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: \SIunits, \sistyle, \unitsdef and units are the leading examples. The \numprint\ package provides a large number of number-related functions, while \dcolumn\ and \rcol\ 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 \SIunits, \sistyle, \unitsdef\ and units. The aim is to have one package to handle all of the possible unit-related needs of \LaTeX\ users.

Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>I Introduction</td>
<td>2</td>
</tr>
<tr>
<td>1 Existing packages</td>
<td>3</td>
</tr>
<tr>
<td>1.1 Literal units</td>
<td>10</td>
</tr>
<tr>
<td>2 The wish-list</td>
<td>4</td>
</tr>
<tr>
<td>2.1 The unit interpreter</td>
<td>10</td>
</tr>
<tr>
<td>3 Road map</td>
<td>5</td>
</tr>
<tr>
<td>3.1 Units with no values</td>
<td>11</td>
</tr>
<tr>
<td>4 Road map</td>
<td>5</td>
</tr>
<tr>
<td>4.1 Free-standing units</td>
<td>11</td>
</tr>
<tr>
<td>4.2 Prefixes and powers</td>
<td>11</td>
</tr>
<tr>
<td>II Using the si package</td>
<td>5</td>
</tr>
<tr>
<td>5 Requirements</td>
<td>5</td>
</tr>
<tr>
<td>5.1 Prefixes and abbreviated units</td>
<td>12</td>
</tr>
<tr>
<td>6 Loading the package</td>
<td>6</td>
</tr>
<tr>
<td>6.1 Specialist units</td>
<td>16</td>
</tr>
<tr>
<td>6.2 Defining new units</td>
<td>17</td>
</tr>
<tr>
<td>7 Tabular material</td>
<td>7</td>
</tr>
<tr>
<td>8 Angles</td>
<td>8</td>
</tr>
<tr>
<td>9 Units and values</td>
<td>9</td>
</tr>
<tr>
<td>9.1 Literal units</td>
<td>10</td>
</tr>
<tr>
<td>9.2 The unit interpreter</td>
<td>10</td>
</tr>
<tr>
<td>9.3 Powers of units</td>
<td>10</td>
</tr>
<tr>
<td>9.4 Units with no values</td>
<td>11</td>
</tr>
<tr>
<td>9.5 Free-standing units</td>
<td>11</td>
</tr>
<tr>
<td>9.6 Prefixes and powers</td>
<td>11</td>
</tr>
<tr>
<td>9.7 Prefixes and abbreviated units</td>
<td>12</td>
</tr>
<tr>
<td>9.8 Specialist units</td>
<td>16</td>
</tr>
<tr>
<td>9.9 Defining new units</td>
<td>17</td>
</tr>
</tbody>
</table>

*This file describes version v.06a, last revised 2008/02/20.
†E-mail: joseph.wright@morningstar2.co.uk
Part I
Introduction

The correct application of units of measurement is very important in technical applications. For this reason, carefully-crafted definitions of a coherent units
system have been laid down by the Conférence Générale des Poids et Mesures1 (CGPM); this has resulted in the Système International d’Unités2 (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.

\LaTeX{} 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 \texttt{si} package is to learn from the existing implementations to provide a coherent and extendable approach to the problem. The original aim of developing \texttt{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 \texttt{si} does not follow any one of the existing packages; the interface is intended to be self-consistent and logical. As a result, \texttt{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 \texttt{\num}, \texttt{\ang} and \texttt{\SI} act in a very similar fashion to those in existing packages. In emulation mode, \texttt{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 \texttt{nist}\footnote{General Conference on Weights and Measures.} [1].\footnote{International System of Units.} 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:\footnote{sistyle also allows the use of German and South African rules “out of the box”.}

- **sistyle**
  - Easy input for numbers, for example typing \texttt{\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 (\texttt{via \sisingle}) or even be dependent on the document language (implemented by \texttt{\sistyleToLang}).

- **Slunits**
  - The look of units can be easily be changed in the whole document by redefining the commands.

---

1 General Conference on Weights and Measures.
2 International System of Units.
3 sistyle also allows the use of German and South African rules “out of the box”.
4 Thanks 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 “10m” 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 SIunits.

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 SIunits-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/s^2”, 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”. Completed

- Ability to understand and alter negative powers/fractions, and type-set these flexibly; thus \unitsym{\metre\per\second} could give “m s\(^{-1}\)”, “m/s”, “m m/s” or “m/s” depending on a package setting (suggested by Stefan Pinnow). Completed

- More logical handling of powers; for example \decicubic\metre or \decicubed\metre give “dm\(^3\)”, but \decicubed\metre does not even though \decicubic\metre cannot be cubed (adapted suggestion from Stefan Pinnow). Completed

\(^5\)\(1.550 \pm 0.002\) \times 10^{-10} \text{m.}

• Use of trailing units (as in \texttt{unitsdef} package, so for example 10\metre to give “10 m” (suggestion from Lan Thuy Pham). \textbf{Completed}

• Support use of non-Latin characters where appropriate, for example \texttt{\textmu} in units as in \texttt{\textit{\textmu m}} to give “\textmu m” (suggested by Martin Heller). \textbf{Completed}

• Integrate the core functionality of the \texttt{SIunits} and \texttt{sistyle} packages (suggested by Danie Els,\footnote{Current maintainer of \texttt{sistyle}.} as well as a key point of the review). \textbf{Completed}

• Modular design, with loadable definitions for different areas and typographic conventions. \textbf{Completed}

• Emulation of existing packages \texttt{units} (\textbf{Completed}), \texttt{unitsdef} (\textbf{Completed}), \texttt{sistyle} (\textbf{Completed}), \texttt{Slunits} (\textbf{Completed}) to allow easy upgrading. \textbf{Completed}

• Typesetting angles in “astronomy” style, for example \ang\texttt{astroang}\{30;5;3.2\} to give 30°5′3.2″ (suggested by Alok Singhal). \textbf{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 \texttt{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.

\texttt{v0.6} Seek feedback on implementation to date (current release);

\texttt{v0.7} Add or modify functionality based on feedback, implement new suggestions for wish list;

\texttt{v0.8} Fix bugs from 0.7 release, interface freeze;

\texttt{v0.9} Release-candidate: fix remaining bugs from v0.8, complete remaining documentation;

\texttt{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).
Part II
Using the si package

4 Requirements

si requires a reasonably up to date \TeX{} system. The package requires \texttt{-}\TeX{}-extensions, which should be available on most systems.\footnote{If you have an old \TeX{} try “\texttt{elatex}” rather than “\texttt{latex}.”} The following packages are also needed:

- \texttt{array} and \texttt{xspace} from the \texttt{tools} bundle, which should be available to everyone;
- \texttt{xkeyval} This processes the option handling, and needs to be at least v2.5;
- \texttt{amstext} From the \texttt{AMS}\TeX{} support bundle — the \texttt{AMS} fonts are also needed to provide the default upright $\mu$.

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

To use the \texttt{fraction=sfrac} option, the \texttt{xfrac} package is needed. This needs various experimental \texttt{\LaTeX}3 packages. As a result, \texttt{si} does not load \texttt{xfrac}. If you want to use \texttt{fraction=sfrac}, you need to load \texttt{xfrac} in your preamble. If the package is not loaded, \texttt{fraction=sfrac} falls back on a \texttt{nicefrac}-like method. The interested user should look at the \texttt{xfrac} documentation for reasons this might not be ideal.\footnote{On the other hand, some fractional units will look really bad with \texttt{sfrc}. Use this option with caution.}

5 Loading the package

\texttt{si} is loaded by the usual \texttt{\LaTeX} method.

\begin{verbatim}
\usepackage{si}\%
\usepackage[key=option]{si}
\end{verbatim}

As is shown in the example, the package can be loaded with one or more options, using the \texttt{xkeyval} 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 \texttt{xkeyval} settings as an optional argument.

6 Numbers

\begin{verbatim}
\num
\end{verbatim}

Numbers are automatically formatted by the \texttt{num} macro. This takes one optional and one mandatory argument: \texttt{\num\{\langle options\rangle\}\{\langle number\rangle\}}. The contents of \texttt{\langle number\rangle} are automatically formatted, in a similar method to that used by \texttt{numprint}. The formatter removes “hard” spaces ($\backslash$, and ~), automatically identifies exponents (by default marked using \texttt{e} or \texttt{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.
Various error-checking systems are built into the package, so that if \( \text{(number)} \) does not contain any numeric characters, a warning is issued. Isolated signs are also detected. The package recognises ( and ) as “extra” characters, which can be used to indicate the error in a number.\(^9\)

\[
1.234(5) = 1.234 \pm 0.005
\]

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.\(^10\)

\[
1234 567 890 1234 567 890
\]

\[
\text{\texttt{\textbackslash num{1234567890}}} \text{\texttt{\textbackslash num[mode=text]{1234567890}}}
\]

\texttt{si} can automatically add zeros and signs to numbers. This can be altered as desired.

\[
1.0
\]

\[
\text{\texttt{\textbackslash num{1.}}} \text{\texttt{\textbackslash num[padnumber=all]{1.}}}\\texttt{\textbackslash \
}
\]

\[
2 + 2
\]

\[
\text{\texttt{\textbackslash num{2}}} \text{\texttt{\textbackslash num[addsign=all]{2}}}\\texttt{\textbackslash \
}
\]

\[
3 \times 10^4 + 3 \times 10^4 + 3 \times 10^{+4}
\]

\[
\text{\texttt{\textbackslash num{3e4}}} \text{\texttt{\textbackslash num[addsign=mant]{3e4}}} \text{\texttt{\textbackslash num[addsign=all]{3e4}}}\\texttt{\textbackslash \
}
\]

\[
0.5 .5
\]

\[
\text{\texttt{\textbackslash num{.5}}} \text{\texttt{\textbackslash num[padnumber=none]{.5}}}\\texttt{\textbackslash 
}
\]

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

\[
1234 1234
\]

\[
\text{\texttt{\textbackslash num{1234}}} \text{\texttt{\textbackslash num[sepfour=true]{1234}}}\\texttt{\textbackslash 
}
\]

\[
1234 12,345
\]

\[
\text{\texttt{\textbackslash num{1234}}} \text{\texttt{\textbackslash num[digitsep=comma]{12345}}}\\texttt{\textbackslash 
}
\]

\[
12345
\]

\[
\text{\texttt{\textbackslash num[sepfour=none]{1234}}}\\texttt{\textbackslash 
}
\]

The formatting of exponents is also customisable.

\[
1 \times 10^{10} 1 \times 10^{10}
\]

\[
\text{\texttt{\textbackslash num{1e10}}} \text{\texttt{\textbackslash num[expproduct=\texttt{\textbackslash product}]{1e10}}}\\texttt{\textbackslash 
}
\]

\[
2 \times 10^{20} 1 \times 5^{10}
\]

\[
\text{\texttt{\textbackslash num{2e20}}} \text{\texttt{\textbackslash num[exppower=5]{1e10}}}\\texttt{\textbackslash 
}
\]

### 7 Tabular material

Centring numbers in tabular content is handled by a new column type, the \( s \) column. This is based closely on the \texttt{dcolumn} method for centring numbers in columns, but adds the functionality of the \texttt{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 \texttt{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, \texttt{tabformat=2.4} provides space for two digits before the decimal point.

\(9\)This is common in chemical crystallography.

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

\(11\)The approach used is actually a combination of \texttt{dcolumn} for centring the material and \texttt{numprint} for processing it. It will therefore give rather different results than the \( n \) and \( N \) column types in \texttt{numprint}.
The 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 \textit{(options)} can be any of the package options to apply only to this value. \textit{(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.

\begin{verbatim}
10° 12.3° 4.5°
1°2′3″ 0°0′1″
+10° −0′1″
\end{verbatim}

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

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

\begin{verbatim}
0°0′1″ 1″
2° 2°0′0″
0°3′0″ 4°0′0″ 0′0″5″
\end{verbatim}

\ang{;;1} \ang{;;1}\%
\ang{2;;} \ang{2;;}\%
\ang{0;3;} \ang{4;;} \ang{;;5}

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

\begin{verbatim}
1.05° 1,05°
3.6789° 3.6789°
9°8′7″ 9°8′7″
\end{verbatim}

\ang{1.05} \ang{[decimalsign=comma]{1.05}}\%
\ang{3.6789} \ang{[digitsep=comma]{3.6789}}\%
\ang{9;8;7} \ang{[anglesep=thin]{9;8;7}}\%

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

\begin{verbatim}
1°2′3.4″
1°2′3.4
\end{verbatim}

\ang{1;2;3.4}\%
\ang{[astroang]{1;2;3.4}}\%

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 siunitx 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)}{(unit)}. The \textit{(number)} argument operates in exactly the same manner as the equivalent argument of the \num macro. \textit{(unit)} will be typeset with a non-breakable space between it and the preceding number, with font control as outlined earlier. Finally, \textit{(preunit)} is a unit to be typeset before the numerical value (most likely to be a currency). Some examples illustrate the general power of the macro.
\[1.23 \text{ J mol}^{-1} \text{ K}^{-1}\]
\[0.23 \times 10^7 \text{ cd}\]
\[1.99/\text{kg}\]
\[70 \text{ m s}^{-1}\]
\[1.345 \text{ A/mol}\]

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 \texttt{ sistyle} and \texttt{SIunits}. The \texttt{ 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 \texttt{unitsep} option;
- ~ The contents of the \texttt{unitspace} option are typeset by a tilde.

This allows ready input of units.

\[10 \text{ kg m s}^{-2}\]
\[1.453 \text{ g/cm}^3\]
\[33.562 \text{ cd s}\]
\[100 \text{ m} \cdot \text{s}^{-2}\]

### 9.2 The unit interpreter

The second operation mode for the \SI macro is based on the behaviour of \texttt{SIunits}. 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.

\[10 \text{ kg m s}^{-2}\]
\[1.453 \text{ g/cm}^3\]
\[33.562 \text{ cd s}\]
\[100 \text{ m} \cdot \text{s}^{-2}\]

On its own, this is very similar to \texttt{SIunits}, and is less convenient than the direct input method. 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.

\[10 \text{ g m}^{-1} \text{ s}^2\]
\[1.453 \text{ g/cm}^3\]
\[33.562 \text{ cd s}\]
\[100 \text{ m}^2 \cdot \text{s}^{-2}\]

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.

\textsuperscript{14}Users of \texttt{SIunits} should note the lack of need for a \texttt{usk}-type macro.
9.3 Powers of units

Including powers in units is handled using a “natural language” method. Thus preceding a unit by \texttt{\Square} or \texttt{\cubic} with raise the unit to the appropriate power, while \texttt{\squared} or \texttt{\cubed} follow the unit they apply to.\footnote{The \texttt{\Square} macro is capitalised to avoid a name clash with pstricks.}

\begin{verbatim}
\texttt{\SI{10}{\ metre^2}}\verb+\verb+ \SI{20}{\ Square\ metre}\verb+\verb+
\texttt{\SI{30}{\ metre^3}}\verb+\verb+ \SI{40}{\ cubic\ metre}
\end{verbatim}

The \texttt{\per} macro intelligently creates reciprocal powers, and also adds the power $-1$ when appropriate.

\begin{verbatim}
\texttt{\SI{10}{\ per\ second^2}}\verb+\verb+ \SI{20}{\ per\ second^2}\verb+\verb+
\texttt{\SI{30}{\ per\ second\cubed}}\verb+\verb+ \SI{40}{\ per\ cubic\ second}\verb+\verb+
\texttt{\SI{50}{\ per\ second}}\verb+\verb+
\end{verbatim}

For powers not defined above or with \texttt{\newpower}, the \texttt{\tothe} macro can be used “in line” to produce a power. As follows from standard English usage, this comes after the unit.\footnote{Suggestions for a macro name for before the unit for the same job are welcome!}

\begin{verbatim}
\texttt{\SI{}{16.86\ metre\tothe{4}}}\verb+\verb+ \SI{}{7.895\ candela\tothe{0.5}}\verb+\verb+
\texttt{\SI{}{7.895\ newton\tothe{-6}}}\verb+\verb+ \SI{}{1.34\ per\ kelvin\tothe{7}}
\end{verbatim}

9.4 Units with no values

For typesetting the symbol for a unit on its own, with the full font control and without extra spaces, the \texttt{\unitsym} macro is provided.\footnote{The same effect can be achieved using the \texttt{\SI} macro with an empty numerical argument.}

\begin{verbatim}
\texttt{\unitsym{kg.m/s^2}}\verb+\verb+ \texttt{\unitsym{\mole\per\cubic\deci\metre}}
\end{verbatim}

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 loose font control of the value.
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={}`, \(\text{kilo}\) and \(\text{gram}\) 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`, \textsf{si} also defines the standard prefixes for powers of ten (Table 3). This leads to the redefinition of \(\text{kilogram}\) as \(\text{kilo}/\text{gram}\). The macro \texttt{\textbackslash deka} is provided, as this is used as an alias for \texttt{\textbackslash deca} in some places. The package also defines a number of derived SI units which have assigned names and symbols (Table 4). Note that \texttt{\textbackslash Gray} is capitalised to avoid a name clash with the \texttt{pstricks} package.\footnote{The macros \texttt{\textbackslash ohm} and \texttt{\textbackslash celsius} are not defined by \textsf{si} if the \texttt{gensymb} package is loaded.} In addition to these units, there

---

### Table 2: The seven base SI units

<table>
<thead>
<tr>
<th>Unit</th>
<th>Macro</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>kilogram</td>
<td>\kilo</td>
<td>kg</td>
</tr>
<tr>
<td>metre</td>
<td>\metre</td>
<td>m</td>
</tr>
<tr>
<td>second</td>
<td>\second</td>
<td>s</td>
</tr>
<tr>
<td>mole</td>
<td>\mole</td>
<td>mol</td>
</tr>
<tr>
<td>kelvin</td>
<td>\kelvin</td>
<td>K</td>
</tr>
<tr>
<td>ampere</td>
<td>\ampere</td>
<td>A</td>
</tr>
<tr>
<td>candela</td>
<td>\candela</td>
<td>cd</td>
</tr>
</tbody>
</table>

### Table 3: The SI prefixes (`load=prefix`)  

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Macro</th>
<th>Power</th>
<th>Symbol</th>
<th>Prefix</th>
<th>Macro</th>
<th>Power</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>yocto</td>
<td>\yocto</td>
<td>(10^{-24})</td>
<td>y</td>
<td>atto</td>
<td>\atto</td>
<td>(10^{-18})</td>
<td>a</td>
</tr>
<tr>
<td>femto</td>
<td>\femto</td>
<td>(10^{-15})</td>
<td>f</td>
<td>pico</td>
<td>\pico</td>
<td>(10^{-12})</td>
<td>p</td>
</tr>
<tr>
<td>nano</td>
<td>\nano</td>
<td>(10^{-9})</td>
<td>n</td>
<td>micro</td>
<td>\micro</td>
<td>(10^{-6})</td>
<td>µ</td>
</tr>
<tr>
<td>milli</td>
<td>\milli</td>
<td>(10^{-3})</td>
<td>m</td>
<td>centi</td>
<td>\centi</td>
<td>(10^{-2})</td>
<td>c</td>
</tr>
<tr>
<td>deci</td>
<td>\deci</td>
<td>(10^{-1})</td>
<td>d</td>
<td>deca</td>
<td>\deca</td>
<td>(10^{1})</td>
<td>da</td>
</tr>
<tr>
<td>hecto</td>
<td>\hecto</td>
<td>(10^{2})</td>
<td>h</td>
<td>kilo</td>
<td>\kilo</td>
<td>(10^{3})</td>
<td>k</td>
</tr>
<tr>
<td>mega</td>
<td>\mega</td>
<td>(10^{6})</td>
<td>M</td>
<td>giga</td>
<td>\giga</td>
<td>(10^{9})</td>
<td>G</td>
</tr>
<tr>
<td>tera</td>
<td>\tera</td>
<td>(10^{12})</td>
<td>T</td>
<td>peta</td>
<td>\peta</td>
<td>(10^{15})</td>
<td>P</td>
</tr>
<tr>
<td>exa</td>
<td>\exa</td>
<td>(10^{18})</td>
<td>E</td>
<td>zetta</td>
<td>\zetta</td>
<td>(10^{21})</td>
<td>Z</td>
</tr>
<tr>
<td>yotta</td>
<td>\yotta</td>
<td>(10^{24})</td>
<td>Y</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 4: The derived SI units with defined names (load=derived)

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

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

<table>
<thead>
<tr>
<th>Unit</th>
<th>Macro</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>electron volt</td>
<td>\electronvolt</td>
<td>eV</td>
</tr>
<tr>
<td>atomic mass unit</td>
<td>\atomicmassunit</td>
<td>u</td>
</tr>
<tr>
<td></td>
<td>\atomicmass</td>
<td>u</td>
</tr>
<tr>
<td>dalton</td>
<td>\dalton</td>
<td>Da</td>
</tr>
</tbody>
</table>

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).

