnum{#2}
2217      {\ifdim #2 pt=\z@\relax\else
2218      \si@ang@signtrue
2219      \fi}}}%
2220      \renewcommand*{\si@ang@secnum}{%
2221      \si@ang@signlessnum{second}}}%
2222      \renewcommand*{\si@ang@mins}
2223      {\si@ang@minnum{#2}\si@sym@minute}%
2224      \renewcommand*{\si@ang@deg}
2225      {\si@ang@pad{0\si@sym@degree}}}%
2226      \si@ifnotmtarg{#1}
2227      {\si@log@debug{Found degrees `#1'}%
2228      \renewcommand*{\si@ang@secnum}{%
2229      \si@ang@signlessnum{second}}}%
2230      \renewcommand*{\si@ang@minnum}{%
2231      \si@ang@signlessnum{minute}}}%
2232      \renewcommand*{\si@ang@deg}

```

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

```

2233      {\si@ang@num{degree}{#1}%
2234      \si@sym@degree}}}%
2235      \si@out@num
2236      {\si@ang@deg\si@anglesep\si@ang@mins\si@anglesep
2237      \si@ang@secs}%

```

The group opened by `\ang` is closed.

```

2238      \endgroup

```

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

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

```

2239 \newcommand*{\si@ang@pad}[1]{\ifsi@ang@padsmall #1\fi}

```

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

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

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

```

2240 \newcommand*{\si@ang@num}[2]{%
2241   \begingroup
2242     \si@ang@fix{#1}%
2243     \si@ang@movesign{#1}%
2244     \si@num{#2}%
2245   \endgroup}
2246 \newcommand*{\si@ang@signlessnum}[2]{%
2247   \begingroup
2248     \si@ang@fix{#1}%
2249     \si@ang@movesign{#1}%
2250     \sisetup{addsign=none}%
2251     \si@num{#2}%

```

```

2252 \endgroup}

\si@ang@killdegree  A mechanism is needed to handle moving the angle unit signs for the astroang
\si@ang@killminute  option. First, some support macros are needed.
\si@ang@killsecond2253 \newcommand*{\si@ang@killdegree}{\let\si@sym@degree\relax}
2254 \newcommand*{\si@ang@killminute}{\let\si@sym@minute\relax}
2255 \newcommand*{\si@ang@killsecond}{\let\si@sym@second\relax}

\si@ang@astrosign #1 : one of degree, minute or second
The method needs two steps, producing the sign over the decimal sign and
preventing duplicate symbols appearing. This is based on a suggestion from
Morten Høgholm, but using TeX internals as \makebox does not work here.
Note the need to correct for \scriptspace (thanks to Donald Arseneau for
that).

2256 \newcommand*{\si@ang@astrosign}[1]{%
2257 \renewcommand*{\si@decimalsymbol}{%
2258 \setbox\si@tempboxa=\hbox{%
2259 \ensuremath{\{\si@ang@decimalsymbol\}}}%
2260 \si@tempdima\wd\si@tempboxa\relax
2261 \setbox\si@tempboxb=\hbox to\z@{%
2262 \hss\unhbox\si@tempboxa\hss}%
2263 \setbox\si@tempboxa=\hbox{%
2264 \csname si@sym@#1\endcsname\hskip-\scriptspace}%
2265 \si@tempdimb\wd\si@tempboxa\relax
2266 \setbox\si@tempboxc=\hbox to\z@{%
2267 \hss\unhbox\si@tempboxa\hss}%
2268 \setbox\si@tempboxd=\hbox{%
2269 \usebox\si@tempboxb\usebox\si@tempboxc}%
2270 \ifdim\si@tempdima>\si@tempdimb\relax
2271 \setbox\si@tempboxa=\hbox to\si@tempdima{%
2272 \hss\unhbox\si@tempboxd\hss}%
2273 \else
2274 \setbox\si@tempboxa=\hbox to\si@tempdimb{%
2275 \hss\unhbox\si@tempboxd\hss}%
2276 \fi
2277 \usebox\si@tempboxa%
2278 \ifdim\si@tempdima>\si@tempdimb\relax\else
2279 \hskip\scriptspace
2280 \fi}%
2281 \renewcommand*{\si@num@decimalhook}{\expandafter\aftergroup
2282 \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.

```
2283 \edef\si@tempa{%
2284   \noexpand\NC@do S\noexpand\NC@do s\the\NC@list}
2285 \newcolumnntype{S}{}
2286 \newcolumnntype{s}{}
2287 \NC@list\expandafter{\si@tempa}
```

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

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

```
2288 \renewcommand*{\NC@rewrite@S}[1][{}]{%
2289   \edef\si@tempa{\the\@temptokena
2290     >\noexpand\si@tab@begin@S[#1]}c%
2291     <\noexpand\si@tab@end@S}%
2292   \@temptokena\expandafter{\si@tempa}%
2293   \NC@find}
2294 \renewcommand*{\NC@rewrite@s}[1][{}]{%
2295   \edef\si@tempa{\the\@temptokena
2296     >\noexpand\si@tab@begin@s[#1]}c%
2297     <\noexpand\si@tab@end@s}%
2298   \@temptokena\expandafter{\si@tempa}%
2299   \NC@find}
```

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

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

\si@tab@gettok

```
2300 \newcommand*{\si@tab@begin@S}[1][{}]{%
2301   \si@log@debug{Processing S column cell contents}%
2302   \let\si@tab@gettok\si@tab@gettok@S
2303   \si@seperrfalse
2304   \renewcommand*{\si@num@sepxpart}{%
2305     \si@log@err{Multiple numbers not allowed in
2306       tables\MessageBreak Only the first number used}
2307     \@ehb}%
2308   \si@tab@begin[#1]}
2309 \newcommand*{\si@tab@begin@s}[1][{}]{%
2310   \si@log@debug{Processing s column cell contents}%
2311   \let\si@tab@gettok\si@tab@gettok@s
2312   \si@tab@begin[#1]}
```

\si@tab@toks

\si@tab@pretoks

\si@tab@posttoks

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

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

```

2316 \newcommand*{\si@tab@begin}[1][\{%
2317   \begin{group}
2318     \sisetup{#1}%
2319     \si@tab@toks{}%
2320     \si@tab@pretoks{}%
2321     \si@tab@posttoks{}%
2322     \si@switchfalse
2323     \si@tab@gettok}

```

<pre>\si@tab@othertok \si@tab@gettok@s \si@tab@next</pre>	<p>The two collection macros are very similar. Both compare the current input with a list of possible fixed values; this is pretty much a direct copy of numprint. If the input is not on the list of choices, it is processed as data for siunitx to handle. The 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.</p>
---	--

113

```

2352             \si@tab@othertok{#1}}%
2353         \fi
2354     \fi
2355 \fi
2356 \fi
2357 \fi
2358 \fi
2359 \si@tab@next}
2360 \newcommand*{\si@tab@othertok}[1]{%
2361 \ifsi@switch
2362     \si@tab@posttoks=\expandafter{\the\si@tab@posttoks#1}%
2363 \else
2364     \si@tab@pretoks=\expandafter{\the\si@tab@pretoks#1}%
2365 \fi}
2366 \newcommand*{\si@tab@gettok@s}[1]{%
2367 \ifx\tabularnewline#1\relax
2368     \let\si@tab@next\si@tab@newline@s
2369 \else
2370     \ifx\end#1\relax
2371         \let\si@tab@next\end
2372     \else
2373         \ifx\si@tab@end@s#1\relax
2374             \let\si@tab@next\si@tab@end@s
2375         \else
2376             \ifx\endtabular#1\relax
2377                 \let\si@tab@next\endtabular
2378             \else
2379                 \ifx\csize#1\relax
2380                     \let\si@tab@next\csize
2381                 \else
2382                     \ifx\relax#1\relax
2383                         \let\si@tab@next\relax
2384                     \else
2385                         \let\si@tab@next\si@tab@gettok@s
2386                     \ifx\ignorespaces#1\relax\else
2387                         \ifx\unskip#1\relax\else
2388                             \si@tab@toks=\expandafter{%
2389                                 \the\si@tab@toks#1}%
2390                             \si@log@debug{Found cell contents `#1'}%
2391                         \fi
2392                     \fi
2393                 \fi
2394             \fi
2395         \fi
2396     \fi
2397 \fi
2398 \fi
2399 \si@tab@next}

```

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

```

2400 \newcommand*{\si@tab@end@S}{%
2401     \ifsi@switch
2402         \let\si@tab@lfill\si@tab@lfill@S
2403         \let\si@tab@rfill\si@tab@rfill@S
2404     \else
2405         \let\si@tab@rfill\si@tab@rfill@t
2406         \let\si@tab@lfill\si@tab@lfill@t
2407     \fi
2408     \si@tab@lfill\relax
2409     \ifsi@switch
2410         \llap{\the\si@tab@pretoks}%
2411         \expandafter\si@tab@numout
2412         \rlap{\the\si@tab@posttoks}%
2413     \else
2414         \the\si@tab@pretoks
2415     \fi
2416     \si@tab@rfill\relax
2417 \endgroup}
2418 \newcommand*{\si@tab@end@s}{%
2419     \si@tab@lfill@s\relax
2420     \ignorespaces
2421     \expandafter\si\expandafter{\the\si@tab@toks}%
2422     \unskip
2423     \si@tab@rfill@s\relax
2424 \endgroup}

```

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

```

2425 \newcommand*{\si@tab@newline@S}{%
2426     \si@tab@end@S
2427     \hfil\relax
2428     \let\si@tab@end\si@tab@end@S
2429     \renewcommand*{\si@tab@end@S}{\let\si@tab@end@S\si@tab@end}%
2430     \tabularnewline}
2431 \newcommand*{\si@tab@newline@s}{%
2432     \si@tab@end@s
2433     \hfil\relax
2434     \let\si@tab@end\si@tab@end@s
2435     \renewcommand*{\si@tab@end@s}{\let\si@tab@end@s\si@tab@end}%
2436     \tabularnewline}

```

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

```

2437 \newcount\si@tempcnta
2438 \newcount\si@tempcntb

```

`\si@tab@numout` If a number is found, then some secondary processing is needed to format it correctly.

```

2439 \newcommand*{\si@tab@numout}{%

```

```

2440 \si@num@intabtrue
2441 \ifsi@tab@fixed
2442   \ifsi@tabautofit
2443     \si@num@dp\si@tab@mantpostcnt\relax
2444     \expandafter\expandafter\expandafter\si@fixdptrue
2445   \fi
2446 \fi
2447 \expandafter\si@num\expandafter{\the\si@tab@toks}%
2448 \si@tab@format}

```

`\si@tab@prebox` The various boxes needed for the column centring are declared Unlike the `dc` column original, private boxes are used here. `\si@tempboxa` is used when a space to measure one of the constituents is needed; it is never used for output.

```

\si@tab@postbox
\si@tab@midbox
\si@tab@expbox
2449 \newbox\si@tab@prebox
2450 \newbox\si@tab@midbox
2451 \newbox\si@tab@postbox
2452 \newbox\si@tab@expbox

```

`\si@tab@format` The formatting set up is taken from `dc` column, but with control of the output form the stored information. The choice of a variable (decimal-centred) column or fixed width boxes is made.

```

2453 \newcommand*{\si@tab@format}{%
2454   \ifsi@tab@fixed
2455     \expandafter\si@tab@fixed
2456   \else
2457     \expandafter\si@tab@unfixed
2458   \fi

```

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

```

2459   \ifsi@colourvalues
2460     \si@fam@colourcmd{\si@valuecolour}%
2461   \fi
2462   \box\si@tab@prebox\box\si@tab@midbox\box\si@tab@postbox%
2463   \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.

```

2464 \newcommand*\si@tab@unfixed{%
2465   \si@log@debug{Using variable width S column}%
2466   \protected@edef\si@num@out{\si@num@out\si@tab@expout}%
2467   \setbox\si@tab@prebox=\hbox
2468     {\expandafter\si@out@num\expandafter{\si@tab@out}}%
2469   \ifx\@empty\si@num@out\@empty
2470     \setbox\si@tab@midbox=\hbox
2471       {\phantom{\ensuremath{\{\si@decimalsymbol\}}}}%
2472   \else
2473     \setbox\si@tab@midbox=\hbox
2474       {\ensuremath{\{\si@decimalsymbol\}}}%
2475   \fi
2476   \setbox\si@tab@postbox=\hbox
2477     {\expandafter\si@out@num\expandafter{\si@num@out}}%
2478   \ifdim\wd\si@tab@prebox>\wd\si@tab@postbox\relax

```

```

2479 \setbox\si@tab@postbox=\hbox to\wd\si@tab@prebox%
2480 {\unhbox\si@tab@postbox\hfill}%
2481 \else
2482 \setbox\si@tab@prebox=\hbox to\wd\si@tab@postbox%
2483 {\hfill\unhbox\si@tab@prebox}%
2484 \fi
2485 \setbox\si@tab@expbox=\hbox{}

```

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

```

\si@tab@postdim2486 \newdimen\si@tab@predim
\si@tab@expdim2487 \newdimen\si@tab@postdim
\si@tempdima2488 \newdimen\si@tab@expdim
\si@tempdimb2489 \newdimen\si@tempdima
2490 \newdimen\si@tempdimb

```

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

```

2491 \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.

```

2492 \newcommand*{\si@tab@fixed}{%
2493 \si@log@debug{Using fixed-width S column}%
2494 \let\si@tab@sp\relax
2495 \setbox\si@tab@midbox=\hbox{}%
2496 \setbox\si@tab@expbox=\hbox{}%
2497 \setbox\si@tempboxa=\hbox{\si@out@num{1}}%
2498 \si@tempdima\wd\si@tempboxa\relax

```

The width for the two output boxes is set up.

```

2499 \si@tab@predim\the\si@tab@mantprecnt\si@tempdima\relax
2500 \si@tab@sepcorr{mantpre}{pre}%
2501 \si@tab@postdim\si@tab@mantpostcnt\si@tempdima\relax
2502 \setbox\si@tempboxa=\hbox{\ensuremath{\{\si@decimalsymbol\}}}%
2503 \advance\si@tab@postdim\wd\si@tempboxa\relax
2504 \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.

```

2505 \setbox\si@tempboxa=\hbox{\si@out@num^{1}}%
2506 \si@tempdima\wd\si@tempboxa\relax
2507 \ifnum\si@tab@expprecnt>z@\relax
2508 \si@tab@expdim\si@tab@expprecnt\si@tempdima\relax
2509 \si@tab@sepcorr{exppre}{exp}%
2510 \fi
2511 \let\si@tab@sp\sp
2512 \ifnum\si@tab@exppostcnt>z@\relax
2513 \advance\si@tab@expdim\si@tab@exppostcnt\si@tempdima\relax
2514 \setbox\si@tempboxa=\hbox{%
2515 \ensuremath{\^{ \si@decimalsymbol}}}%
2516 \advance\si@tab@expdim\wd\si@tempboxa\relax
2517 \si@tab@sepcorr{exppost}{exp}%
2518 \fi

```

Space is reserved for signs.

```

2519 \setbox\si@tempboxa=\hbox{\ensuremath{-}}%
2520 \ifsi@tab@mantsign
2521 \advance\si@tab@predim\wd\si@tempboxa\relax
2522 \fi
2523 \setbox\si@tempboxa=\hbox{\ensuremath{^{\{-}}}%
2524 \ifsi@tab@expsign
2525 \advance\si@tab@expdim\wd\si@tempboxa\relax
2526 \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.

```

2527 \si@tempcnta\si@tab@expprecnt\relax
2528 \advance\si@tempcnta\si@tab@exppostcnt\relax
2529 \ifnum\si@tempcnta>\z@\relax
2530 \setbox\si@tempboxa=\hbox{\ensuremath{%
2531 {} \si@expproduct{} \si@expbase}}%
2532 \advance\si@tab@expdim\wd\si@tempboxa\relax
2533 \fi

```

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

```

2534 \ifsi@tab@alignexp\else
2535 \advance\si@tab@postdim\si@tab@expdim\relax
2536 \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.

```

2537 \setbox\si@tab@prebox=\hbox to\si@tab@predim{\hss\hfill
2538 \expandafter\si@out@num\expandafter{\si@tab@out}}%
2539 \ifx\@empty\si@num@out\@empty
2540 \setbox\si@tab@postbox=\hbox to\si@tab@postdim
2541 {\expandafter\si@out@num\expandafter{\si@num@out}\hfil}%
2542 \else
2543 \ifsi@tab@alignexp\else
2544 \protected@edef\si@num@out{\si@num@out\si@tab@expout}%
2545 \fi
2546 \setbox\si@tab@postbox=\hbox to\si@tab@postdim
2547 {\ensuremath{\{\si@decimalsymbol\}}\expandafter\si@out@num
2548 \expandafter{\si@num@out}\hfil}%
2549 \fi
2550 \ifx\@empty\si@tab@expout\@empty
2551 \ifsi@tab@alignexp
2552 \setbox\si@tab@expbox=\hbox to\si@tab@expdim{\hfil}%
2553 \fi
2554 \else
2555 \ifsi@tab@alignexp
2556 \setbox\si@tab@expbox=\hbox to\si@tab@expdim
2557 {\expandafter\si@out@num\expandafter{\si@tab@expout}%
2558 \hfil}%
2559 \fi
2560 \fi}

```

`\si@tab@sepcorr #1 : one of mantpre, mantpost, exppre or exppost`

#2 : one of pre, post or exp

A spacing correction is needed *if* the number of digits to be allowed for will lead to the introduction of a separator. A counter and dimension are needed for the testing.

```
2561 \newcommand*{\si@tab@sepcorr}[2]{%
2562   \expandafter\si@tempcnta\expandafter\the
2563     \csname si@tab@#1cnt\endcsname\relax
2564   \divide\si@tempcnta\thr@@\relax
2565   \ifsi@sepfour\else
2566     \expandafter\ifnum\expandafter\the
2567       \csname si@tab@#1cnt\endcsname=4\relax
2568     \si@tempcnta\z@\relax
2569   \fi
2570 \fi
```

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

```
2571 \setbox\si@tempboxa=\hbox{%
2572   \ensuremath{\si@tab@sp{\si@digitsep}}}%
2573 \expandafter\advance\csname si@tab@#2dim\endcsname
2574   \si@tempcnta\wd\si@tempboxa}
```

21.13 Units

`\SI` [#1]: keyval options

#2 : number

There are two types of user macros for the units system; those for defining new units, prefixes and powers, and those for using them. There are two macros for using units, `\SI` and `\si`, which work in very similar ways. argument to `\SI`

```
2575 \si@newrobustcmd*{\SI}[2][[]]{%
2576   \@ifnextchar[%
2577     {\si@SI[#1]{#2}}
2578     {\si@SI[#1]{#2}[]}}
```

`\si` [#1]: keyval options

#2 : unit

`\si` is just an alias for `\SI` with no number; everything is handed off into an internal macro. The internal macro also handles the optional prefix

```
2579 \si@newrobustcmd*{\si}[2][[]]{\si@SI[#1]{}[]}{#2}}
```

`\newunit` [#1]: keyval options

`\renewunit` #2 : unit

`\provideunit` #3 : symbol

The `\newunit` and `\renewunit` macros create the new unit macros. To allow a mechanism for checking an existing definition, these macros simply carry out the appropriate tests, before handing off to the internal macro. `\@ifdefinable` is not used here as a customised error is desirable. Other than that, the code here gives very similar results to `\newcommand` and `\renewcommand`. Finally, `\provideunit` adds the unit definition only if it does not already exist.

```

2580 \newcommand*{\newunit}[3][\%
2581   \si@ifdefinable{#2}
2582     {\si@unt@defunit[#1]{#2}{#3}}
2583     {\si@log@err{Unit \string#2 already defined!}\@eha}}
2584 \newcommand*{\renewunit}[3][\%
2585   \si@ifdefinable{#2}
2586     {\si@log@err{Unit \string#2 undefined}\@ehc
2587     \si@unt@defunit[#1]{#2}{#3}}
2588     {\si@log@inf{Redefining unit \string#2}%
2589     \si@unt@defunit[#1]{#2}{#3}}}}
2590 \newcommand*{\provideunit}[3][\%
2591   \si@ifdefinable{#2}
2592     {\si@unt@defunit[#1]{#2}{#3}}
2593     {}}

```

```

\newprefix [#1]: binary
\renewprefix #2 : multiple
\provideprefix #3 : powers-ten
               #4 : symbol

```

The multiples of units are defined here; very similar code is used to the `\newunit`, *etc.*, macros. The multiple prefixes cannot take an optional argument, and must represent some power. Hence the arguments required are different.

```

2594 \newcommand*{\newprefix}[4][\%
2595   \si@ifdefinable{#2}
2596     {\si@unt@defprefix[#1]{#2}{#3}{#4}}
2597     {\si@log@err{Prefix \string#2 already defined!}\@eha}}
2598 \newcommand*{\renewprefix}[4][\%
2599   \si@ifdefinable{#2}
2600     {\si@log@err{Prefix \string#2 undefined}\@ehc
2601     \si@unt@defprefix[#1]{#2}{#3}{#4}}
2602     {\si@log@inf{Redefining prefix \string#2}%
2603     \si@unt@defprefix[#1]{#2}{#3}{#4}}}}
2604 \newcommand*{\provideprefix}[4][\%
2605   \si@ifdefinable{#2}
2606     {\si@unt@defprefix[#1]{#2}{#3}{#4}}
2607     {}}

```

```

\newpower [#1]: post
\renewpower #2 : number
\providepower #3 : power

```

Here power multiples for units are set up. As with units and multiples, a layered approach is used to keep things easy to maintain.

```

2608 \newcommand*{\newpower}[3][\%
2609   \si@ifdefinable{#2}
2610     {\si@unt@defpower[#1]{#2}{#3}}
2611     {\si@log@err{Power \string#2 already defined!}\@eha}}
2612 \newcommand*{\renewpower}[3][\%
2613   \si@ifdefinable{#2}
2614     {\si@log@err{Power \string#2 undefined}\@ehc
2615     \si@unt@defpower[#1]{#2}{#3}}
2616     {\si@log@inf{Redefining power \string#2}%
2617     \si@unt@defpower[#1]{#2}{#3}}}}

```



```

2618 \newcommand*{\providepower}[3][[]]{%
2619   \si@ifdefinable{#2}
2620   {\si@unt@defpower[#1]{#2}{#3}}
2621   {}}

```

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

```

2622 \newif\ifsi@unt@num

```

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

```

2623 \newcommand*{\si@unt@unitarg}{}

```

`\si@SI [#1] :` keyval options

`\si@unt@SIopts` #2 : unit

[#3] : preunit

#4 : unit

The internal processing starts with `\si@SI`, which processes the second optional argument to `\SI` (which is empty for `\si`). Everything is set up in a group, and processing begins by handling the options.

```

2624 \newcommand*{\si@unt@SIopts}{}
2625 \def\si@SI[#1]#2[#3]#4{%
2626   \begingroup
2627     \let\fg\SIfg
2628     \si@ifnotmtarg{#1}
2629     {\sisetup{#1}%
2630      \renewcommand*{\si@unt@SIopts}{#1}}%

```

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

```

2631   \si@unt@numfalse
2632   \si@xspacefalse
2633   \si@ifnotmtarg{#3}
2634   {\si@log@debug{Prefix unit found}%
2635    \si@unt@printunit{#3}}%

```

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

```

2636   \si@ifnotmtarg{#4}
2637   {\renewcommand*{\si@unt@unitarg}{#4}}%
2638   \si@ifnotmtarg{#2}
2639   {\si@log@debug{Number found in \string\SI\space
2640    argument}%
2641    \ifsi@repeatunits\else
2642      \ifsi@trapambigerr
2643        \expandafter\expandafter\expandafter
2644        \si@num@ambigertrue
2645      \fi
2646      \fi
2647      \num{#2}%
2648      \si@unt@numtrue}%

```

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

```

2649   \si@ifnotmtarg{#4}
2650   {\si@ifmtarg{#2}

```

```

2651      {\si@unt@printunit{#4}}
2652      {\si@tempcnta\z@}\relax
2653      \ifsi@addunitpower
2654        \si@unt@countx{#2}%
2655      \fi
2656      \ifnum\si@tempcnta>\z@}\relax
2657        \advance\si@tempcnta\@ne\relax
2658        \edef\si@tempa{\noexpand\tothe{\si@tempcnta}}%
2659        \renewcommand*{\si@tempb}{#4}%
2660        \expandafter\expandafter\expandafter
2661        \si@unt@printunit\expandafter\expandafter
2662        \expandafter{%
2663          \expandafter\si@tempb\si@tempa}%
2664      \else
2665        \si@unt@printunit{#4}%
2666      \fi}}%
2667 \endgroup}

```

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

```

2668 \newcommand*{\si@unt@countx}[1]{%
2669   \si@tempcnta\z@}\relax
2670   \expandafter\si@unt@cntx#1\@empty\@empty}
2671 \def\si@unt@cntx#1#2\@empty{%
2672   \si@str@ifchrstr{#1}{\si@numprod}
2673   {\advance\si@tempcnta\@ne\relax}
2674   {}}%
2675   \ifx\@empty#2\@empty\else
2676     \si@unt@cntx#2\@empty\@empty
2677   \fi}

```

`\si@unt@ifliteral` #1 : unit

`\ifsi@unt@littest` The next stage of the processor is to determine whether or not the argument of the unit macro is processable. For literal arguments, this is not the case, and the argument is typeset “as is”. On the other hand, any units, *etc.*, declared by the package will work with the processor, and so need to be executed before typesetting the result.

```

2678 \newif\ifsi@unt@littest
2679 \newcommand*{\si@unt@ifliteral}[1]{%
2680   \begingroup
2681     \si@unt@littesttrue

```

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

```

2682     \setbox\si@tempboxa=\hbox{\si@unt@out{#1}}%
2683     \ifdim\wd\si@tempboxa>\z@}\relax
2684       \aftergroup\@firstoftwo
2685     \else
2686       \aftergroup\@secondoftwo
2687     \fi
2688   \endgroup}

```

`\ifsi@unt@litout` #1 : unit

`\si@unt@printunit` The printing macro uses the above test to determine how to act. It then carries out the appropriate action: either typesetting or executing. A flag is also provided

so that any macro units inside a partially-literal argument will work (this is also needed to emulate `unitsdef`).

```
2689 \newif\ifsi@unt@litout
2690 \newcommand*{\si@unt@printunit}[1]{%
2691   \si@unt@ifliteral{#1}
```

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

```
2692   {\si@log@debug{Literal items found in unit
2693     argument:\MessageBreak outputting without further
2694     processing}%
2695   \si@unt@litouttrue
2696   \si@unt@addvaluesep
2697   \si@unt@out{#1}}
```

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

```
2698   {\si@log@debug{Macro unit found:\MessageBreak
2699     processing to format output}%
2700   \si@unt@init
2701   \advance\si@unt@depthcnt\@ne\relax
2702   #1%
2703   \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@litvalsep2704 \newcommand*{\si@unt@addvaluesep}{%
\si@unt@stackvalsep2705   \ifsi@unt@num
2706     \expandafter\si@unt@addvalsep
2707   \fi}
2708 \newcommand*{\si@unt@addvalsep}{%
2709   \ifsi@unt@litout
2710     \expandafter\si@unt@litvalsep
2711   \else
2712     \expandafter\si@unt@stackvalsep
2713   \fi}
2714 \newcommand*{\si@unt@stackvalsep}{%
2715   \protected@edef\si@unt@spstack{\si@valuesep}}
2716 \newcommand*{\si@unt@litvalsep}{%
2717   \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@depthcnt2718 \newcommand*{\si@unt@spstack}{}
\ifsi@unt@first2719 \newcommand*{\si@unt@stacka}{}
\ifsi@unt@first2720 \newcommand*{\si@unt@stackb}{}
\si@unt@init2721 \newcount\si@unt@unitcnta
2722 \newcount\si@unt@unitcntb
2723 \newcount\si@unt@depthcnt
2724 \newif\ifsi@unt@first
2725 \si@unt@depthcnt\m@ne\relax
```

```

2726 \newcommand*{\si@unt@init}{%
2727   \beginngroup
2728     \si@unt@litoutfalse
2729     \si@unt@litprefixfalse
2730     \si@unt@firsttrue
2731     \si@unt@perfalse
2732     \si@unt@perseenfalse
2733     \si@unt@prepowerfalse
2734     \si@unt@depthcnt\z@\relax
2735     \si@unt@powerdim\z@\relax
2736     \si@unt@unitcnta\z@\relax
2737     \si@unt@unitcntb\z@\relax
2738     \si@unt@prefixcnt\z@\relax
2739     \renewcommand*{\si@unt@spstack}{}%
2740     \renewcommand*{\si@unt@stacka}{}%
2741     \renewcommand*{\si@unt@stackb}{}%
2742     \renewcommand*{\si@unt@holdstacka}{}%
2743     \renewcommand*{\si@unt@holdstackb}{}%
2744     \renewcommand*{\si@unt@lastadda}{space}%
2745     \renewcommand*{\si@unt@lastaddb}{space}}

```

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

```

2746 \newcommand*{\si@unt@final}{%
2747   \si@unt@third
2748   \si@unt@stackout
2749   \endgroup
2750   \ifsi@xspace
2751     \expandafter\expandafter\expandafter\xspace
2752   \fi}

```

`\si@unt@defunit` [#1]: keyval options

#2 : unit
#3 : symbol

The internal macro for defining a unit does not check for redefinition; that is done by the user macros.

