

si — A comprehensive (SI) units package*

Joseph Wright[†]

Released 2008/02/20

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). A number of \LaTeX packages have been developed to provide consistent application of the various rules: Slunits, sistyle, unitsdef and units are the leading examples. The numprint package provides a large number of number-related functions, while dcolumn and rccol provide tools for typesetting tabular numbers.

The si package takes the best from the existing packages, and adds new features and a consistent interface. A number of new ideas have been incorporated, to fill gaps in the existing provision. The package also provides backward-compatibility with Slunits, sistyle, unitsdef and units. The aim is to have one package to handle all of the possible unit-related needs of \LaTeX users.

Contents		7 Tabular material	7
		8 Angles	8
I Introduction	2	9 Units and values	9
1 Existing packages	3	9.1 Literal units	10
2 The wish-list	4	9.2 The unit interpreter . . .	10
3 Road map	5	9.3 Powers of units	10
		9.4 Units with no values . . .	11
		9.5 Free-standing units	11
		9.6 Pre-defined units, pre- fixes and powers	11
II Using the si package	5	9.7 Prefixed and abbreviated units	12
4 Requirements	5	9.8 Specialist units	16
5 Loading the package	6	9.9 Defining new units . . .	17
6 Numbers	6	10 Font control	18

*This file describes version v.06a, last revised 2008/02/20.

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

11 Package options	18	16.14 Output routine	84
11.1 Font family and style . . .	19	16.15 Finalisation	85
11.2 Spacing and separators . . .	20	17 Loadable modules	88
11.3 Number formatting	20	17.1 Multiple prefixes	88
11.4 Angle formatting	21	17.2 Derived units with specific names	89
11.5 Tabular material	22	17.3 Units with prefixes	90
11.6 Units	22	17.4 Abbreviated units	92
11.7 Symbols	23	17.5 Additional (temporary) SI units	93
11.8 Package control	24	17.6 Units accepted for use with SI	94
11.9 Back-compatibility options	24	17.7 Units based on physical measurements	94
12 Emulation of other packages	24	18 Additional configurations	95
13 Tricks and known issues	25	18.1 Synthetic chemistry	95
14 Reporting a problem	25	18.2 High-energy physics	95
15 Acknowledgements	26	18.3 Binary units	97
III Correct application of (SI) units	26	19 Loadable locales	97
IV Implementation	26	19.1 United Kingdom	97
16 Main package	26	19.2 United States	97
16.1 Setup code	27	19.3 Germany	98
16.2 Logging	29	19.4 South Africa	98
16.3 Option handling	30	20 Emulation code	98
16.4 Compatibility options	40	20.1 units	98
16.5 Constants	41	20.2 unitsdef	99
16.6 Symbols	42	20.3 sistyle	105
16.7 Handling fractions	43	20.4 Slunits	106
16.8 Font control	45	V Notes	114
16.9 Formatting numbers	49	21 Change History	114
16.10 Formatting angles	59	22 Index	114
16.11 New column types	62	23 References	127
16.12 Units	68		
16.13 Locales	83		

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.

L^AT_EX support typesetting numbers and units is currently provided by a number of packages: Slunits, sistyle, units, unitsdef and numprint. Each package has advantages, and no single package has so far displaced use of the others. The aim of the si package is to learn from the existing implementations to provide a coherent and extendable approach to the problem. The original aim of developing si was to produce a “version 2” successor to Slunits or sistyle. However, as the package has been developed a number of inconsistencies in the interfaces of the existing packages have been noted. Thus by default si does not follow any one of the existing packages; the interface is intended to be self-consistent and logical. As a result, si is now intended as a new package. The author hopes that by providing a comprehensive package here, the other “unit” packages will be superseded.

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. In emulation mode, si tries hard to work in exactly the same manner as the emulated package. However, in certain places inconsistencies exist due to changes in the underlying mechanisms used. These are noted where they are known to the package author.

1 Existing packages

Both the Slunits and sistyle package are designed to allow typesetting of SI units, with consistent typography and following the rules laid down by NIST [1].³ The sistyle package concentrates on typography, whereas the Slunits package is focussed on careful application of SI units in place of other systems. The key strengths of the two packages can be summarised as follows:⁴

- sistyle
 - Easy input for numbers, for example typing `\num{5.8e-7}` and getting “ 5.8×10^{-7} ” as output.
 - Input of numbers can be with comma or dot as decimal sign and is independent of output.
 - Output style can follow particular regional conventions (*via* `\sistyle`) or even be dependent on the document language (implemented by `\sistyleToLang`).
- Slunits
 - The look of units can be easily be changed in the whole document by redefining the commands.

¹General Conference on Weights and Measures.

²International System of Units.

³sistyle also allows the use of German and South African rules “out of the box”.

⁴Thanks to Stefan Pinnow for the excellent summary on `comp.text.tex`, on which this is based.

- New units can be added on a document-specific basis (`\addunit`), for example to match journal requirements (e.g. “wt.-%” versus “wt%”).
- Package aims to enforce use of SI units as far as possible.

At the same time, the `unitsdef` package allows “trailing” units, so for example `10\metre` to yield “10 m” with a non-breaking and definable space. However, this does not allow control of the format of the number. The `unitsdef` package is built on top of `units`, which is an even more general. The `numprint` package provides fine control for printing numbers, with features beyond those in `sistyle`. Finally, the `hepunits` package adds various physics units to `Slunits`.

2 The wish-list

The wish-list for the new package has developed as ideas have suggested themselves. This has been both from the package author and various contributors on `comp.text.tex`. Anything on the list is likely to be looked at: nothing is ruled out. Items marked **To do** are definitely going to be looked at, those marked **Ongoing** have at least some code written. Items marked **Completed** seem to work properly, and at this stage seem to be finished (in the sense that changes are not planned). However, that does not mean they are finalised or bug free!

- `keyval` package interface, with modification of settings in document using this system (like `hyperref`). **Completed**
- Remove need for `\usk` separator between unit names when using `Slunits`-style setup. **Completed**
- “Prefix” units, such as currency, possibly as an optional argument to `\SI`: `\SI[per=slash]{10}[\pounds]{\per\kilo\gram}` ⇒ “£10/kg” (suggested by Allan Ristow). **Completed**
- Stand-alone setting of units, for example `\unitsym{kg.m/s^2}` to give “kg m/s²”, for use in table headings, *etc.* (suggested by Allan Ristow). **Completed**
- `numprint`-like handling of numbers (suggested by Allan Ristow). **Completed**
- Ability to handle crystallography-style estimated standard deviations, e.g. `\SI{1.550(2)}{\angstrom}` ⇒ “1.550(2) Å”.⁵ **Completed**
- Ability to understand and alter negative powers/fractions, and typeset these flexibly; thus `\unitsym{\metre\per\second}` could give “m s⁻¹”, “m/s”, “ $\frac{m}{s}$ ” or “m/s” depending on a package setting (suggested by Stefan Pinnow). **Completed**
- More logical handling of powers; for example `\deci\cubic\metre` or `\deci\metre\cubed` give “dm³”, but `\deci\cubed\metre` does not even though `\deci` cannot be cubed (adapted suggestion from Stefan Pinnow). **Completed**

⁵(1.550 ± 0.002) × 10⁻¹⁰ m.

- Use of trailing units (as in `unitsdef` package, so for example `10\metre` to give “10 m” (suggestion from Lan Thuy Pham). **Completed**
- Support use of non-Latin characters where appropriate, for example μ in units as in `\unitsym{\mu m}` to give “ μm ” (suggested by Martin Heller). **Completed**
- Integrate the core functionality of the `Slunits` and `sistyle` packages (suggested by Danie Els,⁶ as well as a key point of the review). **Completed**
- Modular design, with loadable definitions for different areas and typographic conventions. **Completed**
- Emulation of existing packages [`units` (**Completed**), `unitsdef` (**Completed**), `sistyle` (**Completed**), `Slunits` (**Completed**)] to allow easy upgrading. **Completed**
- Typesetting angles in “astronomy” style, for example `\ang[astroang]{30;5;3.2}` to give $30^{\circ}5'3''.2$ (suggested by Alok Singh). **Completed**

3 Road map

The existing units packages provide valuable information on the problems and pitfalls of designing a package in this area. They have also shown how to solve many of the issues arising. However, in writing a new package, consistent interface design has been important. This is logical to the package author, but may not be to anyone else. The functionality provided also aims to cover everything from the existing packages and the suggestions contributed at `comp.text.tex`, but omissions are likely to exist. The current release of `si` is therefore regarded as a development version, to gain feedback from users and to find errors. The current “road map” for future releases is (broadly) given here.

vo.6 Seek feedback on implementation to date (current release);

vo.7 Add or modify functionality based on feedback, implement new suggestions for wish list;

vo.8 Fix bugs from 0.7 release, interface freeze;

vo.9 Release-candidate: fix remaining bugs from vo.8, complete remaining documentation;

v1.0 First release of completed package.

Depending on user feedback, the gap between these releases will vary. However, to finalise all of the potential issues will take some months (to allow time for proper testing).

⁶Current maintainer of `sistyle`.

Part II

Using the si package

4 Requirements

si requires a reasonably up to date T_EX system. The package requires ϵ -T_EX-extensions, which should be available on most systems.⁷ 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 $\mathcal{A}\mathcal{M}\mathcal{S}$ T_EX 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 L^AT_EX₃ packages. As a result, si does not load `xfrac`. If you want to use `fraction=sfrac`, you need to load `xfrac` in your preamble. 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.⁸

5 Loading the package

si is loaded by the usual L^AT_EX method.

```
\usepackage{si}\  
\usepackage[key=option]{si}
```

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

6 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.

⁷If you have an old L^AT_EX try “`elatex`” rather than “`latex`”.

⁸On the other hand, some fractional units will look really bad with `\sfrac`. Use this option with caution.

1 123 12345 123 456	<code>\num{1}</code>	<code>\num{123}</code>	<code>\num{12345}</code>	<code>\num{123456}\</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>	

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.⁹

$1.234(5) = 1.234 \pm 0.005$ `\num{1.234(5)} = \num{1.234} \pm \num{0.005}`

A number of effects are available as options. These are fully explained in Section 11. Some of the more useful options are illustrated here. By default, the output of the package is typeset in maths mode. However, the use of the current text font can be forced.¹⁰

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

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

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

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

1234 1 234	<code>\num{1234}</code>	<code>\num[sepfour=true]{1234}\</code>
12345 12,345	<code>\num{12345}</code>	<code>\num[digitsep=comma]{12345}\</code>
12345	<code>\num{12345}</code>	<code>\num[digitsep=none]{12345}</code>

The formatting of exponents is also customisable.

1×10^{10} $1 \cdot 10^{10}$	<code>\num{1e10}</code>	<code>\num[expproduct=cdot]{1e10}\</code>
2×10^{20} 1×5^{10}	<code>\num{2e20}</code>	<code>\num[exppower=5]{1e10}</code>

7 Tabular material

Centring numbers in tabular content is handled by a new column type, the *s* column. This is based closely on the *dcolumn* method for centring numbers in columns, but adds the functionality of the `\num` macro.¹¹

By default, the decimal marker of the contents is placed at the centre of the column (Table 1). This is achieved by having a negative value for the key `tabformat`. The second method for centring content is to specify a number of digits before and after the decimal sign to be reserved by the package. Thus in the example, `tabformat=2.4` provides space for two digits before the decimal

⁹This is common in chemical crystallography.

¹⁰This document is typeset using lowercase numbers in text mode, which emphasises the effect here.

¹¹The approach used is actually a combination of *dcolumn* for centring the material and `numprint` for processing it. It will therefore give rather different results than the *n* and *N* column types in `numprint`.

Table 1: Behaviour of s column type

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

marker and four after.¹² If an integer is given as the argument of `tabformat`, equal space is reserved before and after the decimal marker for numerals, and the column is typeset flush right. As is shown in the fourth column, any other options may also be set on a column-by-column basis.

```

\begin{table}
  \caption{Behaviour of \texttt{s} column type
  \label{tbl:default}}
  \centering
  \begin{tabular}{ss[tabformat=2.4]s[tabformat=4]s[tabformat=2.4,decimalsign=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}

```

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). The contents of non-numeric cells are centred; this can be altered by using the standard `\multicolumn` macro. The use of digit separators in table columns is accounted for: extra space is reserved if digit separators will be added.

The use of exponents in the body of tables is not recommended; unlike `numprint`, `si` does not provide additional alignment of exponents. Certain strange effects can also result from the exponent marker letters (by default `dDeE`) being picked up by the package in text in columns. For example, using `<{after}` after a `s` column will add “`aftr`” at the end of each value. To avoid this, wrap any problematic text in *two* sets of braces (so for example, put `<{{after}}`). Alternatively, as part of the beginning of the table issue the command

```
\sisetup{numexp={}}
```

after `\begin{table}`.

¹²The separator for the number of digits before and after the decimal mark may be essentially any non-numeric character. Thus `tabformat=2.4`, `tabformat=2,4` and `tabformat=2a4` all give the same result.

8 Angles

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

10°	12.3°	4.5°	<code>\ang{10}</code>	<code>\ang{12.3}</code>	<code>\ang{4,5}\</code>
$1^\circ 2' 3''$	$0^\circ 0' 1''$		<code>\ang{1;2;3}</code>	<code>\ang{;;1}\</code>	
$+10^\circ$	$-0^\circ 1'$		<code>\ang{+10;;}</code>	<code>\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}</code>	<code>\ang[padding=none]{;;1}\</code>
$2^\circ 2' 0''$		<code>\ang{2;;}</code>	<code>\ang[padding=all]{2;;}\</code>
$0^\circ 3' 0''$	$4^\circ 0' 0''$	$0^\circ 0' 5''$	<code>\ssetup{padding=all}</code> <code>\ang{;3;}</code> <code>\ang{4;;}</code> <code>\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°	$1,05^\circ$	<code>\ang{1.05}</code>	<code>\ang[decimalsign=comma]{1.05}\</code>
3.6789°	3.6789°	<code>\ang{3.6789}</code>	<code>\ang[digitsep=comma]{3.6789}\</code>
$9^\circ 8' 7''$	$9^\circ 8' 7''$	<code>\ang{9;8;7}</code>	<code>\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.¹³

$1^\circ 2' 3.4''$	<code>\ang{1;2;3.4}\</code>
$1^\circ 2' 3'' 4$	<code>\ang[astroang]{1;2;3.4}</code>

9 Units and values

`\SI` The core aim of `si` is correctly typesetting values which have units. The main output macro here is `\SI`, which has the same syntax as the equivalent macro in `sistyle` and `unitsdef`. 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.

¹³The exact positioning of the symbols over the decimal marker is currently something of a guess. Some feedback on the “correct” result would be very welcome.

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

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

9.1 Literal units

Units can be input in two ways, inspired by `sistyle` and `Slunits`. The `sistyle`-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 fullstop (point) symbol and comma are made active, and produce the current contents of the `unitsep` option;
- `~` The contents of the `unitspace` option are typeset by a tilde.

This allows ready input of units.

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

9.2 The unit interpreter

The second operation mode for the `\SI` macro is based on the behaviour of `Slunits`. Here, each unit, SI multiple prefixes 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}\l</code>
1.453 g cm ⁻³	<code>\SI{1.453}{\gram\per\cubic\centi\metre}\l</code>
33.562 cd s	<code>\SI{33.562}{\candela\second}\l</code>
100 m · s ⁻²	<code>\SI[unitsep=cdot]{100}{\metre\per\Square\second}\l</code>
4.56 × 10 ³ m s ⁻¹	<code>\SI[prefix=power]{4.56}{\kilo\metre\per\second}\l</code>

On its own, this is very similar to `Slunits`, and is less convenient than the direct input method.¹⁴ 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}\l</code>
1.453 g/cm ³	<code>\SI[per=slash]{1.453}{\gram\per\cubic\centi\metre}\l</code>
33.562 cd s	<code>\SI{33.562}{\candela\second}\l</code>
100 m/s ²	<code>\SI[per=frac,fraction=nice]{100}{\metre\per\Square\second}\l</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.

¹⁴Users of `Slunits` should note the lack of need for a `\usk`-type macro.

9.3 Powers of units

`\Square` Including powers in units is handled using a “natural language” method. Thus preceding a unit by `\Square` or `\cubic` will raise the unit to the appropriate power, while `\squared` or `\cubed` follow the unit they apply to.¹⁵

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

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

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

`\tothe` For powers not defined above or with `\newpower`, the `\tothe` macro can be used “in line” to produce a power. As follows from standard English usage, this comes after the unit.¹⁶

16.86m^4	<code>\SI{16.86}{\metre\tothe{4}}\</code>
$7.895\text{cd}^{0.5}$	<code>\SI{7.895}{\candela\tothe{0.5}}\</code>
7.895N^{-6}	<code>\SI{7.895}{\newton\tothe{-6}}\</code>
1.34K^{-7}	<code>\SI{1.34}{\per\kelvin\tothe{7}}\</code>

9.4 Units with no values

`\unitsym` For typesetting the symbol for a unit on its own, with the full font control and without extra spaces, the `\unitsym` macro is provided.¹⁷ 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>\unitsym{kg.m/s^2}\</code>
$\text{mol}\cdot\text{dm}^{-3}$	<code>\unitsym[mode=text,unitsep=cdot]{\mole\per\cubic\deci\metre}\</code>

9.5 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. `si` supports both operation modes, with the limitation that units trailing values lose font control of the value.

¹⁵The `\Square` macro is capitalised to avoid a name clash with `pstricks`.

¹⁶Suggestions for a macro name for before the unit for the same job are welcome!

¹⁷The same effect can be achieved using the `\SI` macro with an empty numerical argument.

Table 2: The seven base SI units

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

Table 3: The SI prefixes (load=prefix)

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

123 m
123 K
234 A
6 s

```
123\metre\\
\kelvin[123]\\
\sisetup{mode=text} \ampere[234]\\
6\second
```

When used in this way, the units *do not* take an optional `keyval` argument.

9.6 Pre-defined units, prefixes and powers

The package always defines the seven base SI units, irrespective of any package options given (Table 2). 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*.

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`, `si` also defines the standard prefixes for powers of ten (Table 3). This leads to the redefinition of `\kilogram` as `\kilo\gram`. The macro `\deka` is provided, as this is used as an alias for `\deca` in some places. The package also defines a number of derived SI units which have assigned names and symbols (Table 4). Note that `\Gray` is capitalised to avoid a name clash with the `pstricks` package.¹⁸ In addition to these units, there

¹⁸The macros `\ohm` and `\celsius` are not defined by `si` if the `gensymb` package is loaded.

Table 4: The derived SI units with defined names (`load=derived`)

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

Table 5: Units derived from experiments (`load=physical`)

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

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 5), those considered “accepted” (Table 6), and those accepted temporarily (Table 7).¹⁹

9.7 Prefixed and abbreviated units

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

Table 8: Prefixed (`load=prefixed`) and abbreviated (`load=abbr`) units

Unit	Macro	Symbol	Abbreviation
<i>Masses</i>			
kilogram	<code>\kilogram</code>	kg	<code>\kg</code>
femtogram	<code>\femtogram</code>	fg	<code>\fg</code>
picogram	<code>\picogram</code>	pg	<code>\pg</code>

Continued on next page

¹⁹These are supposed to be replaced over time by SI units.

Unit	Macro	Symbol	Abbreviation
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
<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>			
squaremetre	\squaremetre	m ²	
squarecentimetre	\squarecentimetre	cm ²	
squarekilometre	\squarekilometre	km ²	
<i>Volumes</i>			
millilitre	\millilitre	ml	\ml
microlitre	\microlitre	µl	\micl

Continued on next page

Unit	Macro	Symbol	Abbreviation
centimetrecubed	<code>\centimetrecubed</code>	cm^3	<code>\cmc</code>
	<code>\centimetrecubed</code>	cm^3	<code>\cmc</code>
cubicdecimetre	<code>\cubicdecimetre</code>	dm^3	<code>\dmc</code>
<i>Frequencies</i>			
hertz	<code>\hertz</code>	Hz	<code>\Hz</code>
millihertz	<code>\millihertz</code>	mHz	<code>\mHz</code>
kilohertz	<code>\kilohertz</code>	kHz	<code>\kHz</code>
megahertz	<code>\megahertz</code>	MHz	<code>\MHz</code>
gigahertz	<code>\gigahertz</code>	GHz	<code>\GHz</code>
terahertz	<code>\terahertz</code>	THz	<code>\THz</code>
<i>Potentials</i>			
millivolt	<code>\millivolt</code>	mV	<code>\mV</code>
kilovolt	<code>\kilovolt</code>	kV	<code>\kV</code>
<i>Energies</i>			
kilojoule	<code>\kilojoule</code>	kJ	<code>\kJ</code>
electronvolt	<code>\electronvolt</code>	eV	<code>\eV</code>
millielectronvolt	<code>\millielectronvolt</code>	meV	<code>\meV</code>
kiloelectronvolt	<code>\kiloelectronvolt</code>	keV	<code>\keV</code>
megaelectronvolt	<code>\megaelectronvolt</code>	MeV	<code>\MeV</code>
gigaelectronvolt	<code>\gigaelectronvolt</code>	GeV	<code>\GeV</code>
teraelectronvolt	<code>\teraelectronvolt</code>	TeV	<code>\TeV</code>
<i>Powers</i>			
milliwatt	<code>\milliwatt</code>	mW	
kilowatt	<code>\kilowatt</code>	kW	
megawatt	<code>\megawatt</code>	MW	
<i>Capacitance</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>Resistance</i>			
kiloohm	<code>\kiloohm</code>	k Ω	
megaohm	<code>\megaohm</code>	M Ω	
gigaohm	<code>\gigaohm</code>	G Ω	
millisiemens	<code>\millisiemens</code>	mS	
<i>Forces</i>			
millinewton	<code>\millinewton</code>	mN	

Continued on next page

Unit	Macro	Symbol	Abbreviation
kilonewton	<code>\kilonewton</code>	kN	
<i>Other units</i>			
hectopascal	<code>\hectopascal</code>	hPa	
megabecquerel	<code>\megabecquerel</code>	MBq	
millisievert	<code>\millisievert</code>	mSv	

9.8 Specialist units

`\mmHg`
`\molar`
`\molar`

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. `si` comes with the predefined files `alsoload=chemistry` and `alsoload=hep`. The later defines the units from the `hepunits` package not provided elsewhere here. The former adds the common chemistry units `\mmHg`, `\molar` and `\Molar`. The `\Molar` macro is somewhat awkward, as it can be given as either “M” or “M”. The later is obviously easily confused with the sign for the prefix mega.

`\bit`
`\byte`

The package also comes with equipped for `alsoload=binary`. This provides the binary units and prefixes. The extra units are `\bit` and `\byte`, with the new prefixes listed in Table 9.

9.9 Defining new units

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

New units are produced using the `\newunit` macro. This works as might be expected: `\newunit[⟨options⟩]{⟨unit⟩}{⟨symbol⟩}`, where `⟨symbol⟩` can contain literal values, other units, multiple prefixes, powers and `\per`. The `⟨options⟩` argument can be any suitable options, and applies to this unit only. The most obvious example for using this macro is the `\degree` unit.²⁰ The (first) optional argument to `\SI` and `\unitsym` can be used to override the settings for the unit.

3.1415° `\SI{3.1415}{\degree}\`
12345XXX 67 890 XXX `\newunit[valuesep=none]{\odddunit}{XXX}\`
`\SI{12345}{\odddunit}`
`\SI[valuesep=thick]{67890}{\odddunit}`

Output that is only safe in maths mode should be protected with `\ensuremath`; text-only input requires `\text`. In the example below, `\mathnormal` is used to force the font choice only for the single character.²¹

10m π⁻² `\newunit{\SIpi}{\ensuremath{\mathnormal{\pi}}}`
`\SI{10}{\metre\per\SIpi\squared}`

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

Powers are defined: `\newpower[⟨post⟩]{⟨power⟩}{⟨num⟩}`. `⟨power⟩` is the name of the power macro, an `⟨num⟩` is the (positive) number it represents. The later argument is always processed internally by `\num`, but *must* be a number.

²⁰Although the `\ang` macro is preferred for this job.

²¹The `\mathrm` font used for this document has an “B” at the π position.

Table 6: Units accepted for use with SI (load=accepted)

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

Table 7: Additional (temporary) SI units (load=addn)

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

Table 9: Binary prefixes (alsoload=binary)

Prefix	Macro	Power
kibi	\kibi	2 ¹⁰
mebi	\mebi	2 ²⁰
gibi	\gibi	2 ³⁰
tebi	\tebi	2 ⁴⁰
pebi	\pebi	2 ⁵⁰
exbi	\exbi	2 ⁶⁰

Giving the optional argument `post` indicates to the package that the power will come after the unit it applies to; by default it is assumed that it will come before.

kg^4
 m^4

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

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

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

```
\newprefix{\kilo}{k}{3}
```

10 Font control

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

10 10
20° 20°
30 kg
30 kg

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

50

By giving the `obeyfamily` option, the surrounding font family (serif, sans serif, fixed width) is used for inline materials. Inside the display maths environments, the currently active maths font is used. The `obeybold` option causes the bold setting to be obeyed in the same way.

1°1'1" 1°1'1"
2°2'2" 2°2'2"
3°3'3" 3°3'3"
4°4'4" 4°4'4"

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

5°5'5"

11 Package options

`\sisetup` The “native” options for the package are all given using the `keyval` methods. 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 `true`, so giving the option name alone sets the flag to true.

- `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. The default value is given in bold.
- `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. As with simple choice options, the default is given in bold.
- `option=<literal>` The given value is used as a literal by the package.
- `option=<macro>` These options expect a macro name as a value; the macro name is then used by the package. Note that the name does *not* include the leading `\`.
- `option=<length>` Requires a TeX lengths, for example `0.5ex`.
- `option=<list>` Takes a list of one or more items, which are not determined in advance.

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.

11.1 Font family and style

The font used when typesetting material can be tightly controlled using `si`. A number of options affect how the package matches the surrounding font, and the font families used to achieve this.

- `obeyfamily=<bool>` By default, the font family used for typesetting does *not* match the surroundings. This is altered using the `obeyfamily` switch; when active, serif (Roman), sans serif and typewriter fonts are detected.
- `mode=<choice>` The output of `si` can be typeset using either text or maths fonts. By default, maths mode is used, but this can be altered setting the `mode` option to `text`.
- `textmode=<bool>` A shortcut for `[mode=text]`.
- `obeymode=<bool>` The package can detect and use the surrounding maths or text mode, if requested. Default is `false`.
- `obeybold=<bool>` If the typeset text should obey the local value of the bold setting, then this option should be set: the default is `false`.
- `inlinebold=<choice>` For inline maths, the package can check either the surrounding maths or the surrounding text. The options here are `text` and `maths` (or `math`).
- `obeyitalic=<bool>` Italic *versus* upright shape is handled slightly differently to bold. The option works in text mode, but has no effect in maths mode. This is because font changes plus italic is not possible in maths mode (for example, see the result of `\mathit{\mathrr{10}}`).

- `mathdefault= $\langle macro \rangle$` The default shape used for text printed in maths mode. The default is the value stored in `mathrm`.
- `textdefault= $\langle macro \rangle$` The default shape used for text printed in text mode. The default is the value stored in `mathrm`.
- `mathnumdefault= $\langle macro \rangle$` The default shape used for numbers printed in maths mode. The default is the value stored in `mathrm`.
- `textnumdefault= $\langle macro \rangle$` The default shape used for numbers printed in text mode. The default is the value stored in `mathrm`.
- `mathrm= $\langle macro \rangle$` The font command used in maths mode when the surrounding text is serif. The default is `mathrm`; the other maths font defaults follow the same pattern.
- `mathsf= $\langle macro \rangle$` The font command used in maths mode when the surrounding text is sans serif.
- `mathrm= $\langle macro \rangle$` The font command used in maths mode when the surrounding text is fixed width.
- `textrm= $\langle macro \rangle$` The font command used in text mode when the surrounding text is serif. The default is `rmfamily`; the other text font defaults follow the same pattern.
- `textsf= $\langle macro \rangle$` The font command used in text mode when the surrounding text is sans serif.
- `texttt= $\langle macro \rangle$` The font command used in text mode when the surrounding text is fixed width.

11.2 Spacing and separators

The spacings used between items are all user-definable. This is also true for the separators used for decimals, *etc.*.

- `unitsep= $\langle choice, literal \rangle$` This defines the separation of different unit symbols. The $\langle list \rangle$ takes values **thin**, `medium` (alias `med`), `thick` (all maths spacings), `space` (a full space), `cdot` (a centred dot) and `times`. $\langle literal \rangle$ values are typeset in maths mode.
- `unitspace= $\langle choice, literal \rangle$` The spacing represented by an explicit hard space (~) inside a unit macro. Takes the same list as `valuesep`.
- `valuesep= $\langle choice, literal \rangle$` Defines the separation between a value and the associated unit. Valid $\langle list \rangle$ values are **thin**, `medium` (alias `med`), `thick`, `space` and `none`.
- `digitsep= $\langle choice, literal \rangle$` The separation (if any) between groups of digits in large numbers. Valid $\langle list \rangle$ values are **thin**, `medium` (alias `med`), `thick`, `space`, `comma`, `fullstop` (aliases `stop` and `period`) and `none`.
- `decimalsign= $\langle choice, literal \rangle$` The decimal sign, either `comma` or **fullstop** (also aliased as `stop` and `period`).

- `anglesep= \langle choice, literal \rangle` The separator between degrees, minutes and seconds in an angle. The options are `thin`, `medium` (alias `med`), `thick` and `none`.

11.3 Number formatting

There are two types of option for numbers. The first set are concerned with parsing numbers, and are very similar to the settings in `numprint`. These all begin `num`, and take literal values. Notice that the literals are *not* separated in any way in the input.

- `numlist= \langle literal \rangle` The characters which are numbers: **01234567890**.
- `numdecimal= \langle literal \rangle` Decimal markers: `.`, `,`
- `numexp= \langle literal \rangle` Exponent markers: **e**`d`**E**`D`
- `numgobble= \langle literal \rangle` Characters to be gobbled when processing numbers: no default
- `numsign= \langle literal \rangle` Signs (which must be at the start of a number): **+ \mp** `\pm\mp`
- `numextra= \langle literal \rangle` “Extra” characters, to be carried through directly to the output: **()**

The second type of option for numbers controls the output.

- `addsign= \langle choice \rangle` Sets whether a sign is added to numbers without an explicit sign given. Valid choices are `mantissa` (or `mant`), `exponent` (or `exp`), `both` (or `all`) and `none`. The option will also act as a Boolean, taking `true` and `optfalse`, with `addsign` alone equal to giving the `true` (= `all`) value.
- `sign= \langle choice, literal \rangle` The sign used by the above. Choices are **plus**, `minus`, `pm` and `mp` (\pm and \mp , respectively). The sign will always be typeset in maths mode.
- `sepfour= \langle bool \rangle` When separating out numbers (using `digitsep`), four-digit numbers can be skipped. This is the default.
- `expproduct= \langle choice, literal \rangle` The symbol used to indicate a product for exponents (*i.e.* the \times in 2×10^2). The choices are **times** and `cdot`.
- `exppower= \langle choice, literal \rangle` Slightly esoterically, the power used for exponents can be altered. The “choice” list here only recognises `ten`; anything else is used literally.
- `padnumber= \langle choice \rangle` This sets where zeros are added. The choices are **leading** (a leading zero added to `.1`), `trailing` (converts `1.` to `1.0`), `all` (leading and trailing, also available as `both`) and `none` (no zeros added). The option will also act as a Boolean, taking `true` and `optfalse`, with `padnumber` alone equal to giving the `true` (= `all`) value.

11.4 Angle formatting

The angle formatter uses `\num` to format numbers; any options for numbers are therefore applicable here. When typesetting an angle using `\ang`, the following extra option is also relevant.

- `padangle=choice` Determines whether small and large angles are padded. The choices are `none` (no additional zeros are added), `small` (angles with no degrees have 0° added), `large` (angles with no seconds have $0''$ added) and `all` (small and large combined). The option also recognises `true` and `false` as choices, which are equal to `all` and `none`, respectively. If no value is given, `padangle` acts a Boolean choice.
- `astroang=bool` Astronomers place the signs for angles over decimal signs; this is handled here.

11.5 Tabular material

The formatting of data in `s` columns is controlled by a single package option.

- `tabformat=number` The number here determines how to centre decimal numbers in a column. If `number` is zero or negative, then the decimal marker is placed at the centre of the column with the number symmetrically placed around it. If `number` is positive, it is interpreted `\meta{pre}.\meta{post}`, where `pre` is the number of digits before the decimal marker and `post` is the number after. Appropriate space is reserved to centre a number of total length `pre + post` (plus the decimal marker). If the digits supplied are too long, overfull boxes will result. If only `pre` is given, an equal amount of space is reserved before and after the decimal marker, and the number is typeset flush right.

11.6 Units

The output of units (as opposed to the numerical argument of the unit) takes only a few options.

- `xspace=bool` Determines whether to use `xspace` at the end of unit macros when not given inside `\SI`, for example `10\metre away` will give “10m away” with `xspace` turned on, but “10maway” with it turned off.
- `per=choice` Affects how `\per` is interpreted in units. The options available are **reciprocal** (also available as `rp` and `power`), `slash` and `fraction` (or `frac`).
- `fraction=choice` When using `per=frac`, further control of the appearance of the fraction is provided. The options available are `frac` (uses L^AT_EX `\frac` operation), `nice` (also available as `nicefrac`; uses a nice-`frac`-like system), `ugly` (also `uglyfrac`; the same as loading `nicefrac` with the `ugly` option: uses `\frac` for material in maths mode and a slash for

material in text mode) and `sfrac` (uses the `\sfrac` macro from the `xfrac` package.²²

- `denlbrac=<literal>` and `denrbrac=<literal>` When using `per=slash`, using two or more units in the denominator gives an ambiguous fraction. The package therefore adds `denlbrac` and `denrbrac` in such cases.
- `prefix=<choice>` Controls how prefixes to units are handled, with options **symbol** (or `letter`) and `number` (or `power`).
- `prefixpower=<choice, literal>` and `prefixproduct=<choice>` Works in the same way as the general exponent equivalents, but only for prefix modifiers.

11.7 Symbols

User access to control the symbols used for μ , Ω , \AA , $^\circ$ 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.

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

When `si` is loaded, it can check for the presence of the `textcomp` and `upgreek` packages, to provide better symbols for certain items. To prevent this, use the `redefsymbols=false` option.

The `eV` symbol requires some fine-tuning, and so has two options of its own.

²²`xfrac` is part of the experimental system for \LaTeX . As it requires a number of additional packages to work, `si` 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`.

- `eVcorra= $\langle length \rangle$` The correction applied to the gap between “e” and “V” of the unit. The default is `0.3ex`.
- `eVcorrb= $\langle length \rangle$` The correction applied to the gap between “V” of the unit and whatever follows. The default is `0ex`; a change is needed for example in `\unitsym[per=slash]{\electronvolt\per\metre}`, which gives eV/m by default, but eV/m by setting `eVcorrb=0.7ex`. The value needed will depend on the use of the unit and the font metrics used.

11.8 Package control

These macros alter the overall behaviour of the package.

- `load= $\langle list \rangle$` Sets which additional configuration files are loaded. These all have names of the form `si- $\langle option \rangle$.cfg`, where $\langle option \rangle$ should be given in the load list. The package recognises the `load=default` option, which is expanded to the standard list of loaded files. This is to allow easy addition of one or more files without needing to know the default list.
- `noload= $\langle list \rangle$` Excludes files from the above from being loaded, so that a single file can be omitted without needing to type a long list of those to be used.
- `alsoload= $\langle list \rangle$` Adds an item to the list to be loaded, without needing to specify all of the existing list.
- `log= $\langle choice \rangle$` Sets the amount of information written to the log by `si`. The $\langle list \rangle$ is `none`, `errors`, `normal` and `debug`. The last option is also available as a Boolean, and gives *lots* of information in the log.
- `emulate= $\langle choice \rangle$` Causes `si` to emulate the given package. The $\langle list \rangle$ takes values `SIunits`, `sistyle`, `units` and `unitsdef`. This option can only be used when loading the package.

11.9 Back-compatibility options

As well as the options outlined above, at load time a number of options are available to allow `si` to be used as a direct replacement for other unit-management packages. These are the same options as are available in `SIunits`, `sistyle`, `units`, and `unitsdef`. Using a legacy option will cause the package to load the appropriate emulation code

12 Emulation of other packages

`si` has been designed as a replacement for `SIunits`, `sistyle`, `units` and `unitsdef`. It therefore provides options a hooks to reproduce the functions of all of these packages. In this way, `si` should be usable as a straight replacement for the older packages. All of the user macros of $\langle package \rangle$ are (hopefully) available when using the `emulate= $\langle package \rangle$` .²³ This means for example that the `\num` macro

²³User macros means that they are described in the package documentation; simply not containing an @ does not mean they will have been emulated.

takes an optional star when emulating `sistyle`. However, there are some points that should be remembered. In particular, `si` validates numerical input, meaning that places where a number is expected in the older packages *require* a number when emulated by `si`.

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 `si`. 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 `si`, the `cnumprint` macro could be provided using the following code.

```
-123456 -123456 N/mm2 \newcommand*{\numprint}[2][\SI[obeymode]{#2}{#1}]{\numprint{-123456} \numprint[N/mm^2]{-123456}
```

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

13 Tricks and known issues

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 `AMSTeX` bundle) to ensure text mode.

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.²⁴ The use of the `\mathnormal` font may get around this issue.

```
4ß × 10-7 \num[numextra=\pi]{4\pi e-7}
4π × 10-7 \num[numextra=\pi,mathnumdefault=mathnormal]{4\pi e-7}
```

On the other hand, you may want to use text mode. There, `\ensuremath` is needed.

```
4π × 10-7 \newcommand*{\numpi}{\ensuremath{\pi}}
\num[numextra=\numpi,mode=text]{4\numpi e-7}
```

There are several potential pitfalls in this area; experimentation may well be needed.²⁵

²⁴This depends on your font setup; this document uses T1 encoding, which shows the issue, whereas using OT1 does not.

²⁵Any suggestions for the code that runs this are welcome; the issue is how to deal with active characters in the input while not expanding macros.