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

<table>
<thead>
<tr>
<th>Unit</th>
<th>Macro</th>
<th>Symbol</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Masses</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>kilogram</td>
<td>\kilo</td>
<td>kg</td>
<td>\kg</td>
</tr>
<tr>
<td>femtogram</td>
<td>\femto</td>
<td>fg</td>
<td>\fg</td>
</tr>
<tr>
<td>picogram</td>
<td>\pico</td>
<td>pg</td>
<td>\pg</td>
</tr>
</tbody>
</table>

Continued on next page

These are supposed to be replaced over time by SI units.
<table>
<thead>
<tr>
<th>Unit</th>
<th>Macro</th>
<th>Symbol</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>nanogram</td>
<td><code>\nanogram</code></td>
<td><code>ng</code></td>
<td><code>\nanog</code></td>
</tr>
<tr>
<td>microgram</td>
<td><code>\microgram</code></td>
<td><code>µg</code></td>
<td><code>\micg</code></td>
</tr>
<tr>
<td>milligram</td>
<td><code>\milligram</code></td>
<td><code>mg</code></td>
<td><code>\mg</code></td>
</tr>
<tr>
<td>atomic mass</td>
<td><code>\atomicmass</code></td>
<td><code>u</code></td>
<td><code>\amu</code></td>
</tr>
</tbody>
</table>

**Lengths**
- picometre   | `\picometre`  | `pm`   | `\picom`     |
- nanometre    | `\nanometre`  | `nm`   | `\nm`        |
- micrometre   | `\micrometre` | `µm`   | `\micm`      |
- millimetre   | `\millimetre` | `mm`   | `\mm`        |
- centimetre   | `\centimetre` | `cm`   | `\cm`        |
- decimetre    | `\decimetre`  | `dm`   | `\dm`        |
- kilometre    | `\kilometre`  | `km`   | `\km`        |

**Times**
- second       | `\second`     | `s`    | `\Sec`       |
- attosecond   | `\attosecond` | `as`   | `\as`        |
- femtosecond  | `\femtosecond`| `fs`   | `\fs`        |
- picosecond   | `\picosecond` | `ps`   | `\ps`        |
- nanosecond   | `\nanosecond` | `ns`   | `\ns`        |
- microsecond  | `\microsecond`| `µs`   | `\mics`      |
- millisecond  | `\millisecond` | `ms`  | `\ms`        |

**Moles**
- femtomole    | `\femtomole`  | `fmol` | `\fmol`      |
- picomole     | `\picomole`   | `pmol` | `\pmol`      |
- nanomole     | `\nanomole`   | `nmol` | `\nmol`      |
- micromole    | `\micromole`  | `µmol` | `\micmol`    |
- millimole    | `\millimole`  | `mmol` | `\mmol`      |

**Currents**
- picoampere   | `\picoampere` | `pA`   | `\pA`        |
- nanoampere   | `\nanoampere` | `nA`   | `\nA`        |
- microampere  | `\microampere`| `µA`   | `\micA`      |
- milliampere  | `\milliampere`| `mA`   | `\mA`        |
- kiloampere   | `\kiloampere` | `kA`   | `\kA`        |

**Areas**
- squaremetre  | `\squaremetre`| `m^2`  |               |
- squarecentimetre | `\squarecentimetre` | `cm^2`  |
- squarekilometre | `\squarekilometre` | `km^2`  |

**Volumes**
- millilitre   | `\millilitre` | `ml`   | `\ml`        |
- microlitre   | `\microlitre` | `µl`   | `\micl`      |

*Continued on next page*
<table>
<thead>
<tr>
<th>Unit</th>
<th>Macro</th>
<th>Symbol</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>centimetrecubed</td>
<td>\centimetrecubed</td>
<td>cm³</td>
<td>cmc</td>
</tr>
<tr>
<td>cubicdecimetre</td>
<td>\cubicdecimetre</td>
<td>dm³</td>
<td>dmc</td>
</tr>
</tbody>
</table>

### Frequencies

<table>
<thead>
<tr>
<th></th>
<th>Macro</th>
<th>Symbol</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>hertz</td>
<td>\hertz</td>
<td>Hz</td>
<td>Hz</td>
</tr>
<tr>
<td>millihertz</td>
<td>\millihertz</td>
<td>mHz</td>
<td>mHz</td>
</tr>
<tr>
<td>kilohertz</td>
<td>\kilohertz</td>
<td>kHz</td>
<td>kHz</td>
</tr>
<tr>
<td>megahertz</td>
<td>\megahertz</td>
<td>MHz</td>
<td>MHz</td>
</tr>
<tr>
<td>gigahertz</td>
<td>\gigahertz</td>
<td>GHz</td>
<td>GHz</td>
</tr>
<tr>
<td>terahertz</td>
<td>\terahertz</td>
<td>THz</td>
<td>THz</td>
</tr>
</tbody>
</table>

### Potentials

<table>
<thead>
<tr>
<th></th>
<th>Macro</th>
<th>Symbol</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>millivolt</td>
<td>\millivolt</td>
<td>mV</td>
<td>mV</td>
</tr>
<tr>
<td>kilovolt</td>
<td>\kilovolt</td>
<td>nV</td>
<td>kV</td>
</tr>
</tbody>
</table>

### Energies

<table>
<thead>
<tr>
<th></th>
<th>Macro</th>
<th>Symbol</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>kilojoule</td>
<td>\kilojoule</td>
<td>kJ</td>
<td>kJ</td>
</tr>
<tr>
<td>electronvolt</td>
<td>\electronvolt</td>
<td>eV</td>
<td>eV</td>
</tr>
<tr>
<td>millielectronvolt</td>
<td>\millielectronvolt</td>
<td>meV</td>
<td>meV</td>
</tr>
<tr>
<td>kiloelectronvolt</td>
<td>\kiloelectronvolt</td>
<td>keV</td>
<td>keV</td>
</tr>
<tr>
<td>megaelectronvolt</td>
<td>\megaelectronvolt</td>
<td>MeV</td>
<td>MeV</td>
</tr>
<tr>
<td>gigaelectronvolt</td>
<td>\gigaelectronvolt</td>
<td>GeV</td>
<td>GeV</td>
</tr>
<tr>
<td>teraelectronvolt</td>
<td>\teraelectronvolt</td>
<td>TeV</td>
<td>TeV</td>
</tr>
</tbody>
</table>

### Powers

<table>
<thead>
<tr>
<th></th>
<th>Macro</th>
<th>Symbol</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>milliwatt</td>
<td>\milliwatt</td>
<td>mW</td>
<td></td>
</tr>
<tr>
<td>kilowatt</td>
<td>\kilowatt</td>
<td>kW</td>
<td></td>
</tr>
<tr>
<td>megawatt</td>
<td>\megawatt</td>
<td>MW</td>
<td></td>
</tr>
</tbody>
</table>

### Capacitance

<table>
<thead>
<tr>
<th></th>
<th>Macro</th>
<th>Symbol</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>femtofarad</td>
<td>\femtofarad</td>
<td>fF</td>
<td></td>
</tr>
<tr>
<td>picofarad</td>
<td>\picofarad</td>
<td>pF</td>
<td></td>
</tr>
<tr>
<td>nanofarad</td>
<td>\nanofarad</td>
<td>nF</td>
<td></td>
</tr>
<tr>
<td>microfarad</td>
<td>\microfarad</td>
<td>μF</td>
<td></td>
</tr>
<tr>
<td>millifarad</td>
<td>\millifarad</td>
<td>mF</td>
<td></td>
</tr>
</tbody>
</table>

### Resistance

<table>
<thead>
<tr>
<th></th>
<th>Macro</th>
<th>Symbol</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>kiloohm</td>
<td>\kiloohm</td>
<td>kΩ</td>
<td></td>
</tr>
<tr>
<td>megaohm</td>
<td>\megaohm</td>
<td>MΩ</td>
<td></td>
</tr>
<tr>
<td>gigaohm</td>
<td>\gigaohm</td>
<td>GΩ</td>
<td></td>
</tr>
<tr>
<td>millisiemens</td>
<td>\millisiemens</td>
<td>mS</td>
<td></td>
</tr>
</tbody>
</table>

### Forces

<table>
<thead>
<tr>
<th></th>
<th>Macro</th>
<th>Symbol</th>
<th>Abbreviation</th>
</tr>
</thead>
<tbody>
<tr>
<td>millinewton</td>
<td>\millinewton</td>
<td>mN</td>
<td></td>
</tr>
</tbody>
</table>
### Specialist units

In some subject area, there are units which are in common use even though they are outside of the SI system. Unlike the units discussed earlier, these specialist units are not loaded by default. si comes with the predefined files alsoload=chemistry and alsoload=hep. The latter 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 “\text{m}” or “\text{M}”. The later is obviously easily confused with the sign for the prefix mega.

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.

### Defining new units

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. The (first) optional argument to \SI and \unitsym can be used to override the settings for the unit.

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

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.

```
10 \text{m} \pi^{-2} \newunit[\{SI\pi\}]{\text{\textnormal{\textbf{\textit{\mathnormal{\textbf{\textit{\pi}}}}}}}}{\SI{10}{\text{\textnormal{\textbf{\textit{\mathnormal{\textbf{\textit{\metre\per\SI\pi\squared}}}}}}}}}
```

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 \textbf{must} be a number.

---

---

---

---

---

---

---

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

<table>
<thead>
<tr>
<th>Unit</th>
<th>Macro</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>minute</td>
<td>\minute</td>
<td>min</td>
</tr>
<tr>
<td>hour</td>
<td>\hour</td>
<td>h</td>
</tr>
<tr>
<td>day</td>
<td>\Day</td>
<td>d</td>
</tr>
<tr>
<td>degree</td>
<td>\degree</td>
<td>°</td>
</tr>
<tr>
<td>minute (arc)</td>
<td>\arcmin</td>
<td>′</td>
</tr>
<tr>
<td>second (arc)</td>
<td>\arcsec</td>
<td>″</td>
</tr>
<tr>
<td>litre</td>
<td>\litre</td>
<td>l</td>
</tr>
<tr>
<td>tonne</td>
<td>\tonne</td>
<td>t</td>
</tr>
<tr>
<td>neper</td>
<td>\neper</td>
<td>Np</td>
</tr>
<tr>
<td>bel</td>
<td>\bel</td>
<td>B</td>
</tr>
<tr>
<td>percent</td>
<td>\percent</td>
<td>%</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Unit</th>
<th>Macro</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>ånström</td>
<td>\angstrom</td>
<td>Å</td>
</tr>
<tr>
<td>are</td>
<td>\are</td>
<td>a</td>
</tr>
<tr>
<td>hectare</td>
<td>\hectare</td>
<td>ha</td>
</tr>
<tr>
<td>barn</td>
<td>\barn</td>
<td>b</td>
</tr>
<tr>
<td>bar</td>
<td>\BAR</td>
<td>bar</td>
</tr>
<tr>
<td>millibar</td>
<td>\millibar</td>
<td>mbar</td>
</tr>
<tr>
<td>gal</td>
<td>\gal</td>
<td>Gal</td>
</tr>
<tr>
<td>curie</td>
<td>\curie</td>
<td>Ci</td>
</tr>
<tr>
<td>roentgen</td>
<td>\roentgen</td>
<td>R</td>
</tr>
<tr>
<td>rad</td>
<td>\rad</td>
<td>rad</td>
</tr>
<tr>
<td>rem</td>
<td>\rem</td>
<td>rem</td>
</tr>
</tbody>
</table>

### Table 9: Binary prefixes (alsoload=binary)

<table>
<thead>
<tr>
<th>Prefix</th>
<th>Macro</th>
<th>Power</th>
</tr>
</thead>
<tbody>
<tr>
<td>kibi</td>
<td>\kibi</td>
<td>$2^{10}$</td>
</tr>
<tr>
<td>mebi</td>
<td>\mebi</td>
<td>$2^{20}$</td>
</tr>
<tr>
<td>gibi</td>
<td>\gibi</td>
<td>$2^{30}$</td>
</tr>
<tr>
<td>tebi</td>
<td>\tebi</td>
<td>$2^{40}$</td>
</tr>
<tr>
<td>pebi</td>
<td>\pebi</td>
<td>$2^{50}$</td>
</tr>
<tr>
<td>exbi</td>
<td>\exbi</td>
<td>$2^{60}$</td>
</tr>
</tbody>
</table>
Giving the optional argument \texttt{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.

\newpower{\text{quartic}}{4} \newpower[post]{\text{to the forth}}{4} \unitsym{\text{kilogram} \text{to the forth}} \unitsym{\text{quartic} \text{metre}}

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

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

\section{Font control}

Following the lead of \texttt{siunitx}, \texttt{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.

\begin{verbatim}
10 \num{10} $\num{10}$ \\ 
20° \ang{20} $\ang{20}$ \\ 
30 kg \SI{30}{\kilo\gram} \\
50 \ang{1;1;1} $\ang{1;1;1}$ \\
\end{verbatim}

By giving the \texttt{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 \texttt{obeybold} option causes the bold setting to be obeyed in the same way.

\begin{verbatim}
1°1'1" 1°1'1" 1°1'1" \sisetup{obeyfamily,obeybold,obeyitalic} \\
2°2'2" 2°2'2" \ang{1;1;1} $\ang{1;1;1}$ \\
3°3'3" 3°3'3" \ang{2;2;2} $\ang{2;2;2}$ \\
4°4'4" 4°4'4" \ang{3;3;3} $\ang{3;3;3}$ \\
\ang{5;5;5} $\ang{5;5;5}$ \\
\end{verbatim}

\subsection{Package options}

The “native” options for the package are all given using the \texttt{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 \texttt{\sisetup} macro.

The package options take a number of different forms.

- \texttt{option=bool} Simple true/false values. These macros all default to true, so giving the option name alone sets the flag to true.
• **option**=\{\texttt{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**=\{\texttt{choice,literal}\} If the given value is a \{\texttt{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**=\{\texttt{literal}\} The given value is used as a literal by the package.

• **option**=\{\texttt{macro}\} These options expect a macro name as a value; the macro name is then used by the package. Note that the name does \textit{not} include the leading \texttt{.}.

• **option**=\{\texttt{length}\} Requires a \texttt{TeX} lengths, for example 0.5ex.

• **option**=\{\texttt{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**=\{\texttt{bool}\} By default, the font family used for typesetting does \textit{not} match the surroundings. This is altered using the obeyfamily switch; when active, serif (Roman), sans serif and typewriter fonts are detected.

• **mode**=\{\texttt{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**=\{\texttt{bool}\} A shortcut for \texttt{[mode=text]}.

• **obeymode**=\{\texttt{bool}\} The package can detect and use the surrounding maths or text mode, if requested. Default is \texttt{false}.

• **obeybold**=\{\texttt{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**=\{\texttt{choice}\} For inline maths, the package can check either the surrounding maths or the surrounding text. The options here are \texttt{text} and \texttt{maths} (or \texttt{math}).

• **obeyitalic**=\{\texttt{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 $\texttt{\mathit{\mathsf{10}}}$).

19
• mathdefault=(macro) The default shape used for text printed in maths mode. The default is the value stored in mathrm.

• textdefault=(macro) The default shape used for text printed in text mode. The default is the value stored in mathrm.

• mathnumdefault=(macro) The default shape used for numbers printed in maths mode. The default is the value stored in mathrm.

• textnumdefault=(macro) The default shape used for numbers printed in text mode. The default is the value stored in mathrm.

• mathrm=(macro) 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=(macro) The font command used in maths mode when the surrounding text is sans serif.

• mathrm=(macro) The font command used in maths mode when the surrounding text is fixed width.

• textrm=(macro) 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=(macro) The font command used in text mode when the surrounding text is sans serif.

• texttt=(macro) 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=(choice,literal) This defines the separation of different unit symbols. The (list) takes values thin, medium (alias med), thick (all maths spacings), space (a full space), cdot (a centred dot) and times. (literal) values are typeset in maths mode.

• unitspace=(choice,literal) The spacing represented by an explicit hard space (~) inside a unit macro. Takes the same list as valuesep.

• valuesep=(choice,literal) Defines the separation between a value and the associated unit. Valid (list) values are thin, medium (alias med), thick, space and none.

• digitsep=(choice,literal) The separation (if any) between groups of digits in large numbers. Valid (list) values are thin, medium (alias med), thick, space, comma, fullstop (aliases stop and period) and none.

• decimalsign=(choice,literal) 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: \text{edED}
- 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): \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 (± and ±, 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 \text{2 \times 10^5}). 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 $0^\circ$ 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 \(\langle pre\rangle\) is the number of digits before the decimal marker and \(\langle post\rangle\) is the number after. Appropriate space is reserved to centre a number of total length \(\langle pre\rangle + \langle post\rangle\) (plus the decimal marker). If the digits supplied are too long, overfull boxes will result. If only \(\langle pre\rangle\) 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 “10 m away” with \xspace turned on, but “10 m away” 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 \LaTeX \frac operation), *nice* (also available as *nicefrac*; uses a nicefrac-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 \text{sfrac} (uses the \sfrac macro from the xfrac package.\footnote{\text{xfrac} is part of the experimental system for \LaTeX. As it requires a number of additional packages to work, \text{si} does not load \text{xfrac}. If it is unavailable, the \text{sfrac} setting will fall back to using \text{\nicefrac}. See the \text{xfrac} documentation for reasons to prefer \text{sfrac} to \text{\nicefrac}.}

- \text{denlbrac}=\langle \text{lateral} \rangle \text{ and } \text{denrbrac}=\langle \text{lateral} \rangle \text{ When using per=slash, using two or more units in the denominator gives an ambiguous fraction. The package therefore adds \text{denlbrac} and \text{denrbrac} in such cases.}
- \text{prefix}=(\text{choice}) \text{ Controls how prefixes to units are handled, with options symbol (or letter) and number (or power).}
- \text{prefixpower}=(\text{choice}, \text{lateral}) \text{ and } \text{prefixproduct}=(\text{choice}) \text{ 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 \( \mu \), \( \Omega \), \( \text{Å} \), \( ^\circ \) and °C is provided here. These are all literal options, which are available in text and maths mode variants. For example, \text{textmicro} is the code used for the \( \mu \) symbol in text mode. The text mode macros should be safe when forced into text, and the maths ones when forced into maths.