```

2753 \newcommand*{\si@unt@defunit}[3][{}]{%
2754   \si@log@debug{Declaring unit \string#2 with \MessageBreak
2755     meaning \string#3}%

```

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

```

2756   \si@ifnotmtarg{#1}
2757   {\expandafter\@namedef\expandafter{%
2758     \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.

```

2759   \protected\def#2{%
2760     \ifsi@allowoptarg
2761       \expandafter\si@unt@withopt
2762     \else
2763       \expandafter\si@unt@noopt
2764     \fi
2765     {#2}{#3}}

```

```
\si@unt@withopt #1 : unit
\si@unt@noopt #2 : symbol
```

To allow the correct expansion, the potential optional argument to a unit macro has to come last. So `\@ifnextchar` is needed to detect it and pass data through. To keep variation down, when the argument is not allowed, the empty `[]` is supplied.

```
2766 \newcommand*{\si@unt@withopt}[2]{%
2767   \@ifnextchar[%
2768     {\si@unt@opt{#1}{#2}}
2769     {\si@unt@opt{#1}{#2}[]}}
2770 \newcommand*{\si@unt@noopt}[2]{\si@unt@opt{#1}{#2}[]}
```

```
\si@unt@opt #1 : unit
#2 : symbol
[#3]: number
```

The optional argument to the unit macro (if present) is converted to a normal one for ease. The correct route for processing is then picked.

```
2771 \def\si@unt@opt#1#2[#3]{%
2772   \ifsi@unt@littest
2773     \expandafter\si@gobblethree
2774   \else
2775     \ifsi@unt@litout
2776       \expandafter\expandafter\expandafter\@gobbletwo
2777     \else
2778       \expandafter\expandafter\expandafter\si@unt@unit
2779     \fi
2780   \fi
2781   {#3}{#1}{#2}}
```

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

```
\si@gobblethree LATEX does not have a \@gobblethree macro, but one is needed.
```

```
2782 \long\def\si@gobblethree #1#2#3{}
```

```
\si@unt@defprefix [#1]: binary
\ifsi@unt@litprefix #2 : multiple
#3 : powers-ten
#4 : symbol
```

As with units, multiples are defined by an internal macro.

```
2783 \newif\ifsi@unt@litprefix
2784 \si@unt@litprefixtrue
2785 \newcommand*{\si@unt@defprefix}[4][]{%
2786   \si@log@debug{Declaring multiple \string#1 with\MessageBreak
2787     meaning \string#4}%
```

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

```
2788   \expandafter\expandafter\expandafter\def\expandafter
2789     \csname\expandafter\@gobble\string#2@opt@si@endcsname{#1}%
2790   \protected\def#2{%
2791     \ifsi@unt@littest
2792       \expandafter\si@gobblethree
2793     \else
```

```

2794     \ifsi@unt@litout
2795     \expandafter\expandafter\expandafter\@gobbletwo
2796   \else
2797     \ifsi@unt@litprefix
2798     \expandafter\expandafter\expandafter\expandafter
2799     \expandafter\expandafter\expandafter\@gobbletwo
2800   \else
2801     \expandafter\expandafter\expandafter\expandafter
2802     \expandafter\expandafter\expandafter\si@unt@prefix
2803   \fi
2804 \fi
2805 \fi
2806 {#2}{#3}{#4}}

```

```

\si@unt@defpower [#1]: post
#2 : number
#3 : power

```

The definition of powers is complicated by the need to handle both those given before units (such as `\cubic`) and those given after (e.g. `\cubed`). This means that an optional argument is needed.

```

2807 \newcommand*{\si@unt@defpower}[3][{}]{%
2808   \si@log@debug{Declaring power \string#2 with\MessageBreak
2809     meaning \string#3}%

```

Once again the optional argument is saved.

```

2810   \expandafter\expandafter\expandafter\def\expandafter
2811     \csname\expandafter\@gobble\string#2@opt@si\endcsname{#1}%
2812   \protected\def#2{%
2813     \ifsi@unt@littest
2814       \expandafter\@gobbletwo
2815     \else

```

The literal output here does not need to gobble anything.

```

2816     \ifsi@unt@litout
2817     \expandafter\expandafter\expandafter\si@unt@litpower
2818   \else
2819     \expandafter\expandafter\expandafter\si@unt@power
2820   \fi
2821 \fi
2822 {#2}{#3}}

```

```

\si@unt@unithook #1 : number
\si@unt@unit     #2 : unit
                 #3 : symbol

```

The macro for units is actually a processor, rather than typesetting anything, which is handled elsewhere. The first argument to the macro is optional, but does not have square brackets to keep things simple with gobbling.

```

2823 \newcommand*{\si@unt@unithook}{}
2824 \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.

```

2825   \ifnum\si@unt@depthcnt=\m@ne\relax
2826     \expandafter\si@unt@init

```

```

2827 \fi
2828 \advance\si@unt@depthcnt\@ne\relax
2829 \si@log@debug{Unit processing: level \the\si@unt@depthcnt,
2830 \MessageBreak unit \string#2}%
2831 \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.

```

2832 \si@unt@ifliteral{#3}
2833 {\si@unt@addtostack{unit}{#3}%
2834 \ifsi@unt@prepower
2835 \expandafter\si@unt@stkpower
2836 \fi}
2837 {#3}%

```

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

```

2838 \advance\si@unt@depthcnt\m@ne\relax
2839 \ifnum\si@unt@depthcnt=\z@\relax
2840 \expandafter\si@unt@final
2841 \fi}

```

```
\si@unt@firstorsecond #1 : number
```

```
#2 : unit-macro
```

At this stage, the flag will be set for the first item to be processed whichever route the unit has been called by.

```

2842 \newcommand*{\si@unt@firstorsecond}[2]{%
2843 \ifsi@unt@first
2844 \expandafter\si@unt@first
2845 \else
2846 \expandafter\si@unt@second
2847 \fi
2848 {#1}{#2}}%

```

```
\si@unt@first #1 : number
```

```
#2 : unit-macro
```

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

```

2849 \newcommand*{\si@unt@first}[2]{%
2850 \si@ifnotmtarg{#1}
2851 {\num{#1}%
2852 \si@unt@numtrue}%
2853 \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.

```

2854 \ifcsname\expandafter\@gobble\string#2\opt@unt@si\endcsname
2855 \expandafter\si@unt@setopts
2856 \else
2857 \expandafter\@gobble
2858 \fi
2859 {#2}%

```

```

2860 \si@unt@addvaluesep
2861 \si@unt@firstfalse}

```

`\si@unt@setopts` #1 : unit
`\si@unt@setSIopts` A rather long set of `\expandafter` commands to get the options to set safely.

```

2862 \newcommand*{\si@unt@setopts}[1]{%
2863   \expandafter\expandafter\expandafter\expandafter\expandafter
2864     \expandafter\expandafter\si@temptoks\expandafter
2865     \expandafter\expandafter\expandafter\expandafter
2866     \expandafter\expandafter{\expandafter%
2867       \csname\expandafter\@gobble\string#1\opt@unt\si%
2868       \endcsname}%
2869   \expandafter\si@setup\expandafter{\the\si@temptoks}%
2870   \si@log@debug{Applying options '\the\si@temptoks'}
2871   for\MessageBreak unit \string#1}%

```

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

```

2872 \@ifundefined{si@unt@SIopts}{}
2873   {\ifx\@empty\si@unt@SIopts\@empty\else
2874     \expandafter\expandafter\si@unt@setSIopts
2875     \fi}}
2876 \newcommand*{\si@unt@setSIopts}{}
2877 \expandafter\si@temptoks\expandafter{\si@unt@SIopts}%
2878 \expandafter\si@setup\expandafter{\the\si@temptoks}}

```

`\si@unt@second` #1 : number
`\si@unt@third` #2 : unit

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

```

2879 \newcommand*{\si@unt@second}[2]{%
2880   \si@ifnotmtarg{#1}
2881     {\si@log@warn{Optional argument to unit macro\MessageBreak
2882       allowed only for outer unit}}}%
2883   \si@unt@third
2884   \si@unt@addtostack{space}{\ensuremath{\si@unitsep}}}
2885 \newcommand*{\si@unt@third}{%
2886   \ifsi@unt@prepower\else
2887     \expandafter\si@unt@stkpower
2888   \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`.

```

2889 \renewcommand*{\si@tempa}{prefix}%
2890 \expandafter\ifx
2891   \csname si@unt@lastadd\si@unt@checkstack\endcsname\si@tempa
2892 \else
2893   \expandafter\si@unt@spacecheck
2894 \fi
2895 \ifsi@unt@per
2896   \expandafter\si@unt@perseenttrue
2897 \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.

```

2898 \newcommand*{\si@unt@spacecheck}{%
2899   \renewcommand*{\si@tempa}{space}%
2900   \expandafter\ifx
2901     \csname si@unt@lastadd\si@unt@checkstack\endcsname\si@tempa
2902   \else
2903     \expandafter\si@unt@reciptest
2904   \fi}

```

`\si@unt@prefix` #1 : multiple
 #2 : power-ten
 #3 : symbol

Actual output of the prefixes.

```

2905 \newcommand*{\si@unt@prefix}[3]{%
2906   \si@unt@firstorsecond{}{#1}%
2907   \ifsi@prefixsymbolic
2908     \expandafter\si@unt@addprefix
2909   \else
2910     \expandafter\si@unt@countprefix
2911   \fi
2912   {#1}{#2}{#3}}

```

`\si@unt@addprefix` #1 : gobbled
 #2 : gobbled
 #3 : symbol

To add the prefix, a little translation is needed.

```

2913 \newcommand*{\si@unt@addprefix}[3]{%
2914   \si@unt@addtostack{prefix}{#3}}

```

`\si@unt@prefixcnt` A storage area is created.

```

2915 \newcount\si@unt@prefixcnt

```

`\si@unt@countprefix` #1 : multiple
`\si@unt@invprefix` #2 : powers-ten
 #3 : gobbled

To count the prefix numeral, the symbol is thrown away. First, a check is made for binary units.

```

2916 \newcommand*{\si@unt@countprefix}[3]{%
2917   \renewcommand*{\si@tempa}{binary}%
2918   \expandafter\expandafter\expandafter\ifx\expandafter
2919     \csname\expandafter\@gobble\string#1\opt@si\endcsname
2920     \si@tempa
2921   \expandafter\sisetup
2922   \else
2923     \expandafter\@gobble
2924   \fi
2925   {prefixbase=two}%
2926   \si@tempcnta#2\relax
2927   \ifsi@unt@per
2928     \expandafter\si@unt@invprefix
2929   \fi

```

```

2930 \advance\si@unt@prefixcnt\si@tempcnta\relax}
2931 \newcommand*{\si@unt@invprefix}{%
2932 \si@tempcntb\si@tempcnta\relax
2933 \si@tempcnta -\si@tempcntb\relax}

\si@unt@litpower #1 : gobbled
#2 : number
For literal power output, the number can't simply be dumped, so a macro is
needed.
2934 \newcommand*{\si@unt@litpower}[2]{\textsuperscript{#2}}

\ifsi@unt@prepower If a power is seen before a unit, tracking is needed.
2935 \newif\ifsi@unt@prepower

\si@unt@power #1 : power
#2 : number
The handling of powers starts by ensuring that “pre” powers follow \per cleanly.
Then a check is needed for inversion.
2936 \newcommand*{\si@unt@power}[2]{%
2937 \renewcommand*{\si@tempa}{post}%
2938 \expandafter\expandafter\expandafter\ifx\expandafter
2939 \csname\expandafter\@gobble\string#1\opt@si\endcsname
2940 \si@tempa
2941 \expandafter\@gobbletwo
2942 \else
2943 \expandafter\si@unt@firstorsecond
2944 \fi
2945 {}{\power}%
2946 \si@unt@powerdim #2 pt\relax
2947 \ifsi@frac\else
2948 \ifsi@unt@per
2949 \expandafter\expandafter\expandafter\si@unt@invpower
2950 \fi
2951 \fi
2952 \renewcommand*{\si@tempa}{post}%
2953 \si@unt@prepowertrue
2954 \expandafter\expandafter\expandafter\ifx\expandafter
2955 \csname\expandafter\@gobble\string#1\opt@si\endcsname
2956 \si@tempa
2957 \expandafter\si@unt@stackpower
2958 \else
2959 \si@log@debug{Power \strip@pt\si@unt@powerdim\space saved
2960 to be added after\MessageBreak next unit}%
2961 \fi}

\si@unt@powerdim To do sign-inversion on the power, a dimension is used (this allows non-integer
values to be handled).
2962 \newdimen\si@unt@powerdim

\si@unt@stackpower Adding powers to the stack should also clear the power list. If the number is
\si@unt@stkpower already zero, then of course nothing happens.
\si@unt@stkpower2963 \newcommand*{\si@unt@stackpower}{%
2964 \si@unt@prepowerfalse

```

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

```
2965 \si@unt@stkpower
2966 \ifsi@stickyper\else
2967 \si@unt@perfalse
2968 \si@unt@perseenfalse
2969 \fi}
```

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

```
2970 \newcommand*{\si@unt@stkpower}{%
2971 \ifdim\si@unt@powerdim=\m@ne pt\relax
2972 \ifsi@frac\else
2973 \expandafter\expandafter\expandafter\si@unt@stkpw
2974 \fi
2975 \else
2976 \expandafter\si@unt@stkpw
2977 \fi}
```

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

```
2978 \newcommand*{\si@unt@stkpw}{%
2979 \ifdim\si@unt@powerdim=\z@\relax\else
2980 \renewcommand*{\si@tempa}{unit}%
2981 \expandafter\ifx
2982 \csname si@unt@lastadd\si@unt@checkstack\endcsname
2983 \si@tempa
2984 \si@log@debug{Adding power
2985 \strip@pt\si@unt@powerdim\space to output stack}%
2986 \si@unt@addtostack{power}{^{\num[fixdp=false]}{%
2987 \strip@pt\si@unt@powerdim}}}%
2988 \fi
2989 \fi
2990 \si@unt@powerdim\z@\relax}
```

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

```
2991 \newcommand*{\si@unt@invpower}{%
2992 \si@tempdima\si@unt@powerdim\relax
2993 \si@unt@powerdim -\si@tempdima\relax}
```

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

```
2994 \ifdim\si@unt@powerdim=\p@\relax
2995 \si@unt@powerdim\z@\relax
2996 \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@per2997 \newif\ifsi@unt@per
2998 \newif\ifsi@unt@perseen
2999 \si@newrobustcmd*{\si@per}{%
3000 \ifsi@unt@littest\else
3001 \ifsi@unt@litout
3002 \expandafter\expandafter\expandafter /%
3003 \else
```

```

3004     \ifsi@unt@litprefix
3005     \expandafter\expandafter\expandafter\expandafter
3006     \expandafter\expandafter\expandafter /%
3007     \else
3008     \expandafter\expandafter\expandafter\expandafter
3009     \expandafter\expandafter\expandafter\si@unt@per
3010     \fi
3011   \fi
3012 \fi}
3013 \newcommand*{\si@unt@per}{%
3014   \si@unt@firstorsecond{}{\per}%
3015   \ifsi@unt@per
3016     \ifsi@stickyper\else
3017     \expandafter\expandafter\expandafter\si@unt@perfalse
3018     \fi
3019   \else
3020     \expandafter\si@unt@pertrue
3021   \fi}
3022 \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
3023 \newcommand*{\si@unt@reciptest}{%
3024   \ifsi@unt@per
3025     \ifsi@unt@perseen
3026     \expandafter\expandafter\expandafter\si@unt@recip
3027     \fi
3028   \fi}
3029 \newcommand*{\si@unt@recip}{%
3030   \si@unt@powerdim\m@ne pt\relax
3031   \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. First, a track is kept of what has been added at each stage.

```

\si@unt@lastaddb
3032 \newcommand*{\si@unt@lastadda}{}
3033 \newcommand*{\si@unt@lastaddb}{}

```

`\si@unt@addtostack` #1 : type
#2 : token

By indicating the type of data to be added to the stack, problems can be avoided.

```

3034 \newcommand*{\si@unt@addtostack}[2]{%
3035   \renewcommand*{\si@tempa}{#1}%

```

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

```

3036   \expandafter\ifx
3037     \csname si@unt@lastadd\si@unt@checkstack\endcsname\si@tempa
3038     \expandafter\@gobbletwo
3039   \else
3040     \expandafter\si@unt@preplussp
3041   \fi
3042   {#1}{#2}}

```

```
\si@unt@preplussp #1 : type
                  #2 : stack
                  #3 : token
                  #4 : gobbled
```

The space added after a prefix needs to be ignored.

```
3043 \newcommand*{\si@unt@preplussp}[2]{%
3044   \renewcommand*{\si@tempa}{prefix+space}%
3045   \edef\si@tempb{%
3046     \csname si@unt@lastadd\si@unt@checkstack\endcsname+#1}%
3047   \ifx\si@tempa\si@tempb
3048     \expandafter\@gobbletwo
3049   \else
3050     \expandafter\si@unt@stack
3051   \fi
3052   {#1}{#2}}
```

```
\si@unt@stack #1 : type
              #2 : token
```

The macro for actually doing the stacking up.

```
3053 \newcommand*{\si@unt@stack}[2]{%
3054   \expandafter\renewcommand\expandafter*\expandafter{%
3055     \csname si@unt@lastadd\si@unt@checkstack\endcsname}{#1}%
```

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

```
3056   \renewcommand*{\si@tempa}{#1}%
3057   \renewcommand*{\si@tempb}{unit}%
3058   \ifx\si@tempa\si@tempb
3059     \expandafter\si@unt@incnt
3060   \fi
```

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

```
3061   \renewcommand*{\si@tempb}{space}%
3062   \ifx\si@tempa\si@tempb
3063     \expandafter\si@unt@holdspace
3064   \else
3065     \expandafter\si@unt@addstack
3066   \fi
3067   {#2}}
```

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

```
3068 \newcommand*{\si@unt@incnt}{%
3069   \expandafter\advance
3070   \csname si@unt@unitcnt\si@unt@checkstack\endcsname
3071   \@ne\relax}
```

`\si@unt@holdstacka` The stacked material needs somewhere to live.

```
\si@unt@holdstackb3072 \newcommand*{\si@unt@holdstacka}{}
3073 \newcommand*{\si@unt@holdstackb}{}

```

```
\si@unt@holdspace #1 : token
```

`\si@unt@addstack` Depending on the nature of the addition, it is either held or added to the stack. For the “b” space stack, a check is made to ensure that a space cannot be added before the first item.

```

3074 \newcommand*{\si@unt@holdspace}[1]{%
3075 \renewcommand*{\si@tempa}{b}%
3076 \edef\si@tempb{\si@unt@checkstack}%
3077 \ifx\si@tempa\si@tempb
3078 \ifx\@empty\si@unt@stackb\@empty
3079 \else
3080 \expandafter\protected@edef
3081 \csname si@unt@holdstack\si@unt@checkstack\endcsname{#1}%
3082 \fi
3083 \else
3084 \expandafter\protected@edef
3085 \csname si@unt@holdstack\si@unt@checkstack\endcsname{#1}%
3086 \fi}
3087 \newcommand*{\si@unt@addstack}[1]{%
3088 \expandafter\protected@edef
3089 \csname si@unt@stack\si@unt@checkstack\endcsname
3090 {\csname si@unt@stack\si@unt@checkstack\endcsname
3091 \csname si@unt@holdstack\si@unt@checkstack\endcsname#1}%
3092 \expandafter\renewcommand\expandafter*\expandafter{%
3093 \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.

```

3094 \newcommand*{\si@unt@stackout}{%
3095 \si@unt@litouttrue
3096 \ifsi@frac
3097 \expandafter\si@unt@fracout
3098 \else
3099 \expandafter\si@unt@normout
3100 \fi}

```

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

```

3101 \newcommand*{\si@unt@checkstack}{%
3102 \ifsi@frac
3103 \ifsi@unt@per
3104 \expandafter\expandafter\expandafter b%
3105 \else
3106 \expandafter\expandafter\expandafter a%
3107 \fi
3108 \else
3109 \expandafter a%
3110 \fi}

```

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

```

3111 \newcommand*{\si@unt@spaceout}{%
3112 \ensuremath{\si@unt@spstack}}

```

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

```

3113 \newcommand*{\si@unt@prefixout}{%
3114 \ifnum\si@unt@prefixcnt=\z@\relax\else
3115 \ifsi@unt@num
3116 \si@out{\ensuremath{\si@prefixproduct}}}%

```

```

3117 \fi
3118 \si@unt@stackvalsep
3119 \let\si@expbase\si@prefixbase
3120 \num[fixdp=false]{e\the\si@unt@prefixcnt}%
3121 \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.

```

3122 \newcommand*{\si@unt@normout}{%
3123 \si@unt@prefixout
3124 \si@unt@spaceout
3125 \expandafter\si@unt@out\expandafter{\si@unt@stacka}}

```

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

```

3126 \newcommand*{\si@unt@fracout}{%
3127 \si@unt@notambig
3128 \ifx\@empty\si@unt@stacka\@empty
3129 \ifx\@empty\si@unt@stackb\@empty
3130 \ifsi@unt@litout\else
3131 \si@log@err{Empty fractional unit}{The unit
3132 argument\MessageBreak given does not contain any
3133 symbols}%
3134 \fi
3135 \else

```

With an empty numerator, no space is added

```

3136 \ifsi@slash
3137 \si@unt@prefixout
3138 \si@frac{}{\si@unt@stackb}%
3139 \else
3140 \si@unt@prefixout
3141 \si@unt@spaceout
3142 \si@frac{1}{\si@unt@stackb}%
3143 \fi
3144 \fi
3145 \else

```

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

```

3146 \ifx\@empty\si@unt@stackb\@empty
3147 \si@unt@normout
3148 \else
3149 \si@unt@prefixout
3150 \si@unt@spaceout
3151 \si@frac{\si@unt@stacka}{\si@unt@stackb}%
3152 \fi
3153 \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.

`\si@unt@notabg`

```

3154 \newcommand*{\si@unt@notambig}{%
3155 \ifnum\si@unt@unitcntb>\@ne\relax
3156 \ifsi@slash
3157 \ifsi@trapambigfrac
3158 \expandafter\expandafter\expandafter\expandafter

```

```

3159             \expandafter\expandafter\expandafter\si@unt@notabg
3160         \fi
3161     \fi
3162 \fi}
3163 \newcommand*{\si@unt@notabg}{%
3164     \protected@edef\si@unt@stackb{\si@openfrac\si@unt@stackb
3165         \si@closefrac}}

```

\si@unt@out #1 : unit

The final part of the units system is the output routine. This has to cope with units not only as macros but also as direct input (S_Istyle-type input). Non-Latin characters are also handled cleanly. The \scantokens system deals with everything except full stops; these are left out so that a single level system can be used *via* a token register.

```

3166 \begingroup
3167   \catcode'\~=\active
3168   \catcode'\.=\active
3169   \gdef\si@unt@out#1{%
3170       \si@temptoks{#1}%
3171       \si@unt@fullstop
3172       \def.\{\ensuremath{\si@unitsep}}%
3173       \def~{\ensuremath{\si@unitspace}}%
3174       \expandafter\protected@edef\expandafter\si@tempa
3175         \expandafter{\the\si@temptoks}%
3176       \begingroup
3177         \si@unt@nonlatin
3178         \makeatletter
3179         \endlinechar\m@ne
3180         \expandafter\si@out\expandafter{%
3181             \expandafter\scantokens\expandafter{\si@tempa}}%
3182       \endgroup}
3183 \endgroup

```

\si@unt@fullstop \si@unt@stp Two macros modified from kvsetkeys to deal with a single level of active full stops only.

```

3184 \begingroup
3185   \catcode'\.=\active
3186   \catcode'\&=12\relax
3187   \begingroup
3188     \lccode'\.='\.\relax
3189     \lccode'\&='\.\relax
3190   \lowercase{\endgroup}
3191   \gdef\si@unt@fullstop{%
3192       \si@temptoks\expandafter{\expandafter}\expandafter
3193       \si@unt@stp\the\si@temptoks&\@nil}
3194   \gdef\si@unt@stp#1&#2\@nil{%
3195       \edef\si@tempa{\the\si@temptoks}%
3196       \ifx\si@tempa\@empty
3197         \expandafter\@firstoftwo
3198       \else
3199         \expandafter\@secondoftwo
3200       \fi
3201       {\si@temptoks{#1}}

```



```

3202      {\si@temptoks\expandafter{\the\si@temptoks.#1}}%
3203      \si@ifmtarg{#2}
3204      {}
3205      {\si@unt@stp#2\@nil}}
3206 \endgroup

```

`\si@unt@nonlatin` To handle non-Latin symbols in the input, a single macro is provided. If \LaTeX is in use, this can be detected immediately.

```

3207 \newcommand*{\si@unt@nonlatin}{}
3208 \ifdefined\XeTeXrevision
3209   \renewcommand*{\si@unt@nonlatin}{%
3210     \catcode176=\active
3211     \catcode181=\active
3212     \catcode197=\active
3213     \si@unt@sym{176}{\si@sym@degree}%
3214     \si@unt@sym{181}{\si@sym@mu}%
3215     \si@unt@sym{197}{\si@sym@ringA}}%
3216 \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.

```

3217 \AtBeginDocument{
3218   \@ifpackageloaded{inputenc}
3219   {\@for\si@tempa:=latin1,latin5,latin9\do{
3220     \ifx\inputencodingname\si@tempa
3221       \renewcommand*{\si@unt@nonlatin}{%
3222         \catcode176=\active
3223         \catcode181=\active
3224         \catcode197=\active
3225         \si@unt@sym{176}{\si@sym@degree}%
3226         \si@unt@sym{181}{\si@sym@mu}%
3227         \si@unt@sym{197}{\si@sym@ringA}}%
3228       \fi}}
3229   {}}

```

`\si@unt@sym #1 : charcode`

A macro for declaring symbols: a copy of `\DeclareInputMath` from `inputenc`.

```

3230 \newcommand*{\si@unt@sym}[1]{%
3231   \bgroup
3232   \uccode`~#1%
3233   \uppercase{%
3234     \egroup
3235     \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.