14 Reporting a problem

si is quite long and complicated, and works hard to cover all possible eventualities. However, there will be bugs in the code and unexpected interactions with other packages. If you think you have found a bug, please report it. A short test-case demonstrating the problem would be very welcome. The following is a suitable template, and is available as `si-bug.ltx`, by running `si.dtx` through (pdf)L^AT_EX.

```
\listfiles
\documentclass{article}
% Add other packages here.
% Add options need for si package, retain the debug option.
\usepackage[debug]{si}
\begin{document}
This is the bug test-case document for the \textsf{si}
package.\\
Please put your demonstration here, and e-mail to the
package author.
\begin{center}
\texttt{joseph.wright@morningtar2.co.uk}
\end{center}
\end{document}
```

15 Acknowledgements

The package author has learned L^AT_EX tricks from far too many people to thank all of them. However, for this package specific thanks must go to the authors of the existing “unit” packages: Danie Els (`sistyle`), Marcel Heldoorn (`Slunits`), Patrick Happel (`unitsdef`), Axel Reichert (`units`) and Harald Harders (`numprint`). Philip Lehmann, Will Robertson and Heiko Oberdiek deserve much credit for demonstrating L^AT_EX coding best practice. Thanks to the various contributors of ideas for the package: Donald Arseneau, Michele Dondi, Paul Gans, Ben Morrow, Lan Thuy Pham, Stefan Pinnow, Allan Ristow and Patrick Steegstra.

Part III

Correct application of (SI) units

TO DO!

Part IV

Implementation

16 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`, `Slunits`, `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. However, internal package macros may behave differently; if the user has redefined internal macros, then compatibility may be impaired.

The code used here uses \LaTeX rather than \TeX commands where possible.²⁶ 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.

16.1 Setup code

As always, the package starts with identification. A warning is then printed about possible changes.

```
1 \NeedsTeXFormat{LaTeX2e}
2 \ProvidesPackage{si}%
3 [2008/02/20 v.06a A comprehensive (SI) units package]
4 \PackageInfo{si}
5 {This package is experimental. \MessageBreak The interface and
6  functionality is subject to review \MessageBreak and may be changed
7  in later releases}
```

The package requires \eTeX , so the usual test is made.

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

Packages needed for functionality are loaded. `xkeyval` handles the package options, while `amstext` from the \mathcal{AMS} bundle is needed for `\text`. `array` is needed for the new column type for tabular material. `xspace` provides “magic” spacing after macros, if requested. `xkeyval` has to be at least v2.5, as earlier versions do not have the correct macros available. As this will lead to serious errors later, `si` aborts if `xkeyval` is too old.

²⁶This applies to \LaTeX kernel commands only; for example, `ifthenelse` is not used.

```

17 \RequirePackage{xkeyval}
18 \@ifpackagelater{xkeyval}{2005/05/07}
19 {}
20 {\PackageError{si}
21   {xkeyval >= 2.5 required}
22   {si requires the 'xkeyval' package, version 2.5 or
23     later.\MessageBreak The version loaded is:
24     '\@nameuse{ver@xkeyval.sty}'.\MessageBreak
25     This is a fatal error; the package will abort.}%
26   \endinput}
27 \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 28 \newcommand*\si@tempa{}
29 \newcommand*\si@tempb{}
30 \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.

```
31 \newif\ifsi@switch
```

\si@packagecheck As si is intended to replace the other unit-management packages, these are tested for before any further processing. If any are loaded, the package halts compilation; name clashes or unexpected results could occur if this is not tested. Notice that Slunits and sistyle could be loaded with variable capitalisation (at least on Windows); both possibilities are tested. Also notice that unitsdef must be tested before units, so that users of the former get an intelligible message.

\si@tempa

```

32 \newcommand*\si@packagecheck{%
33   \begingroup
34   \@for\si@tempa:=SIunits,siunits,sistyle,SIstyle,unitsdef\do{%
35     \@ifpackageloaded{\si@tempa}
36       {\PackageError{si}
37         {Package '\si@tempa' incompatible}
38         {The '\si@tempa' package and 'si' are
39           incompatible.\MessageBreak Use the
40           'emulate=\si@tempa' package option when loading si.}}
41       {}}

```

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

```

42   \@for\si@tempa:=units\do{%
43     \@ifpackageloaded{\si@tempa}
44       {\PackageWarning{si}
45         {Consider loading the si package with\MessageBreak
46           option 'emulate=\si@tempa', rather than\MessageBreak
47           loading both \si@tempa and si}}
48       {}}
49   \endgroup}

```

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

```

50 \si@packagecheck
51 \AtBeginDocument{\si@packagecheck}

```

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

```
\si@ifdefinable{<macro>}
52 \newcommand*{\si@ifdefinable}[1]{%
53   \expandafter\expandafter\expandafter\@ifundefined%
54   \expandafter\expandafter\expandafter%
55   {\expandafter@gobble\string#1}}
```

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

```
\si@addtolist{<macro>}{<items>}
56 \newcommand*{\si@addtolist}[2]{%
57   \ifx\@empty#1\@empty
58   \edef#1{#2}%
59   \else
60   \edef#1{#1,#2}%
61   \fi}
```

`\si@addtocname` A second item to add to a is macro.

```
\si@temptoks \si@addtomacro{<macro>}{<tokens>}
\si@addtocname{<cname>}{<tokens>}
62 \newtoks{\si@temptoks}
63 \newcommand*{\si@addtocname}[2]{%
64   \@ifundefined{#1}
65   {\@namedef{#1}{#2}}
66   {\si@temptoks\expandafter\expandafter\expandafter{%
67     \cname #1\endcname#2}%
68   \expandafter\edef\cname #1\endcname{\the\si@temptoks}}
```

`\si@xifmtarg` To keep down dependance on other packages, the very short code block from `ifmtarg` is copied here with an internal name.

```
\si@ifnotmtarg
69 \begingroup
70   \catcode`\Q=3
71   \long\gdef\si@xifmtarg#1#2Q#3#4#5\@nil{#4}
72   \long\gdef\si@ifnotmtarg#1{%
73     \si@xifmtarg#1QQ\@firstofone@gobble\@nil}
74 \endgroup
```

16.2 Logging

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

```
\ifsi@logmin 75 \newif\ifsi@debug
\ifsi@lognone 76 \newif\ifsi@logmin
77 \newif\ifsi@lognone
```

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

```
\si@log@err{<error>}{<explanation>}
\si@log@warn{<warning>}
\si@log@inf{<information>}
```

```

78 \newcommand*{\si@log@err}[2]{%
79   \ifsi@lognone\else
80     \ifsi@logmin
81       \PackageWarning{si}{#1}%
82     \else
83       \PackageError{si}{#1}{#2}%
84     \fi
85   \fi}
86 \newcommand*{\si@log@warn}[1]{%
87   \ifsi@lognone\else
88     \ifsi@logmin\else
89       \PackageWarning{si}{#1}%
90     \fi
91   \fi}
92 \newcommand*{\si@log@inf}[1]{%
93   \ifsi@lognone\else
94     \ifsi@logmin\else
95       \PackageInfo{si}{#1}%
96     \fi
97   \fi}

```

`\si@log@debug` The debug macro only gives output if the appropriate package option is set.

```

\si@log@debug{<debug-information>}
98 \newcommand*{\si@log@debug}[1]{%
99   \ifsi@lognone\else
100     \ifsi@debug
101       \PackageInfo{si}{#1}%
102     \fi
103   \fi}

```

16.3 Option handling

`\sisetup` To allow modification of options at run time, a setup macro is provided. The run of strange tests are to prevent problems in arrays and the like.

```

\sisetup{<keyval-options>}
104 \newcommand*{\sisetup}[1]{%
105   \iffalse{\fi\ifnum0='}\fi
106   \setkeys[si]{opt}{#1}%
107   \ifnum0='\fi\iffalse}\fi}

```

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

```

\si@opt@key{<keyname>}{<code>}
108 \newcommand*{\si@opt@key}[2]{%
109   \define@key[si]{opt}{#1}
110   {#2\si@log@debug{Option #1 set to #1}}}

```

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

```

\si@opt@cmdkeys \si@opt@cmdkey[<default>]{<keyname>}{<function>}
\si@opt@cmdkeys[<default>]{<keynames>}
111 \newcommand*{\si@opt@cmdkey}[3][[]]{%

```

```

112 \define@cmdkey[si]{opt}[si@]{#2}[#1]{#3}}
113 \newcommand*\si@opt@cmdkeys[2][]{%
114 \define@cmdkeys[si]{opt}[si@]{#2}[#1]}

```

`\si@opt@boolkey` Keys which only take switch values; anything other than true or false will generate a warning from `xkeyval`. `\si@opt@boolkey[optional-processing]{keyname}`

```

115 \newcommand*\si@opt@boolkey[2][]{%
116 \define@boolkey[si]{opt}[si@]{#2}[true]
117 {#1\si@log@debug{Option #2 set to ##1}}}

```

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

```

\si@opt@choickey[default]{keyname}{choices}{in-list}{not-in-list}
118 \newcommand*\si@opt@choickey[5][]{%
119 \define@choickey**[si]{opt}{#2}[\si@tempa]{#3}[#1]
120 {#4\si@log@debug{Option #2 set to ##1}}
121 {#5\si@log@debug{Option #2 set to ##1}}}

```

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

```

\si@opt@xchoickey{keyname}{choices}{initial}
122 \newcommand*\si@opt@xchoickey[3]{%
123 \si@opt@choickey[#3]{#1}{#2}

```

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.

```

124 {\expandafter\renewcommand\expandafter*%
125 \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.

```

126 {\expandafter\renewcommand\expandafter*%
127 \csname si@#1\endcsname{##1}}

```

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

```

128 \expandafter\newcommand\expandafter*\csname si@#1\endcsname%
129 {\@nameuse{si@fix@#3}}

```

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

```

\si@opt@compatkey{package}{keyname}.
130 \newcommand*\si@opt@compatkey[2]{%
131 \define@boolkey[si]{opt}[si@old@]{#2}[true]
132 {\si@log@debug{Emulating #1 package option\MessageBreak #2}%
133 \sisetup{emulate=#1}%
134 \si@log@debug{Option #2 set to ##1}}
135 \AtEndOfPackage{\si@opt@disablekey{#2}
136 {Compatibility option #2 only\MessageBreak
137 available when loading si package}}}

```

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

```
\si@opt@disablekey{<keyname>}{<warning>}
138 \newcommand*{\si@opt@disablekey}[2]{%
139   \key@ifundefined{si}{opt}{#1}
140   {}
141   {\si@log@debug{Disabling key #1}%
142    \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.

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

`\si@tempa` 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.

`\si@tempb`

```
144 {\si@debugfalse
145  \si@logminfalse
146  \si@lognonfalse
147  \renewcommand*{\si@tempb}{none}%
148  \ifx\si@tempa\si@tempb
149    \si@lognontrue
150  \fi
151  \renewcommand*{\si@tempb}{minimal}%
152  \ifx\si@tempa\si@tempb
153    \si@logmintrue
154  \fi
155  \renewcommand*{\si@tempb}{debug}%
156  \ifx\si@tempa\si@tempb
157    \si@debugtrue
158  \fi
159  \renewcommand*{\si@tempb}{verbose}%
160  \ifx\si@tempa\si@tempb
161    \si@debugtrue
162  \fi}
```

The option has not been recognised: give a warning (if appropriate).

```
163 {\si@log@warn{Unrecognised value '#1' for option log}}
```

A quick method to set `log=debug`.

```
164 \si@opt@boolkey{debug}
```

`\si@emulate` The `emulate` option is used for back-compatibility mode; if the keyword is given with no value, emulation of `SIunits` is assumed.

```
165 \newcommand*{\si@emulate}{}
166 \si@opt@choicekey[SIunits]{emulate}
167 {SIunits,sistyle,numprint,units,unitsdef}
168 {\si@log@debug{Found emulation request for #1 package}%
169  \si@addtolist{\si@emulate}{#1}}
170 {\si@log@warn{Unknown value '#1' for option emulate
171  \MessageBreak No emulation will occur}}
```

The `emulate` option is no longer valid once the package has been loaded.

```
172 \AtEndOfPackage{%
```



```

173 \si@opt@disablekey{emulate}
174   {emulate option only available when\MessageBreak
175     loading package}}

```

`\si@unitsep` The two . . . space 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`²⁷ and `none`;²⁸ other entries will be treated as custom spaces.
`\si@valuesep`

```

176 \si@opt@xchoicekey{unitsep}{thin,med,medium,thick,space,none,cdot,
177   times}{thin}
178 \si@opt@xchoicekey{unitspace}{space,thin,med,medium,thick,none}
179   {thin}
180 \si@opt@xchoicekey{valuesep}{space,thin,med,medium,thick,none}
181   {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.

```

182 \si@opt@xchoicekey{digitsep}
183   {thin,med,medium,thick,none,comma,stop,fullstop,period}{thin}

```

`\si@decimalsign` 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.

```

184 \si@opt@xchoicekey{decimalsign}{comma,stop,fullstop,period,cdot}
185   {fullstop}

```

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

```

186 \si@opt@xchoicekey{anglesep}{thin,med,medium,thick,none}{none}

```

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

```

187 \si@opt@boolkey{obeymode}

```

`\ifsi@textmode` The output of the package can be typeset using either text or maths mode fonts. This is controlled using the `mode` option and the `\ifsi@textmode` switch.

```

188 \newif\ifsi@textmode
189 \si@opt@boolkey{textmode}
190 \si@opt@choicekey{mode}{math,maths,text}

```

`\si@tempa` `\si@tempb` used to for the expansion tests. The default is `none`, as the choice
`\si@tempb` key will not allow other values to get here.

```

191   {\si@textmodefalse
192     \renewcommand*{\si@tempb}{text}%
193     \ifx\si@tempa\si@tempb
194       \si@textmodetrue
195     \fi}
196   {\si@log@warn{Unknown value '#1' for option mode}}

```

²⁷only valid for `unitsep`.

²⁸Only valid for `valuesep`.

`\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.

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

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

```
198 \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`).

```
\si@tempa
\si@tempb 199 \newif\ifsi@inlinebtext
200 \si@opt@choicekey{inlinebold}{text,maths,math}
201   {\si@inlinebtextfalse
202    \renewcommand*{\si@tempb}{text}%
203    \ifx\si@tempa\si@tempb
204      \si@inlinebtexttrue
205    \fi}
206   {\si@log@warn{Unknown value '#1' for option inlinebold}}
```

`\ifsi@obeyitalic` Italic is slightly different to bold, as there is no convenient switch for maths.

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

`\si@mathsdefault` 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@mathsrm
\si@mathssf
\si@mathstt 208 \si@opt@cmdkeys{mathsdefault,mathsrm,mathssf,mathstt}
```

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

```
209 \si@opt@key{mathdefault}{\sisetup{mathsdefault=#1}}
210 \si@opt@key{mathrm}{\sisetup{mathsrm=#1}}
211 \si@opt@key{mathsf}{\sisetup{mathssf=#1}}
212 \si@opt@key{mathtt}{\sisetup{mathstt=#1}}
```

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

```
\si@textrm
\si@textsf 213 \si@opt@cmdkeys{textdefault, textrm, textsf, texttt}
\si@texttt
```

`\si@mathnumdefault` To allow numbers to be set in a different font to text, additional options are set up.

```
\si@textnumdefault
214 \si@opt@cmdkeys{mathnumdefault, textnumdefault}
215 \si@opt@key{mathsnumdefault}{\sisetup{mathnumdefault=#1}}
```

`\si@numlist` 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@numgobble 216 \si@opt@cmdkeys{numlist, numdecimal, numexp, numgobble, numsign, numextra}
\si@numsign
\si@numvalid
```

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.

```
217 \newcommand*{\si@numvalid}{\si@numgobble\si@numexp\si@numsign%
218 \si@numdecimal\si@numlist\si@numextra}
```

`\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`.

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

`\si@expproduct` The marker for multiplication in exponential numbers is set up.

```
220 \si@opt@xchoicekey{expproduct}{times,cdot}{times}
```

`\si@exppower` In the same area, the power for exponents is variable. Only one choice is given.

```
221 \si@opt@xchoicekey{exppower}{ten}{ten}
```

`\si@prefixproduct` The marker for multiplication in prefixes.

```
222 \si@opt@xchoicekey{prefixproduct}{times,cdot,none}{times}
```

`\si@prefixpower` In the same area, the power for prefixes is variable. Here, two choices are needed.

```
223 \si@opt@xchoicekey{prefixpower}{ten,two}{ten}
```

`\ifsi@prefixnum` Unit prefixes can be given as either symbols or numerically.

```
224 \newif\ifsi@prefixnum
225 \si@opt@choicekey{prefix}{symbol,letter,power,number}
```

`\si@tempa` `\si@tempb` used to for the expansion tests. The default is none, as the choice key will not allow other values to get here.

```
226 {\si@prefixnumfalse
227 \renewcommand*{\si@tempb}{power}%
228 \ifx\si@tempa\si@tempb
229 \si@prefixnumtrue
230 \fi
231 \renewcommand*{\si@tempb}{number}%
232 \ifx\si@tempa\si@tempb
233 \si@prefixnumtrue
234 \fi}
235 {\si@log@warn{Unknown value '#1' for option prefix}}
```

`\ifsi@num@padlead` A setting is needed to indicate when to add zeros to decimal numbers, either before the decimal marker (.1 giving "0.1") or after (1. giving "1.0").

```
236 \newif\ifsi@num@padlead
237 \newif\ifsi@num@padtrail
238 \si@opt@choicekey[all]{padnumber}
239 {leading,lead,trailing,trail,all,both,true,none,false}
```

`\si@tempa` `\si@tempb` is used to for the expansion tests. The default is none, as the choice key will not allow other values to get here.

```
240 {\si@num@padleadfalse
241 \si@num@padtrailfalse
242 \renewcommand*{\si@tempb}{leading}%
243 \ifx\si@tempa\si@tempb
244 \si@num@padleadtrue
245 \fi
246 \renewcommand*{\si@tempb}{lead}%
247 \ifx\si@tempa\si@tempb
248 \si@num@padleadtrue
249 \fi}
```

```

250 \renewcommand*{\si@tempb}{trailing}%
251 \ifx\si@tempa\si@tempb
252   \si@num@padtrailtrue
253 \fi
254 \renewcommand*{\si@tempb}{trail}%
255 \ifx\si@tempa\si@tempb
256   \si@num@padtrailtrue
257 \fi
258 \renewcommand*{\si@tempb}{all}%
259 \ifx\si@tempa\si@tempb
260   \si@num@padleadtrue
261   \si@num@padtrailtrue
262 \fi
263 \renewcommand*{\si@tempb}{true}%
264 \ifx\si@tempa\si@tempb
265   \si@num@padleadtrue
266   \si@num@padtrailtrue
267 \fi
268 \renewcommand*{\si@tempb}{both}%
269 \ifx\si@tempa\si@tempb
270   \si@num@padleadtrue
271   \si@num@padtrailtrue
272 \fi}
273 {\si@log@warn{Unknown value '#1' for option padnumber}}

```

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

```

\ifsi@num@signmant 274 \newif\ifsi@num@signmant
\ifsi@num@signexp 275 \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.

```

276 \si@opt@xchoickey{sign}{plus,minus,pm,mp}{plus}
277 \si@opt@choickey[all]{addsign}
278 {mantissa,exponent,mant,exp,all,both,true,none,false}

```

`\si@tempa` **The option is now processed.**

```

\si@tempb 279 {\si@num@signmantfalse
280   \si@num@signexpfalse
281   \renewcommand*{\si@tempb}{mantissa}%
282   \ifx\si@tempa\si@tempb
283     \si@num@signmanttrue
284   \fi
285   \renewcommand*{\si@tempb}{mant}%
286   \ifx\si@tempa\si@tempb
287     \si@num@signmanttrue
288   \fi
289   \renewcommand*{\si@tempb}{exponent}%
290   \ifx\si@tempa\si@tempb
291     \si@num@signexptrue
292   \fi
293   \renewcommand*{\si@tempb}{exp}%
294   \ifx\si@tempa\si@tempb
295     \si@num@signexptrue
296   \fi

```

```

297 \renewcommand*{\si@tempb}{all}%
298 \ifx\si@tempa\si@tempb
299   \si@num@signmanttrue
300   \si@num@signexptrue
301 \fi
302 \renewcommand*{\si@tempb}{true}%
303 \ifx\si@tempa\si@tempb
304   \si@num@signmanttrue
305   \si@num@signexptrue
306 \fi
307 \renewcommand*{\si@tempb}{both}%
308 \ifx\si@tempa\si@tempb
309   \si@num@signmanttrue
310   \si@num@signexptrue
311 \fi}
312 {\si@log@warn{Unknown value '#1' for option addsign}}

```

`\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.

```

313 \newif\ifsi@ang@padsmall
314 \newif\ifsi@ang@padlarge
315 \si@opt@choicekey[all]{padangle}
316 {small,large,all,both,true,none,false}

```

`\si@tempa` `\si@tempb` is used to for the expansion tests. The default is none, as the choice
`\si@tempb` key will not allow other values to get here.

```

317 {\si@ang@padsmallfalse
318  \si@ang@padlargefalse
319  \renewcommand*{\si@tempb}{small}%
320  \ifx\si@tempa\si@tempb
321    \si@ang@padsmalltrue
322  \fi
323  \renewcommand*{\si@tempb}{large}%
324  \ifx\si@tempa\si@tempb
325    \si@ang@padlargetrue
326  \fi
327  \renewcommand*{\si@tempb}{all}%
328  \ifx\si@tempa\si@tempb
329    \si@ang@padsmalltrue
330    \si@ang@padlargetrue
331  \fi
332  \renewcommand*{\si@tempb}{true}%
333  \ifx\si@tempa\si@tempb
334    \si@ang@padsmalltrue
335    \si@ang@padlargetrue
336  \fi
337  \renewcommand*{\si@tempb}{both}%
338  \ifx\si@tempa\si@tempb
339    \si@ang@padsmalltrue
340    \si@ang@padlargetrue
341  \fi}
342 {\si@log@warn{Unknown value '#1' for option padangle}}

```

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

```

343 \si@opt@boolkey{astroang}

\si@tabformat The formatting of numbers in tables is handled by a dcolumn-like system. For
that, a single option is needed to control the centring of data in the table.
344 \si@opt@cmdkey[-1]{tabformat}{}

\ifsi@xspace Unit macros on their own may need xpspace.
345 \si@opt@boolkey{xspace}

\ifsi@frac The option processing for formatting units with \per in them needs two switches.
\ifsi@slash 346 \newif\ifsi@slash
347 \newif\ifsi@frac
348 \si@opt@choickey[reciprocal]{per}
349 {reciprocal, rp, power, slash, frac, fraction}

\si@tempa The usual value testing, with a default to use reciprocal powers.
\si@tempb 350 {\si@slashfalse
351 \si@fracfalse
352 \renewcommand*{\si@tempb}{slash}%
353 \ifx\si@tempa\si@tempb
354 \si@fractrue
355 \si@slashtrue
356 \let\si@frac\si@frc@slash
357 \fi
358 \renewcommand*{\si@tempb}{frac}%
359 \ifx\si@tempa\si@tempb
360 \si@fractrue
361 \fi
362 \renewcommand*{\si@tempb}{fraction}%
363 \ifx\si@tempa\si@tempb
364 \si@fractrue
365 \fi}
366 {\si@log@warn{Unknown value '#1' for option per}}

\si@slash For the slash option, the separator can be customised.
367 \si@opt@xchoickey{slash}{slash}{slash}

\si@denrbrac Macros for the right and left brackets added to potentially-ambiguous denomina-
\si@denlbrac tors.
368 \si@opt@cmdkeys{denrbrac, denlbrac}

\si@tempa In the case of fractional handling of the \per operator, further refinement is
\si@tempb available.
369 \si@opt@choickey[frac]{fraction}
370 {frac, nicefrac, nice, sfrac, xfrac, uglyfrac, ugly}
371 {\let\si@frac\si@frc@frac
372 \renewcommand*{\si@tempb}{nicefrac}%
373 \ifx\si@tempa\si@tempb
374 \let\si@frac\si@frc@nice
375 \fi
376 \renewcommand*{\si@tempb}{uglyfrac}%
377 \ifx\si@tempa\si@tempb
378 \let\si@frac\si@frc@ugly

```

```

379 \fi
380 \renewcommand*{\si@tempb}{nice}%
381 \ifx\si@tempa\si@tempb
382 \let\si@frac\si@frc@nice
383 \fi
384 \renewcommand*{\si@tempb}{sfrac}%
385 \ifx\si@tempa\si@tempb
386 \let\si@frac\si@frc@sfrac
387 \fi
388 \renewcommand*{\si@tempb}{xfraction}%
389 \ifx\si@tempa\si@tempb
390 \let\si@frac\si@frc@sfrac
391 \fi
392 \renewcommand*{\si@tempb}{ugly}%
393 \ifx\si@tempa\si@tempb
394 \let\si@frac\si@frc@ugly
395 \fi
396 {\si@log@warn{Unknown value '#1' for option fraction}}

```

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

```

397 \si@opt@cmdkeys{load,noload}
398 \si@opt@key{alsoload}{\si@addtolist{\si@load}{#1}}
399 \AtEndOfPackage{%
400 \si@opt@disablekey{load}
401 {Configuration files can only be used\MessageBreak
402 when loading si}
403 \si@opt@disablekey{also}
404 {Configuration files can only be used\MessageBreak
405 when loading si}
406 \si@opt@disablekey{noload}
407 {Configuration files can only be used\MessageBreak
408 when loading si}}

```

`\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.

```

409 \si@opt@cmdkeys{textOmega,mathsOmega}
410 \si@opt@key{mathOmega}{\sisetup{mathsOmega=#1}}
411 \newcommand*{\si@mathsOmega}{\text{\ensuremath{\Omega}}}
412 \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.

```

413 \si@opt@cmdkeys{textmu,mathsmu}
414 \si@opt@key{mathmu}{\sisetup{mathsmu=#1}}
415 \DeclareFontEncoding{TS1}{}{}
416 \DeclareFontSubstitution{TS1}{cmr}{m}{n}
417 \DeclareTextSymbol{\si@textmu}{TS1}{181}
418 \DeclareTextSymbolDefault{\si@textmu}{TS1}
419 \DeclareFontFamily{OML}{eur}{\skewchar\font127}
420 \DeclareFontShape{OML}{eur}{m}{n}%
421 {<5> <6> <7> <8> <9> gen * eurm %

```

```

422 <10><10.95><12><14.4><17.28><20.74><24.88>eurm10}{ }
423 \DeclareSymbolFont{greek}{OML}{eur}{m}{n}
424 \DeclareMathSymbol{\si@mathsmu}{\mathord}{greek}{"16}

```

`\si@textdegree` **The angle signs.**

```

\si@mathsdegree 425 \si@opt@cmdkeys{textdegree,mathsdegree,textminute,mathsminute,
\si@textminute 426 textsecond,mathssecond}
\si@mathsminute 427 \si@opt@key{mathdegree}{\sisetup{mathsdegree=#1}}
\si@textsecond 428 \si@opt@key{mathminute}{\sisetup{mathsminute=#1}}
\si@mathssecond 429 \si@opt@key{mathsecond}{\sisetup{mathssecond=#1}}
430 \newcommand*{\si@textdegree}{\ensuremath{{}^{\circ}}}
431 \newcommand*{\si@mathsdegree}{{}^{\circ}}
432 \newcommand*{\si@textminute}{\ensuremath{{}^{\prime}}}
433 \newcommand*{\si@mathsminute}{{}^{\prime}}
434 \newcommand*{\si@textsecond}{\ensuremath{{}^{\prime\prime}}}
435 \newcommand*{\si@mathssecond}{{}^{\prime\prime}}

```

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

```

\si@mathscelsius 436 \si@opt@cmdkeys{textcelsius,mathscelsius}
437 \si@opt@key{mathcelsius}{\sisetup{mathscelsius=#1}}
438 \newcommand*{\si@textcelsius}{\si@textdegree C}
439 \newcommand*{\si@mathscelsius}{\si@mathsdegree\mathrm{C}}

```

`\si@textringA` **The Å sign.**

```

\si@mathsringA 440 \si@opt@cmdkeys{textringA,mathsringA}
441 \si@opt@key{mathringA}{\sisetup{mathsringA=#1}}
442 \newcommand*{\si@textringA}{\AA}
443 \newcommand*{\si@mathsringA}{\text{\AA}}

```

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

```

444 \si@opt@boolkey{redefsymbols}
445 \AtBeginDocument{%
446 \si@opt@disablekey{redefsymbols}
447 {Symbols can only be redefined\MessageBreak
448 when loading si}}

```

`\si@eVcorra`

```

\si@eVcorrb 449 \newlength\si@eVcorra
450 \newlength\si@eVcorrb
451 \si@opt@key{eVcorra}{\setlength\si@eVcorra{#1}}
452 \si@opt@key{eVcorrb}{\setlength\si@eVcorrb{#1}}

```

`\si@locale` **Handling typographic conventions needs three keys. `locale` is used to set the**

`\si@loadlocales` **locale, whereas `\loadlocales` reads in the definitions at package load time.**

```

\si@loctolang 453 \si@opt@cmdkeys{loadlocales,loctolang}
454 \si@opt@cmdkey{locale}{%
455 \sisetup{loadlocales=#1}}%
456 \AtEndOfPackage{\si@loc@set{#1}}
457 \AtBeginDocument{%
458 \si@opt@disablekey{loadlocales}
459 {Locale files can only be loaded\MessageBreak
460 in the preamble}
461 \si@opt@disablekey{loctolang}

```



```

462     {Locale files can only be loaded\MessageBreak
463       in the preamble}
464   \si@opt@cmdkey{locale}{\si@loc@set{#1}}

```

16.4 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`

```

465 \si@opt@compatkey{units}{ugly}
466 \si@opt@compatkey{units}{nice}
467 \si@opt@compatkey{units}{loose}
468 \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 469 \si@opt@compatkey{unitsdef}{OHM}
\ifsi@gensymb 470 \si@opt@compatkey{unitsdef}{ohm}
471 \si@opt@compatkey{unitsdef}{redef-gensymb}
472 \newif\ifsi@gensymb

```

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

```

\ifsi@old@liter 473 \si@opt@compatkey{unitsdef}{LITER}
\ifsi@old@noxspace 474 \si@opt@compatkey{unitsdef}{liter}
\ifsi@old@noconfig 475 \si@opt@compatkey{unitsdef}{noxspace}
476 \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 477 \si@opt@compatkey{unitsdef}{noabbr}
\ifsi@old@nomolabbr 478 \si@opt@compatkey{unitsdef}{noampereageabbr}
\ifsi@old@novoltageabbr 479 \si@opt@compatkey{unitsdef}{nofrequncyabbr}
\ifsi@old@novolumeabbr 480 \si@opt@compatkey{unitsdef}{nomolabbr}
\ifsi@old@noweightabbr 481 \si@opt@compatkey{unitsdef}{novoltageabbr}
\ifsi@old@noenergyabbr 482 \si@opt@compatkey{unitsdef}{novolumeabbr}
\ifsi@old@nolengthabbr 483 \si@opt@compatkey{unitsdef}{noweightabbr}
\ifsi@old@notimeabbr 484 \si@opt@compatkey{unitsdef}{noenergyabbr}
485 \si@opt@compatkey{unitsdef}{nolengthabbr}
486 \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 487 \si@opt@compatkey{SIunits}{cdot}
\ifsi@old@mediumspace 488 \si@opt@compatkey{SIunits}{thickspace}
\ifsi@old@thinspace 489 \si@opt@compatkey{SIunits}{mediumspace}
\ifsi@old@thickqspace 490 \si@opt@compatkey{SIunits}{thinspace}
\ifsi@old@mediumqspace 491 \si@opt@compatkey{SIunits}{thickqspace}
\ifsi@old@thingqspace 492 \si@opt@compatkey{SIunits}{mediumqspace}
493 \si@opt@compatkey{SIunits}{thingqspace}

```

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

```

\ifsi@old@square 494 \si@opt@compatkey{SIunits}{amssymb}
\ifsi@old@pstricks
\ifsi@old@Gray
\ifsi@old@italian

```

```

495 \si@opt@compatkey{SIunits}{squaresn}
496 \si@opt@compatkey{SIunits}{pstricks}
497 \si@opt@compatkey{SIunits}{Gray}
498 \si@opt@compatkey{SIunits}{italian}

```

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

```

\ifsi@old@binary 499 \si@opt@compatkey{SIunits}{textstyle}
\ifsi@old@noams 500 \si@opt@compatkey{SIunits}{binary}
\ifsi@old@derivedinbase 501 \si@opt@compatkey{SIunits}{noams}
\ifsi@old@derived 502 \si@opt@compatkey{SIunits}{derivedinbase}
503 \si@opt@compatkey{SIunits}{derived}

```

16.5 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 used as a space description, two macros are needed for the same output.

```

\si@fix@med
\si@fix@medium 504 \newcommand*{\si@fix@thin}{\,}
\si@fix@thick 505 \newcommand*{\si@fix@med}{\;}
\si@fix@space 506 \newcommand*{\si@fix@medium}{\;}
507 \newcommand*{\si@fix@thick}{\;}
508 \newcommand*{\si@fix@space}{\text{~}}

```

`\si@fix@cdot` Next there are macros for material that is not simply whitespace. To allow several options, the full-stop gets lots of names.

```

\si@fix@comma
\si@fix@stop 509 \newcommand*{\si@fix@cdot}{\cdot}
\si@fix@fullstop 510 \newcommand*{\si@fix@comma}{,}
\si@fix@period 511 \newcommand*{\si@fix@stop}{.}
\si@fix@times 512 \newcommand*{\si@fix@fullstop}{.}
513 \newcommand*{\si@fix@period}{.}
514 \newcommand*{\si@fix@times}{\times}

```

`\si@fix@plus` Signs for numbers are needed.

```

\si@fix@minus 515 \newcommand*{\si@fix@plus}{+}
\si@fix@pm 516 \newcommand*{\si@fix@minus}{-}
\si@fix@mp 517 \newcommand*{\si@fix@pm}{\pm}
518 \newcommand*{\si@fix@mp}{\mp}

```

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

```

\si@fix@ten 519 \newcommand*{\si@fix@two}{2}
520 \newcommand*{\si@fix@ten}{10}

```

`\si@fix@slash` Another optional component that will probably not be used by many people.

```

521 \newcommand*{\si@fix@slash}{/}

```

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

```

522 \newcommand*{\si@fix@none}{}

```

16.6 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.

```
523 \newcommand*{\si@symbol}[1]{%
524   \expandafter\DeclareRobustCommand\expandafter*\expandafter{%
525     \csname si@sym@#1\endcsname}{%
526     \ifmmode
527       \expandafter\csname si@maths#1\expandafter\endcsname%
528     \else
529       \expandafter\csname si@text#1\expandafter\endcsname%
530     \fi}}
```

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

```
\si@sym@ringA 531 \si@symbol{Omega}
  \si@sym@mu 532 \si@symbol{ringA}
\si@sym@degree 533 \si@symbol{mu}
\si@sym@minute 534 \si@symbol{degree}
\si@sym@second 535 \si@symbol{minute}
\si@sym@celsius 536 \si@symbol{second}
537 \si@symbol{celsius}
```

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

```
538 \AtBeginDocument{%
539   \ifsi@redefsymbols
540     \ifpackageloaded{textcomp}
541       {\si@log@debug{Redefining symbols using textcomp}%
542       \renewcommand*{\si@textdegree}{\textdegree}%
543       \renewcommand*{\si@mathsdegree}{\text{\textdegree}}%
544       \mathptmx will give issues with textcomp and the Ω sign.
```

```
544     \ifpackageloaded{mathptmx}{ }
545     {\renewcommand*{\si@textmu}{\textmu}%
546     \renewcommand*{\si@textOmega}{\textohm}}%
```

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

```
547     \long\def\si@tempa{OT1}%
548     \ifx\si@tempa\encodingdefault
549       \renewcommand*{\si@mathsringA}{\text{\capitalring{A}}}%
550       \renewcommand*{\si@textringA}{\capitalring{A}}
551     \fi}{ }
552   \ifpackageloaded{upgreek}
553     {\si@log@debug{Redefining symbols using upgreek}%
554     \renewcommand*{\si@mathsmu}{\upmu}%
555     \renewcommand*{\si@mathsOmega}{\Upomega}}{ }
556   \fi}
```

16.7 Handling fractions

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

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

```

557 \newcommand*\si@frc@frac}[2]{%
558   \ensuremath{\frac{\expandafter\si@unt@out\expandafter{#1}}%
559     {\expandafter\si@unt@out\expandafter{#2}}}}
560 \let\si@frac\si@frc@frac
561 \newcommand*\si@frc@slash}[2]{%
562   \expandafter\si@unt@out\expandafter{#1}%
563   \si@out@text{\ensuremath{\si@slash}}%
564   \expandafter\si@unt@out\expandafter{#2}}
565 \newcommand*\si@frc@nice}[2]{%
566   \ensuremath{\si@frc@nicefrac{\expandafter\si@unt@out%
567     \expandafter{#1}}{\expandafter\si@unt@out\expandafter{#2}}}}
568 \newcommand*\si@frc@sfrac}[2]{%
569   \sfrac{\expandafter\si@unt@out\expandafter{#1}}%
570     {\expandafter\si@unt@out\expandafter{#2}}}
571 \AtBeginDocument{%
572   \@ifpackageloaded{xfrac}
573     {}
574     {\si@log@inf{xfrac package unavailable\MessageBreak
575       using 'fraction=sfrac' will fall back on\MessageBreak
576       nicefrac-like method}%
577     \renewcommand*\si@frc@sfrac}[2]{%
578     \si@log@warn{xfrac package unavailable}%
579     \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`.²⁹

```

\si@frc@displen
\si@frc@textlen 580 \newlength\si@frc@displen
\si@frc@suplen 581 \newlength\si@frc@textlen
\si@frc@ssuplen 582 \newlength\si@frc@suplen
583 \newlength\si@frc@ssuplen
584 \newcommand*\si@frc@nicefrac}{%
585   \ifmmode
586     \expandafter\si@frc@mathsnf%
587   \else
588     \expandafter\si@frc@textnf%
589   \fi}

```

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

```

\si@frc@mathsnf{<numerator>}{<denominator>}
590 \newcommand*\si@frc@mathsnf}[2]{%
591   \begingroup
592     \settoheight{\si@frc@displen}{\ensuremath{\displaystyle{M}}}%
593     \settoheight{\si@frc@textlen}{\ensuremath{\textstyle{M}}}%
594     \settoheight{\si@frc@suplen}{\ensuremath{\scriptstyle{M}}}%
595     \settoheight{\si@frc@ssuplen}{\ensuremath{\scriptscriptstyle{M}}}%
596     \addtolength{\si@frc@displen}{-\si@frc@ssuplen}%
597     \addtolength{\si@frc@textlen}{-\si@frc@ssuplen}%
598     \addtolength{\si@frc@suplen}{-\si@frc@ssuplen}%
599     \mathchoice
600       {\raisebox{\si@frc@displen}{\ensuremath{\scriptstyle{#1}}}}%
601       {\raisebox{\si@frc@textlen}{\ensuremath{\scriptstyle{#1}}}}%

```

²⁹The original is licensed under the GPL; thanks to the author Axel Reichert for permission to copy the code here.

```

602     {\raisebox{\si@frc@suplen}%
603      {\ensuremath{\scriptscriptstyle{#1}}}}%
604     {\raisebox{\si@frc@ssuplen}%
605      {\ensuremath{\scriptscriptstyle{#1}}}}%
606     \mkern-2mu/\mkern-1mu%
607     \bgroup
608     \mathchoice
609       {\scriptstyle}%
610       {\scriptstyle}%
611       {\scriptscriptstyle}%
612       {\scriptscriptstyle}%
613     {#2}%
614     \egroup
615     \endgroup}

```

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

```

\si@frc@textnf{<numerator>}{<denominator>}
616 \newcommand*\si@frc@textnf[2]{%
617   \begingroup
618     \settoheight{\si@frc@textlen}{M}%
619     \settoheight{\si@frc@ssuplen}{\fontsize\sf@size\z@}%
620     \selectfont{M}%
621     \addtolength{\si@frc@textlen}{-\si@frc@ssuplen}%
622     \raisebox{\si@frc@textlen}{\fontsize\sf@size\z@}%
623     \selectfont{#1}}%
624     \hspace{-0.25ex}/\hspace{-0.25ex}%
625     \hbox{\fontsize\sf@size\z@\selectfont{#2}}%
626   \endgroup}

```

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

```

\si@tempa \si@frc@ugly{<numerator>}
\si@tempb
627 \newcommand*\si@frc@ugly[1]{%
628   \def\si@tempa{#1}%
629   \ifmmode
630     \expandafter\si@frc@frac%
631   \else
632     \def\si@tempb{1}%
633     \ifx\si@tempa\si@tempb

```

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

```

634     \setbox\si@tempbox=\hbox{\ensuremath{\si@valuesep}}%
635     \hskip-\wd\si@tempbox%
636     \def\si@tempa{}%
637     \fi
638     \expandafter\si@frc@slash%
639     \fi
640     {\si@tempa}}

```

16.8 Font control

A number of controls and tests are needed to control the font used for output. Underlying all of this is the $\mathcal{A}\mathcal{M}\mathcal{S}$ package `amstext` package, providing the `\text`

command. Much of the font control system here is taken more or less verbatim from `sistyle`; modifications have been made to fit the `si` interface.