- \text{textOmega}
- \text{mathsOmega}
- \text{textmu}
- \text{mathsmu}
- \text{textdegree}
- \text{mathsdegree}
- \text{textminute}
- \text{mathsminute}
- \text{textsecond}
- \text{mathssecond}
- \text{textringA}
- \text{mathsringA}
- \text{textcelsius}
- \text{mathscelsius}

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

The \text{eV} symbol requires some fine-tuning, and so has two options of its own.
• \texttt{eVcorra=\{length\}} The correction applied to the gap between “e” and “V” of the unit. The default is 0.3ex.

• \texttt{eVcorrb=\{length\}} 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 \texttt{\textbackslash unitsym\{per=slash\}\{\textbackslash electronvolt\per\textbackslash metre\}}, which gives \(\text{eV/m}\) by default, but \(\text{eV/m}\) by setting \texttt{eVcorrb=0.7ex}. The value needed will depend on the use of the unit and the font metrics used.

\section*{11.8 Package control}

These macros alter the overall behaviour of the package.

• \texttt{load\{list\}} Sets which additional configuration files are loaded. These all have names of the form \texttt{si-\{option\}.cfg}, where \texttt{\{option\}} should be given in the \texttt{load} list. The package recognises the \texttt{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.

• \texttt{noload\{list\}} 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.

• \texttt{alsoload\{list\}} Adds an item to the list to be loaded, without needing to specify all of the existing list.

• \texttt{log\{choice\}} Sets the amount of information written to the log by \texttt{si}. The \texttt{\{list\}} is \texttt{none, errors, normal and debug}. The last option is also available as a Boolean, and gives \texttt{lots} of information in the log.

• \texttt{emulate\{choice\}} Causes \texttt{si} to emulate the given package. The \texttt{\{list\}} takes values \texttt{SIunits, sistyle, units and unitsdef}. This option can only be used when loading the package.

\section*{11.9 Back-compatibility options}

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

\section*{12 Emulation of other packages}

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

\textsuperscript{23}User 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 si. 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.

\numprint{-123456} \numprint[N/mm^2]{-123456}

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

13 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 AMS\TeX 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. The use of the \mathnormal font may get
around this issue.

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

There are several potential pitfalls in this area; experimentation may well be
needed.\footnote{This depends on your font setup; this document uses T1 encoding,
which shows the issue, whereas using OT1 does not.}

\footnote{Any 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)LATEX.

\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 \LaTeX tricks from far too many people to thank all of them. However, for this package specific thanks must go to the authors of the existing “unit” packages: Danie Els (sistyle), Marcel Heldoorn (Slunits), Patrick Happel (unitsdef), Axel Reichert (units) and Harald Harders (numprint). Philip Lehmann, Will Robertson and Heiko Oberdiek deserve much credit for demonstrating \LaTeX 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 \texttt{sistyle}, \texttt{Slunits}, \texttt{numprint}, \texttt{unitsdef} or \texttt{units}. Some ideas have also been borrowed from \texttt{biblatex}; again these will not be specifically noted. Code from other packages will be marked when used.

User-space commands (those not containing \texttt{@}) 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.\footnote{This applies to \LaTeX\ kernel commands only; for example, \texttt{ifthenelse} is not used.} For example, \texttt{\newcommand*} is used in place of \texttt{\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.

\begin{verbatim}
\NeedsTeXFormat{LaTeX2e}
\ProvidesPackage{si} \[2008/02/20 v.06a A comprehensive (SI) units package\]
\PackageInfo{si}{This package is experimental. The interface and functionally is subject to review and may be changed in later releases}\n
The package requires e-\TeX, so the usual test is made.
\begingroup
\@ifundefined{eTeXversion}{\PackageError{si}{Not running under e-\TeX.\nThis package requires e-\TeX. Try compiling the document with\MessageBreak\LaTeX\ 'elatex' instead of 'latex'. When using\MessageBreak pdf\LaTeX, try 'pdfelatex' instead of 'pdflatex'\MessageBreak and may be changed in later releases}{}\n\endgroup
\endinput
\end{verbatim}

Packages needed for functionality are loaded. \texttt{xkeyval} handles the package options, while \texttt{amsmath} from the \texttt{AMS} bundle is needed for \texttt{\text}. \texttt{array} is needed for the new column type for tabular material. \texttt{xspace} provides “magic” spacing after macros, if requested. \texttt{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, \texttt{si} aborts if \texttt{xkeyval} is too old.
\RequirePackage{xkeyval}
\@ifpackagelater{xkeyval}{2005/05/07}
{
\PackageError{si}{xkeyval >= 2.5 required}{si requires the `xkeyval' package, version 2.5 or later.\MessageBreak The version loaded is:\MessageBreak `\@nameuse{ver@xkeyval.sty}'.\MessageBreak This is a fatal error; the package will abort.}%
\endinput
\RequirePackage{amstext,array,xspace}

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

Various items will need a switch. To avoid name pollution, a single switch is defined here; grouping will keep the definition local.
\newif{\si@switch}
\si@packagecheck
\ifsi@switch
As \textit{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 \textit{SIunits} and \textit{sistyle} could be loaded with variable capitalisation (at least on Windows); both possibilities are tested. Also notice that \textit{unitsdef} must be tested before \textit{units}, so that users of the former get an intelligible message.
\newcommand*{\si@packagecheck}{%\begingroup\@for\si@tempa:=SIunits,siunits,sistyle,SIstyle,unitsdef\do{\@ifpackageloaded{\si@tempa}{\PackageError{si}{Package `\si@tempa' incompatible}{The `\si@tempa' package and `si' are incompatible.\MessageBreak Use the `emulate=\si@tempa' package option when loading si.}}{}}\endgroup}Some packages should not cause a clash, but are emulated and would be better handled that way.
\@for\si@tempa:=units\do{%\@ifpackageloaded{\si@tempa}{\PackageWarning{si}{Consider loading the si package with\MessageBreak option `emulate=\si@tempa', rather than\MessageBreak loading both \si@tempa and si}}{}}\endgroup

The check is carried out on loading and at the beginning of the document, so that packages loaded both before and after \textit{si} are caught.
\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{\textbackslash{macro}}
\newcommand*{\si@ifdefinable}[1]{\expandafter\expandafter\expandafter\@ifundefined%\expandafter\expandafter\expandafter{\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{\textbackslash{macro}}{\textbackslash{items}}
\newcommand*{\si@addtolist}[2]{\ifx\@empty#1\@empty\edef#1{#2}\else\edef#1{#1,#2}\fi}

\si@addtocsname  A second item to add to a is macro.
\si@addtocsname{\textbackslash{macro}}{\textbackslash{csname}}{\textbackslash{tokens}}
\newtoks{\si@temptoks}
\newcommand*{\si@addtocsname}[3]{\ifundefined{#1}{\@namedef{#1}{#2}}{\si@temptoks\expandafter\expandafter\expandafter{\csname #1\endcsname#2}}\expandafter\edef\csname #1\endcsname{\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.
\begingroup\catcode\textasciitilde=3\long\gdef\si@xifmtarg#1#2#3#4#5\@nil{#4}\long\gdef\si@ifnotmtarg#1{\si@xifmtarg#1QQ\@firstofone\@gobble\@nil}\endgroup

16.2  Logging
\si@debug  To control logging, some new switches are declared.
\newif{\si@debug}
\newif{\si@logmin}
\newif{\si@lognone}
\newif{\si@logerr}
\newif{\si@logwarn}
\newif{\si@loginf}
\si@log@err{\textbackslash{error}}{\textbackslash{explanation}}
\si@log@warn{\textbackslash{warning}}
\si@log@inf{\textbackslash{information}}
\newcommand*{\si@log@err}[2]{% 
\ifsi@lognone \else 
\PackageWarning{si}{#1}% 
\else 
\PackageError{si}{#1}{#2}% 
\fi 
\fi}
\newcommand*{\si@log@warn}[1]{% 
\ifsi@lognone \else 
\ifsi@logmin \else 
\PackageWarning{si}{#1}% 
\fi \fi}
\newcommand*{\si@log@inf}[1]{% 
\ifsi@lognone \else 
\ifsi@logmin \else 
\PackageInfo{si}{#1}% 
\fi \fi}
\si@log@debug

The debug macro only gives output if the appropriate package option is set.
\si@log@debug{⟨debug-information⟩}
\newcommand*{\si@log@debug}[1]{% 
\ifsi@lognone \else 
\ifsi@debug \else \fi 
\PackageInfo{si}{#1}% 
\fi \fi}

\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⟩}
\newcommand*{\sisetup}[1]{% 
\iffalse \fi \ifnum0='\else \fi 
\setkeys[si]{opt}{#1}% 
\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⟩}
\newcommand*{\si@opt@key}[2]{% 
\define@key[si]{opt}{#1}{#2\si@log@debug{Option #1 set to ##1}}}
\si@opt@cmdkey\si@opt@cmdkeys
The command versions of the above.
\si@opt@cmdkey[⟨default⟩]{⟨keyname⟩}{⟨function⟩}
\si@opt@cmdkeys[⟨default⟩]{⟨keynames⟩}
\newcommand*{\si@opt@cmdkey}[3]{%
\define@cmdkey[si]{opt}[si@]{#2}[#1]{#3}

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

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

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

\si@opt@choicekey A “fill in the blanks” choice key. In all cases, \texttt{\si@tempa} is used to hold the value given to the key, so that \texttt{\ifx} testing can occur. \texttt{\si@opt@choicekey[default]}\{\textit{keyname}\}\{\textit{choices}\}\{\textit{in-list}\}\{\textit{not-in-list}\}

\newcommand*{\si@opt@choicekey}[5]{%  
\define@choicekey*+[si]{opt}{#2}\[\texttt{\si@tempa}\]{#3}\[#1\texttt{\si@log@debug{Option #2 set to ##1}}\]{#4\texttt{\si@log@debug{Option #2 set to ##1}}}{#5\texttt{\si@log@debug{Option #2 set to ##1}}}}

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

\newcommand*{\si@opt@xchoicekey}[3]{%  
\texttt{\si@opt@choicekey[#3]}\{\texttt{#1}\texttt{\si@log@debug{Option #2 set to ##1}}}\texttt{\endcsname}\{\texttt{\nameuse{si@fix@##1}}}\texttt{\endcsname}\{\texttt{\nameuse{si@fix@#3}}}\texttt{\endcsname}\}

\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. \texttt{\si@opt@compatkey[package]}\{\textit{keyname}\}.

\newcommand*{\si@opt@compatkey}[2]{%  
\define@boolkey[si]{opt}[si@old@]{#2}\{true\}{\texttt{\si@log@debug{Emulating #1 package option\MessageBreak #2}}}\texttt{\sisetup{emulate=#1}}\texttt{\si@log@debug{Option #2 set to ##1}}}\texttt{\AtEndOfPackage{\texttt{\si@opt@disablekey{#2}}\MessageBreak \texttt{Compatibility option #2 only\MessageBreak available when loading si package}}}\texttt{\endcsname}}
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}

A quick method to set log=debug.

\si@opt@boolkey{debug}

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

\si@emulate

The emulate option is no longer valid once the package has been loaded.
The two \si@unitsep options control the size of spaces between the number and the unit (\si@valuesep), and that used to represent a product (\si@unitsep). Known values here are thin, med, medium, thick, cdot\textsuperscript{27} and none\textsuperscript{28}, other entries will be treated as custom spaces.

\si@xchoicekey{unitsep}{thin,med,medium,thick,space,none,cdot,times}{thin}
\si@xchoicekey{unitspace}{space,thin,med,medium,thick,none}{thin}
\si@xchoicekey{valuesep}{space,thin,med,medium,thick,none}{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.

\si@xchoicekey{digitsep}{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.

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

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

\si@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.
\newif\ifsi@textmode
\si@boolkey{textmode}
\si@choicekey{mode}{math,maths,text}
\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.

\if\si@textmodefalse
\renewcommand*{\si@tempb}{text}\
\ifx\si@tempa\si@tempb
\si@textmodetrue
\fi}\fi\else\fi
\log@warn{Unknown value '#1' for option mode}}

\textsuperscript{27}Only valid for unitsep.
\textsuperscript{28}Only valid for valuesep.
The package can work to match the font family (serif, sans serif, typewriter) of the surrounding text. This is controlled by a Boolean option.

The package can attempt to respect bold, or may ignore it.

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).

Italic is slightly different to bold, as there is no convenient switch for maths.

The fonts used by the package default to the obvious \LaTeX ones; however, this needs to be exposed to user modification. First the maths mode fonts are sorted out.

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

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

To allow numbers to be set in a different font to text, additional options are set up.

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.

The various valid characters are collected together in a single macro for later. In common with the above macros, this one starts \num.... The order here is the order the values are tested later on.
With four digits in a number, separating may or may not be desired. Note that this option is the same as one for `numprint`.
\si@opt@boolkey{sepfour}

The marker for multiplication in exponential numbers is set up.
\si@opt@xchoicekey{expproduct}{times,cdot}{times}

In the same area, the power for exponents is variable. Only one choice is given.
\si@opt@xchoicekey{exppower}{ten}{ten}

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

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

Unit prefixes can be given as either symbols or numerically.
\newif\ifsi@prefixnum
\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.
\si@tempb
\renewcommand*{\si@tempb}{power}\
\fii
\renewcommand*{\si@tempb}{leading}\
\fii
\si@prefixnumfalse
\renewcommand*{\si@tempb}{number}\
\fii
\si@prefixnumtrue
\fi
\ifx\si@tempa\si@tempb
\si@prefixnumtrue
\fi

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”).
\newif\ifsi@num@padlead
\newif\ifsi@num@padtrail
\si@opt@choicekey[all]{padnumber}{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.
\si@tempb
\renewcommand*{\si@tempb}{leading}\
\fii
\renewcommand*{\si@tempb}{lead}\
\fii
\si@num@padleadfalse
\renewcommand*{\si@tempb}{number}\
\fii
\si@num@padtrailfalse
\renewcommand*{\si@tempb}{trailing}\
\fii
\si@num@padtrailtrue
\fi
\ifx\si@tempa\si@tempb
\si@num@padtrailtrue
\fi

Some new switches for adding signs to numbers
\ifsii@num@signmant \ifsi@num@signexp
\fi\fi

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

The option is now processed.
\fi\fi

\renewcommand*{\si@tempb}{all}\
\ifx\si@tempa\si@tempb
  \si@num@signmanttrue
  \si@num@signexptrue
\fi
\renewcommand*{\si@tempb}{true}\
\ifx\si@tempa\si@tempb
  \si@num@signmanttrue
  \si@num@signexptrue
\fi
\renewcommand*{\si@tempb}{both}\
\ifx\si@tempa\si@tempb
  \si@num@signmanttrue
  \si@num@signexptrue
\fi
\renewcommand*{\si@tempb}{true}\
\ifx\si@tempa\si@tempb
  \si@num@signmanttrue
  \si@num@signexptrue
\fi
\renewcommand*{\si@tempb}{both}\
\ifx\si@tempa\si@tempb
  \si@num@signmanttrue
  \si@num@signexptrue
\fi}
\si@log@warn{Unknown value ‘#1’ for option addsign)}
\ifsii@ang@padsmall
  A switch for determining whether to typeset $\ang{;;1}$ as 0°0′1″ or 1″. First, two new Boolean switches are needed to indicate padding.
\fi
\ifsii@ang@padlarge
\fi
\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.
\ifsii@ang@padsmallfalse
  \si@ang@padsmallfalse
\fi
\ifsii@ang@padlargefalse
  \si@ang@padlargefalse
\fi
\renewcommand*{\si@tempb}{small}\
\ifx\si@tempa\si@tempb
  \si@ang@padsmalltrue
\fi
\renewcommand*{\si@tempb}{large}\
\ifx\si@tempa\si@tempb
  \si@ang@padlargetrue
\fi
\renewcommand*{\si@tempb}{all}\
\ifx\si@tempa\si@tempb
  \si@ang@padsmalltrue
  \si@ang@padlargetrue
\fi
\renewcommand*{\si@tempb}{true}\
\ifx\si@tempa\si@tempb
  \si@ang@padsmalltrue
  \si@ang@padlargetrue
\fi
\renewcommand*{\si@tempb}{both}\
\ifx\si@tempa\si@tempb
  \si@ang@padsmalltrue
  \si@ang@padlargetrue
\fi}
\si@log@warn{Unknown value ‘#1’ for option padangle)}
\ifsii@astroang
  A slightly odd option to allow the method used by astronomers for angles.
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.

Unit macros on their own may need \xspace.

The option processing for formatting units with \per{} in them needs two switches.

The usual value testing, with a default to use reciprocal powers.

For the slash option, the separator can be customised.

Macros for the right and left brackets added to potentially-ambiguous denominators.

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

\si@load \si@noload

The various non-Latin symbols need to be handled, and given user interfaces. Some definitions are more complex than others; for $\Omega$ things are easy.

\si@textOmega \si@mathsOmega

For the $\mu$ symbol, some direct loading of symbols is needed as the maths mu sign ($\mu$) is wrong.
\textdegree The angle signs.
\mathsdegree Finally, degrees Celsius, which may need the degree symbol.
\textringA The Å sign.
\mathsringA A flag for using textcomp and upgreek to provide better symbols.
\eVcorra Handling typographic conventions needs three keys. locale is used to set the locale, whereas \loadlocales reads in the definitions at package load time.
16.4 Compatibility options

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. That is left to the emulation files, loaded at the end of the package. First of all, the units options are dealt with; there are not many.

The unitsdef package is unfortunately much more profligate with options. The first set are to do with support for gensymb.

The second set are more general functionality.

The final set are for control of abbreviations, and are a good demonstration of why to use xkeyval!

The Slunits package has lots of options. These ones are all related to spacing.

These options are used by Slunits to control clashes with other packages.
\ifsi@old@textstyle The miscellaneous options.
\ifsi@old@binary \si@opt@compatkey{SIunits}{textstyle}
\ifsi@old@noams \si@opt@compatkey{SIunits}{binary}
\ifsi@old@derivedinbase \si@opt@compatkey{SIunits}{noams}
\ifsi@old@derived \si@opt@compatkey{SIunits}{derivedinbase}
\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 they are intended as constants. The package may rely on the contents of these macros for functionality.

First, there are the various space macros. To allow both \text{med} and \text{medium} to be used as a space description, two macros are needed for the same output.

