Non-Decimal Units for LaTeX

Mikkel Eide Eriksen
mikkel.eriksen@gmail.com

October 5, 2023

1 Preface

Many historical unit systems were non-decimal. For example, the Danish rigsdaler[1] — where 1 rigsdaler consists of 16 mark, each again consisting of 16 skilling for a total of 96 skilling per rigsdaler — was used from 1625 to 1875, when currency was decimalised to the current system of 1 krone = 100 øre.

Units for such measures as length, area, weight, and so on were also often non-decimal, and in fact remain so in the few places of the world that have not made the change to the metric system.

The non-decimal numbers were chosen due to their larger number of division factors, which simplified mental arithmetic — eg. when sharing an amount of money or dividing goods.

This package enables creation and configuration of such units to facilitate their presentation in textual and tabular contexts, as well as simple arithmetic.

In order to do this, values are divided into *segments*, which are separated by decimal points: for example, the historical Danish monetary value 1 Rdl. 2 ß 3 ß is entered as `1.2.3`, which the code then formats appropriately.

Issues can be reported at `https://github.com/mikkelee/latex-units/issues` but keep in mind I am not very experienced with LaTeX ;)

[1] `https://en.wikipedia.org/wiki/Danish_rigsdaler`

2 Configuration

The package is configured in the following manner:

`\usepackage[`⟨*options*⟩`]{non-decimal-units}`

Where ⟨*options*⟩ may contain one or more of the following unit systems. See **??** for details.

`british`	Currencies
`danish`	Currencies, areas, and weights
`german`	Currencies

Alternately, one may configure new units via **??**[→ P. ??] and **??**[→ P. ??].

`\nduKeys{`⟨*options*⟩`}`

Can be used to set options globally (in the preamble) or locally (in a group). See further documentation for possible keys/values.

3 Usage

3.1 Formatting Values

The central macro is `\nduValue`. It formats values for display and is configurable in a number of ways.

`\nduValue{`⟨*unit group*⟩`}[`⟨*options*⟩`]{`⟨*value*⟩`}`

Formats ⟨*value*⟩ according to the setup configured for the ⟨*unit group*⟩, as well as any provided ⟨*options*⟩.
If no special configuration is made, the number of decimal points and the values between them determine how many and which units are displayed. For example, empty values are skipped unless the **??** → P. ?? key is set.

Example usage: `\nduValue` macro

```
\nduValue{danish rigsdaler}{1.2.3}\\
\nduValue{danish rigsdaler}{1}\\
\nduValue{danish rigsdaler}{.2}\\
\nduValue{danish rigsdaler}{..3}\\
```

1 Rdl. 2 ß 3 ß
1 Rdl.
2 ß
3 ß

3.1.1 Options

`display=values only`

`display=formatted` (initially `formatted`)

`display=symbols only`

Changes which information is included in the expansion.
Because only present values will be included, `display=symbols only` can be used to list the segment units (though it may be preferable to use **??**$^{\rightarrow \text{P. ??}}$ or **??**$^{\rightarrow \text{P. ??}}$).

```
\nduValue{danish hartkorn}
  [display=symbols only]
  {0.0.0.0.0}

\nduValue{danish hartkorn}
  [display=values only]
  {0.0...}
```

Td. Sk. Fj. Alb. §
0 0

`format={⟨...⟩}` (initially `\VALUE\nobreakspace\SYMBOL`)

Sets how a given base unit should be formatted for display.
The placeholders `\VALUE` and `\SYMBOL` will be substituted when the value is typeset.

`replace nil with=⟨...⟩` (no default, initially empty)

`treat zero as nil` (initially not set)

The key `replace nil with` replaces nil (empty) segments with a custom string.
The key `treat zero as nil` replaces 0 with nothing, which in turn means that setting both will replace both zero and nil with the custom string.

`unit depth`=⟨*unit name*⟩ (initially no restriction)

When calculating or displaying a value, only the segments up to and including ⟨*unit name*⟩ will be considered.
In this document, the depth has been globally set to `skilling`, but older historical sub-units can be included by locally setting the depth to eg. `hvid` (or indeed not restricting it globally).
If the ⟨*unit name*⟩ is not present in the current unit group, it has no effect.

```
\nduValue{danish rigsdaler}
  [unit depth=skilling]
  {1.2.3.4.5}

\nduValue{danish rigsdaler}
  [unit depth=penning]
  {1.2.3.4.5}
```

1 Rdl. 2 ß 3 ß
1 Rdl. 2 ß 3 ß 4 Hv. 5 ß

`unit separator`=⟨...⟩ (initially `\nobreakspace`)