```

\meter3236 \newunit{\kilogram}{kg}
\mole3237 \newunit{\metre}{m}
\kelvin3238 \newunit{\meter}{\metre}
\candela3239 \newunit{\mole}{mol}
\second3240 \newunit{\second}{s}
\ampere3241 \newunit{\ampere}{A}
3242 \newunit{\kelvin}{K}

```

```

3243 \newunit{\candela}{cd}

\Square Unlike multiples (which can be skipped if needed), the basic powers are also
\ssquare always defined.
\squared3244 \newpower{\Square}{2}
\cubic3245 \newpower{\ssquare}{2}
\cubed3246 \newpower[post]{\squared}{2}
3247 \newpower{\cubic}{3}
3248 \newpower[post]{\cubed}{3}

\tothe The user macros for arbitrary powers are simple calling macros to a common
\raiseto internal macro.
3249 \newcommand*{\tothe}{\si@tothe{\tothe}}
3250 \newcommand*{\raiseto}{\si@tothe{\raiseto}}

\si@tothe #1 : either \tothe or \raiseto
\tothe@opt@si #2 : number
\raiseto@opt@si A macro for arbitrary powers, which comes after the unit and so needs to be
marked as such. Two fake option-storage macros are created so that everything
works correctly.
3251 \newcommand*{\si@tothe}[2]{%
3252   \ifsi@unt@littest
3253     \expandafter\@gobbletwo
3254   \else
3255     \ifsi@unt@litout
3256       \expandafter\expandafter\expandafter\si@unt@litpower
3257     \else
3258       \expandafter\expandafter\expandafter\si@unt@power
3259     \fi
3260   \fi
3261   {#1}{#2}}
3262 \newcommand*{\tothe@opt@si}[post]{%
3263 \newcommand*{\raiseto@opt@si}{%

```

21.14 Locales

```

\si@loc@load #1 : locale
\si@loc@ssetup When loading a locale, the setup is saved rather than applied. Anything other
than simple settings should be inside \addtolocale, which is already defined.
3264 \newcommand*{\si@loc@load}[1]{%
3265   \let\si@loc@ssetup\sisetup
3266   \renewcommand*{\sisetup}[1]{%
3267     \expandafter\gdef\csname si@loc@#1\endcsname{##1}}%
3268   \si@loadfile{#1}%
3269   \let\sisetup\si@loc@ssetup}

\si@loc@set #1 : locale
Setting the locale transfers the settings to \sisetup, and runs any extra macros.
3270 \newcommand*{\si@loc@set}[1]{%
3271   \ifcsname si@loc@#1\endcsname
3272     \si@log@inf{Setting locale to `#1'}%
3273     \expandafter\expandafter\expandafter\expandafter

```

```

3274 \expandafter\expandafter\expandafter\si@temptoks
3275 \expandafter\expandafter\expandafter\expandafter
3276 \expandafter\expandafter\expandafter{%
3277 \expandafter\csname si@loc@#1\endcsname}%
3278 \expandafter\sisetup\expandafter{\the\si@temptoks}%
3279 \ifcsname si@loc@#1@extra\endcsname
3280 \csname si@loc@#1@extra\endcsname
3281 \fi
3282 \else
3283 \ifcsname si@loc@#1@extra\endcsname
3284 \si@log@inf{Setting locale to '#1'}%
3285 \csname si@loc@#1@extra\endcsname
3286 \else
3287 \si@log@warn{Unknown locale '#1'}%
3288 \fi
3289 \fi}

```

\si@loc@ltol #1 : list of locales and languages

The necessary loading for language modules occurs.

```

3290 \newcommand*{\si@loc@ltol}[1]{%
3291 \def\si@tempa##1:##2\@nil{\si@loc@load{##1}}
3292 \@for\si@tempb:=#1\do{%
3293 \expandafter\si@tempa\si@tempb:\@nil}
3294 \AtBeginDocument{
3295 \ifpackageloaded{babel}
3296 {\def\si@tempa##1:##2:##3\@nil{%
3297 \expandafter\addto\expandafter{%
3298 \csname extras##2\endcsname}%
3299 {\si@loc@set{##1}}}%
3300 \@for\si@tempb:=#1\do{%
3301 \expandafter\si@tempa\si@tempb:\@nil}%
3302 \expandafter\selectlanguage\expandafter{\language}}
3303 {\si@log@warn{babel not loaded \MessageBreak
3304 loctolang option ignored}}}}
3305 \AtBeginDocument{
3306 \ifpackageloaded{babel}
3307 {\renewcommand*{\si@loc@ltol}[1]{%
3308 \def\si@tempa##1:##2\@nil{\si@loc@load{##1}}%
3309 \@for\si@tempb:=#1\do{%
3310 \expandafter\si@tempa\si@tempb:\@nil}%
3311 \def\si@tempa##1:##2:##3\@nil{%
3312 \expandafter\addto\expandafter{%
3313 \csname extras##2\endcsname}%
3314 {\si@loc@set{##1}}}%
3315 \@for\si@tempb:=#1\do{%
3316 \expandafter\si@tempa\si@tempb:\@nil}}}
3317 {\renewcommand*{\si@loc@ltol}[1]{%
3318 \si@log@warn{babel not loaded \MessageBreak
3319 loctolang option ignored}}}}

```

\addtolocale #1 : locale

#2 : commands

Arbitrary macros may need to be added to the locale.

```

3320 \newcommand*{\addtolocale}[2]{%

```

```
3321 \si@addtocname{si@loc@#1@extra}{#2}}
```

21.15 Output routine

```
\si@out #1 : text
```

With all of the setup done, the text can finally be typeset. This is done inside a `\text` block, which takes care of `\ensuremath`, *etc.* First of all, the various catcode settings needed to get maths-in-text mode are made. `\makeatletter` is needed so that `\scantokens` still allows internal macros to work.

```
3322 \begingroup
3323 \catcode`\^=\active
3324 \catcode`\-=\active
3325 \gdef\si@out#1{%
3326   \begingroup
3327     \catcode`\^=\active
3328     \makeatletter
3329     \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`.

```
3330   \ifsi@fam@set\else
3331     \expandafter\si@fam@set
3332     \fi
3333   \text{%
3334     \si@colourcmd{\si@colour}%
3335     \si@fam@italic\si@fam@bold\si@fam@text
```

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

```
3336   \ifsi@textmode
3337     \expandafter\si@out@text
3338   \else
3339     \expandafter\si@out@maths
3340   \fi
3341   {\scantokens{#1}}}%
3342 \endgroup
3343 \check@mathfonts}
```

```
\si@out@text #1 : text
```

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

```
3344 \gdef\si@out@text#1{%
3345   \let^\si@out@sp
3346   \let\textsuperscript\si@out@sp
3347   \catcode`\-=\active\relax
3348   \let-\si@out@minus
3349   #1}
3350 \gdef\si@out@maths#1{%
3351   \let^\sp
3352   \let\textsuperscript\sp
3353   $\si@fam@maths{#1}$}
3354 \endgroup
```

```
\si@out@sp #1 : text
```

`\textsuperscript` gives slightly different alignment of numbers to using `^` in text mode. To avoid this, a slightly different definition is used. Elsewhere

`\si@InputIfFileExists` To allow reading configuration files after the start of the document, a private copy of `\InputIfFileExists` is needed. The main macro here simply checks if the modified version is needed.

```

3378 \let\si@InputIfFileExists\InputIfFileExists
3379 \AtBeginDocument{
3380   \renewcommand*{\si@InputIfFileExists}{%
3381     \ifsi@outerinput
3382       \expandafter\si@outerinput
3383     \else
3384       \expandafter\InputIfFileExists
3385     \fi}}

```

`\si@outerinput` #1 : file
 #2 : true-code
 #3 : false-code
 The outer call to load a file needs to set @ to a letter and turn off printing using `\nullfont`.

```

3386 \newcommand*{\si@outerinput}[3]{%
3387   \makeatletter
3388   \nullfont
3389   \si@outerinputfalse
3390   \IfFileExists{#1}%
3391     {#2\@addtofilelist{#1}\@input \filef@und\normalsize}
3392     {#3}%
3393   \normalsize
3394   \makeatother}

```

`\requiresiconfigs` #1 : config-file
 The configuration files depend on each other.

```

3395 \newcommand*{\requiresiconfigs}[1]{%
3396   \@for\si@tempb:=#1\do{\si@loadfile{\si@tempb}}}

```

`\si@loademfile` #1 : file
 For emulation files, an additional check is made.

```

3397 \newcommand*{\si@loademfile}[1]{%
3398   \@ifpackageloaded{#1}
3399     {\si@log@err{Emulation clash for package '#1'}
3400      {You have asked for emulation of package
3401        '#1'\MessageBreak
3402        (perhaps by giving siunitx a back-compatibility
3403        option)\MessageBreak but the package is already
3404        loaded!}}
3405     {\si@loadfile{#1}}}

```

`\si@emclash` #1 : package
 #2 : package
 A macro for emulation clashes.

```

3406 \newcommand*{\si@emclash}[2]{%
3407   \si@log@err{Emulation clash: '#1' and '#2'}
3408   {You have asked for emulation of package '#1'\MessageBreak
3409     but have already loaded emulation of '#2'}}

```

```
\si@emulating #1 : package
               #2 : version
```

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.

```
3410 \newcommand*\si@emulating}[2]{%
3411   \@namedef{ver@#1.sty}{#2 siunitx emulation of #1}%
3412   \let\si@tempa\si@blockpkgs
3413   \renewcommand*\si@blockpkgs{}%
3414   \@for\si@tempb:=\si@tempa\do{%
3415     \renewcommand*\si@tempa{#1}%
3416     \ifx\si@tempa\si@tempb\else
3417       \lowercase{\edef\si@tempa{#1}}%
3418       \lowercase{\edef\si@tempc{\si@tempb}}%
3419       \ifx\si@tempa\si@tempc
3420         \@namedef{ver@\si@tempc.sty}{#2 siunitx emulation of
3421           #1}%
3422       \else
3423         \si@addtolist{\si@blockpkgs}{\si@tempb}%
3424       \fi
3425     \fi}%
3426   \let\si@tempa\si@checkpkgs
3427   \renewcommand*\si@checkpkgs{}%
3428   \renewcommand*\si@tempb{#1}%
3429   \@for\si@tempc:=\si@tempa\do{%
3430     \ifx\si@tempb\si@tempc\else
3431       \si@addtolist{\si@checkpkgs}{\si@tempc}%
3432     \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.

```
3433 \sisetup{
3434   addsign=none,
3435   allowzeroexp=false,
3436   angformat=unchanged,
3437   astroang=false,%(
3438   closeerr=),%(
3439   closefrac=),
3440   colour=black,
3441   colourall=false,
3442   colourneg=false,
3443   decimalsymbol=fullstop,
3444   detectdisplay=true,
3445   digitsep=thin,
3446   dp=3,
3447   eVcorra=0.3ex,
3448   eVcorrb=0ex,
3449   errspace=none,
3450   fixdp=false,
3451   inlinebold=text,
3452   load=default,
3453   mathsrn=mathrm,
3454   mathssf=mathsf,
```

```

3455 mathstt=mathtt,
3456 mode=maths,
3457 negcolour=red,
3458 noload={},
3459 numaddn={},%(
3460 numcloseerr=),%
3461 numdecimal={.,},
3462 numdigits=0123456789,
3463 numexp=eEdD,
3464 numgobble={},
3465 numopenerr=(,%)
3466 numprod=x,
3467 numsign=+-\pm\mp,
3468 obeybold=false,
3469 obeyitalic=false,
3470 obeymode=false,
3471 openerr=(,%)
3472 openfrac=(,%)
3473 padangle=small,
3474 padnumber=lead,
3475 per=reciprocal,
3476 prefixbase=ten,
3477 prefixproduct=times,
3478 prefixsymbolic=true,
3479 prespace=false,
3480 redefsymbols=true,
3481 repeatunits=true,
3482 retainplus=false,
3483 seperr=false,
3484 sepfour=false,
3485 sign=plus,
3486 slash=slash,
3487 stickyper=false,
3488 strictarc=true,
3489 tabalignexp=true,
3490 tabautofit=false,
3491 tabformat=3.2,
3492 tabnumalign=centredecimal,
3493 tabtextalign=centre,
3494 tabunitalign=left,
3495 textrm=rmfamily,
3496 textsf=sffamily,
3497 texttt=ttfamily,
3498 tightpm=false,
3499 trapambigerr=true,
3500 trapambigfrac=true,
3501 unitsep=thin,
3502 valuesep=thin,
3503 xspace=false}
3504 \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.

```

3505 \ifx\@empty\si@emulate\@empty\else

```



```

3506 \for\si@tempa:=\si@emulate\do{%
3507 \si@loademfile{\si@tempa}}
3508 \fi

```

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

```

3509 \newcommand*{\si@expanddefault}[2]{%
3510 \expandafter\ifx\expandafter\@empty\csname si@#1\endcsname
3511 \@empty
3512 \else
3513 \renewcommand*{\si@tempb}{default}%
3514 \renewcommand*{\si@tempc}{}%
3515 \expandafter\for\expandafter\si@tempa\expandafter
3516 : \expandafter=\csname si@#1\endcsname\do{%
3517 \ifx\si@tempa\si@tempb
3518 \si@addtolist{\si@tempc}{#2}%
3519 \else
3520 \si@addtolist{\si@tempc}{\si@tempa}%
3521 \fi}
3522 \expandafter\edef\csname si@#1\endcsname{\si@tempc}%
3523 \expandafter\si@addtolist\expandafter{%
3524 \csname si@no#1\endcsname}%
3525 {default}%
3526 \renewcommand*{\si@tempc}{}%
3527 \expandafter\for\expandafter\si@tempa\expandafter
3528 : \expandafter=\csname si@#1\endcsname\do{%
3529 \si@switchfalse
3530 \expandafter\for\expandafter\si@tempb\expandafter
3531 : \expandafter=\csname si@no#1\endcsname\do{%
3532 \ifx\si@tempa\si@tempb
3533 \si@switchtrue
3534 \fi
3535 \ifsi@switch\else
3536 \si@addtolist{\si@tempc}{\si@tempa}%
3537 \fi}}
3538 \for\si@tempa:=\si@tempc\do{%
3539 \si@loadfile{\si@tempa}}%
3540 \fi}

```

The configuration and abbreviation files are loaded.

```

3541 \si@expanddefault{load}{prefix,named,addn,prefixed,accepted,%
3542 physical,abbr}

```

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

```

3543 \IfFileExists{siunitx.cfg}
3544 {\si@log@inf{Local configuration file found}%
3545 \InputIfFileExists{siunitx.cfg}{}{}}
3546 {}
3547 \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.

```
\zepto3548 \ProvidesFile{si-prefix.cfg}
\atto3549 [\si@svn@version siunitx:
\fercto3550 SI Multiple prefixes]
\pico3551 \newprefix{\yocto}{-24}{y}
\nano3552 \newprefix{\zepto}{-21}{z}
\micro3553 \newprefix{\atto}{-18}{a}
\Micro3554 \newprefix{\femto}{-15}{f}
\milli3555 \newprefix{\pico}{-12}{p}
\centi3556 \newprefix{\nano}{-9}{n}
\deci
3557 \ifsi@old@OHM
3558 \newprefix{\Micro}{-6}{\si@sym@mu}
3559 \else
3560 \ifsi@gensymb\else
3561 \newprefix{\micro}{-6}{\si@sym@mu}
3562 \fi
3563 \fi
3564 \newprefix{\milli}{-3}{m}
3565 \newprefix{\centi}{-2}{c}
3566 \newprefix{\deci}{-1}{d}
```

`\deca` Now the large ones.

```
\hecto3567 \newprefix{\deca}{1}{da}
\kilo3568 \newprefix{\hecto}{2}{h}
\mega3569 \newprefix{\kilo}{3}{k}
\giga3570 \newprefix{\mega}{6}{M}
\tera3571 \newprefix{\giga}{9}{G}
\peta3572 \newprefix{\tera}{12}{T}
\exa3573 \newprefix{\peta}{15}{P}
\zetta3574 \newprefix{\exa}{18}{E}
\yotta3575 \newprefix{\zetta}{21}{Z}
\yotta3576 \newprefix{\yotta}{24}{Y}
```

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

```
3577 \newprefix{\deka}{1}{da}
```

`\gram` As the base unit of mass is the kilogram, rather than the gram, a bit of extra work
`\kilogram` 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.

```
3578 \newunit{\gram}{g}
3579 \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
`\coulomb` one, but here the UK standard is used.

```
\farad
\Gray
\ggray
\hertz
\henry
\joule
\katal
\lumen
\lux
\newton
```

```

3580 \ProvidesFile{si-named.cfg}
3581 [\si@svn@version siunitx: SI Named units]
3582 \newunit{\becquerel}{Bq}
3583 \newunit{\coulomb}{C}
3584 \newunit{\farad}{F}
3585 \newunit{\Gray}{Gy}
3586 \newunit{\ggray}{Gy}
3587 \newunit{\hertz}{Hz}
3588 \newunit{\henry}{H}
3589 \newunit{\joule}{J}
3590 \newunit{\katal}{kat}
3591 \newunit{\lumen}{lm}
3592 \newunit{\lux}{lx}
3593 \newunit{\newton}{N}

\ohm    Some testing is needed for unitsdef compatibility.
\Ohm3594 \ifsi@old@OHM
\pascal3595 \newunit{\Ohm}{\si@sym@Omega}
\siemens3596 \else
\sievert3597 \ifsi@gensymb\else
\tesla    To be on the safe side, use \provideunit.
\volt3598 \provideunit{\ohm}{\si@sym@Omega}
\watt3599 \fi
\weber3600 \fi
3601 \newunit{\pascal}{Pa}
3602 \newunit{\siemens}{S}
3603 \newunit{\sievert}{Sv}
3604 \newunit{\tesla}{T}
3605 \newunit{\volt}{V}
3606 \newunit{\watt}{W}
3607 \newunit{\weber}{Wb}

\celsius    The degree celsius is a named unit.
\Celsius3608 \ifsi@old@OHM
3609 \newunit{\Celsius}{\si@sym@celsius}
3610 \else
3611 \ifsi@gensymb\else
3612 \newunit{\celsius}{\si@sym@celsius}
3613 \fi
3614 \fi

\radian    The radian and steradian are officially derived units.
\steradian3615 \newunit{\radian}{rad}
3616 \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`.

```

3617 \ProvidesFile{si-prefixed.cfg}
3618 [\si@svn@version siunitx:
3619 SI Prefixed units]
3620 \requiresiconfigs{prefix,named,accepted,physical}

```

```

\picometre    Extra distances.
\nanometre3621 \newunit{\picometre}{\pico\metre}
\micrometre3622 \newunit{\nanometre}{\nano\metre}
\millimetre3623 \newunit{\micrometre}{\micro\metre}
\centimetre3624 \newunit{\millimetre}{\milli\metre}
\decimetre3625 \newunit{\centimetre}{\centi\metre}
\kilometre3626 \newunit{\decimetre}{\deci\metre}
3627 \newunit{\kilometre}{\kilo\metre}

\femtogram    Extra masses.
\picogram3628 \newunit{\femtogram}{\femto\gram}
\nanogram3629 \newunit{\picogram}{\pico\gram}
\microgram3630 \newunit{\nanogram}{\nano\gram}
\milligram3631 \newunit{\microgram}{\micro\gram}
3632 \newunit{\milligram}{\milli\gram}

\femtomole    Now some moles.
\picomole3633 \newunit{\femtomole}{\femto\mole}
\nanomole3634 \newunit{\picomole}{\pico\mole}
\micromole3635 \newunit{\nanomole}{\nano\mole}
\millimole3636 \newunit{\micromole}{\micro\mole}
3637 \newunit{\millimole}{\milli\mole}

\attosecond   Prefixed seconds.
\femtosecond3638 \newunit{\attosecond}{\atto\second}
\picosecond3639 \newunit{\femtosecond}{\femto\second}
\nanosecond3640 \newunit{\picosecond}{\pico\second}
\microsecond3641 \newunit{\nanosecond}{\nano\second}
\millisecond3642 \newunit{\microsecond}{\micro\second}
3643 \newunit{\millisecond}{\milli\second}

\picoampere   The last prefixed base units are amperes.
\nanoampere3644 \newunit{\picoampere}{\pico\ampere}
\microampere3645 \newunit{\nanoampere}{\nano\ampere}
\milliampere3646 \newunit{\microampere}{\micro\ampere}
\kiloampere3647 \newunit{\milliampere}{\milli\ampere}
3648 \newunit{\kiloampere}{\kilo\ampere}

\millivolt    More electricity-related units.
\kilovolt3649 \newunit{\millivolt}{\milli\volt}
\milliwatt3650 \newunit{\kilovolt}{\kilo\volt}
\kilowatt3651 \newunit{\milliwatt}{\milli\watt}
\megawatt3652 \newunit{\kilowatt}{\kilo\watt}
\femtofarad3653 \newunit{\megawatt}{\mega\watt}
\picofarad3654 \newunit{\femtofarad}{\femto\farad}
\nanofarad3655 \newunit{\picofarad}{\pico\farad}
\microfarad3656 \newunit{\nanofarad}{\nano\farad}
\millifarad3657 \newunit{\microfarad}{\micro\farad}
3658 \newunit{\millifarad}{\milli\farad}
\millisiemens3659 \newunit{\millifarad}{\milli\siemens}

\kiloohm      For resistance, checks are needed again for the definition of \ohm.
\megaohm3660 \ifsi@old@OHM
\gigaohm

```

```

3661 \newunit{\kiloohm}{\kilo\Ohm}
3662 \newunit{\megaohm}{\mega\Ohm}
3663 \newunit{\gigaohm}{\giga\Ohm}
3664 \else
3665 \ifsi@gensymb\else
3666 \newunit{\kiloohm}{\kilo\ohm}
3667 \newunit{\megaohm}{\mega\ohm}
3668 \newunit{\gigaohm}{\giga\ohm}
3669 \fi
3670 \fi

```

`\microlitre` Volumes (unlike `unitsdef`, with litre and metre spelled correctly). `\millilitre` and `\microlitre` are defined as they are the two officially-sanctioned prefixes for the litre.

```

\cubiccentimetre3671 \newunit{\microlitre}{\micro\litre}
\centimetrecubed3672 \newunit{\millilitre}{\milli\litre}
\cubicmicrometre3673 \newunit{\cubicmetre}{\metre\cubed}
\cubicmillimetre3674 \newunit{\cubiccentimetre}{\centi\metre\cubed}
\cubicdecimetre3675 \newunit{\centimetrecubed}{\centi\metre\cubed}
3676 \newunit{\cubicmicrometre}{\micro\metre\cubed}
3677 \newunit{\cubicmillimetre}{\milli\metre\cubed}
3678 \newunit{\cubicdecimetre}{\cubic\deci\metre}

```

`\squaremetre` Areas, with metre spelled corrected; `\are` and `\hectare` are in the “temporarily accepted” file.

```

\squarecentimetre3679 \newunit{\squaremetre}{\Square\metre}
\squarekilometre3680 \newunit{\squarecentimetre}{\Square\centi\metre}
3681 \newunit{\centimetresquared}{\centi\metre\squared}
3682 \newunit{\squarekilometre}{\Square\kilo\metre}

```

`\millijoule` Some energy is needed by now! Notice the definition of `\kilowatthour`

```

\kilojoule3683 \newunit{\millijoule}{\milli\joule}
\megajoule3684 \newunit{\kilojoule}{\kilo\joule}
\millielelectronvolt3685 \newunit{\megajoule}{\mega\joule}
\kiloelectronvolt3686 \newunit{\millielelectronvolt}{\milli\electronvolt}
\megaelectronvolt3687 \newunit{\kiloelectronvolt}{\kilo\electronvolt}
\gigaelectronvolt3688 \newunit{\megaelectronvolt}{\mega\electronvolt}
\teraelectronvolt3689 \newunit{\gigaelectronvolt}{\giga\electronvolt}
\kilowatthour3690 \newunit{\teraelectronvolt}{\tera\electronvolt}
3691 \newunit[unitsep=none]{\kilowatthour}{\kilo\watt\hour}

```

`\millihertz` Frequencies.

```

\kilohertz3692 \newunit{\millihertz}{\milli\hertz}
\megahertz3693 \newunit{\kilohertz}{\kilo\hertz}
\gigahertz3694 \newunit{\megahertz}{\mega\hertz}
\terahertz3695 \newunit{\gigahertz}{\giga\hertz}
3696 \newunit{\terahertz}{\tera\hertz}

```

`\millinewton` A few more from various areas.

```

\kilonewton3697 \newunit{\millinewton}{\milli\newton}
\hectopascal3698 \newunit{\kilonewton}{\kilo\newton}
\megabecquerel3699 \newunit{\hectopascal}{\hecto\pascal}
\millisievert3700 \newunit{\megabecquerel}{\mega\becquerel}
3701 \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.
\micA3702 \ProvidesFile{si-abbr.cfg}
\mA3703  [\si@svn@version siunitx: Abbreviated units]
\kA3704 \requiresiconfigs{prefix,named,accepted,physical}
3705 \newunit{\pA}{\pico\ampere}
3706 \newunit{\nA}{\nano\ampere}
3707 \newunit{\micA}{\micro\ampere}
3708 \newunit{\mA}{\milli\ampere}
3709 \newunit{\kA}{\kilo\ampere}

\Hz  Then frequencies.
\mHz3710 \newunit{\Hz}{\hertz}
\kHz3711 \newunit{\mHz}{\milli\hertz}
\MHz3712 \newunit{\kHz}{\kilo\hertz}
\GHz3713 \newunit{\MHz}{\mega\hertz}
\THz3714 \newunit{\GHz}{\giga\hertz}
3715 \newunit{\THz}{\tera\hertz}

\fmol  Amounts of substance.
\pmol3716 \newunit{\fmol}{\femto\mole}
\nmol3717 \newunit{\pmol}{\pico\mole}
\micmol3718 \newunit{\nmol}{\nano\mole}
\mmol3719 \newunit{\micmol}{\micro\mole}
3720 \newunit{\mmol}{\milli\mole}

\kV  Potentials.
\mV3721 \newunit{\kV}{\kilo\volt}
3722 \newunit{\mV}{\milli\volt}

\ml  Volumes and areas
\micl3723 \provideunit{\ml}{\milli\litre}
\cmcl3724 \provideunit{\micl}{\micro\litre}
\dmc3725 \newunit{\cmc}{\centi\metre\cubed}
\cms3726 \newunit{\dmc}{\deci\metre\cubed}
3727 \newunit{\cms}{\centi\metre\squared}

\fg  Masses.
\SIfg3728 \newunit{\kg}{\kilo\gram}
\kg  There is a name clash with babel here in French; hopefully there will not be too
\fg  many complaints.
\pg
3729 \AtBeginDocument{\provideunit{\fg}{\femto\gram}}
\nanog3730 \newunit{\SIfg}{\femto\gram}
\micg3731 \newunit{\pg}{\pico\gram}
\mg3732 \newunit{\nanog}{\nano\gram}
\amu3733 \newunit{\micg}{\micro\gram}
3734 \newunit{\mg}{\milli\gram}
3735 \newunit{\amu}{\atomicmass}

```

`\kJ` Energies.

```
\eV3736 \newunit{\kJ}{\kilo\joule}
\meV3737 \newunit{\eV}{\electronvolt}
\keV3738 \newunit{\meV}{\milli\electronvolt}
\MeV3739 \newunit{\keV}{\kilo\electronvolt}
\GeV3740 \newunit{\MeV}{\mega\electronvolt}
\TeV3741 \newunit{\GeV}{\giga\electronvolt}
\kWh3742 \newunit{\TeV}{\tera\electronvolt}
3743 \newunit[unitsep=none]{\kWh}{\kilo\watt\hour}
```

`\picom` Lengths.

```
\nm3744 \newunit{\picom}{\pico\metre}
\micm3745 \newunit{\nm}{\nano\metre}
\mm3746 \newunit{\micm}{\micro\metre}
\cm3747 \newunit{\mm}{\milli\metre}
\dm3748 \newunit{\cm}{\centi\metre}
\km3749 \newunit{\dm}{\deci\metre}
3750 \newunit{\km}{\kilo\metre}
```

`\Sec` Finally, times.

```
\as3751 \newunit{\Sec}{\second}
\fs3752 \newunit{\as}{\atto\second}
\ps3753 \newunit{\fs}{\femto\second}
```

`\ns` The letter class (and others) define `\ps` for postscripts, so `\provideunit` is best here.

