Non-Decimal Units for LaTeX

Mikkel Eide Eriksen
mikkel.eriksen@gmail.com

October 3, 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 ℳ 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 page 17 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 name⟩}[⟨options⟩]{⟨value⟩}`

Formats ⟨*value*⟩ according to the setup configured for the ⟨*unit name*⟩, as well as any provided ⟨*options*⟩. The number of decimal points and the values between them determine how many and which segments are displayed.
Empty segments 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 ℔ 3 ß
1 Rdl.
2 ℔
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 those segments with a value will be included, show=symbols can be used to list the segment units (though if only one or two is needed, it may be preferable to use **??**→ P. ??).

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

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

Td. Sk. Fj. Alb. ₰
Td. Sk.

See also section 4 for further discussion on possible options.

`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.
These keys also apply in non-tabular contexts, but are probably most useful here.

Example usage: `replace nil with` key

```
\begingroup
\nduKeys{
  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.	ℳ	ß
a	1 Rdl.	2 ℳ	3 ß
b	100 Rdl.	— ℳ	— ß
c	— Rdl.	1 ℳ	— ß

3.2 Tabular Data

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

Additionally, the `\nduHeader` macro provides a convenient header showing the unit symbols.

All segments will be included in the headers and cells, whether they contain a value or not. 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⟩]`

Formats the unit symbols in boxes suitable for a header. See page 13 for configuration of symbols.

3.2.1 Options

aligned (initially not set)

set aligned for environment (initially set for tabular)

Setting aligned will format the presently displayed header in aligned boxes, desirable in tabular contexts.
Additionally, 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 usage: \nduHeader and \nduValue macros with aligned key.

```
\begingroup
\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 Rdl.	2 ℔	3 ß
b			100 Rdl.
c			1 ℔

cell width=⟨*length*⟩ (initially 5em)

Changes the width of each segment.

Example usage: **cell width** key

```
\begingroup
\nduKeys{
  cell width=3em,
}
\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 Rdl.	2 ℔	3 ß
b			100 Rdl.
c			1 ℔

3.3 Arithmetical Operations

Basic arithmetic functions can be used to build a result for display. Internally, this is done by converting the value to a representation, which is the total number of the smallest usable unit, eg. 1 Rdl. 2 ℔ 3 ß is 131 skilling.

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

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 first time a variable is used, it is assumed that the value is 0. The given value is then converted to its internal representation and stored in the variable. 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.

The `\nduResult` macro takes a stored ⟨*variable*⟩ and formats it for display in the same way as `\nduValue`→P. 3.

Both may be further configured via the ⟨*options*⟩ in the same way as the other macros.

Note that the `treat zero as nil` key does not work for math results at present.

Example usage: `\nduMath` and `\nduResult` macros

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

1 Rdl. 2 ℳ 3 ß

Example usage: `\nduResult` macro

```
\nduHeader{danish rigsdaler} % TODO newline?
\nduResult{danish rigsdaler}[aligned]{example 1}
```

```
  Rdl.
     ℳ
     ß
1 Rdl.
   2 ℳ
   3 ß
```

3.3.1 Options

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.

Example usage: add to variable key

```
\begingroup
\nduKeys{
  cell width=3em,
  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 Rdl.	2 ₰	3 ß
b	100 Rdl.	1 ₰	— ß
total	101 Rdl.	3 ₰	3 ß

Results are global and remain accessible outside the group:

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

101 Rdl. 3 ₰ 3 ß

Adding an additional 15 skilling to the existing result gives:

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

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 ℳ 4 ß

4 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⟩}`

See below for available settings.

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

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 `\nduValue`→ P. 3 macro, except without the first argument (ie. the name of the unit).

```
\nduNewUnitGroup{my sletdaler}
%  [units/sletdaler/symbol={S\textsuperscript{dl}}] % TODO
  {sletdaler, ort, skilling}
  [\mySldl]
\mySldl{1.2.3}
```

1 Sldl. 2 O. 3 ß

4.0.1 Options

segment separator=⟨...⟩ (initially ~)

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

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

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

1.2.3.4
1 Rdl.—2 ℳ—3 ß

unit depth=⟨*string*⟩ (initially no restriction)

When calculating or displaying a value, only the segments up to and including ⟨*integer*⟩ will be considered.
In this document, the depth has been globally set to `skilling` for `danish rigsdaler`, but the older historical sub-units can be included by locally setting the depth to eg. `hvid` (or indeed not restricting it globally).

