siunitx: A LaTeX Swiss army knife for units

Joseph Wright*

August 2, 2009

Abstract

The siunitx package is a complete system for typesetting units in LaTeX. It provides a large number of settings, allowing the user complete control of the output for a fixed input. The package grew out of SIunits and SIstyle, but covers many additional areas. This article provides a brief overview of why the new package was written and some of its key features.

1 Background

In November 2007, a seemingly simple query about a bug in the SIunits package Heldorn [2007] was posted to `comp.text.tex`. I suggested a bug fix, but on contacting the author of SIunits found he had no time for further maintenance. I therefore found myself somewhat accidentally as the new package maintainer. After fixing the bug at hand, I decided to have a good look over the code, and to ask for suggestions for improvements. It soon became very clear that unit support in LaTeX needed serious attention, and that tweaks to SIunits would not be enough. At that point, siunitx was born.

2 Units in print

The need for packages to control printing units is not immediately obvious. However, the existence of SIunits, SIstyle Els [2006], fancyunits Bauke [2007], unitsdef Happel [2005] and units Reichert [1998] attests to the desire to go beyond direct input of units (and values).

Units are important, and without clear rules problems almost automatically arise. There are therefore internationally agreed units: the *Système International d'Unités* Conférence Générale des Poids et Mesures [2008], usually referred to as SI units. To go with the agreed units are a set of rules on how to print both units and the associated values; NIST have a good set of guidelines for authors National Institute of Standards and Technology [2008]. Doing this properly and consistently is much better achieved using pre-build macros, rather than checking every single use by hand.

*joseph.wright@morningstar2.co.uk

At the same time, different publishers have their own rules. These do not always follow the 'official' rules, and authors do not want to change their source for every different use. This again makes the strong case for using adjustable macros for writing units; chancing the style is then only a question of altering the definition.

Finally, values as well as units have rules, and these rules vary depending on publisher and country. The numprint package Harders [2008] provides a range of tools to automatically format numbers, to match the desired output style. It also includes basic abilities to include units with these numbers.

3 Enter siunitx

Given the existing range of packages, the question arises 'why another package?'. The answer is that, although there are several other packages, none covers everything. For example, SIunits provides macros such as `\metre` to maintain consistency, but is poor on control of numbers, whereas Systyle is stronger on numbers but does not provide unit macros. So a combined package, which can do everything, seemed desirable. Many of the ideas that I and others had were also well beyond anything available in the existing solutions. That said, a lot of the internals of siunitx is taken more-or-less verbatim from the existing packages. The numprint system has been extensively recycled here; most of the number-interpreting system is based on the numprint method. Indeed, while the initial aim of the new package was to deal primarily with *units*, much of the work I have undertaken has been concerned with *numbers*.

The design of siunitx is intended to be flexible. Almost everything is made available as an option, and given the number of options, key–value controls were the only way to do this. The package documentation covers (hopefully) all of them, but a few examples here will illustrate the ideas.

3.1 Some basics

The basic macro of the package is `\SI`. This takes a unit and a value, and typesets them together. The font used for this is controlled, and the space between them cannot be broken. Thus for example `\SI{10}{\metre}` gives '10 m'. The second argument is a unit, which can be given as a series of macros, or can be literal text: `\SI{2}{N.m^2}` gives '2 N m^2'. Here, there is not need to instruct the package to use maths mode for superscripts: this is handled automatically.

Both the number and unit parts of the input can be used independently, with macros `\num` and `\si`, respectively. The `\num` macro can interpret and format a wide range of numerical input. So `\num{1e2,3}` gives '$1 \times 10^{2.3}$' (when using U.K. settings, at least). As with the rest of the package, the way that numbers are interpreted can be adjusted by setting the appropriate package options. For angles, the `\ang` macro is available; it can work with angles as decimal and arcs: `\ang{1.23}` and `\ang{1;2;3}` give '1.23°' and '1°2′3″', respectively.

Package options can be set in three ways. As with most packages, load-time options are recognised. There is also a `\sisetup` macro, which adjusts settings for all subsequent input. Finally, all of the user macros accept an optional first argument; this allows changes for a specific item only.