\newcommand*{\si@fix@thin}{\,}
\newcommand*{\si@fix@med}{\:}
\newcommand*{\si@fix@medium}{\:}
\newcommand*{\si@fix@thick}{\;}
\newcommand*{\si@fix@space}{\text{~}}
\newcommand*{\si@fix@cdot}{\cdot}
\newcommand*{\si@fix@comma}{{,}}
\newcommand*{\si@fix@stop}{{.}}
\newcommand*{\si@fix@fullstop}{{.}}
\newcommand*{\si@fix@period}{{.}}
\newcommand*{\si@fix@times}{\times}
\newcommand*{\si@fix@plus}{+}
\newcommand*{\si@fix@minus}{-}
\newcommand*{\si@fix@pm}{\pm}
\newcommand*{\si@fix@mp}{\mp}
\newcommand*{\si@fix@two}{2}
\newcommand*{\si@fix@ten}{10}
\newcommand*{\si@fix@slash}{/}
\newcommand*{\si@fix@none}{\text{}}

Another optional component that will probably not be used by many people.
\newcommand*{\si@fix@slash}{/}
\newcommand*{\si@fix@none}{\text{}}

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

42
16.6 Symbols

\texttt{\textbackslash 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.

\begin{verbatim}
\newcommand*{\si@symbol}[1]{% 
  \expandafter\DeclareRobustCommand\expandafter*\expandafter{\csname si@sym@#1\endcsname}{% 
    \ifmmode \expandafter\csname si@maths#1\expandafter\endcsname% 
    \else \expandafter\csname si@text#1\expandafter\endcsname% \fi}}
\end{verbatim}

\texttt{\textbackslash si@symbol{Omega}} \texttt{\textbackslash si@symbol{ringA}} \texttt{\textbackslash si@symbol{mu}} \texttt{\textbackslash si@symbol{degree}} \texttt{\textbackslash si@symbol{minute}} \texttt{\textbackslash si@symbol{second}} \texttt{\textbackslash si@symbol{celsius}}

The various symbols are now declared.

\texttt{\textbackslash si@symbol{Omega}} \texttt{\textbackslash si@symbol{ringA}} \texttt{\textbackslash si@symbol{mu}} \texttt{\textbackslash si@symbol{degree}} \texttt{\textbackslash si@symbol{minute}} \texttt{\textbackslash si@symbol{second}} \texttt{\textbackslash si@symbol{celsius}}

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

\begin{verbatim}
\AtBeginDocument{% 
  \ifsi@redefsymbols \ifpackageloaded{textcomp} \if@logdebug{Redefining symbols using textcomp}% 
    \renewcommand*{\si@textdegree}{\textdegree}% \renewcommand*{\si@mathsdegree}{\text{\textdegree}}% \fi \else \fi \fi \fi}%
\end{verbatim}

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

\begin{verbatim}
\longdef\si@tempa{OT1}% \ifx\si@tempa\encodingdefault \renewcommand*{\si@mathsringA}{\text{\capitalring{A}}}% \renewcommand*{\si@textringA}{\capitalring{A}}% \fi}\fi %
\end{verbatim}

\texttt{\textbackslash upgreek}\texttt{\textbackslash @ifpackageloaded{upgreek}}\texttt{\if@logdebug{Redefining symbols using upgreek}% \renewcommand*{\si@mathsmu}{\upmu}% \renewcommand*{\si@mathsOmega}{\Upomega}}\fi\fi

16.7 Handling fractions

\texttt{\textbackslash si@frac} Various methods of handling fractions are provided.

\texttt{\textbackslash si@frc@frac} \texttt{\textbackslash si@frc@slash} \texttt{\textbackslash si@frc@nice} \texttt{\textbackslash si@frc@sfrc}
\newcommand*{\si@frc@frac}[2]{\ensuremath{\frac{\expandafter\si@unt@out\expandafter{#1}}{\expandafter\si@unt@out\expandafter{#2}}}}
\let\si@frac\si@frc@frac
\newcommand*{\si@frc@slash}[2]{\expandafter\si@unt@out\expandafter{#1}\si@out@text{\ensuremath{\si@slash}}\expandafter\si@unt@out\expandafter{#2}}
\newcommand*{\si@frc@nice}[2]{\ensuremath{\si@frc@nicefrac{\expandafter\si@unt@out\expandafter{#1}}{\expandafter\si@unt@out\expandafter{#2}}}}
\newcommand*{\si@frc@sfrac}[2]{\sfrac{\expandafter\si@unt@out\expandafter{#1}}{\expandafter\si@unt@out\expandafter{#2}}}
\AtBeginDocument{\if\@ifpackageloaded{xfrac}\else\fi{\si@log@inf{xfrac package unavailable}\MessageBreak using 'fraction=sfrac' will fall back on \nicefrac-like method}}
\renewcommand*{\si@frc@sfrac}[2]{\si@log@warn{xfrac package unavailable}\si@frc@nice{#1}{#2}}

\@ifpackageloaded{xfrac}{\si@frc@nicefrac}
\si@frc@displen
\si@frc@textlen
\si@frc@suplen
\si@frc@ssuplen
\newcommand*{\si@frc@nicefrac}{\if\@ifpackageloaded{xfrac}\@ifpackageloaded{xfrac package unavailable}\MessageBreak using 'fraction=sfrac' will fall back on \nicefrac-like method\fi{\newcommand*{\si@frc@nicefrac}[2]{\expandafter\si@frc@mathsnf{\expandafter\si@unt@out\expandafter{#1}}{\expandafter\si@unt@out\expandafter{#2}}}}\else\fi}
\si@frc@mathsnf{\langle\text{numerator}\rangle}{\langle\text{denominator}\rangle}
\newcommand*{\si@frc@mathsnf}[2]{\begingroup\settoheight{\si@frc@displen}{\displaystyle{M}}\settoheight{\si@frc@textlen}{\textstyle{M}}\settoheight{\si@frc@suplen}{\scriptstyle{M}}\settoheight{\si@frc@ssuplen}{\scriptscriptstyle{M}}\addtolength{\si@frc@displen}{-\si@frc@ssuplen}\addtolength{\si@frc@textlen}{-\si@frc@ssuplen}\addtolength{\si@frc@suplen}{-\si@frc@ssuplen}\mathchoice{\raisebox{\si@frc@displen}{\scriptstyle{#1}}}{\raisebox{\si@frc@textlen}{\scriptstyle{#1}}}{\scriptstyle{#1}}{\scriptscriptstyle{#1}}\endgroup}}

To avoid needing units installed, the \nicefrac macro needs to be emulated here. The code is taken (with permission) from kgnicefrac.\footnote{The original is licensed under the GPL; thanks to the author Axel Reichert for permission to copy the code here.}
\newlength{\si@frc@displen}
\newlength{\si@frc@textlen}
\newlength{\si@frc@suplen}
\newlength{\si@frc@ssuplen}
\newcommand*{\si@frc@nicefrac}{\if\@ifpackageloaded{xfrac}\@ifpackageloaded{xfrac package unavailable}\MessageBreak using 'fraction=sfrac' will fall back on \nicefrac-like method\fi{\newcommand*{\si@frc@nicefrac}[2]{\expandafter\si@frc@mathsnf{\expandafter\si@unt@out\expandafter{#1}}{\expandafter\si@unt@out\expandafter{#2}}}}\else\fi}
\si@frc@mathsnf{\langle\text{numerator}\rangle}{\langle\text{denominator}\rangle}
\newcommand*{\si@frc@mathsnf}[2]{\begingroup\settoheight{\si@frc@displen}{\displaystyle{M}}\settoheight{\si@frc@textlen}{\textstyle{M}}\settoheight{\si@frc@suplen}{\scriptstyle{M}}\settoheight{\si@frc@ssuplen}{\scriptscriptstyle{M}}\addtolength{\si@frc@displen}{-\si@frc@ssuplen}\addtolength{\si@frc@textlen}{-\si@frc@ssuplen}\addtolength{\si@frc@suplen}{-\si@frc@ssuplen}\mathchoice{\raisebox{\si@frc@displen}{\scriptstyle{#1}}}{\raisebox{\si@frc@textlen}{\scriptstyle{#1}}}{\scriptstyle{#1}}{\scriptscriptstyle{#1}}\endgroup}}
A stripped down version of the \texttt{nicefrac} system for text mode.

\begin{verbatim}
\newcommand*{\si@frc@textnf}[2]{\begingroup
\settoheight{\si@frc@textlen}{#1}
\settoheight{\si@frc@ssuplen}{\textfont{#1}}
\addtolength{\si@frc@textlen}{-\si@frc@ssuplen}
\raisebox{\si@frc@textlen}{\textfont{#1}}/\hbox{\textfont{#2}}\endgroup}
\end{verbatim}

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

\begin{verbatim}
\newcommand*{\si@frc@ugly}[1]{\def\si@tempa{#1}
\ifmmode\expandafter\si@frc@frac\else\def\si@tempb{1}
\ifx\si@tempa\si@tempb
The slash switch cannot be used, so the possibility of the numerator being one is handled here.
\setbox\si@tempbox=\hbox{\ensuremath{\si@valuesep}}
\hskip-\wd\si@tempbox
\def\si@tempa{1}
\else
\expandafter\si@frc@slash\fi\fi}
\end{verbatim}

\section{Font control}

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

\begin{verbatim}
\newcommand*{\si@fam@getmfam}{%  
  \sbox{0}{$%  
  \@ifundefined{mathsf}{\si@log@debug{No mathsf family found}%  
    \global\chardef\si@fam@sf=99}%  
  \mathsf{\global\chardef\si@fam@sf=\fam}}%  
  \@ifundefined{mathtt}{\si@log@debug{No mathtt family found}%  
    \global\chardef\si@fam@tt=99}%  
  \mathtt{\global\chardef\si@fam@tt=\fam}}%  
\AtBeginDocument{\si@fam@getmfam}
\end{verbatim}

The font families in use in the document are needed.

\begin{verbatim}
\newcommand*{\si@fam@ifbtext}{\si@fam@ifbmaths{%  
  \expandafter\si@fam@ifbtext%  \else  
  \expandafter\si@fam@ifbmaths%  \fi}}
\newcommand*{\si@fam@ifbmaths}{%  
  \def\si@tempa{\textbf{\textsf{bold}}}%  \if\math@version\si@tempa%  \fi}%  
\AtBeginDocument{\si@fam@getmfam}
\end{verbatim}

These tests check for bold in text and maths mode, respectively.

\begin{verbatim}
\newcommand*{\si@fam@ifbinline}{%  
  \if\si@obeymode  
  \if\is@maths\sisetup{mode=maths}%  \else\fi\fi}%  
\AtBeginDocument{\si@fam@getmfam}
\end{verbatim}

For compatibility with \texttt{units}, a method to change the behaviour when in inline maths is needed for the bold detector.

\begin{verbatim}
\newcommand*{\si@fam@mode}{%  
  \if\si@obeymode  
  \if\is@maths\sisetup{mode=maths}%  \else\fi\fi}%  
\AtBeginDocument{\si@fam@getmfam}
\end{verbatim}

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

\begin{verbatim}
\newcommand*{\si@fam@mode}{%  
  \if\si@obeymode\expandafter\@car\f@series\@nil\else\fi}%  
\AtBeginDocument{\si@fam@getmfam}
\end{verbatim}

Detection of the current mode needs to happen “early” (before any change of \texttt{\ensuremath}). So a short macro is provided to do the job.

\begin{verbatim}
\newcommand*{\si@fam@getmfam}{%  
  \if\si@obeymode\expandafter\@car\f@series\@nil\else\fi}%  
\AtBeginDocument{\si@fam@getmfam}
\end{verbatim}

\newcommand*{\si@fam@getmfam}{%  \sbox{0}{$%  \@ifundefined{mathsf}{\si@log@debug{No mathsf family found}%  \global\chardef\si@fam@sf=99}%  \mathsf{\global\chardef\si@fam@sf=\fam}}%  \@ifundefined{mathtt}{\si@log@debug{No mathtt family found}%  \global\chardef\si@fam@tt=99}%  \mathtt{\global\chardef\si@fam@tt=\fam}}%  \AtBeginDocument{\si@fam@getmfam}
A marker is set up to check if font-matching has been taken place.

Using the code from `sistyle` as a base, a set of tests are used to set the current font families and weights.

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

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

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.

Inline maths is now handled.
\begin{verbatim}
\expandafter\let\expandafter\si@fam@text\csname\si@textsf\endcsname
\else
  \ifx\f@family\si@tempb
    \si@log@debug{Font detection: tt}\%
    \expandafter\let\expandafter\si@fam@maths\csname\si@mathstt\endcsname
    \expandafter\let\expandafter\si@fam@text\csname\si@texttt\endcsname
  \else
    \si@log@debug{Font detection: rm}\%
    \expandafter\let\expandafter\si@fam@maths\csname\si@mathsdefault\endcsname
    \expandafter\let\expandafter\si@fam@text\csname\si@textdefault\endcsname
  \fi
\fi
\fi
Not in maths mode, so the text mode checks are carried out.
\else
  \si@log@debug{Font detection: text}\%
  \ifx\f@family\si@tempa
    \si@log@debug{Font detection: sf}\%
    \expandafter\let\expandafter\si@fam@maths\csname\si@mathssf\endcsname
    \expandafter\let\expandafter\si@fam@text\csname\si@textsf\endcsname
  \else
    \ifx\f@family\si@tempb
      \si@log@debug{Font detection: tt}\%
      \expandafter\let\expandafter\si@fam@maths\csname\si@mathstt\endcsname
      \expandafter\let\expandafter\si@fam@text\csname\si@texttt\endcsname
    \else
      \si@log@debug{Font detection: rm}\%
      \expandafter\let\expandafter\si@fam@maths\csname\si@mathsdefault\endcsname
      \expandafter\let\expandafter\si@fam@text\csname\si@textdefault\endcsname
    \fi
  \fi
\fi
If the local font is not to be matched, setting the fonts is rather less complex.
\else
  \si@log@debug{Font detection: inactive}\%
  \expandafter\let\expandafter\si@fam@maths\csname\si@mathsdefault\endcsname
  \expandafter\let\expandafter\si@fam@text\csname\si@textdefault\endcsname
\fi
\si@fam@bold \si@fam@setbold
With the font family set, the next check is for bold text. This again needs to
\end{verbatim}
examine the current mode. Things are a bit more complex than in\textit{style} as it is possible to be typesetting in either text or maths mode. The bold commands are set up with $\texttt{\textbackslash def}$, as nested calls can occur.

\begin{verbatim}
\def\si@fam@bold{\unboldmath\mdseries}\
\def\si@fam@setbold{\boldmath\bfseries}\
\ifsi@obeybold\si@log@debug{Weight detection: checking weight}\
\ifmmode\Display maths.\else Text mode.\fi\fi
\ifdim\displaywidth>0pt\relax\si@fam@ifbmaths\else\si@fam@ifbinline\fi
\let\si@fam@bold\si@fam@setbold\si@log@debug{Weight detection: bold weight}\fi
\let\si@fam@italic\upshape\ifsi@obeyitalic\si@log@debug{Italic detection: checking italic}\
\si@fam@ifitext\else\si@log@inf{maths mode - obeyitalic inactive}\fi\fi
\end{verbatim}

16.9 Formatting numbers

\texttt{\num} The system used here is modelled on that in\texttt{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\texttt{numprint}. The user macro $\texttt{\num}$ sets any local keys, then calls the number formatting macro on the processed number.

\begin{verbatim}
\DeclareRobustCommand*{\num}{\num[\texttt{\langle options\rangle}]\texttt{\langle num\rangle}}\fi
\end{verbatim}
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 \texttt{\si@out@...} macros. No grouping is applied here; any call to \texttt{\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.

\texttt{\si@num}\[⟨num⟩\] The argument of the macro is fully expanded before any processing. By using \texttt{\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 \texttt{\scantokens}, hence the use of \texttt{\@empty}.

\texttt{\si@ifnotmtarg{\si@tempa}{}} The input is now validated. Further processing takes place a little later.

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

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.

\texttt{\si@num@xdef} When carrying out the \texttt{\edef} used to fully-expand a number, \texttt{\,} and \texttt{~} 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.
\sionum\xdef(macro)(num)\@empty
\sionum\xdef\xdef#1\@empty\xdef#1(#2)\}
\sionumvalid Assuming that there is a non-space argument to \sionum, 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.
\sionumvalid(char)(chars)\@empty
\sionum@ifchr{#1}{\sionumvalid}{%}
\ifx\@empty#2\@empty\else
\sionumvalid#2\@empty\@empty\@empty%\fi
\si@switchtrue}%
If an invalid character has been picked up, the whole parsing system has to bail out.
\si@log@err{Invalid character '#1' in numerical input}%
\si@num@ifchr{⟨test-chars⟩}{⟨valid-chars⟩}{}
\newcommand*{\si@num@ifchr}[2]{% 
\begingroup \si@switchfalse 
\def\si@tempa{#1} \edef\si@tempb{#2} 
Now the test can occur for the initial comparison string.
\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.
\ifi\si@switch% 
\endgroup\expandafter\@firstoftwo\else
\endgroup\expandafter\@secondoftwo\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
\def\si@num@chrstr#1\@empty%\si@tempb is used to hold the single character to check against \si@tempa, while \si@tempb stores the remaining characters to be compared.
If \texttt{\@empty} \texttt{\@empty}, then the recursion has bottomed-out, and all of the comparisons are done. If not, go round again.

Various storage macros are needed.

The number processor starts by saving \texttt{#1} (odd things happen otherwise), and locally clearing the stacks.

The input is split into an mantissa and an exponent.

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 \texttt{\@empty} \texttt{\@empty}. This will change if there is a number in \texttt{\@empty} \texttt{\@empty}. 

52
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.

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

With everything done, the result is output.

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.

When building up the mantissa and exponent, everything must be expandable, so \edef can be used rather than \g@addto@macro.
If the recursion has not bottomed out, another loop occurs.

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.

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

If a sign has to be added to unsigned numbers, this is done here.

There is a sign, so it is added to the output stack.
A sign but no number can only be correct if the input is something like \(-e10\) to give \(-10^{10}\).

\(\text{\texttt{\textbackslash si@tempb}}\) is no longer needed, so can be reused.