```
\ms3754 \provideunit{\ps}{\pico\second}
3755 \newunit{\ns}{\nano\second}
3756 \newunit{\mics}{\micro\second}
3757 \newunit{\ms}{\milli\second}
```

22.5 Additional (temporary) SI units

`\angstrom` Some units are “temporarily” acceptable for use in the SI system. These are defined here, although some are in very general use.

```
\are3758 \ProvidesFile{si-addn.cfg}
\bar3759 [\si@svn@version siunitx:
\BAR3760 SI Additional units]
\bbar3761 \newunit{\angstrom}{\si@sym@ringA}
\millibar3762 \newunit{\are}{a}
\gal3763 \newunit{\hectare}{\hecto\are}
\curie3764 \newunit{\bar}{b}
\roentgen3765 \newunit{\BAR}{bar}
\rad3766 \newunit{\bbar}{bar}
\rem3767 \newunit{\millibar}{\milli\BAR}
3768 \newunit{\gal}{Gal}
3769 \newunit{\curie}{Ci}
3770 \newunit{\roentgen}{R}
3771 \newunit{\rad}{rad}
3772 \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
\hour3773 \ProvidesFile{si-accepted.cfg}
\Day3774 [\si@svn@version siunitx: SI Accepted units]
\dday3775 \newunit{\minute}{min}
\degree3776 \newunit{\hour}{h}
\Degree3777 \newunit{\Day}{d}
\arcmin3778 \newunit{\dday}{d}
\arcsec3779 \ifsi@old@OHM
3780 \newunit[valuesep=none]{\Degree}{\si@sym@degree}
\litre3781 \else
\liter3782 \ifsi@gensymb\else
\tonne3783 \newunit[valuesep=none]{\degree}{\si@sym@degree}
\neper3784 \fi
\bel3785 \fi
\percent3786 \newunit[valuesep=none]{\arcmin}{\si@sym@minute}
3787 \newunit[valuesep=none]{\arcsec}{\si@sym@second}
3788 \newunit{\litre}{l}
3789 \newunit{\liter}{L}
3790 \newunit{\tonne}{t}
3791 \newunit{\neper}{Np}
3792 \newunit{\bel}{B}
3793 \newunit{\percent}{\%}
```

22.7 Units based on physical measurements

`\si@eVspacea` A few units based on physical measurements exist. For `\eV`, the need for a negative kern does make things a bit complicated.

```
\si@eVspaceb
\electronvolt3794 \ProvidesFile{si-physical.cfg}
\atomicmassunit3795 [\si@svn@version siunitx:
\atomicmass3796 SI Physically-measured units]
3797 \newcommand*{\si@eVspacea}{\text{\kern-\si@eVcorra}}%
3798 \newcommand*{\si@eVspaceb}{\text{\kern-\si@eVcorrb}}%
3799 \newunit{\electronvolt}{e\protect\si@eVspacea V\protect%
3800 \si@eVspaceb}
3801 \newunit{\atomicmass}{u}
3802 \newunit{\atomicmassunit}{u}
```

23 Additional configurations

To provide flexibility for people in specific areas, specialised units can be set up. These are then stored separately to ease use.

23.1 Synthetic chemistry

```
\mmHg Some useful units for synthetic chemists; although \mmHg and \Molar are out-
\molar side of the SI system, they are used a lot. These are set up using \provideunit
\Molar
\torr
\dalton
```


as people may have their own definitions.

```

3803 \ProvidesFile{si-synchem.cfg}
3804 [\si@svn@version siunitx: Units for
3805     synthetic chemists]
3806 \requiresiconfigs{prefix}
3807 \newunit{\mmHg}{mmHg}
3808 \newunit{\molar}{\mole\per\cubic\deci\metre}
3809 \newunit{\Molar}{\textsc{m}}
3810 \newunit{\torr}{Torr}
3811 \newunit{\dalton}{Da}

```

23.2 High-energy physics

Some units inspired by hepunits.

```

3812 \ProvidesFile{si-hep.cfg}
3813 [\si@svn@version siunitx: Units for
3814     high-energy physics]
3815 \requiresiconfigs{prefix,named}

```

`\micron` These specific to high-energy physics, but are not defined elsewhere in siunitx.⁵⁴

```

\mrad3816 \provideunit{\micron}{\micro\metre}
\gauss3817 \newunit{\mrad}{\milli\rad}
3818 \newunit{\gauss}{G}

```

`\clight` The speed of light is used in units for the area, although of course it is not strictly a unit. However, this makes it easier to use in other units.

```

3819 \newunit{\clight}{\ensuremath{\mathnormal{c}}}
3820 \newunit{\eVperc}{\eV\per\clight}

```

`\nanobarn` Various prefixed barns.

```

\picobarn3821 \newunit{\nanobarn}{\nano\barn}
\fb3822 \newunit{\picobarn}{\pico\barn}
\attobarn3823 \newunit{\femtobarn}{\femto\barn}
\zeptobarn3824 \newunit{\attobarn}{\atto\barn}
\yoctobarn3825 \newunit{\zeptobarn}{\zepto\barn}
3826 \newunit{\yoctobarn}{\yocto\barn}

```

`\nb` Abbreviations for the same, but using `\provideunit` to avoid any clashes.

```

\pb3827 \provideunit{\nb}{\nano\barn}
\fb3828 \provideunit{\pb}{\pico\barn}
\ab3829 \provideunit{\fb}{\femto\barn}
\zb3830 \provideunit{\ab}{\atto\barn}
\yb3831 \provideunit{\zb}{\zepto\barn}
3832 \provideunit{\yb}{\yocto\barn}

```

23.3 Astronomy

`\parsec` For astronomy, the `\parsec` unit is needed.

```

\lightyear3833 \ProvidesFile{si-astro.cfg}

```

⁵⁴It is not clear from hepunits, but the assumption is that `\mrad` represents millirad and not milliradian.

```

3834 [\si@svn@version siunitx:
3835   Units for astronomy]
3836 \newunit{\parsec}{pc}
3837 \newunit{\lightyear}{ly}

```

23.4 Binary units

\kibi The binary units, as specified by the IEC and made available by Slunits. First, the
\mebi binary prefixes.

```

\gibi3838 \ProvidesFile{si-binary.cfg}
\tebi3839 [\si@svn@version siunitx: Binary units]
\pebi3840 \newprefix[binary]{\kibi}{10}{Ki}
\exbi3841 \newprefix[binary]{\mebi}{20}{Mi}
\zebi3842 \newprefix[binary]{\gibi}{30}{Gi}
\yobi3843 \newprefix[binary]{\tebi}{40}{Ti}
3844 \newprefix[binary]{\pebi}{50}{Pi}
3845 \newprefix[binary]{\exbi}{60}{Ei}
3846 \newprefix[binary]{\zebi}{70}{Zi}
3847 \newprefix[binary]{\yobi}{80}{Yi}

```

\bit Now the units.

```

\byte3848 \newunit{\bit}{bit}
3849 \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

```

3850 \ProvidesFile{si-UK.cfg}
3851 [\si@svn@version siunitx: UK locale]
3852 \sisetup{
3853   unitsep=thin,
3854   expproduct=times,
3855   valuesep=thin,
3856   decimalsymbol=fullstop,
3857   digitsep=thin,
3858   sepfour=false}

```

24.2 United States

The same as for the UK.

```

3859 \ProvidesFile{si-USA.cfg}
3860 [\si@svn@version siunitx: USA locale]
3861 \sisetup{
3862   unitsep=thin,
3863   expproduct=times,

```

⁵⁵The package author is in the UK.

```

3864 valuesep=thin,
3865 decimalsymbol=fullstop,
3866 digitsep=thin,
3867 sepfour=false}

```

24.3 Germany

Germany, hopefully.

```

3868 \ProvidesFile{si-DE.cfg}
3869 [\si@svn@version siunitx: Germany locale]
3870 \sisetup{
3871   unitsep=cdot,
3872   valuesep=thin,
3873   decimalsymbol=comma,
3874   expproduct=cdot,
3875   digitsep=thin,
3876   sepfour=false}

```

24.4 South Africa

Design decisions taken from the same section of SIstyle.

```

3877 \ProvidesFile{si-ZA.cfg}
3878 [\si@svn@version siunitx:
3879   South Africa locale]
3880 \sisetup{
3881   unitsep=cdot,
3882   valuesep=thin,
3883   expproduct=times,
3884   decimalsymbol=comma,
3885   digitsep=thin,
3886   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.

```

3887 \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.

```

3888 \ProvidesFile{si-units.cfg}
3889 [\si@svn@version siunitx: Emulation of units]
3890 \si@emulating{units}{1998/08/04 v0.9b}
3891 \si@ifloaded{SIunits}
3892 {\si@emclash{units}{SIunits}\endinput}{}

```

```

3893 \si@ifloaded{sistyle}
3894 {\si@emclash{units}{sistyle}\endinput}{}
```

To emulate units, \per must give fractions.

```

3895 \sisetup{per=fraction, fraction=nice, obeybold, inlinebold=maths,
3896   , obeymode}
3897 \ifsi@old@tight
3898   \sisetup{valuesep=thin}
3899 \fi
3900 \ifsi@old@loose
3901   \sisetup{valuesep=space}
3902 \fi
3903 \ifsi@old@ugly
3904   \sisetup{fraction=ugly}
3905 \fi
```

`\unit` [#1]: **number**

#2 : unit

The units version of `\unit` is similar to `\SI`. Here and in `\unitfrac` the `\num` macro is used; thus the number given really has to be a number. However, if users are using `siunitx` rather than `units` they should expect more checking of input. As the `units` package uses the current mode, this has to be detected.

```

3906 \si@newrobustcmd*{\unit}[2][]{%
3907   \ifmmode
3908     \SI{#1}{#2}%
3909   \else
3910     \SI[obeyfamily,obeyitalic]{#1}{#2}%
3911   \fi}
```

`\unitfrac` [#1]: **number**

#2 : numerator

#3 : denominator

`\unitfrac` is a bit more of a hack.

```

3912 \si@newrobustcmd*{\unitfrac}[3][]{%
3913   \begingroup
3914     \si@fam@mode%
3915     \ifmmode\else
3916       \sisetup{obeyfamily,obeyitalic}%
3917     \fi
3918     \si@ifnotmtarg{#1}
3919     {\num{#1}\ensuremath{\si@valuesep}}%
3920     \si@frac{#2}{#3}
3921   \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.

```

3922 \ProvidesFile{si-unitsdef.cfg}
3923 [\si@svn@version siunitx:
3924   Emulation of unitsdef]
3925 \si@emulating{unitsdef}{2005/01/04 v0.2}
```

```

3926 \si@ifloaded{SIunits}
3927   {\si@emclash{unitsdef}{SIunits}\endinput}{}
3928 \si@ifloaded{sistyle}
3929   {\si@emclash{unitsdef}{sistyle}\endinput}{}

```

Emulation of units is needed for unitsdef to work.

```

3930 \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.

```

3931 \RequirePackage{textcomp}
3932 \RequirePackage[T1]{fontenc}

```

The multitude of package options for `unitsdef` need to be handled.

```

3933 \sisetup{mode=text,allowoptarg,prespace}
3934 \ifsi@old@noxspace
3935   \sisetup{xspace=false}
3936 \fi

```

The various options for loading unit abbreviations have to be handled. Here, any request to avoid abbreviations prevents any loading.

```

3937 \ifsi@old@noabbr
3938   \sisetup{noload=abbr}
3939 \fi
3940 \ifsi@old@nofrequncyabbr
3941   \sisetup{noload=abbr}
3942 \fi
3943 \ifsi@old@nomolabbr
3944   \sisetup{noload=abbr}
3945 \fi
3946 \ifsi@old@novoltageabbr
3947   \sisetup{noload=abbr}
3948 \fi
3949 \ifsi@old@novolumeabbr
3950   \sisetup{noload=abbr}
3951 \fi
3952 \ifsi@old@noweightabbr
3953   \sisetup{noload=abbr}
3954 \fi
3955 \ifsi@old@noenergyabbr
3956   \sisetup{noload=abbr}
3957 \fi
3958 \ifsi@old@nolengthabbr
3959   \sisetup{noload=abbr}
3960 \fi
3961 \ifsi@old@notimeabbr
3962   \sisetup{noload=abbr}
3963 \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.

```

3964 \newcommand*{\unitvaluesep}{\,}
3965 \renewcommand*{\si@valuesep}{\text{\unitvaluesep}}
3966 \define@choicekey*+{si}{key}{valuesep}[\si@tempa]

```

```

3967 {space,thin,med,medium,thick,none}
3968 {\renewcommand*\unitvaluesep\@nameuse{si@fix@##1}}%
3969 \si@log@debug{Option valuesep set to ##1}}
3970 {\si@log@debug{Option valuesep set to ##1}}%
3971 \renewcommand*\unitvaluesep{##1}}

```

`\unitsignonly` Some rather straight-forward definitions, with just a bit of fun to get the spacing correct.

```

\ilu
\arc3972 \let\unitsignonly\si
3973 \si@newrobustcmd*{\ilu}[2][]{%
3974 \begingroup
3975 #1\unitvaluesep%
3976 \unit{#2}%
3977 \endgroup}
3978 \let\arc\ang

```

`\unitSIdéf` 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`.

```

3979 \newcommand*{\unitSIdéf}{\upshape}
3980 \newcommand*{\si@unitSIdéf}{\unitSIdéf\selectfont}
3981 \sisetup{textrm=si@unitSIdéf}

```

`\per` Rather awkwardly, `unitsdef` uses `\per` in a different way to `siunitx`.

```

3982 \let\per\relax
3983 \si@newrobustcmd*{\per}[2]{%
3984 \begingroup
3985 \si@xspacefalse
3986 \renewcommand*{\unitvaluesep}{}%
3987 \unitfrac{#1}{#2}%
3988 \endgroup}

```

`\unittimes` Some pretty pretty straight-forward stuff again; notice that the automatic analyser for units has to be turned off for this to work.

```

\unitsep
\unitsuperscript3989 \newcommand*{\unittimes}{\ensuremath{\cdot}}
3990 \newcommand*{\unitsep}{\,,}
3991 \renewcommand*{\si@unt@unithook}{\si@unt@litouttrue}
3992 \sisetup{unitsep=none}
3993 \newcommand*{\unitsuperscript}{\tothe}

```

`\newnosepunit` Simple aliases.

```

\renewnosepunit3994 \newcommand*{\newnosepunit}{\newunit[valuesep=none]}
3995 \newcommand*{\renewnosepunit}{\renewunit[valuesep=none]}

```

`\setTextOmega` Controlling symbols is a simple translation job; as only one setting is used by `siunitx` in text mode, a bit of extra work is needed.

```

\setMathmu3996 \newcommand*{\setTextOmega}[2]{%
\setMathmu3997 \renewcommand*{\si@textOmega}{%
\setTextCelsius3998 \begingroup
\setMathCelsius3999 \edef\si@tempa{\sfdefault}%
\setMathDegree4000 \ifx\f@family\si@tempa
\setTextDegree4001 \expandafter#2%
4002 \else

```

```

4003     \expandafter#1%
4004     \fi
4005   \endgroup}}
4006 \newcommand*{\setMathOmega}[1]{\sisetup{mathsOmega=#1}}
4007 \newcommand*{\setTextmu}[2]{%
4008   \renewcommand*{\si@textmu}{%
4009     \begingroup
4010       \edef\si@tempa{\sfdefault}%
4011       \ifx\f@family\si@tempa
4012         \expandafter#2%
4013       \else
4014         \expandafter#1%
4015       \fi
4016     \endgroup}}
4017 \newcommand*{\setMathmu}[1]{\sisetup{mathsmu=#1}}
4018 \newcommand*{\setTextCelsius}[2]{%
4019   \renewcommand*{\si@textcelsius}{%
4020     \begingroup
4021       \edef\si@tempa{\sfdefault}%
4022       \ifx\f@family\si@tempa
4023         \expandafter#2%
4024       \else
4025         \expandafter#1%
4026       \fi
4027     \endgroup}}
4028 \newcommand*{\setMathCelsius}[1]{\sisetup{mathscelsius=#1}}
4029 \newcommand*{\setMathDegree}[2]{%
4030   \renewcommand*{\si@textdegree}{%
4031     \begingroup%
4032       \edef\si@tempa{\sfdefault}%
4033       \ifx\f@family\si@tempa
4034         \expandafter#2%
4035       \else
4036         \expandafter#1%
4037       \fi
4038     \endgroup}}
4039 \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.

```

4040 \ifsi@old@OHM
4041   \ifsi@old@ohm
4042     \si@log@inf{Both 'ohm' and 'OHM' options given\MessageBreak
4043       Using default behaviour for unitsdef}
4044     \expandafter\expandafter\expandafter\si@old@OHMfalse
4045   \fi
4046 \fi

```

`\liter` Tonne is spelled as “ton” by `unitsdef`, which is wrong in the UK at least (1 ton = 40 cwt = 2240 lb!).

```

\days4047 \ifsi@old@liter
4048   \ifsi@old@LITER
4049     \si@log@inf{Both 'liter' and 'LITER' options
4050       given\MessageBreak Using default behaviour for unitsdef}

```

```

4051 \else
4052 \renewunit{\liter}{l}
4053 \fi
4054 \fi
4055 \newunit{\ton}{t}
4056 \newunit{\days}{d}

```

`\picometer` Extra distances.

```

\nanometer4057 \newunit{\picometer}{\pico\meter}
\micrometer4058 \newunit{\nanometer}{\nano\meter}
\millimeter4059 \newunit{\micrometer}{\micro\meter}
\centimeter4060 \newunit{\millimeter}{\milli\meter}
\decimeter4061 \newunit{\centimeter}{\centi\meter}
\kilometer4062 \newunit{\decimeter}{\deci\meter}
4063 \newunit{\kilometer}{\kilo\meter}

```

`\femtoliter` Volumes with US spellings.

```

\picoliter4064 \newunit{\femtoliter}{\femto\liter}
\nanoliter4065 \newunit{\picoliter}{\pico\liter}
\microliter4066 \newunit{\nanoliter}{\nano\liter}
\milliliter4067 \newunit{\microliter}{\micro\liter}
\centiliter4068 \newunit{\milliliter}{\milli\liter}
\deciliter4069 \newunit{\centiliter}{\centi\liter}
\hectoliter4070 \newunit{\deciliter}{\deci\liter}
4071 \newunit{\hectoliter}{\hecto\liter}
\cubicmeter4072 \newunit{\cubicmeter}{\meter\cubed}
\cubicmicrometer4073 \newunit{\cubicmicrometer}{\micro\meter\cubed}
\cubicmillimeter4074 \newunit{\cubicmillimeter}{\milli\meter\cubed}

```

`\squaremeter` Areas, including the mis-spellings for `\are` and `\hectare`.

```

\squarecentimeter4075 \newunit{\squaremeter}{\Square\meter}
\squarekilometer4076 \newunit{\squarecentimeter}{\Square\centi\meter}
\ar4077 \newunit{\squarekilometer}{\Square\kilo\meter}
\hectar4078 \newunit{\ar}{a}
4079 \newunit{\hectar}{\hecto\ar}

```

`\kv` The code for `unitsdef` has the capitalisation wrong for `\kV` and `\mV`.

```

\mv4080 \ifsi@old@noabbr
4081 \else
4082 \ifsi@old@novoltageabbr\else
4083 \newunit{\kv}{\kilo\volt}
4084 \newunit{\mv}{\milli\volt}
4085 \fi
4086 \fi

```

`\sek` There are some slightly different abbreviations, plus some which are not officially allowed.

```

\fl4087 \ifsi@old@noabbr\else
\pl4088 \ifsi@old@notimeabbr\else
\nl4089 \newunit{\sek}{\second}
\micl4090 \fi
\ml4091 \ifsi@old@noweightabbr\else
\cl4092 \newunit{\fg}{\femto\gram}
\dl
\hl

```



```

4093 \fi
4094 \ifsi@old@novolumeabbr\else
4095 \newunit{\fl}{\femto\liter}
4096 \newunit{\pl}{\pico\liter}
4097 \newunit{\nl}{\nano\liter}
4098 \newunit{\micl}{\micro\liter}
4099 \newunit{\ml}{\milli\liter}
4100 \newunit{\cl}{\centi\liter}
4101 \newunit{\dl}{\deci\liter}
4102 \newunit{\hl}{\hecto\liter}
4103 \fi
4104 \fi

```

`\calory` **unitsdef** spells calorie incorrectly, and it is also not an SI unit.

```

\kilocalory4105 \newunit{\calory}{cal}
4106 \newunit{\kilocalory}{\kilo\calory}

```

`\uBar` **unitsdef** uses `\ubar` for bar.

```

4107 \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.

```

4108 \catcode'\-=11\relax
4109 \ifsi@old@redef-gensymb
4110 \expandafter\si@gensymbtrue
4111 \fi
4112 \catcode'\-=12\relax
4113 \ifsi@gensymb
4114 \RequirePackage{gensymb}
4115 \AtBeginDocument{
4116 \let\gensymbohm\ohm
4117 \let\gensymbcelsius\celsius
4118 \let\gensymbmicro\micro
4119 \let\gensymbdegree\degree
4120 \let\ohm\@undefined
4121 \let\celsius\@undefined
4122 \let\micro\@undefined
4123 \let\degree\@undefined
4124 \ifsi@old@OHM\else
4125 \newunit{\ohm}{\si@sym@Omega}
4126 \newunit{\celsius}{\si@sym@celsius}
4127 \newprefix{\micro}{\si@sym@mu}{-6}
4128 \newunit{\degree}{\si@sym@degree}
4129 \fi}
4130 \fi

```

The configuration files can now be loaded.

```

4131 \requiresiconfigs{prefix,named,addn,accepted}

```

The `noconfig` option could be ignored, but it costs little to let it be used.

```

4132 \ifsi@old@noconfig\else
4133 \InputIfFileExists{unitsdef.cfg}

```

```

4134     {\si@log@inf{unitsdef config file loaded}}
4135     {\si@log@inf{unitsdef config file not found}}
4136 \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.

```

4137 \ProvidesFile{si-sistyle.cfg}
4138 [\si@svn@version siunitx: Emulation of
4139     Sstyle]
4140 \si@emulating{sistyle}{2006/12/20 v2.3}
4141 \sisetup{%
4142     sepfour=true,
4143     obeyfamily,
4144     obeyitalic=true,
4145     numsign=+-,
4146     strictarc=false,
4147     unitsep=cdot}

```

\SIobeyboldtrue Some simple switches, but not using \newif.

```

\SIobeyboldfalse4148 \newcommand*{\SIobeyboldtrue}{\sisetup{obeybold=true}}
4149 \newcommand*{\SIobeyboldfalse}{\sisetup{obeybold=false}}

```

\num To get the correct behaviour for \num, some redefinitions are needed to handle to optional *.

```

\si@sis@numstar4150 \let\num\relax
4151 \si@newrobustcmd*{\num}{%
4152     \@ifstar
4153     {\si@sis@numstar}
4154     {\si@sis@num}}
4155 \newcommand*{\si@sis@num}[2][{}]{%
4156     \begingroup%
4157     \sisetup{#1}%
4158     \expandafter\si@out@num\expandafter{\si@num{#2}}%
4159     \endgroup}
4160 \newcommand*{\si@sis@numstar}[2][{}]{%
4161     \begingroup%
4162     \sisetup{mode=text,obeybold}%
4163     \sisetup{#1}%
4164     \expandafter\si@out@num\expandafter{\si@num{#2}}%
4165     \endgroup}

```

\pnt The \pnt macro is needed as . is active inside \SI. The name is exactly the same as in Sstyle, but the implementation is different. This is not defined by the main package as there are better ways of including numbers in the output than this.

```

4166 \newcommand*{\pnt}{\ensuremath{\si@decimalsymbol}}

```

\SIgroupfourtrue Switches for grouping four characters.

```

\SIgroupfourfalse4167 \newcommand*{\SIgroupfourtrue}{\sisetup{sepfour=true}}
4168 \newcommand*{\SIgroupfourfalse}{\sisetup{sepfour=false}}

```

`\SIunitsep` Whatever is given here is passed through to `\sisetup`.

```
\SIunitspace4169 \newcommand*{\SIunitsep}[1]{\sisetup{valuesep={#1}}}
\SIunitdot4170 \newcommand*{\SIunitspace}[1]{\sisetup{unitspace={#1}}}
4171 \newcommand*{\SIunitdot}[1]{\sisetup{unitsep={#1}}}
```

`\SIdecimalsymbol` The same is true here, with the appropriate translation.

```
\SIthousandsep4172 \newcommand*{\SIdecimalsymbol}[1]{\sisetup{decimalsymbol={#1}}}
\SIproductsign4173 \newcommand*{\SIthousandsep}[1]{\sisetup{digitsep={#1}}}
\SIdecimalsign4174 \newcommand*{\SIproductsign}[1]{\sisetup{expproduct={#1}}}
4175 \newcommand*{\SIdecimalsign}[1]{\sisetup{decimalsymbol={#1}}}
```

`\si@sis@savefont` #1 : setting

#2 : argument

The font definitions need a bit of extra work doing. As both settings here have @ as a letter, all should be fine.

```
4176 \newcommand*{\si@sis@savefont}[2]{%
4177   \@namedef{si@sis@#1}{#2}%
4178   \sisetup{#1=si@sis@#1}}
```

`\SIMathrm` The font control macros have to ensure that a macro name is passed to `\sisetup`.

```
\SIMathsf4179 \newcommand*{\SIMathrm}[1]{\si@sis@savefont{mathrm}{#1}}
\SIMathtt4180 \newcommand*{\SIMathsf}[1]{\si@sis@savefont{mathsf}{#1}}
4181 \newcommand*{\SIMathtt}[1]{\si@sis@savefont{mathtt}{#1}}
```

`\SIdefaultMfam` The same for the default keys.

```
\SIdefaultNfam4182 \newcommand*{\SIdefaultMfam}[1]{\si@sis@savefont{mathrm}{#1}}
\SIdefaultTfam4183 \newcommand*{\SIdefaultNfam}[1]{\si@sis@savefont{mathnumrm}{#1}}
4184 \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.

```
4185 \si@newrobustcmd*{\ensureupmath}[1]{%
4186   \begingroup
4187     \sisetup{mode=maths,obeyitalic=false}%
4188     \si@out{#1}%
4189   \endgroup}
```

`\degC` A few extra symbol names are needed.

```
\degF4190 \newcommand*{\degC}{\si@sym@celsius}
\arcdeg4191 \newcommand*{\arcdeg}{\si@sym@degree}
4192 \newcommand*{\degF}{\si@sym@degree F}
```

`\AddToSIstyle` Finally, the locale control.

```
\SIstyle4193 \newcommand*{\SIstyle}[1]{\sisetup{locale=#1}}
\SIstyleToLang4194 \newcommand*{\SIstyleToLang}[2]{\sisetup{loctolang=#1:#2}}
\si@sis@addtolocale4195 \newcommand*{\AddToSIstyle}{%
4196   \si@switchfalse
4197   \@ifstar
4198     {\si@switchtrue
4199       \si@sis@addtolocale}
4200     {\si@sis@addtolocale}}
4201 \newcommand*{\si@sis@addtolocale}[2]{%
```

```

4202 \ifsi@switch
4203   \expandafter\let\csname si@loc@#1@extra\endcsname\relax
4204 \fi
4205 \addtolocale{#1}{#2}}

```

25.4 Slunits

Slunits emulation starts in much the same way.

```

4206 \ProvidesFile{si-SIunits.cfg}
4207 [\si@svn@version siunitx: Emulation of
4208   SIunits]
4209 \si@emulating{SIunits}{2007/12/02 v1.36}
4210 \sisetup{
4211   unitsep=thick,
4212   valuesep=thick,
4213   prefixproduct=\si@valuesep,
4214   trapambigfrac=false,
4215   stickyper}
4216 \requiresiconfigs{prefix,named,accepted,physical}

```

`\reciprocal` A few very simple translations, using the internal version of `\per` to allow changes of output style.