When displaying a value, this string will be inserted between each segment.

```
\nduValue{danish hartkorn}[
    display=values only,
    unit separator=.
  ]
  {1.2.3.4}

\nduValue{danish rigsdaler}
  [unit separator={---}]
  {1.2.3}
```

1.2.3.4
1 Rdl.—2 ß—3 ß

4 Tabular Data

Example of tabular data

```
\begingroup
\nduKeys{
%  has been set in this document's preamble:
%  tabularray column type=U,
  treat zero as nil,
  replace nil with=---,
}
\begin{tblr}{r U{danish rigsdaler}}
  \toprule
  & HEADER \\
  \midrule
  a & 1.2.3 \\
  b & 100.0.0 \\
  c & .1. \\
  \bottomrule
\end{tblr}
\endgroup
```

	Rdl.	~~ß~~	ß
a	1	2	3
b	100	—	—
c	—	1	—

In order to align values in a tabular context, the `aligned` key causes `\nduValue` to wrap each segment in a cell of equal width.

All segments will be included in the headers and cells, whether they contain a value or not (provided `unit depth` allows it). If no value is provided for the segment, and no nil replacement is specified with the **??**$^{\rightarrow \text{P. ??}}$ key, the cell will be empty.

`\nduHeader{`⟨*unit name*⟩`}[`⟨*options*⟩`]`

Convenient header showing the unit symbols. See **??** for configuration of symbols.

4.1 Options for Tabular Data

`aligned` (initially not set)

`set aligned for environment` (initially set for `tabular`)

Setting `aligned` will format the presently displayed value in aligned cells, desirable in tabular contexts.
The `set aligned for environment` key can be set to an environment name, causing `aligned` to automatically be set for those enviroments, using `\AtBeginEnvironment`. It can be set multiple times, once for each required environment.

Example of tabular data

```
\begingroup
\nduKeys{
%   has been set in this document's preamble:
%  set aligned for environment=tabular,
  treat zero as nil,
  replace nil with=---,
}
\begin{tabular}{r r}
  \toprule
  & \nduHeader{danish rigsdaler} \\
  \midrule
  a & \nduValue{danish rigsdaler}{1.2.3} \\
  b & \nduValue{danish rigsdaler}{100.0.0} \\
  c & \nduValue{danish rigsdaler}{.1.} \\
  \bottomrule
\end{tabular}
\endgroup
```

	Rdl.	℔	ß
a	1	2	3
b	100	—	—
c	—	1	—

`cell width`=⟨*length*⟩ (initially `3em`)

Changes the width of each segment.

Example usage: `cell width` key

```
\begingroup
\nduKeys{
  treat zero as nil,
  cell width=5em,
}
\begin{tabular}{r r}
  \toprule
  & \nduHeader{danish rigsdaler} \\
  \midrule
  a & \nduValue{danish rigsdaler}{1.2.3} \\
  b & \nduValue{danish rigsdaler}{100..} \\
  c & \nduValue{danish rigsdaler}{.1.} \\
  \bottomrule
\end{tabular}
\endgroup
```

	Rdl.	℟	ß
a	1	2	3
b	100		
c		1	

5 Arithmetical Operations

Basic arithmetic functions can be used to build a result for display. This is done by converting the value to an internal representation and storing it in a variable. The first time a variable is used, it is assumed that the value is 0.

Results can be gathered in two ways, either manually via the `\nduMath` macro, or automatically via the `add to variable` and `subtract from variable` keys, the latter being especially suitable in tabular contexts.

`\nduMath{`⟨*unit name*⟩`}[`⟨*options*⟩`]{`⟨*variable*⟩`}{`⟨*operator*⟩`}{`⟨*value*⟩`}`

The first arguments of `\nduMath` are identical to those of the **??**[→ P. ??] macro. In addition, it has ⟨*variable*⟩ and ⟨*operator*⟩ (one of `+ - * /`) arguments. The command does not expand to any output.
Note that mixing units in the same variable is not currently supported, and will likely give incorrect results.

Example usage: `\nduMath` macro

```
\nduMath{danish rigsdaler}{example 1}{+}{0.0.10}
\nduMath{danish rigsdaler}{example 1}{+}{..8}
\nduMath{danish rigsdaler}{example 1}{+}{0.2}
\nduMath{danish rigsdaler}{example 1}{+}{0.5.1}
% there is no output, the result 1.2.3
% will be seen in the following example.
```

`\nduResult{`⟨*unit name*⟩`}[`⟨*options*⟩`]{`⟨*variable*⟩`}`

The `\nduResult` macro takes a stored ⟨*variable*⟩ and formats it via ⟨*options*⟩ for display in the same way as **??**[→ P. ??].