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

The first one or two characters of the mantissa or exponent may contain a sign. To test for this, the first two characters of the number are split off, and examined. 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.

\begin{verbatim}
\def\si@num@gensign#1#2#3\@empty{
  \ifx\si@num@gensign#1#2#3\@empty\@empty
    \ifx\empty\si@tempb\empty
      \expandafter\def\csname si@num@#1\endcsname{}\%
    \else
      \edef\si@tempa{mant}\%
      \if\si@tempa\si@tempc\%
        \ifx\@empty\si@num@exp\@empty
          \si@log@warn{Sign but no number for ‘\si@num@arg’}%
        \else
          \si@log@warn{Sign but no number for ‘\si@num@arg’}%
        \fi
        \else
          \expandafter\protected@edef\csname si@num@#1\endcsname si@num@#1\@empty\%
          {\si@tempb}%
        \fi
      \fi
    \fi
  \else
    \edef\si@num@gensign#1#2#3\@empty{
      \si@num@ifchr{#1}{\si@numsign}{%
        \si@num@ifchr{#2}{\si@numsign}{%
          \if #1
            \if -#2
              \si@log@debug{Found sign combination +- for ‘\si@num@arg’}%
              \def\si@tempa{pm}\%
            \else
              \si@log@warn{Unknown sign combination ‘#1#2’}%
              \def\si@tempa{#1#2}\%
            \fi
          \else
            \if +#1
              \if +#2
                \si@log@debug{Found sign combination ++ for ‘\si@num@arg’}%
                \def\si@tempa{pm}\%
              \else
                \si@log@warn{Unknown sign combination ‘#1#2’}%
                \def\si@tempa{#1#2}\%
              \fi
            \else
              \if +#2
                \si@log@debug{Found sign combination ++ for ‘\si@num@arg’}%
                \def\si@tempa{pm}\%
              \else
                \si@log@warn{Unknown sign combination ‘#1#2’}%
                \def\si@tempa{#1#2}\%
              \fi
            \fi
          \else
            \if +#2
              \si@log@debug{Found sign combination ++ for ‘\si@num@arg’}%
              \def\si@tempa{pm}\%
            \else
              \si@log@warn{Unknown sign combination ‘#1#2’}%
              \def\si@tempa{#1#2}\%
            \fi
          \fi
        \fi
      \fi
    \fi
\end{verbatim}
Only one valid sign character.

\edef\si@tempb{#3}}%

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

\edef\si@tempa{}%

\si@num@digits\si@num@digits\si@num@digits\si@tempb\si@tempc

The core digit processor divides the number into the parts before and after the decimal point marker.
\begin{itemize}
\item \csname si@num@mant\endcsname\csname si@num@exp\endcsname
\end{itemize}

The package switch is used to indicate finding a decimal marker.
\si@switchfalse
\expandafter\expandafter\expandafter\si@num@split%
\csname si@num@mant\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.
\ifx\@empty\si@tempa\@empty
\if\si@num@padlead
\si@log@debug{Adding leading zero for \textbackslash\textbackslash si@num@arg'}%
\def\si@tempa{0}%
\fi
\fi
\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 \si@switch can be reused.
\ifx\@empty\si@tempb\@empty
\si@num@padtrail
\si@log@debug{Adding trailing zero for \textbackslash\textbackslash si@num@arg'}%
\def\si@tempb{0}%
\fi
\fi
\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.
\ifx\@empty\si@tempa\@empty
\if\si@num@padlead
\else
\si@num@sepdigits{#1}%
\fi
\fi
\ifx\@empty\si@tempb\@empty
\else
\si@num@sepdigits{#1}%
\fi
\fi
The \texttt{\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 \texttt{\si@tempa}; after a decimal is found, characters are added to \texttt{\si@tempb}.

\begin{verbatim}
def\si@num@split#1#2\@empty{\if\@empty#2\@empty\else\si@num@split#2\@empty\@empty\fi}{% \if\@empty#1\@empty\else\si@log@err{Duplicate decimal marker in \texttt{\si@num@arg}}{\MessageBreak\Only a single decimal marker (from the list \texttt{\si@numdecimal}) may occur in a numerical argument}% \else\si@log@debug{Found decimal marker \texttt{"#1"} in \texttt{\si@num@arg}}%\si@switchtrue\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.\si@num@ifchr{#1}{\si@numsign}{%\si@log@err{Misplaced sign in \texttt{\si@num@arg}}{Sign characters \texttt{\si@numsign} can only occur at the start of a number}% The current character is added to the appropriate stack.\si@num@ifextra{\si@tempa}{}{\protect\edef\si@tempb{\si@tempb#1}}\si@log@debug{Adding \texttt{"#1"} to integer part for \texttt{\si@num@arg}}\protect\edef\si@tempa{\si@tempa#1}\fi}}% Unless the recursion has bottomed, loop round again.\fi}%\si@num@decimalhook A hook is needed to attach things inside the group to happen afterwards, if the number is a decimal.\newcommand*{\si@num@decimalhook}{}\si@num@sepdigits The \texttt{\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.\newcommand*{\si@num@sepdigits}{%\si@num@ifextra{\si@tempa}{}{\expandafter\si@num@int\expandafter{\si@tempa}}}\si@num@sepdigits\expandafter{\si@num@int\expandafter{\si@tempa}}% If the input is a pure number, then separation is attempted.
The construction is finalised by re-combining the number.

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

The formatting code for separating thousands is taken more-or-less directly from \texttt{si-style}. A few changes are made to fit the various conventions here. Following on from the code above, \texttt{\si@tempa} is used to store the integer part of the number, and \texttt{\si@tempb} is used for the decimal part.
\si@num@iffive\[1\]{\si@num@five#1@empty@empty@empty@empty@empty@end}
\def\si@num@five#1#2#3#4#5@end{
  \ifx\@empty#5\@empty
    \expandafter\@secondoftwo\else
    \expandafter\@firstoftwo\fi}
\si@num@intfmt The business end of the integer formatter. \si@num@intfmt{(char)}{(char)}{(char)}{(char)}
\si@num@fiint \si@num@fiint{(char)}{(char)}{(char)}{(char)}
\newcommand*{\si@num@intfmt}[4]{{\ifx\@empty#4\@empty
  \si@num@intsep#1@empty@empty@empty@empty@empty@empty@empty\else
  \expandafter\@secondoftwo\else
  \expandafter\@firstoftwo\fi}\fi\fi\fi\fi}
\si@num@intsep For adding separation to integers, an extra function is needed.
\si@tempa \si@tempa{(char)}{(char)}{(char)}{(char)}
\newcommand*{\si@num@intsep}[4]{{\protected@edef\si@tempa{\si@tempa#1#2#3}\if\relax#4\relax
  \else
  \protected@edef\si@tempa{\si@tempa\noexpand\ensuremath{\noexpand\si@digitsep}}\expandafter\si@num@intsep\expandafter#4\fi}
\si@num@dec Formatting a decimal uses a similar mechanism, but with a few alterations needed.
\si@tempb \si@tempb{(decimal-part)}\si@tempb{(char)}{(char)}{(char)}{(char)}
\newcommand*{\si@num@dec}[1]{{\def\si@tempb{}\if\si@sepfour
  \si@num@decfmt#1@empty@empty@empty@empty@empty\else
  \si@num@iffive{#1}{\si@num@decfmt#1@empty@empty@empty@empty@empty}{\protected@edef\si@tempb{\si@tempb#1}}\fi}}
16.10 Formatting angles

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

\ang\ang\ang\ang\ang

A mechanism is needed to handle moving the angle unit sings for the astroang option. This requires two steps, producing the sign over the decimal sign and preventing duplicate symbols appearing.

\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@degs

First, the three macros that will contain the measures must exist.
Either the signs need to be moved, or this needs to be killed off.

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

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

The group opened by \ang is closed.
Padding is only added if requested; the zero is a literal.
\begin{verbatim}
\if\si@ang@padsmall
#1\%
\else
\relax\%
\fi
\end{verbatim}

Modified versions of \num, one to typeset angles without a leading sign and the other with.
\begin{verbatim}
\newcommand*{\si@ang@num}{⟨degree/minute/second⟩⟨num⟩}
\newcommand{\si@ang@signlessnum}{⟨degree/minute/second⟩⟨num⟩}
\end{verbatim}

\section{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.

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.

An optional argument can now be picked up (this does not work using the 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.
With the assignment done, the normal action of the array package is continued.

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

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.

The current cell could be the end of a line.

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.

If \begin...\end has not been used for the table, then \endtabular might crop up.
\let\si@tab@next\endtabular
\else

Apparently, \texttt{tabularx} might have a \texttt{\csname} at the end of the cell.
\ifx\csname#1
\let\si@tab@next\csname
\else
\relax
is always a possibility.
\ifx\relax#1\relax
\let\si@tab@next\relax
\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 \texttt{\@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.
\let\si@tab@next\si@tab@gettok
\si@num@ifchr{#1}{\si@numvalid}
{\si@switchtrue
\si@log@debug{Found valid cell contents '#1'}\
\si@tab@numtoks=\expandafter{\the\si@tab@numtoks#1}}
{\si@log@debug{Found other cell contents \string#1}\
\si@tab@othertok{#1}}
\fi
\fi
\fi
\fi
\fi
\fi
\fi

Finally, execute whatever should be the next step.
\si@tab@next}

\si@tab@othertok  Unrecognised input is added to a token register, either before or after the number.
\si@tab@othertok{\texttt{\{chars\}}}
\newcommand*{\si@tab@othertok}[1]{%
\if\si@switch
\expandafter\si@tab@numout%
\else
\fi}

\si@tab@end  At the end of the cell, the actual output has to occur.
\newcommand{\si@tab@end}{%
\hfil%
\the\si@tab@pretoks%
\if\si@switch
\expandafter\si@tab@numout%
\else
\fi}
Counters are needed for the digit-counting system.

If a number is found, then some secondary processing is needed to format it correctly.

A storage macro is needed.

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

The crucial sub-macro is redirected.

With that done, things continue as in the original.

The pre-decimal part of the number is in \texttt{\the\texttt{\@tab@mantout}}, with the post-decimal part in \texttt{\@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 \texttt{\@tab@num\@out}.

For the exponent, processing is back to normal.
\si@num@digits{exp}%
\ifx\@empty\si@num@mant\@empty\else
\protected@edef\si@num@out%
  {\si@num@out\noexpand\textsuperscript{\noexpand\si@exppower}}%
\fi
\protected@edef\si@num@out%
  {\si@num@out\si@exppower%
    \noexpand\textsuperscript{\noexpand\si@num@expout}}%
\fi
\ifx\@empty\si@num@mant\@empty
  \ifx\@empty\si@num@exp\@empty
    \si@log@err{Invalid number format \si@num@arg'}%
    {Something is wrong with the number format; does it contain \MessageBreak any numbers (from the list \si@numlist')?}%
  \else
    \si@num@ifextra{\si@tempa}{\expandafter\si@num@int\expandafter{\si@tempa}}%
  \fi
\fi
\newcommand*{\si@tab@num@sepdigits}{\langle num \rangle}
\newcommand*{\si@tab@num@sepdigits}{\expandafter\protected@edef\csname si@num@#1out\endcsname{\csname si@num@#1out\endcsname\si@tempa}%
\expandafter\protected@edef\csname si@tab@#1out\endcsname{\si@tempb}}%
\si@tab@format
\newbox\si@tab@prebox
\newbox\si@tab@postbox
\newbox\si@tempbox
\si@tab@format
\si@tab@num@sepdigits
\si@tab@num@sepdigits
Need to clear both storage areas.
\renewcommand*{\si@num@out{}}%
\renewcommand*{\si@tab@mantout{}}%
\fi
\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{(num)}
\newcommand*{\si@tab@num@sepdigits}{\langle num \rangle}
\newcommand*{\si@tab@num@sepdigits}{\expandafter\protected@edef\csname si@num@#1out\endcsname{\csname si@num@#1out\endcsname\si@tempa}%
\expandafter\protected@edef\csname si@tab@#1out\endcsname{\si@tempb}}%
\si@tab@prebox
\si@tab@postbox
\si@tempbox
The various boxes needed for the column centring are declared Unlike the dcolumn 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.
\newbox\si@tab@prebox
\newbox\si@tab@postbox
\newbox\si@tempbox
\si@tab@format
The formatting set up is taken from dcolumn, 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 dcolumn, so that a value of zero gives a column centred on
the decimal marker.
\def\si@tab@format{\relax
  \ifnum\z@<\si@tempcnta
    \expandafter\si@tab@right\relax%
  \else
    \expandafter\si@tab@centre\relax%
  \fi
  (#1)%

Output of the formatted data occurs here; both positioning macros produce
formatted data in boxes zero and two.
\box\si@tab@prebox\box\si@tab@postbox}

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.
\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.
  \setbox\si@tab@prebox=\hbox{
    \expandafter\si@out@num\expandafter{\si@num@mantout}}%
  \ifx\@empty\si@tab@mantout\@empty
    \ifx\@empty\si@num@out\@empty
      \setbox\si@tab@postbox=\hbox{\phantom{\ensuremath{{\si@decimalsign}}}}%
    \else
      \setbox\si@tab@postbox=\hbox{\expandafter\si@out@num\expandafter{\si@num@out}}%
    \fi
  \else
    \setbox\si@tab@postbox=\hbox{\ensuremath{{\si@decimalsign}}%
    \expandafter\si@out@num\expandafter{\si@num@out}}%
  \fi
  \ifdim \wd\si@tab@prebox>\wd\si@tab@postbox
    \setbox\si@tab@postbox=\hbox to\wd\si@tab@prebox{\unhbox\si@tab@postbox\hfill}%
  \else
    \setbox\si@tab@prebox=\hbox to\wd\si@tab@postbox{\unhbox\si@tab@prebox\hfill}%
  \fi
}

Which of the two boxes is wider is now checked, and the smaller is padded out.
\ifdim \wd\si@tab@prebox>\wd\si@tab@postbox
  \setbox\si@tab@postbox=\hbox to\wd\si@tab@prebox{\unhbox\si@tab@postbox\hfill}%
\else
  \setbox\si@tab@prebox=\hbox to\wd\si@tab@postbox{\unhbox\si@tab@prebox\hfill}%
\fi

Some storage dimensions are declared.
\newdimen{\si@tab@predim} \newdimen{\si@tab@postdim} \newdimen{\si@tempdima} \newdimen{\si@tempdimb}

67
The column is not centred on the decimal marker; the user specifies how many
caracters on each side are allowed for.
\newcommand*{\si@tab@right}{(num)}
\newcommand*{\si@tab@right}[1]{% 
The width of a character is measured, and stored.
\setbox\si@tempbox=\hbox{\si@out@num{1}}
\setbox\si@tempdima=\wd\si@tempbox

If #1 is empty, then no special processing is needed for box two. On the other
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.
\ifx\relax#1\relax
\hfill
\let\si@tab@prea\relax
\let\si@tab@preb\relax
\else
\si@tab@predim\the\si@tempcnta\si@tempdima
\edef\si@tab@prea{to\si@tab@predim}\
\edef\si@tab@preb{\hss\hfill}\
\si@tempcnta\@gobble#1\relax
\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.
\setbox\si@tempcnta=\hbox{\ensuremath{\{\si@decimalsign\}}}%
\advance\si@tempdima\wd\si@tempbox

The pre-decimal part of the number is now added to box zero, with the post
decimal part in box two if needed.
\setbox\si@tempbox=\hbox{\si@tab@prea\si@tab@preb}%
\expandafter\si@out@num\expandafter{\si@num@mantout}}%
\ifx\@empty\si@tab@mantout\@empty
\setbox\si@tempbox=\hbox to\si@tab@postdim{\ensuremath{{\si@decimalsign}}\expandafter\si@out@num\expandafter{\si@num@out}\hfil}%
\else
\setbox\si@tempbox=\hbox to\si@tab@postdim{\expandafter\si@out@num\expandafter{\si@num@out}\hfil}%
\fi

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.
\newcommand*{\si@tab@sepcorr}[1]{%
Calculate how many groups of three there are, then allow for not separating four characters if \
\texttt{\textbackslash{ifs}i@sepfour} is false.

\begin{verbatim}
1412 \divide\texttt{\textbackslash{s}i@tempcntb}\thr@@
1413 \texttt{\textbackslash{i}fsi@sepfour}\texttt{\textbackslash{else}}
1414 \texttt{\textbackslash{ifnum}\texttt{\textbackslash{the}}\texttt{\textbackslash{s}i@tempcnta}=4}
1415 \texttt{\textbackslash{s}i@tempcntb}\texttt{\textbackslash{z}@}
1416 \texttt{\textbackslash{fi}}
1417 \texttt{\textbackslash{fi}}
1418 \setbox\texttt{\textbackslash{s}i@tempbox}=\hbox{\ensuremath{\texttt{\textbackslash{s}i@digitsep}}}\%}
1419 \expandafter\texttt{\textbackslash{advance}}\texttt{\textbackslash{csname} si@tab@#1dim}\texttt{\textbackslash{endcsname}}\%
1420 \texttt{\textbackslash{s}i@tempcntb}\texttt{\textbackslash{wd}}\texttt{\textbackslash{s}i@tempbox}\
\end{verbatim}

\begin{verbatim}
16.12 Units
\end{verbatim}

\begin{verbatim}\SI \unitsym
\newunit \renewunit \provideunit
\end{verbatim}

\begin{verbatim}
1421 \DeclareRobustCommand*{\SI}[2][{}]{
1422 \@ifnextchar[{}
1423 {\texttt{\textbackslash{s}i@SI}[#1]{#2}}
1424 {\texttt{\textbackslash{s}i@SI}[#1]{#2}}}
1425 \DeclareRobustCommand*{\unitsym}[2][{}]{
1426 \texttt{\textbackslash{s}i@SI}[{}]{#2}}
\end{verbatim}

\begin{verbatim}
\newcommand*{\newunit}[3][{}]{
\texttt{\textbackslash{s}i@ifdefinable}{#2}
{\texttt{\textbackslash{s}i@unt@defunit}[#1]{#2}{#3}}
\texttt{\textbackslash{s}i@log@err{Unit \string#2 already defined!}\@eha}}
\end{verbatim}

\begin{verbatim}
\newcommand*{\renewunit}[3][{}]{
\texttt{\textbackslash{s}i@ifdefinable}{#2}
{\texttt{\textbackslash{s}i@unt@defunit}[#1]{#2}{#3}}
\texttt{\textbackslash{s}i@unt@defunit}[#1]{#2}}}
\end{verbatim}

\begin{verbatim}
\newcommand*{\provideunit}[3][{}]{
\texttt{\textbackslash{s}i@ifdefinable}{#2}
{\texttt{\textbackslash{s}i@unt@defunit}[#1]{#2}{#3}}
}\
\end{verbatim}

\begin{verbatim}
1426 \newcommand*{\newunit}[3][{}]{
1427 \texttt{\textbackslash{s}i@ifdefinable}{#2}
1428 {\texttt{\textbackslash{s}i@unt@defunit}[#1]{#2}{#3}}
1429 {\texttt{\textbackslash{s}i@log@err{Unit \string#2 already defined!}\@eha}}
1430 \newcommand*{\renewunit}[3][{}]{
1431 \texttt{\textbackslash{s}i@ifdefinable}{#2}
1432 {\texttt{\textbackslash{s}i@log@err{Unit \string#2 undefined}\@ehc}}
1433 \texttt{\textbackslash{s}i@unt@defunit}[#1]{#2}{#3}}
1434 {\texttt{\textbackslash{s}i@unt@defunit}[#1]{#2}{#3}}
1435 \newcommand*{\provideunit}[3][{}]{
1436 \texttt{\textbackslash{s}i@ifdefinable}{#2}
1437 {\texttt{\textbackslash{s}i@unt@defunit}[#1]{#2}{#3}}
1438 {}}\end{verbatim}
The multiples of units are defined here; very similar code is used to the \newunit, etc., macros. The multiple prefixes cannot take an optional argument, and must represent some power. Hence the arguments required are different.

\newprefix{(multiple)}{(powers-ten)}{(symbol)}
\renewprefix{(multiple)}{(powers-ten)}{(symbol)}
\provideprefix{(multiple)}{(powers-ten)}{(symbol)}

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

\newpower[\text{post}]{\text{num}}{\text{power}}
\renewpower[\text{post}]{\text{num}}{\text{power}}
\providepower[\text{post}]{\text{num}}{\text{power}}

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.