```

\per4217 \newcommand*{\reciprocal}{\sisetup{per=reciprocal}\si@per}
\usk4218 \let\rp\reciprocal
\power4219 \renewcommand*{\per}{\sisetup{per=slash}\si@per}
\rmsquare4220 \newcommand*{\usk}{}
\rmcubic4221 \newcommand*{\power}[1]{#1\tothe}
\fourth4222 \newcommand*{\rmsquare}{\sisetup{per=reciprocal}\si@per\Square}
\rmfourth4223 \newcommand*{\rmcubic}{\sisetup{per=reciprocal}\si@per\cubic}
4224 \newpower{\fourth}{4}
4225 \newcommand*{\rmfourth}{\sisetup{per=reciprocal}\si@per\fourth}

```

`\rmsquared` Here, some low-level switch changing is needed.

```

\rmcubed4226 \newcommand*{\rmsquared}{%
4227   \sisetup{per=reciprocal}\si@unt@pertrue\si@unt@perseenttrue%
4228   \squared}
4229 \newcommand*{\rmcubed}{%
4230   \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
4231 \newcommand*{\SIsetup}[1]{%
4232   \@for\si@tempa:=#1\do{%
4233     \ifundefined{ifsi@old@#1}
4234       {\si@log@warn{Unknown SIunits option '#1'}}
4235       {\csname si@old@#1true\endcsname}}
4236   \si@siu@setup}
4237 \newcommand*{\si@siu@setup}{%
4238   \ifsi@old@cdot
4239     \sisetup{unitsep=cdot}%
4240   \fi
4241   \ifsi@old@thickspace
4242     \sisetup{unitsep=thick}%

```

```

4243 \fi
4244 \ifsi@old@mediumspace
4245     \sisetup{unitsep=medium}%
4246 \fi
4247 \ifsi@old@thinspace
4248     \sisetup{unitsep=thin}%
4249 \fi
4250 \ifsi@old@thickqspace
4251     \sisetup{valuesep=thick}%
4252 \fi
4253 \ifsi@old@mediumqspace
4254     \sisetup{valuesep=medium}%
4255 \fi
4256 \ifsi@old@thingspace
4257     \sisetup{valuesep=thin}%
4258 \fi}
4259 \si@siu@setup

```

`\square` **Slunits** does slightly different things about the clash with `\square`, and either
`\squareen` **redefines this macro or provides** `\squareen`.

```

4260 \ifsi@old@squaren
4261     \newpower{\squareen}{2}
4262 \fi
4263 \AtBeginDocument{%
4264     \@ifundefined{square}
4265     {\newpower{\square}{2}}
4266     {\ifsi@old@amssymb
4267         \renewpower{\square}{2}
4268         \else
4269         \ifsi@old@squaren\else
4270             \si@log@warn{\string\square\space already
4271                 defined\MessageBreak SIunits mode may cause
4272                 errors}%
4273         \fi
4274     \fi}}

```

`\gray` The potential clash with **PStricks** is also handled differently; here, `\Gray` will
already be defined by the **siunitx** kernel.

```

4275 \AtBeginDocument{
4276     \@ifundefined{gray}
4277     {\newunit{\gray}{Gy}}
4278     {\ifsi@old@pstricks
4279         \renewunit{\gray}{Gy}
4280         \else
4281         \ifsi@old@Gray\else
4282             \si@log@warn{\string\gray\space already
4283                 defined\MessageBreak SIunits mode may cause
4284                 errors}%
4285         \fi
4286     \fi}}

```

`\unit` The `\unit` macro is defined.

```

\unita4287 \ifsi@old@italian
4288     \let\unita\SI

```

```

4289 \else
4290   \let\unit\SI
4291 \fi

```

The miscellaneous options are moped up.

```

4292 \ifsi@old@textstyle
4293   \sisetup{mode=text}
4294 \fi
4295 \ifsi@old@binary
4296   \sisetup{also load= binary}
4297 \fi
4298 \ifsi@old@noams
4299   \AtBeginDocument{%
4300     \renewcommand*{\si@textmu}{\ensuremath\si@mathsmu}}
4301 \fi

```

`\arcminute` The unit macros defined by Slunits that are not defined by siunitx (by default).

```

\arcsecond4302 \newunit[valuesep=none]{\arcminute}{\si@sym@minute}
\rperminute4303 \newunit[valuesep=none]{\arcsecond}{\si@sym@second}
\ton4304 \newunit{\rperminute}{r/min}
\degrecelsius4305 \newunit{\ton}{t}
4306 \newunit{\degrecelsius}{\celsius}

```

`\addunit` This is an alias for `\newunit`.

```

4307 \let\addunit\newunit

```

`\addprefix` A little more work for `\addprefix`.

```

4308 \newcommand*{\addprefix}[2]{\newprefix{#1}{#2}}

```

`\si@siu@newunit` [#1]: power

`\si@siu@power` #2 : numerator

`\si@siu@newunithook` #3 : denominator

To save some code, making new units which need a . . .np variant is handled by a dedicated macro.

```

4309 \newcommand*{\si@siu@newunit}[3][[]]{%

```

A test is needed to sort out `\square`.

```

4310 \renewcommand*{\si@tempa}{#1}%
4311 \renewcommand*{\si@tempb}{square}%
4312 \renewcommand*{\si@siu@power}{}%
4313 \ifx\@empty\si@tempa\@empty\else
4314   \ifx\si@tempa\si@tempb
4315     \renewcommand*{\si@siu@power}{\ssquare}%
4316   \else
4317     \edef\si@siu@power{%
4318       \expandafter\noexpand\csname #1\endcsname}%
4319   \fi
4320 \fi

```

The necessary information is now stored in temporary macros.

```

4321 \edef\si@tempa{%
4322   \expandafter\noexpand\csname #2per#1#3\endcsname}%
4323 \edef\si@tempb{%
4324   \expandafter\noexpand\csname #2\endcsname\noexpand\per

```

```

4325 \expandafter\noexpand\si@siu@power
4326 \expandafter\noexpand\csname #3\endcsname}%
4327 \expandafter\expandafter\expandafter\newunit\expandafter%
4328 \expandafter\expandafter{\expandafter\si@tempa\expandafter}%
4329 \expandafter{\si@tempb}
4330 \edef\si@tempa{%
4331 \expandafter\noexpand\csname #2per#1#3np\endcsname}%
4332 \edef\si@tempb{%
4333 \expandafter\noexpand\csname #2\endcsname\noexpand
4334 \reciprocal\expandafter\noexpand\si@siu@power
4335 \expandafter\noexpand\csname #3\endcsname}%
4336 \expandafter\expandafter\expandafter\newunit\expandafter
4337 \expandafter\expandafter{\expandafter\si@tempa\expandafter}%
4338 \expandafter{\si@tempb}%
4339 \si@siu@newunithook[#1]{#2}{#3}}
4340 \providecommand*\si@siu@newunithook}[3][{}{}

```

The basic units are now defined; these only have a single csname in each of the numerator and denominator.

```

4341 \si@siu@newunit{gray}{second}
4342 \si@siu@newunit[square]{metre}{second}
4343 \si@siu@newunit{joule}{mole}
4344 \si@siu@newunit[cubic]{mole}{metre}
4345 \si@siu@newunit[square]{radian}{second}
4346 \si@siu@newunit{radian}{second}
4347 \si@siu@newunit[cubic]{squaremetre}{metre}
4348 \si@siu@newunit[cubic]{katal}{metre}
4349 \si@siu@newunit{coulomb}{mol}
4350 \si@siu@newunit[square]{ampere}{metre}
4351 \si@siu@newunit[cubic]{kilogram}{metre}
4352 \si@siu@newunit[cubic]{coulomb}{metre}
4353 \si@siu@newunit{volt}{metre}
4354 \si@siu@newunit[square]{coulomb}{squaremetre}
4355 \si@siu@newunit{farad}{metre}
4356 \si@siu@newunit[square]{watt}{metre}
4357 \si@siu@newunit[square]{joule}{metre}
4358 \si@siu@newunit[cubic]{newton}{metre}
4359 \si@siu@newunit{newton}{kilogram}
4360 \si@siu@newunit{joule}{kelvin}
4361 \si@siu@newunit{joule}{kilogram}
4362 \si@siu@newunit{coulomb}{kilogram}
4363 \si@siu@newunit{squaremetre}{second}
4364 \si@siu@newunit[square]{squaremetre}{second}
4365 \si@siu@newunit[square]{candela}{metre}
4366 \si@siu@newunit{ampere}{metre}
4367 \si@siu@newunit{joule}{tesla}
4368 \si@siu@newunit{henry}{metre}
4369 \si@siu@newunit{kilogram}{second}
4370 \si@siu@newunit[square]{kilogram}{metre}
4371 \si@siu@newunit{kilogram}{metre}
4372 \si@siu@newunit[square]{newton}{metre}
4373 \si@siu@newunit{watt}{kilogram}
4374 \si@siu@newunit[cubic]{watt}{metre}
4375 \si@siu@newunit{squaremetre}{kilogram}

```

```

4376 \si@siu@newunit{cubicmetre}{kilogram}
4377 \si@siu@newunit{newton}{metre}
4378 \si@siu@newunit[cubic]{squaremetre}{second}
4379 \si@siu@newunit{metre}{second}
4380 \si@siu@newunit[cubic]{joule}{metre}
4381 \si@siu@newunit{cubicmetre}{second}

```

\si@siu@newunitx For the more complex units, a slightly different approach is used; four arguments are required, and have to cover everything.

```

4382 \newcommand*{\si@siu@newunitx}[4]{%
4383   \expandafter\newunit\expandafter{\csname #1per#2\endcsname}
4384   {#3\per#4}%
4385   \expandafter\newunit\expandafter{\csname #1per#2np\endcsname}
4386   {#3\reciprocal#4}
4387   \si@siu@newunitxhook{#1}{#2}{#3}{#4}}
4388 \providecommand*{\si@siu@newunitxhook}[4]{}

```

The units are defined.

```

4389 \si@siu@newunitx{kilogramsquaremetre}{second}
4390   {\kilogram\squaremetre}{\second}
4391 \si@siu@newunitx{squaremetre}{newtonsecond}{\squaremetre}
4392   {\newton\second}
4393 \si@siu@newunitx{kilogrammetre}{second}{\kilogram\metre}
4394   {\second}
4395 \si@siu@newunitx{kilogram}{squaremetresecond}{\kilogram}
4396   {\squaremetre\second}
4397 \si@siu@newunitx{joule}{molekelvin}{\joule}{\mole\kelvin}
4398 \si@siu@newunitx{kilogram}{kilomole}{\kilogram}{\kilo\mole}
4399 \si@siu@newunitx{kilogrammetre}{squaresecond}{\kilogram\metre}
4400   {\second\squared}
4401 \si@siu@newunitx{watt}{squaremetresteradian}{\watt}
4402   {\squaremetre\steradian}
4403 \si@siu@newunitx{joule}{kilogramkelvin}{\joule}
4404   {\kilogram\kelvin}
4405 \si@siu@newunitx{watt}{metrekkelvin}{\watt}{\metre\kelvin}
4406 \si@siu@newunitx{kilogram}{cubicmetrecoulomb}{\kilogram}
4407   {\cubic\metre\coulomb}
4408 \si@siu@newunitx{kilogram}{secondcubicmetre}{\kilogram}
4409   {\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.

```

4410 \newunit{\si@siu@unity}{1}
4411 \si@siu@newunitx{}{squaremetresecond}{\si@siu@unity}
4412   {\squaremetre\second}

```

A few compound units that are best defined directly.

```

4413 \newunit{\pascalsecond}{\pascal\second}
4414 \newunit{\amperemetresecond}{\ampere\metre\second}
4415 \newunit{\ohmmetre}{\ohm\metre}
4416 \newunit{\newtonmetre}{\newton\metre}
4417 \let\newtonmetrenp\newtonmetre
4418 \newunit{\kilogramsquaremetre}{\kilogram\squaremetre}
4419 \let\kilogramsquaremetrenp\kilogramsquaremetre

```


`\si@siu@newprefix #1 : prefix`

To generate the prefixes correctly, a small saving in repetition.

```

4420 \newcommand*{\si@siu@newprefix}[1]{%
4421   \edef\si@tempa{\expandafter\noexpand\csname #1d\endcsname}%
4422   \edef\si@tempb{\expandafter\noexpand\csname #1\endcsname}%
4423   \expandafter\expandafter\expandafter\newcommand\expandafter
4424     \expandafter\expandafter*\expandafter\expandafter
4425     \expandafter{\expandafter\si@tempa\expandafter}\expandafter
4426     {\expandafter\si@prefixsymbolicfalse\si@tempb}}

```

This is now implemented.

```

4427 \si@siu@newprefix{yocto}
4428 \si@siu@newprefix{zepto}
4429 \si@siu@newprefix{atto}
4430 \si@siu@newprefix{femto}
4431 \si@siu@newprefix{pico}
4432 \si@siu@newprefix{nano}
4433 \si@siu@newprefix{micro}
4434 \si@siu@newprefix{milli}
4435 \si@siu@newprefix{centi}
4436 \si@siu@newprefix{deca}
4437 \si@siu@newprefix{deka}
4438 \si@siu@newprefix{hecto}
4439 \si@siu@newprefix{kilo}
4440 \si@siu@newprefix{mega}
4441 \si@siu@newprefix{giga}
4442 \si@siu@newprefix{tera}
4443 \si@siu@newprefix{peta}
4444 \si@siu@newprefix{exa}
4445 \si@siu@newprefix{zetta}
4446 \si@siu@newprefix{yotta}
4447 \ifsi@old@binary
4448   \si@siu@newprefix{kibi}
4449   \si@siu@newprefix{mebi}
4450   \si@siu@newprefix{gibi}
4451   \si@siu@newprefix{tebi}
4452   \si@siu@newprefix{pebi}
4453   \si@siu@newprefix{exbi}
4454 \fi

```

`\derradian` The derived units may need to be defined.

```

\dersteradian4455 \ifsi@old@derived
  \derhertz4456   \newunit{\derradian}{\metre\reciprocal\metre}
  \dernewton4457   \newunit{\dersteradian}{\squaremetre\rpsquare\metre}
  \derpascal4458   \newunit{\derhertz}{\reciprocal\second}
  \derjoule4459    \newunit{\dernewton}{\metre\kilogram\second\rpsquared}
  \derwatt4460     \newunit{\derpascal}{\newton\rpsquare\metre}
  \dercoulomb4461  \newunit{\derjoule}{\newton\metre}
  \dervolt4462     \newunit{\derwatt}{\joule\reciprocal\second}
  \derfarad4463    \newunit{\dercoulomb}{\ampere\second}
  \derohm4464     \newunit{\dervolt}{\watt\reciprocal\ampere}
  \derohm4465     \newunit{\derfarad}{\coulomb\reciprocal\volt}
  \derohm4466     \newunit{\derohm}{\volt\reciprocal\ampere}

```

```

\dersiemens    In two blocks!
\derweber4467  \newunit{\dersiemens}{\ampere\reciprocal\volt}
\dertesla4468  \newunit{\derweber}
\derhenry4469  {\squaremetre\kilogram\second\rpsquared\reciprocal\ampere}
\dercelsius4470 \newunit{\dertesla}{\weber\rpsquare\metre}
\derlumen4471  \newunit{\derhenry}{\weber\reciprocal\ampere}
\derlux4472    \newunit{\dercelsius}{\kelvin}
\derbecquerel4473 \newunit{\derlumen}{\candela\steradian}
\dergray4474   \newunit{\derlux}{\lumen\rpsquare\metre}
\dersievert4475 \newunit{\derbecquerel}{\derhertz}
\derkatal4476  \newunit{\dergray}{\joule\reciprocal\kilogram}
\derkatal4477  \newunit{\dersievert}{\dergray}
4478           \newunit{\derkatal}{\rp\second\usk\mole}
4479 \fi

\radianbase    Also the "derived-in-base".
\steradianbase4480 \ifsi@old@derivedinbase
\hertzbase4481  \newunit{\radianbase}{\metre\reciprocal\metre}
\newtonbase4482 \newunit{\steradianbase}{\squaremetre\rpsquare\metre}
\pascalbase4483 \newunit{\hertzbase}{\reciprocal\second}
\joulebase4484  \newunit{\newtonbase}{\metre\kilogram\second\rpsquared}
\wattbase4485    \newunit{\pascalbase}{\reciprocal\metre\kilogram\second%
4486              \rpsquared}
\coulombbase4487 \newunit{\joulebase}{\squaremetre\kilogram\second\rpsquared}
\voltbase4488   \newunit{\wattbase}{\squaremetre\kilogram\rpcubic\second}
\faradbase4489  \newunit{\coulombbase}{\ampere\second}
\ohmbase4490    \newunit{\voltbase}
4491              {\squaremetre\kilogram\rpcubic\second\reciprocal\ampere}
4492              \newunit{\faradbase}
4493              {\rpsquare\metre\reciprocal\kilogram\fourth\second\ampere%
4494                \squared}
4495              \newunit{\ohmbase}
4496              {\squaremetre\kilogram\rpcubic\second\rpsquare\ampere}

\siemensbase   Also in two blocks.
\weberbase4497  \newunit{\siemensbase}
\teslabase4498  {\rpsquare\metre\reciprocal\kilogram\cubic\second\ampere%
\henrybase4499  \squared}
\celsiusbase4500 \newunit{\weberbase}
\lumenbase4501  {\squaremetre\kilogram\second\rpsquared\reciprocal\ampere}
\luxbase4502    \newunit{\teslabase}{\kilogram\second\rpsquared\reciprocal%
4503              \ampere}
\becquerelbase4504 \newunit{\henrybase}
\graybase4505   {\squaremetre\kilogram\second\rpsquared\rpsquare\ampere}
\sievertbase4506 \newunit{\celsiusbase}{\kelvin}
\katalbase4507  \newunit{\lumenbase}{\candela\squaremetre\rpsquare\metre}
4508              \newunit{\luxbase}{\candela\squaremetre\rpfourth\metre}
4509              \newunit{\becquerelbase}{\hertzbase}
4510              \newunit{\graybase}{\squaremetre\second\rpsquared}
4511              \newunit{\sievertbase}{\graybase}
4512              \newunit{\katalbase}{\rp\second\mole}
4513 \fi

```

Any configuration file is used if found.

```

4514 \InputIfFileExists{SIunits.cfg}
4515 {\si@log@inf{SIunits config file loaded}}
4516 {\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.

```

4517 \ProvidesFile{si-hepunits.cfg}
4518 [\si@svn@version Emulation of siunitx:
4519     hepunits]
4520 \si@emulating{hepunits}{2007/09/27}
4521 \requiresiconfigs{SIunits,accepted,prefix,hep}

```

`\invbarn` Inverses barn units.

```

\invnanobarn4522 \ifsi@old@noprefixcmds\else
\invpicobarn4523 \newunit{\invbarn}{\per\barn}
\invfemtobarn4524 \newunit{\invnanobarn}{\per\nano\barn}
\invattobarn4525 \newunit{\invpicobarn}{\per\pico\barn}
\invzeptobarn4526 \newunit{\invfemtobarn}{\per\femto\barn}
\invyoctobarn4527 \newunit{\invattobarn}{\per\atto\barn}
4528 \newunit{\invzeptobarn}{\per\zepto\barn}
4529 \newunit{\invyoctobarn}{\per\yocto\barn}

```

`\invnb` Also available abbreviated.

```

\invpb4530 \newunit{\invnb}{\per\nano\barn}
\invfb4531 \newunit{\invpb}{\per\pico\barn}
\invab4532 \newunit{\invfb}{\per\femto\barn}
\invzb4533 \newunit{\invab}{\per\atto\barn}
\invyb4534 \newunit{\invzb}{\per\zepto\barn}
4535 \newunit{\invyb}{\per\yocto\barn}
4536 \fi

```

`\invcmsqpersecond` Luminosity.

```

\invcmsqpersec4537 \newunit{\invcmsqpersecond}{\per\Square\centi\metre\per\second}
\lumiunits4538 \newunit{\invcmsqpersec}{\per\Square\centi\metre\per\second}
4539 \newunit{\lumiunits}{\per\Square\centi\metre\per\second}

```

`\inveV` The inverse of an electron-volt, plus prefixes.

```

\minveV4540 \newunit{\inveV}{\per\electronvolt}
\minveV4541 \newunit{\minveV}{\per\milli\electronvolt}
\kinveV4542 \newunit{\kinveV}{\per\kilo\electronvolt}
\MinveV4543 \newunit{\MinveV}{\per\mega\electronvolt}
\GinveV4544 \newunit{\GinveV}{\per\giga\electronvolt}
\TinveV4545 \newunit{\TinveV}{\per\tera\electronvolt}

```

`\eVoverc` Some combinations of electron-volts and the speed of light. As these are called over, they are set with a slash. The `eVcorrb` values have been set for Computer Modern.

```

4546 \newunit[per=slash,eVcorrb=0.6ex]{\eVoverc}
4547 {\electronvolt\per\cight}
4548 \newunit[per=slash,eVcorrb=0.6ex]{\eVovercsq}
4549 {\electronvolt\per\Square\cight}

```

```

\meVoverc  Prefixed combinations, first of the speed of light.
\keVoverc4550 \newunit[per=slash,eVcorrb=0.6ex]{\meVoverc}
\MeVoverc4551 {\milli\electronvolt\per\clight}
\GeVoverc4552 \newunit[per=slash,eVcorrb=0.6ex]{\keVoverc}
\TeVoverc4553 {\kilo\electronvolt\per\clight}
4554 \newunit[per=slash,eVcorrb=0.6ex]{\MeVoverc}
4555 {\mega\electronvolt\per\clight}
4556 \newunit[per=slash,eVcorrb=0.6ex]{\GeVoverc}
4557 {\giga\electronvolt\per\clight}
4558 \newunit[per=slash,eVcorrb=0.6ex]{\TeVoverc}
4559 {\tera\electronvolt\per\clight}

\meVovercsq Then of the square.
\keVovercsq4560 \newunit[per=slash,eVcorrb=0.6ex]{\meVovercsq}
\MeVovercsq4561 {\milli\electronvolt\per\Square\clight}
\GeVovercsq4562 \newunit[per=slash,eVcorrb=0.6ex]{\keVovercsq}
\TeVovercsq4563 {\kilo\electronvolt\per\Square\clight}
4564 \newunit[per=slash,eVcorrb=0.6ex]{\MeVovercsq}
4565 {\mega\electronvolt\per\Square\clight}
4566 \newunit[per=slash,eVcorrb=0.6ex]{\GeVovercsq}
4567 {\giga\electronvolt\per\Square\clight}
4568 \newunit[per=slash,eVcorrb=0.6ex]{\TeVovercsq}
4569 {\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.

```

4570 \ProvidesFile{si-fancynum.cfg}
4571 [\si@svn@version Emulation of siunitx:
4572   fancynum]
4573 \si@emulating{fancynum}{2000/08/08 0.92}
4574 \sisetup{decimalsymbol=cdot,digitsep=comma}

```

\fnum The \fnum macro is rather restricted, but this is not reproduced. Instead, it is \let as an alias to the \num macro.

```

4575 \let\fnum\num

```

\setfnumdsym The control macros are defined.

```

\setfnumgsym4576 \newcommand*\setfnumdsym[1]{\sisetup{decimalsymbol={#1}}}
\setfnummsym4577 \newcommand*\setfnumgsym[1]{\sisetup{digitsep={#1}}}
4578 \newcommand*\setfnummsym[1]{\sisetup{expproduct={#1}}}

```

The various package options are now processed if necessary.

```

4579 \ifsi@old@english
4580 \sisetup{decimalsymbol=cdot,digitsep=comma}
4581 \fi
4582 \ifsi@old@french
4583 \sisetup{decimalsymbol=comma,digitsep=fullstop}
4584 \fi
4585 \ifsi@old@tight
4586 \sisetup{expproduct=tighttimes}
4587 \fi

```

```

4588 \ifsi@old@loose
4589 \sisetup{expproduct=times}
4590 \fi
4591 \ifsi@old@thinspaces
4592 \sisetup{digitsep=thin}
4593 \fi
4594 \ifsi@old@commas
4595 \sisetup{digitsep=comma}
4596 \fi
4597 \ifsi@old@plain
4598 \sisetup{digitsep=none}
4599 \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.

```

4600 \ProvidesFile{si-fancyunits.cfg}
4601 [\si@svn@version Emulation of siunitx:
4602   fancyunits]
4603 \si@emulating{fancyunits}{2007/02/01 v1.0.1}
4604 \si@ifloaded{SIunits}
4605 {\si@log@err{SIunits emulation loaded\MessageBreak before
4606   fancyunits emulation}{You need to load the fancyunits
4607   emulation\MessageBreak code before that for
4608   SIunits.\MessageBreak Try emulate=fancyunits as the first
4609   option when\MessageBreak loading siunitx}}{}

```

\si@siu@newunithook To create the extra macros provided by fancyunits, the Slunits emulation code is
\si@siu@newunitxhook changed to add the “uf” variants.

```

4610 \newcommand*{\si@siu@newunithook}[3][{}]{%
4611   \edef\si@tempa{%
4612     \expandafter\noexpand\csname #2per#1#3uf\endcsname}%
4613   \renewcommand*{\si@tempb}{stickyper,per=fraction,
4614     fraction=nice}%
4615   \edef\si@tempc{%
4616     \noexpand\sisetup{\si@tempb}%
4617     \expandafter\noexpand\csname #2\endcsname\noexpand\si@per%
4618     \expandafter\noexpand\si@siu@power%
4619     \expandafter\noexpand\csname #3\endcsname}%
4620   \expandafter\expandafter\expandafter\newunit\expandafter
4621     \expandafter\expandafter{\expandafter\si@tempa\expandafter}%
4622     \expandafter{\si@tempc}%
4623   \edef\si@tempa{%
4624     \expandafter\noexpand\csname #2per#1#3Uf\endcsname}%
4625   \renewcommand*{\si@tempb}{stickyper,per=fraction,
4626     fraction=frac}%
4627   \edef\si@tempc{%
4628     \noexpand\sisetup{\si@tempb}%
4629     \noexpand\def\noexpand\si@frc@hook{\noexpand\textstyle}%
4630     \expandafter\noexpand\csname #2\endcsname\noexpand\si@per%
4631     \expandafter\noexpand\si@siu@power%

```

```

4632 \expandafter\noexpand\csname #3\endcsname}%
4633 \expandafter\expandafter\expandafter\newunit\expandafter
4634 \expandafter\expandafter{\expandafter\si@tempa\expandafter}%
4635 \expandafter{\si@tempc}%
4636 \edef\si@tempa{%
4637 \expandafter\noexpand\csname #2per#1#3UF\endcsname}%
4638 \edef\si@tempc{%
4639 \noexpand\sisetup{\si@tempb}%
4640 \noexpand\def\noexpand\si@frc@hook{\noexpand\displaystyle}%
4641 \expandafter\noexpand\csname #2\endcsname\noexpand\si@per%
4642 \expandafter\noexpand\si@siu@power%
4643 \expandafter\noexpand\csname #3\endcsname}%
4644 \expandafter\expandafter\expandafter\newunit\expandafter
4645 \expandafter\expandafter{\expandafter\si@tempa\expandafter}%
4646 \expandafter{\si@tempc}}
4647 \newcommand*{\si@siu@newunitxhook}[4]{%
4648 \expandafter\newunit\expandafter{\csname #1per#2uf\endcsname}
4649 {\sisetup{stickyper,per=fraction,fraction=nice}%
4650 #3\si@per#4}%
4651 \expandafter\newunit\expandafter{\csname #1per#2Uf\endcsname}
4652 {\sisetup{stickyper,per=fraction,fraction=frac}%
4653 \renewcommand*{\si@frc@hook}{\textstyle}%
4654 #3\si@per#4}%
4655 \expandafter\newunit\expandafter{\csname #1per#2UF\endcsname}
4656 {\sisetup{stickyper,per=fraction,fraction=frac}%
4657 \renewcommand*{\si@frc@hook}{\displaystyle}%
4658 #3\si@per#4}}

```

With that done, the emulation modules can be loaded.