`\si@fam@getmfam` The font families in use in the document are needed.

```

\si@fam@sf 641 \newcommand*{\si@fam@getmfam}{%
\si@fam@tt 642   \sbox{0}{${%
643     \@ifundefined{mathsf}
644       {\si@log@debug{No mathsf family found}}%
645       \global\chardef\si@fam@sf=99}%
646     {\mathsf{\global\chardef\si@fam@sf=\fam}}%
647     \@ifundefined{mathtt}
648       {\si@log@debug{No mathtt family found}}%
649       \global\chardef\si@fam@tt=99}%
650     {\mathtt{\global\chardef\si@fam@tt=\fam}}%
651     $}}
652 \AtBeginDocument{\si@fam@getmfam}

```

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

```

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

653 \newcommand*{\si@fam@ifbtext}[1]{%
654   \if b\expandafter\@car\f@series\@nil
655     #1\fi}
656 \newcommand{\si@fam@ifbmaths}[1]{%
657   \def\si@tempa{bold}%
658   \ifx\math@version\si@tempa
659     #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.

```

660 \newcommand*{\si@fam@ifbinline}{%
661   \ifsi@inlinebtext
662     \expandafter\si@fam@ifbtext%
663   \else
664     \expandafter\si@fam@ifbmaths%
665   \fi}

```

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

```

\si@tempa \si@fam@ifitext{\code}

666 \newcommand*{\si@fam@ifitext}[1]{%
667   \if n\expandafter\@car\f@series\@nil\else
668     #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.

```

669 \newcommand*{\si@fam@mode}{%
670   \ifsi@obeymode
671     \ifmmode
672       \ssetup{mode=maths}%
673     \else

```

```

674     \sisetup{mode=text}%
675     \fi
676     \fi}

```

`\ifsi@fam@set` A marker is set up to check if font-matching has been taken place.

```
677 \newif\ifsi@fam@set
```

`\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.

```
678 \newcommand*{\si@fam@set}{%
679   \si@fam@settrue

```

`\si@tempa` The temporary macros are needed for the `\ifx` tests.

```

\si@tempb 680 \edef\si@tempa{\sfdefault}%
681 \edef\si@tempb{\ttdefault}%

```

The surrounding font family is only tested if matching is requested.

```

682 \ifsi@obeyfamily
683   \si@log@debug{Font detection: checking font}%

```

`\si@fam@maths` Next, checks are needed for maths *versus* text mode, and if in maths mode, whether this is inline or display. Once that is done, font families can be tested.

```

\si@fam@text
684   \ifmmode
685     \ifdim\displaywidth>0pt\relax
686       \si@log@debug{Font detection: display maths}%
687       \ifnum\the\fam=\si@fam@sf
688         \si@log@debug{Font detection: sf}%
689         \expandafter\let\expandafter\si@fam@maths
690         \csname\si@mathssf\endcsname
691         \expandafter\let\expandafter\si@fam@text
692         \csname\si@textsf\endcsname
693       \else
694         \ifnum\the\fam=\si@fam@tt
695           \si@log@debug{Font detection: tt}%
696           \expandafter\let\expandafter\si@fam@maths
697           \csname\si@mathstt\endcsname
698           \expandafter\let\expandafter\si@fam@text
699           \csname\si@texttt\endcsname
700         \else
701           \si@log@debug{Font detection: rm}%
702           \expandafter\let\expandafter\si@fam@maths
703           \csname\si@mathsdefault\endcsname
704           \expandafter\let\expandafter\si@fam@text
705           \csname\si@textdefault\endcsname
706         \fi
707       \fi

```

Inline maths is now handled.

```

708   \else
709     \si@log@debug{Font detection: inline maths}%
710     \ifx\f@family\si@tempa
711       \si@log@debug{Font detection: sf}%
712       \expandafter\let\expandafter\si@fam@maths
713       \csname\si@mathssf\endcsname

```

```

714         \expandafter\let\expandafter\si@fam@text
715         \csname\si@textsf\endcsname
716     \else
717         \ifx\f@family\si@tempb
718         \si@log@debug{Font detection: tt}%
719         \expandafter\let\expandafter\si@fam@maths
720         \csname\si@mathstt\endcsname
721         \expandafter\let\expandafter\si@fam@text
722         \csname\si@texttt\endcsname
723     \else
724         \si@log@debug{Font detection: rm}%
725         \expandafter\let\expandafter\si@fam@maths
726         \csname\si@mathsdefault\endcsname
727         \expandafter\let\expandafter\si@fam@text
728         \csname\si@textdefault\endcsname
729     \fi
730 \fi
731 \fi

```

Not in maths mode, so the text mode checks are carried out.

```

732 \else
733     \si@log@debug{Font detection: text}%
734     \ifx\f@family\si@tempa
735     \si@log@debug{Font detection: sf}%
736     \expandafter\let\expandafter\si@fam@maths
737     \csname\si@mathssf\endcsname
738     \expandafter\let\expandafter\si@fam@text
739     \csname\si@textsf\endcsname
740 \else
741     \ifx\f@family\si@tempb
742     \si@log@debug{Font detection: tt}%
743     \expandafter\let\expandafter\si@fam@maths
744     \csname\si@mathstt\endcsname
745     \expandafter\let\expandafter\si@fam@text
746     \csname\si@texttt\endcsname
747 \else
748     \si@log@debug{Font detection: rm}%
749     \expandafter\let\expandafter\si@fam@maths
750     \csname\si@mathsdefault\endcsname
751     \expandafter\let\expandafter\si@fam@text
752     \csname\si@textdefault\endcsname
753 \fi
754 \fi
755 \fi

```

If the local font is not to be matched, setting the fonts is rather less complex.

```

756 \else
757     \si@log@debug{Font detection: inactive}%
758     \expandafter\let\expandafter\si@fam@maths
759     \csname\si@mathsdefault\endcsname
760     \expandafter\let\expandafter\si@fam@text
761     \csname\si@textdefault\endcsname
762 \fi

```

`\si@fam@bold` With the font family set, the next check is for bold text. This again needs to
`\si@fam@setbold`

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 commands are set up with `\def`, as nested calls can occur.

```
763 \def\si@fam@bold{\unboldmath\mdseries}%
764 \def\si@fam@setbold{\boldmath\bfseries}%
765 \ifsi@obeybold
766 \si@log@debug{Weight detection: checking weight}%
767 \ifmmode
```

Display maths.

```
768 \ifdim\displaywidth>0pt\relax
769 \si@fam@ifbmaths
770 {\let\si@fam@bold\si@fam@setbold
771 \si@log@debug{Weight detection: bold weight}}%
```

Inline maths.

```
772 \else
773 \si@fam@ifbinline
774 {\let\si@fam@bold\si@fam@setbold
775 \si@log@debug{Weight detection: bold weight}}%
776 \fi
```

Text mode.

```
777 \else
778 \si@fam@ifbtext%
779 {\let\si@fam@bold\si@fam@setbold
780 \si@log@debug{Weight detection: bold weight}}%
781 \fi
782 \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.

```
783 \let\si@fam@italic\upshape
784 \ifsi@obeyitalic
785 \si@log@debug{Italic detection: checking italic}%
786 \si@fam@ifitext
787 {\let\si@fam@italic\relax
788 \si@log@debug{Italic detection: italic}}%
789 \ifsi@textmode\else
790 \si@log@inf{maths mode - obeyitalic inactive}%
791 \fi
792 \fi}
```

16.9 Formatting numbers

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

```
\num[options]{num}
793 \DeclareRobustCommand*\num}[2][]{%
```

```

794 \begingroup%
795   \sisetup{#1}%
796   \si@fam@mode%
797   \si@log@debug{Processing \string\num\space input `#2'}%
798   \expandafter\si@out@num\expandafter{\si@num{#2}}%
799 \endgroup}

```

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

```
\si@num{<num>}
```

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

`\si@tempa` 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. A bit of trickery is needed to avoid getting an extraneous space introduced here by `\scantokens`, hence the use of `\@empty`.

```

801 \begingroup
802   \makeatletter%
803   \@makeother{\,%
804   \@makeother{.}%
805   \@makeother{+}%
806   \@makeother{-}%
807   \def~{}%
808   \def\,%
809   \catcode`\~=\active\relax
810   \catcode`\^=\active\relax
811   \scantokens{\si@num@xdef\si@tempa{#1}\@empty}%
812 \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 no argument, nothing happens; everything else in the `\si@num` macro occurs only if the argument is filled.

```
813 \si@ifnotmtarg{\si@tempa}{%
```

The input is now validated. Further processing takes place a little later.

```
814   \expandafter\si@num@valid\si@tempa\@empty\@empty%
```

If the input is valid, the input is passed to the number formatter.

```

815   \ifsi@switch
816     \expandafter\si@num@format\expandafter\si@tempa%
817   \else

```

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

```
818     {#1}%
```

```
819   \fi}}
```

`\si@num@xdef` When carrying out the `\edef` used to fully-expand a number, `\`, and `~` are deactivated, so that macros do not end up in the number. By using a separate

macro, `\scantokens` is easier. `\protected@xdef` is not used here, as the argument given should only contain single (processable) characters or macros that expand to the same, not other macros or characters.

```
\si@num@xdef<macro><num>\@empty
820 \def\si@num@xdef#1#2\@empty{\xdef#1{#2}}
```

`\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.

```
\si@num@valid<char><chars>\@empty
821 \def\si@num@valid#1#2\@empty{%
822   \si@num@ifchr{#1}{\si@numvalid}{%
823     \ifx\@empty#2\@empty\else
824       \si@num@valid#2\@empty\@empty%
825     \fi
826     \si@switchtrue}%
```

If an invalid character has been picked up, the whole parsing system has to bail out.

```
827   {\si@log@err{Invalid character '#1' in numerical input}%
828     {Only characters from the list '\si@numvalid'\MessageBreak
829       should be present in the argument of the \string\num
830       macro\MessageBreak (or derivative such as an 's' column)}}%
831   \si@switchfalse}}
```

`\si@num@ifchr` A test is needed to check one string only contains characters from a second. The main macro sets up for the recursion system below.

```
\si@tempa \si@num@ifchr{<test-chars>}{<valid-chars>}
\si@tempb
```

```
832 \newcommand*{\si@num@ifchr}[2]{%
833   \begingroup
834     \si@switchfalse
835     \def\si@tempa{#1}%
836     \edef\si@tempb{#2}%
```

Now the test can occur for the initial comparison string.

```
837   \expandafter\si@num@chrstr\si@tempb\@empty\@empty%
```

By ending the group inside the `\if`, `\global` is avoided, and the switch can be used for other jobs.

```
838   \ifsi@switch%
839     \endgroup\expandafter\@firstoftwo%
840   \else
841     \endgroup\expandafter\@secondoftwo%
842   \fi}
```

`\si@num@chrstr` The second part of the comparison macro does the actual work. This takes one character of the string of valid input at a time, and compares it to the single character in `\si@tempa`.

```
\si@num@chrstr<char><chars>\@empty
843 \def\si@num@chrstr#1#2\@empty{%
```

```
\si@tempb \si@tempc is used to hold the single character to check against \si@tempa,
\si@tempc while \si@tempb stores the remaining characters to be compared.
```

```

844 \def\si@tempc{#1}%
845 \edef\si@tempb{#2}%
846 \ifx\si@tempa\si@tempc
847   \si@switchtrue
848 \else

```

If \si@tempb is \@empty, then the recursion has bottomed-out, and all of the comparisons are done. If not, go round again.

```

849   \ifx\@empty\si@tempb\@empty\else
850     \si@num@chrstr#2\@empty%
851   \fi
852 \fi}

```

\si@num@exp Various storage macros are needed.

```

\si@num@mant 853 \newcommand*\si@num@exp{}
\si@num@expout 854 \newcommand*\si@num@mant{}
\si@num@mantout 855 \newcommand*\si@num@expout{}
\si@num@out 856 \newcommand*\si@num@mantout{}
857 \newcommand*\si@num@out{}

```

\si@num@format The number processor starts by saving #1 (odd things happen otherwise), and locally clearing the stacks.

\si@num@arg \si@num@format {*num*}

```

858 \newcommand*\si@num@format}[1]{%
859   \protected@edef\si@num@arg{#1}%
860   \def\si@num@exp{}%
861   \def\si@num@mant{}%
862   \def\si@num@expout{}%
863   \def\si@num@mantout{}%
864   \si@log@debug{Formatting number '\si@num@arg'}%
865   \si@switchfalse

```

The input is split into an mantissa and an exponent.

```
866 \expandafter\si@num@mantexp\si@num@arg\@empty\@empty%
```

The mantissa and exponent are now processed separately. Firstly, a sign is tested for. If there is one, it is added to \si@num@out, while the rest of the number is returned “as is” in \si@num@... .

```
867 \si@num@sign{mant}%

```

\si@num@out To allow for the case where the mantissa is only a sign, but the exponent contains a number, the output is initially defined to whatever is in \si@num@mantout. This will change if there is a number in \si@num@mant.

```

868 \protected@edef\si@num@out{\si@num@mantout}%
869 \ifx\@empty\si@num@mant\@empty\else
870   \si@num@digits{mant}%
871   \protected@edef\si@num@out{\si@num@mantout}%
872 \fi
873 \si@num@sign{exp}%
874 \ifx\@empty\si@num@exp\@empty\else
875   \si@num@digits{exp}%

```

Allowance is made for the possibility of negative exponential-only numbers. Precautions are taken for the multiply sign (which is always in maths mode), and superscripts. `\textsuperscript` is used here, as this will work in text or maths mode in the output routine.

```

876     \ifx\@empty\si@num@mant\@empty\else
877         \protected@edef\si@num@out%
878             {\si@num@out\noexpand\ensuremath{{}\noexpand\si@expproduct{}}}%
879     \fi
880     \protected@edef\si@num@out%
881         {\si@num@out\si@exppower%
882             \noexpand\textsuperscript{\si@num@expout}}%
883     \fi

```

If there is nothing in either number macro, then something is wrong.

```

884     \ifx\@empty\si@num@mant\@empty
885         \ifx\@empty\si@num@exp\@empty
886             \si@log@err{Invalid number format '\si@num@arg'}%
887             {Something is wrong with the number format; does it
888             contain \MessageBreak any numbers (from the list %
889             '\si@numlist')?}%
890         \renewcommand*\si@num@out{}%
891     \fi
892 \fi%

```

With everything done, the result is output.

```

893     \si@num@out}

```

`\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.

```

\si@num@mantexp⟨char⟩⟨chars⟩\@empty
894 \def\si@num@mantexp#1#2\@empty{%
895     \si@num@ifchr{#1}{\si@num@gobble}{}%
896     \si@num@ifchr{#1}{\si@num@exp}%
897     {\ifsi@switch
898         \si@log@err{Duplicate exponent marker found}%
899         {Only a single exponent character (from the list
900         '\si@num@exp')\MessageBreak may occur in a
901         numerical argument}%
902     \fi
903     \si@switchtrue
904     \si@log@debug{Exponent marker '#1' found in '\si@num@arg'}}%

```

When building up the mantissa and exponent, everything must be expandable, so `\edef` can be used rather than `\g@addto@macro`.

```

905     {\ifsi@switch
906         \si@log@debug{Adding '#1' to exponent for '\si@num@arg'}%
907         \protected@edef\si@num@exp{\si@num@exp#1}%
908     \else
909         \si@log@debug{Adding '#1' to mantissa for '\si@num@arg'}%
910         \protected@edef\si@num@mant{\si@num@mant#1}%
911     \fi}}%

```

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

```
912 \ifx\@empty#2\@empty\else
913   \si@num@mantexp#2\@empty\@empty\@empty%
914 \fi}
```

`\si@num@sign` The digit processor does several things to convert the run of digits, plus potentially a sign and a decimal point into the correct format for output.

`\si@num@sign{<mant/exp>}`

```
915 \newcommand*{\si@num@sign}[1]{%
916   \expandafter\ifx\expandafter\@empty\csname si@num@#1\endcsname%
917   \@empty\else
```

If the whole argument is not empty, then the sign-testing macro is run. This will return the sign in `\si@tempa` and the digits in `\si@tempb`.

```
918   \expandafter\expandafter\expandafter\si@num@gensign%
919   \csname si@num@#1\endcsname\@empty\@empty\@empty%
```

`\si@tempa` If a sign has to be added to unsigned numbers, this is done here.

```
\si@tempc 920   \edef\si@tempc{#1}%
921   \ifx\@empty\si@tempa\@empty
922     \def\si@tempa{mant}%
923     \ifx\si@tempa\si@tempc
924       \ifsi@num@signmant
925         \si@log@debug{Adding sign \si@sign\space to mantissa for
926           '\si@num@arg'}%
927         \protected@edef\si@tempa{\si@sign}%
928       \else
929         \def\si@tempa{}%
930       \fi
931     \else
932       \ifsi@num@signexp
933         \si@log@debug{Adding sign \si@sign\space to exponent for
934           '\si@num@arg'}%
935         \protected@edef\si@tempa{\si@sign}%
936       \else
937         \def\si@tempa{}%
938       \fi
939     \fi
940   \fi
```

If there is no sign, then the original macro contains a pure number, and nothing happens (an empty number has already been tested for).

```
941   \ifx\@empty\si@tempa\@empty
942     \def\si@tempa{mant}%
943     \ifx\si@tempa\si@tempc
944       \si@log@debug{Unsigned mantissa for '\si@num@arg'}%
945     \else
946       \si@log@debug{Unsigned exponent for '\si@num@arg'}%
947     \fi
948   \else
```

There is a sign, so it is added to the output stack.

```
949   \expandafter\protected@edef\csname si@num@#1\out\endcsname%
950   {\noexpand\ensuremath{\si@tempa}}%
```

A sign but no number can only be correct if the input is something like $-e10$ to give -10^{10} .

```
951     \ifx\@empty\si@tempb\@empty
952     \expandafter\def\csname si@num@#1\endcsname{}
```

\si@tempa \si@tempa is no longer needed, so can be reused.

```
953     \def\si@tempa{mant}%
```

Checks to see if this is a mantissa, and that there is an exponent.

```
954     \ifx\si@tempa\si@tempc
955     \ifx\@empty\si@num@exp\@empty
956     \si@log@warn{Sign but no number for '\si@num@arg'}%
957     \fi
958     \else
959     \si@log@warn{Sign but no number for '\si@num@arg'}%
960     \fi
961     \else
962     \expandafter\protected@edef\csname si@num@#1\endcsname%
963     {\si@tempb}%
964     \fi
965     \fi
966     \fi}
```

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

```
\si@num@gensign<char><char><chars>\@empty
967 \def\si@num@gensign#1#2#3\@empty{%
968   \si@num@ifchr{#1}{\si@numsign}{%
969     \si@num@ifchr{#2}{\si@numsign}{%
970       \if +#1
971       \if -#2
972         \si@log@debug{Found sign combination +- for '\si@num@arg'}%
973         \def\si@tempa{\pm}%
974       \else
975         \si@log@warn{Unknown sign combination '#1#2'}%
976         \def\si@tempa{#1#2}%
977       \fi
978     \else
979       \if -#1
980       \if +#2
981         \si@log@debug{Found sign combination -+ for '\si@num@arg'}%
982         \def\si@tempa{\mp}%
983       \else
984         \si@log@warn{Unknown sign combination '#1#2'}%
985         \def\si@tempa{#1#2}%
986       \fi
987     \else
988       \si@log@warn{Unknown sign combination '#1#2'}%
989       \def\si@tempa{#1#2}%
990     \fi
```

```

991     \fi
992     \edef\si@tempb{#3}}%

```

Only one valid sign character.

```

993     {\si@log@debug{Found single sign character '#1' for
994         '\si@num@arg'}%
995     \def\si@tempa{#1}%
996     \edef\si@tempb{#2#3}}}%

```

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

```

997     {\si@log@debug{No sign found for '\si@num@arg'}%
998     \def\si@tempa{}%
999     \edef\si@tempb{#1#2#3}}}%

```

`\si@num@digits` The core digit processor divides the number into the parts before and after the decimal point marker.

```

\si@tempb \si@num@digits{<mant/exp>}
\si@tempc

```

```

1000 \newcommand*{\si@num@digits}[1]{%
1001   \def\si@tempa{}%
1002   \def\si@tempb{}%

```

The package switch is used to indicate finding a decimal marker.

```

1003   \si@switchfalse
1004   \expandafter\expandafter\expandafter\si@num@split%
1005   \csname si@num@#1\endcsname\@empty\@empty%

```

The pre-decimal part of the number is now in `\si@tempa`, and the post-decimal part in `\si@tempb`. A quick check is made on the pre-decimal part of the number.

```

1006   \ifx\@empty\si@tempa\@empty
1007     \ifsi@num@padlead
1008       \si@log@debug{Adding leading zero for '\si@num@arg'}%
1009       \def\si@tempa{0}%
1010     \fi
1011   \fi

```

A second test is needed, in case a zero should be added when a decimal marker is followed by nothing at all. Here, the fact that a decimal marker was found is needed; the test is done now so `\ifsi@switch` can be reused.

```

1012   \ifx\@empty\si@tempb\@empty
1013     \ifsi@num@padtrail
1014       \ifsi@switch
1015         \si@log@debug{Adding trailing zero for '\si@num@arg'}%
1016         \def\si@tempb{0}%
1017       \fi
1018     \fi
1019   \fi

```

The contents of `\si@tempa` and `\si@tempb` are now completed. Some error checking is done, in case an odd argument has been given.

```

1020   \ifx\@empty\si@tempa\@empty
1021     \ifx\@empty\si@tempb\@empty\else
1022       \si@num@sepdigits{#1}%
1023     \fi
1024   \else
1025     \si@num@sepdigits{#1}%

```



```
1026 \fi}
```

`\si@num@split` The `\si@num@split` macro compares each character in the input against the list of characters valid at this stage: numbers, decimal markers and “extra” characters. Before finding a decimal marker, numbers and extra characters are added to `\si@tempa`; after a decimal is found, characters are added to `\si@tempb`.

```
\si@num@split⟨char⟩⟨chars⟩\@empty
1027 \def\si@num@split#1#2\@empty{%
1028 \si@num@ifchr{#1}{\si@numdecimal}}%
1029 \ifsi@switch
1030 \si@log@err{Duplicate decimal marker in '\si@num@arg'}
1031 {Only a single decimal marker (from the list
1032 '\si@numdecimal')\MessageBreak may occur in a
1033 numerical argument}}%
1034 \else
1035 \si@log@debug{Found decimal marker '#1' in '\si@num@arg'}}%
1036 \si@switchtrue
1037 \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.

```
1038 \si@num@ifchr{#1}{\si@numsign}}%
1039 \si@log@err{Misplaced sign in '\si@num@arg'}
1040 {Sign characters '\si@numsign' can only occur\MessageBreak
1041 at the start of a number}}%
```

The current character is added to the appropriate stack.

```
1042 \ifsi@switch
1043 \si@log@debug{Adding '#1' to decimal part for '\si@num@arg'}}%
1044 \protected@edef\si@tempb{\si@tempb#1}}%
1045 \else
1046 \si@log@debug{Adding '#1' to integer part for '\si@num@arg'}}%
1047 \protected@edef\si@tempa{\si@tempa#1}}%
1048 \fi}}%
```

Unless the recursion has bottomed, loop round again.

```
1049 \ifx\@empty#2\@empty\else
1050 \si@num@split#2\@empty\@empty%
1051 \fi}
```

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

```
1052 \newcommand*{\si@num@decimalhook}{}%
```

`\si@num@sepdigits` The `\si@num@sepdigits` macro is only called if at least one of the mantissa and exponent contain something to output. The integer and decimal parts of the number are processed separately. First, a check is made to see if each part contains “extra” characters; if it does, no digit-separating is even attempted.

```
\si@num@sepdigits{⟨mant/exp⟩}
1053 \newcommand*{\si@num@sepdigits}[1]{%
1054 \si@num@ifextra{\si@tempa}{}%
```

`\si@tempc` If the input is a pure number, then separation is attempted.

```
1055 {\expandafter\si@num@int\expandafter{\si@tempa}}%
```

```

1056 \def\si@tempc{}%
1057 \ifx\@empty\si@tempb\@empty\else
1058   \protected@edef\si@tempc{\noexpand\ensuremath{\{ \noexpand%
1059     \si@decimalsign}}}%
1060   \si@num@decimalhook%
1061   \si@num@ifextra{\si@tempb}{}%
1062   {\expandafter\si@num@dec\expandafter{\si@tempb}}%
1063 \fi

```

The construction is finalised by re-combining the number.

```

1064 \expandafter\protected@edef\csname si@num@#1out\endcsname%
1065   {\csname si@num@#1out\endcsname\si@tempa\si@tempc\si@tempb}}

```

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

```

\si@num@extra \si@num@ifextra{<integer>}
\si@num@extra<char><chars>\@empty
1066 \newcommand*{\si@num@ifextra}[1]{%
1067   \begingroup
1068     \si@switchfalse
1069     \expandafter\si@num@extra#1\@empty\@empty%
1070     \ifsi@switch
1071       \si@log@debug{Found ‘extra’ characters in ‘#1’}%
1072       \endgroup\expandafter\@firstoftwo%
1073     \else
1074       \endgroup\expandafter\@secondoftwo%
1075     \fi}
1076 \def\si@num@extra#1#2\@empty{%
1077   \ifx\@empty#1\@empty\else
1078     \si@num@ifchr{#1}{\si@num@extra}{\si@switchtrue}{}%
1079     \ifx\@empty#2\@empty\else
1080       \si@num@extra#2\@empty\@empty%
1081     \fi
1082 \fi}

```

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

```

\si@num@int {\integer-part}
1083 \newcommand*{\si@num@int}[1]{%
1084   \def\si@tempa{}%
1085   \ifsi@sepfour
1086     \si@num@intfmt{#1}\@empty\@empty\@empty%
1087   \else
1088     \si@num@iffive{#1}
1089     {\si@num@intfmt{#1}\@empty\@empty\@empty}
1090     {\def\si@tempa{#1}}%
1091   \fi}

```

`\si@num@iffive` A test is needed for the presence of more than four characters.
`\si@num@five` `\si@num@iffive{<num>}`

`\si@num@five<char><char><char><char><chars>\end`

```

1092 \newcommand*{\si@num@iffive}[1]{%
1093   \si@num@five#1\@empty\@empty\@empty\@empty\@empty\end}
1094 \def\si@num@five#1#2#3#4#5\end{%
1095   \ifx\@empty#5\@empty
1096     \expandafter\@secondoftwo%
1097   \else
1098     \expandafter\@firstoftwo%
1099   \fi}

```

`\si@num@intfmt` **The business end of the integer formatter.** `\si@num@intfmt{<char>}{<char>}{<char>}{<char>}`
`\si@num@fiint` `\si@num@fiint<chars>\fi\fi\fi`

```

1100 \newcommand*{\si@num@intfmt}[4]{%
1101   \ifx\@empty#2\@empty
1102     \si@num@intsep#1\relax
1103   \else
1104     \ifx\@empty#3\@empty
1105       \si@num@intsep\@empty\@empty#1#2\relax
1106     \else
1107       \ifx\@empty#4\@empty
1108         \si@num@intsep\@empty#1#2#3\relax
1109       \else
1110         \si@num@fiint{#1#2#3#4}%
1111       \fi
1112     \fi
1113   \fi}
1114 \def\si@num@fiint#1\fi\fi\fi{\fi\fi\fi\si@num@intfmt{#1}}

```

`\si@num@intsep` **For adding separation to integers, an extra function is needed.**

```

\si@tempa \si@num@intsep{<char>}{<char>}{<char>}{<char>}
1115 \newcommand*{\si@num@intsep}[4]{%
1116   \protected@edef\si@tempa{\si@tempa#1#2#3}%
1117   \if\relax#4\relax
1118   \else
1119     \protected@edef\si@tempa{\si@tempa\noexpand\ensuremath{\noexpand%
1120       \si@digitsep}}%
1121     \expandafter\si@num@intsep\expandafter#4%
1122   \fi}

```

`\si@num@dec` **Formatting a decimal uses a similar mechanism, but with a few alterations needed.**

`\si@num@decfmt` `\si@num@dec{<decimal-part>}`
`\si@tempb` `\si@num@decfmt{<char>}{<char>}{<char>}{<char>}`

```

1123 \newcommand*{\si@num@dec}[1]{%
1124   \def\si@tempb{ }%
1125   \ifsi@sepfour
1126     \si@num@decfmt#1\@empty\@empty\@empty\@empty%
1127   \else
1128     \si@num@iffive{#1}
1129     {\si@num@decfmt#1\@empty\@empty\@empty\@empty}
1130     {\protected@edef\si@tempb{\si@tempb#1}}%

```

```

1131 \fi
1132 }
1133 \newcommand*{\si@num@decfmt}[4]{%
1134 \protected@edef\si@tempb{\si@tempb#1#2#3}%
1135 \ifx\@empty#4\@empty%
1136 \else
1137 \protected@edef\si@tempb{\si@tempb\noexpand\ensuremath{\noexpand%
1138 \si@digitsep}}%
1139 \expandafter\si@num@decfmt\expandafter#4%
1140 \fi}

```

16.10 Formatting angles

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

```

\ang[options]{decimal-angle}
\ang[options]{deg;min;sec}

```

```

1141 \DeclareRobustCommand*\ang}[2][]{%
1142 \begingroup
1143 \sisetup{#1}%
1144 \si@fam@mode%
1145 \si@log@debug{Processing \string\ang\space input `#2'}%
1146 \@makeother{;}%
1147 \makeatletter%
1148 \scantokens{\si@ang@parse#2;;;\@nil}}

```

`\si@ang@parse` With the correct catcodes in place, processing can take place strip out the semicolons.

```

\si@ang@parse<num>;<num>;<num>;<chars>\@nil

```

```

1149 \def\si@ang@parse#1;#2;#3;#4\@nil{\si@ang@set{#1}{#2}{#3}}

```

`\si@ang@killdegree` A mechanism is needed to handle moving the angle unit signs for the `astroang` option. This requires two steps, producing the sign over the decimal sign and preventing duplicate symbols appearing.

```

\si@ang@killminute
\si@ang@killsecond
\si@ang@astrosign \si@ang@astrosign{<degree/minute/second>}
\si@ang@decimalsign1150 \newcommand*{\si@ang@killdegree}{\let\si@sym@degree\relax}
1151 \newcommand*{\si@ang@killminute}{\let\si@sym@minute\relax}
1152 \newcommand*{\si@ang@killsecond}{\let\si@sym@second\relax}
1153 \newcommand*{\si@ang@astrosign}[1]{%
1154 \renewcommand*{\si@decimalsign}{%
1155 \rlap{\si@ang@decimalsign}%
1156 \expandafter\csname si@sym@#1\endcsname}%
1157 \def\si@num@decimalhook{\expandafter\aftergroup%
1158 \csname si@ang@kill#1\endcsname}}%

```

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

```

\si@ang@set{<num>}{<num>}{<num>}

```

```

1159 \newcommand*{\si@ang@set}[3]{%

```

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

```

\si@ang@mins1160 \ifsi@ang@padlarge

```

```

\si@ang@secs

```

```

1161     \newcommand*{\si@ang@degs}{0\si@sym@degree}%
1162     \newcommand*{\si@ang@mins}{0\si@sym@minute}%
1163     \newcommand*{\si@ang@secs}{0\si@sym@second}%
1164 \else
1165     \newcommand*{\si@ang@degs}{}%
1166     \newcommand*{\si@ang@mins}{}%
1167     \newcommand*{\si@ang@secs}{}%
1168 \fi
1169 \protected@edef\si@ang@decimalsign{\si@decimalsign}%

```

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

```

1170 \ifsi@astroang
1171     \let\si@ang@movesign\si@ang@astroang
1172 \else
1173     \let\si@ang@movesign\@gobble
1174 \fi

```

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

```

1175 \newcommand*{\si@ang@secnum}{\si@ang@num{second}}%
1176 \newcommand*{\si@ang@minnum}{\si@ang@num{minute}}%
1177 \si@ifnotmtarg{#3}
1178     {\si@log@debug{Found seconds `#3'}%
1179     \renewcommand*{\si@ang@secs}
1180     {\si@ang@secnum{#3}\si@sym@second}%
1181     \renewcommand*{\si@ang@mins}
1182     {\si@ang@pad{0\si@sym@minute}}%
1183     \renewcommand*{\si@ang@degs}
1184     {\si@ang@pad{0\si@sym@degree}}}%
1185 \si@ifnotmtarg{#2}
1186     {\si@log@debug{Found minutes `#2'}%
1187     \renewcommand*{\si@ang@secnum}{\si@ang@signlessnum{second}}%
1188     \renewcommand*{\si@ang@mins}
1189     {\si@ang@minnum{#2}\si@sym@minute}%
1190     \renewcommand*{\si@ang@degs}
1191     {\si@ang@pad{0\si@sym@degree}}}%
1192 \si@ifnotmtarg{#1}
1193     {\si@log@debug{Found degrees `#1'}%
1194     \renewcommand*{\si@ang@secnum}{\si@ang@signlessnum{second}}%
1195     \renewcommand*{\si@ang@minnum}{\si@ang@signlessnum{minute}}%
1196     \renewcommand*{\si@ang@degs}

```

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

```

1197     {\si@ang@num{degree}{#1}%
1198     \si@sym@degree}}%
1199 \si@out@num%
1200     {\si@ang@degs\si@anglesep\si@ang@mins\si@anglesep\si@ang@secs}%

```

The group opened by `\ang` is closed.

```

1201 \endgroup}

```

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

```
\si@ang@pad{<num>}
1202 \newcommand*{\si@ang@pad}[1]{%
1203   \ifsi@ang@padsmall
1204     #1%
1205   \else
1206     \relax%
1207   \fi}
```

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

```
\si@ang@signlessnum{<degree/minute/second>}{<num>}
\si@ang@num{<degree/minute/second>}{<num>}
1208 \newcommand{\si@ang@num}[2]{%
1209   \begingroup%
1210     \si@ang@movesign{#1}%
1211     \si@num{#2}%
1212   \endgroup}
1213 \newcommand{\si@ang@signlessnum}[2]{%
1214   \begingroup%
1215     \si@ang@movesign{#1}%
1216     \ssetup{addsign=none}%
1217     \si@num{#2}%
1218   \endgroup}
```

16.11 New column types

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 first part of the job is to make a new column type. The letters D, N and R are taken by other packages, so s (for si) is chosen. As in rccol and numprint, initially no definition is given as lots of code needs to be added.

```
1219 \newcolumnntype{s}{} 
```

`\NC@rewrite@s` Following the numprint approach, the `\NC@rewrite@s` macro is now changed to provide a hook for the collection of the tabular material. 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. After resetting the storage token registers, the internal macro which handles optional arguments is called.

```
1220 \renewcommand*{\NC@rewrite@s}{%
1221   \@ifnextchar[%
1222     {\si@tab@rewrite}
1223     {\si@tab@rewrite[]}}
1224 }
```

`\si@tab@rewrite` An optional argument can now be picked up (this does not work using the `\si@tempa` optional argument to `\renewcommand` for `\NC@rewrite@s`). 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.

```
\si@tab@rewrite[<options>]
```

```

1225 \def\si@tab@rewrite[#1]{%
1226   \edef\si@tempa{\the\@temptokena
1227     >\noexpand\si@tab@begin[#1]\noexpand\ignorespaces}c%
1228     <\noexpand\si@tab@end}}%
1229   \@temptokena\expandafter\si@tempa}%

```

With the assignment done, the normal action of the array package is continued.

```
1230   \NC@find}
```

`\si@tab@numtoks` 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).

```

1231 \newtoks\si@tab@numtoks
1232 \newtoks\si@tab@pretoks
1233 \newtoks\si@tab@posttoks

```

`\si@tab@begin` The lead-off macro starts by setting any local values for `\sisetup`. Although this is an internal macro, square brackets for the option list are retained to make the option that this argument may be empty.