\ifs@unt@num
\fi
\s@SI \s@@SIopts

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

\def\s@SI[#1]{\begin{group}
\ifs@notmtarg[#1]
\sisetup[#1]
\edef\s@unt@SIopts{#1}
\fi
\s@SI[#1]{#2}{#3}{#4}
\end{group}
The prefix unit is handled before any processing of the number; the flags are set to get spacing correct.

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

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.

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

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).

The unit includes one or more literal items; typeset using the unit typesetting macro.
For processable output, the argument is executed; the macros are all designed for this.

\texttt{\texttt{\textbackslash si@log@debug\{Macro unit found:\MessageBreak processing to format output\}\%}}
\texttt{\texttt{\textbackslash si@unt@init\%}}
\texttt{\texttt{\textbackslash advance\textbackslash si@unt@depthcnt\textbackslash ne\textbackslash relax}}
\texttt{\texttt{\textbackslash i\%}}
\texttt{\texttt{\textbackslash si@unt@final\}}} 

\texttt{\texttt{\textbackslash si@unt@addvaluesep}} To ensure no problems pop up with expansion, adding the value–unit space is handled by a macro.
\texttt{\texttt{\textbackslash si@unt@addvaluesep}}
\texttt{\texttt{\textbackslash si@unt@litvalsep}}
\texttt{\texttt{\textbackslash si@unt@stackvalsep}}
\texttt{\texttt{\textbackslash si@unt@addvalsep}}
\texttt{\texttt{\textbackslash si@unt@litvalsep}}
\texttt{\texttt{\textbackslash si@unt@stackvalsep}}
\texttt{\texttt{\textbackslash si@unt@spstack}}
\texttt{\texttt{\textbackslash si@unt@stacka}}
\texttt{\texttt{\textbackslash si@unt@stackb}}
\texttt{\texttt{\textbackslash si@unt@unitcnta}}
\texttt{\texttt{\textbackslash si@unt@unitcntb}}
\texttt{\texttt{\textbackslash si@unt@depthcnt}}
\texttt{\texttt{\textbackslash si@unt@litoutfalse}}
\texttt{\texttt{\textbackslash si@unt@firsttrue}}
\texttt{\texttt{\textbackslash si@unt@perfalse}}
\texttt{\texttt{\textbackslash si@unt@perseenfalse}}
\texttt{\texttt{\textbackslash si@unt@prepowerfalse}}
\texttt{\texttt{\textbackslash si@unt@depthcnt\textbackslash m\textbackslash ne\textbackslash relax}}
\texttt{\texttt{\textbackslash si@unt@init\}}} 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 \texttt{\textbackslash SI}.
\texttt{\texttt{\textbackslash si@unt@init\}}}
The finalisation macro finishes off the output and resets the flags.

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⟩\}

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.

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.

For literal output, the third argument is all that is needed. \si@unt@defprefix\{⟨multiple⟩\}\{⟨powers-ten⟩\}\{⟨symbol⟩\}

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

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

Once again the optional argument is saved.

\expandafter\expandafter\expandafter\edef\expandafter\csname\expandafter\@gobble\string#2@opt@si\endcsname{#1}

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.

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@unithook
\si@unt@unit

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

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

\begin{verbatim}
\si@unt@ifliteral{#3}
{\si@unt@addtostack{unit}{#3}\
  \ifsi@unt@prepower
  \expandafter\si@unt@stkpower\
  \fi}
{#3}\
\end{verbatim}

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

\begin{verbatim}
\advance\si@unt@depthcnt\m@ne\relax
\ifnum\si@unt@depthcnt=\z@\relax
\expandafter\si@unt@final\
\fi}
\si@unt@firstorsecond
\end{verbatim}

\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.

\begin{verbatim}
\si@unt@firstorsecond{{\textit{num}}}{{\textit{macro}}}
\newcommand{{\si@unt@firstorsecond}}[2]{%\ifs@unt@first
\expandafter\si@unt@first\
\else
\expandafter\si@unt@second\
\fi}
(#1){#2}\
\end{verbatim}

\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.

\begin{verbatim}
\si@unt@first{{\textit{num}}}{{\textit{unit}}}
\newcommand*{{\si@unt@first}}[2]{%\si@ifnotmtarg{#1}\
{\num{#1}}\
\si@unt@unithook\
\ifcsname\expandafter\@gobble\string#2@opt@unt@si\endcsname
\expandafter\si@unt@setopts\
\else
\expandafter\@gobble\
\fi}
(#1){#2}\
\end{verbatim}

\si@unt@setopts
A rather long set of \expandafter commands to get the options to set safely.

\begin{verbatim}
\si@unt@setoptions{\unit}
\newcommand*{{\si@unt@setoptions}}{1}{%\expandafter\expandafter\expandafter\expandafter\expandafter\expandafter\expandafter%
The user options are reloaded, if defined, to ensure that they still work as expected.

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.

A check is made to avoid adding \(-1\) to prefixes. If \textsf{frac} is active, then the \textsf{b} stack will be in use, otherwise it will be \textsf{a}.

A check to prevent adding \(-1\) at the very beginning of the unit, where there is a space on the stack.

Actual output of the prefixes.
To add the prefix, a little translation is needed.
\newcommand*{\si@addtostack}{prefix}{#2}

On the other hand, to count the prefix numeral, the symbol is thrown away.
\newcount{\si@prefixcnt}
\newcommand*{\si@countprefix}{\langle \text{powers-ten} \rangle}{\langle \text{gobble} \rangle}
\newcommand*{\si@invprefix}{\textsuperscript{num}}

For literal power output, the number can't simply be dumped, so a macro is needed.
\newcommand*{\si@litpower}{\langle \text{gobble} \rangle}{\langle \text{num} \rangle}

The handling of powers starts by checking if the number needs to be reversed.
\newif{\si@prepower}
\newcommand*{\si@power}{\langle \text{power} \rangle}{\langle \text{num} \rangle}

To do sign-inversion on the power, a dimension is used (this allows non-integer values to be handled).
\newdimen{\si@powerdim}
Adding powers to the stack should also clear the power list. If the number is already zero, then of course nothing happens.

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

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

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

A macro to change the sign of the current power.

The \texttt{\per} macro sets the correct flags; almost everything else is done elsewhere. There is always the case of two \texttt{\per} instructions; so the flag is flipped rather than set arbitrarily. The second flag is needed so that \texttt{\per} can give powers of $-1$ properly.
A test is needed for adding $-1$ when needed. The second macro is fired only if the power should be reciprocal.

\newcommand*{% si@unt@reciptest \ifsi@unt@per \ifsi@unt@perseen \expandafter\expandafter\expandafter\si@unt@recip \fi \fi}

\newcommand*{% si@unt@recip si@unt@powerdim\m@ne pt\relax si@unt@stackpower}

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

\si@unt@addtostack{% type \langle token \rangle}

\newcommand*{% si@unt@lastadda}
\newcommand*{% si@unt@lastaddb}
\newcommand*{% si@unt@addtostack}[2]{% \edef\si@tempa{#1}\edef\si@tempb{si@unt@lastadd\csname si@unt@checkstack\endcsname+#1}\ifx\si@tempa\si@tempb \expandafter\@gobbletwo \else \expandafter\si@unt@stack \fi}

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

\expandafter\ifx\csname si@unt@lastadd\si@unt@checkstack\endcsname\#1\#2\else \expandafter\si@unt@preplussp \fi

\newcommand*{% si@unt@preplussp}[2]{% \def\si@tempa{prefix+space}\edef\si@tempb{si@unt@lastadd\csname si@unt@checkstack\endcsname+#1}\ifx\si@tempa\si@tempb \expandafter\@gobbletwo \else \expandafter\si@unt@stack \fi}

The space added after a prefix needs to be ignored.

\expandafter\if\csname si@unt@checkstack\endcsname\csname si@unt@lastadd\csname si@unt@checkstack\endcsname\#1\#2\else \expandafter\si@unt@preplussp \fi

\newcommand*{% si@unt@preplussp}[2]{% \edef\si@tempa{prefix+space}\edef\si@tempb{si@unt@lastadd\csname si@unt@checkstack\endcsname+#1}\ifx\si@tempa\si@tempb \expandafter\@gobbletwo \else \expandafter\si@unt@stack \fi}

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

\edef\si@tempa{#1}\edef\si@tempb{si@unt@lastadd\csname si@unt@checkstack\endcsname+#1}\ifx\si@tempa\si@tempb \expandafter\@gobbletwo \else \expandafter\si@unt@stack \fi
If a space is added, it is actually held until the next add.

\expandafter\si@unt@inccnt

The appropriate counter is added to.

\newcommand*{\si@unt@inccnt}{%}
\expandafter\advance\csname si@unt@unitcnt\si@unt@checkstack\endcsname\@ne\relax}

Depending on the nature of the addition, it is either held or added to the stack.

\si@unt@holdstacka \si@unt@addstack{\{tokens\}}
\si@unt@holdstackb

The stack contents are actually typeset here. First the spacing between units and values is added.

\newcommand*{\si@unt@stackout}{%}
\expandafter\ifsi@frac
\expandafter\si@unt@fracout
\else
\expandafter\si@unt@normout\fi}

Which stack is in use needs to be tested.

\newcommand*{\si@unt@checkstack}{%}
\expandafter\ifsi@frac
\expandafter\expandafter\expandafter b%
\else
\expandafter\expandafter\expandafter a%
\fi
\texttt{\textbackslash si@unt@spaceout} \quad The space before a unit might not be needed, so it crops up a few times in the output routine.

\begin{verbatim}
1830 \newcommand*{\si@unt@spaceout}{%  
1831 \ensuremath{\si@unt@spstack}}
\end{verbatim}

\texttt{\textbackslash si@unt@prefixout} \quad If the prefix counter is not zero, then there is something to typeset.

\begin{verbatim}
1832 \newcommand*{\si@unt@prefixout}{%  
1833 \ifnum\si@unt@prefixcnt=\z@elax\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}
\end{verbatim}

\texttt{\textbackslash si@unt@normout} \quad The normal output mode is set up here; nothing much needs to be done as there is no need for complex checks.

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

\texttt{\textbackslash si@unt@fracout} \quad For fractions, some checks are needed.

\begin{verbatim}
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 argument\MessageBreak given does not contain any symbols}%  
1850 \fi  
1851 \else  
1852 \ifsi@slash  
1853 \si@unt@prefixout%  
1854 \si@frac{}{\si@unt@stackb}%  
1855 \else  
1856 \si@frac{1}{\si@unt@stackb}%  
1857 \fi  
1858 \fi}
\end{verbatim}

With an empty numerator, no space is added

\begin{verbatim}
1854 \ifsi@slash  
1855 \si@unt@prefixout%  
1856 \si@frac{}{\si@unt@stackb}%  
1857 \else  
1858 \si@frac{1}{\si@unt@stackb}%  
1859 \fi  
1860 \fi}
\end{verbatim}

\texttt{\textbackslash si@unt@fracout} \quad If the denominator is empty, then the usual output system can be used.

\begin{verbatim}
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}
\end{verbatim}
A trap is set for adding brackets to units using a slash, when more than one unit is in the denominator.

\newcommand*{\si@unt@notambig}{\ifnum\si@unt@unitcntb>\@ne\relax\if\si@slash\expandafter\expandafter\expandafter\si@unt@notabg\fi\fi}
\newcommand*{\si@unt@notabg}{\protected@edef\si@unt@stackb{\si@denlbrac\si@unt@stackb}}

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.

To handle non-Latin symbols in the input, a single macro is provided. Initially, it does nothing

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.

The degree symbol is character 176 and the micro symbol is character 181 in latin1.
A macro for declaring symbols: a copy of \DeclareInputMath from inputenc.
\begin{verbatim}
\newcommand*{\si@unt@sym}{⟨charcode⟩}
\newcommand*{\bgroup}{\uccode'\~#1}
\newcommand*{\egroup}{\def~}
\end{verbatim}

\begin{verbatim}
\newcommand*{\kilogram}{kg}
\newcommand*{\metre}{m}
\newcommand*{\mole}{mol}
\newcommand*{\second}{s}
\newcommand*{\ampere}{A}
\newcommand*{\kelvin}{K}
\newcommand*{\candela}{cd}
\end{verbatim}

With the system set up, the basic unit macros are implemented. The only units defined whatever options are given are the base SI units.
\begin{verbatim}
\newcommand*{\Square}{2}
\newcommand*{\squared}{2}
\newcommand*{\cubic}{3}
\newcommand*{\cubed}{3}
\end{verbatim}

A macro for arbitrary powers, which comes after the unit and so needs to be marked as such.
\begin{verbatim}
\newcommand*{\tothe}{post}
\end{verbatim}
\begin{verbatim}
\let\si@loc@sisetup\sisetup
\renewcommand*{\sisetup}{\expandafter\def\csname si@loc@#1\endcsname{##1}}
\let\si@loadfile\si@loc@load
\end{verbatim}

When loading a locale, the setup is saved rather than applied. Anything other than simple settings should be inside \addtolocale, which is already defined.
\begin{verbatim}
\newcommand*{\si@loc@load}{⟨locale⟩}
\let\si@loc@load\si@loc@load
\let\si@loc@sisetup\si@loc@sisetup
\end{verbatim}

16.13 Locales

When loading a locale, the setup is saved rather than applied. Anything other than simple settings should be inside \addtolocale, which is already defined.
\si@loc@set  Setting the locale transfers the settings to \sisetup, and runs any extra macros.
\si@loc@set{(locale)}

1940 \newcommand*{\si@loc@set}[1]{% 
1941 \ifsname si@loc@#1\endcsname 
1942 \si@log@inf{Setting locale to ‘#1’}% 
1943 \expandafter\expandafter\expandafter\expandafter\expandafter% 
1944 \expandafter\expandafter\expandafter\expandafter\expandafter% 
1945 \expandafter\expandafter\expandafter\expandafter\expandafter% 
1946 \expandafter\csname si@loc@#1\endcsname% 
1947 \expandafter\sisetup\expandafter(\the\si@temptoks)% 
1948 \ifsname si@loc@#1extra\endcsname 
1949 \csname si@loc@#1extra\endcsname% 
1950 \fi 
1951 \else 
1952 \ifsname si@loc@#1extra\endcsname 
1953 \si@log@inf{Setting locale to ‘#1’}% 
1954 \csname si@loc@#1extra\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{(locale)}{(commands)}

1959 \newcommand*{\addtolocale}[2]{\si@addtocsname{si@loc@#1extra}{#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{(text)}

1960 \begingroup 
1961 \catcode\'^=\active\relax 
1962 \catcode\'-=\active\relax 
1963 \gdef\si@out@text#1{\begingroup 
1964 \makeatletter 
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 \ifs@fam@set \else 
1968 \expandafter\si@fam@set% 
1969 \fi 
1970 \text(\sf\textstyle\it\bf\texttt 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 pars 
of the package.
1971 \ifs@textmode
\let\textsuperscript
\catcode'\-=\active\relax%
\let-\si@out@minus

The \empty is needed here to mop up any extra space.
\scantokens{#1\empty}%
\else
\let\sp\textsuperscript
\let\sp\textsuperscript
\si@fam@maths{\scantokens{#1}}%
\fi%

Everything is done; a bit of tidying up is needed.
\endgroup
\check@mathfonts}
\endgroup
\si@out@minus An active minus sign is needed.
\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⟩}
\newcommand*{\si@out@num}[1]{%}
\begingroup
\sisetup{%}
\textdefault=\si@textnumdefault,%
\mathdefault=\si@mathnumdefault%}
\si@out@text{#1}%
\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.
To keep the code easy to maintain, the reusable filename components are macros rather than literals.
\newcommand*{\si@extension}{cfg}
\newcommand*{\si@fileprefix}{si-}
\ifloaded A bit of borrowing from the \LaTeX{} kernel.
\newcommand*{\si@ifloaded}[1][]{\@ifl@aded\si@extension{#1}}
\loadfile Loading configuration files is handled here.
\newcommand*{\si@loadfile}[1][]{\@ifl@aded\si@extension{#1}}
\requirecfgs The configuration files depend on each other.
\requirecfgs{\cfg-\file}
\newcommand*{\si@requirecfgs}[1]{%\@for\si@tempb:=#1\do{\si@loadfile{\si@tempb}}} 

For emulation files, an additional check is made.
\newcommand*{\si@loademfile}[1]{\@ifpackageloaded{#1}{\si@loadfile{#1}}}

For emulation clashes.
\newcommand*{\si@emclash}[2]{\si@log@err{Emulation clash: '#1' and '#2'}{You have asked for emulation of package '#1'(perhaps by giving si a back-compatibility option) but the package is already loaded!}}

A check is now made so that emulation takes place one file at a time, and that each file is loaded only once.
\ifx\@empty\si@emulate\@empty\else\@for\si@tempa:=\si@emulate\do{\si@loademfile{\si@tempa}}\fi

For turning the list of default choices into a loadable list.
\newcommand*{\si@expanddefault}[2]{\expandafter\ifx\expandafter\@empty\csname si@#1\endcsname\@empty\else\def\si@tempb{default}\def\si@tempc{}\expandafter\@for\expandafter\si@tempa\expandafter:=\csname si@#1\endcsname\do{\ifx\si@tempa\si@tempb\si@addtolist{\si@tempc}{#2}\else\si@addtolist{\si@tempc}{\si@tempa}\fi}\expandafter\edef\csname si@#1\endcsname{\si@tempc}\expandafter\si@addtolist{\csname si@no#1\endcsname}{default}\def\si@tempc{}\expandafter\@for\expandafter\si@tempa\expandafter:=\csname si@#1\endcsname\do{\si@switchfalse\expandafter\@for\expandafter\si@tempb\expandafter:=\csname si@no#1\endcsname\do{\ifx\si@tempa\si@tempb\si@switchtrue\fi}\fi\si@switchfalse}}\fi\def\si@tempa{\si@tempb}
The configuration and abbreviation files are loaded.
\@for\si@tempa:=\si@tempc\do{\si@loadfile{\si@tempa}}

The locale files are loaded; here there is a need to check on both loadlocales and loctolang.
\ifx\@empty\si@loadlocales\@empty\else\@for\si@tempa:=\si@loadlocales\do{\si@loc@load{\si@tempa}}\fi
For loctolang.
\ifx\@empty\si@loctolang\@empty\else\def\si@tempa#1:#2\@nil{\si@loc@load{#1}}\@for\si@tempb:=\si@loctolang\do{\expandafter\si@tempa\si@tempb::\@nil}\AtBeginDocument{\@ifpackageloaded{babel}{\def\si@tempa#1:#2:#3\@nil{\expandafter\addto\expandafter{\csname extras#2\endcsname}{\si@loc@set{#1}}}\@for\si@tempb:=\si@loctolang\do{\expandafter\si@tempa\si@tempb::\@nil}}{\si@log@warn{babel not loaded - option\MessageBreak loctolang ignored}}}\fi

The very last job is to load a local configuration file, if one exists.
\IfFileExists{si.cfg}{\si@log@inf{Local configuration file found}\InputIfFileExists{si.cfg}{}{}{}}

\section{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).