3.2 The unit processor

One of the new ideas in siunitx is the unit processor. This takes macro-based units, and can reformat them depending on the desired output. 'Out of the box', `\SI{30}{\metre\per\second}` gives '$30\,\mathrm{m\,s^{-1}}$', whereas if we set the option `per=slash`, the same input gives '30 m/s'. This can be extended further, allowing the interpretation of powers, including reciprocals. Thus, given the input `\SI{20}{\per\Square\second}`, the result can be '$20\,\mathrm{s^{-2}}$', '$20/\mathrm{s^2}$', '$20\,\frac{1}{\mathrm{s^2}}$' or '$20\,{}^1\!/_{\mathrm{s}^2}$', depending on the options set.

3.3 Special effects

A lot of work has gone into making more complex ideas available for users of siunitx. Thus the idea of 'repeated' units is easily handled:

```
\SI{1 x 2 x 3}{\metre}
```

yields '$1\,\mathrm{m} \times 2\,\mathrm{m} \times 3\,\mathrm{m}$', for example. Setting option `repeatunits=false`, the alternative (and not entirely desirable) '$1 \times 2 \times 3\,\mathrm{m}$' results.

Similarly, the conversion of angles between decimal and degree–minute–second format is possible. So `\ang{1.23}` normally gives the expected '1.23°'; setting `angformat=arc` gives the alternative form '$1^\circ 13' 48.0''$'.

In many parts of the natural sciences, errors in physical measurements are important. Two ways are common for showing these, as a separate error part, and as a bracketed error in the last digit: 1.23 ± 0.04 and 1.24(4), respectively. Using siunitx, both forms can result from the same input: `\num{1.24(4)}`. The separated form is obtained by setting the `seperr` flag. This works with exponents and units, for example:

```
\SI[seperr]{1.23(4)e5}{\candela}
```

gives '$(1.23 \pm 0.04) \times 10^5\,\mathrm{cd}$'. Notice the automatic addition of brackets to prevent ambiguity: this is an adjustable package option.

The final extra to highlight is the provision of a new column for tabular environments, the `S` column. This brings the ability to use the `\num` macro to tables, but also brings control of alignment (including exponents). A short example shows the effect:

```
\begin{table}
  \centering
  \begin{tabular}{cS[tabformat=3.1]}
    \toprule
    {Entry} & {Distance/\si{\metre}} \\
```

Entry	Distance/m
1	102.3
2	123.2
3	2.3
4	145
5	2.3

```
    \midrule
    1 & 102.3 \\
    2 & 123.2 \\
    3 &   2,3 \\
    4 & 145.  \\
    5 &   2.3 \\
    \bottomrule
  \end{tabular}
\end{table}
```

Here, the contents of the `S` column are centred under the column heading. Space is reserved for three digits before the decimal point, and one digit after. Thus, while the decimal points are aligned they are *not* at the centre of the column. A range of options are provided to allow control of positioning in `S` columns.

3.4 Emulation

Replacing the existing packages raises the issue of users moving existing manuscripts to siunitx. To aid this transition, emulation modes are available for all of the existing solutions. This means that users of the existing packages can experiment with siunitx without needing to learn new default settings or input conventions.

4 Conclusions

This short article can only provide a flavour of the abilities of siunitx. However, it hopefully highlights the possibilities made available by the new package. The package documentation includes a full range of examples, with almost all of the package options illustrated. I hope that I have covered a wide range of needs, and that siunitx will make it easier for LaTeX users to get units right without working too hard. Suggestions for new features are always welcome, particularly when they come with some interface ideas as well!

References

Marcel Heldorn. The siunits package. Available from CTAN, `macros/latex/contrib/siunits`, 2007.

D. N. J. Els. The sistyle package. Available from CTAN, macros/latex/contrib/sistyle, 2006.

Heiko Bauke. fancyunits, 2007. http://www.mpi-hd.mpg.de/personalhomes/bauke/LaTeX/Tips_und_Tricks/fancyunits/index.php.

Patrick Happel. unitsdef—Typesetting units with LaTeX 2$_\varepsilon$. Available from CTAN, macros/latex/contrib/unitsdef, 2005.

Alex Reichert. units.sty—nicefrac.sty. Available from CTAN, macros/latex/contrib/units, 1998.

Conférence Générale des Poids et Mesures. The International System of Units (SI), 2008. http://www.bipm.org/en/si/.

National Institute of Standards and Technology. International System of Units from NIST, 2008. http://physics.nist.gov/cuu/Units/index.html.

Harald Harders. The numprint package. Available from CTAN, macros/latex/contrib/numprint, 2008.