```
\si@tab@begin[options]
```

```

1234 \newcommand*\si@tab@begin[1][]{%
1235   \begingroup
1236     \sisetup{#1}%
1237     \si@tab@numtoks{}%
1238     \si@tab@pretoks{}%
1239     \si@tab@posttoks{}%
1240     \si@switchfalse
1241     \si@log@debug{Processing s column cell contents}%
1242     \si@tab@gettok}

```

`\si@tab@gettok` The iteration macro for collecting up tokens in the input. This is a pretty much direct copy from `numprint`. First #1 is checked against various special values.

```
\si@tab@next \si@tab@gettok{cell-contents}
```

```
1243 \newcommand*\si@tab@gettok[1]{%
```

The current cell could be the end of a line.

```

1244   \ifx\tabularnewline#1
1245     \let\si@tab@next\tabularnewline
1246   \else

```

If the table is in the usual `\begin... \end` construct, the `\end` might appear.

```

1247     \ifx\end#1
1248       \let\si@tab@next\end
1249     \else

```

The check for `\si@tab@end` deals with the likely situation that the current cell is not the last of the line; the result will be that the end-of-cell macro will be present.

```

1250       \ifx\si@tab@end#1
1251         \let\si@tab@next\si@tab@end
1252       \else

```

If `\begin... \end` has not been used for the table, then `\endtabular` might crop up.

```
1253       \ifx\endtabular#1
```

```

1254         \let\si@tab@next\endtabular
1255         \else

```

Apparently, `tabularx` might have a `\csname` at the end of the cell.

```

1256         \ifx\csname#1
1257         \let\si@tab@next\csname
1258         \else
    \relax is always a possibility.
1259         \ifx\relax#1\relax
1260         \let\si@tab@next\relax
1261         \else

```

If the macro gets here, then the input should be stored, either as part of a number or to be appended to the number. This is checked by using `\si@numvalid`; initially this will hold the valid characters defined before, but will be altered to `\@empty` if collection of items after a number is in operation. The system is set to recur, and the input is saved to the appropriate token register.

```

1262         \let\si@tab@next\si@tab@gettok
1263         \si@num@ifchr{#1}{\si@numvalid}
1264         {\si@switchtrue
1265         \si@log@debug{Found valid cell contents '#1'}%
1266         \si@tab@numtoks=\expandafter{\the\si@tab@numtoks#1}}
1267         {\si@log@debug{Found other cell contents \string#1}%
1268         \si@tab@othertok{#1}}%
1269         \fi
1270     \fi
1271 \fi
1272 \fi
1273 \fi
1274 \fi

```

Finally, execute whatever should be the next step.

```

1275 \si@tab@next}

```

`\si@tab@othertok` Unrecognised input is added to a token register, either before or after the number.
`\si@tab@othertok{<chars>}`

```

1276 \newcommand*\si@tab@othertok}[1]{%
1277 \ifsi@switch

```

If working after a number has been found, it is necessary to prevent any more input being added to the number.

```

1278     \si@tab@posttoks=\expandafter{\the\si@tab@posttoks#1}%
1279 \else
1280     \si@tab@pretoks=\expandafter{\the\si@tab@pretoks#1}%
1281 \fi}

```

`\si@tab@end` At the end of the cell, the actual output has to occur.

```

1282 \newcommand{\si@tab@end}{%
1283 \hfil%
1284 \the\si@tab@pretoks%

```

If no number was found, then output is skipped.

```

1285 \ifsi@switch
1286 \expandafter\si@tab@numout%
1287 \fi

```



```

1288 \the\si@tab@posttoks%
1289 \hfil%
1290 \endgroup}

```

`\si@tempcnta` Counters are needed for the digit-counting system.

```

\si@tempcntb
1291 \newcount\si@tempcnta
1292 \newcount\si@tempcntb

```

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

```

1293 \newcommand*{\si@tab@numout}{%
1294 \let\si@num@format\si@tab@num@format

```

Using the modified form of `\si@num`, the input (in `\si@tab@numtoks` is parsed. This results in the mantissa before the decimal place in `\si@num@mantout` and the rest of the mantissa, plus any exponent part, in `\si@num@out`. The part of the mantissa after the decimal marker (if any) is stored in `\si@tab@mantout`, which can therefore be used as a flag for the inclusion of a decimal sign.

```

1295 \expandafter\si@num\expandafter{\the\si@tab@numtoks}%
1296 \afterassignment\si@tab@format\expandafter\si@tempcnta%
1297 \si@tab@format\relax}

```

`\si@tab@mantout` A storage macro is needed.

```

1298 \newcommand*{\si@tab@mantout}{}

```

`\si@tab@num@format` A modified version of `\si@num@format` is needed, as the “decomposed” number is needed directly by the table formatting system.

```

\si@tab@num@format{<num>}
1299 \newcommand*{\si@tab@num@format}[1]{%

```

`\si@tab@org@sepdigits` The crucial sub-macro is redirected.

```

1300 \let\si@tab@org@sepdigits\si@num@sepdigits
1301 \let\si@num@sepdigits\si@tab@num@sepdigits

```

`\si@num@arg` With that done, things continue as in the original.

```

1302 \edef\si@num@arg{#1}%
1303 \si@switchfalse
1304 \expandafter\si@num@mantexp\si@num@arg\@empty\@empty%
1305 \si@num@sign{mant}%
1306 \ifx\@empty\si@num@mant\@empty\else
1307 \si@num@digits{mant}%

```

`\si@num@out` The pre-decimal part of the the number is in `\si@num@mantout`, with the post-decimal part in `\si@tab@mantout`. This ensures that there is no need to shuffle the location of any sign. The macro now continues to build up everything after the decimal sign in `\si@num@out`.

```

1308 \protected@edef\si@num@out{\si@tab@mantout}%
1309 \fi

```

For the exponent, processing is back to normal.

```

1310 \let\si@num@sepdigits\si@tab@org@sepdigits
1311 \si@num@sign{exp}%
1312 \ifx\@empty\si@num@exp\@empty\else

```

```

1313 \si@num@digits{exp}%
1314 \ifx\@empty\si@num@mant\@empty\else
1315 \protected@edef\si@num@out%
1316 {\si@num@out\noexpand\ensuremath{\noexpand\si@expproduct}}%
1317 \fi
1318 \protected@edef\si@num@out%
1319 {\si@num@out\si@exppower%
1320 \noexpand\textsuperscript{\si@num@expout}}%
1321 \fi
1322 \ifx\@empty\si@num@mant\@empty
1323 \ifx\@empty\si@num@exp\@empty
1324 \si@log@err{Invalid number format '\si@num@arg'}%
1325 {Something is wrong with the number format; does it
1326 contain \MessageBreak any numbers (from the list %
1327 '\si@numlist'?)?}%

```

`\si@tab@mantout` Need to clear both storage areas.

```

1328 \renewcommand*\si@num@out{}%
1329 \renewcommand*\si@tab@mantout{}%
1330 \fi
1331 \fi}

```

`\si@tab@num@sepdigits` An altered version of `\si@num@sepdigits` is needed, so that the division of the data is made before and after the decimal sign for the mantissa.

```

\si@tab@num@sepdigits{\langle num \rangle}
1332 \newcommand*\si@tab@num@sepdigits[1]{%
1333 \si@num@ifextra{\si@tempa}{%
1334 {\expandafter\si@num@int\expandafter{\si@tempa}}%
1335 \def\si@tempc{}%
1336 \ifx\@empty\si@tempb\@empty\else
1337 \si@num@ifextra{\si@tempb}{%
1338 {\expandafter\si@num@dec\expandafter{\si@tempb}}%
1339 \fi

```

The storage of the results is different to the original version. The pre-decimal part (plus any sign) is stored in the `\si@num@#1out` macro, while the post-decimal part ends up in `\si@tab@#1out`. No decimal sign is added in at all.

```

1340 \expandafter\protected@edef\csname si@num@#1out\endcsname%
1341 {\csname si@num@#1out\endcsname\si@tempa}%
1342 \expandafter\protected@edef\csname si@tab@#1out\endcsname%
1343 {\si@tempb}}%

```

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

```

1344 \newbox\si@tab@prebox
1345 \newbox\si@tab@postbox
1346 \newbox\si@tempbox

```

`\si@tab@format` The formatting set up is taken from `dcolum`, with a few minor changes to fit the scheme used here. There is only one argument here, as the appearance of the decimal sign is handled by the `keyval` system. The numerical test here has been changed, compared to `dcolum`, so that a value of zero gives a column centred on

the decimal marker.

```
\si@tab@format{<num>}\relax
1347 \def\si@tab@format#1\relax{%
1348   \ifnum\z@<\si@tempcnta
1349     \expandafter\si@tab@right%
1350   \else
1351     \expandafter\si@tab@centre%
1352   \fi
1353   {#1}%
```

Output of the formatted data occurs here; both positioning macros produce formatted data in boxes zero and two.

```
1354   \box\si@tab@prebox\box\si@tab@postbox}
```

`\si@tab@centre` This macro is executed if the decimal marker is at the centre of the column. The argument is needed here to throw away anything left on the input stack by `\si@tab@format`. Unlike `dcolumn`, only a single macro is needed here, as the (divided) number is already available.

```
\si@tab@centre{<gobble>}
1355 \newcommand*\si@tab@centre[1]{%
```

Box zero is used to hold the pre-decimal part, with box two holding the post-decimal part *if* it is needed.

```
1356   \setbox\si@tab@prebox=\hbox%
1357     {\expandafter\si@out@num\expandafter{\si@num@mantout}}%
1358   \ifx\@empty\si@tab@mantout\@empty
1359     \ifx\@empty\si@num@out\@empty
1360       \setbox\si@tab@postbox=\hbox%
1361         {\phantom{\ensuremath{\{\si@decimalsign\}}}}%
1362     \else
1363       \setbox\si@tab@postbox=\hbox%
1364         {\expandafter\si@out@num\expandafter{\si@num@out}}%
1365     \fi
1366   \else
1367     \setbox\si@tab@postbox=\hbox%
1368       {\ensuremath{\{\si@decimalsign\}}}%
1369     \expandafter\si@out@num\expandafter{\si@num@out}}%
1370   \fi
```

Which of the two boxes is wider is now checked, and the smaller is padded out.

```
1371   \ifdim \wd\si@tab@prebox>\wd\si@tab@postbox
1372     \setbox\si@tab@postbox=\hbox to\wd\si@tab@prebox%
1373       {\unhbox\si@tab@postbox\hfill}%
1374   \else
1375     \setbox\si@tab@prebox=\hbox to\wd\si@tab@postbox%
1376       {\hfill\unhbox\si@tab@prebox}%
1377   \fi}
```

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

```
\si@tab@postdim1378 \newdimen\si@tab@predim
\si@tempdima1379 \newdimen\si@tab@postdim
\si@tempdimb1380 \newdimen\si@tempdima
1381 \newdimen\si@tempdimb
```

`\si@tab@right` The column is not centred on the decimal marker; the user specifies how many characters on each side are allowed for.

`\si@tab@right{<num>}`

1382 `\newcommand*\si@tab@right[1]{%`

The width of a character is measured, and stored.

1383 `\setbox\si@tempbox=\hbox{\si@out@num{1}}}`

1384 `\si@tempdima\wd\si@tempbox`

`\si@tab@prea` If #1 is empty, then no special processing is needed for box two. On the other
`\si@tab@preb` hand, if there is something in #1 then a bit of re-arranging is done. In particular notice that `\si@tempcnta` is used with the pre-decimal value, before the post-decimal setting is saved.

1385 `\ifx\relax#1\relax`

1386 `\hfill`

1387 `\let\si@tab@prea\relax`

1388 `\let\si@tab@preb\relax`

1389 `\else`

1390 `\si@tab@predim\the\si@tempcnta\si@tempdima`

1391 `\si@tab@sepcorr{pre}%`

1392 `\edef\si@tab@prea{to\si@tab@predim}%`

1393 `\edef\si@tab@preb{\hss\hfill}%`

1394 `\si@tempcnta@gobble#1\relax`

1395 `\fi`

The width of the box needed is calculated by multiplying the width of a character (in `\si@tempdima` by the number of characters requested (in `\si@tempcnta`). The width of the decimal sign is also allowed for and added on.

1396 `\si@tab@postdim\si@tempcnta\si@tempdima`

1397 `\setbox\si@tempbox=\hbox{\ensuremath{\{\si@decimalsign\}}}%`

1398 `\advance\si@tab@postdim\wd\si@tempbox`

1399 `\si@tab@sepcorr{post}%`

The pre-decimal part of the number is now added to box zero, with the post decimal part in box two if needed.

1400 `\setbox\si@tab@prebox=\hbox\si@tab@prea{\si@tab@preb%`

1401 `\expandafter\si@out@num\expandafter{\si@num@mantout}}%`

1402 `\ifx\@empty\si@tab@mantout\@empty`

1403 `\setbox\si@tab@postbox=\hbox to\si@tab@postdim%`

1404 `{\expandafter\si@out@num\expandafter{\si@num@out}\hfil}%`

1405 `\else`

1406 `\setbox\si@tab@postbox=\hbox to\si@tab@postdim%`

1407 `{\ensuremath{\{\si@decimalsign\}}\expandafter\si@out@num%`

1408 `\expandafter{\si@num@out}\hfil}%`

1409 `\fi}`

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

`\si@tab@sepcorr{<num>}`

1410 `\newcommand*\si@tab@sepcorr[1]{%`

1411 `\si@tempcntb\the\si@tempcnta\relax`

Calculate how many groups of three there are, then allow for not separating four characters if `\ifsi@sepfour` is false.

```
1412 \divide\si@tempcntb\thr@@
1413 \ifsi@sepfour\else
1414   \ifnum\the\si@tempcnta=4
1415     \si@tempcntb\z@
1416   \fi
1417 \fi
```

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

```
1418 \setbox\si@tempbox=\hbox{\ensuremath{\si@digitsep}}%
1419 \expandafter\advance\csname si@tab@#1dim\endcsname%
1420 \si@tempcntb\wd\si@tempbox%
```

16.12 Units

`\SI` There are two types of user macros for the units system; those for defining new units, prefixes and powers, and those for using them. There are two macros for using units, `\SI` and `\unitsym`, which work in very similar ways. `\unitsym` is just an alias for `\SI` with no number; everything is handed off into an internal macro. The internal macro also handles the optional prefix argument to `\SI`

```
\SI[<options>]{<num>}\unitsym[<options>]{<unit>}
1421 \DeclareRobustCommand*\SI}[2][{}]{%
1422   \@ifnextchar[%
1423     {\si@SI[#1]{#2}}
1424     {\si@SI[#1]{#2}[]}}
1425 \DeclareRobustCommand*\unitsym}[2][{\si@SI[#1]{}[]}{#2}]
```

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

```
\newunit[<valuesep=none>]{<unit>}{<symbol>}
\renewunit[<valuesep=none>]{<unit>}{<symbol>}
\provideunit[<valuesep=none>]{<unit>}{<symbol>}
1426 \newcommand*\newunit}[3][{}]{%
1427   \si@ifdefinable{#2}
1428     {\si@unt@defunit[#1]{#2}{#3}}
1429     {\si@log@err{Unit \string#2 already defined!}\@eha}}
1430 \newcommand*\renewunit}[3][{}]{%
1431   \si@ifdefinable{#2}
1432     {\si@log@err{Unit \string#2 undefined}\@ehc
1433     \si@unt@defunit[#1]{#2}{#3}}
1434     {\si@unt@defunit[#1]{#2}{#3}}}
1435 \newcommand*\provideunit}[3][{}]{%
1436   \si@ifdefinable{#2}
1437     {\si@unt@defunit[#1]{#2}{#3}}
1438   {}}
```

```

\newprefix   The multiples of units are defined here; very similar code is used to the
\renewprefix \newunit, etc., macros. The multiple prefixes cannot take an optional argument,
\provideprefix and must represent some power. Hence the arguments required are different.
\newprefix{<multiple>} {<powers-ten>} {<symbol>}
\renewprefix{<multiple>} {<powers-ten>} {<symbol>}
\provideprefix{<multiple>} {<powers-ten>} {<symbol>}

1439 \newcommand*{\newprefix}[3]{%
1440   \si@ifdefinable{#1}
1441     {\si@unt@defprefix{#1}{#2}{#3}}
1442     {\si@log@err{Prefix \string#1 already defined!}\@eha}}
1443 \newcommand*{\renewprefix}[3]{%
1444   \si@ifdefinable{#1}
1445     {\si@log@err{Prefix \string#1 undefined}\@ehc
1446     \si@unt@defprefix{#1}{#2}{#3}}
1447     {\si@unt@defprefix{#1}{#2}{#3}}}
1448 \newcommand*{\provideprefix}[3]{%
1449   \si@ifdefinable{#1}
1450     {\si@unt@defprefix{#1}{#2}{#3}}
1451   {}}

\newpower   Here power multiples for units are set up. As with units and multiples, a layered
\renewpower approach is used to keep things easy to maintain. The optional argument here is
\providepower not a keyval one: only post is a valid value.
\newpower[<post>]{<num>}{<power>}
\renewpower[<post>]{<num>}{<power>}
\providepower[<post>]{<num>}{<power>}

1452 \newcommand*{\newpower}[3][{}]{%
1453   \si@ifdefinable{#2}
1454     {\si@unt@defpower[#1]{#2}{#3}}
1455     {\si@log@err{Power \string#2 already defined!}\@eha}}
1456 \newcommand*{\renewpower}[3][{}]{%
1457   \si@ifdefinable{#2}
1458     {\si@log@err{Power \string#2 undefined}\@ehc
1459     \si@unt@defpower[#1]{#2}{#3}}
1460     {\si@unt@defpower[#1]{#2}{#3}}}
1461 \newcommand*{\providepower}[3][{}]{%
1462   \si@ifdefinable{#2}
1463     {\si@unt@defpower[#1]{#2}{#3}}
1464   {}}

\ifsi@unt@num A flag is needed to tell the processor whether there is a number, to get the correct
              spacing. The flag is true outside of the processor
1465 \newif\ifsi@unt@num\si@unt@numtrue

\si@SI       The internal processing starts with \si@SI, which processes the second optional
\si@unt@SIopts argument to \SI (which is empty for \unitsym). Everything is set up in a
              group, and processing begins by handling the options.
\si@SI[<options>]{<unit>} [ <preunit> ] {<unit>}

1466 \def\si@SI[#1]#2[#3]#4{%
1467   \begingroup
1468     \si@ifnotmtarg{#1}
1469     {\sisetup{#1}}%
1470     \edef\si@unt@SIopts{#1}}%

```

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

```

1471 \si@unt@numfalse
1472 \si@xspacefalse
1473 \si@ifnotmtarg{#3}
1474 { \si@log@debug{Prefix unit found}%
1475 \si@unt@printunit{#3}}%

```

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

```

1476 \si@ifnotmtarg{#2}
1477 { \si@log@debug{Number found in \string\SI\space argument}%
1478 \num{#2}%
1479 \si@unt@numtrue}%
1480 \si@ifnotmtarg{#4}
1481 { \si@unt@printunit{#4}}%
1482 \endgroup}

```

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

```

\si@unt@ifliteral{<text>}
1483 \newif\ifsi@unt@littest
1484 \newcommand*{\si@unt@ifliteral}[1]{%
1485 \begingroup
1486 \si@unt@littesttrue

```

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

```

1487 \setbox\si@tempbox=\hbox{\si@unt@out{#1}}%
1488 \ifdim\wd\si@tempbox>\z@\relax
1489 \endgroup\expandafter\@firstoftwo%
1490 \else
1491 \endgroup\expandafter\@secondoftwo%
1492 \fi}

```

`\ifsi@unt@litout` The printing macro uses the above test to determine how to act. It then carries out the appropriate action: either typesetting or executing. A flag is also provided so that any macro units inside a partially-literal argument will work (this is also needed to emulate `unitsdef`).

```

\si@unt@printunit{<unit>}
1493 \newif\ifsi@unt@litout
1494 \newcommand*{\si@unt@printunit}[1]{%
1495 \si@unt@ifliteral{#1}

```

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

```

1496 { \si@log@debug{Literal items found in unit argument:\MessageBreak
1497 outputting without further processing}%
1498 \si@unt@litouttrue
1499 \si@unt@addvaluesep%
1500 \si@unt@out{#1}}

```

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

```

1501     {\si@log@debug{Macro unit found:\MessageBreak
1502         processing to format output}}%
1503     \si@unt@init%
1504     \advance\si@unt@depthcnt\@ne\relax
1505     #1%
1506     \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@addvaluesep
\si@unt@addvaluesep
\si@unt@litvaluesep1507 \newcommand*{\si@unt@addvaluesep}{%
\si@unt@stackvaluesep1508 \ifsi@unt@num
1509     \expandafter\si@unt@addvaluesep%
1510     \fi}
1511 \newcommand*{\si@unt@addvaluesep}{%
1512     \ifsi@unt@litout
1513         \expandafter\si@unt@litvaluesep%
1514     \else
1515         \expandafter\si@unt@stackvaluesep%
1516     \fi}
1517 \newcommand*{\si@unt@stackvaluesep}{%
1518     \protected@edef\si@unt@spstack{\si@valuesep}}
1519 \newcommand*{\si@unt@litvaluesep}{%
1520     \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@spstack1521 \newcommand*{\si@unt@spstack}{%
\ifsi@unt@first1522 \newcommand*{\si@unt@stacka}{%
\si@unt@init1523 \newcommand*{\si@unt@stackb}{%
1524     \newcount\si@unt@unitcnta
1525     \newcount\si@unt@unitcntb
1526     \newcount\si@unt@depthcnt
1527     \newif\ifsi@unt@first
1528     \si@unt@depthcnt\m@ne\relax
1529     \newcommand*{\si@unt@init}{%
1530         \begingroup
1531             \si@unt@litoutfalse
1532             \si@unt@firsttrue
1533             \si@unt@perfalse
1534             \si@unt@perseenfalse
1535             \si@unt@prepowerfalse
1536             \si@unt@depthcnt\z@\relax
1537             \si@unt@powerdim\z@\relax
1538             \si@unt@unitcnta\z@\relax
1539             \si@unt@unitcntb\z@\relax
1540             \si@unt@prefixcnt\z@\relax
1541             \renewcommand*{\si@unt@spstack}{%
1542                 \renewcommand*{\si@unt@stacka}{%

```



```

1543 \renewcommand*\si@unt@stackb}{}%
1544 \renewcommand*\si@unt@holdstacka}{}%
1545 \renewcommand*\si@unt@holdstackb}{}%
1546 \renewcommand*\si@unt@lastadda}{space}%
1547 \renewcommand*\si@unt@lastaddb}{space}%

```

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

```

1548 \newcommand*\si@unt@final}{%
1549 \si@unt@third%
1550 \si@unt@stackout%
1551 \endgroup
1552 \ifsi@xspace
1553 \expandafter\expandafter\expandafter\xspace%
1554 \fi}

```

`\si@unt@defunit` The internal macro for defining a unit does not check for redefinition; that is done by the user macros. `\si@unt@defunit [valuesep=none] {unit} {symbol}`

```

1555 \newcommand*\si@unt@defunit[3][{}%
1556 \si@log@debug{Declaring unit \string#2 with \MessageBreak
1557 meaning \string#3}%

```

The optional argument can only have the value `valuesep=none`, which is used to prevent a space occurring between a number and the units (for example, with `\degree`). The optional argument needs to be saved; `\edef` is used so there is no issue with redefinition. The macro name is effectively “reversed” so that life is easier with the expansions here.

```

1558 \si@ifnotmtarg{#1}
1559 {\expandafter\expandafter\expandafter\def\expandafter%
1560 \csname\expandafter@gobble\string#2@opt@unt@si@endcsname{#1}}%

```

The unit macro itself is now defined. The definition simply selects the correct path for the rest of the processing to go down. To avoid needing specialised gobbling macros, the optional nature of the first argument is dropped.

```

1561 \DeclareRobustCommand*{#2}[1][{}%
1562 \ifsi@unt@littest
1563 \expandafter\si@gobblethree
1564 \else

```

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

```

1565 \ifsi@unt@litout
1566 \expandafter\expandafter\expandafter@gobbletwo
1567 \else
1568 \expandafter\expandafter\expandafter\si@unt@unit%
1569 \fi
1570 \fi
1571 {##1}{#2}{#3}}

```

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

```

1572 \long\def \si@gobblethree #1#2#3{}

```

`\si@unt@defprefix` As with units, multiples are defined by an internal macro. `\si@unt@defprefix {multiple} {powers-t}`

```

1573 \newcommand*\si@unt@defprefix[3][{}%
1574 \si@log@debug{Declaring multiple \string#1 with \MessageBreak

```

```

1575     meaning \string#3}%
1576 \DeclareRobustCommand{#1}{%
1577     \ifsi@unt@littest
1578       \expandafter\si@gobblethree
1579     \else
1580       \ifsi@unt@litout
1581         \expandafter\expandafter\expandafter\@gobbletwo
1582       \else
1583         \expandafter\expandafter\expandafter\si@unt@prefix%
1584       \fi
1585     \fi
1586     {#1}{#2}{#3}}

```

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

```
\si@unt@defpower[<post>]{<power>}{<num>}
```

```

1587 \newcommand*{\si@unt@defpower}[3][]{%
1588 \si@log@debug{Declaring power \string#2 with \MessageBreak
1589     meaning \string#3}%

```

Once again the optional argument is saved.

```

1590 \expandafter\expandafter\expandafter\edef\expandafter%
1591   \csname\expandafter\@gobble\string#2@opt@si\endcsname{#1}%
1592 \DeclareRobustCommand{#2}{%
1593   \ifsi@unt@littest
1594     \expandafter\@gobbletwo
1595   \else

```

The literal output here does not need to gobble anything, but uses `\textsuperscript` to get the correct effect. This will of course give very odd results for prefix powers.

```

1596     \ifsi@unt@litout
1597       \expandafter\expandafter\expandafter\si@unt@litpower%
1598     \else
1599       \expandafter\expandafter\expandafter\si@unt@power%
1600     \fi
1601   \fi
1602   {#2}{#3}}

```

`\si@unt@unithook` 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.

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

```

1603 \newcommand*{\si@unt@unithook}{}
1604 \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.

```

1605 \ifnum\si@unt@depthcnt=\m@ne\relax
1606   \expandafter\si@unt@init%
1607 \fi
1608 \advance\si@unt@depthcnt\@ne\relax
1609 \si@log@debug{Unit processing: level \the\si@unt@depthcnt,
1610   \MessageBreak unit \string#2}%
1611 \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.

```
1612 \si@unt@ifliteral{#3}
1613   {\si@unt@addtostack{unit}{#3}%
1614     \ifsi@unt@prepower
1615       \expandafter\si@unt@stkpower%
1616     \fi}
1617   {#3}%
```

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

```
1618 \advance\si@unt@depthcnt\m@ne\relax
1619 \ifnum\si@unt@depthcnt=\z@\relax
1620   \expandafter\si@unt@final%
1621 \fi}
```

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

```
\si@unt@firstorsecond{<num>}{<macro>}
1622 \newcommand{\si@unt@firstorsecond}[2]{%
1623   \ifsi@unt@first
1624     \expandafter\si@unt@first%
1625   \else
1626     \expandafter\si@unt@second%
1627   \fi
1628   {#1}{#2}}%
```

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

```
\si@unt@first{<num>}{<unit>}
1629 \newcommand*\si@unt@first[2]{%
1630   \si@ifnotmtarg{#1}
1631   {\num{#1}}%
1632   \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.

```
1633 \ifcsname\expandafter\@gobble\string#2@opt@unt@si\endcsname
1634   \expandafter\si@unt@setopts%
1635 \else
1636   \expandafter\@gobble
1637 \fi
1638 {#2}%
1639 \si@unt@addvaluesep%
1640 \si@unt@firstfalse}
```

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

```
1641 \newcommand*\si@unt@setopts[1]{%
1642   \expandafter\expandafter\expandafter\expandafter\expandafter%
1643   \expandafter\expandafter\si@temptoks\expandafter\expandafter%
```

```

1644 \expandafter\expandafter\expandafter\expandafter\expandafter{%
1645 \expandafter\csname\expandafter@gobble\string#1@opt@unt@si%
1646 \endcsname}%
1647 \expandafter\sisetup\expandafter{\the\si@temptoks}%
1648 \si@log@debug{Applying options ``\the\si@temptoks''
1649 for\MessageBreak unit \string#1}%

```

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

```

1650 \@ifundefined{si@unt@SIopts}{}
1651 {\ifx\@empty\si@unt@SIopts\@empty\else
1652 \expandafter\expandafter\si@unt@setSIopts%
1653 \fi}}
1654 \newcommand*{\si@unt@setSIopts}{%
1655 \expandafter\si@temptoks\expandafter{\si@unt@SIopts}%
1656 \expandafter\sisetup\expandafter{\the\si@temptoks}}

```

`\si@unt@second` 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.

```

\si@unt@third \si@unt@second{<num>}{<unit>}
1657 \newcommand{\si@unt@second}[2]{%
1658 \si@ifnotmtarg{#1}
1659 {\si@log@warn{Optional argument to unit macro\MessageBreak
1660 allowed only for outer unit}}%
1661 \si@unt@third%
1662 \si@unt@addtostack{space}{\ensuremath{\si@unitsep}}}}
1663 \newcommand*{\si@unt@third}{%
1664 \ifsi@unt@prepower\else
1665 \expandafter\si@unt@stkpower%
1666 \fi

```

`\si@tempa` 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`.

```

1667 \def\si@tempa{prefix}%
1668 \expandafter\ifx\csname si@unt@lastadd\si@unt@checkstack%
1669 \endcsname\si@tempa\else
1670 \expandafter\si@unt@spacecheck%
1671 \fi
1672 \ifsi@unt@per
1673 \expandafter\si@unt@perseenttrue
1674 \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.

```

1675 \newcommand*{\si@unt@spacecheck}{%
1676 \def\si@tempa{space}%
1677 \expandafter\ifx\csname si@unt@lastadd\si@unt@checkstack%
1678 \endcsname\si@tempa\else
1679 \expandafter\si@unt@reciptest%
1680 \fi}

```

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

```

\si@unt@prefix{<multiple>}{<powers-ten>}{<symbol>}

```

```

1681 \newcommand*{\si@unt@prefix}[3]{%
1682   \si@unt@firstorsecond}{#1}%
1683   \ifsi@prefixnum
1684     \expandafter\si@unt@countprefix%
1685   \else
1686     \expandafter\si@unt@addprefix%
1687   \fi
1688   {#2}{#3}}

```

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

```

\si@unt@countprefix{\gobble}{\symbol}
1689 \newcommand*{\si@unt@addprefix}[2]{\si@unt@addtostack{prefix}{#2}}

```

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

```

\si@unt@countprefix \si@unt@countprefix{\powers-ten}{\gobble}

```

`\si@unt@invprefix` `\newcount\si@unt@prefixcnt`

```

1690 \newcommand*{\si@unt@countprefix}[2]{%
1691   \si@tempcnta#1\relax
1692   \ifsi@unt@per
1693     \si@unt@invprefix%
1694   \fi
1695   \advance\si@unt@prefixcnt\si@tempcnta\relax}
1696 \newcommand*{\si@unt@invprefix}{%
1697   \si@tempcntb\si@tempcnta\relax
1698   \si@tempcnta-\si@tempcntb\relax}
1699

```

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

```

\si@unt@litpower{\gobble}{\num}
1700 \newcommand*{\si@unt@litpower}[2]{\textsuperscript{#2}}

```

`\ifsi@unt@prepower` The handling of powers starts by checking if the number needs to be reversed.

```

\si@unt@power \si@unt@power{\power}{\num}

```

```

1701 \newif\ifsi@unt@prepower
1702 \newcommand*{\si@unt@power}[2]{%
1703   \si@unt@powerdim #2 pt\relax
1704   \ifsi@frac\else
1705     \ifsi@unt@per
1706       \expandafter\expandafter\expandafter\si@unt@invpower%
1707     \fi
1708   \fi
1709   \def\si@tempa{post}%
1710   \si@unt@prepowertrue
1711   \expandafter\expandafter\expandafter\ifx\expandafter%
1712     \csname\expandafter\@gobble\string#1\opt@si\endcsname\si@tempa
1713   \expandafter\si@unt@stackpower%
1714   \fi}

```

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

```

1715 \newdimen\si@unt@powerdim

```

`\si@unt@stackpower` Adding powers to the stack should also clear the power list. If the number is already zero, then of course nothing happens.

```
\si@unt@stkpower 1716 \newcommand*{\si@unt@stackpower}{%
1717 \si@unt@prepowerfalse
```

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

```
1718 \si@unt@stkpower%
1719 \si@unt@perfalse
1720 \si@unt@perseenfalse}
```

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

```
1721 \newcommand*{\si@unt@stkpower}{%
1722 \ifdim\si@unt@powerdim=\m@ne pt\relax
1723 \ifsi@frac\else
1724 \expandafter\expandafter\expandafter\si@unt@stkpwr%
1725 \fi
1726 \else
1727 \expandafter\si@unt@stkpwr%
1728 \fi}
```

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

```
1729 \newcommand*{\si@unt@stkpwr}{%
1730 \ifdim\si@unt@powerdim=\z@\relax\else
1731 \edef\si@tempa{unit}%
1732 \expandafter\ifx\csname si@unt@lastadd\si@unt@checkstack%
1733 \endcsname\si@tempa
1734 \si@unt@addtostack{power}{^{\num{\strip@pt\si@unt@powerdim}}}%
1735 \fi
1736 \fi
1737 \si@unt@powerdim\z@\relax}
```

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

```
1738 \newcommand*{\si@unt@invpower}{%
1739 \si@tempdima\si@unt@powerdim\relax
1740 \si@unt@powerdim -\si@tempdima\relax
```

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

```
1741 \ifdim\si@unt@powerdim=\p@\relax
1742 \si@unt@powerdim\z@\relax
1743 \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 than set arbitrarily. The second flag is needed so that `\per` can give powers of -1 properly.
`\si@per`

```
1744 \newif\ifsi@unt@per
1745 \newif\ifsi@unt@perseen
1746 \DeclareRobustCommand*\per{\si@per}
1747 \newcommand*{\si@per}{%
1748 \si@unt@firstorsecond}{\per}%
1749 \ifsi@unt@per
1750 \expandafter\si@unt@perfalse
1751 \else
1752 \expandafter\si@unt@pertrue
1753 \fi}
```

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

```

1754 \newcommand*\si@unt@reciptest}{%
1755   \ifsi@unt@per
1756     \ifsi@unt@perseen
1757       \expandafter\expandafter\expandafter\si@unt@recip%
1758     \fi
1759   \fi}
1760 \newcommand*\si@unt@recip}{%
1761   \si@unt@powerdim\m@ne pt\relax
1762   \si@unt@stackpower}

```

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

```

\si@unt@lastaddb \si@unt@addtostack \si@tempa \si@unt@addtostack{<type>}{<token>}
\si@tempb1763 \newcommand*\si@unt@lastadda}{%
1764 \newcommand*\si@unt@lastaddb}{%
1765 \newcommand*\si@unt@addtostack}[2]{%
1766   \edef\si@tempa{#1}%

```

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

```

1767   \expandafter\ifx\csname si@unt@lastadd\si@unt@checkstack\endcsname%
1768     \si@tempa
1769     \expandafter\@gobbletwo
1770   \else
1771     \expandafter\si@unt@preplussp%
1772   \fi
1773   {#1}{#2}}

```

`\si@unt@preplussp` The space added after a prefix needs to be ignored. `\si@unt@prespace{<type>}{<stack>}{<token>}{<g`

```

\si@tempa1774 \newcommand*\si@unt@preplussp}[2]{%
\si@tempb1775   \def\si@tempa{prefix+space}%
1776   \edef\si@tempb{\csname si@unt@lastadd\si@unt@checkstack%
1777     \endcsname+#1}%
1778   \ifx\si@tempa\si@tempb
1779     \expandafter\@gobbletwo
1780   \else
1781     \expandafter\si@unt@stack%
1782   \fi
1783   {#1}{#2}}

```

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

```

\si@unt@stack{<type>}{<tokens>}
1784 \newcommand*\si@unt@stack}[2]{%
1785   \expandafter\edef\csname si@unt@lastadd\si@unt@checkstack%
1786     \endcsname{#1}%

```

`\si@tempa` A count is kept of the number of *units* added to each stack.

```

\si@tempb1787 \edef\si@tempa{#1}%
1788 \def\si@tempb{unit}%
1789 \ifx\si@tempa\si@tempb

```

```

1790 \expandafter\si@unt@incnt%
1791 \fi

```

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

```

1792 \def\si@tempb{space}%
1793 \ifx\si@tempa\si@tempb
1794 \expandafter\si@unt@holdspace%
1795 \else
1796 \expandafter\si@unt@addstack%
1797 \fi
1798 {#2}}

```

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

```

1799 \newcommand*\si@unt@incnt}{%
1800 \expandafter\advance\csname si@unt@unitcnt\si@unt@checkstack%
1801 \endcsname\@ne\relax}

```

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

```

\si@unt@addstack \si@unt@holdspace{<tokens>}
\si@unt@holdstacka \si@unt@addstack{<tokens>}
\si@unt@holdstackb

```

```

1802 \newcommand*\si@unt@holdstacka}{%
1803 \newcommand*\si@unt@holdstackb}{%
1804 \newcommand*\si@unt@holdspace}[1]{%
1805 \expandafter\protected@edef\csname si@unt@holdstack%
1806 \si@unt@checkstack\endcsname{#1}}
1807 \newcommand*\si@unt@addstack}[1]{%
1808 \expandafter\protected@edef\csname si@unt@stack%
1809 \si@unt@checkstack\endcsname%
1810 {\csname si@unt@stack\si@unt@checkstack\endcsname%
1811 \csname si@unt@holdstack\si@unt@checkstack\endcsname#1}%
1812 \expandafter\renewcommand\expandafter*\expandafter{%
1813 \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.

