siunitx: Past, present and future

Joseph Wright

Abstract

The siunitx package provides a powerful toolkit for typesetting numbers and units in LaTeX. By incorporating detail about the agreed rules for presenting scientific data, siunitx enables authors to concentrate on the meaning of their input and leave the package to deal with the formatting. Here, I look at the background to the package, what led me from version 1 to version 2, and why version 3 is now under development.

1 Introduction

Typesetting units naturally lends itself, in TeX, to using macro support. The formal rules for SI units [1] link unit names with unit symbols, and it's not surprising that several authors have created packages that tackle some or all of the subtleties involved. I've detailed some of these issues and packages before, when I last looked at siunitx for *TUGboat* [6]. Here, I'll briefly recap some of those key points, then explore what is driving efforts toward a third version of the package.

2 Early days

Before siunitx, there were a variety of units-related packages, including one called SIunits [3], which deals with providing semantic macros for units. My involvement started when I followed up an apparently-innocuous post to `comp.text.tex` from Stefan Pinnow in November 2007, reporting a straightforward bug

```
I want to report that \reciprocal,
\rpsquare, \rpcubic, etc. output is
written as "-1" instead of a $-1$, when
the package option "textstyle" is used.
I tried to contact Mr. Heldoorn, but he
didn't answer until now. Does anyone
have an idea what to do?
```

Being young(ish) and foolish, after looking at the bug itself and finding that SIunits was no longer maintained, I volunteered to pick up the package. Even more foolishly, I then followed up with the following in November 2007:

```
As some of you may have noticed,
following a recent bug report
concerning the SIunits package,
I have taken over as the package
maintainer. I have uploaded a bug fix
for the specific issue to CTAN, and so
hopefully it will appear within a day
or two.
   It has been suggested by the
maintainer of the SIstyle package
that integration of the two be would
worth considering. Other suggestions
have also been made in the newsgroup
and by private mail. I am therefore
planning to review the existing
situation and see what improvements
are needed/desirable. As well as
SIunits and SIstyle, I am going to
look at numprint, units, unitsdef and
hepunits for inspiration/points to
consider/etc.
   So far, I have some outline ideas,
for example: ...
```

I soon had quite a list, and work on the new package began in earnest, and by February 2008 the first testing release was out. After a little more work, and a rename, the first official version of siunitx hit CTAN on the 16th of April 2008 [7].

3 Key features

The core features of siunitx have been present from that first version, and are pretty well-known. I'd summarise them as

- Automatic, semantic formatting of quantities (numbers with units)
- Parsing and manipulation of numbers
- Control of printing of numbers, units and quantities
- Alignment of numbers in tables
- Unified key–value interface for controlling options

For me, the package has always been about *units*, and the fundamental idea that input such as

`\joule\per\mole\per\kelvin`

can give $\mathrm{J\,mol^{-1}\,K^{-1}}$ or $\frac{\mathrm{J}}{\mathrm{mol\,K}}$ or $\mathrm{J/(mol\,K)}$, depending on the settings in force. This idea has been there since day one, and the code has carried through more-or-less unchanged.

4 From version one to version two

Version one of siunitx worked well for delivering on those key features. Releases progressed rapidly, culminating in version 1.4c in February 2010. However, adding new features was a problem: internally it was a bit of a mess. For example, if you look at the old code you'll see:

- Internals *other than unit parsing* taken from existing packages and somewhat haphazard

- Sub-optimal key–value choices
- Essentially no internal API
- Poor self-coded loops
- ...

Around this time, Will Robertson contacted me to ask what I thought of the LaTeX3 language expl3 [4]. This was before I joined the LaTeX team, and expl3 looked a bit different than today, but the central ideas were all there. I liked the ideas, but at the time was a bit wary of loading an external library (and thus a dependency). So I started by picking out the ideas and re-coding them in my development setup for version two. It soon became clear that I needed a *lot* of the ideas, and I realised that I'd be much better off just requiring expl3.

Work on the second version of siunitx took me into expl3 programming, and asking the team for lots of features. In particular, I wanted to have key–value support built-in, rather than needing to use another package to do that. So I wrote some code, which I called keys3, to solve the problem. It turned out that was my application to join the team: it's today l3keys in expl3 itself!

The rewrite gave me a chance to significantly revise internal API aspects, and to significantly improve performance. It also came of course with new features for users, and completely new names for the key–value interface. In the v2.0 release, I didn't include backward-compatibility: I soon learned, and that's been there since a few days after the second generation release.

5 From version two to version three

Version two retains most of the features that version one had, but as well as the good ones, it turns out it keeps some of the bad ones too! In particular

- Assumptions about fonts: OpenType, *etc.*
- No code-level API expl3
- Internals still too messy
- Testing the PDF documentation only
- Monolithic source
- Still too slow

5.1 Font control

The font assumptions carry all the way through from SIstyle [2], and which I adjusted only slightly. The approach currently used is

1. Detect current font type using LaTeX internal data
2. Insert everything inside `\text`
3. Apply `\ensuremath` inside the box
4. Perhaps apply `\text` again (for text mode output)
5. Force the font with *e.g.* `\mathrm` or `\rmfamily`

That requires a lot of work, and more importantly is unreliable: it's not always easy to get the right font 'inside' the output section. It also fails very badly with OpenType math mode fonts, where the ideas in classical TeX about math families simply don't apply. So there is a new approach: for version three

1. Detect current font type using LaTeX internal data
2. Set any aspects *that are needed*
3. Only use an `\mbox` if math version has to be altered

This 'minimal change' approach is much faster than the current one, and is much better at respecting font changes. I'm still finalising compatibility for current edge-case setups, but I believe the new code is much preferable overall.