```

4659 \requiresiconfigs{SIunits,addn,astro}
4660 \sisetup{obeyall}

```

There is one fancyunits-specific option to handle. The other options all get sent through to the Slunits system.⁵⁶

```

4661 \ifsi@old@spaceqspace
4662 \sisetup{valuesep=space}
4663 \fi

```

`\paminute` fancyunits provides some extra units, plus tonne spelled incorrectly (again).

```

\parsecond4664 \newunit{\paminute}{'}
\AstroE4665 \newunit{\parsecond}{''}
\oersted4666 \newunit{\AstroE}{AE}
\ton4667 \newunit{\oersted}{OE}
4668 \provideunit{\ton}{t}

```

`\decaD` An additional prefix.

```

4669 \let\decaD\decad

```

`\ufrac` The fractional unit macros need to be reproduced.

```

\Ufrac4670 \newcommand*{\ufrac}[2]{%
\UFrac4671 \si[stickyper,per=fraction,fraction=nice]{#1\si@per#2}}

```

⁵⁶As Slunits is rather more widely known than fancyunits, any other options which could be for either are assumed to be for Slunits.

```

4672 \newcommand*{\Ufrac}[2]{%
4673   \ensuremath{\textstyle{%
4674     \si[stickyper,per=fraction,fraction=frac]{#1\si@per#2}}}%
4675 \newcommand*{\UFrac}[2]{%
4676   \ensuremath{\displaystyle{%
4677     \si[stickyper,per=fraction,fraction=frac]{#1\si@per#2}}}%