\subsection{Multiple prefixes}
The various SI multiple prefixes are defined here. First the small powers.
\yocto\ ProvidesFile{si-prefix.cfg}
\zepto\ [2008/02/20 v.06a SI Multiple prefixes]\atto\ \newprefix{\yocto}{-24}{y}
\femto\ \newprefix{\zepto}{-21}{z}\pico\ \newprefix{\atto}{-18}{a}
\nano\ \micro\ \Micro\ \milli\ \centi\ \deci
Some testing is needed for unitsdef compatibility.

\ifsi@old@OHM
\newprefix{\Micro}{-6}{\si@sym@mu}
\else
\ifsi@gensymb\else
\newprefix{\micro}{-6}{\si@sym@mu}
\fi
\fi
\newprefix{\milli}{-3}{m}
\newprefix{\centi}{-2}{c}
\newprefix{\deci}{-1}{d}
\deca  Now the large ones.
\hecto\newprefix{\deca}{1}{da}
\kilo\newprefix{\hecto}{2}{h}
\mega\newprefix{\kilo}{3}{k}
\giga\newprefix{\mega}{6}{M}
\tera\newprefix{\giga}{9}{G}
\peta\newprefix{\tera}{12}{T}
\exa\newprefix{\peta}{15}{P}
\zetta\newprefix{\exa}{18}{E}
\yotta\newprefix{\zetta}{21}{Z}
\deka  Apparently, “deka” is common in the US for deca.
\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.
\newunit{\gram}{g}
\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
\farad\ProvidesFile{si-named.cfg}
\Gray  [2008/02/20 v.06a SI Named units]
\hertz\newunit{\becquerel}{Bq}
\henry\newunit{\coulomb}{C}
\joule\newunit{\farad}{F}
\katal\newunit{\Gray}{Gy}
\lumen\newunit{\hertz}{Hz}
\lux\newunit{\henry}{H}
\newton\newunit{\joule}{J}
\newunit{\katal}{kat}
\newunit{\lumen}{lm}
Some testing is needed for unitsdef compatibility.

To be on the safe side, use provideunit.

The radian and steradian are officially derived units.

17.3 Units with prefixes

As in unitsdef, some commonly used prefixed units are set up. This requires

\ProvidesFile{si-prefix.cfg}
\[2008/02/20 v.06a S1 Prefixed units]
\si@requirecfgs{prefix,named,accepted,physical}

Extra distances.

Extra masses.
\textbf{NEW_UNIT}{\textbackslash picogram}{\textbackslash pico\text{gram}}
\textbf{NEW_UNIT}{\textbackslash nanogram}{\textbackslash nano\text{gram}}
\textbf{NEW_UNIT}{\textbackslash microgram}{\textbackslash micro\text{gram}}
\textbf{NEW_UNIT}{\textbackslash milligram}{\textbackslash milli\text{gram}}
\textbf{femtomole} Now some moles.
\textbf{NEW_UNIT}{\textbackslash femtomole}{\textbackslash femto\text{mole}}
\textbf{NEW_UNIT}{\textbackslash picomole}{\textbackslash pico\text{mole}}
\textbf{NEW_UNIT}{\textbackslash nanomole}{\textbackslash nano\text{mole}}
\textbf{NEW_UNIT}{\textbackslash micromole}{\textbackslash micro\text{mole}}
\textbf{NEW_UNIT}{\textbackslash millimole}{\textbackslash milli\text{mole}}
\textbf{attosecond} Prefixed seconds.
\textbf{NEW_UNIT}{\textbackslash attosecond}{\textbackslash atto\text{second}}
\textbf{NEW_UNIT}{\textbackslash picosecond}{\textbackslash pico\text{second}}
\textbf{NEW_UNIT}{\textbackslash nanosecond}{\textbackslash nano\text{second}}
\textbf{NEW_UNIT}{\textbackslash microsecond}{\textbackslash micro\text{second}}
\textbf{NEW_UNIT}{\textbackslash millisecond}{\textbackslash milli\text{second}}
\textbf{picoampere} The last prefixed base units are amperes.
\textbf{NEW_UNIT}{\textbackslash picoampere}{\textbackslash pico\text{ampere}}
\textbf{NEW_UNIT}{\textbackslash nanoampere}{\textbackslash nano\text{ampere}}
\textbf{NEW_UNIT}{\textbackslash microampere}{\textbackslash micro\text{ampere}}
\textbf{NEW_UNIT}{\textbackslash milliampere}{\textbackslash milli\text{ampere}}
\textbf{millivolt} More electricity-related units.
\textbf{NEW_UNIT}{\textbackslash millivolt}{\textbackslash milli\text{volt}}
\textbf{NEW_UNIT}{\textbackslash kilovolt}{\textbackslash kilo\text{volt}}
\textbf{NEW_UNIT}{\textbackslash megawatt}{\textbackslash mega\text{watt}}
\textbf{NEW_UNIT}{\textbackslash femtofarad}{\textbackslash femto\text{farad}}
\textbf{NEW_UNIT}{\textbackslash picofarad}{\textbackslash pico\text{farad}}
\textbf{NEW_UNIT}{\textbackslash nanofarad}{\textbackslash nano\text{farad}}
\textbf{NEW_UNIT}{\textbackslash microfarad}{\textbackslash micro\text{farad}}
\textbf{NEW_UNIT}{\textbackslash millifarad}{\textbackslash milli\text{farad}}
\textbf{millisiemen} For resistance, checks are needed again for the definition of \ohm.
\textbf{NEW_UNIT}{\textbackslash kiloohm}{\textbackslash kilo\Ohm}
\textbf{NEW_UNIT}{\textbackslash megohm}{\textbackslash mega\Ohm}
\textbf{NEW_UNIT}{\textbackslash gigaohm}{\textbackslash giga\Ohm}
Volumes (unlike unitsdef, with litre and metre spelled correctly). Only 
\millilitre and \microlitre are defined as they are the two officially-
sanctioned prefixes for the litre.
\cubicmetre

Areas, with metre spelled corrected; \are and \hectare are in the “temporarily accepted” file.
\squaremetre

Some energy is needed by now!
\millijoule

Frequencies.
\millihertz

A few more from various areas.
\millinewton

17.4 Abbreviated units
\pA The abbreviated units are sorted in one file. To allow back-compatibility with 
unitsdef, each one is inside an \if block for the appropriate option. First currents.
\nA
\micA
\newunit{\mA}{\milli\ampere}
\newunit{\kA}{\kilo\ampere}

\Hz Then frequencies.
\newunit{\mHz}{\milli\hertz}
\newunit{\kHz}{\kilo\hertz}
\newunit{\MHz}{\mega\hertz}
\newunit{\GHz}{\giga\hertz}
\newunit{\THz}{\tera\hertz}

\newunit{\Hz}{\hertz}
\newunit{\mHz}{\milli\hertz}
\newunit{\kHz}{\kilo\hertz}
\newunit{\MHz}{\mega\hertz}
\newunit{\GHz}{\giga\hertz}
\newunit{\THz}{\tera\hertz}

\fmol Amounts of substance.
\newunit{\pmol}{\pico\mole}
\newunit{\nmol}{\nano\mole}
\newunit{\micmol}{\micro\mole}
\newunit{\mmol}{\milli\mole}

\kV Potentials.
\newunit{\mV}{\milli\volt}
\newunit{\mv}{\milli\volt}

\ml Volumes.
\newunit{\cm}{\centi\litre}
\newunit{\dcm}{\deci\litre}
\newunit{\ml}{\milli\litre}

\kg Masses.
\newunit{\fg}{\femto\gram}
\newunit{\pg}{\pico\gram}
\newunit{\nanog}{\nano\gram}
\newunit{\micg}{\micro\gram}
\newunit{\mg}{\milli\gram}
\newunit{\amu}{\atomicmass}

\kJ Energies.
\newunit{\eV}{\electronvolt}
\newunit{\meV}{\milli\electronvolt}
\newunit{\keV}{\kilo\electronvolt}
\newunit{\MeV}{\mega\electronvolt}
\newunit{\GeV}{\giga\electronvolt}
\newunit{\TeV}{\tera\electronvolt}

\picom Lengths.
\newunit{\nm}{\nano\metre}
\newunit{\mcm}{\micro\metre}
\newunit{\cm}{\centi\metre}
\newunit{\dm}{\deci\metre}
\newunit{\km}{\kilo\metre}

There is a name clash with babel here in French; hopefully there will not be too many complaints.
The letter class (and others) define \ps for postscripts, so \provideunit is best here.

\provideunit{\ps}{\pico\second}
\provideunit{\ns}{\nano\second}
\provideunit{\mics}{\micro\second}
\provideunit{\ms}{\milli\second}

17.5 Additional (temporary) SI units

Some units are “temporarily” acceptable for use in the SI system. These are defined here, although some are in very general use.

\provideFile{si-addn.cfg} [2008/02/20 v.06a SI Additional units]

\provideFile{si-accepted.cfg} [2008/02/20 v.06a SI accepted units]

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.

\provideFile{si-accepted.cfg} [2008/02/20 v.06a SI accepted units]
17.7 Units based on physical measurements

A few units based on physical measurements exist. For \eV, the need for a negative kern does make things a bit complicated.

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

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.

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.
The first units are not specific to high-energy physics, but are not defined elsewhere in SI.

Various prefixed barns

<table>
<thead>
<tr>
<th>Unit</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>nanobarn</td>
<td>(\text{n\text{n}})</td>
</tr>
<tr>
<td>picobarn</td>
<td>(\text{p\text{b}})</td>
</tr>
<tr>
<td>femtobarn</td>
<td>(\text{f\text{b}})</td>
</tr>
<tr>
<td>attobarn</td>
<td>(\text{a\text{b}})</td>
</tr>
<tr>
<td>zeptobarn</td>
<td>(\text{z\text{b}})</td>
</tr>
<tr>
<td>yoctobarn</td>
<td>(\text{y\text{o}})</td>
</tr>
</tbody>
</table>

Inverses barn units.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>invnanobarn</td>
<td>(\text{in\text{n}})</td>
</tr>
<tr>
<td>invpicobarn</td>
<td>(\text{in\text{p}})</td>
</tr>
<tr>
<td>invfemtobarn</td>
<td>(\text{in\text{f}})</td>
</tr>
<tr>
<td>invattobarn</td>
<td>(\text{in\text{a}})</td>
</tr>
<tr>
<td>invzeptobarn</td>
<td>(\text{in\text{z}})</td>
</tr>
<tr>
<td>invyoctobarn</td>
<td>(\text{in\text{y}})</td>
</tr>
</tbody>
</table>

Also available abbreviated.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>invnb</td>
<td>(\text{in\text{n}})</td>
</tr>
<tr>
<td>invpb</td>
<td>(\text{in\text{p}})</td>
</tr>
<tr>
<td>invfb</td>
<td>(\text{in\text{f}})</td>
</tr>
<tr>
<td>invab</td>
<td>(\text{in\text{a}})</td>
</tr>
<tr>
<td>invzb</td>
<td>(\text{in\text{z}})</td>
</tr>
<tr>
<td>invyb</td>
<td>(\text{in\text{y}})</td>
</tr>
</tbody>
</table>

Luminosity.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\text{in\text{c}})</td>
<td>(\text{ms\text{g}}) per second</td>
</tr>
<tr>
<td>(\text{in\text{c}})</td>
<td>(\text{ms\text{g}}) per second</td>
</tr>
<tr>
<td>(\text{lu\text{ni}})</td>
<td>(\text{m\text{m}}) (\text{ms\text{g}}) per second</td>
</tr>
<tr>
<td>(\text{clight})</td>
<td>The speed of light is used in units for the area, although of course it is not strictly a unit.</td>
</tr>
<tr>
<td>(\text{in\text{v\text{e}}})</td>
<td>The inverse of an electron-volt, plus prefixes.</td>
</tr>
<tr>
<td>(\text{e\text{V}over\text{c}})</td>
<td>Some combinations of electron-volts and the speed of light. As these are called over, they are set with a slash. The (\text{e\text{V\text{c}}orrb}) values have been set for Computer Modern.</td>
</tr>
</tbody>
</table>
Prefixed combinations, first of the speed of light.

\keVoverc  \newunit[\per=\text{\textbackslash eVcorr=0.6ex}]\{\keVoverc\}
\MeVoverc  \newunit[\per=\text{\textbackslash eVcorr=0.6ex}]\{\MeVoverc\}
\GeVoverc  \newunit[\per=\text{\textbackslash eVcorr=0.6ex}]\{\GeVoverc\}
\TeVoverc  \newunit[\per=\text{\textbackslash eVcorr=0.6ex}]\{\TeVoverc\}

Then of the square.

\keVovercsq \newunit[\per=\text{\textbackslash eVcorr=0.6ex}]\{\keVovercsq\}
\MeVovercsq \newunit[\per=\text{\textbackslash eVcorr=0.6ex}]\{\MeVovercsq\}
\GeVovercsq \newunit[\per=\text{\textbackslash eVcorr=0.6ex}]\{\GeVovercsq\}
\TeVovercsq \newunit[\per=\text{\textbackslash eVcorr=0.6ex}]\{\TeVovercsq\}

18.3 Binary units

\kibi \newprefix{\kibi}{10}\{\text{Ki}\}
\mebi \newprefix{\mebi}{20}\{\text{Mi}\}
\gibi \newprefix{\gibi}{30}\{\text{Gi}\}
\tebi \newprefix{\tebi}{40}\{\text{Ti}\}
\pebi \newprefix{\pebi}{50}\{\text{Pi}\}
\exbi \newprefix{\exbi}{60}\{\text{Ei}\}

\bit Now the units.
\byte \newunit[\{\text{bit}\}]{\bit}

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.
\ProvidesFile{si-UK.cfg} [2008/02/20 v.06a UK locale]
\sisetup{
  unitsep=thin,
  expproduct=times,
  valuesep=thin,
  decimalsign=fullstop,
  digitsep=thin,
  sepfour=false}

19.2 United States

The same as for the UK.
\ProvidesFile{si-USA.cfg} [2008/02/20 v.06a USA locale]
\sisetup{
  unitsep=thin,
  expproduct=times,
  valuesep=thin,
  decimalsign=fullstop,
  digitsep=thin,
  sepfour=false}

19.3 Germany

Germany, hopefully.
\ProvidesFile{si-germany.cfg} [2008/02/20 v.06a Germany locale]
\sisetup{
  unitsep=cdot,
  valuesep=thin,
  decimalsign=comma,
  expproduct=cdot,
  digitsep=thin,
  sepfour=false}

19.4 South Africa

Taken from sistyle.
\ProvidesFile{si-south-africa.loc} [2008/02/20 v.06a UK Locale]
\sisetup{
  unitsep=cdot,
  valuesep=thin,
  expproduct=times,
  decimalsign=comma,
  digitsep=thin,
  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.

```
\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.

```
\ProvidesFile{si-units.cfg}
\[2008/02/20 v.06a Emulation of units\]
\si@ifloaded{SIunits}{\si@emclash{units}{SIunits}\endinput}{}
\si@ifloaded{sistyle}{\si@emclash{units}{sistyle}\endinput}{}
```

To emulate units, \texttt{\per} must give fractions.

```
\sisetup{per=fraction,fraction=nice,obeybold,inlinebold=maths,}
\texttt{obeymode}\}
\ifsii@old@tight
\sisetup{valuesep=thin}\fi
\ifsii@old@loose
\sisetup{valuesep=space}\fi
\ifsii@old@ugly
\sisetup{fraction=ugly}\fi
```

\texttt{\unit} The units version of \texttt{\unit} is similar to \texttt{\SI}. Here and in \texttt{\unitfrac} the \texttt{\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\[\langle\texttt{num}\rangle\]{\langle\texttt{unit}\rangle}
```

\texttt{\DeclareRobustCommand*{\unit}[2]{{}}\ifmmode\SI{#1}{#2}\else\SI[obeyfamily,obeyitalic]{#1}{#2}\fi}

\texttt{\unitfrac} \texttt{\unitfrac} is a bit more of a hack.

```
\unitfrac\[\langle\texttt{num}\rangle\]{\langle\texttt{numerator}\rangle}{\langle\texttt{denominator}\rangle}
```

\texttt{\DeclareRobustCommand*{\unitfrac}[3]{{}}\begingroup\si@fam@mode\ifmmode\SI{\#1}{\#2}\else\SI[obeyfamily,obeyitalic]{\#1}{\#2}\fi\endgroup}

\texttt{\sisetup{obeyfamily,obeyitalic}}%
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.

Emulation of units is needed for unitsdef to work.

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.

The multitude of package options for unitsdef need to be handled.

The various options for loading unit abbreviations have to be handled. Here, any request to avoid abbreviations prevents any loading.
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.

\newcommand*{\unitvaluesep}{\,}
\renewcommand{\si@valuesep}{\text{\unitvaluesep}}
\si@opt@choicekey{valuesep}{space,thin,med,medium,thick,none}
{\renewcommand*{\unitvaluesep}{\@nameuse{si@fix@##1}}}
{\renewcommand*{\unitvaluesep}{##1}}

Some rather straight-forward definitions, with just a bit of fun to get the spacing correct.
\DeclareRobustCommand{\unitsignonly}{\unitsym}
\DeclareRobustCommand[2]{\arc}{\ang}
\newcommand*{\unittimes}{\ensuremath{\cdot}}

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 \textcsname...\endcsname.
\newcommand*{\unitSIdef}{\upshape}
\newcommand*{\si@unitSIdef}{\unitSIdef\selectfont}
\sisetup{textdefault=si@unitSIdef,textnumdefault=si@unitSIdef}

Rather awkwardly, unitsdef uses \per in a different way to si.
\DeclareRobustCommand[2]{\per}{%}
\begingroup
\si@xspacefalse
\renewcommand{\unitvaluesep}{()}
\unitfrac{#1}{#2}{%}
\endgroup%
\newcommand*{\unitsuperscript}{\ensuremath{\cdot}}

Some pretty straight-forward stuff again; notice that the automatic analyser for units has to be turned off for this to work.
Simple aliases.

Controlling symbols is a simple translation job; as only one setting is used by si in text mode, a bit of extra work is needed.

Controlling symbols is a simple translation job; as only one setting is used by si in text mode, a bit of extra work is needed.

Controlling symbols is a simple translation job; as only one setting is used by si in text mode, a bit of extra work is needed.
The \texttt{ohm} and \texttt{OHM} options are checked, and some sanity is ensured. This needs to happen before loading the configuration files.