Example usage: `\nduResult` macro

```
\begingroup
\nduKeys{
  aligned,
  cell width=3em,
}
\nduHeader{danish rigsdaler}\\
\nduResult{danish rigsdaler}{example 1}
\endgroup
```

Rdl.	ß	ß
1	2	3

5.1 Options for Arithmetical Operations

`add to variable`=⟨...⟩

`subtract from variable`=⟨...⟩

Setting either of these keys will cause all uses of `\nduValue` in the current group to be added to or subtracted from the variable with the given name. It can of course also be set on individual invocations of the command.

Example usage: `add to variable` key

```
\begingroup
\nduKeys{
  replace nil with=---,
  add to variable=example 2
}
\begin{tabular}{r r}
  \toprule
  & \nduHeader{danish rigsdaler} \\
  \midrule
  a & \nduValue{danish rigsdaler}{1.2.3} \\
  b & \nduValue{danish rigsdaler}{100.1.} \\
  \bottomrule
  total & \nduResult{danish rigsdaler}{example 2} \\ % = 101.3.3
\end{tabular}
\endgroup
```

	Rdl.	℔	ß
a	1	2	3
b	100	1	—
total	101	3	3

Results are global and remain accessible outside the group:

```
\nduResult{danish rigsdaler}{example 2}

And let's add add an additional 15 skilling:

\nduMath{danish rigsdaler}{example 2}{+}{0.0.15}
\nduResult{danish rigsdaler}{example 2} % = 101.4.2
```

101 Rdl. 3 ℔ 3 ß
And let's add add an additional 15 skilling:
101 Rdl. 4 ℔ 2 ß

`normalize` (initially not set)

Reformats an amount, which is useful for quick conversions.

Example usage: `normalize` key

```
100 skilling equal
\nduValue{danish rigsdaler}[normalize]{..100} % 1.0.4
```

100 skilling equal 1 Rdl. 0 ℔ 4 ß

6 Accessing Information About Units

`\nduSymbol{`⟨*unit name*⟩`}`

Expands to the symbol of the given base unit.
Set by **??**→P.??.

`\nduFactor{`⟨*unit name*⟩`}{`⟨*unit name*⟩`}`

Expands to the conversion between two base units.
Set by **??**→P.??.

```
That is, 1 \nduSymbol{rigsdaler} consists of
\nduFactor{rigsdaler}{skilling} \nduSymbol{skilling}.
```

That is, 1 Rdl. consists of 96 ß.

7 Creating New Units

If the included units are not suitable, more can be created. Pull requests are also welcome at `https://github.com/mikkelee/latex-units`.

`\nduNewBaseUnit{`⟨*unit name*⟩`}{`⟨*key/value pairs*⟩`}`

Creates a new base unit. It must contain at least a `symbol`, but a `factor` is also required for the math functions.

`\nduNewUnitGroup{`⟨*unit name*⟩`}[`⟨*key/value pairs*⟩`]{`⟨*ordered base units*⟩`}[`⟨*control sequence*⟩`]`

In order for the math functions to work, every base unitsin the group must have a conversion path to the right-most base unit, eg. if a unit group consists of base units `A, B, C`, there must be defined factors for `A`→`B` and `B`→`C`. The factor `A`→`C` is optional; if not configured, an attempt to calculate and cache it will be made internally.
It is possible to create shortcut macros for commonly used ⟨*unit name*⟩s with optional overriding options. These macros take the same arguments as the full **??**[→P.??] macro, except without the first argument (ie. the name of the unit).
Including too many sub units may make the math results awkward, as the algorithm is greedy.

```
\nduNewUnitGroup{my sletdaler}
  [units/sletdaler/symbol={Sletd.}]
  {sletdaler, ort, skilling}
  [\mySldl]
\mySldl{1.2.3}
```

1 Sletd. 2 O. 3 ß

7.1 Options For Base Units

`units/`⟨*unit name*⟩`/symbol=`⟨*symbol*⟩

Configures a symbol displaying the unit. This is used in `\nduHeader` and is also available via `\SYMBOL` when defining the **??**[→ P. ??] (see below).

`units/`⟨*unit name*⟩`/format={`⟨*prefix*⟩`}{`⟨*suffix*⟩`}`

Sets how a given base unit should be formatted for display. If none is given, the general top-level `format` key is used.

`units/`⟨*unit name*⟩`/factor=`⟨*integer*⟩ ⟨*unit name*⟩

The conversion factor of a unit is how many of an underlying unit the given unit consists of. This can be specified multiple times.
This is used by the math macros and keys to calculate the sums and products.
Can be accessed via **??**[→ P. ??].