5.2 The code API and testing

The development of siunitx version two very much parallels that of expl3 as a truly usable language: in the time I've been using it, expl3 has gone from a set of clever experiments to a well-established approach to coding in TeX. But keeping all of siunitx up-to-date with ideas has been tricky.

The biggest issue is that when I wrote the current release code, there were just user commands such as `\num` and internal implementation. However, it's now clear that each user command should have a *documented* code-level interface. Moreover, these interfaces should all be properly tested: the team have created l3build precisely for that [5]. Combining these ideas, the new code will take the input

```
\siunitx_unit_format:nN
  { \joule \per \mole }
  \l_tmpa_tl
\tl_show:N \l_tmpa_tl
```

and create as output

```
> \l_tmpa_tl=
  \mathrm {J}\,\mathrm {mol}^{-1}
```

Notice that this is easy to test, and highlights another new idea: the parsing step should produce the same results as a user typing in the formatting 'by hand'.

5.3 First alpha release

At the time of writing, the development of version three has reached the first alpha stage: the code is usable but there are real gaps. Currently, all of the following are working:

- Core functionality:
  - Unit parsing and formatting
  - Real number formatting
  - Tabular columns
- Existing API: `\num`, `\SI`, `\si`, `S`-column
- New (experimental) document API: `\unit`, `\qty`

There are some big areas left to do, such as multi-part numbers, ranges, lists and most importantly the compatibility layer for dealing with existing documents. However, that is all relatively manageable, and I expect to be done around the end of the year. So 2019 should see siunitx reach version 3.0.0, and I hope retain its place as *the* units package for LaTeX.

References

[1] Bureau International des Poids et Mesures. The international system of units (SI), 2010. bipm.org/en/measurement-units/

[2] D. Els. The SIstyle package, 2008. ctan.org/pkg/sistyle

[3] M. Heldoorn and J. Wright. The SIunits package: Consistent application of SI units, 2007. ctan.org/pkg/siunits

[4] LaTeX Project. The expl3 package and LaTeX3 programming, 2018. ctan.org/pkg/expl3

[5] LaTeX Project. l3build: Checking and building packages, 2018. ctan.org/pkg/l3build

[6] J. Wright. siunitx: A comprehensive (SI) units package. *TUGboat* 32(1):95–98, 2011. tug.org/TUGboat/tb32-1/tb100wright-siunitx.pdf

[7] J. Wright. siunitx — a comprehensive (SI) units package, 2018. ctan.org/pkg/siunitx

◇ Joseph Wright
Morning Star
2, Dowthorpe End
Earls Barton
Northampton NN6 0NH
United Kingdom
joseph dot wright (at)
morningstar2.co.uk

Appendix: demos

Simple number formatting:

123	`\num{123} \\`
1234	`\num{1234} \\`
12 345	`\num{12345} \\`
0.123	`\num{0.123} \\`
0.1234	`\num{0,1234} \\`
0.123 45	`\num{.12345} \\`
3.45×10^{-4}	`\num{3.45d-4} \\`
-10^{10}	`\num{-e10}`

Angles:

10°	`\ang{10} \\`
12.3°	`\ang{12.3} \\`
4.5°	`\ang{4,5} \\`
$1°2'3''$	`\ang{1;2;3} \\`
$1''$	`\ang{;;1} \\`
10°	`\ang{+10;;} \\`
$-0°1'$	`\ang{-0;1;}`

Units as macros:

$$\mathrm{kg\,m\,s^{-2}}$$
$$\mathrm{g\,cm^{-3}}$$
$$\mathrm{V^{2}\,lm^{3}\,F^{-1}}$$
$$\mathrm{m^{2}\,Gy^{-1}\,lx^{3}}$$
$$\mathrm{H\,s}$$

```
\si{\kilo\gram\metre\per\square\second} \\
\si{\gram\per\cubic\centi\metre}        \\
\si{\square\volt\cubic\lumen\per\farad} \\
\si{\metre\squared\per\gray\cubic\lux}  \\
\si{\henry\second}
```

Quantities:

$$1.23\,\mathrm{J\,mol^{-1}\,K^{-1}}$$
$$0.23 \times 10^{7}\,\mathrm{cd}$$
$$1.99/\mathrm{kg}$$
$$1.345\,\frac{\mathrm{C}}{\mathrm{mol}}$$

```
\SI[mode = text]{1.23}{J.mol^{-1}.K^{-1}} \\
\SI{.23e7}{\candela} \\
\SI[per-mode = symbol]
  {1.99}{\per\kilogram} \\
\SI[per-mode = fraction]
  {1,345}{\coulomb\per\mole}
```