\pow  A slightly-shortened named \power.
4678 \let\pow\power

\Squaremetre  An alias for \squaremetre.
4679 \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.
4680 \si@siu@newunit{Gray}{second}
4681 \si@siu@newunit[square]{Squaremetre}{metre}
4682 \si@siu@newunitx{Squaremetre}{newtonsecond}{\Square\metre}
4683   {\newton\second}
4684 \si@siu@newunit{Squaremetre}{second}
4685 \si@siu@newunit[square]{Squaremetre}{squaresecond}
4686 \si@siu@newunit{Squaremetre}{kilogram}
4687 \si@siu@newunit[cubic]{Squaremetre}{second}

```

Part V

Notes

26 Change History

1.of	\si@out: Fixed bug with colour and spacing 139	\si@tab@gettok@s: Fixed issues with alignment of s column contents 112
v1.0	General: First official release 1	v1.ob
v1.0a	\si@fix@space: Fixed problem with space option 74	\si@num@rndpost: Corrected bug in rounding code 100
	\si@loc@ltol: babel support fixed for default document language 138	v1.oc
	\si@tab@end@S: Fixed mixed number/text column alignment 113	General: Fixed excess loading of maths fonts 71
		v1.od
		\si@unt@out: Sanitise one level of unit input only 135
		v1.oh
		General: Simplified warnings for tabformat 66

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	
\%	3793
\&	3186, 3189
\@@input	3391
\@addtofilelist	3391
\@filef@und	3391
\^	22, 32, 1232, 3323, 3327
\`	20, 30
\~	23, 33, 1231, 3167, 3232
A	
\ab	20 , <u>3827</u>
\addprefix	<u>4308</u>
addsign (option)	25
\addtolocale	36 , <u>3320</u> , 4205
\AddToSistyle	<u>4193</u>
\addunit	<u>4307</u>
allowoptarg (option)	30
allowzeroexp (option)	25
alsoload (option)	32
\ampere	14 , <u>3236</u> , 3644–3648, 3705–3709, 4414, 4463, 4464, 4466, 4467, 4469, 4471, 4489, 4491, 4493, 4496, 4498, 4501, 4503, 4505
\amu	16 , <u>3728</u>
\ang	10 , <u>2009</u> , 2033, 2034, 2038, 3978
angformat (option)	26
anglesep (option)	23
\angstrom	16 , <u>3758</u>
\ar	<u>4075</u>
\arc	<u>3972</u>
\arcdeg	<u>4190</u>
\arcmin	15 , <u>3773</u>
\arcminute	<u>4302</u>
\arcsec	15 , <u>3773</u>
\arcsecond	<u>4302</u>
\are	16 , <u>3758</u>
\as	16 , <u>3751</u>
astroang (option)	27
\AstroE	<u>4664</u>

<code>\atomicmass</code>	15 , 16 , 3735 , 3794	<code>\coulombbase</code>	4480
<code>\atomicmassunit</code>	15 , 3794	<code>\cubed</code>	12 , 3244 , 3673 – 3677 , 3725 , 3726 , 4072 – 4074 , 4230
<code>\atto</code>	14 , 3548 , 3638 , 3752 , 3824 , 3830 , 4527 , 4533	<code>\cubic</code>	12 , 3244 , 3678 , 3808 , 4223 , 4407 , 4498
<code>\attobarn</code>	20 , 3821	<code>\cubiccentimetre</code>	17 , 3671
<code>\attosecond</code>	16 , 3638	<code>\cubicdecimetre</code>	17 , 3671
B			
<code>\BAR</code>	16 , 3758	<code>\cubicmeter</code>	4064
<code>\barn</code> ..	16 , 3758 , 3821 – 3832 , 4523 – 4535	<code>\cubicmetre</code>	3671 , 4409
<code>\bbar</code>	16 , 3758	<code>\cubicmicrometer</code>	4064
<code>\becquerel</code>	15 , 3580 , 3700	<code>\cubicmicrometre</code>	3671
<code>\becquerelbase</code>	4497	<code>\cubicmillimeter</code>	4064
<code>\bel</code>	15 , 3773	<code>\cubicmillimetre</code>	3671
<code>\bit</code>	19 , 3848	<code>\curie</code>	16 , 3758
<code>\byte</code>	19 , 3848	D	
C			
<code>\calory</code>	4105	<code>\dalton</code>	19 , 3803
<code>\candela</code> ...	14 , 3236 , 4473 , 4507 , 4508	<code>\Day</code>	15 , 3773
<code>\Celsius</code>	3608	<code>\days</code>	4047
<code>\celsius</code> 15 , 3608 , 4117 , 4121 , 4126 , 4306		<code>\dday</code>	15 , 3773
<code>\celsiusbase</code>	4497	<code>debug (option)</code>	32
<code>\centi</code> ..	14 , 3548 , 3625 , 3674 , 3675 , 3680 , 3681 , 3725 , 3727 , 3748 , 4061 , 4069 , 4076 , 4100 , 4537 – 4539	<code>\deca</code>	14 , 3567
<code>\centiliter</code>	4064	<code>\decaD</code>	4669
<code>\centimeter</code>	4057	<code>\decad</code>	4669
<code>\centimetre</code>	16 , 3621	<code>\deci</code>	14 , 3548 , 3626 , 3678 , 3726 , 3749 , 3808 , 4062 , 4070 , 4101
<code>\centimetrecubed</code>	17 , 3671	<code>\deciliter</code>	4064
<code>\centimetresquared</code>	17 , 3679	<code>decimalsymbol (option)</code>	24
<code>\cl</code>	4087	<code>\decimeter</code>	4057
<code>\clight</code>	20 , 3819 , 4547 , 4549 , 4551 , 4553 , 4555 , 4557 , 4559 , 4561 , 4563 , 4565 , 4567 , 4569	<code>\decimetre</code>	16 , 3621
<code>closeerr (option)</code>	24	<code>\degC</code>	4190
<code>closefrac (option)</code>	29	<code>\degF</code>	4190
<code>\cm</code>	16 , 3744	<code>\Degree</code>	3773
<code>\cmc</code>	17 , 3723	<code>\degree</code>	15 , 3773 , 4119 , 4123 , 4128
<code>\cms</code>	17 , 3723	<code>\degreecelsius</code>	4302
<code>color (option)</code>	31	<code>\deka</code>	3577
<code>colorall (option)</code>	31	<code>\derbecquerel</code>	4467
<code>colorneg (option)</code>	31	<code>\dercelsius</code>	4467
<code>colorunit (option)</code>	31	<code>\dercoulomb</code>	4455
<code>colorvalue (option)</code>	31	<code>\derfarad</code>	4455
<code>colour (option)</code>	31	<code>\dergray</code>	4467
<code>colourall (option)</code>	31	<code>\derhenry</code>	4467
<code>colourneg (option)</code>	31	<code>\derhertz</code>	4455 , 4475
<code>colourunits (option)</code>	31	<code>\derjoule</code>	4455
<code>colourvalues (option)</code>	31	<code>\derkatal</code>	4467
<code>\coulomb</code>	15 , 3580 , 4407 , 4465	<code>\derlumen</code>	4467
		<code>\derlux</code>	4467
		<code>\dernewton</code>	4455
		<code>\derohm</code>	4455

<code>\derpascal</code>	4455	<code>\femtosecond</code>	16 , 3638
<code>\derradian</code>	4455	<code>\fg</code>	16 , 2627 , 3728 , 4087
<code>\dersiemens</code>	4467	<code>fixdp (option)</code>	26
<code>\dersievert</code>	4467	<code>\fl</code>	4087
<code>\dersteradian</code>	4455	<code>\fmol</code>	17 , 3716
<code>\dertesla</code>	4467	<code>\fnum</code>	4575
<code>\dervolt</code>	4455	<code>\fourth</code>	4217 , 4493
<code>\derwatt</code>	4455	<code>fraction (option)</code>	29
<code>\derweber</code>	4467	<code>\fs</code>	16 , 3751
<code>detectdisplay (option)</code>	23		
<code>digitsep (option)</code>	23 , 24		
<code>\dimexpr</code>	2102 , 2110 , 2131 , 2133		
<code>\dl</code>	4087		
<code>\dm</code>	16 , 3744		
<code>\dmc</code>	17 , 3723		
<code>\document</code>	1026		
<code>dp (option)</code>	26		
	E		G
<code>\electronvolt</code>		<code>\gal</code>	16 , 3758
..... 15 , 17 , 3686 – 3690 , 3737 –		<code>\gauss</code>	20 , 3816
3742 , 3794 , 4540 – 4545 , 4547 ,		<code>\gensymbcelsius</code>	4108
4549 , 4551 , 4553 , 4555 , 4557 ,		<code>\gensymbdegree</code>	4108
4559 , 4561 , 4563 , 4565 , 4567 , 4569		<code>\gensymbmicro</code>	4108
<code>emulate (option)</code>	32	<code>\gensymbohm</code>	4108
<code>\ensureupmath</code>	4185	<code>\GeV</code>	17 , 3736
<code>errspace (option)</code>	23	<code>\GeVoverc</code>	4550
<code>\eV</code>	17 , 3736 , 3820	<code>\GeVovercsq</code>	4560
<code>eVcorra (option)</code>	31	<code>\ggray</code>	15 , 3580
<code>eVcorrb (option)</code>	31	<code>\GHz</code>	17 , 3710
<code>\everyeof</code>	1233	<code>\gibi</code>	20 , 3838
<code>\eVoverc</code>	4546	<code>\giga</code> ... 14 , 3567 , 3663 , 3668 , 3689 ,	
<code>\eVovercsq</code>	4546	3695 , 3714 , 3741 , 4544 , 4557 , 4567	
<code>\eVperc</code>	20 , 3819	<code>\gigaelectronvolt</code>	17 , 3683
<code>\exa</code>	14 , 3567	<code>\gigahertz</code>	17 , 3692
<code>\exbi</code>	20 , 3838	<code>\gigaohm</code>	18 , 3660
<code>expbase (option)</code>	25	<code>\GinveV</code>	4540
<code>expproduct (option)</code>	25	<code>\gram</code>	3578 , 3628 – 3632 , 3728 – 3734 , 4092
		<code>\Gray</code>	15 , 3580
		<code>\gray</code>	4275
		<code>\graybase</code>	4497
	F		H
<code>\farad</code>	15 , 3580 , 3654 – 3658	<code>\hectar</code>	4075
<code>\faradbase</code>	4480	<code>\hectare</code>	16 , 3758
<code>\fb</code>	20 , 3827	<code>\hecto</code>	14 ,
<code>\femto</code> 14 , 3548 , 3628 , 3633 , 3639 , 3654 ,		3567 , 3699 , 3763 , 4071 , 4079 , 4102	
3716 , 3729 , 3730 , 3753 , 3823 ,		<code>\hectoliter</code>	4064
3829 , 4064 , 4092 , 4095 , 4526 , 4532		<code>\hectopascal</code>	18 , 3697
<code>\femtobarn</code>	20 , 3821	<code>\henry</code>	15 , 3580
<code>\femtofarad</code>	18 , 3649	<code>\henrybase</code>	4497
<code>\femtogram</code>	16 , 3628	<code>\hertz</code> 15 , 17 , 3580 , 3692 – 3696 , 3710 – 3715	
<code>\femtoliter</code>	4064	<code>\hertzbase</code>	4480 , 4509
<code>\femtomole</code>	17 , 3633	<code>\hl</code>	4087
		<code>\hour</code>	15 , 3691 , 3743 , 3773
		<code>\Hz</code>	17 , 3710

I

<code>\ifdefined</code>	1027, 1034, 3208	<code>\ifsi@old@LITER</code>	837, 4048
<code>\ifsi@addunitpower</code>	672, 2653	<code>\ifsi@old@liter</code>	837, 4047
<code>\ifsi@allowoptarg</code>	649, 2760	<code>\ifsi@old@loose</code> .	829, 871, 3900, 4588
<code>\ifsi@allowzeroexp</code>	383, 1324	<code>\ifsi@old@mediumqspace</code> .	851, 4253
<code>\ifsi@ang@fixdp</code>	2018, 2059	<code>\ifsi@old@mediumspace</code> ..	851, 4244
<code>\ifsi@ang@padlarge</code>	470, 2185	<code>\ifsi@old@nice</code>	829
<code>\ifsi@ang@padsmall</code>	470, 2239	<code>\ifsi@old@noabbr</code> 841, 3937, 4080, 4087	
<code>\ifsi@ang@sign</code>	1669, 2183	<code>\ifsi@old@noamperageabbr</code> ...	841
<code>\ifsi@ang@toarc</code>	499, 2047	<code>\ifsi@old@noams</code>	863, 4298
<code>\ifsi@ang@todec</code>	499, 2054	<code>\ifsi@old@noconfig</code>	837, 4132
<code>\ifsi@astroang</code>	521, 2195	<code>\ifsi@old@noenergyabbr</code> .	841, 3955
<code>\ifsi@colourneg</code>	765, 1312	<code>\ifsi@old@nofrequncyabbr</code> 841, 3940	
<code>\ifsi@colourunits</code>	726, 1105	<code>\ifsi@old@nolengthabbr</code> .	841, 3958
<code>\ifsi@colourvalues</code> .	726, 1094, 2459	<code>\ifsi@old@nomolabbr</code>	841, 3943
<code>\ifsi@debug</code>	121, 146	<code>\ifsi@old@noprefixcmds</code> .	868, 4522
<code>\ifsi@detectdisplay</code> 337, 1126, 1144		<code>\ifsi@old@notimeabbr</code> 841, 3961, 4088	
<code>\ifsi@fam@set</code>	1069, 1085, 3330	<code>\ifsi@old@novoltageabbr</code>	
<code>\ifsi@fixdp</code>	632, 1693, 2018 841, 3946, 4082	
<code>\ifsi@frac</code> .	650, 2947, 2972, 3096, 3102	<code>\ifsi@old@novolumeabbr</code>	
<code>\ifsi@gensymb</code>	833, 841, 3949, 4094	
3560, 3597, 3611, 3665, 3782, 4113		<code>\ifsi@old@noweightabbr</code>	
<code>\ifsi@inlinebtext</code>	329, 1049 841, 3952, 4091	
<code>\ifsi@logmin</code>	121, 126, 134, 140	<code>\ifsi@old@noxspace</code>	837, 3934
<code>\ifsi@lognone</code> .	121, 125, 133, 139, 145	<code>\ifsi@old@OHM</code>	833, 3557,
<code>\ifsi@num@ambigerr</code>	1349, 1727	3594, 3608, 3660, 3779, 4040, 4124	
<code>\ifsi@num@delplus</code> ...	565, 573, 1533	<code>\ifsi@old@ohm</code>	833, 4041
<code>\ifsi@num@erropen</code> .	1278, 1376, 1409	<code>\ifsi@old@plain</code>	873, 4597
<code>\ifsi@num@intab</code> 1220, 1338, 1379, 1403		<code>\ifsi@old@pstricks</code>	858, 4278
<code>\ifsi@num@padlead</code> ..	388, 1580, 1817	<code>\ifsi@old@redef</code>	4109
<code>\ifsi@num@padtrail</code>	388, 1585	<code>\ifsi@old@redef-gensymb</code>	833
<code>\ifsi@num@signexp</code>	425, 1553	<code>\ifsi@old@spaceqspace</code> ..	876, 4661
<code>\ifsi@num@signmant</code>	425, 1547	<code>\ifsi@old@squaren</code> ..	858, 4260, 4269
<code>\ifsi@num@textmode</code>	297, 1073	<code>\ifsi@old@textstyle</code>	863, 4292
<code>\ifsi@obeybold</code>	328, 1140	<code>\ifsi@old@thickqspace</code> ..	851, 4250
<code>\ifsi@obeyfamily</code>	327, 1118	<code>\ifsi@old@thickspace</code> ...	851, 4241
<code>\ifsi@obeyitalic</code>	336, 1158	<code>\ifsi@old@thinqspace</code> ...	851, 4256
<code>\ifsi@obeymode</code>	296, 1058	<code>\ifsi@old@thinspace</code>	851, 4247
<code>\ifsi@old@amssymb</code>	858, 4266	<code>\ifsi@old@thinspace</code> ...	873, 4591
<code>\ifsi@old@binary</code> ...	863, 4295, 4447	<code>\ifsi@old@tight</code> .	829, 871, 3897, 4585
<code>\ifsi@old@cdot</code>	851, 4238	<code>\ifsi@old@ugly</code>	829, 3903
<code>\ifsi@old@commas</code>	873, 4594	<code>\ifsi@out@num</code>	1072, 1087, 3357
<code>\ifsi@old@derived</code>	863, 4455	<code>\ifsi@outerinput</code>	3377, 3381
<code>\ifsi@old@derivedinbase</code> 863, 4480		<code>\ifsi@prefixsymbolic</code> ...	387, 2907
<code>\ifsi@old@english</code>	869, 4579	<code>\ifsi@prespace</code>	643
<code>\ifsi@old@french</code>	869, 4582	<code>\ifsi@redefsymbols</code>	816, 915
<code>\ifsi@old@Gray</code>	858, 4281	<code>\ifsi@repeatunits</code>	
<code>\ifsi@old@italian</code>	858, 4287 672, 1360, 2005, 2641	
		<code>\ifsi@retainplus</code>	379, 1525
		<code>\ifsi@seperr</code>	375, 1348, 1743

L	
<code>\language</code>	3302
<code>\lccode</code>	3188, 3189
<code>\lightyear</code>	21, 3833
<code>\liter</code>	14, 15, 3773, 4047, 4064–4071, 4095–4102
<code>\litre</code>	14, 15, 3671, 3672, 3723, 3724, 3773
<code>\llap</code>	2410
<code>load (option)</code>	32
<code>locale (option)</code>	31
<code>loctolang (option)</code>	31
<code>log (option)</code>	32
<code>\lumen</code>	15, 3580, 4474
<code>\lumenbase</code>	4497
<code>\lumiunits</code>	4537
<code>\lux</code>	15, 3580
<code>\luxbase</code>	4497
M	
<code>\mA</code>	17, 3702
<code>\makeatother</code>	3394
<code>mathrm (option)</code>	23
<code>mathscelsius (option)</code>	30
<code>mathsdegree (option)</code>	30
<code>mathsf (option)</code>	23
<code>mathsminute (option)</code>	30
<code>mathsmu (option)</code>	30
<code>mathsOmega (option)</code>	30
<code>mathsringA (option)</code>	30
<code>mathsrm (option)</code>	23
<code>mathssecond (option)</code>	30
<code>mathssf (option)</code>	23
<code>mathstt (option)</code>	23
<code>mathtt (option)</code>	23
<code>\mebi</code>	20, 3838
<code>\mega</code>	14, 3567, 3653, 3662, 3667, 3685, 3688, 3694, 3700, 3713, 3740, 4543, 4555, 4565
<code>\megabecquerel</code>	18, 3697
<code>\megaelectronvolt</code>	17, 3683
<code>\megahertz</code>	17, 3692
<code>\megajoule</code>	3683
<code>\megaohm</code>	18, 3660
<code>\megawatt</code>	18, 3649
<code>\meter</code>	13, 14, 3236, 4057–4063, 4072–4077
<code>\metre</code>	13, 14, 3236, 3621–3627, 3673–3682, 3725–3727, 3744–3750, 3808, 3816, 4393, 4399, 4405, 4407, 4414–4416, 4456, 4457, 4459–4461, 4470, 4474, 4481, 4482, 4484, 4485, 4493, 4498, 4507, 4508, 4537–4539, 4682
<code>\MeV</code>	17, 3736
<code>\meV</code>	17, 3736
<code>\MeVoverc</code>	4550
<code>\meVoverc</code>	4550
<code>\MeVovercsq</code>	4560
<code>\meVovercsq</code>	4560
<code>\mg</code>	16, 3728
<code>\MHz</code>	17, 3710
<code>\mHz</code>	17, 3710
<code>\micA</code>	17, 3702
<code>\micg</code>	16, 3728
<code>\micl</code>	17, 3723, 4087
<code>\micm</code>	16, 3744
<code>\micmol</code>	17, 3716
<code>\Micro</code>	3548
<code>\micro</code>	14, 3548, 3623, 3631, 3636, 3642, 3646, 3657, 3671, 3676, 3707, 3719, 3724, 3733, 3746, 3756, 3816, 4059, 4067, 4073, 4098, 4118, 4122, 4127
<code>\microampere</code>	17, 3644
<code>\microfarad</code>	18, 3649
<code>\microgram</code>	16, 3628
<code>\microliter</code>	4064
<code>\microlitre</code>	17, 3671
<code>\micrometer</code>	4057
<code>\micrometre</code>	16, 3621
<code>\micromole</code>	17, 3633
<code>\micron</code>	20, 3816
<code>\microsecond</code>	16, 3638
<code>\mics</code>	16, 3751
<code>\milli</code>	14, 3548, 3624, 3632, 3637, 3643, 3647, 3649, 3651, 3658, 3659, 3672, 3677, 3683, 3686, 3692, 3697, 3701, 3708, 3711, 3720, 3722, 3723, 3734, 3738, 3747, 3757, 3767, 3817, 4060, 4068, 4074, 4084, 4099, 4541, 4551, 4561
<code>\milliampere</code>	17, 3644
<code>\millibar</code>	16, 3758
<code>\millielectronvolt</code>	17, 3683
<code>\millifarad</code>	18, 3649
<code>\milligram</code>	16, 3628
<code>\millihertz</code>	17, 3692
<code>\millijoule</code>	3683
<code>\milliliter</code>	4064
<code>\millilitre</code>	17, 3671
<code>\millimeter</code>	4057
<code>\millimetre</code>	16, 3621

<code>\millimole</code>	17 , 3633	<code>\newpower</code>	19 , 2608 , 3244 – 3248 , 4224 , 4261 , 4265
<code>\millinewton</code>	18 , 3697	<code>\newprefix</code>	19 , 2594 , 3551 – 3556 , 3558 , 3561 , 3564 – 3577 , 3840 – 3847 , 4127 , 4308
<code>\millisecond</code>	16 , 3638	<code>\newton</code>	15 , 3580 , 3697 , 3698 , 4392 , 4416 , 4460 , 4461 , 4683
<code>\millisiemens</code>	18 , 3649	<code>\newtonbase</code>	4480
<code>\millisievert</code>	18 , 3697	<code>\newunit</code>	18 , 2580 , 3236 – 3243 , 3578 , 3582 – 3593 , 3595 , 3601 – 3607 , 3609 , 3612 , 3615 , 3616 , 3621 – 3659 , 3661 – 3663 , 3666 – 3668 , 3671 – 3701 , 3705 – 3722 , 3725 – 3728 , 3730 – 3753 , 3755 – 3757 , 3761 – 3772 , 3775 – 3778 , 3780 , 3783 , 3786 – 3793 , 3799 , 3801 , 3802 , 3807 – 3811 , 3817 – 3826 , 3836 , 3837 , 3848 , 3849 , 3994 , 4055 – 4079 , 4083 , 4084 , 4089 , 4092 , 4095 – 4102 , 4105 – 4107 , 4125 , 4126 , 4128 , 4277 , 4302 – 4307 , 4327 , 4336 , 4383 , 4385 , 4410 , 4413 – 4416 , 4418 , 4456 – 4468 , 4470 – 4478 , 4481 – 4485 , 4487 – 4490 , 4492 , 4495 , 4497 , 4500 , 4502 , 4504 , 4506 – 4512 , 4523 – 4535 , 4537 – 4546 , 4548 , 4550 , 4552 , 4554 , 4556 , 4558 , 4560 , 4562 , 4564 , 4566 , 4568 , 4620 , 4633 , 4644 , 4648 , 4651 , 4655 , 4664 – 4667
<code>\minute</code>	15 , 3773	<code>\nl</code>	4087
<code>\MinveV</code>	4540	<code>\nm</code>	16 , 3744
<code>\minveV</code>	4540	<code>\nmol</code>	17 , 3716
<code>\ml</code>	17 , 3723 , 4087	<code>noload (option)</code>	32
<code>\mm</code>	16 , 3744	<code>\ns</code>	16 , 3751
<code>\mmHg</code>	19 , 3803	<code>\num</code>	5 , 1212 , 1243 , 2647 , 2851 , 2986 , 3120 , 3919 , 4150 , 4575
<code>\mmol</code>	17 , 3716	<code>numaddn (option)</code>	24
<code>mode (option)</code>	22	<code>numcloseerr (option)</code>	24
<code>\Molar</code>	19 , 3803	<code>numdecimal (option)</code>	24
<code>\molar</code>	19 , 3803	<code>numdigits (option)</code>	24
<code>\mole</code> ...	14 , 3236 , 3633 – 3637 , 3716 – 3720 , 3808 , 4397 , 4398 , 4478 , 4512	<code>numexp (option)</code>	24
<code>\mp</code>	1246	<code>numgobble (option)</code>	24
<code>\mrad</code>	20 , 3816	<code>numopenerr (option)</code>	24
<code>\ms</code>	16 , 3751	<code>numprod (option)</code>	25
<code>\mV</code>	17 , 3721	<code>numsign (option)</code>	24
<code>\mv</code>	4080		
N			
<code>\nA</code>	17 , 3702	O	
<code>\nano</code>	14 , 3548 , 3622 , 3630 , 3635 , 3641 , 3645 , 3656 , 3706 , 3718 , 3732 , 3745 , 3755 , 3821 , 3827 , 4058 , 4066 , 4097 , 4524 , 4530		
<code>\nanoampere</code>	17 , 3644	<code>obeyall (option)</code>	22
<code>\nanobarn</code>	20 , 3821	<code>obeybold (option)</code>	22
<code>\nanofarad</code>	18 , 3649	<code>obeyfamily (option)</code>	22
<code>\nanog</code>	16 , 3728	<code>obeyitalic (option)</code>	22
<code>\nanogram</code>	16 , 3628		
<code>\nanoliter</code>	4064		
<code>\nanometer</code>	4057		
<code>\nanometre</code>	16 , 3621		
<code>\nanomole</code>	17 , 3633		
<code>\nanosecond</code>	16 , 3638		
<code>\nb</code>	20 , 3827		
<code>\NC@list</code>	2283		
<code>\NC@rewrite@S</code>	2288		
<code>\NC@rewrite@s</code>	2288		
<code>negcolor (option)</code>	31		
<code>negcolour (option)</code>	31		
<code>\neper</code>	15 , 3773		
<code>\newnosepunit</code>	3994		

obeymode (option)	22	mathsrm	23
\oersted	4664	mathssecond	30
\Ohm	3594, 3661–3663	mathssf	23
\ohm	15, 3594, 3666–3668, 4116, 4120, 4125, 4415	mathstt	23
\ohmbase	4480	mathtt	23
\Omega	777, 778	mode	22
openerr (option)	24	negcolor	31
openfrac (option)	29	negcolour	31
options:		noload	32
addsign	25	numaddn	24
allowoptarg	30	numcloseerr	24
allowzeroexp	25	numdecimal	24
alsoload	32	numdigits	24
angformat	26	numexp	24
anglesep	23	numgobble	24
astroang	27	numopenerr	24
closeerr	24	numprod	25
closefrac	29	numsign	24
color	31	obeyall	22
colorall	31	obeybold	22
colorneg	31	obeyfamily	22
colorunit	31	obeyitalic	22
colorvalue	31	obeymode	22
colour	31	openerr	24
colourall	31	openfrac	29
colourneg	31	padangle	26
colourunits	31	padnumber	25
colourvalues	31	per	29
debug	32	prefixbase	29
decimalsymbol	24	prefixproduct	29
detectdisplay	23	prefixsymbolic	29
digitsep	23, 24	prespace	30
dp	26	redefsymbols	31
emulate	32	repeatunits	25
errspace	23	retainplus	25
eVcorra	31	seperr	24
eVcorrb	31	sepfour	24
expbase	25	sign	25
expproduct	25	slash	29
fixdp	26	stickyper	29
fraction	29	strict	32
load	32	strictarc	26
locale	31	tabalign	28
loctolang	31	tabalignexp	27
log	32	tabautofit	28
mathrm	23	tabformat	27
mathscelsius	30	tabnumalign	27
mathsdegree	30	tabtextalign	28
mathsf	23	tabunitalign	28
mathsminute	30	textcelsius	30
mathsmu	30	textdegree	30
mathsOmega	30	textminute	30
mathsringA	30	textmode	22
		textmu	30

textOmega	30	\peta	14, 3567
textringA	30	\pg	16, 3728
textrm	23	\pico	14, 3548, 3621, 3629, 3634, 3640, 3644, 3655, 3705, 3717, 3731, 3744, 3754, 3822, 3828, 4057, 4065, 4096, 4525, 4531
textsecond	30	\picoampere	17, 3644
textsf	23	\picobarn	20, 3821
texttt	23	\picofarad	18, 3649
tightpm	24	\picogram	16, 3628
trapambigerr	24	\picoliter	4064
trapambigfrac	29	\picom	16, 3744
unitcolor	31	\picometer	4057
unitcolour	31	\picometre	16, 3621
unitmathsrn	23	\picomole	17, 3633
unitmathssf	23	\picosecond	16, 3638
unitmathstt	23	\pl	4087
unitmode	22	\pm	592, 892, 893, 1246, 3467
unitsep	23	\pmol	17, 3716
unitspace	23	\pnt	4166
unittextrm	23	\pow	4678
unittextsf	23	\power	2945, 4217, 4678
unittexttt	23	prefixbase (option)	29
valuecolor	31	prefixproduct (option)	29
valuecolour	31	prefixsymbolic (option)	29
valuemathrm	23	prespace (option)	30
valuemathssf	23	\prime	802–805
valuemathsrn	23	\protected@xdef	1235
valuemathssf	23	\providecommand	4340, 4388
valuemathstt	23	\providepower	19, 2608
valuemathtt	23	\provideprefix	19, 2594
valuemode	22	\provideunit 18, 2580, 3598, 3723, 3724, 3729, 3754, 3816, 3827–3832, 4668
valuesep	23	\ps	16, 3751
valuetextrm	23		
valuetextsf	23		
valuetexttt	23		
xspace	30		

P			
\pA	17, 3702		
padangle (option)	26		
padnumber (option)	25		
\pamminute	4664		
\parsec	21, 3833		
\parsecond	4664		
\pascal	15, 3594, 3699, 4413		
\pascalbase	4480		
\pb	20, 3827		
\pebi	20, 3838		
\per	12, 2997, 3808, 3820, 3982, 4217, 4324, 4384, 4523–4535, 4537–4545, 4547, 4549, 4551, 4553, 4555, 4557, 4559, 4561, 4563, 4565, 4567, 4569		
per (option)	29		
\percent	15, 3773		

R			
\rad	16, 3758, 3817		
\radian	15, 3615		
\radianbase	4480		
\raiseto	12, 3249		
\raiseto@opt@si	3251		
\reciprocal	4217, 4334, 4386, 4456, 4458, 4462, 4464–4467, 4469, 4471, 4476, 4481, 4483, 4485, 4491, 4493, 4498, 4501, 4502		
redefsymbols (option)	31		
\rem	16, 3758		
\renewnosepunit	3994		
\renewpower	19, 2608, 4267		
\renewprefix	19, 2594		

<code>\renewunit</code>	<code>\si@addunitpowertrue</code>
... 18 , 2580 , 3579, 3995, 4052, 4279	683
<code>\repeat</code>	<code>\si@ang@arc</code>
1854	2044, 2045
<code>repeatunits (option)</code>	<code>\si@ang@arcdeg</code>
25	2119
<code>\requiresiconfigs</code>	<code>\si@ang@arcfix</code>
.... 36 , 3395 , 3620, 3704, 3806,	2053, 2059 , 2120
3815, 3930, 4131, 4216, 4521, 4659	<code>\si@ang@arcmin</code>
<code>retainplus (option)</code>	2119
25	<code>\si@ang@arcsec</code>
<code>\roentgen</code>	2119
16 , 3758	<code>\si@ang@arctodec</code>
<code>\rp</code>	2055, 2084
4217 , 4478, 4512	<code>\si@ang@astrosign</code>
<code>\rpcubed</code>	2196, 2256
4226	<code>\si@ang@dec</code>
<code>\rpcubic</code>	2023, 2045
4217 , 4488, 4491, 4496	<code>\si@ang@decimalsymbol</code> .
<code>\rperminute</code>	2194 , 2259
4302	<code>\si@ang@dectoarc</code>
<code>\rpfourth</code>	2048, 2119
4217 , 4508	<code>\si@ang@deg</code>
<code>\rpsquare</code>	2185
.. 4217 , 4457, 4460, 4470, 4474,	<code>\si@ang@fix</code>
4482, 4493, 4496, 4498, 4505, 4507	2046,
<code>\rpsquared</code> .	2053, 2059 , 2085, 2120, 2242, 2248
4226, 4459, 4469, 4484,	<code>\si@ang@fixdptrue</code>
4486, 4487, 4501, 4502, 4505, 4510	2121
	<code>\si@ang@ifnum</code>
	. 2073 , 2090–2092, 2125, 2204, 2216
	<code>\si@ang@killdegree</code>
	2253
	<code>\si@ang@killminute</code>
	2253
	<code>\si@ang@killsecond</code>
	2253
	<code>\si@ang@minnum</code>
	2200
	<code>\si@ang@mins</code>
	2185
	<code>\si@ang@movesign</code> ..
	2195, 2243, 2249
	<code>\si@ang@notnum</code> ..
	2116–2118, 2164, 2179
	<code>\si@ang@num</code> ...
	2200, 2201, 2233, 2240
	<code>\si@ang@pad</code> ...
	2211, 2213, 2225, 2239
	<code>\si@ang@padlargefalse</code>
	475
	<code>\si@ang@padlargetrue</code>
	482, 487, 492, 497
	<code>\si@ang@padsmallfalse</code>
	474
	<code>\si@ang@padsmalltrue</code>
	478, 486, 491, 496
	<code>\si@ang@parse</code>
	2016, 2017
	<code>\si@ang@secnum</code>
	2200
	<code>\si@ang@secs</code>
	2185
	<code>\si@ang@sepint</code>
	2130, 2132, 2165
	<code>\si@ang@signlessnum</code>
	2221, 2229, 2231, 2240
	<code>\si@ang@signtrue</code>
	2206, 2218
	<code>\si@ang@sint</code>
	2165
	<code>\si@ang@strippt</code> ...
	2136, 2141, 2165
	<code>\si@ang@toarcfalse</code>
	503
	<code>\si@ang@toarctrue</code>
	515, 519
	<code>\si@ang@todectrue</code>
	504
	<code>\si@ang@todectrue</code>
	507, 511
	<code>\si@ang@typeset</code>
	. 2050, 2057, 2114, 2163, 2182, 2184
	<code>\si@anglesep</code>
	293 , 2236
	<code>\si@blockpkgs</code> ...
	51 , 3412, 3413, 3423
	<code>\si@catcodes</code>
	19 , 3547

S

<code>\Sec</code>	16 , 3751
<code>\second</code>	14 ,
16 , 3236 , 3638–3643, 3751–3757,	
4089, 4390, 4392, 4394, 4396,	
4400, 4409, 4412–4414, 4458,	
4459, 4462, 4463, 4469, 4478,	
4483–4485, 4487–4489, 4491,	
4493, 4496, 4498, 4501, 4502,	
4505, 4510, 4512, 4537–4539, 4683	
<code>\sek</code>	4087
<code>\selectlanguage</code>	3302
<code>seperr (option)</code>	24
<code>sepfour (option)</code>	24
<code>\setfnumdsym</code>	4576
<code>\setfnumgsym</code>	4576
<code>\setfnummsym</code>	4576
<code>\setMathCelsius</code>	3996
<code>\setMathDegree</code>	3996
<code>\setMathmu</code>	3996
<code>\setMathOmega</code>	3996
<code>\setTextCelsius</code>	3996
<code>\setTextDegree</code>	3996
<code>\setTextmu</code>	3996
<code>\setTextOmega</code>	3996
<code>\SI</code> 11 , 2575 , 2639, 3908, 3910, 4288, 4290	
<code>\si</code> 13 , 2421, 2579, 3972, 4671, 4674, 4677	
<code>\si@addtocname</code>	86 , 3321
<code>\si@addtolist</code>	80 , 273, 715,
3423, 3431, 3518, 3520, 3523, 3536	
<code>\si@addunitpowerfalse</code>	676

<code>\si@checkpkgs</code> ... 51 , 3426 , 3427 , 3431	<code>\si@fam@tt</code> 1026 , 1172
<code>\si@closeerr</code> 375 , 1371 , 1411	<code>\si@fileprefix</code> 3363 , 3368 , 3371 , 3374
<code>\si@closefrac</code> 685 , 3165	<code>\si@fix@cdot</code> 882
<code>\si@colour</code> 1085 , 3334	<code>\si@fix@comma</code> 882
<code>\si@colourcmd</code> 1085 , 3334	<code>\si@fix@fullstop</code> 882
<code>\si@colournegfalse</code> 768	<code>\si@fix@med</code> 877
<code>\si@colournegtrue</code> 771	<code>\si@fix@medium</code> 877
<code>\si@colourunitsfalse</code> . 730 , 745 , 754	<code>\si@fix@minus</code> 890
<code>\si@colourunitstrue</code> .. 733 , 748 , 757	<code>\si@fix@mp</code> 890
<code>\si@colourvaluesfalse</code> 737 , 744 , 753	<code>\si@fix@none</code> 898
<code>\si@colourvaluestrue</code> . 740 , 749 , 758	<code>\si@fix@period</code> 882
<code>\si@debugfalse</code> 229	<code>\si@fix@plus</code> 890
<code>\si@debugtrue</code> 242 , 246	<code>\si@fix@pm</code> 468 , 890
<code>\si@decimalsymbol</code>	<code>\si@fix@slash</code> 897
... 291 , 1703 , 1830 , 2194 , 2257 ,	<code>\si@fix@space</code> 877
2471 , 2474 , 2502 , 2515 , 2547 , 4166	<code>\si@fix@stop</code> 882
<code>\si@digitsep</code> ... 288 , 1972 , 1989 , 2572	<code>\si@fix@ten</code> 895
<code>\si@emclash</code> 3406 , 3892 , 3894 , 3927 , 3929	<code>\si@fix@thick</code> 877
<code>\si@emulate</code> 272 , 3505 , 3506	<code>\si@fix@thin</code> 877
<code>\si@emulating</code> 3410 , 3890 ,	<code>\si@fix@tightcdot</code> 882
3925 , 4140 , 4209 , 4520 , 4573 , 4603	<code>\si@fix@tightpm</code> 466 , 890
<code>\si@errspace</code> 278 , 1370	<code>\si@fix@tighttimes</code> 882
<code>\si@eVcorra</code> 821 , 3797	<code>\si@fix@times</code> 882
<code>\si@eVcorrb</code> 821 , 3798	<code>\si@fix@two</code> 895
<code>\si@eVspacea</code> 3794	<code>\si@fixdpfalse</code> 2019 , 2068 , 2071
<code>\si@eVspaceb</code> 3794	<code>\si@fixdptrue</code> .. 640 , 2066 , 2089 , 2444
<code>\si@expanddefault</code> 3509 , 3541	<code>\si@frac</code> 661 , 689 , 692 , 696 , 700 , 704 ,
<code>\si@expbase</code> 380 , 1399 , 2531 , 3119	708 , 712 , 933 , 3138 , 3142 , 3151 , 3920
<code>\si@expproduct</code> 380 , 1395 , 2531	<code>\si@fracfalse</code> 656
<code>\si@extension</code> . 3363 , 3368 , 3371 , 3374	<code>\si@fracttrue</code> 659 , 665 , 669
<code>\si@fam@bold</code> 1139 , 1210 , 3335	<code>\si@frc@displen</code> .. 959 , 971 , 979 , 983
<code>\si@fam@boldify</code> 1208	<code>\si@frc@frac</code> 689 , 933 , 1015
<code>\si@fam@colourcmd</code>	<code>\si@frc@hook</code> 933 , 4629 , 4640 , 4653 , 4657
..... 1065 , 1095 , 1106 , 1313 , 2460	<code>\si@frc@mathsnf</code> 965 , 969
<code>\si@fam@detmaths</code> 1127 , 1164	<code>\si@frc@nice</code> 692 , 700 , 933
<code>\si@fam@detttext</code> 1123 , 1129 , 1134 , 1164	<code>\si@frc@nicefrac</code> 944 , 959
