Non-Decimal Units for LaTeX

Mikkel Eide Eriksen
mikkel.eriksen@gmail.com

October 10, 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`.

[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 page 15 for details.

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

Alternately, one may configure new units via `\nduNewBaseUnit` → P. 13 and `\nduNewUnitGroup` → P. 13.

`\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 `replace nil with` →P. 5 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`

`display=numprint`

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 `\nduHeader` → P. 9 or `\nduSymbol` → P. 12).

```
\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

Finally, it is possible to use the numprint package to format numbers, especially useful for larger values. Note that this may be counter to the formatting of some units (eg. British pounds), in which case one may prefer the `use numprint` key instead.

```
\nduValue{danish rigsdaler}
  [display=numprint]
  {1000}
```

1,000 Rdl.

`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

`use numprint` (not set initially)

When displaying a value, use the numprint package, including when using the `format` key. You can of course also configure the numprint settings, either in a group or locally.

```
\begingroup
\selectlanguage{ngerman}
\nduValue{danish rigsdaler}
  [use numprint]
  {1000.0}
\endgroup
```

1 000 Rdl. 0 ß

4 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 global 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 two being especially suitable in tabular contexts.

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

The first arguments of `\nduMath` are identical to those of the `\nduValue` →P. 3 macro. In addition, it has ⟨*variable*⟩ and ⟨*operator*⟩ (one of `+ - * /`) arguments. The command does not expand to any output.
Note that mixing unit groups 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 `\nduValue` →P. 3.

Example usage: `\nduResult` macro

```
\nduResult{danish rigsdaler}{example 1} % = 1.2.3

And let's add an additional 15 skilling:
\nduMath{danish rigsdaler}{example 1}{+}{0.0.15}
\nduResult{danish rigsdaler}{example 1} % = 1.3.2
```

1 Rdl. 2 ß 3 ß
And let's add an additional 15 skilling: 1 Rdl. 3 ß 2 ß

4.1 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.} \\
  \midrule
  total & \nduResult{danish rigsdaler}{example 2} \\ % = 101.3.3
  \bottomrule
\end{tabular}
\endgroup
```

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

`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

5 Tabular Data

There are a couple of ways to display values in tabular context, centered around explicitly or implicitly setting the `aligned` key, which causes `\nduValue` to wrap each segment in a cell of configurable width.

Additionally, all segments will be included in 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 `replace nil with` → P. 5 key, the cell will be empty.

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

Convenient header showing the unit symbols. See page 13 for configuration of symbols.

5.1 Options for Tabular Data

`aligned` (initially not set)

Causes each value to be wrapped in right-aligned cells of configurable width.

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

Changes the width of each cell. One may supply a bracketed comma-separated list of widths. If the list is shorter than the number of base units in the group, the last width will be repeated. See page 10 for example.

`set aligned for environment=`⟨*name*⟩ (initially not set)

`tabularray column type=`⟨*letter*⟩ (initially not set)

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. See page 10 for example.
The `tabularray column type` key can be used to create a column type, which automatically wraps the column contents in `\nduValue`. The column type takes two arguments, a unit group and a set of key values for further configuration. Additionally, the special values `HEADER`, `RESULT`, and `SKIP` will respectively use `\nduHeader`, `\nduResult`, and skip the cell. See page 11 for example.[a]

[a] Thanks to Yukai Chou for help with tabularray integration.

Example of tabular data using `set aligned for environment`

```
\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	—

Example usage: `cell widths` key

```
\begingroup
\nduKeys{
  cell widths={5em, 1.5em},
}
\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	

Example of tabular data using `tabularray column type`

```
\begingroup
%  has been set in this document's preamble:
%  tabularray column type=U
\begin{tblr}{
  r
  U{danish rigsdaler}{add to variable=table result 1}|
  U{danish rigsdaler}{add to variable=table result 2}
}
  \toprule
  & HEADER & HEADER\\
  \midrule
  a & 1.2.3 & ..15 \\
  b & 100.0.0 & ..10 \\
  c & .1. & ..2 \\
  \midrule
  total & RESULT & RESULT \\
  \bottomrule
\end{tblr}
\endgroup
```

	Rdl.	₰	ß	Rdl.	₰	ß
a	1	2	3			15
b	100	0	0			10
c		1				2
total	101	3	3	0	1	11

6 Accessing Information About Units

`\nduSymbol{`⟨*unit name*⟩`}`

Expands to the symbol of the given base unit.
Set by `units/`⟨*unit name*⟩`/symbol` → P. 14.

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

Expands to the conversion between two base units.
Set by `units/`⟨*unit name*⟩`/factor` → P. 14.

```
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 (see below).

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

In order for the math functions to work, every base unit in 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 `B`→`C` and either `A`→`C` or `A`→`B`; if only the latter is configured, an attempt to calculate and cache it will be made internally.
It is possible to create shortcut macros for commonly used unit groups with optional overriding options. These macros take the same arguments as the full `\nduValue` →P. 3 macro, except without the first argument (ie. the name of the unit).
Including several 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 `\nduValue`, `\nduHeader`, and is also available via `\SYMBOL` when defining the `units/⟨unit name⟩/format` (see also `??`→ P. ??).

`units/⟨unit name⟩/format={⟨...⟩}`

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 `\nduFactor`→ P. 12.

These keys can of course also be set temporarily in `\nduValue`→ P. 3

```
\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 ℔ 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 = { Rbdl. } ,
        symbol = { 96~skilling } ,
}

\nduNewBaseUnit { rigsdaler specie } {
        symbol = { Rds. } ,
        symbol = { 192~skilling } ,
}

\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 = { 3~hvid } ,
}

\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 = { 8~skæppe } ,
}

\nduNewBaseUnit { skæppe } {
        symbol = { Sk. } ,
        factor = { 4~fjerdingkar } ,
}

\nduNewBaseUnit { fjerdingkar } {
        symbol = { Fj. } ,
        factor = { 3~album } ,
}

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

\nduNewUnitGroup { danish~hartkorn } {
        tønde,
        skæppe,
        fjerdingkar,
```

```
        album,
        penning
}[\hartkorn]

%%%% WEIGHT %%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%

\nduNewBaseUnit { skippund } {
        symbol = { Spd. } ,
        factor = { 20~lispund } ,
}

\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
}
```