```

1814 \newcommand*\si@unt@stackout}{%
1815 \ifsi@frac
1816 \expandafter\si@unt@fracout%
1817 \else
1818 \expandafter\si@unt@normout%
1819 \fi}

```

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

```

1820 \newcommand*\si@unt@checkstack}{%
1821 \ifsi@frac
1822 \ifsi@unt@per
1823 \expandafter\expandafter\expandafter b%
1824 \else
1825 \expandafter\expandafter\expandafter a%
1826 \fi
1827 \else
1828 \expandafter a%
1829 \fi}

```


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

```
1830 \newcommand*{\si@unt@spaceout}{%
1831   \ensuremath{\si@unt@spstack}}
```

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

```
1832 \newcommand*{\si@unt@prefixout}{%
1833   \ifnum\si@unt@prefixcnt=\z@\relax\else
1834     \ifsi@unt@num
1835       \si@out@text{\ensuremath{\{\}\si@prefixproduct{\}}}%
1836     \fi
1837     \let\si@exppower\si@prefixpower
1838     \num{e\the\si@unt@prefixcnt}%
1839   \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.

```
1840 \newcommand*{\si@unt@normout}{%
1841   \si@unt@prefixout%
1842   \si@unt@spaceout%
1843   \expandafter\si@unt@out\expandafter{\si@unt@stacka}}
```

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

```
1844 \newcommand*{\si@unt@fracout}{%
1845   \si@unt@notambig%
1846   \ifx\@empty\si@unt@stacka\@empty
1847     \ifx\@empty\si@unt@stackb\@empty
1848       \ifsi@unt@litout\else
1849         \si@log@err{Empty fractional unit}{The unit
1850           argument\MessageBreak given does not contain any
1851           symbols}%
1852       \fi
1853     \else
```

With an empty numerator, no space is added

```
1854     \ifsi@slash
1855       \si@unt@prefixout%
1856       \si@frac{\}\si@unt@stackb}%
1857     \else
1858       \si@unt@prefixout%
1859       \si@unt@spaceout%
1860       \si@frac{1}\si@unt@stackb}%
1861     \fi
1862   \fi
1863 \else
```

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

```
1864   \ifx\@empty\si@unt@stackb\@empty
1865     \si@unt@normout%
1866   \else
1867     \si@unt@prefixout%
1868     \si@unt@spaceout%
1869     \si@frac{\si@unt@stacka}\si@unt@stackb}%
1870   \fi
1871 \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.

```

1872 \newcommand*{\si@unt@notambig}{%
1873   \ifnum\si@unt@unitcntb>\@ne\relax
1874     \ifsi@slash
1875       \expandafter\expandafter\expandafter\si@unt@notabg%
1876     \fi
1877   \fi}
1878 \newcommand*{\si@unt@notabg}{%
1879   \protected@edef\si@unt@stackb{\si@denlbrac\si@unt@stackb%
1880     \si@denrbrac}}

```

`\si@unt@out` The final part of the units system is the output routine. This has to cope with units not only as macros but also as direct input (`sistyle`-type input). Non-Latin characters are also handled cleanly. As usual, `\scantokens` is used to make life easier.

```

\si@unt@out{\langle unit \rangle}
1881 \begingroup
1882   \catcode`\~=\active
1883   \catcode`\.=\active
1884   \gdef\si@unt@out#1{%
1885     \begingroup
1886       \si@unt@nonlatin%
1887       \makeatletter%
1888       \catcode`\~=\active
1889       \catcode`\.=\active
1890       \def~{\ensuremath{\si@unitspace}}%
1891       \def.\{\ensuremath{\si@unitsep}}%
1892       \scantokens{\si@out@text{#1}\@empty}%
1893     \endgroup}
1894 \endgroup

```

`\si@unt@nonlatin` To handle non-Latin symbols in the input, a single macro is provided. Initially, it does nothing

```

1895 \newcommand*{\si@unt@nonlatin}{\relax}

```

The meaning of different characters depends on the encoding used. Thus a test is made for the presence of a suitable package *and* the encoding in use. The various characters can then be handled.

`\si@tempa`

```

1896 \AtBeginDocument{%
1897   \@ifpackageloaded{inputenc}
1898     {\def\si@tempa{latin1}}%
1899     \ifx\inputencodingname\si@tempa

```

The degree symbol is character 176 and the micro symbol is character 181 in `latin1`.

```

1900       \si@unt@sym{176}{\si@sym@degree}%
1901       \si@unt@sym{181}{\si@sym@mu}%
1902       \si@unt@sym{229}{\si@sym@ringA}%
1903     \fi}

```

No `inputenc` available.

```

1904     {}

```

`\si@unt@sym` A macro for declaring symbols: a copy of `\DeclareInputMath` from `inputenc`.

```
\si@unt@sym{charcode}
1905 \newcommand*{\si@unt@sym}[1]{%
1906   \bgroup
1907     \uccode`~#1%
1908     \uppercase{%
1909       \egroup
1910       \def~}}
```

`\kilogram` With the system set up, the basic unit macros are implemented. The only units defined whatever options are given are the base SI units.

```
\metre
\mole1911 \newunit{\kilogram}{kg}
\kelvin1912 \newunit{\metre}{m}
\candela1913 \newunit{\mole}{mol}
\second1914 \newunit{\second}{s}
\ampere1915 \newunit{\ampere}{A}
1916 \newunit{\kelvin}{K}
1917 \newunit{\candela}{cd}
```

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

```
\cubic1918 \newpower{\Square}{2}
\cubed1919 \newpower[post]{\squared}{2}
1920 \newpower{\cubic}{3}
1921 \newpower[post]{\cubed}{3}
```

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

```
\tothe@opt@si
\tothe{num}
1922 \newcommand*{\tothe}[1]{%
1923   \ifsi@unt@littest
1924     \expandafter\@gobbletwo
1925   \else
1926     \ifsi@unt@litout
1927       \expandafter\expandafter\expandafter\si@unt@litpower%
1928     \else
1929       \expandafter\expandafter\expandafter\si@unt@power%
1930     \fi
1931   \fi
1932   {\tothe}{#1}}
1933 \newcommand*{\tothe@opt@si}[post]
```

16.13 Locales

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

```
\si@loc@ssetup
\si@loc@load{locale}
1934 \newcommand*{\si@loc@load}[1]{%
1935   \let\si@loc@ssetup\ssetup
1936   \renewcommand*{\ssetup}[1]{%
1937     \expandafter\def\csname si@loc@#1\endcsname{##1}}
1938   \si@loadfile{#1}%
1939   \let\ssetup\si@loc@ssetup}
```

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

```
\si@loc@set{\langle locale \rangle}
1940 \newcommand*{\si@loc@set}[1]{%
1941   \ifcsname si@loc@#1\endcsname
1942     \si@log@inf{Setting locale to '#1'}%
1943     \expandafter\expandafter\expandafter\expandafter\expandafter%
1944       \expandafter\expandafter\si@temptoks\expandafter\expandafter%
1945       \expandafter\expandafter\expandafter\expandafter\expandafter{%
1946         \expandafter\csname si@loc@#1\endcsname}%
1947     \expandafter\ssetup\expandafter{\the\si@temptoks}%
1948     \ifcsname si@loc@#1@extra\endcsname
1949       \csname si@loc@#1@extra\endcsname%
1950     \fi
1951   \else
1952     \ifcsname si@loc@#1@extra\endcsname
1953       \si@log@inf{Setting locale to '#1'}%
1954       \csname si@loc@#1@extra\endcsname%
1955     \else
1956       \si@log@warn{Unknown locale '#1'}%
1957     \fi
1958   \fi}
```

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

```
\addtolocale{\langle locale \rangle}{\langle commands \rangle}
1959 \newcommand*{\addtolocale}[2]{\si@addtocname{si@loc@#1@extra}{#2}}
```

16.14 Output routine

`\si@out@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.

```
\si@out@text{\langle text \rangle}
1960 \begingroup
1961   \catcode`\^=\active\relax
1962   \catcode`\-=\active\relax
1963   \gdef\si@out@text#1{%
1964     \begingroup
1965       \catcode`\^=\active\relax%
1966       \makeatletter%
```

The various font families can now be set up. First a check is made in case there are nested calls to `\si@out@text`.

```
1967     \ifsi@fam@set\else
1968       \expandafter\si@fam@set%
1969     \fi
1970     \text{\si@fam@italic\si@fam@bold\si@fam@text%
```

In text mode, `^` will execute `\textsuperscript`, whereas in maths mode it will be converted to `\sp`, which the kernel defines as `^` with catcode 7. `\scantokens` is used to set the catcodes here, plus any others that have been set by other parts of the package.

```
1971     \ifsi@textmode
```

```

1972     \let^\textsuperscript
1973     \catcode'\-=\active\relax%
1974     \let-\si@out@minus

```

The \@empty is needed here to mop up any extra space.

```

1975     \scantokens{#1\@empty}%
1976     \else
1977     \let^\sp
1978     \let\textsuperscript\sp
1979     $\si@fam@maths{\scantokens{#1}}$%
1980     \fi}%

```

Everything is done; a bit of tidying up is needed.

```

1981     \endgroup
1982     \check@mathfonts}
1983 \endgroup

```

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

```

1984 \newcommand*{\si@out@minus}{\ensuremath{-}}

```

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

```

\si@out@num{<num>}
1985 \newcommand*{\si@out@num}[1]{%
1986   \begingroup
1987   \sisetup{%
1988     textdefault=\si@textnumdefault,%
1989     mathdefault=\si@mathnumdefault}%
1990   \si@out@text{#1}%
1991   \endgroup}

```

16.15 Finalisation

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

```

1992 \sisetup{
1993   unitsep=thin,
1994   valuesep=thin,
1995   decimalsign=fullstop,
1996   digitsep=thin,
1997   obeybold=false,
1998   inlinebold=text,
1999   obeyitalic=false,
2000   sign=plus,
2001   addsign=none,
2002   obeymode=false,
2003   mode=maths,
2004   padangle=small,
2005   padnumber=lead,
2006   sepfour=false,
2007   tabformat=-1,
2008   xspace=false,
2009   per=reciprocal,
2010   slash=slash,

```

```

2011 mathsdefault=\si@mathsrn,
2012 mathsrn=mathrm,
2013 mathssf=mathsf,
2014 mathstt=mathtt,
2015 textdefault=\si@textrn,
2016 textrn=rnfamily,
2017 textsf=sffamily,
2018 texttt=ttfamily,
2019 mathsnumdefault=\si@mathsrn,
2020 textnumdefault=\si@textrn,
2021 numlist=0123456789,
2022 numdecimal={.,},
2023 numexp=eEdD,
2024 numgobble={},
2025 numsign=+-\pm\mp,
2026 numextra=(),
2027 redefsymbols=true,
2028 load=default,
2029 noload={},
2030 eVcorra={0.3ex},
2031 eVcorrb={0ex},
2032 denlbrac=(,
2033 denrbrac=),
2034 astroang=false,
2035 loadlocales={},
2036 loctolang={},
2037 prefix=symbol,
2038 prefixpower=ten,
2039 prefixproduct=times}
2040 \ProcessOptionsX[si]<opt>

```

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

```

2041 \newcommand*\si@extension}{cfg}
2042 \newcommand*\si@fileprefix}{si-}

```

`\si@ifloaded` A bit of borrowing from the L^AT_EX kernel.

```

\si@ifloaded{<package>}
2043 \newcommand*\si@ifloaded}[1]{%
2044 \@ifl@aded\si@extension{\si@fileprefix#1}}

```

`\si@loadfile` Loading configuration files is handled here.

```

\si@loadfile{<file>}
2045 \newcommand*\si@loadfile}[1]{%
2046 \si@ifloaded{#1}{%
2047 \InputIfFileExists{\si@fileprefix#1.\si@extension}
2048 {}
2049 {\si@log@err{Failed to load file
2050 \si@fileprefix#1.\si@extension}
2051 {The configuration file requested could not be found}}}}

```

`\si@requirecfgs` The configuration files depend on each other.

```

\si@tempb \si@requirecfgs{<cfg-file>}

```

```

2052 \newcommand*{\si@requirecfgs}[1]{%
2053   \@for\si@tempb:=#1\do{\si@loadfile{\si@tempb}}}

```

`\si@loademfile` For emulation files, an additional check is made.

```

\si@loademfile{\si@loademfile{<file>}}
2054 \newcommand*{\si@loademfile}[1]{%
2055   \@ifpackageloaded{#1}
2056     {\si@log@err{Emulation clash for package `#1'}
2057      {You have asked for emulation of package `#1'\MessageBreak
2058       (perhaps by giving si a back-compatibility
2059       option)\MessageBreak but the package is already loaded!}}
2060     {\si@loadfile{#1}}}}

```

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

```

2061 \newcommand*{\si@emclash}[2]{%
2062   \si@log@err{Emulation clash: `#1' and `#2'}
2063   {You have asked for emulation of package `#1'\MessageBreak
2064    but have already loaded emulation of `#2'}}

```

`\si@tempa` A check is now made so that emulation takes place one file at a time, and that each file is loaded only once.

```

2065 \ifx\@empty\si@emulate\@empty\else
2066   \@for\si@tempa:=\si@emulate\do{%
2067     \si@loademfile{\si@tempa}}
2068 \fi

```

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

```

\si@tempa2069 \newcommand*{\si@expanddefault}[2]{%
\si@tempb2070 \expandafter\ifx\expandafter\@empty\csname si@#1\endcsname\@empty
\si@tempc2071 \else
2072   \def\si@tempb{default}
2073   \def\si@tempc{}
2074   \expandafter\@for\expandafter\si@tempa\expandafter:\expandafter
2075     =\csname si@#1\endcsname\do{%
2076     \ifx\si@tempa\si@tempb
2077       \si@addtolist{\si@tempc}{#2}
2078     \else
2079       \si@addtolist{\si@tempc}{\si@tempa}
2080     \fi}
2081   \expandafter\edef\csname si@#1\endcsname{\si@tempc}
2082   \expandafter\si@addtolist\expandafter{\csname si@no#1\endcsname}
2083   {default}
2084   \def\si@tempc{}
2085   \expandafter\@for\expandafter\si@tempa\expandafter:\expandafter
2086     =\csname si@#1\endcsname\do{%
2087     \si@switchfalse
2088     \expandafter\@for\expandafter\si@tempb\expandafter:\expandafter
2089       =\csname si@no#1\endcsname\do{%
2090       \ifx\si@tempa\si@tempb
2091         \si@switchtrue
2092       \fi
2093       \ifsi@switch\else
2094         \si@addtolist{\si@tempc}{\si@tempa}

```

```

2095     \fi}}
2096   \@for\si@tempa:=\si@tempc\do{%
2097     \si@loadfile{\si@tempa}}
2098 \fi}

```

The configuration and abbreviation files are loaded.

```

2099 \si@expanddefault{load}{prefix,named,addn,prefixed,accepted,%
2100   physical,abbr}

```

\si@tempa The locale files are loaded; here there is a need to check on both loadlocales
\si@tempb and loctolang.

```

2101 \ifx\@empty\si@loadlocales\@empty\else
2102   \@for\si@tempa:=\si@loadlocales\do{%
2103     \si@loc@load{\si@tempa}}
2104 \fi

```

For loctolang.

```

2105 \ifx\@empty\si@loctolang\@empty\else
2106   \def\si@tempa#1:#2\@nil{\si@loc@load{#1}}
2107   \@for\si@tempb:=\si@loctolang\do{%
2108     \expandafter\si@tempa\si@tempb:\@nil}
2109   \AtBeginDocument{%
2110     \ifpackage{babel}
2111       {\def\si@tempa#1:#2:#3\@nil{%
2112         \expandafter\addto\expandafter{\csname extras#2\endcsname}%
2113         {\si@loc@set{#1}}}%
2114         \@for\si@tempb:=\si@loctolang\do{%
2115           \expandafter\si@tempa\si@tempb:\@nil}}
2116       {\si@log@warn{babel not loaded - option\MessageBreak
2117         loctolang ignored}}
2118 \fi

```

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

```

2119 \IfFileExists{si.cfg}
2120   {\si@log@inf{Local configuration file found}%
2121     \InputIfFileExists{si.cfg}{}{}}
2122 {}

```

17 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).

17.1 Multiple prefixes

\yocto The various SI multiple prefixes are defined here. First the small powers.

```

\zepto2123 \ProvidesFile{si-prefix.cfg}
\atto2124 [2008/02/20 v.06a SI Multiple prefixes]
\femto2125 \newprefix{\yocto}{-24}{y}
\pico2126 \newprefix{\zepto}{-21}{z}
\nano2127 \newprefix{\atto}{-18}{a}
\micro
\Micro
\milli
\centi
\deci

```



```

2128 \newprefix{\femto}{-15}{f}
2129 \newprefix{\pico}{-12}{p}
2130 \newprefix{\nano}{-9}{n}

```

Some testing is needed for unitsdef compatibility.

```

2131 \ifsi@old@OHM
2132 \newprefix{\Micro}{-6}{\si@sym@mu}
2133 \else
2134 \ifsi@gensymb\else
2135 \newprefix{\micro}{-6}{\si@sym@mu}
2136 \fi
2137 \fi
2138 \newprefix{\milli}{-3}{m}
2139 \newprefix{\centi}{-2}{c}
2140 \newprefix{\deci}{-1}{d}

```

`\deca` Now the large ones.

```

\hecto2141 \newprefix{\deca}{1}{da}
\kilo2142 \newprefix{\hecto}{2}{h}
\mega2143 \newprefix{\kilo}{3}{k}
\giga2144 \newprefix{\mega}{6}{M}
\tera2145 \newprefix{\giga}{9}{G}
\peta2146 \newprefix{\tera}{12}{T}
\exa2147 \newprefix{\peta}{15}{P}
\zetta2148 \newprefix{\exa}{18}{E}
\yotta2149 \newprefix{\zetta}{21}{Z}
2150 \newprefix{\yotta}{24}{Y}

```

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

```

2151 \newprefix{\deka}{1}{da}

```

`\gram` As the base unit of mass is the kilogram, rather than the gram, a bit of extra work is needed; by default the package only defines `\kilogram`, but with the prefixes available, this is altered to be `\kilo\gram`. For that, the `\gram` must be defined first.

```

2152 \newunit{\gram}{g}
2153 \renewunit{\kilogram}{\kilo\gram}

```

17.2 Derived units with specific names

`\becquerel` Derived units with specific names and symbols are defined. Litre is an awkward one, but here the UK standard is used.

```

\coulomb
\farad2154 \ProvidesFile{si-named.cfg}
\Gray2155 [2008/02/20 v.06a SI Named units]
\hertz2156 \newunit{\becquerel}{Bq}
\henry2157 \newunit{\coulomb}{C}
\joule2158 \newunit{\farad}{F}
\katal2159 \newunit{\Gray}{Gy}
\lumen2160 \newunit{\hertz}{Hz}
\lux2161 \newunit{\henry}{H}
\newton2162 \newunit{\joule}{J}
2163 \newunit{\katal}{kat}
2164 \newunit{\lumen}{lm}

```

```

2165 \newunit{\lux}{lx}
2166 \newunit{\newton}{N}

\ohm Some testing is needed for unitsdef compatibility.
\Ohm2167 \ifsi@old@OHM
\pascal2168 \newunit{\Ohm}{\si@sym@Omega}
\siemens2169 \else
\sievert2170 \ifsi@gensymb\else
\tesla To be on the safe side, use \provideunit.
\volt2171 \provideunit{\ohm}{\si@sym@Omega}
\watt2172 \fi
\weber2173 \fi
2174 \newunit{\pascal}{Pa}
2175 \newunit{\siemens}{S}
2176 \newunit{\sievert}{Sv}
2177 \newunit{\tesla}{T}
2178 \newunit{\volt}{V}
2179 \newunit{\watt}{W}
2180 \newunit{\weber}{Wb}

\celsius The degree celsius is a named unit.
\Celsius2181 \ifsi@old@OHM
2182 \newunit{\Celsius}{\si@sym@celsius}
2183 \else
2184 \ifsi@gensymb\else
2185 \newunit{\celsius}{\si@sym@celsius}
2186 \fi
2187 \fi

\radian The radian and steradian are officially derived units.
\steradian2188 \newunit{\radian}{rad}
2189 \newunit{\steradian}{sr}

```

17.3 Units with prefixes

As in `unitsdef`, some commonly used prefixed units are set up. This requires `si-prefix.cfg` and `si-named.cfg`.

```

2190 \ProvidesFile{si-prefixed.cfg}
2191 [2008/02/20 v.06a SI Prefixed units]
2192 \si@requirecfgs{prefix,named,accepted,physical}

\picometre Extra distances.
\nanometre2193 \newunit{\picometre}{\pico\metre}
\micrometre2194 \newunit{\nanometre}{\nano\metre}
\millimetre2195 \newunit{\micrometre}{\micro\metre}
\centimetre2196 \newunit{\millimetre}{\milli\metre}
\decimetre2197 \newunit{\centimetre}{\centi\metre}
\kilometre2198 \newunit{\decimetre}{\deci\metre}
2199 \newunit{\kilometre}{\kilo\metre}

\femtogram Extra masses.
\picogram2200 \newunit{\femtogram}{\femto\gram}
\nanogram
\microgram
\milligram

```

```

2201 \newunit{\picogram}{\pico\gram}
2202 \newunit{\nanogram}{\nano\gram}
2203 \newunit{\microgram}{\micro\gram}
2204 \newunit{\milligram}{\milli\gram}

\femtomole Now some moles.
\picomole2205 \newunit{\femtomole}{\femto\mole}
\nanomole2206 \newunit{\picomole}{\pico\mole}
\micromole2207 \newunit{\nanomole}{\nano\mole}
\millimole2208 \newunit{\micromole}{\micro\mole}
2209 \newunit{\millimole}{\milli\mole}

\attosecond Prefixed seconds.
\femtosecond2210 \newunit{\attosecond}{\atto\second}
\picosecond2211 \newunit{\femtosecond}{\femto\second}
\nanosecond2212 \newunit{\picosecond}{\pico\second}
\microsecond2213 \newunit{\nanosecond}{\nano\second}
\millisecond2214 \newunit{\microsecond}{\micro\second}
2215 \newunit{\millisecond}{\milli\second}

\picoampere The last prefixed base units are amperes.
\nanoampere2216 \newunit{\picoampere}{\pico\ampere}
\microampere2217 \newunit{\nanoampere}{\nano\ampere}
\milliampere2218 \newunit{\microampere}{\micro\ampere}
\kiloampere2219 \newunit{\milliampere}{\milli\ampere}
2220 \newunit{\kiloampere}{\kilo\ampere}

\millivolt More electricity-related units.
\kilovolt2221 \newunit{\millivolt}{\milli\volt}
\milliwatt2222 \newunit{\kilovolt}{\nano\volt}
\kilowatt2223 \newunit{\milliwatt}{\milli\watt}
\megawatt2224 \newunit{\kilowatt}{\kilo\watt}
\femtofarad2225 \newunit{\megawatt}{\mega\watt}
\picofarad2226 \newunit{\femtofarad}{\femto\farad}
\picofarad2227 \newunit{\picofarad}{\pico\farad}
\nanofarad2228 \newunit{\nanofarad}{\nano\farad}
\microfarad2229 \newunit{\microfarad}{\micro\farad}
\millifarad2230 \newunit{\millifarad}{\milli\farad}
\millisiemens2231 \newunit{\millisiemens}{\milli\siemens}

\kiloohm For resistance, checks are needed again for the definition of \ohm.
\megaohm2232 \ifsi@old@OHM
\gigaohm2233 \newunit{\kiloohm}{\kilo\Ohm}
2234 \newunit{\megaohm}{\mega\Ohm}
2235 \newunit{\gigaohm}{\giga\Ohm}
2236 \else
2237 \ifsi@gensymb\else
2238 \newunit{\kiloohm}{\kilo\ohm}
2239 \newunit{\megaohm}{\mega\ohm}
2240 \newunit{\gigaohm}{\giga\ohm}
2241 \fi
2242 \fi

```

```

\microlitre    Volumes (unlike unitsdef, with litre and metre spelled correctly). Only
\millilitre    \millilitre and \microlitre are defined as they are the two officially-
\cubicmetre    sanctioned prefixes for the litre.
\cubiccentimetre2243 \newunit{\microlitre}{\micro\litre}
\centimetrecubed2244 \newunit{\millilitre}{\milli\litre}
\cubicmicrometre2245 \newunit{\cubicmetre}{\metre\cubed}
\cubicmillimetre2246 \newunit{\cubiccentimetre}{\centi\metre\cubed}
\cubicdecimetre2247 \newunit{\centimetrecubed}{\centi\metre\cubed}
                2248 \newunit{\cubicmicrometre}{\micro\metre\cubed}
                2249 \newunit{\cubicmillimetre}{\milli\metre\cubed}
                2250 \newunit{\cubicdecimetre}{\cubic\deci\metre}

\squaremetre    Areas, with metre spelled corrected; \are and \hectare are in the “temporarily
\squarecentimetre    accepted” file.
\squarekilometre2251 \newunit{\squaremetre}{\Square\metre}
                2252 \newunit{\squarecentimetre}{\Square\centi\metre}
                2253 \newunit{\squarekilometre}{\Square\kilo\metre}

\millijoule    Some energy is needed by now!
\kilojoule2254 \newunit{\millijoule}{\milli\joule}
\megajoule2255 \newunit{\kilojoule}{\kilo\joule}
\millielelectronvolt2256 \newunit{\megajoule}{\mega\joule}
\kiloelectronvolt2257 \newunit{\millielelectronvolt}{\milli\electronvolt}
\megaelectronvolt2258 \newunit{\kiloelectronvolt}{\kilo\electronvolt}
\gigaelectronvolt2259 \newunit{\megaelectronvolt}{\mega\electronvolt}
\teraelectronvolt2260 \newunit{\gigaelectronvolt}{\giga\electronvolt}
                2261 \newunit{\teraelectronvolt}{\tera\electronvolt}

\millihertz    Frequencies.
\kilohertz2262 \newunit{\millihertz}{\milli\hertz}
\megahertz2263 \newunit{\kilohertz}{\kilo\hertz}
\gigahertz2264 \newunit{\megahertz}{\mega\hertz}
\terahertz2265 \newunit{\gigahertz}{\giga\hertz}
                2266 \newunit{\terahertz}{\tera\hertz}

\millinewton    A few more from various areas.
\kilonewton2267 \newunit{\millinewton}{\milli\newton}
\hectopascal2268 \newunit{\kilonewton}{\kilo\newton}
\megabecquerel2269 \newunit{\hectopascal}{\hecto\pascal}
\millisievert2270 \newunit{\megabecquerel}{\mega\becquerel}
                2271 \newunit{\millisievert}{\milli\sievert}

```

17.4 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.
\micA2272 \ProvidesFile{si-abbr.cfg}
\mA2273    [2008/02/20 v.06a Abbreviated units]
\kA2274 \si@requirecfgs{prefix,named,accepted,physical}
                2275 \newunit{\pA}{\pico\ampere}
                2276 \newunit{\nA}{\nano\ampere}
                2277 \newunit{\micA}{\micro\ampere}

```

```
2278 \newunit{\mA}{\milli\ampere}
2279 \newunit{\kA}{\kilo\ampere}
```

\Hz Then frequencies.

```
\mHz2280 \newunit{\Hz}{\hertz}
\kHz2281 \newunit{\mHz}{\milli\hertz}
\MHz2282 \newunit{\kHz}{\kilo\hertz}
\GHz2283 \newunit{\MHz}{\mega\hertz}
\THz2284 \newunit{\GHz}{\giga\hertz}
2285 \newunit{\THz}{\tera\hertz}
```

\fmol Amounts of substance.

```
\pmol2286 \newunit{\fmol}{\femto\mole}
\nmol2287 \newunit{\pmol}{\pico\mole}
\micmol2288 \newunit{\nmol}{\nano\mole}
\mmol2289 \newunit{\micmol}{\micro\mole}
2290 \newunit{\mmol}{\milli\mole}
```

\kV Potentials.

```
\mV2291 \newunit{\kV}{\kilo\volt}
2292 \newunit{\mV}{\milli\volt}
```

\ml Volumes.

```
\micl2293 \newunit{\ml}{\milli\litre}
\cmcl2294 \newunit{\micl}{\micro\litre}
\dmc2295 \newunit{\cmcl}{\centi\metre\cubed}
2296 \newunit{\dmc}{\deci\metre\cubed}
```

\kg Masses.

```
\fg2297 \newunit{\kg}{\kilo\gram}
```

\pg There is a name clash with babel here in French; hopefully there will not be too many complaints.

```
\nanog
\micg2298 \provideunit{\fg}{\femto\gram}
\pg2299 \newunit{\pg}{\pico\gram}
\amu2300 \newunit{\nanog}{\nano\gram}
2301 \newunit{\micg}{\micro\gram}
2302 \newunit{\mg}{\milli\gram}
2303 \newunit{\amu}{\atomicmass}
```

\kJ Energies.

```
\eV2304 \newunit{\kJ}{\kilo\joule}
\meV2305 \newunit{\eV}{\electronvolt}
\keV2306 \newunit{\meV}{\milli\electronvolt}
\MeV2307 \newunit{\keV}{\kilo\electronvolt}
\GeV2308 \newunit{\MeV}{\mega\electronvolt}
\TeV2309 \newunit{\GeV}{\giga\electronvolt}
2310 \newunit{\TeV}{\tera\electronvolt}
```

\picom Lengths.

```
\nm2311 \newunit{\picom}{\pico\metre}
\micm2312 \newunit{\nm}{\nano\metre}
\mm2313 \newunit{\micm}{\micro\metre}
\cm
\dm
\km
```

```

2314 \newunit{\mm}{\milli\metre}
2315 \newunit{\cm}{\centi\metre}
2316 \newunit{\dm}{\deci\metre}
2317 \newunit{\km}{\kilo\metre}

```

\Sec Finally, times.

```

\as2318 \newunit{\Sec}{\second}
\fs2319 \newunit{\as}{\atto\second}
\ps2320 \newunit{\fs}{\femto\second}

```

\ns The letter class (and others) define \ps for postscripts, so \provideunit is best
\mic here.

```

\ms2321 \provideunit{\ps}{\pico\second}
2322 \newunit{\ns}{\nano\second}
2323 \newunit{\mics}{\micro\second}
2324 \newunit{\ms}{\milli\second}

```

17.5 Additional (temporary) SI units

\angstrom Some units are “temporarily” acceptable for use in the SI system. These are
\are defined here, although some are in very general use.

```

\hectare2325 \ProvidesFile{si-addn.cfg}
\bar2326 [2008/02/20 v.06a SI Additional units]
\BAR2327 \newunit{\angstrom}{\si@sym@ringA}
\millibar2328 \newunit{\are}{a}
\gal2329 \newunit{\hectare}{\hecto\are}
\curie2330 \newunit{\bar}{b}
2331 \newunit{\BAR}{bar}
\roentgen2332 \newunit{\millibar}{\milli\BAR}
\rad2333 \newunit{\gal}{Gal}
\rem2334 \newunit{\curie}{Ci}
2335 \newunit{\roentgen}{R}
2336 \newunit{\rad}{rad}
2337 \newunit{\rem}{rem}

```

17.6 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
\hour2338 \ProvidesFile{si-accepted.cfg}
\Day2339 [2008/02/20 v.06a SI accepted units]
\degree2340 \newunit{\minute}{min}
\Degree2341 \newunit{\hour}{h}
\arcmin2342 \newunit{\Day}{d}
\arcsec2343 \ifsi@old@OHM
2344 \newunit[valuesep=none]{\Degree}{\si@sym@degree}
\litre2345 \else
\tonne2346 \ifsi@gensymb\else
\neper2347 \newunit[valuesep=none]{\degree}{\si@sym@degree}
\bel2348 \fi
\percent2349 \fi

```

```

2350 \newunit [valuesep=none] {\arcmin} {\si@sym@minute}
2351 \newunit [valuesep=none] {\arcsec} {\si@sym@second}
2352 \newunit {\litre} {l}
2353 \newunit {\tonne} {t}
2354 \newunit {\neper} {Np}
2355 \newunit {\bel} {B}
2356 \newunit {\percent} {\%}

```

17.7 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
\electronvolt2357 \ProvidesFile{si-physical.cfg}
\atomicmassunit2358 [2008/02/20 v.06a SI physically-measured units]
\atomicmass2359 \newcommand*{\si@eVspacea} {\text{\kern-\si@eVcorra}}%
\dalton2360 \newcommand*{\si@eVspaceb} {\text{\kern-\si@eVcorrb}}%
2361 \newunit {\electronvolt} {e\protect\si@eVspacea V\protect\si@eVspaceb}
2362 \newunit {\atomicmass} {u}
2363 \newunit {\atomicmassunit} {u}
2364 \newunit {\dalton} {Da}

```

18 Additional configurations

To provide flexibility for people in specific areas, specialised units can be set up. These are then stored separately to ease use.

18.1 Synthetic chemistry

`\mmHg` Some useful units for synthetic chemists; although `\mmHg` and `\Molar` are outside of the SI system, they are used a lot. These are set up using `\provideunit` as people may have their own definitions.

```

\Molar
\torr2365 \ProvidesFile{si-synchem.cfg}
2366 [2008/02/20 v.06a Units for synthetic chemists]
2367 \si@requirecfgs{prefix}
2368 \newunit {\mmHg} {mmHg}
2369 \newunit {\molar} {\mole\per\cubic\deci\metre}
2370 \newunit {\Molar} {\textsc{m}}
2371 \newunit {\torr} {Torr}

```

18.2 High-energy physics

The units here basically add the units from the `\hepunits` package which are not defined elsewhere here. It is not entirely clear if `\mrad` refers to radians or rad: feedback would be welcome. This set of commands is not in the emulation block as it does *not* seek to emulate `hepunits`: that package is a blot-on to `Slunits`. The units here have the same name as those in `hepunits` but stick with the new package interface.

```

2372 \ProvidesFile{si-hep.cfg}
2373 [2008/02/20 v.06a Units for high-energy physics]
2374 \si@requirecfgs{prefix,named}

```

`\micron` The first units are not specific to high-energy physics, but are not defined elsewhere in si.

```
\mrad
\gauss2375 \newunit{\micron}{\micro\metre}
2376 \newunit{\mrad}{\milli\radian}
2377 \newunit{\gauss}{G}
```

`\nanobarn` Various prefixed barns

```
\picobarn2378 \newunit{\nanobarn}{\nano\barn}
\femtobarn2379 \newunit{\picobarn}{\pico\barn}
\attobarn2380 \newunit{\femtobarn}{\femto\barn}
\zeptobarn2381 \newunit{\attobarn}{\atto\barn}
\yoctobarn2382 \newunit{\zeptobarn}{\zepto\barn}
2383 \newunit{\yoctobarn}{\yocto\barn}
```

`\invbarn` Inverses barn units.

```
\invnanobarn2384 \newunit{\invbarn}{\per\barn}
\invpicobarn2385 \newunit{\invnanobarn}{\per\nano\barn}
\invfemtobarn2386 \newunit{\invpicobarn}{\per\pico\barn}
\invattobarn2387 \newunit{\invfemtobarn}{\per\femto\barn}
\invzeptobarn2388 \newunit{\invattobarn}{\per\atto\barn}
\invyoctobarn2389 \newunit{\invzeptobarn}{\per\zepto\barn}
2390 \newunit{\invyoctobarn}{\per\yocto\barn}
```

`\invnb` Also available abbreviated.

```
\invpb2391 \newunit{\invnb}{\per\nano\barn}
\invfb2392 \newunit{\invpb}{\per\pico\barn}
\invab2393 \newunit{\invfb}{\per\femto\barn}
\invzb2394 \newunit{\invab}{\per\atto\barn}
\invyb2395 \newunit{\invzb}{\per\zepto\barn}
2396 \newunit{\invyb}{\per\yocto\barn}
```

`\invcmsqpersecond` Luminosity.

```
\invcmsqpersec2397 \newunit{\invcmsqpersecond}{\per\Square\centi\metre\per\second}
\lumiunits2398 \newunit{\invcmsqpersec}{\per\Square\centi\metre\per\second}
2399 \newunit{\lumiunits}{\per\Square\centi\metre\per\second}
```

`\clight` The speed of light is used in units for the area, although of course it is not strictly a unit.

```
2400 \newunit{\clight}{\ensuremath{\mathnormal{c}}}
```

`\inveV` The inverse of an electron-volt, plus prefixes.

```
\minveV2401 \newunit{\inveV}{\per\electronvolt}
\minveV2402 \newunit{\minveV}{\milli\per\electronvolt}
\kinveV2403 \newunit{\kinveV}{\kilo\per\electronvolt}
\MinveV2404 \newunit{\MinveV}{\mega\per\electronvolt}
\GinveV2405 \newunit{\GinveV}{\giga\per\electronvolt}
\TinveV2406 \newunit{\TinveV}{\tera\per\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.

```
2407 \newunit[per=slash,eVcorrb=0.6ex]{\eVoverc}
```



```

2408 {\electronvolt\per\clight}
2409 \newunit[per=slash,eVcorrb=0.6ex]{\eVovercsq}
2410 {\electronvolt\per\Square\clight}

```

`\meVoverc` **Prefixed combinations, first of the speed of light.**

```

\keVoverc2411 \newunit[per=slash,eVcorrb=0.6ex]{\meVoverc}
\MeVoverc2412 {\milli\electronvolt\per\clight}
\GeVoverc2413 \newunit[per=slash,eVcorrb=0.6ex]{\keVoverc}
\TeVoverc2414 {\kilo\electronvolt\per\clight}
2415 \newunit[per=slash,eVcorrb=0.6ex]{\MeVoverc}
2416 {\mega\electronvolt\per\clight}
2417 \newunit[per=slash,eVcorrb=0.6ex]{\GeVoverc}
2418 {\giga\electronvolt\per\clight}
2419 \newunit[per=slash,eVcorrb=0.6ex]{\TeVoverc}
2420 {\tera\electronvolt\per\clight}

```

`\meVovercsq` **Then of the square.**

```

\keVovercsq2421 \newunit[per=slash,eVcorrb=0.6ex]{\meVovercsq}
\MeVovercsq2422 {\milli\electronvolt\per\Square\clight}
\GeVovercsq2423 \newunit[per=slash,eVcorrb=0.6ex]{\keVovercsq}
\TeVovercsq2424 {\kilo\electronvolt\per\Square\clight}
2425 \newunit[per=slash,eVcorrb=0.6ex]{\MeVovercsq}
2426 {\mega\electronvolt\per\Square\clight}
2427 \newunit[per=slash,eVcorrb=0.6ex]{\GeVovercsq}
2428 {\giga\electronvolt\per\Square\clight}
2429 \newunit[per=slash,eVcorrb=0.6ex]{\TeVovercsq}
2430 {\tera\electronvolt\per\Square\clight}

```

18.3 Binary units

`\kibi` The binary units, as specified by the IEC and made available by Slunits. First, the
`\mebi` binary prefixes.

```

\gibi2431 \ProvidesFile{si-binary.cfg}
\tebi2432 [2008/02/20 v.06a Binary units]
\pebi2433 \newprefix{\kibi}{10}{Ki}
\exbi2434 \newprefix{\mebi}{20}{Mi}
2435 \newprefix{\gibi}{30}{Gi}
2436 \newprefix{\tebi}{40}{Ti}
2437 \newprefix{\pebi}{50}{Pi}
2438 \newprefix{\exbi}{60}{Ei}

```

`\bit` Now the units.