<code>\si@fam@ifbinline</code> 1048 , 1151	<code>\si@frc@sfrac</code> 704 , 708 , 933
<code>\si@fam@ifbmaths</code> .. 1041 , 1052 , 1145	<code>\si@frc@slash</code> 661 , 933 , 1023
<code>\si@fam@ifbtext</code> 1041 , 1050 , 1147 , 1154	<code>\si@frc@ssuplen</code>
<code>\si@fam@ifitext</code> 1054 , 1160	. 959 , 977 , 979 – 981 , 989 , 1004 , 1006
<code>\si@fam@italic</code> 1157 , 3335	<code>\si@frc@suplen</code> ... 959 , 975 , 981 , 987
<code>\si@fam@maths</code> 1114 , 1167 ,	<code>\si@frc@textlen</code>
1174 , 1180 , 1189 , 1196 , 1202 , 3353	959 , 973 , 980 , 985 , 1003 , 1006 , 1007
<code>\si@fam@mode</code> .. 1057 , 1215 , 2012 , 3914	<code>\si@frc@textnf</code> 967 , 1001
<code>\si@fam@set</code> 1071 , 3331	<code>\si@frc@ugly</code> 696 , 712 , 1012
<code>\si@fam@setbold</code> 1149 , 1151 , 1154 , 1208	<code>\si@gensymbtrue</code> 4110
<code>\si@fam@settrue</code> 1111	<code>\si@gobblethree</code> ... 2773 , 2782 , 2792
<code>\si@fam@sfs</code> 1026 , 1165	<code>\si@ifdefinable</code>
<code>\si@fam@text</code> 1114 , 1169 , 76 , 2581 , 2585 , 2591 ,
1176 , 1182 , 1191 , 1198 , 1204 , 3335	2595 , 2599 , 2605 , 2609 , 2613 , 2619

<code>\si@ifl@aded</code>	<u>3365</u>	<code>\si@mathsminute</code>	<u>795</u>
<code>\si@ifloaded</code>	<u>3365</u> , 3370,	<code>\si@mathsmu</code>	<u>779</u> , 4300
3887, 3891, 3893, 3926, 3928, 4604		<code>\si@mathsOmega</code>	<u>775</u> , 931
<code>\si@ifmtarg</code>	<u>92</u> , 2020, 2650, 3203	<code>\si@mathsringA</code>	<u>812</u> , 925
<code>\si@ifnotmtarg</code> <u>92</u> , 598, 1237, 2202,		<code>\si@mathsrm</code> ...	<u>1085</u> , 1115, 1181, 1203
2214, 2226, 2628, 2633, 2636,		<code>\si@mathssecond</code>	<u>795</u>
2638, 2649, 2756, 2850, 2880, 3918		<code>\si@mathssf</code>	<u>1085</u> , 1168, 1190
<code>\si@inlinebtextfalse</code>	331	<code>\si@mathstt</code>	<u>1085</u> , 1175, 1197
<code>\si@inlinebtexttrue</code>	334	<code>\si@negcolour</code>	<u>765</u> , 1320
<code>\si@InputIfFileExists</code> .	3371, <u>3378</u>	<code>\si@newcmd</code>	<u>100</u>
<code>\si@load</code>	<u>714</u>	<code>\si@newcommand</code>	<u>100</u>
<code>\si@loademfile</code>	<u>3397</u> , 3507	<code>\si@newrobustcmd</code>	
<code>\si@loadfile</code>		<u>100</u> , 1212, 2009, 2575, 2579, 2999,	
.. 725, 3268, <u>3369</u> , 3396, 3405, 3539		3906, 3912, 3973, 3983, 4151, 4185	
<code>\si@loc@load</code> ...	826, <u>3264</u> , 3291, 3308	<code>\si@noload</code>	<u>714</u>
<code>\si@loc@ltol</code>	828, <u>3290</u>	<code>\si@num</code>	1218, <u>1221</u> , 1995,
<code>\si@loc@set</code>	827, <u>3270</u> , 3299, 3314	1999, 2244, 2251, 2447, 4158, 4164	
<code>\si@loc@sisetup</code>	<u>3264</u>	<code>\si@num@addexp</code>	1449, <u>1460</u>
<code>\si@locale</code>	<u>825</u>	<code>\si@num@addmnt</code>	1451, <u>1460</u>
<code>\si@loctolang</code>	<u>825</u>	<code>\si@num@addmntexp</code> .	1461, 1463, <u>1464</u>
<code>\si@log@debug</code>		<code>\si@num@addpostzero</code> ...	1589, <u>1642</u>
144, 192, 199, 202, 206, 209, 217,		<code>\si@num@addprezero</code>	1581, <u>1642</u>
219, 226, 917, 930, 1119, 1122,		<code>\si@num@addpzero</code> ..	1643, 1645, <u>1646</u>
1125, 1133, 1137, 1141, 1159,		<code>\si@num@addsign</code>	1483, <u>1537</u>
1162, 1166, 1173, 1179, 1188,		<code>\si@num@addtmp</code>	1798, 1799, <u>1800</u>
1195, 1201, 1209, 1217, 1282,		<code>\si@num@addtmpa</code>	1792, <u>1798</u>
1436, 1444, 1465, 1498, 1508,		<code>\si@num@addtmpb</code>	1790, 1798
1521, 1530, 1561, 1616, 1641,		<code>\si@num@addunit</code> ...	1993, 1997, <u>2002</u>
1647, 1668, 1714, 1853, 1861,		<code>\si@num@ambig</code>	<u>1267</u> , 1412–1414
2013, 2021, 2024, 2203, 2215,		<code>\si@num@ambigertrue</code> ..	1332, 2644
2227, 2301, 2310, 2346, 2350,		<code>\si@num@arg</code>	
2390, 2465, 2493, 2634, 2639,		556, 1279, 1291, 1298, 1305, 1384,	
2692, 2698, 2754, 2786, 2808,		1436, 1445, 1465, 1499, 1509,	
2829, 2870, 2959, 2984, 3969, 3970		1522, 1530, 1562, 1611, 1617,	
<code>\si@log@err</code> ..	124, 583, 637, 1240,	1622, 1641, 1647, 1771, 1780, 1785	
1384, 1439, 1611, 1621, 1770,		<code>\si@num@assign</code>	<u>1537</u>
1779, 1784, 2032, 2037, 2305,		<code>\si@num@checkerr</code>	1687, <u>1727</u>
2583, 2586, 2597, 2600, 2611,		<code>\si@num@cntdgt</code>	<u>1803</u>
2614, 3131, 3373, 3399, 3407, 4605		<code>\si@num@cntdigits</code>	
<code>\si@log@inf</code>	<u>132</u> , 953, 1031,	. 1750, 1752, <u>1803</u> , 1820, 1832, 1841	
1038, 1502, 1512, 1516, 2588,		<code>\si@num@dec</code>	1705, <u>1975</u>
2602, 2616, 3272, 3284, 3544,		<code>\si@num@decfmt</code>	<u>1975</u>
4042, 4049, 4134, 4135, 4515, 4516		<code>\si@num@decimalhook</code> <u>1675</u> , 1701, 2281	
<code>\si@log@warn</code>	<u>132</u> , 203,	<code>\si@num@delplusfalse</code>	1493
227, 957, 1298, 1305, 2180, 2881,		<code>\si@num@delplustrue</code>	1535
3287, 3303, 3318, 4234, 4270, 4282		<code>\si@num@digits</code>	600, 1577, <u>1608</u>
<code>\si@logminfalse</code>	230	<code>\si@num@dp</code> ..	<u>632</u> , 1845, 1848, 1853,
<code>\si@logmintrue</code>	238	1854, 1861, 1917–1919, 1921,	
<code>\si@lognonefalse</code>	231	1922, 2086, 2087, 2122, 2123, 2443	
<code>\si@lognonetrue</code>	234		
<code>\si@mathscelsius</code>	<u>806</u>		
<code>\si@mathsdegree</code>	<u>795</u> , 811, 919		

\si@num@err <u>1267</u> , <u>1288</u> , <u>1347</u> , 1357, 1358, 1371, 1372, 1406, 1692, 1738, 1746, 1816, 1829, 1995	\si@num@padtrailtrue 404, 408, 413, 418, 423
\si@num@erropenfalse <u>1417</u>	\si@num@pd <u>1852</u>
\si@num@erropenttrue <u>1353</u>	\si@num@pm 592, <u>1246</u>
\si@num@exp 557, <u>1267</u> , 1284, 1296, 1304, 1323, 1325, 1330, 1356, 1359, 1361, 1382, 1401	\si@num@post <u>1626</u> , <u>1634</u>
\si@num@expsign 572, <u>1267</u> , 1285, 1297, 1300, 1359, 1362, 1400	\si@num@postdec 596, 607, 608, 1341, 1344, <u>1571</u> , 1575, 1584, 1599, 1660, 1679, 1689, 1700, 1704, 1705, 1708, 1729, 1752, 1841, 1842, 1857, 1862, 1865, 1867, 1931, 1976, 1982, 1985, 1988, 1989
\si@num@extra <u>1709</u>	\si@num@posterr 1732, <u>1740</u>
\si@num@fiint <u>1953</u>	\si@num@postrnd ... <u>1858</u> , 1909–1911
\si@num@finddigits 1568, <u>1573</u>	\si@num@pre <u>1628</u> , <u>1634</u>
\si@num@finderr <u>1758</u>	\si@num@predec 595, 604, 605, 1340, <u>1571</u> , 1574, 1579, 1594, 1598, 1660, 1673, 1674, 1679, 1689, 1697, 1698, 1708, 1863, 1864, 1866, 1899, 1902, 1937, 1943, 1969, 1971, 1972
\si@num@findsign 1477, <u>1491</u>	\si@num@preerr 1730, <u>1734</u>
\si@num@findxpart 1291, <u>1421</u>	\si@num@prepost ... 1635, 1637, <u>1638</u>
\si@num@five <u>1945</u>	\si@num@prernd <u>1858</u> , 1887–1889, 1900
\si@num@fixdp 1694, <u>1840</u>	\si@num@procerr 1407, <u>1992</u>
\si@num@fixpm 555, 1222, <u>1246</u>	\si@num@procnun ... 1321, 1322, <u>1563</u>
\si@num@format 1239, <u>1279</u>	\si@num@psterr <u>1740</u>
\si@num@gensign 1473, <u>1476</u>	\si@num@rev <u>1870</u>
\si@num@ifextra 1680, 1697, 1704, <u>1709</u>	\si@num@reverse ... 1862, 1863, <u>1870</u>
\si@num@iffive 1941, <u>1945</u> , 1980	\si@num@rnd 1869, <u>1879</u>
\si@num@ifvalid ... 1238, <u>1251</u> , 2079	\si@num@rndpost <u>1879</u>
\si@num@in <u>1267</u> , <u>1283</u> , 1293, 1427	\si@num@rndpre <u>1879</u>
\si@num@int 1698, <u>1936</u>	\si@num@round 1846, <u>1858</u>
\si@num@intabfalse 1216	\si@num@sepdigits . 1602, 1605, <u>1676</u>
\si@num@intabtrue 2440	\si@num@seperr 1735, 1741, <u>1758</u>
\si@num@intfmt 1939, 1942, <u>1953</u>	\si@num@sepmantexp . 562, 1293, <u>1432</u>
\si@num@intsep 1955, 1958, 1961, <u>1968</u>	\si@num@sepsign 563, 571, 1294, 1295, 1468
\si@num@killsign 1526, <u>1533</u>	\si@num@sepxpart .. 1419, <u>1996</u> , 2304
\si@num@largeerr 1754, <u>1825</u>	\si@num@serr <u>1813</u>
\si@num@lerr <u>1825</u>	\si@num@sign 1479, 1489, <u>1491</u> , 1536, 1560
\si@num@mant 558, <u>1267</u> , 1286, 1302, 1326, 1327, 1336, 1383, 1391	\si@num@signexpfalse 431
\si@num@mantexp <u>1432</u>	\si@num@signexptrue 442, 446, 451, 456, 461
\si@num@mantsign 564, <u>1267</u> , 1287, 1303, 1306, 1311, 1336, 1340	\si@num@signmantfalse 430
\si@num@movedigit <u>1825</u>	\si@num@signmanttrue 434, 438, 450, 455, 460
\si@num@mp 593, <u>1246</u>	\si@num@smallerr 1756, <u>1813</u>
\si@num@nosign <u>1649</u>	\si@num@unsign 1593, <u>1649</u>
\si@num@nozero 1596, <u>1672</u>	\si@num@valid <u>1251</u>
\si@num@out 1267, <u>1335</u> , 1404, 2466, 2469, 2477, 2539, 2541, 2544, 2548	
\si@num@pad 1849, <u>1852</u>	
\si@num@padleadfalse 392	
\si@num@padleadtrue 396, 400, 412, 417, 422	
\si@num@padtrailfalse 393	

<code>\si@num@value</code>	1480, 1490, <u>1491</u>	<code>\si@opt@xchoicekey</code>	
<code>\si@num@xpart</code>	204, 278, 281, 283, 285, 288,
.....	<u>1267</u> , 1289, 1418, 1425, 2001	291, 293, 380, 382, 384, 386, 427, 671	
<code>\si@numaddn</code>	369, 372, 1678	<code>\si@out</code> 941, 3116, 3180, <u>3322</u> , 3361, 4188	
<code>\si@numcloseerr</code>	<u>369</u> , 372, 1776	<code>\si@out@maths</code>	3339, <u>3344</u>
<code>\si@numdecimal</code>		<code>\si@out@minus</code>	3348, <u>3356</u>
...	<u>369</u> , 374, 1323, 1609, 1613, 1817	<code>\si@out@num</code> 1218, 1410, 1413, 2235,	
<code>\si@numdigits</code>	369, 374, 1387	2468, 2477, 2497, 2505, 2538,	
<code>\si@numexp</code> .	<u>369</u> , 373, 1358, 1437, 1441	2541, 2547, 2557, <u>3358</u> , 4158, 4164	
<code>\si@numextra</code>	<u>371</u> , 1678, 1722	<code>\si@out@numtrue</code>	3360
<code>\si@numgobble</code>	<u>369</u> , 373, 1435	<code>\si@out@sp</code>	3345, 3346, <u>3355</u>
<code>\si@numopenerr</code>	<u>369</u> , 372, 1768	<code>\si@out@text</code>	3337, <u>3344</u>
<code>\si@numprod</code>	<u>369</u> , 374, 1422, 2672	<code>\si@outerinput</code>	3382, <u>3386</u>
<code>\si@numsign</code>		<code>\si@outerinputfalse</code>	3389
...	<u>369</u> , 373, 1494, 1495, 1620, 1623	<code>\si@outerinputtrue</code>	3377
<code>\si@numtextmodfalse</code> .	300, 308, 316	<code>\si@packagecheck</code>	<u>51</u>
<code>\si@numtextmodtrue</code> ..	304, 312, 319	<code>\si@per</code>	<u>2997</u> , 4217, 4219, 4222,
<code>\si@numvalid</code> <u>371</u> , 1242, 1262, 2075, 2344		4223, 4225, 4617, 4630, 4641,	
<code>\si@obeyboldfalse</code>	339	4650, 4654, 4658, 4671, 4674, 4677	
<code>\si@obeyboldtrue</code>	345	<code>\si@pm</code>	<u>463</u> , 1500, 1994
<code>\si@obeyfamilyfalse</code>	342	<code>\si@prefixbase</code>	<u>386</u> , 3119
<code>\si@obeyfamilytrue</code>	348	<code>\si@prefixproduct</code>	<u>384</u> , 3116
<code>\si@obeyitalicfalse</code>	340	<code>\si@prefixsymbolicfalse</code>	4426
<code>\si@obeyitalictrue</code>	346	<code>\si@repeatunitsfalse</code> ...	675, 1352
<code>\si@obeymodfalse</code>	341	<code>\si@repeatunitstrue</code>	679
<code>\si@obeymodtrue</code>	347	<code>\si@seperrfalse</code>	1423, 2303
<code>\si@old@OHMfalse</code>	4044	<code>\si@SI</code>	2577–2579, <u>2624</u>
<code>\si@openerr</code>	<u>375</u> , 1351, 1371	<code>\si@sign</code>	<u>425</u> , 1560, 1561
<code>\si@openfrac</code>	<u>685</u> , 3164	<code>\si@sis@addtolocale</code>	<u>4193</u>
<code>\si@opt@boolkey</code>		<code>\si@sis@num</code>	<u>4150</u>
...	<u>197</u> , 248, 249, 296, 327, 328,	<code>\si@sis@numstar</code>	<u>4150</u>
336, 337, 375, 376, 378, 379, 383,		<code>\si@sis@savefont</code> ..	<u>4176</u> , 4179–4184
387, 463, 521, 522, 547, 632, 641–		<code>\si@siu@newprefix</code>	
643, 649, 652, 685, 726, 727, 765, 816		<u>4420</u> , 4427–4446, 4448–4453
<code>\si@opt@choicekey</code>	200,	<code>\si@siu@newunit</code>	4309,
228, 299, 307, 315, 321, 330, 338,		4341–4381, 4680, 4681, 4684–4687	
390, 428, 472, 501, 524, 610, 621,		<code>\si@siu@newunithook</code> ...	<u>4309</u> , <u>4610</u>
653, 674, 687, 728, 735, 742, 751, 766		<code>\si@siu@newunitx</code> <u>4382</u> , 4389, 4391,	
<code>\si@opt@cmdkey</code>	<u>193</u> , 554	4393, 4395, 4397–4399, 4401,	
<code>\si@opt@cmdkeys</code>	<u>195</u> ,	4403, 4405, 4406, 4408, 4411, 4682	
350, 351, 364, 365, 369, 377, 686,		<code>\si@siu@newunitxhook</code> ..	<u>4382</u> , <u>4610</u>
714, 760, 773, 775, 779, 795, 806, 812		<code>\si@siu@power</code> .	<u>4309</u> , 4618, 4631, 4642
<code>\si@opt@compatkey</code>		<code>\si@siu@setup</code>	<u>4231</u>
.....	<u>215</u> , 829–835, 837–876	<code>\si@siu@unity</code>	<u>4410</u>
<code>\si@opt@disablekey</code>	220,	<code>\si@slash</code>	<u>671</u> , 941
223, 264, 269, 275, 717, 720, 818		<code>\si@slashfalse</code>	655
<code>\si@opt@key</code>	<u>190</u> ,	<code>\si@slashttrue</code>	660
227, 273, 352–363, 366–368, 545,		<code>\si@str@chrstr</code>	<u>150</u>
634, 715, 724, 761–764, 774, 776,		<code>\si@str@ifchrstr</code>	
780, 797–799, 807, 813, 823–825, 828		<u>150</u> , 184, 1262, 1422,
<code>\si@opt@proctform</code>	579, 580, <u>594</u>		

1435, 1437, 1494, 1495, 1609,
 1620, 1722, 1768, 1776, 2344, 2672
 \si@str@ifonlychrs
 ... 171, 635, 1323, 1660, 1673, 1690
 \si@str@onlychrs 171
 \si@svn@id 1
 \si@svn@ver 1
 \si@svn@version 1, 3549,
 3581, 3618, 3703, 3759, 3774,
 3795, 3804, 3813, 3834, 3839,
 3851, 3860, 3869, 3878, 3889,
 3923, 4138, 4207, 4518, 4571, 4601
 \si@switchfalse
 . 152, 185, 561, 597, 1266, 1290,
 1292, 1576, 1661, 1711, 1759,
 1868, 1894, 1916, 2322, 3529, 4196
 \si@switchtrue 165, 173, 1253, 1423,
 1447, 1618, 1659, 1722, 1774,
 1897, 1924, 1929, 2345, 3533, 4198
 \si@sym@celsius
 907, 3609, 3612, 4126, 4190
 \si@sym@degree 907, 2186,
 2213, 2225, 2234, 2253, 3213,
 3225, 3780, 3783, 4128, 4191, 4192
 \si@sym@minute 907,
 2187, 2211, 2223, 2254, 3786, 4302
 \si@sym@mu
 .. 907, 3214, 3226, 3558, 3561, 4127
 \si@sym@Omega .. 907, 3595, 3598, 4125
 \si@sym@ringA .. 907, 3215, 3227, 3761
 \si@sym@second
 .. 907, 2188, 2209, 2255, 3787, 4303
 \si@symbol 899, 907–913
 \si@tab@begin 2308, 2312, 2316
 \si@tab@begin@S 2290, 2300
 \si@tab@begin@s 2296, 2300
 \si@tab@end 2425
 \si@tab@end@S 2291,
 2331, 2332, 2400, 2426, 2428, 2429
 \si@tab@end@s 2297,
 2373, 2374, 2400, 2432, 2434, 2435
 \si@tab@expbox
 . 2449, 2463, 2485, 2496, 2552, 2556
 \si@tab@expdim . 2486, 2508, 2513,
 2516, 2525, 2532, 2535, 2552, 2556
 \si@tab@expout
 . 1267, 1335, 2466, 2544, 2550, 2557
 \si@tab@exppostcnt
 548, 2512, 2513, 2528
 \si@tab@expprecnt
 548, 2507, 2508, 2527
 \si@tab@expsignfalse 560
 \si@tab@expsigntrue 574, 577
 \si@tab@fixed 2455, 2492
 \si@tab@fixedfalse 539, 543
 \si@tab@fixedtrue 526
 \si@tab@format 2448, 2453
 \si@tab@gettok 2300, 2323
 \si@tab@gettok@S 2302, 2324
 \si@tab@gettok@s 2311, 2324
 \si@tab@lfill 2400
 \si@tab@lfill@S 523, 2402
 \si@tab@lfill@s 622, 2419
 \si@tab@lfill@t 611, 2406
 \si@tab@mantpostcnt
 548, 581, 2443, 2501
 \si@tab@mantprecnt .. 548, 582, 2499
 \si@tab@mantsignfalse 559
 \si@tab@mantsigntrue 566, 569
 \si@tab@midbox
 2449, 2462, 2470, 2473, 2495
 \si@tab@newline@S 2326, 2425
 \si@tab@newline@s 2368, 2425
 \si@tab@next 2324
 \si@tab@numout 2411, 2439
 \si@tab@othertok 2324
 \si@tab@out ... 1267, 1335, 2468, 2538
 \si@tab@postbox 2449, 2462,
 2476, 2478–2480, 2482, 2540, 2546
 \si@tab@postdim
 . 2486, 2501, 2503, 2535, 2540, 2546
 \si@tab@posttoks
 2313, 2321, 2362, 2412
 \si@tab@prebox 2449, 2462,
 2467, 2478, 2479, 2482, 2483, 2537
 \si@tab@predim 2486, 2499, 2521, 2537
 \si@tab@pretoks
 2313, 2320, 2364, 2410, 2414
 \si@tab@rfill 2400
 \si@tab@rfill@S 523, 2403
 \si@tab@rfill@s 622, 2423
 \si@tab@rfill@t 611, 2405
 \si@tab@sepcorr
 2500, 2504, 2509, 2517, 2561
 \si@tab@sp 2491, 2494, 2511, 2572
 \si@tab@toks 2313, 2319,
 2348, 2349, 2388, 2389, 2421, 2447
 \si@tab@unfixed 2457, 2464
 \si@tab@numalign 523
 \si@tempa 42, 56, 57, 59, 60,
 62, 65, 66, 69, 70, 153, 164, 174,
 176, 201, 205, 233, 237, 241, 245,
 259, 264, 265, 303, 311, 318, 324,

333, 344, 395, 399, 403, 407, 411,
416, 421, 433, 437, 441, 445, 449,
454, 459, 477, 481, 485, 490, 495,
506, 510, 514, 518, 530, 534, 538,
542, 614, 618, 625, 629, 658, 664,
668, 678, 682, 691, 695, 699, 703,
707, 711, 725, 732, 739, 747, 756,
770, 923, 924, 1013, 1018, 1021,
1025, 1045, 1046, 1112, 1187,
1235, 1237–1239, 1310, 1311,
1337, 1342, 1345, 1369, 1375,
1377, 1380, 1393, 1394, 1398,
1399, 1539, 1541, 1679, 1680,
1689, 1690, 1760, 1765, 1826,
1830, 1838, 1871, 1873, 1875,
1886, 1890, 1908, 1912, 2026,
2028, 2031, 2062, 2064, 2094,
2099, 2102, 2106, 2110, 2133,
2136, 2138, 2140–2142, 2145,
2146, 2151, 2157, 2167, 2169,
2175, 2178, 2283, 2287, 2289,
2292, 2295, 2298, 2658, 2663,
2889, 2891, 2899, 2901, 2917,
2920, 2937, 2940, 2952, 2956,
2980, 2983, 3035, 3037, 3044,
3047, 3056, 3058, 3062, 3075,
3077, 3174, 3181, 3195, 3196,
3219, 3220, 3291, 3293, 3296,
3301, 3308, 3310, 3311, 3316,
3412, 3414–3417, 3419, 3426,
3429, 3506, 3507, 3515, 3517,
3520, 3527, 3532, 3536, 3538,
3539, 3966, 3999, 4000, 4010,
4011, 4021, 4022, 4032, 4033,
4232, 4310, 4313, 4314, 4321,
4328, 4330, 4337, 4421, 4425,
4611, 4621, 4623, 4634, 4636, 4645
\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, 613, 614, 617,
618, 624, 625, 628, 629, 657, 658,
663, 664, 667, 668, 677, 678, 681,
682, 690, 691, 694, 695, 698, 699,
702, 703, 706, 707, 710, 711, 731,
732, 738, 739, 746, 747, 755, 756,
769, 770, 1017, 1018, 1113, 1194,
1540, 1541, 1684, 1686, 1736,
1738, 1742, 1746, 1750, 1761,
1764, 1818, 1820, 1822, 1830,
1832, 1834, 1839, 2027, 2028,
2030, 2031, 2063, 2064, 2659,
2663, 3045, 3047, 3057, 3058,
3061, 3062, 3076, 3077, 3292,
3293, 3300, 3301, 3309, 3310,
3315, 3316, 3396, 3414, 3416,
3418, 3423, 3428, 3430, 3513,
3517, 3530, 3532, 4311, 4314,
4323, 4329, 4332, 4338, 4422,
4426, 4613, 4616, 4625, 4628, 4639
\si@tempboxa
46, 1019, 1020, 1028, 1035, 2258,
2260, 2262, 2263, 2265, 2267,
2271, 2274, 2277, 2497, 2498,
2502, 2503, 2505, 2506, 2514,
2516, 2519, 2521, 2523, 2525,
2530, 2532, 2571, 2574, 2682, 2683
\si@tempboxb 46, 2261, 2269
\si@tempboxc 46, 2266, 2269
\si@tempboxd 46, 2268, 2272, 2275
\si@tempc 42, 163, 164, 1685,
1686, 1699, 1702, 1708, 3418–
3420, 3429–3431, 3514, 3518,
3520, 3522, 3526, 3536, 3538,
4615, 4622, 4627, 4635, 4638, 4646
\si@tempcnta 1751, 1753,
1804, 1808, 1814, 1821, 1827,
1833, 1843, 1845, 1848, 1854,
1856, 1880, 1918, 1922, 1934,
2437, 2527–2529, 2562, 2564,
2568, 2574, 2652, 2656–2658,
2669, 2673, 2926, 2930, 2932, 2933
\si@tempcntb 1751, 1753, 1814, 1821,
1827, 1833, 1890, 1892, 1895,
1896, 1899, 1912, 1914, 1923,
1927, 1928, 1931, 2437, 2932, 2933
\si@tempdima 2093, 2096,
2098, 2102, 2105, 2110, 2115,
2126, 2128, 2131, 2133, 2137,
2138, 2166, 2170, 2260, 2270,
2271, 2278, 2486, 2498, 2499,
2501, 2506, 2508, 2513, 2992, 2993
\si@tempdimb
..... 2265, 2270, 2274, 2278, 2486
\si@temptoks 50, 89,
91, 2147, 2149, 2152, 2154, 2158,
2160, 2163, 2864, 2869, 2870,
2877, 2878, 3170, 3175, 3192,

3193, 3195, 3201, 3202, 3274, 3278
 \si@textcelsius 806, 4019
 \si@textdegree .. 795, 809, 918, 4030
 \si@textminute 795
 \si@textmodefalse 1076, 1082
 \si@textmodetrue 1074, 1080
 \si@textmu 779, 921, 4008, 4300
 \si@textOmega 775, 922, 3997
 \si@textringA 812, 927
 \si@textrm 1085, 1117, 1183, 1205
 \si@textsecond 795
 \si@textsf 1085, 1170, 1192
 \si@texttt 1085, 1177, 1199
 \si@tothe 3249, 3250, 3251
 \si@unitcolour 726, 1108
 \si@unitmathsrm 350, 1099
 \si@unitmathssf 350, 1100
 \si@unitmathstt 350, 1101
 \si@unitsep 278, 2884, 3172
 \si@unitSIdef 3979
 \si@unitSPACE 278, 3173
 \si@unittextmodefalse 301, 309, 322
 \si@unittextmodetrue . 305, 313, 325
 \si@unittextrm 364, 1102
 \si@unittextsf 364, 1103
 \si@unittexttt 364, 1104
 \si@unt@addprefix 2908, 2913
 \si@unt@addstack 3065, 3074
 \si@unt@addtostack
 2833, 2884, 2914, 2986, 3034
 \si@unt@addvalsep 2704
 \si@unt@addvaluesep 2696, 2704, 2860
 \si@unt@checkstack
 2891, 2901, 2982,
 3037, 3046, 3055, 3070, 3076,
 3081, 3085, 3089–3091, 3093, 3101
 \si@unt@cntx 2670, 2671, 2676
 \si@unt@countprefix ... 2910, 2916
 \si@unt@countx 2654, 2668
 \si@unt@defpower
 2610, 2615, 2617, 2620, 2807
 \si@unt@defprefix
 2596, 2601, 2603, 2606, 2783
 \si@unt@defunit
 2582, 2587, 2589, 2592, 2753
 \si@unt@depthcnt 2701,
 2718, 2825, 2828, 2829, 2838, 2839
 \si@unt@final 2703, 2746, 2840
 \si@unt@first 2844, 2849
 \si@unt@firstfalse 2861
 \si@unt@firstorsecond
 2831, 2842, 2906, 2943, 3014
 \si@unt@firsttrue 2730
 \si@unt@fracout 3097, 3126
 \si@unt@fullstop 3171, 3184
 \si@unt@holdspace 3063, 3074
 \si@unt@holdstacka 2742, 3072
 \si@unt@holdstackb 2743, 3072
 \si@unt@ifliteral . 2678, 2691, 2832
 \si@unt@incnt 3059, 3068
 \si@unt@init 2700, 2718, 2826
 \si@unt@invpower 2949, 2991
 \si@unt@invprefix 2916
 \si@unt@lastadda 2744, 3032
 \si@unt@lastaddb 2745, 3032
 \si@unt@litoutfalse 2728
 \si@unt@litouttrue 2695, 3095, 3991
 \si@unt@litpower .. 2817, 2934, 3256
 \si@unt@litprefixfalse 2729
 \si@unt@litprefixtrue 2784
 \si@unt@littesttrue 2681
 \si@unt@litvalsep 2704
 \si@unt@nonlatin 3177, 3207
 \si@unt@noopt 2763, 2766
 \si@unt@normout ... 3099, 3122, 3147
 \si@unt@notabg 3154
 \si@unt@notambig 3127, 3154
 \si@unt@numfalse 644, 2631
 \si@unt@numtrue 646, 2003, 2648, 2852
 \si@unt@opt 2768–2770, 2771
 \si@unt@out 935, 936, 940, 942, 944,
 945, 948, 949, 2682, 2697, 3125, 3166
 \si@unt@per 2997
 \si@unt@perfalse .. 2731, 2967, 3017
 \si@unt@perseenfalse .. 2732, 2968
 \si@unt@perseenttrue ... 2896, 4227
 \si@unt@pertrue ... 3020, 4227, 4230
 \si@unt@power 2819, 2936, 3258
 \si@unt@powerdim 2735,
 2946, 2959, 2962, 2971, 2979,
 2985, 2987, 2990, 2992–2995, 3030
 \si@unt@prefix 2802, 2905
 \si@unt@prefixcnt
 2738, 2915, 2930, 3114, 3120
 \si@unt@prefixout
 3113, 3123, 3137, 3140, 3149
 \si@unt@preplussp 3040, 3043
 \si@unt@prepowerfalse . 2733, 2964
 \si@unt@prepowertrue 2953

<code>\si@unt@printunit</code>	<code>\sievvert</code>
. 2006, 2635, 2651, 2661, 2665, 2689	15 , 3594 , 3701
<code>\si@unt@recip</code>	<code>\sievvertbase</code>
3023	4497
<code>\si@unt@reciptest</code>	<code>\SIfig</code>
2903 , 3023	2627, 3728
<code>\si@unt@second</code>	sign (option)
2846 , 2879	25
<code>\si@unt@setopts</code>	<code>\SIGroupfourfalse</code>
2855 , 2862	4167
<code>\si@unt@setSIopts</code>	<code>\SIGroupfourtrue</code>
2862	4167
<code>\si@unt@SIopts</code>	<code>\SIMathrm</code>
2624 , 2873, 2877	4179
<code>\si@unt@spacecheck</code>	<code>\SIMathsf</code>
2893 , 2898	4179
<code>\si@unt@spaceout</code>	<code>\SImathhtt</code>
.....	4179
3111 , 3124, 3141, 3150	<code>\SIobeyboldfalse</code>
<code>\si@unt@spstack</code> ...	4148
2715, 2718 , 3112	<code>\SIobeyboldtrue</code>
<code>\si@unt@stack</code>	4148
3050, 3053	<code>\SIproductsign</code>
<code>\si@unt@stacka</code> 2718 , 3125, 3128, 3151	4172
<code>\si@unt@stackb</code>	<code>\SIsetup</code>
2718 , 3078,	4231
3129 , 3138, 3142, 3146, 3151, 3164	<code>\sisetup</code>
<code>\si@unt@stackout</code>	4, 21 , 189, 218,
2748, 3094	251, 352–363, 366–368, 545, 590,
<code>\si@unt@stackpower</code> 2957, 2963 , 3031	761–764, 774, 776, 780, 797–799,
<code>\si@unt@stackvalsep</code> ...	807, 813, 1060, 1062, 1214, 2011,
2704, 3118	2050, 2113, 2250, 2318, 2629,
<code>\si@unt@stkpower</code> ..	2869, 2878, 2921, 3265, 3266,
2835 , 2887, 2963	3269, 3278, 3433, 3852, 3861,
<code>\si@unt@stkpwr</code>	3870, 3880, 3895, 3898, 3901,
2963	3904, 3916, 3933, 3935, 3938,
<code>\si@unt@stp</code>	3941, 3944, 3947, 3950, 3953,
3184	3956, 3959, 3962, 3981, 3992,
<code>\si@unt@sym</code> 3213–3215, 3225–3227, 3230	4006, 4017, 4028, 4039, 4141,
<code>\si@unt@third</code>	4148, 4149, 4157, 4162, 4163,
2747, 2879	4167–4175, 4178, 4187, 4193,
<code>\si@unt@unit</code>	4194, 4210, 4217, 4219, 4222,
2778, 2823	4223, 4225, 4227, 4230, 4239,
<code>\si@unt@unitarg</code> 2004, 2006, 2623 , 2637	4242, 4245, 4248, 4251, 4254,
<code>\si@unt@unitcnta</code>	4257, 4293, 4296, 4574, 4576–
2718	4578, 4580, 4583, 4586, 4589,
<code>\si@unt@unitcntb</code>	4592, 4595, 4598, 4616, 4628,
2718 , 3155	4639, 4649, 4652, 4656, 4660, 4662
<code>\si@unt@unithook</code> ..	<code>\SIstyle</code>
2823 , 2853, 3991	4193
<code>\si@unt@withopt</code>	<code>\SIstyleToLang</code>
2761, 2766	4193
<code>\si@valuecolour</code>	<code>\SIthousandsep</code>
726 , 1097, 2460	4172
<code>\si@valuemathrm</code>	<code>\SIunitdot</code>
350, 1088	4169
<code>\si@valuemathssf</code>	<code>\SIunitsep</code>
350 , 1089	4169
<code>\si@valuemathstt</code>	<code>\SIunitspace</code>
350 , 1090	4169
<code>\si@valuesep</code>	slash (option)
278 ,	29
1019, 2715, 2717, 3919, 3965, 4213	<code>\Square</code>
<code>\si@valuetextrm</code>	12 , 3244 , 3679, 3680, 3682, 4075–
364 , 1091	4077, 4222, 4537–4539, 4549,
<code>\si@valuetextsf</code>	4561, 4563, 4565, 4567, 4569, 4682
364 , 1092	<code>\square</code>
<code>\si@valuetextttt</code>	4260
364 , 1093	<code>\squarecentimeter</code>
<code>\si@xargdef</code>	4075
100	<code>\squarecentimetre</code>
<code>\si@xifmtarg</code>	17 , 3679
92	<code>\squared</code>
<code>\si@xspacefalse</code>	12 , 3244 ,
2632, 3985	3681, 3727, 4228, 4400, 4494, 4499
<code>\SIdecimalsign</code>	<code>\squarekilometer</code>
4172	4075
<code>\SIdecimalsymbol</code>	
4172	
<code>\SIdefaultMfam</code>	
4182	
<code>\SIdefaultNfam</code>	
4182	
<code>\SIdefaultTfam</code>	
4182	
<code>\siemens</code>	
15 , 3594 , 3659	
<code>\siemensbase</code>	
4497	

<code>\squarekilometre</code>	17 , 3679	<code>\tothe</code>	12 , 2658 , 3249 , 3993 , 4221
<code>\squaremeter</code>	4075	<code>\tothe@opt@si</code>	3251
<code>\Squaremetre</code>	4679	<code>trapambigerr (option)</code>	24
<code>\squaremetre</code> 17 , 3679 , 4390 , 4391 , 4396 , 4402 , 4412 , 4418 , 4457 , 4469 , 4482 , 4487 , 4488 , 4491 , 4496 , 4501 , 4505 , 4507 , 4508 , 4510 , 4679	<code>trapambigfrac (option)</code>	29
<code>\squaren</code>	4260	U	
<code>\ssquare</code>	12 , 3244 , 4315	<code>\uBar</code>	4107
<code>\steradian</code>	15 , 3615 , 4402 , 4473	<code>\UFrac</code>	4670
<code>\steradianbase</code>	4480	<code>\Ufrac</code>	4670
<code>stickyper (option)</code>	29	<code>\ufrac</code>	4670
<code>strict (option)</code>	32	<code>\unit</code>	3906 , 3976 , 4287
<code>strictarc (option)</code>	26	<code>\unita</code>	4287
T		<code>unitcolor (option)</code>	31
<code>tabalign (option)</code>	28	<code>unitcolour (option)</code>	31
<code>tabalignexp (option)</code>	27	<code>\unitfrac</code>	3912 , 3987
<code>tabautofit (option)</code>	28	<code>unitmathsrm (option)</code>	23
<code>tabformat (option)</code>	27	<code>unitmathssf (option)</code>	23
<code>tabnumalign (option)</code>	27	<code>unitmathstt (option)</code>	23
<code>tabtextalign (option)</code>	28	<code>unitmode (option)</code>	22
<code>tabunitalign (option)</code>	28	<code>\unitsep</code>	3989
<code>\tebi</code>	20 , 3838	<code>unitsep (option)</code>	23
<code>\tera</code>	14 , 3567 , 3690 , 3696 , 3715 , 3742 , 4545 , 4559 , 4569	<code>\unitSIdef</code>	3979
<code>\teraelectronvolt</code>	17 , 3683	<code>\unitsignonly</code>	3972
<code>\terahertz</code>	17 , 3692	<code>unitspace (option)</code>	23
<code>\tesla</code>	15 , 3594	<code>\unitsuperscript</code>	3989
<code>\teslabase</code>	4497	<code>unittextrm (option)</code>	23
<code>\TeV</code>	17 , 3736	<code>unittextsf (option)</code>	23
<code>\TeVoverc</code>	4550	<code>unittexttt (option)</code>	23
<code>\TeVovercsq</code>	4560	<code>\unittimes</code>	3989
<code>textcelsius (option)</code>	30	<code>\unitvaluesep</code>	3964 , 3975 , 3986
<code>textdegree (option)</code>	30	<code>\unskip</code>	2387 , 2422
<code>textminute (option)</code>	30	<code>\usk</code>	4217 , 4478
<code>textmode (option)</code>	22	V	
<code>textmu (option)</code>	30	<code>valuecolor (option)</code>	31
<code>textOmega (option)</code>	30	<code>valuecolour (option)</code>	31
<code>textringA (option)</code>	30	<code>valuemathrm (option)</code>	23
<code>textrm (option)</code>	23	<code>valuemathsf (option)</code>	23
<code>textsecond (option)</code>	30	<code>valuemathsrm (option)</code>	23
<code>textsf (option)</code>	23	<code>valuemathssf (option)</code>	23
<code>texttt (option)</code>	23	<code>valuemathstt (option)</code>	23
<code>\THz</code>	17 , 3710	<code>valuemathttt (option)</code>	23
<code>tightpm (option)</code>	24	<code>valuemode (option)</code>	22
<code>\TinveV</code>	4540	<code>valuesep (option)</code>	23
<code>\ton</code>	4047 , 4302 , 4664	<code>valuetextrm (option)</code>	23
<code>\tonne</code>	15 , 3773	<code>valuetextsf (option)</code>	23
<code>\torr</code>	19 , 3803	<code>valuetextttt (option)</code>	23
		<code>\volt</code>	15 , 3594 , 3649 , 3650 , 3721 , 3722 , 4083 , 4084 , 4465 – 4467
		<code>\voltbase</code>	4480

W			
\watt 15, 3594, 3651– 3653, 3691, 3743, 4401, 4405, 4464	\yobi 3838
\wattbase 4480	\yocto	. 14, 3548, 3826, 3832, 4529, 4535
\weber 15, 3594, 4470, 4471	\yctobarn 20, 3821
\weberbase 4497	\yotta 14, 3567
X		Z	
\XeTeXrevision 3208	\zb 20, 3827
xspace (option) 30	\zebi 3838
Y		\zepto	. 14, 3548, 3825, 3831, 4528, 4534
\yb 20, 3827	\zeptobarn 20, 3821
		\zetta 14, 3567

28 References

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