These keys can of course also be set temporarily in **??**[→ P. ??]

```
\nduValue{danish rigsdaler}
  [units/mark/symbol=Mk.]
  {.9.}

\nduValue{danish rigsdaler}
  [units/rigsdaler/format={\VALUE~Rigsdaler og}]
  {1.2.3}

\nduValue{danish rigsdaler}[
    unit separator={---},
    units/rigsdaler/format={(\VALUE)},
    units/mark/format={[\VALUE]},
    units/skilling/format={\{\VALUE\}},
  ]
  {1.2.3}
```

9 Mk.
1 Rigsdaler og 2 ß 3 ß
(1)—[2]—{3}

8 Included Units

On the following pages are the units included with the package.

Listing of units loaded with the `british` option

```
%%% CURRENCY %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

% https://en.wikipedia.org/wiki/£sd

\nduNewBaseUnit { pound~sterling } {
        symbol = { £ } ,
        factor = { 20~shilling } ,
        format = { \SYMBOL\VALUE } ,
}

\nduNewBaseUnit { shilling } {
        symbol = { s } ,
        factor = { 12~penny } ,
        format = { \VALUE\SYMBOL } ,
}

\nduNewBaseUnit { penny } {
        symbol = { d } ,
        format = { \VALUE\SYMBOL } ,
}

\nduNewUnitGroup { british~pound~sterling~lsd } [
        unit~separator = {.~} ,
] {
        pound~sterling ,
        shilling ,
        penny
}
```

Listing of units loaded with the `danish` option

```
%%% CURRENCY %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\nduNewBaseUnit { rigsdaler } {
        symbol = { Rdl. } ,
        factor = { 6~mark } ,
}

\nduNewBaseUnit { rigsbankdaler } {
        symbol = { 6~mark } ,
}

\nduNewBaseUnit { speciedaler } {
        symbol = { Spdl. } ,
        factor = { 84~skilling } ,
}

\nduNewBaseUnit { sletdaler } {
        symbol = { Sldl. } ,
        factor = { 4~mark } ,
}

\nduNewBaseUnit { ort } {
        symbol = { O. } ,
        factor = { 24~skilling } ,
}

\nduNewBaseUnit { mark } {
        symbol = { Mk. } ,
        factor = { 16~skilling } ,
}

\nduNewBaseUnit { skilling } {
        symbol = { Sk. } ,
        factor = { 12~penning } ,
}

\nduNewBaseUnit { hvid } {
        symbol = { Hv. } ,
        factor = { 4~penning } ,
}

\nduNewBaseUnit { penning } {
        symbol = { P. } ,
}

\nduNewUnitGroup { danish~rigsdaler } {
        rigsdaler ,
        mark ,
        skilling ,
        hvid ,
        penning
```

```
}[\rdl]

\nduNewUnitGroup { danish~sletdaler } {
        sletdaler ,
        mark ,
        skilling ,
        hvid ,
        penning
}[\sldl]

\nduNewUnitGroup { danish~rigsbankdaler } {
        rigsbankdaler ,
        skilling
}[\rbdl]

\nduNewUnitGroup { danish~speciedaler } {
        speciedaler ,
        skilling
}[\spdl]

%%%% AREA %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\nduNewBaseUnit { tønde } {
        symbol = { Td. } ,
        factor = { 96~skæppe } ,
}

\nduNewBaseUnit { skæppe } {
        symbol = { Sk. } ,
        factor = { 96~skilling } ,
}

\nduNewBaseUnit { fjerdingkar } {
        symbol = { Fj. } ,
        factor = { 96~skilling } ,
}

\nduNewBaseUnit { album } {
        symbol = { Alb. } ,
        factor = { 4~penning } ,
}

\nduNewUnitGroup { danish~hartkorn } {
        tønde,
        skæppe,
        fjerdingkar,
        album,
        penning
}[\hartkorn]

%%%% WEIGHT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
```

```
\nduNewBaseUnit { skippund } {
        symbol = { Spd. } ,
        factor = { 320~skålpund } ,
}

\nduNewBaseUnit { lispund } {
        symbol = { Lpd. } ,
        factor = { 16~skålpund } ,
}

\nduNewBaseUnit { skålpund } {
        symbol = { Pd. } ,
}

\nduNewUnitGroup { danish~pund } {
        skippund,
        lispund,
        skålpund
}
```

Listing of units loaded with the `german` option

```
%%% CURRENCY %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\nduNewBaseUnit { reichsthaler } {
        symbol = { Rthl. } ,
        factor = { 30~groschen } ,
}

\nduNewBaseUnit { groschen } {
        symbol = { Gr. } ,
        factor = { 12~pfennig } ,
}

\nduNewBaseUnit { pfennig } {
        symbol = { Pf. } ,
}

\nduNewUnitGroup { german~reichsthaler } {
        reichsthaler ,
        groschen ,
        pfennig
}
```