```

\byte2439 \newunit{\bit}{bit}
2440 \newunit{\byte}{B}

```

19 Loadable locales

Some short files to provide the correct settings for various places.

19.1 United Kingdom

This is also used for the USA, and is the default.

```
2441 \ProvidesFile{si-UK.cfg}
2442 [2008/02/20 v.06a UK locale]
2443 \sisetup{
2444   unitsep=thin,
2445   expproduct=times,
2446   valuesep=thin,
2447   decimalsign=fullstop,
2448   digitsep=thin,
2449   sepfour=false}
```

19.2 United States

The same as for the UK.

```
2450 \ProvidesFile{si-USA.cfg}
2451 [2008/02/20 v.06a USA locale]
2452 \sisetup{
2453   unitsep=thin,
2454   expproduct=times,
2455   valuesep=thin,
2456   decimalsign=fullstop,
2457   digitsep=thin,
2458   sepfour=false}
```

19.3 Germany

Germany, hopefully.

```
2459 \ProvidesFile{si-germany.cfg}
2460 [2008/02/20 v.06a Germany locale]
2461 \sisetup{
2462   unitsep=cdot,
2463   valuesep=thin,
2464   decimalsign=comma,
2465   expproduct=cdot,
2466   digitsep=thin,
2467   sepfour=false}
```

19.4 South Africa

Taken from sistyle.

```
2468 \ProvidesFile{si-south-africa.loc}
2469 [2008/02/20 v.06a UK Locale]
2470 \sisetup{
2471   unitsep=cdot,
2472   valuesep=thin,
2473   expproduct=times,
2474   decimalsign=comma,
2475   digitsep=thin,
2476   sepfour=false}
```

20 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.

20.1 units

The very first thing to do here is a reload check, as things could go wrong with `unitsdef` emulation.

```
2477 \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.

```
2478 \ProvidesFile{si-units.cfg}
2479 [2008/02/20 v.06a Emulation of units]
2480 \si@ifloaded{SIunits}
2481 {\si@emclash{units}{SIunits}\endinput{}}
2482 \si@ifloaded{sistyle}
2483 {\si@emclash{units}{sistyle}\endinput{}}
```

To emulate `units`, `\per` must give fractions.

```
2484 \sisetup{per=fraction, fraction=nice, obeybold, inlinebold=maths,
2485 , obeymode}
2486 \ifsi@old@tight
2487 \sisetup{valuesep=thin}
2488 \fi
2489 \ifsi@old@loose
2490 \sisetup{valuesep=space}
2491 \fi
2492 \ifsi@old@ugly
2493 \sisetup{fraction=ugly}
2494 \fi
```

`\unit` The `units` version of `\unit` is similar to `\SI`. Here and in `\unitfrac` the `\num` macro is used; thus the number given really has to be a number. However, if users are using `si` rather than `units` they should expect more checking of input. As the `units` package uses the current mode, this has to be detected.

```
\unit [\langle num \rangle] {\langle unit \rangle}
```

```
2495 \DeclareRobustCommand*\unit[2][]{%
2496 \ifmmode
2497 \SI{#1}{#2}%
2498 \else
2499 \SI[obeyfamily,obeyitalic]{#1}{#2}%
2500 \fi}
```

`\unitfrac` `\unitfrac` is a bit more of a hack.

```
\unitfrac [\langle num \rangle] {\langle numerator \rangle} {\langle denominator \rangle}
```

```
2501 \DeclareRobustCommand*\unitfrac[3][]{%
2502 \begingroup
2503 \si@fam@mode%
2504 \ifmmode\else
2505 \sisetup{obeyfamily,obeyitalic}%
```

```

2506     \fi
2507     \si@ifnotmtarg{#1}
2508         {\num{#1}\ensuremath{\si@valuesep}}%
2509     \si@frac{#2}{#3}
2510 \endgroup}

```

20.2 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.

```

2511 \ProvidesFile{si-unitsdef.cfg}
2512 [2008/02/20 v.06a Emulation of unitsdef]
2513 \si@ifloaded{SIunits}
2514 {\si@emclash{unitsdef}{SIunits}\endinput}{}
2515 \si@ifloaded{sistyle}
2516 {\si@emclash{unitsdef}{sistyle}\endinput}{}

```

Emulation of units is needed for unitsdef to work.

```

2517 \si@ifloaded{units}{}
2518 {\InputIfFileExists{\si@fileprefix units.\si@cfgextension}
2519 {}
2520 {\si@log@err{Could not load \si@fileprefix
2521 units.\si@cfgextension}
2522 {The file \si@fileprefix units.\si@cfgextension is
2523 required to emulate\MessageBreak
2524 unitsdef, but cannot be found\MessageBreak
2525 Is the si package properly installed?}
2526 \endinput}}

```

The `unitsdef` package loads some packages that `si` does not. In particular, it loads `textcomp` and `fontenc`. This could be important for output, and so the same is done here.

```

2527 \RequirePackage{textcomp}
2528 \RequirePackage[T1]{fontenc}

```

The multitude of package options for `unitsdef` need to be handled.

```

2529 \sisetup{mode=text}
2530 \ifsi@old@noxspace
2531 \sisetup{xspace=false}
2532 \fi

```

The various options for loading unit abbreviations have to be handled. Here, any request to avoid abbreviations prevents any loading.

```

2533 \ifsi@old@noabbr
2534 \sisetup{noload=abbr}
2535 \fi
2536 \ifsi@old@nofrequencyabbr
2537 \sisetup{noload=abbr}
2538 \fi
2539 \ifsi@old@nomolabbr
2540 \sisetup{noload=abbr}
2541 \fi
2542 \ifsi@old@novoltageabbr
2543 \sisetup{noload=abbr}

```

```

2544 \fi
2545 \ifsi@old@novolumeabbr
2546   \ssetup{noload=abbr}
2547 \fi
2548 \ifsi@old@noweightabbr
2549   \ssetup{noload=abbr}
2550 \fi
2551 \ifsi@old@noenergyabbr
2552   \ssetup{noload=abbr}
2553 \fi
2554 \ifsi@old@nolengthabbr
2555   \ssetup{noload=abbr}
2556 \fi
2557 \ifsi@old@notimeabbr
2558   \ssetup{noload=abbr}
2559 \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.

```

2560 \newcommand*{\unitvaluesep}{\,}
2561 \renewcommand*{\si@valuesep}{\text{\unitvaluesep}}
2562 \si@opt@choicetype{valuesep}{space,thin,med,medium,thick,none}
2563   {\renewcommand*\unitvaluesep\@nameuse{si@fix@##1}}
2564   {\renewcommand*\unitvaluesep{##1}}

```

`\unitsignonly` Some rather straight-forward definitions, with just a bit of fun to get the spacing correct.

```

\ilu
\arc
2565 \DeclareRobustCommand*{\unitsignonly}{\unitsym}
2566 \DeclareRobustCommand*{\ilu}[2][]{%
2567   \begingroup
2568     #1\unitvaluesep%
2569     \unit{#2}%
2570   \endgroup}
2571 \DeclareRobustCommand*{\arc}{\ang}

```

`\unitSIdef` The `unitsdef` package uses a different approach to setting the font inside its version of `\SI`. The problem is the same as for `\unitvaluesep`, but with the added problem that `si` uses `\csname ... \endcsname`.

```

2572 \newcommand*{\unitSIdef}{\upshape}
2573 \newcommand*{\si@unitSIdef}{\unitSIdef\selectfont}
2574 \ssetup{textdefault=si@unitSIdef,textnumdefault=si@unitSIdef}

```

`\per` Rather awkwardly, `unitsdef` uses `\per` in a different way to `si`.

```

2575 \DeclareRobustCommand*{\per}[2]{%
2576   \begingroup
2577     \si@xspacefalse
2578     \renewcommand*{\unitvaluesep}{}%
2579     \unitfrac{#1}{#2}%
2580   \endgroup}

```

`\unittimes` Some pretty straight-forward stuff again; notice that the automatic analyser for units has to be turned off for this to work.

```

\unitsep
\unitsuperscript
2581 \newcommand*{\unittimes}{\ensuremath{\cdot}}

```

```

2582 \newcommand*\unitsep{\,}
2583 \renewcommand*\si@unt@unithook{\si@unt@litouttrue}
2584 \sisetup{unitsep=none}
2585 \newcommand*\unitsuperscript{\tothe}

```

`\newnosepunit` Simple aliases.

```

\renewnosepunit2586 \newcommand*\newnosepunit{\newunit[valuesep=none]}
2587 \newcommand*\renewnosepunit{\renewunit[valuesep=none]}

```

`\setTextOmega` Controlling symbols is a simple translation job; as only one setting is used by si
`\setMathOmega` in text mode, a bit of extra work is needed.

```

\setTextmu2588 \newcommand*\setTextOmega}[2]{%
\setMathmu2589 \renewcommand*\si@textOmega}{%
\setTextCelsius2590 \begingroup
\setMathCelsius2591 \edef\si@tempa{\sfdefault}%
\setMathDegree2592 \ifx\f@family\si@tempa
\setTextDegree2593 \expandafter#2%
2594 \else
2595 \expandafter#1%
2596 \fi
2597 \endgroup}}
2598 \newcommand*\setMathOmega}[1]{\sisetup{mathsOmega=#1}}
2599 \newcommand*\setTextmu}[2]{%
2600 \renewcommand*\si@textmu}{%
2601 \begingroup
2602 \edef\si@tempa{\sfdefault}%
2603 \ifx\f@family\si@tempa
2604 \expandafter#2%
2605 \else
2606 \expandafter#1%
2607 \fi
2608 \endgroup}}
2609 \newcommand*\setMathmu}[1]{\sisetup{mathsmu=#1}}
2610 \newcommand*\setTextCelsius}[2]{%
2611 \renewcommand*\si@textcelsius}{%
2612 \begingroup
2613 \edef\si@tempa{\sfdefault}%
2614 \ifx\f@family\si@tempa
2615 \expandafter#2%
2616 \else
2617 \expandafter#1%
2618 \fi
2619 \endgroup}}
2620 \newcommand*\setMathCelsius}[1]{\sisetup{mathscelsius=#1}}
2621 \newcommand*\setMathDegree}[2]{%
2622 \renewcommand*\si@textdegree}{%
2623 \begingroup%
2624 \edef\si@tempa{\sfdefault}%
2625 \ifx\f@family\si@tempa
2626 \expandafter#2%
2627 \else
2628 \expandafter#1%
2629 \fi
2630 \endgroup}}

```

```
2631 \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.

```
2632 \ifsi@old@OHM
2633   \ifsi@old@ohm
2634     \si@log@inf{Both 'ohm' and 'OHM' options given}\MessageBreak
2635     Using default behaviour for unitsdef}
2636   \expandafter\expandafter\expandafter\si@old@OHMfalse
2637 \fi
2638 \fi
```

`\meter` For some reason, `unitsdef` spells metre and litre incorrectly (the names have an official spelling). Tonne is also spelled as “ton”, which is wrong in the UK at least (1 ton = 40 cwt = 2240 lb!)

```
\days2639 \newunit{\meter}{\metre}
2640 \newunit{\liter}{L}
2641 \ifsi@old@liter
2642   \ifsi@old@LITER
2643     \si@log@inf{Both 'liter' and 'LITER' options given}\MessageBreak
2644     Using default behaviour for unitsdef}
2645   \else
2646     \renewunit{\liter}{l}
2647 \fi
2648 \fi
2649 \newunit{\ton}{t}
2650 \newunit{\days}{d}
```

`\picometer` Extra distances.

```
\nanometer2651 \newunit{\picometer}{\pico\meter}
\micrometer2652 \newunit{\nanometer}{\nano\meter}
\millimeter2653 \newunit{\micrometer}{\micro\meter}
\centimeter2654 \newunit{\millimeter}{\milli\meter}
\decimeter2655 \newunit{\centimeter}{\centi\meter}
\kilometer2656 \newunit{\decimeter}{\deci\meter}
2657 \newunit{\kilometer}{\kilo\meter}
```

`\femtoliter` Volumes with US spellings.

```
\picoliter2658 \newunit{\femtoliter}{\femto\liter}
\nanoliter2659 \newunit{\picoliter}{\pico\liter}
\microliter2660 \newunit{\nanoliter}{\nano\liter}
\milliliter2661 \newunit{\microliter}{\micro\liter}
\centiliter2662 \newunit{\milliliter}{\milli\liter}
\deciliter2663 \newunit{\centiliter}{\centi\liter}
\hectoliter2664 \newunit{\deciliter}{\deci\liter}
\cubicmeter2665 \newunit{\hectoliter}{\hecto\liter}
\cubicmeter2666 \newunit{\cubicmeter}{\meter\cubed}
\cubicmicrometer2667 \newunit{\cubicmicrometer}{\micro\meter\cubed}
\cubicmillimeter2668 \newunit{\cubicmillimeter}{\milli\meter\cubed}
```

`\squaremeter` Areas, including the mis-spellings for `\are` and `\hectare`.

```
\squarecentimeter2669 \newunit{\squaremeter}{\Square\meter}
\squarekilometer2670 \newunit{\squarecentimeter}{\Square\centi\meter}
\ar
\hectar
```

```

2671 \newunit{\squarekilometer}{\Square\kilo\meter}
2672 \newunit{\ar}{a}
2673 \newunit{\hectar}{\hecto\ar}

```

`\kv` The code for `unitsdef` has the capitalisation wrong for `\kV` and `\mV`.

```

\mv2674 \ifsi@old@noabbr
2675 \else
2676 \ifsi@old@novoltageabbr\else
2677 \newunit{\kv}{\kilo\volt}
2678 \newunit{\mv}{\milli\volt}
2679 \fi
2680 \fi

```

`\sek` There are some slightly different abbreviations, plus some which are not officially allowed.

```

\fg2681 \ifsi@old@noabbr\else
\fl2682 \ifsi@old@notimeabbr\else
\nl2683 \newunit{\sek}{\second}
\micl2684 \fi
\ml2685 \ifsi@old@noweightabbr\else
\cl2686 \newunit{\fg}{\femto\gram}
\dl2687 \fi
\hl2688 \ifsi@old@novolumeabbr\else
2689 \newunit{\fl}{\femto\liter}
2690 \newunit{\pl}{\pico\liter}
2691 \newunit{\nl}{\nano\liter}
2692 \renewunit{\micl}{\micro\liter}
2693 \renewunit{\ml}{\milli\liter}
2694 \newunit{\cl}{\centi\liter}
2695 \newunit{\dl}{\deci\liter}
2696 \newunit{\hl}{\hecto\liter}
2697 \fi
2698 \fi

```

`\calory` `unitsdef` spells calorie incorrectly, and it is also not an SI unit.

```

\kilocalory2699 \newunit{\calory}{cal}
2700 \newunit{\kilocalory}{\kilo\calory}

```

`\uBar` `unitsdef` uses `\ubar` for bar.

```

2701 \newunit{\uBar}{ba}

```

`\gensymbohm` If the options relating to `gensymb` are given, then the package *has* to be loaded.
`\gensymbcelsius` The definitions are then renamed; a slight awkward feature is that the hyphen character needs to be a letter. To avoid needing to worry about this again, a
`\gensymbmicro` second switch is set up.
`\gensymbdegree`

```

2702 \catcode`\-=11\relax
2703 \ifsi@old@redef-gensymb
2704 \expandafter\si@gensymbtrue
2705 \fi
2706 \catcode`\-=12\relax
2707 \ifsi@gensymb
2708 \RequirePackage{gensymb}
2709 \AtBeginDocument{

```



```

2710 \let\gensymbohmsymbol\ohm
2711 \let\gensymbcelsius\celsius
2712 \let\gensymbmicro\micro
2713 \let\gensymbdegree\degree
2714 \let\ohm\@undefined
2715 \let\celsius\@undefined
2716 \let\micro\@undefined
2717 \let\degree\@undefined
2718 \ifsi@old@OHM\else
2719 \newunit{\ohm}{\si@sym@Omega}
2720 \newunit{\celsius}{\si@sym@celsius}
2721 \newprefix{\micro}{\si@sym@mu}{-6}
2722 \newunit{\degree}{\si@sym@degree}
2723 \fi}
2724 \fi

```

The configuration files can now be loaded.

```
2725 \si@requirecfgs{prefix,named,addn,accepted}
```

The noconfig option could be ignored, but it costs little to let it be used.

```

2726 \ifsi@old@noconfig\else
2727 \InputIfFileExists{unitsdef.cfg}
2728 {\si@log@inf{unitsdef config file loaded}}
2729 {\si@log@inf{unitsdef config file not found}}
2730 \fi

```

20.3 sistyle

After setting the necessary defaults, the emulation code defines the macros in `sistyle` as given in the manual for that package.

```

2731 \ProvidesFile{si-sistyle.cfg}
2732 [2008/02/20 v.06a Emulation of sistyle]
2733 \sisetup{%
2734 sepfour=true,
2735 obeyfamily,
2736 obeyitalic=true,
2737 numsign=+-,
2738 numextra={},
2739 unitsep=cdot}

```

`\SIobeyboldtrue` Some simple switches, but not using `\newif`.

```

\SIobeyboldfalse2740 \newcommand*{\SIobeyboldtrue}{\sisetup{obeybold=true}}
2741 \newcommand*{\SIobeyboldfalse}{\sisetup{obeybold=false}}

```

`\num` To get the correct behaviour for `\num`, some redefinitions are needed to handle `\si@sis@num` to optional `*`.

```

\si@sis@numstar2742 \DeclareRobustCommand{\num}{%
2743 \@ifstar
2744 {\si@sis@numstar}
2745 {\si@sis@num}}
2746 \newcommand*{\si@sis@num}[2][]{%
2747 \beginngroup%
2748 \sisetup{#1}%

```

```

2749     \expandafter\si@out@num\expandafter{\si@num{#2}}%
2750 \endgroup}
2751 \newcommand*{\si@sis@numstar}[2][ ]{%
2752   \begingroup%
2753     \ssetup{mode=text,obeybold}%
2754     \ssetup{#1}%
2755     \expandafter\si@out@num\expandafter{\si@num{#2}}%
2756 \endgroup}

```

`\pnt` The `\pnt` macro is needed as `.` is active inside `\SI`. The name is exactly the same as in `sistyle`, but the implementation is different. This is not defined by the main package as there are better ways of including numbers in the output than this.

```
2757 \newcommand*{\pnt}{\ensuremath{\si@decimalsign}}
```

`\SIgroupfourtrue` Switches for grouping four characters.

```

\SIgroupfourfalse2758 \newcommand*{\SIgroupfourtrue}{\ssetup{sepfour=true}}
2759 \newcommand*{\SIgroupfourfalse}{\ssetup{sepfour=false}}

```

`\SIunitsep` Whatever is given here is passed through to `\ssetup`.

```

\SIunitspace2760 \newcommand*{\SIunitsep}[1]{\ssetup{valuesep={#1}}}
\SIunitdot2761 \newcommand*{\SIunitspace}[1]{\ssetup{unitspace={#1}}}
2762 \newcommand*{\SIunitdot}[1]{\ssetup{unitsep={#1}}}

```

`\SIdecimalsign` The same is true here, with the appropriate translation.

```

\SIthousandsep2763 \newcommand*{\SIdecimalsign}[1]{\ssetup{decimalsign={#1}}}
\SIproductsign2764 \newcommand*{\SIthousandsep}[1]{\ssetup{digitsep={#1}}}
2765 \newcommand*{\SIproductsign}[1]{\ssetup{expproduct={#1}}}

```

`\si@sis@savefont` The font definitions need a bit of extra work doing. As both settings here have `@` as a letter, all should be fine.

```

\si@sis@savefont{<setting>}{<argument>}
2766 \newcommand{\si@sis@savefont}[2]{%
2767   \@namedef{si@sis@#1}{#2}%
2768   \ssetup{#1=si@sis@#1}}

```

`\SImathrm` The font control macros have to ensure that a macro name is passed to `\ssetup`.

```

\SImathsf2769 \newcommand*{\SImathrm}[1]{\si@sis@savefont{mathrm}{#1}}
\SImathhtt2770 \newcommand*{\SImathsf}[1]{\si@sis@savefont{mathsf}{#1}}
2771 \newcommand*{\SImathhtt}[1]{\si@sis@savefont{mathtt}{#1}}

```

`\SIdefaultMfam` The same for the default keys.

```

\SIdefaultNfam2772 \newcommand*{\SIdefaultMfam}[1]{\si@sis@savefont{mathdefault}{#1}}
\SIdefaultTfam2773 \newcommand*{\SIdefaultNfam}[1]{\si@sis@savefont{mathnumdefault}{#1}}
2774 \newcommand*{\SIdefaultTfam}[1]{\si@sis@savefont{textdefault}{#1}}

```

`\ensureupmath` The `\ensureupmath` command guarantees processing by the font-matching system. The argument cannot be processed here, so care is needed.

```

2775 \DeclareRobustCommand*{\ensureupmath}[1]{%
2776   \begingroup
2777     \ssetup{mode=maths,obeyitalic=false}%
2778     \si@out@text{#1}%
2779 \endgroup}

```

`\degC` A few extra symbol names are needed.

```
\degF2780 \newcommand*{\degC}{\si@sym@celsius}
\arcdeg2781 \newcommand*{\arcdeg}{\si@sym@degree}
2782 \newcommand*{\degF}{\si@sym@degree F}
```

`\AddToSIstyle` Finally, the locale control.

```
\SIstyle2783 \newcommand*{\SIstyle}[1]{\sisetup{locale=#1}}
\SIstyleToLang2784 \newcommand*{\SIstyleToLang}[2]{\sisetup{loctolang=#1:#2}}
\si@sis@addtolocale2785 \newcommand*{\AddToSIstyle}{%
2786 \si@switchfalse
2787 \@ifstar
2788 {\si@switchtrue
2789 \si@sis@addtolocale}
2790 {\si@sis@addtolocale}}
2791 \newcommand*{\si@sis@addtolocale}[2]{%
2792 \ifsi@switch
2793 \expandafter\let\csname si@loc@#1@extra\endcsname\relax
2794 \fi
2795 \addtolocale{#1}{#2}}
```

20.4 Slunits

Slunits emulation starts in much the same way.

```
2796 \ProvidesFile{si-SIunits.cfg}
2797 [2008/02/20 v.06a Emulation of SIunits]
2798 \sisetup{
2799 unitsep=thick,
2800 valuesep=thick,
2801 prefixproduct=\si@valuesep}
2802 \si@requirecfgs{prefix,named,accepted,physical}
```

`\reciprocal` A few very simple translations, using the internal version of `\per` to allow changes of output style.

```
\per2803 \newcommand*{\reciprocal}{\sisetup{per=reciprocal}\si@per}
\usk2804 \let\rp\reciprocal
\power2805 \renewcommand*{\per}{\sisetup{per=slash}\si@per}
\rpsquare2806 \newcommand*{\usk}{}
\rpcubic2807 \newcommand*{\power}[1]{#1\tothe}
\fourth2808 \newcommand*{\rpsquare}{\sisetup{per=reciprocal}\si@per\Square}
\rfourth2809 \newcommand*{\rpcubic}{\sisetup{per=reciprocal}\si@per\cubic}
2810 \newpower{\fourth}{4}
2811 \newcommand*{\rfourth}{\sisetup{per=reciprocal}\si@per\fourth}
```

`\rpsquared` Here, some low-level switch changing is needed.

```
\rpcubed2812 \newcommand*{\rpsquared}{%
2813 \sisetup{per=reciprocal}\si@unt@pertrue\si@unt@perseenttrue\squared}
2814 \newcommand*{\rpcubed}{\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@tempa
\si@siu@setup2815 \newcommand*{\SIsetup}[1]{%
2816 \@for\si@tempa:=#1\do{%
2817 \ifundefined{ifsi@old@#1}
```

```

2818     {\si@log@warn{Unknown SIunits option '#1'}}
2819     {\csname si@old@#1true\endcsname}}
2820 \si@siu@setup}
2821 \newcommand*{\si@siu@setup}{%
2822 \ifsi@old@cdot
2823   \sisetup{unitsep=cdot}%
2824 \fi
2825 \ifsi@old@thickspace
2826   \sisetup{unitsep=thick}%
2827 \fi
2828 \ifsi@old@mediumspace
2829   \sisetup{unitsep=medium}%
2830 \fi
2831 \ifsi@old@thinspace
2832   \sisetup{unitsep=thin}%
2833 \fi
2834 \ifsi@old@thickqspace
2835   \sisetup{valuesep=thick}%
2836 \fi
2837 \ifsi@old@mediumqspace
2838   \sisetup{valuesep=medium}%
2839 \fi
2840 \ifsi@old@thingospace
2841   \sisetup{valuesep=thin}%
2842 \fi}
2843 \si@siu@setup

```

`\square` **SIunits** does slightly different things about the clash with `\square`, and either `\square` or `\square` redefines this macro or provides `\square`.

```

2844 \ifsi@old@squaren
2845   \newpower{\square}{2}
2846 \fi
2847 \AtBeginDocument{%
2848   \@ifundefined{square}
2849     {\newpower{\square}{2}}
2850     {\ifsi@old@amssymb
2851       \renewpower{\square}{2}
2852       \else
2853         \ifsi@old@squaren\else
2854           \si@log@warn{\string\square\space already
2855             defined\MessageBreak SIunits mode may cause
2856             errors}%
2857         \fi
2858       \fi}}

```

`\gray` The potential clash with `PStricks` is also handled differently; here, `\Gray` will already be defined by the `si` kernel.

```

2859 \AtBeginDocument{%
2860   \@ifundefined{gray}
2861     {\newunit{\gray}{Gy}}
2862     {\ifsi@old@pstricks
2863       \renewunit{\gray}{Gy}
2864       \else
2865         \ifsi@old@Gray\else

```

```

2866         \si@log@warn{\string\gray\space already
2867         defined\MessageBreak SIunits mode may cause
2868         errors}%
2869         \fi
2870     \fi}}

```

`\unit` The `\unit` macro is defined.

```

\unita2871 \ifsi@old@italian
2872     \let\unita\SI
2873 \else
2874     \let\unit\SI
2875 \fi

```

The miscellaneous options are moped up.

```

2876 \ifsi@old@textstyle
2877     \sisetup{mode=text}
2878 \fi
2879 \ifsi@old@binary
2880     \sisetup{alsoload=binary}
2881 \fi
2882 \ifsi@old@noams
2883     \AtBeginDocument{%
2884         \renewcommand*{\si@textmu}{\ensuremath\si@mathsmu}}
2885 \fi

```

`\arcminute` The unit macros defined by `SIunits` that are not defined by `si` (by default).

```

\arcsecond2886 \newunit[valuesep=none]{\arcminute}{\si@sym@minute}
\bar2887 \newunit[valuesep=none]{\arcsecond}{\si@sym@second}
\dday2888 \newunit{\bar}{bar}
\liter2889 \newunit{\dday}{day}
\rperminute2890 \newunit{\liter}{L}
\ton2891 \newunit{\rperminute}{r/min}
\degreecelsius2892 \newunit{\ton}{t}
2893 \newunit{\degreecelsius}{\celsius}

```

`\addunit` This is an alias for `\newunit`.

```

2894 \let\addunit\newunit

```

`\addprefix` A little more work for `\addprefix`.

```

2895 \newcommand*{\addprefix}[2]{\newprefix{#1}{#2}}

```

`\graypersecond` `SIunits` provides lots of macros with rather long names, which are not really needed with `si`. However, they have to be defined somewhere. There are a lot of them, so a few are tackled at a time.

```

\metrepersquaresecond2896 \addunit{\graypersecond}{\gray\per\second}
\joulepermole2897 \addunit{\graypersecondnp}{\gray\reciprocal\second}
\joulepermolenp2898 \addunit{\metrepersquaresecond}{\metre\per\second\squared}
\molepercubicmetre2899 \addunit{\metrepersquaresecondnp}{\metre\second\rpsquared}
\molepercubicmetrenp2900 \addunit{\joulepermole}{\joule\per\mole}
\radianpersquaresecond2901 \addunit{\joulepermolenp}{\joule\reciprocal\mole}
\radianpersquaresecondnp2902 \addunit{\molepercubicmetre}{\mole\per\cubic\metre}
\gramsquaremetrepersecond2903 \addunit{\molepercubicmetrenp}{\mole\rpcubic\metre}
2904 \addunit{\radianpersquaresecond}{\radian\per\second\squared}

```

2905 \addunit{\radianpersquaresecondnp}{\radian\second\rpsquared}
 2906 \addunit{\kilogramsquaremetrepersecond}
 2907 {\kilogram\usk\squaremetre\per\second}

ramsquaremetrepersecondnp **Some more.**

\radianpersecond2908 \addunit{\kilogramsquaremetrepersecondnp}
 \radianpersecondnp2909 {\kilogram\usk\squaremetre\reciprocal\second}
 \squaremetrepercubicmetre2910 \addunit{\radianpersecond}{\radian\per\second}
 \squaremetrepercubicmetrenp2911 \addunit{\radianpersecondnp}{\radian\reciprocal\second}
 \katalpercubicmetre2912 \addunit{\squaremetrepercubicmetre}{\squaremetre\per\cubic\metre}
 \katalpercubicmetrenp2913 \addunit{\squaremetrepercubicmetrenp}{\squaremetre\percubic\metre}
 \coulombpermol2914 \addunit{\katalpercubicmetre}{\katal\per\cubic\metre}
 \coulombpermolnp2915 \addunit{\katalpercubicmetrenp}{\katal\percubic\metre}
 \amperepersquaremetre2916 \addunit{\coulombpermol}{\coulomb\per\mole}
 \amperepersquaremetrenp2917 \addunit{\coulombpermolnp}{\coulomb\reciprocal\mole}
 \amperepersquaremetrenp2918 \addunit{\amperepersquaremetre}{\ampere\per\squaremetre}
 2919 \addunit{\amperepersquaremetrenp}{\ampere\rpsquare\metre}

\kilogrampercubicmetre **Some more.**

\kilogrampercubicmetrenp2920 \addunit{\kilogrampercubicmetre}{\kilogram\per\cubic\metre}
 \squaremetrepernewtonsecond2921 \addunit{\kilogrampercubicmetrenp}{\kilogram\percubic\metre}
 \squaremetrepernewtonsecondnp2922 \addunit{\squaremetrepernewtonsecond}
 \pascalsecond2923 {\squaremetre\per\newton\second}
 \coulombpercubicmetre2924 \addunit{\squaremetrepernewtonsecondnp}
 \coulombpercubicmetrenp2925 {\squaremetre\reciprocal\newton\reciprocal\second}
 \ampere metre second2926 \addunit{\pascalsecond}{\pascal\second}
 \voltpermetre2927 \addunit{\coulombpercubicmetre}{\coulomb\per\cubic\metre}
 \voltpermetrenp2928 \addunit{\coulombpercubicmetrenp}{\coulomb\percubic\metre}
 \coulombpersquaremetre2929 \addunit{\ampere metre second}{\ampere\metre\second}
 2930 \addunit{\voltpermetre}{\volt\per\metre}
 2931 \addunit{\voltpermetrenp}{\volt\reciprocal\metre}
 2932 \addunit{\coulombpersquaremetre}{\coulomb\per\squaremetre}

\coulombpersquaremetrenp **Some more.**

\faradpermetre2933 \addunit{\coulombpersquaremetrenp}{\coulomb\rpsquare\metre}
 \faradpermetrenp2934 \addunit{\faradpermetre}{\farad\per\metre}
 \ohmmetre2935 \addunit{\faradpermetrenp}{\farad\reciprocal\metre}
 \kilowatthour2936 \addunit{\ohmmetre}{\ohm\metre}
 \wattpersquaremetre2937 \addunit{\kilowatthour}{\kilo\watt\hour}
 \wattpersquaremetrenp2938 \addunit{\wattpersquaremetre}{\watt\per\squaremetre}
 \joulepersquaremetre2939 \addunit{\wattpersquaremetrenp}{\watt\rpsquare\metre}
 \joulepersquaremetrenp2940 \addunit{\joulepersquaremetre}{\joule\per\squaremetre}
 \joulepersquaremetrenp2941 \addunit{\joulepersquaremetrenp}{\joule\rpsquare\metre}
 \newtonpercubicmetre2942 \addunit{\newtonpercubicmetre}{\newton\per\cubic\metre}
 \newtonpercubicmetrenp2943 \addunit{\newtonpercubicmetrenp}{\newton\percubic\metre}

\newtonperkilogram **Some more.**

\newtonperkilogramnp2944 \addunit{\newtonperkilogram}{\newton\per\kilogram}
 \jouleperkelvin2945 \addunit{\newtonperkilogramnp}{\newton\reciprocal\kilogram}
 \jouleperkelvinnp2946 \addunit{\jouleperkelvin}{\joule\per\kelvin}
 \jouleperkilogram2947 \addunit{\jouleperkelvinnp}{\joule\reciprocal\kelvin}
 \jouleperkilogramnp2948 \addunit{\jouleperkilogram}{\joule\per\kilogram}
 \coulombperkilogram2949 \addunit{\jouleperkilogramnp}{\joule\reciprocal\kilogram}
 \coulombperkilogramnp2950 \addunit{\coulombperkilogram}{\coulomb\per\kilogram}

\squaremetrepersecond
 \squaremetrepersecondnp
 \squaremetreperpersquaresecond

2951 \addunit{\coulombperkilogramnp}{\coulomb\reciprocal\kilogram}
 2952 \addunit{\squaremetrepersecond}{\squaremetre\per\second}
 2953 \addunit{\squaremetrepersecondnp}{\squaremetre\reciprocal\second}
 2954 \addunit{\squaremetreppersquaresecond}
 2955 {\squaremetre\per\second\squared}

aremetreppersquaresecondnp **Some more.**

\kilogrammetrepersecond2956 \addunit{\squaremetreppersquaresecondnp}
 \kilogrammetrepersecondnp2957 {\squaremetre\second\rpsquared}
 \candelapersquaremetre2958 \addunit{\kilogrammetrepersecond}{\kilogram\metre\per\second}
 \candelapersquaremetrenp2959 \addunit{\kilogrammetrepersecondnp}
 \amperepermetre2960 {\kilogram\metre\reciprocal\second}
 \amperepermetrenp2961 \addunit{\candelapersquaremetre}{\candela\per\squaremetre}
 \joulepertesla2962 \addunit{\candelapersquaremetrenp}{\candela\rpsquare\metre}
 \jouleperteslanp2963 \addunit{\amperepermetre}{\ampere\per\metre}
 \henrypermetre2964 \addunit{\amperepermetrenp}{\ampere\reciprocal\metre}
 \henrypermetrenp2965 \addunit{\joulepertesla}{\joule\per\tesla}
 \henrypermetrenp2966 \addunit{\jouleperteslanp}{\joule\reciprocal\tesla}
 2967 \addunit{\henrypermetre}{\henry\per\metre}
 2968 \addunit{\henrypermetrenp}{\henry\reciprocal\metre}

\kilogrampersecond **Some more.**

\kilogrampersecondnp2969 \addunit{\kilogrampersecond}{\kilogram\per\second}
 ogramppersquaremetresecond2970 \addunit{\kilogrampersecondnp}{\kilogram\reciprocal\second}
 rampersquaremetresecondnp2971 \addunit{\kilogramppersquaremetresecond}
 \kilogramppersquaremetre2972 {\kilogram\per\squaremetre\second}
 \kilogramppersquaremetrenp2973 \addunit{\kilogramppersquaremetresecondnp}
 \kilogrampermetre2974 {\kilogram\rpsquare\metre\reciprocal\second}
 \kilogrampermetrenp2975 \addunit{\kilogramppersquaremetre}{\kilogram\per\squaremetre}
 \joulepermolekelvin2976 \addunit{\kilogramppersquaremetrenp}{\kilogram\rpsquare\metre}
 \joulepermolekelvinnp2977 \addunit{\kilogrampermetre}{\kilogram\per\metre}
 \kilogramperkilomole2978 \addunit{\kilogrampermetrenp}{\kilogram\reciprocal\metre}
 \kilogramperkilomolenp2979 \addunit{\joulepermolekelvin}{\joule\per\mole\kelvin}
 2980 \addunit{\joulepermolekelvinnp}
 2981 {\joule\reciprocal\mole\reciprocal\kelvin}
 2982 \addunit{\kilogramperkilomole}{\kilogram\per\kilo\mole}

\kilogramperkilomolenp **Some more.**

\kilogramsquaremetre2983 \addunit{\kilogramperkilomolenp}{\kilogram\kilo\reciprocal\mole}
 \kilogramsquaremetrenp2984 \addunit{\kilogramsquaremetre}{\kilogram\squaremetre}
 ogrammetreppersquaresecond2985 \addunit{\kilogramsquaremetrenp}{\kilogramsquaremetre}
 rammetreppersquaresecondnp2986 \addunit{\kilogrammetreppersquaresecond}
 \newtonpersquaremetre2987 {\kilogram\metre\per\second\squared}
 \newtonpersquaremetrenp2988 \addunit{\kilogrammetreppersquaresecondnp}
 \persquaremetresecond2989 {\kilogram\metre\second\rpsquared}
 \persquaremetresecondnp2990 \addunit{\newtonpersquaremetre}{\newton\per\squaremetre}
 \wattperkilogram2991 \addunit{\newtonpersquaremetrenp}{\newton\rpsquare\metre}
 \wattperkilogramnp2992 \addunit{\persquaremetresecond}{1\per\squaremetre\second}
 \wattperkilogramnp2993 \addunit{\persquaremetresecondnp}{\rpsquare\metre\reciprocal\second}
 2994 \addunit{\wattperkilogram}{\watt\per\kilogram}
 2995 \addunit{\wattperkilogramnp}{\watt\reciprocal\kilogram}

\wattpercubicmetre **Some more.**

\wattpercubicmetrenp2996 \addunit{\wattpercubicmetre}{\watt\per\cubic\metre}

ttpersquaremetresteradian
 persquaremetresteradiannp
 \jouleperkilogramkelvin
 \jouleperkilogramkelvinnp
 \squaremetreperkilogram
 \rpsquaremetreperkilogram
 \cubicmetreperkilogram
 \rpcubicmetreperkilogram
 \newtonpermetre