For some reason, \texttt{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!)

\begin{verbatim}
\newcommand*{\setTextDegree}[1]{\sisetup{textdegree=#1}}
\newcommand*{\si@log@inf}{\message{Both 'ohm' and 'OHM' options given\MessageBreak Using default behaviour for unitsdef}}
\ifsi@old@ohm\fi
\ifsi@old@OHM\fi
\ifsi@old@liter\fi
\ifsi@old@LITER\fi
\newunit{\meter}{\metre}
\newunit{\liter}{L}
\newunit{\ton}{t}
\newunit{\days}{d}
\newunit{\picometer}{\pico\meter}
\newunit{\nanometer}{\nano\meter}
\newunit{\micrometer}{\micro\meter}
\newunit{\millimeter}{\milli\meter}
\newunit{\centimeter}{\centi\meter}
\newunit{\decimeter}{\deci\meter}
\newunit{\kilometer}{\kilo\meter}
\newunit{\femtoliter}{\femto\liter}
\newunit{\picoliter}{\pico\liter}
\newunit{\nanoliter}{\nano\liter}
\newunit{\microliter}{\micro\liter}
\newunit{\milliliter}{\milli\liter}
\newunit{\centiliter}{\centi\liter}
\newunit{\deciliter}{\deci\liter}
\newunit{\hectoliter}{\hecto\liter}
\newunit{\cubicmeter}{\meter\cubed}
\newunit{\cubicmicrometer}{\micro\meter\cubed}
\newunit{\cubicmillimeter}{\milli\meter\cubed}
\newunit{\squaremeter}{\Square\meter}
\newunit{\squarecentimeter}{\Square\centi\meter}
\newunit{\squarekilometer}{\Square\centi\meter}\ar
\newunit{\ar}{\hectare}
\end{verbatim}
\textit{The code for unitsdef has the capitalisation wrong for kHz and mV.}
\textit{There are some slightly different abbreviations, plus some which are not officially allowed.}
\textit{unitsdef spells calorie incorrectly, and it is also not an SI unit.}
\textit{unitsdef uses \ubar for bar.}
\textit{If the options relating to gensymb are given, then the package has to be loaded. The definitions are then renamed; a slight awkward feature is that the hyphen character needs to be a letter. To avoid needing to worry about this again, a second switch is set up.}
The configuration files can now be loaded.

The noconfig option could be ignored, but it costs little to let it be used.

After setting the necessary defaults, the emulation code defines the macros in sistyle as given in the manual for that package.

Some simple switches, but not using \newif.

To get the correct behaviour for \num, some redefinitions are needed to handle to optional *.
The \pnt macro is needed as it is active inside \SI. The name is exactly the same as in \sisetyle, 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.

\texttt{\SIgroupfourtrue} Switches for grouping four characters.
\texttt{\SIgroupfourfalse} The value given here is passed through to \sisetup.
\texttt{\SIunitsep} Whatever is given here is passed through to \sisetup.
\texttt{\SIdecimalsign} The same is true here, with the appropriate translation.
\texttt{\SIthousandsep} The font definitions need a bit of extra work doing. As both settings here have @ as a letter, all should be fine.
\texttt{\SImathrm} The font control macros have to ensure that a macro name is passed to \sisetup.
\texttt{\SIdefaultMfam} The same for the default keys.
\texttt{\SIdefaultTfam} The same for the \mathdefault and \textdefault keys.
A few extra symbol names are needed.
\degC \newcommand*{\degC}{\si@sym@celsius}
\arcdeg \newcommand*{\arcdeg}{\si@sym@degree}
\degF \newcommand*{\degF}{\si@sym@degree F}

Finally, the locale control.
\SIstyle \newcommand*{\SIstyle}{\sisetup{locale=#1}}
\SIstyleToLang \newcommand*{\SIstyleToLang}{\sisetup{loctolang=#1:#2}}
\si@sis@addtolocale \newcommand*{\AddToSIstyle}{\si@switchfalse\@ifstar{\si@switchtrue\si@sis@addtolocale}}
\newcommand*{\si@sis@addtolocale}{\ifsi@switch\expandafter\let\csname si@loc@#1@extra\endcsname\relax\fi\addtolocale{#1}{#2}}

SIunits emulation starts in much the same way.
\ProvidesFile{si-SIunits.cfg}
\sisetup{unitsep=thick, valuesep=thick, prefixproduct=\si@valuesep}
\si@requirecfgs{prefix,named,accepted,physical}

A few very simple translations, using the internal version of \per to allow changes of output style.
\newcommand*{\reciprocal}{\sisetup{per=reciprocal}\si@per}
\let\rp\reciprocal
\renewcommand*{\per}{\sisetup{per=slash}\si@per}
\newcommand*{\usk}{}
\newcommand*{\power}[1]{#1\tothe}
\newcommand*{\rpsquare}{\sisetup{per=reciprocal}\si@per\Square}
\newcommand*{\rpcubic}{\sisetup{per=reciprocal}\si@per\cubic}
\newpower{\fourth}{4}
\newcommand*{\rpfourth}{\sisetup{per=reciprocal}\si@per\fourth}
\newcommand*{\rpsquared}{\sisetup{per=reciprocal}\si@per\un@per\Square}
\newcommand*{\rpcubed}{\sisetup{per=reciprocal}\si@per\un@per\cubic}

Here, some low-level switch changing is needed.
\newcommand*{\SIsetup}{\sisetup{per=reciprocal}\si@un@per\per\un@per\Square\un@per\cubic\newpower{\fourth}{4}}

The various package spacing options are processed. They also have to be correctly handled by the \SIsetup macro.
\newcommand*{\SIsetup}{\if@undefined{ifs@old@\SIsetup}{\SIsetup}{}}
\square \text{SIunits does slightly different things about the clash with} \square, \text{and either redefines this macro or provides} \squaren.
\squaren
\AtBeginDocument{\@ifundefined{squaren}{\newpower{\squaren}{2}}{\ifsi@old@amssymb\renewpower{\squaren}{2}\else\ifsi@old@squaren\else\si@log@warn{\string\square\space already defined}\MessageBreak SIunits mode may cause errors}\fi\fi}}
\gray \text{The potential clash with PStricks is also handled differently; here, \Gray will already be defined by the si kernel.}
\AtBeginDocument{\@ifundefined{gray}{\newunit{\gray}{Gy}}{\ifsi@old@pstricks\renewunit{\gray}{Gy}\else\ifsi@old@Gray\else\si@log@warn{\string\gray\space already defined}\MessageBreak SIunits mode may cause errors}\fi\fi}}
The \unit macro is defined.

The miscellaneous options are moped up.

\addprefix A little more work for \addprefix.

Slunits 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.
Some more.

\[ \frac{\text{radian per second}}{\text{radian per second}} \]
\[ \frac{\text{kg m}^2 \text{s}^{-3}}{\text{kg m} \text{s}^{-2}} \]
\[ \frac{\text{rad} \text{s}^{-1}}{\text{rad} \text{s}^{-1}} \]
\[ \frac{\text{m}^2 \text{kg}^{-1} \text{s}^{-2}}{\text{m}^2 \text{kg} \text{s}^{-2}} \]
\[ \frac{\text{mol} \text{C} \text{m}^{-2} \text{s}^{-1}}{\text{mol} \text{C} \text{m}^{-2} \text{s}^{-1}} \]
\[ \frac{\text{A} \text{m}^{-2}}{\text{A} \text{m}^2} \]
\[ \frac{\text{C} \text{m}^{-2} \text{s}^{-1}}{\text{C} \text{m}^{-2} \text{s}^{-1}} \]
\[ \frac{\text{W} \text{m}^{-2}}{\text{W} \text{m}^{-2}} \]
\[ \frac{\text{J} \text{m}^{-2}}{\text{J} \text{m}^{-2}} \]
\[ \frac{\text{N} \text{m}^{-3}}{\text{N} \text{m}^{-3}} \]
\[ \frac{\text{J} \text{K}^{-1}}{\text{J} \text{K}^{-1}} \]
\[ \frac{\text{W} \text{K}^{-1}}{\text{W} \text{K}^{-1}} \]
\[ \frac{\text{J} \text{mol}^{-1}}{\text{J} \text{mol}^{-1}} \]
\[ \frac{\text{C} \text{m}^{-2} \text{s}^{-1}}{\text{C} \text{m}^{-2} \text{s}^{-1}} \]
\[ \frac{\text{F} \text{m}^{-1}}{\text{F} \text{m}^{-1}} \]
\[ \frac{\text{ohm m}}{\text{ohm m}} \]
\[ \frac{\text{W} \text{K}^{-1}}{\text{W} \text{K}^{-1}} \]
\text{Some more.}
Some more.

\newtonpermetrenp  \wattpermetrekelvinnp
\wattpermetrekelvin
\newtonmetre
\newtonmetrenp
\squaremetrepercubicsecondnp
\squaremetrepercubicsecond
\metrepersecondnp
\metrepersecond
\joulepercubicmetrenp
\joulepercubicmetre
\squaremetreperkilogram
\squaremetreperkilogramn
\meterpersecondn
\meterpersecond
\joulepermetrekelvin
\joulepermetrekelvin
\newtonpermetrenp
\newtonpermetre
\kilogrampercubicmetre
\kilogrampercubicmetrenp
\cubicmetrepersecondnp
\cubicmetrepersecond
\kilogrampersecondcubicmetre
\kilogrampersecondcubicmetrenp

The prefixes giving numerical output need a trick. First the small values.
\yoctod  \zeptod  \attod  \femtod  \picod  \nanod  \microd  \millid  \centid
\decad  \dekad  \hectod  \kiloc  \megad  \gigad  \terad  \petad  \exad  \zettad  \yottad

The the larger ones.
\newunit{\decad}{\si@prefixnumtrue\deca}
\newunit{\dekad}{\si@prefixnumtrue\deka}
\newunit{\hectod}{\si@prefixnumtrue\hecto}
\newunit{\kilod}{\si@prefixnumtrue\kilo}
\newunit{\megad}{\si@prefixnumtrue\mega}
\newunit{\gigad}{\si@prefixnumtrue\giga}
\newunit{\terad}{\si@prefixnumtrue\tera}
\newunit{\petad}{\si@prefixnumtrue\peta}
\newunit{\exad}{\si@prefixnumtrue\exa}
\newunit{\zettad}{\si@prefixnumtrue\zetta}
\newunit{\yottad}{\si@prefixnumtrue\yotta}

The binary versions need a little more work.
\newunit{\kibid}{\si@prefixnumtrue\let\si@prefixpower\si@fix@two\kibi}
\newunit{\mebid}{\si@prefixnumtrue\let\si@prefixpower\si@fix@two\mebi}
\newunit{\gibid}{\si@prefixnumtrue\let\si@prefixpower\si@fix@two\gibi}
\newunit{\tebid}{\si@prefixnumtrue\let\si@prefixpower\si@fix@two\tebi}
\newunit{\pebid}{\si@prefixnumtrue\let\si@prefixpower\si@fix@two\pebi}
\newunit{\exbid}{\si@prefixnumtrue\let\si@prefixpower\si@fix@two\exbi}

The derived units may need to be defined.
\newunit{\derradian}{\metre\reciprocal\metre}
\newunit{\dersteradian}{\squaremetre\rpsquare\metre}
\newunit{\derhertz}{\reciprocal\second}
\newunit{\dernewton}{\metre\kilogram\second\rpsquared}
\newunit{\derpascal}{\newton\rpsquare\metre}
\newunit{\derjoule}{\newton\metre}
\newunit{\derwatt}{\joule\reciprocal\second}
\newunit{\dercoulomb}{\ampere\second}
\newunit{\derohm}{\volt\reciprocal\ampere}
\newunit{\dersiemens}{\ampere\reciprocal\volt}
\newunit{\derweber}{\squaremetre\kilogram\second\rpsquared\reciprocal\ampere}
\newunit{\derhertz}{\reciprocal\second}
\newunit{\dercelsius}{\kelvin}
\newunit{\derlumen}{\candela\steradian}
\newunit{\derlux}{\lumen\rpsquare\metre}
\newunit{\derbecquerel}{\derhertz}
\newunit{\dersievert}{\dergray}
\newunit{\derkatal}{\rp\second\usk\mole}
\newunit{\f1}{\f1}
Also the “derived-in-base”.

\ifsi@old@derivedinbase
\newunit{\radian}{\metre\reciprocal\metre}
\newunit{\steradian}{\square\metre\radian\square\metre}
\newunit{\hertz}{\reciprocal\second}
\newunit{\newton}{\metre\kilogram\second\radian\squared}
\newunit{\pascal}{\reciprocal\metre\kilogram\second\radian\squared}
\newunit{\joule}{\square\metre\kilogram\second\radian\squared}
\newunit{\watt}{\square\metre\kilogram\cubic\second\radian\squared}
\newunit{\coulomb}{\ampere\second}
\newunit{\volt}{\square\metre\kilogram\cubic\second\radian\squared\reciprocal\ampere}
\newunit{\farad}{\radian\squared\metre\reciprocal\kilogram\fourth\second\ampere\squared}
\newunit{\ohm}{\square\metre\kilogram\cubic\second\radian\squared\ampere}
\fi

Also in two blocks.
\newunit{\siemens}{\radian\squared\metre\reciprocal\kilogram\cubic\second\ampere\squared}
\newunit{\weber}{\square\metre\kilogram\second\radian\squared\reciprocal\ampere}
\newunit{\tesla}{\kilogram\second\radian\squared\reciprocal\ampere}
\newunit{\henry}{\square\metre\kilogram\second\radian\squared\ampere}
\newunit{\celsius}{\kelvin}
\newunit{\lumen}{\candela\square\metre\radian\squared\metre}
\newunit{\lux}{\candela\square\metre\radian\fourth\metre}
\newunit{\becquerel}{\hertz}
\newunit{\gray}{\square\metre\second\radian\squared}
\newunit{\sievert}{\gray}
\newunit{\katal}{\radian\second\mole}
\fi

Any configuration file is used if found.
\InputIfFileExists{SIunits.cfg}
\ifsi@log\ifsi@log\ifsi@log\ifsi@log\ifsi@log\fi\fi\fi\fi\fi
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  
\~  809, 1882, 1888, 1907
A
addprefix  2895  
addlocale  1959, 2795  
AddToSiStyle  2783  
addunit  2894, 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, 3032
\ampere  12, 1911, 2216–2220,  
2275–2279, 2918,  
2919, 2929, 2963,  
2964, 3074, 3075,  
3077, 3078, 3080,  
3082, 3099, 3101,  
3103, 3106, 3108,  
3110, 3111, 3113
\amperemetres  2920  
\amperemetre  2956  
\amperemetremp  2956  
\amperesquare  2908  
\amperesquaremp  2908
\celsius  2181, 2711,  
2715, 2720, 2893  
\celsiusbase  3107  
\centi  12, 2123, 2197,  
2202, 2205, 2252,  
2295, 2315, 2397– 
2399, 2555, 2663,  
2670, 2694, 3042
\centid  3034  
\centiliter  2658  
\centimeter  2651
\centimetrecubed  2783
\cl  2851
\clight  2400,  
2408, 2410, 2412,  
2414, 2416, 2418,  
2420, 2422, 2424,  
2426, 2428, 2430
\cm  14, 2111
\cmc  15, 15, 2293
\coulomb  13, 2154, 2916, 2917,  
2927, 2928, 2932,  
2933, 2950, 2951,  
3025, 3027, 3076
\coulombbase  3091
\coulombperm2  2290
\coulombperm2mp  2292
\coulombperm2mp2  2292
\coulombkilogram  2944
\coulombkilogrammp  2944
\coulombkilogrammp2  2944
\coulombkilogrammp2mp  2944
\coulombper  12, 2111, 2961, 2962,  
3084, 3115, 3116
\candelapersquare  2956  
\candelapersquaremp  2956
\candelapersquaremp2  2956
\candelapersquaremp2mp  2933
\celsius  2181, 2666–2668, 2814
<table>
<thead>
<tr>
<th>Symbol</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>molar</td>
<td>16, 16, 2365</td>
</tr>
<tr>
<td>mole</td>
<td>12, 1911, 2205-2209, 2236-2290, 2369, 2900-2903, 2916, 2917, 2979, 2983, 3089, 3120</td>
</tr>
<tr>
<td>newtonbase</td>
<td>3011</td>
</tr>
<tr>
<td>newtonmetre</td>
<td>3011</td>
</tr>
<tr>
<td>newtonpercubicmetre</td>
<td>2933</td>
</tr>
<tr>
<td>newtonperkilogram</td>
<td>2933</td>
</tr>
<tr>
<td>molepercubicmetre</td>
<td>2896</td>
</tr>
<tr>
<td>moleperkilogram</td>
<td>2896</td>
</tr>
<tr>
<td>mrad</td>
<td>3375</td>
</tr>
<tr>
<td>ns</td>
<td>15, 2318</td>
</tr>
<tr>
<td>nV</td>
<td>15, 2291</td>
</tr>
<tr>
<td>mv</td>
<td>2674</td>
</tr>
<tr>
<td>N</td>
<td></td>
</tr>
<tr>
<td>nA</td>
<td>15, 2272</td>
</tr>
<tr>
<td>nano</td>
<td>12, 2123, 2194, 2202, 2207, 2213, 2217, 2222, 2228, 2276, 2288, 2300, 2312, 2322, 2378, 2385, 2391, 2652, 2660, 2691, 3039</td>
</tr>
<tr>
<td>nanometer</td>
<td>2651</td>
</tr>
<tr>
<td>nanometerre</td>
<td>14, 2193</td>
</tr>
<tr>
<td>nanomole</td>
<td>15, 2205</td>
</tr>
<tr>
<td>nanometer</td>
<td>14, 2193</td>
</tr>
<tr>
<td>nanosquareem</td>
<td>2211</td>
</tr>
<tr>
<td>nanosquarcmetre</td>
<td>2211</td>
</tr>
<tr>
<td>NC@rewrites</td>
<td>1220</td>
</tr>
<tr>
<td>neper</td>
<td>13, 2318</td>
</tr>
<tr>
<td>newnosepunit</td>
<td>2586</td>
</tr>
<tr>
<td>newpower</td>
<td>17, 1452, 1918-1921, 2810, 2845, 2849</td>
</tr>
<tr>
<td>newprefix</td>
<td>17, 1439, 2125-2130, 2132, 2135, 2138-2151, 2433-2438, 2721, 2895</td>
</tr>
<tr>
<td>newton</td>
<td>13, 2154, 2267, 2268, 2923, 2925, 2942-2945, 2990, 2991, 3010, 3011, 3015, 3071, 3072</td>
</tr>
<tr>
<td>newtonbase</td>
<td>3091</td>
</tr>
<tr>
<td>nm</td>
<td>14, 2311</td>
</tr>
<tr>
<td>newtonmetre</td>
<td>3011</td>
</tr>
<tr>
<td>newtonpercubicmetre</td>
<td>2933</td>
</tr>
<tr>
<td>newtonperkilogram</td>
<td>2933</td>
</tr>
<tr>
<td>molepercubicmetre</td>
<td>2896</td>
</tr>
<tr>
<td>moleperkilogram</td>
<td>2896</td>
</tr>
<tr>
<td>mrad</td>
<td>3375</td>
</tr>
<tr>
<td>ns</td>
<td>15, 2318</td>
</tr>
<tr>
<td>nV</td>
<td>15, 2291</td>
</tr>
<tr>
<td>mv</td>
<td>2674</td>
</tr>
<tr>
<td>newnosepunit</td>
<td>2586</td>
</tr>
<tr>
<td>newpower</td>
<td>17, 1452, 1918-1921, 2810, 2845, 2849</td>
</tr>
<tr>
<td>newprefix</td>
<td>17, 1439, 2125-2130, 2132, 2135, 2138-2151, 2433-2438, 2721, 2895</td>
</tr>
<tr>
<td>newton</td>
<td>13, 2154, 2267, 2268, 2923, 2925, 2942-2945, 2990, 2991, 3010, 3011, 3015, 3071, 3072</td>
</tr>
</tbody>
</table>

120
23 References