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

1 Rdl. 2 ℳ 3 ß 4 Hv. 5 ₰

⟨*n*⟩/symbol=⟨*symbol*⟩ (no default, initially undefined)

Configures a symbol displaying the unit. This is used in `\nduHeader` and is also available via `\nduSym` when defining the `segment` ⟨*n*⟩`/display`$^{\to \text{P. 15}}$ (see below).
If none is configured, an attempt to look up a common symbol by its name is made. These can be configured with **??**$^{\to \text{P. ??}}$.

`segment ⟨n⟩/prefix=`⟨...⟩ (initially set to `{}`)

`segment ⟨n⟩/suffix=`⟨...⟩ (initially set to `{ \nduSym}`)

When displaying a value, segments will be wrapped between the ⟨*prefix*⟩ and ⟨*suffix*⟩.
The macro `\nduSym` is available here to show the symbol configured for the segment.

`segment ⟨n⟩/display={`⟨*prefix*⟩`}{`⟨*suffix*⟩`}`

Sets both `segment ⟨n⟩/prefix` and `segment ⟨n⟩/suffix` at the same time.

`segment ⟨n⟩/factor=`⟨*integer*⟩ (no default, initially undefined)

The conversion factor of a segment is how many of the underlying segment the given segment consists of.
This is used in the math macros, in order to calculate the correct segment values.
Can be accessed via **??**→ P. ??.

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}[
    segment 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}

`create macro named`=⟨*control sequence*⟩ (no default, initially empty)

Units may provide a default shortcut macro, for example the `danish rigsdaler` unit configures `\rdl`.
This is done via **??**$^{\rightarrow \text{P. ??}}$ which describes the arguments of the resulting macros.

```
\rdl{2.3.}
```

2 Rdl. 3 ß

5 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 = { 240~penny } ,
        format = { \SYMBOL\VALUE } ,
}

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

\nduNewBaseUnit { penny } {
        symbol = { d } ,
        factor = { 1~penny } ,
        format = { \VALUE\SYMBOL } ,
}

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

Listing of units loaded with the **danish** option

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

\nduNewBaseUnit { rigsdaler } {
        symbol = { Rdl. } ,
        factor = { 96~skilling } ,
        factor = { 1152~penning } ,
}

\nduNewBaseUnit { rigsbankdaler } {
        symbol = { Rbdl. } ,
        factor = { 96~skilling } ,
}

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

\nduNewBaseUnit { sletdaler } {
        symbol = { Sldl. } ,
        factor = { 64~skilling } ,
        factor = { 768~penning } ,
}

\nduNewBaseUnit { ort } {
        symbol = { O. } ,
        factor = { 24~skilling } ,
        factor = { 288~penning } ,
}

\nduNewBaseUnit { mark } {
        symbol = { Mk. } ,
        factor = { 16~skilling } ,
        factor = { 192~penning } ,
}

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

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

```
\nduNewBaseUnit { penning } {
        symbol = { P. } ,
        factor = { 1~penning } ,
}

\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 = { 360~pfennig } ,
}

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

```
\nduNewBaseUnit { pfennig } {
        symbol = { Pf. } ,
        factor = { 1~pfennig } ,
}

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

Index

add to variable key, 9
aligned key, 6

british key, 2, 15

cell width key, 7
create macro named key, 14

danish key, 2, 16

german key, 2, 17

Keys
- add to variable, 9
- aligned, 6
- british, 2, 15
- cell width, 7
- create macro named, 14
- danish, 2, 16
- german, 2, 17
- replace nil with, 4
- restrict segment depth, 12
- segment ⟨n⟩/display, 13
- segment ⟨n⟩/factor, 13
- segment ⟨n⟩/name, 12
- segment ⟨n⟩/symbol, 13
- segment separator, 12
- set aligned for environment, 6
- show, 4
- subtract from variable, 9
- treat zero as nil, 4

\nduCommonFactors, 11
\nduCommonSymbols, 11
\nduFactor, 10
\nduHeader, 6, 13
\nduMath, 8
\nduName, 10
\nduNewMacro, 11
\nduNewUnit, 11
\nduResult, 8
\nduset, 2
\nduSym, 13
\nduSymbol, 10
\nduValue, 3, 6

replace nil with key, 4

restrict segment depth key, 12

segment ⟨n⟩/display key, 13
segment ⟨n⟩/factor key, 13
segment ⟨n⟩/name key, 12
segment ⟨n⟩/symbol key, 13
segment separator key, 12
set aligned for environment key, 6
show key, 4
subtract from variable key, 9

treat zero as nil key, 4

\usepackage, 2