2997 \addunit{\wattpercubicmetrenp}{\watt\rpcubic\metre}
 2998 \addunit{\wattpersquaremetresteradian}{\watt\per\squaremetre\steradian}
 2999 \addunit{\wattpersquaremetresteradianp}
 3000 {\watt\rsquare\metre\rp\steradian}
 3001 \addunit{\jouleperkilogramkelvin}{\joule\per\kilogram\kelvin}
 3002 \addunit{\jouleperkilogramkelvinp}
 3003 {\joule\reciprocal\kilogram\reciprocal\kelvin}
 3004 \addunit{\squaremetreperkilogram}{\squaremetre\per\kilogram}
 3005 \addunit{\rsquaremetreperkilogram}
 3006 {\squaremetre\reciprocal\kilogram}
 3007 \addunit{\cubicmetreperkilogram}{\cubic\metre\per\kilogram}
 3008 \addunit{\rpcubicmetreperkilogram}
 3009 {\cubic\metre\reciprocal\kilogram}
 3010 \addunit{\newtonpermetre}{\newton\per\metre}

\newtonpermetrenp **Some more.**

\wattpermetrekelvin3011 \addunit{\newtonpermetrenp}{\newton\reciprocal\metre}
 \wattpermetrekelvinp3012 \addunit{\wattpermetrekelvin}{\watt\per\metre\kelvin}
 \newtonmetre3013 \addunit{\wattpermetrekelvinp}
 \newtonmetrenp3014 {\watt\reciprocal\metre\reciprocal\kelvin}
 \squaremetrepercubicsecond3015 \addunit{\newtonmetre}{\newton\metre}
 \squaremetrepercubicsecondnp3016 \addunit{\newtonmetrenp}{\newtonmetre}
 \metrepersecond3017 \addunit{\squaremetrepercubicsecond}{\squaremetre\per\cubic\second}
 \metrepersecondnp3018 \addunit{\squaremetrepercubicsecondnp}
 \joulepercubicmetre3019 {\squaremetre\rpcubic\second}
 \joulepercubicmetrenp3020 \addunit{\metrepersecond}{\metre\per\second}
 \joulepercubicmetrenp3021 \addunit{\metrepersecondnp}{\metre\reciprocal\second}
 3022 \addunit{\joulepercubicmetre}{\joule\per\cubicmetre}
 3023 \addunit{\joulepercubicmetrenp}{\joule\rpcubic\metre}

\kilogrampercubicmetrecoulomb **Last block.**

\kilogrampercubicmetrecoulombnp3024 \addunit{\kilogrampercubicmetrecoulomb}
 \cubicmetrepersecond3025 {\kilogram\per\cubic\metre\coulomb}
 \rpcubicmetrepersecond3026 \addunit{\kilogrampercubicmetrecoulombnp}
 \kilogrampersecondcubicmetre3027 {\kilogram\rpcubic\metre\reciprocal\coulomb}
 \kilogrampersecondcubicmetrenp3028 \addunit{\cubicmetrepersecond}{\cubicmetre\per\second}
 3029 \addunit{\rpcubicmetrepersecond}{\cubicmetre\reciprocal\second}
 3030 \addunit{\kilogrampersecondcubicmetre}
 3031 {\kilogram\per\second\cubicmetre}
 3032 \addunit{\kilogrampersecondcubicmetrenp}
 3033 {\kilogram\reciprocal\second\rpcubic\metre}

\yoctod **The prefixes giving numerical output need a trick. First the small values.**

\zeptod3034 \newunit{\yoctod}{\si@prefixnumtrue\yocto}
 \attod3035 \newunit{\zeptod}{\si@prefixnumtrue\zepto}
 \femtod3036 \newunit{\attod}{\si@prefixnumtrue\atto}
 \picod3037 \newunit{\femtod}{\si@prefixnumtrue\femto}
 \nanod3038 \newunit{\picod}{\si@prefixnumtrue\pico}
 \microd3039 \newunit{\nanod}{\si@prefixnumtrue\nano}
 \millid3040 \newunit{\microd}{\si@prefixnumtrue\micro}
 \centid3041 \newunit{\millid}{\si@prefixnumtrue\milli}
 3042 \newunit{\centid}{\si@prefixnumtrue\centi}

\decad **The the larger ones.**

\dekad
 \hectod
 \kilogd
 \megad
 \gigad
 \terad
 \petad
 \exad
 \zettad
 \yottad


```

3043 \newunit{\decad}{\si@prefixnumtrue\deca}
3044 \newunit{\dekad}{\si@prefixnumtrue\deka}
3045 \newunit{\hectod}{\si@prefixnumtrue\hecto}
3046 \newunit{\kilod}{\si@prefixnumtrue\kilo}
3047 \newunit{\megad}{\si@prefixnumtrue\mega}
3048 \newunit{\gigad}{\si@prefixnumtrue\giga}
3049 \newunit{\terad}{\si@prefixnumtrue\tera}
3050 \newunit{\petad}{\si@prefixnumtrue\peta}
3051 \newunit{\exad}{\si@prefixnumtrue\exa}
3052 \newunit{\zettad}{\si@prefixnumtrue\zetta}
3053 \newunit{\yottad}{\si@prefixnumtrue\yotta}

```

`\kibid` The binary versions need a little more work.

```

\mebid3054 \newunit{\kibid}{%
\gibid3055 \si@prefixnumtrue\let\si@prefixpower\si@fix@two\kibi}
\tebid3056 \newunit{\mebid}{%
\pebid3057 \si@prefixnumtrue\let\si@prefixpower\si@fix@two\mebi}
\exbid3058 \newunit{\gibid}{%
3059 \si@prefixnumtrue\let\si@prefixpower\si@fix@two\gibi}
3060 \newunit{\tebid}{%
3061 \si@prefixnumtrue\let\si@prefixpower\si@fix@two\tebi}
3062 \newunit{\pebid}{%
3063 \si@prefixnumtrue\let\si@prefixpower\si@fix@two\pebi}
3064 \newunit{\exbid}{%
3065 \si@prefixnumtrue\let\si@prefixpower\si@fix@two\exbi}

```

`\derradian` The derived units may need to be defined.

```

\dersteradian3066 \ifsi@old@derived
\derhertz3067 \newunit{\derradian}{\metre\reciprocal\metre}
\dernewton3068 \newunit{\dersteradian}{\squaremetre\rpsquare\metre}
\derpascal3069 \newunit{\derhertz}{\reciprocal\second}
\derjoule3070 \newunit{\dernewton}{\metre\kilogram\second\rpsquared}
\derwatt3071 \newunit{\derpascal}{\newton\rpsquare\metre}
\dercoulomb3072 \newunit{\derjoule}{\newton\metre}
\dervolt3073 \newunit{\derwatt}{\joule\reciprocal\second}
\derfarad3074 \newunit{\dercoulomb}{\ampere\second}
\derohm3075 \newunit{\dervolt}{\watt\reciprocal\ampere}
\derohm3076 \newunit{\derfarad}{\coulomb\reciprocal\volt}
3077 \newunit{\derohm}{\volt\reciprocal\ampere}

```

`\dersiemens` In two blocks!

```

\derweber3078 \newunit{\dersiemens}{\ampere\reciprocal\volt}
\dertesla3079 \newunit{\derweber}
\derhenry3080 {\squaremetre\kilogram\second\rpsquared\reciprocal\ampere}
\dercelsius3081 \newunit{\dertesla}{\weber\rpsquare\metre}
\derlumen3082 \newunit{\derhenry}{\weber\reciprocal\ampere}
\derlux3083 \newunit{\dercelsius}{\kelvin}
\derbecquerel3084 \newunit{\derlumen}{\candela\steradian}
\dergray3085 \newunit{\derlux}{\lumen\rpsquare\metre}
\dersievert3086 \newunit{\derbecquerel}{\derhertz}
\derkatal3087 \newunit{\dersievert}{\joule\reciprocal\kilogram}
\derkatal3088 \newunit{\derkatal}{\rp\second\usk\mole}
3089 \newunit{\derkatal}{\rp\second\usk\mole}
3090 \fi

```

```

\radianbase    Also the “derived-in-base”.
\steradianbase3091 \ifsi@old@derivedinbase
  \hertzbase3092  \newunit{\radianbase}{\metre\reciprocal\metre}
  \newtonbase3093 \newunit{\steradianbase}{\squaremetre\rpsquare\metre}
  \pascalbase3094 \newunit{\hertzbase}{\reciprocal\second}
  \joulebase3095  \newunit{\newtonbase}{\metre\kilogram\second\rpsquared}
  \wattbase3096   \newunit{\pascalbase}{\reciprocal\metre\kilogram\second\rpsquared}
  \coulombbase3097 \newunit{\joulebase}{\squaremetre\kilogram\second\rpsquared}
  \voltbase3098   \newunit{\wattbase}{\squaremetre\kilogram\rpcubic\second}
  \faradbase3099  \newunit{\coulombbase}{\ampere\second}
  \ohmbase3100    \newunit{\voltbase}
  \ohmbase3101    {\squaremetre\kilogram\rpcubic\second\reciprocal\ampere}
  \ohmbase3102    \newunit{\faradbase}
  \ohmbase3103    {\rpsquare\metre\reciprocal\kilogram\fourth\second\ampere%
  \ohmbase3104    \squared}
  \ohmbase3105    \newunit{\ohmbase}
  \ohmbase3106    {\squaremetre\kilogram\rpcubic\second\rpsquare\ampere}

\siemensbase   Also in two blocks.
  \weberbase3107  \newunit{\siemensbase}
  \teslabase3108  {\rpsquare\metre\reciprocal\kilogram\cubic\second\ampere\squared}
  \henrybase3109  \newunit{\weberbase}
  \celsiusbase3110 {\squaremetre\kilogram\second\rpsquared\reciprocal\ampere}
  \lumenbase3111  \newunit{\teslabase}{\kilogram\second\rpsquared\reciprocal\ampere}
  \luxbase3112    \newunit{\henrybase}
  \luxbase3113    {\squaremetre\kilogram\second\rpsquared\rpsquare\ampere}
\becquerelbase3114 \newunit{\celsiusbase}{\kelvin}
  \graybase3115   \newunit{\lumenbase}{\candela\squaremetre\rpsquare\metre}
  \sievertbase3116 \newunit{\luxbase}{\candela\squaremetre\rpfourth\metre}
  \katalbase3117  \newunit{\becquerelbase}{\hertzbase}
  \katalbase3118  \newunit{\graybase}{\squaremetre\second\rpsquared}
  \katalbase3119  \newunit{\sievertbase}{\graybase}
  \katalbase3120  \newunit{\katalbase}{\rp\second\mole}
3121 \fi

Any configuration file is used if found.
3122 \InputIfFileExists{SIunits.cfg}
3123  {\si@log@inf{SIunits config file loaded}}
3124  {\si@log@inf{SIunits config file not found}}

```

Part V

Notes

21 Change History

0.6

General: First public release 1

vo.6a

\si@requirecfgs: Changed temporary macro used to fix bug with memoir and KOMA-script . 86

22 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	
$\%$	2356
$\+$	805
$\,$	504, 803, 808, 2560, 2582
$\-$	806, 1962, 1973, 2702, 2706
$\.$	804, 1883, 1889
$\:$	505, 506
$\;$	507, 1146
$\^$	810, 1961, 1965
\sim ..	809, 1882, 1888, 1907
A	
\addprefix	2895
\addtolocale ..	<u>1959</u> , 2795
\AddToIstyle ..	<u>2783</u>
\addunit ..	<u>2894</u> , 2896–2906, 2908, 2910–2922, 2924, 2926–2954, 2956, 2958, 2959, 2961–2971, 2973, 2975–2980, 2982–2986, 2988, 2990–2999, 3001, 3002, 3004, 3005, 3007, 3008, 3010–3013, 3015–3018, 3020–3024, 3026, 3028–3030, <u>3032</u>
\ampere	<u>12</u> , 1911, 2216–2220, 2275–2279, 2918, 2919, 2929, 2963, 2964, 3074, 3075, 3077, 3078, 3080, 3082, 3099, 3101, 3103, 3106, 3108, 3110, 3111, 3113
$\text{\ampere metre second}$	<u>2920</u>
\ampere per metre ..	<u>2956</u>
$\text{\ampere per metre renp}$	<u>2956</u>
$\text{\ampere per square metre}$	<u>2908</u>
$\text{\ampere per square metre renp}$	<u>2908</u>
\amu	<u>14</u> , <u>2297</u>
\ang	<u>8</u> , <u>1141</u> , 2571
\angstrom	<u>14</u> , <u>2325</u>
\ar	<u>2669</u>
\arc	<u>2565</u>
\arcdeg	<u>2780</u>
\arcmin	<u>13</u> , <u>2338</u>
\arcminute	<u>2886</u>
\arcsec	<u>13</u> , <u>2338</u>
\arcsecond	<u>2886</u>
\are	<u>14</u> , <u>2325</u>
\as	<u>14</u> , <u>2318</u>
\atomicmass	<u>13</u> , <u>14</u> , 2303, <u>2357</u>
\atomicmassunit ..	<u>13</u> , <u>2357</u>
\atto	<u>12</u> , <u>2123</u> , 2210, 2319, 2381, 2388, 2394, 3036
\attobarn	<u>2378</u>
\attod	<u>3034</u>
\attosecond ..	<u>14</u> , <u>2210</u>
B	
\BAR	<u>14</u> , <u>2325</u>
\barn ..	<u>14</u> , <u>2325</u> , 2378–2396
\bbar	<u>2886</u>
\becquerel ..	<u>13</u> , <u>2154</u> , 2270
\becquerel base ..	<u>3107</u>
\bel	<u>13</u> , <u>2338</u>
\bit	<u>16</u> , <u>2439</u>
\byte	<u>16</u> , <u>2439</u>
C	
\calory	<u>2699</u>
\candela	<u>12</u> , 1911, 2961, 2962, 3084, 3115, 3116
$\text{\candelapersquaremetre}$..	<u>2956</u>
$\text{\candelapersquaremetre renp}$..	<u>2956</u>
\Celsius	<u>2181</u>
\celsius	<u>13</u> , <u>2181</u> , 2711, 2715, 2720, 2893
\celsius base ..	<u>3107</u>
\centi ..	<u>12</u> , <u>2123</u> , 2197, 2246, 2247, 2252, 2295, 2315, 2397–2399, 2655, 2663, 2670, 2694, 3042
\centid	<u>3034</u>
\centiliter	<u>2658</u>
\centimeter	<u>2651</u>
\centimetre ..	<u>14</u> , <u>2193</u>
\centimetrecubed ..	<u>15</u> , <u>15</u> , <u>2243</u>
\cl	<u>2681</u>
\cflight	<u>2400</u> , 2408, 2410, 2412, 2414, 2416, 2418, 2420, 2422, 2424, 2426, 2428, 2430
\cm	<u>14</u> , <u>2311</u>
\cmc	<u>15</u> , <u>15</u> , <u>2293</u>
\coulomb	<u>13</u> , 2154, 2916, 2917, 2927, 2928, 2932, 2933, 2950, 2951, 3025, 3027, 3076
\coulomb base ..	<u>3091</u>
$\text{\coulombpercubicmetre}$..	<u>2920</u>
$\text{\coulombpercubicmetre renp}$..	<u>2920</u>
$\text{\coulombperkilogram}$..	<u>2944</u>
$\text{\coulombperkilogramnp}$..	<u>2944</u>
\coulombpermol ..	<u>2908</u>
\coulombpermolnp ..	<u>2908</u>
$\text{\coulombpersquaremetre}$..	<u>2920</u>
$\text{\coulombpersquaremetre renp}$..	<u>2933</u>
\cubed ..	<u>10</u> , <u>1918</u> , 2245–2249, 2295, 2296, 2666–2668, 2814

\hectoliter	<u>2658</u>	\ifsi@old@binary	\ifsi@old@pstricks
\hectopascal .	<u>16</u> , <u>2267</u>	<u>499</u> , <u>2879</u>
\henry <u>13</u> , <u>2154</u> , <u>2967</u> , <u>2968</u>		\ifsi@old@cdot ..	\ifsi@old@redef <u>2703</u>
\henrybase	<u>3107</u>	<u>487</u> , <u>2822</u>
\henrypermetre .	<u>2956</u>	\ifsi@old@derived	\ifsi@old@redef-gensymb
\henrypermetrenp	<u>2956</u>	<u>499</u> , <u>3066</u>
\hertz	<u>13</u> ,	\ifsi@old@derivedinbase	\ifsi@old@squaren
<u>15</u> , <u>2154</u> , <u>2262</u> –		<u>494</u> , <u>2844</u> , <u>2853</u>
<u>2266</u> , <u>2280</u> – <u>2285</u>		\ifsi@old@Gray ..	\ifsi@old@textstyle
\hertzbase .	<u>3091</u> , <u>3117</u>	<u>499</u> , <u>2876</u>
\hl	<u>2681</u>	\ifsi@old@italian	\ifsi@old@thickqspace
\hour	<u>13</u> , <u>2338</u> , <u>2937</u>	<u>487</u> , <u>2834</u>
\Hz	<u>15</u> , <u>2280</u>	\ifsi@old@LITER .	\ifsi@old@thickspace
		<u>487</u> , <u>2825</u>
		\ifsi@old@liter .	\ifsi@old@thingspace
		<u>487</u> , <u>2840</u>
		\ifsi@old@literate ..	\ifsi@old@thinspace
		<u>473</u> , <u>2641</u>
		\ifsi@old@loose .	\ifsi@old@tight ..
		<u>487</u> , <u>2831</u>
		\ifsi@old@mediumqspace	\ifsi@old@tight ..
		<u>465</u> , <u>2486</u>
		\ifsi@old@mediumspace	\ifsi@old@ugly ..
		<u>487</u> , <u>2837</u>
		\ifsi@old@mediumspace
		<u>465</u> , <u>2492</u>
		\ifsi@old@nice ..	\ifsi@prefixnum .
		<u>465</u>
		\ifsi@old@noabbr	<u>224</u> , <u>1683</u>
		\ifsi@redefsymbols
		<u>477</u> , <u>2533</u> , <u>2674</u> , <u>2681</u>
		\ifsi@old@noamperageabbr	<u>444</u> , <u>539</u>
		\ifsi@sepfour ...
		<u>477</u>	<u>219</u> , <u>1085</u> , <u>1125</u> , <u>1413</u>
		\ifsi@old@noams .	\ifsi@slash
		<u>346</u> , <u>1854</u> , <u>1874</u>
		\ifsi@old@noconfig	\ifsi@switch ..
		<u>31</u> ,
		\ifsi@old@noenergyabbr	<u>815</u> , <u>838</u> , <u>897</u> ,
		<u>905</u> , <u>1014</u> , <u>1029</u> ,
		\ifsi@old@nofrequencyabbr	<u>1042</u> , <u>1070</u> , <u>1277</u> ,
		<u>1285</u> , <u>2093</u> , <u>2792</u>
		\ifsi@old@nolengthabbr	\ifsi@textmode ..
		<u>188</u> , <u>789</u> , <u>1971</u>
		\ifsi@old@nomolabbr	\ifsi@unt@first .
		<u>1521</u> , <u>1623</u>
		\ifsi@old@notimeabbr	\ifsi@unt@litout
		<u>1493</u> ,
		\ifsi@old@novoltageabbr	<u>1512</u> , <u>1565</u> , <u>1580</u> ,
		<u>1596</u> , <u>1848</u> , <u>1926</u>
		\ifsi@old@novolumeabbr	\ifsi@unt@littest
		<u>1483</u> , <u>1562</u> ,
		\ifsi@old@noweightabbr	<u>1577</u> , <u>1593</u> , <u>1923</u>
		\ifsi@unt@num ...
		\ifsi@old@noxspace	<u>1465</u> , <u>1508</u> , <u>1834</u>
		\ifsi@unt@per ...
		\ifsi@old@OHM ...	<u>1672</u> , <u>1693</u> , <u>1705</u> ,
		<u>1744</u> , <u>1755</u> , <u>1822</u>
		\ifsi@old@ohm	\ifsi@unt@perseen
		<u>469</u> , <u>2131</u> ,
		\ifsi@old@amssymb	<u>1744</u> , <u>1756</u>
		\ifsi@unt@prepower
		\ifsi@old@ohm	<u>2167</u> , <u>2181</u> , <u>2232</u> ,
		<u>2343</u> , <u>2632</u> , <u>2718</u>
		\ifsi@old@ohm	\ifsi@unt@prepower
		<u>469</u> , <u>2633</u>
			<u>1614</u> , <u>1664</u> , <u>1701</u>

I

<code>\kilohertz</code> ... 15 , 2262	2639, 2898, 2899,	<code>\microfarad</code> .. 16 , 2221
<code>\kilojoule</code> ... 15 , 2254	2902, 2903, 2912–	<code>\microgram</code> ... 14 , 2200
<code>\kilometer</code> 2651	2915, 2919–2921,	<code>\microliter</code> 2658
<code>\kilometre</code> ... 14 , 2193	2927–2931, 2933–	<code>\microlitre</code> .. 15 , 2243
<code>\kilonewton</code> .. 16 , 2267	2936, 2939, 2941–	<code>\micrometer</code> 2651
<code>\kiloohm</code> 16 , 2232	2943, 2958, 2960,	<code>\micrometre</code> .. 14 , 2193
<code>\kilovolt</code> 15 , 2221	2962–2964, 2967,	<code>\micromole</code> ... 15 , 2205
<code>\kilowatt</code> 16 , 2221	2968, 2974, 2976–	<code>\micron</code> 2375
<code>\kilowatthour</code> .. 2933	2978, 2987, 2989,	<code>\microsecond</code> . 15 , 2210
<code>\kinveV</code> 2401	2991, 2993, 2996,	<code>\mics</code> 15 , 2318
<code>\kJ</code> 15 , 2304	2997, 3000, 3007,	<code>\milli</code> 12 , 2123 ,
<code>\km</code> 14 , 2311	3009–3012, 3014,	2196, 2204, 2209,
<code>\kV</code> 15 , 2291	3015, 3020, 3021,	2215, 2219, 2221,
<code>\kv</code> 2674	3023, 3025, 3027,	2223, 2230, 2231,
	3033, 3067, 3068,	2244, 2249, 2254,
	3070–3072, 3081,	2257, 2262, 2267,
	3085, 3092, 3093,	2271, 2278, 2281,
	3095, 3096, 3103,	2290, 2292, 2293,
	3108, 3115, 3116	2302, 2306, 2314,
<code>\liter</code> 2639, 2658–2665, 2689–2696, 2886	<code>\metrepersecond</code> 3011	2324, 2332, 2376,
<code>\litre</code> 13 , 2243 , 2244 ,	<code>\metrepersecondnp</code> 3011	2402, 2412, 2422,
2293, 2294, 2338	<code>\metrepersecondnp</code> 3011	2654, 2662, 2668,
<code>\lumen</code> ... 13 , 2154 , 3085	<code>\metrepersecondnp</code> 2896	2678, 2693, 3041
<code>\lumenbase</code> 3107	<code>\metrepersecondnp</code> 2896	<code>\milliampere</code> . 15 , 2216
<code>\lumiunits</code> 2397	<code>\MeV</code> 16 , 2304	<code>\millibar</code> 14 , 2325
<code>\lux</code> 13 , 2154	<code>\meV</code> 16 , 2304	<code>\millid</code> 3034
<code>\luxbase</code> 3107	<code>\MeVoverc</code> 2411	<code>\millielectronvolt</code> 16 , 2254
	<code>\meVoverc</code> 2411	<code>\millifarad</code> .. 16 , 2221
	<code>\MeVovercsq</code> 2421	<code>\milligram</code> ... 14 , 2200
	<code>\meVovercsq</code> 2421	<code>\millihertz</code> .. 15 , 2262
	<code>\mg</code> 14 , 2297	<code>\millijoule</code> ... 2254
	<code>\MHz</code> 15 , 2280	<code>\milliliter</code> ... 2658
	<code>\mHz</code> 15 , 2280	<code>\millilitre</code> .. 15 , 2243
	<code>\micA</code> 15 , 2272	<code>\millimeter</code> ... 2651
	<code>\micg</code> 14 , 2297	<code>\millimetre</code> .. 14 , 2193
	<code>\micl</code> 15 , 2293 , 2681	<code>\millimole</code> ... 15 , 2205
	<code>\micm</code> 14 , 2311	<code>\millinewton</code> . 16 , 2267
	<code>\micmol</code> 15 , 2286	<code>\millisecond</code> . 15 , 2210
	<code>\Micro</code> 2123	<code>\millisiemens</code> 16 , 2221
	<code>\micro</code> 12 ,	<code>\millisievert</code> 16 , 2267
	2123, 2195, 2203,	<code>\millivolt</code> ... 15 , 2221
	2208, 2214, 2218,	<code>\milliwatt</code> ... 16 , 2221
	2229, 2243, 2248,	<code>\minute</code> 13 , 2338
	2277, 2289, 2294,	<code>\MinveV</code> 2401
	2301, 2313, 2323,	<code>\minveV</code> 2401
	2375, 2653, 2661,	<code>\ml</code> 15 , 2293 , 2681
	2667, 2692, 2712,	<code>\mm</code> 14 , 2311
	2716, 2721, 3040	<code>\mmHg</code> 16 , 2365
	<code>\microampere</code> . 15 , 2216	<code>\mmol</code> 15 , 2286
	<code>\microd</code> 3034	<code>\Molar</code> 2365

<code>\molar</code>	16 , 16 , 2365	<code>\newtonbase</code>	3091	<code>\nmol</code>	15 , 2286
<code>\mole</code>	12 , 1911 , 2205 – 2209 , 2286 – 2290 , 2369 , 2900 – 2903 , 2916 , 2917 , 2979 , 2981 – 2983 , 3089 , 3120	<code>\newtonmetre</code>	3011	<code>\ns</code>	15 , 2318
<code>\molepercubicmetre</code>	2896	<code>\newtonmetrenp</code>	3011	<code>\num</code>	6 , 793 , 829 , 1478 , 1631 , 1734 , 1838 , 2508 , 2742
<code>\molepercubicmetrenp</code>	2896	<code>\newtonpercubicmetre</code>	2933	O	
<code>\mrad</code>	2375	<code>\newtonpercubicmetrenp</code>	2933	<code>\Ohm</code>	2167 , 2233 – 2235
<code>\ms</code>	15 , 2318	<code>\newtonperkilogram</code>	2944	<code>\ohm</code>	13 , 2167 , 2238 – 2240 , 2710 , 2714 , 2719 , 2936
<code>\mV</code>	15 , 2291	<code>\newtonperkilogramnp</code>	2944	<code>\ohmbase</code>	3091
<code>\mv</code>	2674	<code>\newtonpermetre</code>	2996	<code>\ohmmetre</code>	2933
N		<code>\newtonpermetrenp</code>	3011	<code>\Omega</code>	411 , 412
<code>\nA</code>	15 , 2272	<code>\newtonpersquaremetre</code>	2983	P	
<code>\nano</code>	12 , 2123 , 2194 , 2202 , 2207 , 2213 , 2217 , 2222 , 2228 , 2276 , 2288 , 2300 , 2312 , 2322 , 2378 , 2385 , 2391 , 2652 , 2660 , 2691 , 3039	<code>\newtonpersquaremetrenp</code>	2983	<code>\pA</code>	15 , 2272
<code>\nanoampere</code>	15 , 2216	<code>\newunit</code>	17 , 1426 , 1911 – 1917 , 2152 , 2156 – 2166 , 2168 , 2174 – 2180 , 2182 , 2185 , 2188 , 2189 , 2193 – 2231 , 2233 – 2235 , 2238 – 2240 , 2243 – 2271 , 2275 – 2297 , 2299 – 2320 , 2322 – 2324 , 2327 – 2337 , 2340 – 2342 , 2344 , 2347 , 2350 – 2356 , 2361 – 2364 , 2368 – 2371 , 2375 – 2407 , 2409 , 2411 , 2413 , 2415 , 2417 , 2419 , 2421 , 2423 , 2425 , 2427 , 2429 , 2439 , 2440 , 2586 , 2639 , 2640 , 2649 – 2673 , 2677 , 2678 , 2683 , 2686 , 2689 – 2691 , 2694 – 2696 , 2699 – 2701 , 2719 , 2720 , 2722 , 2861 , 2886 – 2894 , 3034 – 3054 , 3056 , 3058 , 3060 , 3062 , 3064 , 3067 – 3079 , 3081 – 3089 , 3092 – 3100 , 3102 , 3105 , 3107 , 3109 , 3111 , 3112 , 3114 – 3120	<code>\pascal</code>	13 , 2167 , 2269 , 2926
<code>\nanobarn</code>	2378	<code>\newunit</code>	17 , 1426 , 1911 – 1917 , 2152 , 2156 – 2166 , 2168 , 2174 – 2180 , 2182 , 2185 , 2188 , 2189 , 2193 – 2231 , 2233 – 2235 , 2238 – 2240 , 2243 – 2271 , 2275 – 2297 , 2299 – 2320 , 2322 – 2324 , 2327 – 2337 , 2340 – 2342 , 2344 , 2347 , 2350 – 2356 , 2361 – 2364 , 2368 – 2371 , 2375 – 2407 , 2409 , 2411 , 2413 , 2415 , 2417 , 2419 , 2421 , 2423 , 2425 , 2427 , 2429 , 2439 , 2440 , 2586 , 2639 , 2640 , 2649 – 2673 , 2677 , 2678 , 2683 , 2686 , 2689 – 2691 , 2694 – 2696 , 2699 – 2701 , 2719 , 2720 , 2722 , 2861 , 2886 – 2894 , 3034 – 3054 , 3056 , 3058 , 3060 , 3062 , 3064 , 3067 – 3079 , 3081 – 3089 , 3092 – 3100 , 3102 , 3105 , 3107 , 3109 , 3111 , 3112 , 3114 – 3120	<code>\pascalbase</code>	3091
<code>\nanofarad</code>	16 , 2221	<code>\newunit</code>	17 , 1426 , 1911 – 1917 , 2152 , 2156 – 2166 , 2168 , 2174 – 2180 , 2182 , 2185 , 2188 , 2189 , 2193 – 2231 , 2233 – 2235 , 2238 – 2240 , 2243 – 2271 , 2275 – 2297 , 2299 – 2320 , 2322 – 2324 , 2327 – 2337 , 2340 – 2342 , 2344 , 2347 , 2350 – 2356 , 2361 – 2364 , 2368 – 2371 , 2375 – 2407 , 2409 , 2411 , 2413 , 2415 , 2417 , 2419 , 2421 , 2423 , 2425 , 2427 , 2429 , 2439 , 2440 , 2586 , 2639 , 2640 , 2649 – 2673 , 2677 , 2678 , 2683 , 2686 , 2689 – 2691 , 2694 – 2696 , 2699 – 2701 , 2719 , 2720 , 2722 , 2861 , 2886 – 2894 , 3034 – 3054 , 3056 , 3058 , 3060 , 3062 , 3064 , 3067 – 3079 , 3081 – 3089 , 3092 – 3100 , 3102 , 3105 , 3107 , 3109 , 3111 , 3112 , 3114 – 3120	<code>\pascalsecond</code>	2920
<code>\nanogram</code>	14 , 2200	<code>\newunit</code>	17 , 1426 , 1911 – 1917 , 2152 , 2156 – 2166 , 2168 , 2174 – 2180 , 2182 , 2185 , 2188 , 2189 , 2193 – 2231 , 2233 – 2235 , 2238 – 2240 , 2243 – 2271 , 2275 – 2297 , 2299 – 2320 , 2322 – 2324 , 2327 – 2337 , 2340 – 2342 , 2344 , 2347 , 2350 – 2356 , 2361 – 2364 , 2368 – 2371 , 2375 – 2407 , 2409 , 2411 , 2413 , 2415 , 2417 , 2419 , 2421 , 2423 , 2425 , 2427 , 2429 , 2439 , 2440 , 2586 , 2639 , 2640 , 2649 – 2673 , 2677 , 2678 , 2683 , 2686 , 2689 – 2691 , 2694 – 2696 , 2699 – 2701 , 2719 , 2720 , 2722 , 2861 , 2886 – 2894 , 3034 – 3054 , 3056 , 3058 , 3060 , 3062 , 3064 , 3067 – 3079 , 3081 – 3089 , 3092 – 3100 , 3102 , 3105 , 3107 , 3109 , 3111 , 3112 , 3114 – 3120	<code>\pebi</code>	17 , 2431 , 3063
<code>\nanoliter</code>	2658	<code>\newunit</code>	17 , 1426 , 1911 – 1917 , 2152 , 2156 – 2166 , 2168 , 2174 – 2180 , 2182 , 2185 , 2188 , 2189 , 2193 – 2231 , 2233 – 2235 , 2238 – 2240 , 2243 – 2271 , 2275 – 2297 , 2299 – 2320 , 2322 – 2324 , 2327 – 2337 , 2340 – 2342 , 2344 , 2347 , 2350 – 2356 , 2361 – 2364 , 2368 – 2371 , 2375 – 2407 , 2409 , 2411 , 2413 , 2415 , 2417 , 2419 , 2421 , 2423 , 2425 , 2427 , 2429 , 2439 , 2440 , 2586 , 2639 , 2640 , 2649 – 2673 , 2677 , 2678 , 2683 , 2686 , 2689 – 2691 , 2694 – 2696 , 2699 – 2701 , 2719 , 2720 , 2722 , 2861 , 2886 – 2894 , 3034 – 3054 , 3056 , 3058 , 3060 , 3062 , 3064 , 3067 – 3079 , 3081 – 3089 , 3092 – 3100 , 3102 , 3105 , 3107 , 3109 , 3111 , 3112 , 3114 – 3120	<code>\pebid</code>	3054
<code>\nanometer</code>	2651	<code>\newunit</code>	17 , 1426 , 1911 – 1917 , 2152 , 2156 – 2166 , 2168 , 2174 – 2180 , 2182 , 2185 , 2188 , 2189 , 2193 – 2231 , 2233 – 2235 , 2238 – 2240 , 2243 – 2271 , 2275 – 2297 , 2299 – 2320 , 2322 – 2324 , 2327 – 2337 , 2340 – 2342 , 2344 , 2347 , 2350 – 2356 , 2361 – 2364 , 2368 – 2371 , 2375 – 2407 , 2409 , 2411 , 2413 , 2415 , 2417 , 2419 , 2421 , 2423 , 2425 , 2427 , 2429 , 2439 , 2440 , 2586 , 2639 , 2640 , 2649 – 2673 , 2677 , 2678 , 2683 , 2686 , 2689 – 2691 , 2694 – 2696 , 2699 – 2701 , 2719 , 2720 , 2722 , 2861 , 2886 – 2894 , 3034 – 3054 , 3056 , 3058 , 3060 , 3062 , 3064 , 3067 – 3079 , 3081 – 3089 , 3092 – 3100 , 3102 , 3105 , 3107 , 3109 , 3111 , 3112 , 3114 – 3120	<code>\per</code>	11 , 1744 , 2369 , 2384 – 2399 , 2401 – 2406 , 2408 , 2410 , 2412 , 2414 , 2416 , 2418 , 2420 , 2422 , 2424 , 2426 , 2428 , 2430 , 2575 , 2803 , 2896 , 2898 , 2900 , 2902 , 2904 , 2907 , 2910 , 2912 , 2914 , 2916 , 2918 , 2920 , 2923 , 2927 , 2930 , 2932 , 2934 , 2938 , 2940 , 2942 , 2944 , 2946 , 2948 , 2950 , 2952 , 2955 , 2958 , 2961 , 2963 , 2965 , 2967 , 2969 , 2972 , 2975 , 2977 , 2979 , 2982 , 2987 , 2990 , 2992 , 2994 , 2996 , 2998 , 3001 , 3004 , 3007 , 3010 , 3012 , 3017 , 3020 , 3022 , 3025 , 3028 , 3031
<code>\nanometre</code>	14 , 2193	<code>\newunit</code>	17 , 1426 , 1911 – 1917 , 2152 , 2156 – 2166 , 2168 , 2174 – 2180 , 2182 , 2185 , 2188 , 2189 , 2193 – 2231 , 2233 – 2235 , 2238 – 2240 , 2243 – 2271 , 2275 – 2297 , 2299 – 2320 , 2322 – 2324 , 2327 – 2337 , 2340 – 2342 , 2344 , 2347 , 2350 – 2356 , 2361 – 2364 , 2368 – 2371 , 2375 – 2407 , 2409 , 2411 , 2413 , 2415 , 2417 , 2419 , 2421 , 2423 , 2425 , 2427 , 2429 , 2439 , 2440 , 2586 , 2639 , 2640 , 2649 – 2673 , 2677 , 2678 , 2683 , 2686 , 2689 – 2691 , 2694 – 2696 , 2699 – 2701 , 2719 , 2720 , 2722 , 2861 , 2886 – 2894 , 3034 – 3054 , 3056 , 3058 , 3060 , 3062 , 3064 , 3067 – 3079 , 3081 – 3089 , 3092 – 3100 , 3102 , 3105 , 3107 , 3109 , 3111 , 3112 , 3114 – 3120	<code>\percent</code>	13 , 2338
<code>\nanomole</code>	15 , 2205	<code>\newunit</code>	17 , 1426 , 1911 – 1917 , 2152 , 2156 – 2166 , 2168 , 2174 – 2180 , 2182 , 2185 , 2188 , 2189 , 2193 – 2231 , 2233 – 2235 , 2238 – 2240 , 2243 – 2271 , 2275 – 2297 , 2299 – 2320 , 2322 – 2324 , 2327 – 2337 , 2340 – 2342 , 2344 , 2347 , 2350 – 2356 , 2361 – 2364 , 2368 – 2371 , 2375 – 2407 , 2409 , 2411 , 2413 , 2415 , 2417 , 2419 , 2421 , 2423 , 2425 , 2427 , 2429 , 2439 , 2440 , 2586 , 2639 , 2640 , 2649 –		

<code>\peta</code>	12 , 2141 , 3050		
<code>\petad</code>	3043		
<code>\pg</code>	14 , 2297		
<code>\pico</code>	12 , 2123 , 2193 , 2201 , 2206 , 2212 , 2216 , 2227 , 2275 , 2287 , 2299 , 2311 , 2321 , 2379 , 2386 , 2392 , 2651 , 2659 , 2690 , 3038		
<code>\picoampere</code> ..	15 , 2216		
<code>\picobarn</code>	2378		
<code>\picod</code>	3034		
<code>\picofarad</code> ...	16 , 2221		
<code>\picogram</code>	14 , 2200		
<code>\picoliter</code>	2658		
<code>\picom</code>	14 , 2311		
<code>\picometer</code>	2651		
<code>\picometre</code> ...	14 , 2193		
<code>\picomole</code>	15 , 2205		
<code>\picosecond</code> ..	14 , 2210		
<code>\pl</code>	2681		
<code>\pm</code>	517 , 973 , 2025		
<code>\pmol</code>	15 , 2286		
<code>\pnt</code>	2757		
<code>\power</code>	2803		
<code>\prime</code>	432 – 435		
<code>\providepower</code>	17 , 1452		
<code>\provideprefix</code>	17 , 1439		
<code>\provideunit</code>	17 , 1426 , 2171 , 2298 , 2321		
<code>\ps</code>	14 , 2318		
R			
<code>\rad</code>	14 , 2325		
<code>\radian</code>	13 , 2188 , 2376 , 2904 , 2905 , 2910 , 2911		
<code>\radianbase</code>	3091		
<code>\radianpersecond</code>	2908		
<code>\radianpersecondnp</code>	2908		
<code>\radianpersquaresecond</code>	2896		
<code>\radianpersquaresecondnp</code>	2896		
<code>\reciprocal</code>	2803 , 2897 , 2901 , 2909 , 2911 , 2917 , 2925 , 2931 , 2935 , 2945 , 2947 , 2949 , 2951 , 2953 , 2960 , 2964 , 2966 , 2968 , 2970 , 2974 , 2978 , 2981 , 2983 , 2993 , 2995 , 3003 , 3006 , 3009 , 3011 , 3014 , 3021 , 3027 , 3029 , 3033 , 3067 , 3069 , 3073 , 3075 – 3078 , 3080 , 3082 , 3087 , 3092 , 3094 , 3096 , 3101 , 3103 , 3108 , 3110 , 3111		
<code>\rem</code>	14 , 2325		
<code>\renewnosepunit</code>	2586		
<code>\renewpower</code>	17 , 1452 , 2851		
<code>\renewprefix</code> .	17 , 1439		
<code>\renewunit</code>	17 , 1426 , 2153 , 2587 , 2646 , 2692 , 2693 , 2863		
<code>\roentgen</code>	14 , 2325		
<code>\rp</code>	2803 , 3000 , 3089 , 3120		
<code>\rpcubed</code>	2812		
<code>\rpcubic</code>	2803 , 2903 , 2913 , 2915 , 2921 , 2928 , 2943 , 2997 , 3019 , 3023 , 3027 , 3033 , 3098 , 3101 , 3106		
<code>\rpcubicmetreperkilogram</code>	2996		
<code>\rpcubicmetrepersecond</code>	3024		
<code>\rperminute</code>	2886		
<code>\rpfourth</code> ..	2803 , 3116		
<code>\rpsquare</code>	2803 , 2919 , 2933 , 2939 , 2941 , 2962 , 2974 , 2976 , 2991 , 2993 , 3000 , 3068 , 3071 , 3081 , 3085 , 3093 , 3103 , 3106 , 3108 , 3113 , 3115		
<code>\rpsquared</code> ..	2812 , 2899 , 2905 , 2957 , 2989 , 3070 , 3080 , 3095 – 3097 , 3110 , 3111 , 3113 , 3118		
<code>\rpsquaremetreperkilogram</code>	2996		
S			
<code>\Sec</code>	14 , 2318		
<code>\second</code>	12 , 14 , 1911 , 2210 – 2215 , 2318 – 2324 , 2397 – 2399 , 2683 , 2896 – 2899 , 2904 , 2905 , 2907 , 2909 – 2911 , 2923 , 2925 , 2926 , 2929 , 2952 , 2953 , 2955 , 2957 , 2958 , 2960 , 2969 , 2970 , 2972 , 2974 , 2987 , 2989 , 2992 , 2993 , 3017 , 3019 – 3021 , 3028 , 3029 , 3031 , 3033 , 3069 , 3070 , 3073 , 3074 , 3080 , 3089 , 3094 – 3099 , 3101 , 3103 , 3106 , 3108 , 3110 , 3111 , 3113 , 3118 , 3120		
<code>\sek</code>	2681		
<code>\setMathCelsius</code>	2588		
<code>\setMathDegree</code> .	2588		
<code>\setMathmu</code>	2588		
<code>\setMathOmega</code> ..	2588		
<code>\setTextCelsius</code>	2588		
<code>\setTextDegree</code> .	2588		
<code>\setTextmu</code>	2588		
<code>\setTextOmega</code> ..	2588		
<code>\SI</code>	9 , 1421 , 1477 , 2497 , 2499 , 2872 , 2874		
<code>\si@addtocname</code> .	62 , 1959		
<code>\si@addtolist</code> .	56 , 169 , 398 , 2077 , 2079 , 2082 , 2094		
<code>\si@ang@astrosign</code>	1150 , 1171		
<code>\si@ang@decimalsign</code>	1150 , 1169		
<code>\si@ang@degsg</code> ...	1160		
<code>\si@ang@killdegree</code>	1150		
<code>\si@ang@killminute</code>	1150		
<code>\si@ang@killsecond</code>	1150		
<code>\si@ang@minnum</code> .	1175		
<code>\si@ang@mins</code> ...	1160		

<code>\si@ang@movesign</code>	<code>\si@fam@getmfam</code> . 641	<code>\si@fracfalse</code> ... 351
.. 1170 , 1210 , 1215	<code>\si@fam@ifbinline</code>	<code>\si@fractrue</code>
<code>\si@ang@num</code> . 1175 , 660 , 773 354 , 360 , 364
1176 , 1197 , 1208	<code>\si@fam@ifbmaths</code>	<code>\si@frc@displen</code> .
<code>\si@ang@pad</code> . 1182 , 653 , 664 , 769	. 580 , 592 , 596 , 600
1184 , 1191 , 1202	<code>\si@fam@ifbtext</code> .	<code>\si@frc@frac</code>
<code>\si@ang@padlargefalse</code> 653 , 662 , 778 371 , 557 , 630
..... 318	<code>\si@fam@ifitext</code> .	<code>\si@frc@mathsnf</code> .
<code>\si@ang@padlargetrue</code> 666 , 786 586 , 590
. 325 , 330 , 335 , 340	<code>\si@fam@italic</code> ..	<code>\si@frc@nice</code>
<code>\si@ang@padsmallfalse</code> 783 , 1970 374 , 382 , 557
..... 317	<code>\si@fam@maths</code> 684 , 1979	<code>\si@frc@nicefrac</code>
<code>\si@ang@padsmalltrue</code>	<code>\si@fam@mode</code> 566 , 580
. 321 , 329 , 334 , 339	669 , 796 , 1144 , 2503	<code>\si@frc@sfrac</code> ...
<code>\si@ang@parse</code> ...	<code>\si@fam@set</code> . 678 , 1968 386 , 390 , 557
..... 1148 , 1149	<code>\si@fam@setbold</code> .	<code>\si@frc@slash</code> ...
<code>\si@ang@secnum</code> . 1175	. 763 , 770 , 774 , 779 356 , 557 , 638
<code>\si@ang@secs</code> ... 1160	<code>\si@fam@settrue</code> . 679	<code>\si@frc@ssuplen</code> .
<code>\si@ang@set</code> 1149 , 1159	<code>\si@fam@sf</code> ... 641 , 687 580 , 595 –
<code>\si@ang@signlessnum</code>	<code>\si@fam@text</code> 684 , 1970	598 , 604 , 619 , 621
..... 1187 ,	<code>\si@fam@tt</code> ... 641 , 694	<code>\si@frc@suplen</code> ..
1194 , 1195 , 1208	<code>\si@fileprefix</code> ..	. 580 , 594 , 598 , 602
<code>\si@anglesep</code> 186 , 1200 2041 ,	<code>\si@frc@textlen</code> .
<code>\si@cfgextension</code>	2044 , 2047 , 2050 ,	.. 580 , 593 , 597 ,
.. 2518 , 2521 , 2522	2518 , 2520 , 2522	601 , 618 , 621 , 622
<code>\si@debugfalse</code> .. 144	<code>\si@fix@cdot</code> 509	<code>\si@frc@textnf</code> 588 , 616
<code>\si@debugtrue</code> 157 , 161	<code>\si@fix@comma</code> ... 509	<code>\si@frc@ugly</code>
<code>\si@decimalsign</code> .	<code>\si@fix@fullstop</code> 509 378 , 394 , 627
184 , 1059 , 1154 ,	<code>\si@fix@med</code> 504	<code>\si@gensymbtrue</code> 2704
1169 , 1361 , 1368 ,	<code>\si@fix@medium</code> .. 504	<code>\si@gobblethree</code> .
1397 , 1407 , 2757	<code>\si@fix@minus</code> ... 515	.. 1563 , 1572 , 1578
<code>\si@denlbrac</code> 368 , 1879	<code>\si@fix@mp</code> 515	<code>\si@ifdefinable</code> 52 ,
<code>\si@denrbrac</code> 368 , 1880	<code>\si@fix@none</code> 522	1427 , 1431 , 1436 ,
<code>\si@digitsep</code>	<code>\si@fix@period</code> .. 509	1440 , 1444 , 1449 ,
182 , 1120 , 1138 , 1418	<code>\si@fix@plus</code> 515	1453 , 1457 , 1462
<code>\si@emclash</code>	<code>\si@fix@pm</code> 515	<code>\si@ifloaded</code>
.... 2061 , 2481 ,	<code>\si@fix@slash</code> ... 521 2043 , 2046 ,
2483 , 2514 , 2516	<code>\si@fix@space</code> ... 504	2477 , 2480 , 2482 ,
<code>\si@emulate</code>	<code>\si@fix@stop</code> 509	2513 , 2515 , 2517
... 165 , 2065 , 2066	<code>\si@fix@ten</code> 519	<code>\si@ifnotmtarg</code> 69 ,
<code>\si@eVcorra</code> . 449 , 2359	<code>\si@fix@thick</code> ... 504	813 , 1177 , 1185 ,
<code>\si@eVcorrb</code> . 449 , 2360	<code>\si@fix@thin</code> 504	1192 , 1468 , 1473 ,
<code>\si@eVspacea</code> ... 2357	<code>\si@fix@times</code> ... 509	1476 , 1480 , 1558 ,
<code>\si@eVspaceb</code> ... 2357	<code>\si@fix@two</code> .. 519 ,	1630 , 1658 , 2507
<code>\si@expanddefault</code>	3055 , 3057 , 3059 ,	<code>\si@inlinebtextfalse</code>
..... 2069 , 2099	3061 , 3063 , 3065 201
<code>\si@exppower</code>	<code>\si@frac</code> 356 ,	<code>\si@inlinebtexttrue</code>
221 , 881 , 1319 , 1837	371 , 374 , 378 , 204
<code>\si@expproduct</code> ..	382 , 386 , 390 ,	<code>\si@load</code> 397
.... 220 , 878 , 1316	394 , 557 , 1856 ,	<code>\si@loademfile</code> ..
<code>\si@extension</code> 2041 ,	1860 , 1869 , 2509 2054 , 2067
2044 , 2047 , 2050		<code>\si@loadfile</code>
<code>\si@fam@bold</code> 763 , 1970	 1938 , 2045 ,
		2053 , 2060 , 2097

<code>\si@loadlocales</code> .	<code>\si@logmintrue</code> ..	<code>\si@num@format</code> ..
... 453 , 2101 , 2102	<code>\si@lognonefalse</code> 146 816 , 858 , 1294
<code>\si@loc@load</code>	<code>\si@lognonetrue</code> . 149	<code>\si@num@gensign</code> .
.. 1934 , 2103 , 2106	<code>\si@mathnumdefault</code> 918 , 967
<code>\si@loc@set</code> 214 , 1989	<code>\si@num@ifchr</code> 822 ,
456 , 464 , 1940 , 2113	<code>\si@mathscelsius</code> 436	832 , 895 , 896 ,
<code>\si@loc@ssetup</code> 1934	<code>\si@mathsdefault</code>	968 , 969 , 1028 ,
<code>\si@locale</code> 208 ,	1038 , 1078 , 1263
<code>\si@loctolang</code> ...	703 , 726 , 750 , 759	<code>\si@num@ifextra</code> .
453 , 2105 , 2107 , 2114	<code>\si@mathsdegree</code> 1054 , 1061 ,
<code>\si@log@debug</code> . 98 , 425 , 439 , 543	1066 , 1333 , 1337
110 , 117 , 120 , 121 ,	<code>\si@mathsminute</code> . 425	<code>\si@num@iffive</code> ..
132 , 134 , 141 ,	<code>\si@mathsmu</code> 1088 , 1092 , 1128
168 , 541 , 553 , 413 , 554 , 2884	<code>\si@num@int</code>
644 , 648 , 683 ,	<code>\si@mathsOmega</code> 409 , 555	.. 1055 , 1083 , 1334
686 , 688 , 695 ,	<code>\si@mathsringA</code> 440 , 549	<code>\si@num@intfmt</code> ..
701 , 709 , 711 ,	<code>\si@mathsrms</code> 1086 , 1089 , 1100
718 , 724 , 733 ,	... 208 , 2011 , 2019	<code>\si@num@intsep</code> ..
735 , 742 , 748 ,	<code>\si@mathssecond</code> . 425 1102 ,
757 , 766 , 771 ,	<code>\si@mathssf</code>	1105 , 1108 , 1115
775 , 780 , 785 ,	. 208 , 690 , 713 , 737	<code>\si@num@mant</code>
788 , 797 , 864 ,	<code>\si@mathstt</code> 853 , 861 ,
904 , 906 , 909 ,	. 208 , 697 , 720 , 744	869 , 876 , 884 ,
925 , 933 , 944 ,	<code>\si@noload</code>	910 , 1306 , 1314 , 1322
946 , 972 , 981 ,	<code>\si@num</code>	<code>\si@num@mantexp</code> .
993 , 997 , 1008 ,	798 , 866 , 894 , 1304
1015 , 1035 , 1043 ,	800 , 1211 , 1217 ,	<code>\si@num@mantout</code> .
1046 , 1071 , 1145 ,	1295 , 2749 , 2755 853 , 863 ,
1178 , 1186 , 1193 ,	<code>\si@num@arg</code> .. 858 ,	868 , 871 , 1357 , 1401
1241 , 1265 , 1267 ,	866 , 886 , 904 ,	<code>\si@num@out</code>
1474 , 1477 , 1496 ,	906 , 909 , 926 ,	.. 853 , 868 , 893 ,
1501 , 1556 , 1574 ,	934 , 944 , 946 ,	1308 , 1359 , 1364 ,
1588 , 1609 , 1648	956 , 959 , 972 ,	1369 , 1404 , 1408
<code>\si@log@err</code>	981 , 994 , 997 ,	<code>\si@num@padleadfalse</code>
... 78 , 827 , 886 ,	1008 , 1015 , 1030 , 240
898 , 1030 , 1039 ,	1035 , 1039 , 1043 ,	<code>\si@num@padleadtrue</code>
1324 , 1429 , 1432 ,	1046 , 1302 , 1324 244 ,
1442 , 1445 , 1455 ,	<code>\si@num@chrstr</code> 837 , 843	248 , 260 , 265 , 270
1458 , 1849 , 2049 ,	<code>\si@num@dec</code>	<code>\si@num@padtrailfalse</code>
2056 , 2062 , 2520	.. 1062 , 1123 , 1338 241
<code>\si@log@inf</code>	<code>\si@num@decfmt</code> . 1123	<code>\si@num@padtrailtrue</code>
... 78 , 574 , 790 ,	<code>\si@num@decimalhook</code> 252 ,
1942 , 1953 , 2120 ,	.. 1052 , 1060 , 1157	256 , 261 , 266 , 271
2634 , 2643 , 2728 ,	<code>\si@num@digits</code> 870 ,	<code>\si@num@sepdigits</code>
2729 , 3123 , 3124	875 , 1000 , 1307 , 1313	1022 , 1025 , 1053 ,
<code>\si@log@warn</code> 78 , 142 ,	<code>\si@num@exp</code> .. 853 ,	1300 , 1301 , 1310
163 , 170 , 196 , 206 ,	860 , 874 , 885 ,	<code>\si@num@sign</code> . 867 ,
235 , 273 , 312 ,	907 , 955 , 1312 , 1323	873 , 915 , 1305 , 1311
342 , 366 , 396 ,	<code>\si@num@expout</code> ..	<code>\si@num@signexpfalse</code>
578 , 956 , 959 ,	. 853 , 862 , 882 , 1320 280
975 , 984 , 988 ,	<code>\si@num@extra</code> .. 1066	<code>\si@num@signexptrue</code>
1659 , 1956 , 2116 ,	<code>\si@num@fiint</code> .. 1100 291 ,
2818 , 2854 , 2866	<code>\si@num@five</code> ... 1092	295 , 300 , 305 , 310
<code>\si@logminfalse</code> . 145		

<code>\si@num@signmantfalse</code>	176, 178, 180,	903, 1036, 1078,
.....	279	1264, 2091, 2788
<code>\si@num@signmanttrue</code>	220–223, 276, 367	<code>\si@sym@celsius</code> .
.....	283, <u>531</u> , 2182,
287, 299, 304, 309	<code>\si@out@minus</code> ...	2185, 2720, 2780
<code>\si@num@split</code> 1974, <u>1984</u>	<code>\si@sym@degree</code> ..
.....	1004, <u>1027</u>	531, 1150, 1161,
<code>\si@num@valid</code>	814, <u>821</u>	1184, 1191, 1198,
<code>\si@num@xdef</code> .	811, <u>820</u>	1900, 2344, 2347,
<code>\si@num@decimal</code> ..	1985, 2749, 2755	2722, 2781, 2782
<u>216</u> , 218, 1028, 1032	<code>\si@out@text</code>	<code>\si@sym@minute</code> <u>531</u> ,
<code>\si@num@exp</code>	563, 1835, 1892,	1151, 1162, 1182,
. <u>216</u> , 217, 896, 900	<u>1960</u> , 1990, 2778	1189, 2350, 2886
<code>\si@num@extra</code>	<code>\si@packagecheck</code> . <u>32</u>	<code>\si@sym@mu</code> <u>531</u> , 1901,
.... <u>216</u> , 218, 1078	<code>\si@per</code>	2132, 2135, 2721
<code>\si@num@gobble</code> ...	1744, 2803, 2805,	<code>\si@sym@Omega</code> ...
..... <u>216</u> , 217, 895	2808, 2809, 2811	<u>531</u> , 2168, 2171, 2719
<code>\si@num@list</code>	<code>\si@prefixnumfalse</code>	<code>\si@sym@ringA</code> ...
. <u>216</u> , 218, 889, 1327 226	... <u>531</u> , 1902, 2327
<code>\si@num@sign</code> <u>216</u> , 217,	<code>\si@prefixnumtrue</code>	<code>\si@sym@second</code> ..
968, 969, 1038, 1040 229,	<u>531</u> , 1152, 1163,
<code>\si@num@valid</code>	233, 3034–3053,	1180, 2351, 2887
. <u>217</u> , 822, 828, 1263	3055, 3057, 3059,	<code>\si@symbol</code> <u>523</u> , 531–537
<code>\si@old@OHMfalse</code> 2636	3061, 3063, 3065	<code>\si@tab@begin</code> ...
<code>\si@opt@boolkey</code> .	<code>\si@prefixpower</code> <u>1227</u> , <u>1234</u>
.. <u>115</u> , 164, 187, <u>223</u> , 1837,	<code>\si@tab@centre</code> ..
189, 197, 198, 207,	3055, 3057, 3059, <u>1351</u> , <u>1355</u>
219, 343, 345, 444	3061, 3063, 3065	<code>\si@tab@end</code> . <u>1228</u> ,
<code>\si@opt@choicekey</code>	<code>\si@prefixproduct</code>	<u>1250</u> , <u>1251</u> , <u>1282</u>
..... <u>118</u> , 123, <u>222</u> , 1835	<code>\si@tab@format</code> ..
143, 166, 190, 200,	<code>\si@requirecfgs</code> <u>1296</u> , <u>1347</u>
225, 238, 277, <u>2052</u> ,	<code>\si@tab@gettok</code> ..
315, 348, 369, 2562	2192, 2274, <u>2367</u> , <u>1242</u> , <u>1243</u>
<code>\si@opt@cmdkey</code> ..	2374, 2725, 2802	<code>\si@tab@mantout</code> .
. <u>111</u> , 344, 454, 464	<code>\si@SI</code> . 1423–1425, <u>1466</u> <u>1298</u> , 1308,
<code>\si@opt@cmdkeys</code> .	<code>\si@sign</code>	<u>1328</u> , <u>1358</u> , 1402
.. <u>111</u> , 208, 213,	925, 927, 933, 935	<code>\si@tab@next</code> ... <u>1243</u>
214, 216, 368,	<code>\si@sis@addtolocale</code>	<code>\si@tab@num@format</code>
397, 409, 413, <u>2783</u> <u>1294</u> , <u>1299</u>
425, 436, 440, 453	<code>\si@sis@num</code> <u>2742</u>	<code>\si@tab@num@sepdigits</code>
<code>\si@opt@compatkey</code>	<code>\si@sis@numstar</code> <u>2742</u> <u>1301</u> , <u>1332</u>
..... <u>130</u> ,	<code>\si@sis@savefont</code>	<code>\si@tab@numout</code> ..
465–471, 473–503	.. <u>2766</u> , 2769–2774 <u>1286</u> , <u>1293</u>
<code>\si@opt@disablekey</code>	<code>\si@siu@setup</code> .. <u>2815</u>	<code>\si@tab@numtoks</code> .
..... <u>135</u> , <u>138</u> ,	<code>\si@slash</code> <u>367</u> , 563 <u>1231</u> ,
173, 400, 403,	<code>\si@slashfalse</code> .. 350	<u>1237</u> , <u>1266</u> , <u>1295</u>
406, 446, 458, 461	<code>\si@slashttrue</code> ... 355	<code>\si@tab@org@sepdigits</code>
<code>\si@opt@key</code> .. <u>108</u> ,	<code>\si@switchfalse</code> <u>1300</u> , 1310
142, 209–212,	.. 831, 834, 865,	<code>\si@tab@othertok</code>
215, 398, 410,	1003, 1068, 1240, <u>1268</u> , <u>1276</u>
414, 427–429,	1303, 2087, 2786	<code>\si@tab@postbox</code> .
437, 441, 451, 452	<code>\si@switchtrue</code> <u>1344</u> ,
<code>\si@opt@xchoicekey</code> 826, 847,	1354, 1360, 1363,
..... <u>122</u> ,		

1367, 1371–1373,	627, 680, 717,	\si@textsf
1375, 1403, 1406	741, 832, 837,	. 213, 692, 715, 739
\si@tab@postdim .	844, 849, 951,	\si@texttt
.... 1378, 1396,	963, 967, 1000,	. 213, 699, 722, 746
1398, 1403, 1406	1044, 1057, 1061,	\si@unitsep
\si@tab@posttoks	1062, 1065, 1123,	... 176, 1662, 1891
..... 1231,	1336–1338, 1343,	\si@unitSIdef .. 2572
1239, 1278, 1288	1763, 1774, 1787,	\si@unitspace 176, 1890
\si@tab@prea 1385, 1400	2052, 2069, 2101	\si@unt@addprefix
\si@tab@preb 1385, 1400	\si@tempbox 634, 635, 1686, 1689
\si@tab@prebox ..	1344, 1383, 1384,	\si@unt@addstack
.... 1344, 1354,	1397, 1398, 1418, 1796, 1802
1356, 1371, 1372,	1420, 1487, 1488	\si@unt@addtostack
1375, 1376, 1400	\si@tempc 1613, 1662,
\si@tab@predim ..	844, 920, 943,	1689, 1734, 1763
.. 1378, 1390, 1392	954, 1000, 1055,	\si@unt@addvalsep
\si@tab@pretoks .	1065, 1335, 2069 1507
..... 1231,	\si@tempcnta	\si@unt@addvaluesep
1238, 1280, 1284	1291, 1296, 1348,	.. 1499, 1507, 1639
\si@tab@rewrite .	1390, 1394, 1396,	\si@unt@checkstack
.. 1222, 1223, 1225	1411, 1414, 1692,	1668, 1677, 1732,
\si@tab@right ...	1696, 1698, 1699	1767, 1776, 1785,
..... 1349, 1382	\si@tempcntb 1291,	1800, 1806, 1809–
\si@tab@sepcorr .	1411, 1412, 1415,	1811, 1813, 1820
.. 1391, 1399, 1410	1420, 1698, 1699	\si@unt@countprefix
\si@tab@format 344, 1297	\si@tempdima 1684, 1690
\si@tempa 28, 32, 119,	1378, 1384, 1390,	\si@unt@defpower
144, 191, 199,	1396, 1739, 1740 1454, 1459,
226, 240, 279,	\si@tempdimb ... 1378	1460, 1463, 1587
317, 350, 369,	\si@temptoks	\si@unt@defprefix
538, 547, 548, 62, 1643, 1441, 1446,
627, 653, 666,	1647, 1648, 1655,	1447, 1450, 1573
680, 710, 734,	1656, 1944, 1947	\si@unt@defunit .
801, 816, 832,	\si@textcelsius 1428, 1433,
846, 920, 941– 436, 2611	1434, 1437, 1555
943, 950, 953,	\si@textdefault .	\si@unt@depthcnt
954, 967, 1001, 213, 1504,
1006, 1009, 1020,	705, 728, 752, 761	1521, 1605, 1608,
1047, 1054, 1055,	\si@textdegree ..	1609, 1618, 1619
1065, 1083, 1115,	. 425, 438, 542, 2622	\si@unt@final ...
1225, 1333, 1334,	\si@textminute .. 425	.. 1506, 1548, 1620
1341, 1667, 1676,	\si@textmodefalse 191	\si@unt@first ...
1678, 1709, 1712,	\si@textmodetrue 194 1624, 1629
1731, 1733, 1763,	\si@textmu	\si@unt@firstfalse
1774, 1787, 1896,	413, 545, 2600, 2884 1640
2065, 2069, 2101,	\si@textnumdefault	\si@unt@firstorsecond
2591, 2592, 2602, 214, 1988 1611,
2603, 2613, 2614,	\si@textOmega ...	1622, 1682, 1748
2624, 2625, 2815 409, 546, 2589	\si@unt@firsttrue
\si@tempb	\si@textringA 440, 550 1532
144, 191, 199,	\si@textrm	\si@unt@fracout .
226, 240, 279,	... 213, 2015, 2020 1816, 1844
317, 350, 369,	\si@textsecond .. 425	\si@unt@holdspace
	 1794, 1802

<code>\si@unt@holdstacka</code>	<code>\si@unt@powerdim</code>	<code>\si@unt@stkpower</code>
..... 1544 , 1802 1537 ,	.. 1615 , 1665 , 1716
<code>\si@unt@holdstackb</code>	1703 , 1715 , 1722 ,	<code>\si@unt@stkpwr</code> . 1716
..... 1545 , 1802	1730 , 1734 , 1737 ,	<code>\si@unt@sym</code>
<code>\si@unt@ifliteral</code>	1739 – 1742 , 1761	.. 1900 – 1902 , 1905
.. 1483 , 1495 , 1612	<code>\si@unt@prefix</code> ..	<code>\si@unt@third</code> ...
<code>\si@unt@incnt</code> 1583 , 1681 1549 , 1657
..... 1790 , 1799	<code>\si@unt@prefixcnt</code>	<code>\si@unt@unit</code> 1568 , 1603
<code>\si@unt@init</code> 1540 ,	<code>\si@unt@unitcnta</code> 1521
.. 1503 , 1521 , 1606	1690 , 1833 , 1838	<code>\si@unt@unitcntb</code>
<code>\si@unt@invpower</code>	<code>\si@unt@prefixout</code> 1521 , 1873
..... 1706 , 1738	... 1832 , 1841 ,	<code>\si@unt@unithook</code>
<code>\si@unt@invprefix</code>	1855 , 1858 , 1867	.. 1603 , 1632 , 2583
..... 1690	<code>\si@unt@preplussp</code>	<code>\si@valuesep</code> . 176 ,
<code>\si@unt@lastadda</code> 1771 , 1774	634 , 1518 , 1520 ,
..... 1546 , 1763	<code>\si@unt@prepowerfalse</code>	2508 , 2561 , 2801
<code>\si@unt@lastaddb</code> 1535 , 1717	<code>\si@xifmtarg</code>
..... 1547 , 1763	<code>\si@unt@prepowertrue</code>	69
<code>\si@unt@litoutfalse</code> 1710	<code>\si@xspacefalse</code> .
..... 1531	<code>\si@unt@printunit</code> 1472 , 2577
<code>\si@unt@litouttrue</code>	.. 1475 , 1481 , 1493	<code>\SIdecimalsign</code> . 2763
..... 1498 , 2583	<code>\si@unt@recip</code> .. 1754	<code>\SIdefaultMfam</code> . 2772
<code>\si@unt@litpower</code>	<code>\si@unt@reciptest</code>	<code>\SIdefaultNfam</code> . 2772
.. 1597 , 1700 , 1927 1679 , 1754	<code>\SIdefaultTfam</code> . 2772
<code>\si@unt@littesttrue</code>	<code>\si@unt@second</code> ..	<code>\siemens</code> . 13 , 2167 , 2231
..... 1486 1626 , 1657	<code>\siemensbase</code> ... 3107
<code>\si@unt@litvalsep</code>	<code>\si@unt@setopts</code> .	<code>\sievert</code> . 13 , 2167 , 2271
..... 1507 1634 , 1641	<code>\sievertbase</code> ... 3107
<code>\si@unt@nonlatin</code>	<code>\si@unt@setSIopts</code>	<code>\SIgroupfourfalse</code>
..... 1886 , 1895 1641 2758
<code>\si@unt@normout</code> .	<code>\si@unt@SIopts</code> ..	<code>\SIgroupfourtrue</code> 2758
.. 1818 , 1840 , 1865	.. 1466 , 1651 , 1655	<code>\SIMathrm</code>
<code>\si@unt@notabg</code> . 1872	<code>\si@unt@spacecheck</code> 2769
<code>\si@unt@notambig</code> 1670 , 1675	<code>\SIMathsf</code>
..... 1845 , 1872	<code>\si@unt@spaceout</code> 2769
<code>\si@unt@numfalse</code> 1471 1830 ,	<code>\SIobeyboldfalse</code> 2740
<code>\si@unt@numtrue</code> .	1842 , 1859 , 1868	<code>\SIobeyboldtrue</code> 2740
..... 1465 , 1479	<code>\si@unt@spstack</code> .	<code>\SIproductsign</code> . 2763
<code>\si@unt@out</code> 1518 , 1521 , 1831	<code>\SIsetup</code>
.. 558 , 559 , 562 ,	<code>\si@unt@stack</code> ...	2815
564 , 566 , 567 , 1781 , 1784	<code>\sisetup</code> ... 18 , 104 ,
569 , 570 , 1487 ,	<code>\si@unt@stacka</code> ..	133 , 209 – 212 ,
1500 , 1843 , 1881 1521 ,	215 , 410 , 414 ,
<code>\si@unt@perfalse</code>	1843 , 1846 , 1869	427 – 429 , 437 ,
.. 1533 , 1719 , 1750	<code>\si@unt@stackb</code> ..	441 , 455 , 672 ,
<code>\si@unt@perseenfalse</code> 1521 ,	674 , 795 , 1143 ,
..... 1534 , 1720	1847 , 1856 , 1860 ,	1216 , 1236 , 1469 ,
<code>\si@unt@perseenttrue</code>	1864 , 1869 , 1879	1647 , 1656 , 1935 ,
..... 1673 , 2813	<code>\si@unt@stackout</code>	1936 , 1939 , 1947 ,
<code>\si@unt@pertrue</code> 1550 , 1814	1987 , 1992 , 2443 ,
.. 1752 , 2813 , 2814	<code>\si@unt@stackpower</code>	2452 , 2461 , 2470 ,
<code>\si@unt@power</code> 1713 , 1716 , 1762	2484 , 2487 , 2490 ,
.. 1599 , 1701 , 1929	<code>\si@unt@stackvalsep</code>	2493 , 2505 , 2529 ,
 1507	2531 , 2534 , 2537 ,

2540, 2543, 2546,	<code>\squaremetrepercubicmetre</code>	U
2549, 2552, 2555, <u>2908</u>	<code>\uBar</code> <u>2701</u>
2558, 2574, 2584,	<code>\squaremetrepercubicmetreamp</code> .. <u>2495, 2569, 2871</u>	
2598, 2609, 2620, <u>2908</u>	<code>\unita</code> <u>2871</u>
2631, 2733, 2740,	<code>\squaremetrepercubicsecond</code>	<code>\unitfrac</code> .. <u>2501, 2579</u>
2741, 2748, 2753, <u>3011</u>	<code>\unitsep</code> <u>2581</u>
2754, 2758–2765,	<code>\squaremetrepercubicsecondnp</code>	<code>\unitSIdef</code> <u>2572</u>
2768, 2777, 2783, <u>3011</u>	<code>\unitsignonly</code> .. <u>2565</u>
2784, 2798, 2803,	<code>\squaremetreperkilogram</code>	<code>\unitsuperscript</code> <u>2581</u>
2805, 2808, 2809, <u>2996</u>	<code>\unitsym</code> . <u>11, 1421, 2565</u>
2811, 2813, 2814,	<code>\squaremetrepernewtonsecond</code>	<code>\unittimes</code> <u>2581</u>
2823, 2826, 2829, <u>2920</u>	<code>\unitvaluesep</code> ...
2832, 2835, 2838,	<code>\squaremetrepernewtonsecondnp</code>	.. <u>2560, 2568, 2578</u>
2841, 2877, 2880 <u>2920</u>	<code>\usk</code> <u>2803, 2907, 2909, 3089</u>
<code>\SIstyle</code> <u>2783</u>	<code>\squaremetrepersecond</code> <u>2944</u>
<code>\SIstyleToLang</code> . <u>2783</u> <u>2944</u>	V
<code>\SIthousandsep</code> . <u>2763</u>	<code>\squaremetrepersecondnp</code>	<code>\volt</code> . <u>13, 2167, 2221,</u>
<code>\SIunitdot</code> <u>2760</u> <u>2944</u>	<u>2222, 2291, 2292,</u>
<code>\SIunitsep</code> <u>2760</u>	<code>\squaremetrepersquaresecond</code>	<u>2677, 2678, 2930,</u>
<code>\SIunitspace</code> ... <u>2760</u> <u>2956</u>	<u>2931, 3076–3078</u>
<code>\Square</code> <u>10, 1918, 2251–</u>	<code>\squaremetrepersquaresecondnp</code>	<code>\voltbase</code> <u>3091</u>
<u>2253, 2397–2399,</u> <u>2944</u>	<code>\voltpermetre</code> .. <u>2920</u>
<u>2410, 2422, 2424,</u>	<code>\squaremetrepersteradian</code> <u>13, 2188,</u>	<code>\voltpermetrenp</code> <u>2920</u>
<u>2426, 2428, 2430,</u>	<u>2998, 3000, 3084</u>	
<u>2669–2671, 2808</u>	<code>\steradianbase</code> . <u>3091</u>	W
<code>\square</code> <u>2844</u>		<code>\watt</code> <u>13, 2167, 2223–</u>
<code>\squarecentimeter</code>		<u>2225, 2937–2939,</u>
..... <u>2669</u>	T	<u>2994–2998, 3000,</u>
<code>\squarecentimetre</code>	<code>\tebi</code> <u>17, 2431, 3061</u>	<u>3012, 3014, 3075</u>
..... <u>15, 2251</u>	<code>\tebid</code> <u>3054</u>	<code>\wattbase</code> <u>3091</u>
<code>\squared</code> <u>10, 1918, 2813,</u>	<code>\tera</code> <u>12,</u>	<code>\wattpercubicmetre</code>
<u>2898, 2904, 2955,</u>	<u>2141, 2261, 2266,</u> <u>2996</u>
<u>2987, 3104, 3108</u>	<u>2285, 2310, 2406,</u>	<code>\wattpercubicmetrenp</code>
<code>\squarekilometer</code> <u>2669</u>	<u>2420, 2430, 3049</u> <u>2996</u>
<code>\squarekilometre</code>	<code>\terad</code> <u>3043</u>	<code>\wattperkilogram</code> <u>2983</u>
..... <u>15, 2251</u>	<code>\teraelectronvolt</code>	<code>\wattperkilogramnp</code>
<code>\squaremeter</code> ... <u>2669</u> <u>16, 2254</u> <u>2983</u>
<code>\squaremetre</code> <u>15, 2251, 2907,</u>	<code>\terahertz</code> ... <u>15, 2262</u>	<code>\wattpermetrekelvin</code>
<u>2909, 2912, 2913,</u>	<code>\tesla</code> <u>13, 2167, 2965, 2966</u> <u>3011</u>
<u>2918, 2923, 2925,</u>	<code>\teslabase</code> <u>3107</u>	<code>\wattpermetrekelvinnp</code>
<u>2932, 2938, 2940,</u>	<code>\TeV</code> <u>16, 2304</u> <u>3011</u>
<u>2952, 2953, 2955,</u>	<code>\TeVoverc</code> <u>2411</u>	<code>\wattpersquaremetre</code>
<u>2957, 2961, 2972,</u>	<code>\TeVovercsq</code> <u>2421</u> <u>2933</u>
<u>2975, 2984, 2990,</u>	<code>\THz</code> <u>15, 2280</u>	<code>\wattpersquaremetrenp</code>
<u>2992, 2998, 3004,</u>	<code>\TinveV</code> <u>2401</u> <u>2933</u>
<u>3006, 3017, 3019,</u>	<code>\ton</code> <u>2639, 2886</u>	<code>\wattpersquaremetresteradian</code>
<u>3068, 3080, 3093,</u>	<code>\tonne</code> <u>13, 2338</u> <u>2996</u>
<u>3097, 3098, 3101,</u>	<code>\torr</code> <u>2365</u>	<code>\wattpersquaremetresteradiannp</code>
<u>3106, 3110, 3113,</u>	<code>\tothe</code> <u>11, 1922, 2585, 2807</u> <u>2996</u>
<u>3115, 3116, 3118</u>	<code>\tothe@opt@si</code> .. <u>1922</u>	<code>\weber</code> <u>13, 2167, 3081, 3082</u>
		<code>\weberbase</code> <u>3107</u>

Y		Z	
\yocto	12 , 2123 , 2383 , 2390 , 2396 , 3034	\yotta ...	12 , 2141 , 3053 2389 , 2395 , 3035
\yoctobarn 2378	\yottad 3043
\yoctod 3034	\zepto ..	2123 , 2382 , 2389 , 2395 , 3035
		\zeptobarn 2378
		\zeptod 3034
		\zetta ...	12 , 2141 , 3052
		\zettad 3043

23 References

- [1] <http://physics.nist.gov/cuu/Units/rules.html>.