Non-Decimal Units for LaTeX

Mikkel Eide Eriksen
mikkel.eriksen@gmail.com

September 30, 2020

1 Preface

Many historical unit systems were non-decimal. For example, the Danish rigsdaler[1] — where 1 rigsdaler consists of 6 mark, each again consisting of 16 skilling for a total of 96 skilling or 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 high 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 summing.

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

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 and areas
- `german` Currencies

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

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

3 Usage

3.1 Formatting Values

`\nduFormatValue{`⟨*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.

Example usage: `\nduFormatValue` macro

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

1 Rdl. 2 ℔ 3 ß
1 Rdl.
2 ℔
3 ß

3.1.1 Options

`show=values`

`show=values and symbols` (initially `values and symbols`)

`show=symbols`

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. ??).

```
\nduFormatValue{danish hartkorn}
  [show=symbols]
  {0.0.0.0.0}

\nduFormatValue{danish hartkorn}
  [show=symbols]
  {0.0...}
```

Td. Skp. Fjk. Alb. Pg.
Td. Skp.

See also **??** for further discussion on possible options.

3.2 Tabular Data

In order to align values in a tabular context, the `\nduAlignedHeader` and `\nduAlignedValue` macros wrap each segment in a `\makebox` of equal width.

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 **??** → P. ?? key, the box will be empty.

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

Formats the unit symbols in boxes suitable for a header. See **??** for configuration of symbols.

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

See **??** → P. ?? for possible arguments.

Example usage: `\nduAlignedHeader` and `\nduAlignedValue` macros

```
\begin{tabular}{r r}
  \toprule
    & \nduAlignedHeader{danish rigsdaler} \\
  \midrule
  a & \nduAlignedValue{danish rigsdaler}{1.2.3} \\
  b & \nduAlignedValue{danish rigsdaler}{100..} \\
  c & \nduAlignedValue{danish rigsdaler}{.1.} \\
  \bottomrule
\end{tabular}
```

	Rdl.	℔	ß
a	1	2	3
b	100		
c		1	

3.2.1 Options

`aligned value width`=⟨*length*⟩ (initially `5em`)

Changes the width of each segment.

Example usage: `aligned value width` key

```
\begingroup
\nduset{
  aligned value width=3em,
}
\begin{tabular}{r r}
  \toprule
  & \nduAlignedHeader{danish rigsdaler} \\
  \midrule
  a & \nduAlignedValue{danish rigsdaler}{1.2.3} \\
  b & \nduAlignedValue{danish rigsdaler}{100..} \\
  c & \nduAlignedValue{danish rigsdaler}{.1.} \\
  \bottomrule
\end{tabular}
\endgroup
```

	Rdl.	℔	ß
a	1	2	3
b	100		
c		1	

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

Replaces nil (empty) segments with a string.

Example usage: `replace nil with` key

```
\begingroup
\nduset{
  replace nil with=---,
}
\begin{tabular}{r r}
  \toprule
  & \nduAlignedHeader{danish rigsdaler} \\
  \midrule
  a & \nduAlignedValue{danish rigsdaler}{1.2.3} \\
  b & \nduAlignedValue{danish rigsdaler}{100..} \\
  c & \nduAlignedValue{danish rigsdaler}{.1.} \\
  \bottomrule
\end{tabular}
\endgroup
```

	Rdl.	ℵ	ß
a	1	2	3
b	100	—	—
c	—	1	—

3.3 Summing Values

Values can be accumulated in a named sum in two ways, either manually via the `\nduAddToSum` macro, or automatically via the `sum to` key.

`\nduAddToSum{`⟨*unit name*⟩`}[`⟨*options*⟩`]{`⟨*sum name*⟩`}{`⟨*value*⟩`}`

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

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

The arguments of `\nduAddToSum` are identical to those of the **??**→P.?? macro, except for the addition of the ⟨*sum name*⟩ argument, under which the sum will be accumulated. It does not expand to any output.
The `\nduFormatSum` macro takes the ⟨*sum name*⟩ and formats it for display in the same way as **??**→P.??.
Likewise, `\nduAlignedSum` macro formats a sum in the same way as **??**→P.??.
All three may be further configured via the ⟨*options*⟩.

Example usage: `\nduAddToSum` and `\nduFormatSum` macros

```
\nduAddToSum{danish rigsdaler}{example 1}{0.0.10}
\nduAddToSum{danish rigsdaler}{example 1}{..8}
\nduAddToSum{danish rigsdaler}{example 1}{0.2}
\nduAddToSum{danish rigsdaler}{example 1}{0.5.1}
\nduFormatSum{danish rigsdaler}{example 1} % = 1.2.3
```

1 Rdl. 2 ß 3 ß

Example usage: `\nduAlignedSum` macro

```
\nduAlignedHeader{danish rigsdaler}\\
\nduAlignedSum{danish rigsdaler}{example 1}
```

Rdl.	ß	ß
1	2	3

3.3.1 Options

`sum to`=⟨*name*⟩ (initially empty)

Setting this key will cause all uses of `\nduFormatValue` and `\nduAlignedValue` in the current group to be summed under the given name.

Example usage: `sum to` key

```
\begingroup
\nduset{
  aligned value width=3em,
  replace nil with=---,
  sum to=example 2
}
\begin{tabular}{r r}
  \toprule
  & \nduAlignedHeader{danish rigsdaler} \\
  \midrule
  a & \nduAlignedValue{danish rigsdaler}{1.2.3} \\
  b & \nduAlignedValue{danish rigsdaler}{100.1.} \\
  \bottomrule
  total & \nduAlignedSum{danish rigsdaler}{example 2} \\ % = 101.3.3
\end{tabular}
\endgroup
```

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

Sums are global and remain accessible outside the group:

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

101 Rdl. 3 ℔ 3 ß

Adding an additional 15 skilling to the existing sum gives:

```
\nduAddToSum{danish rigsdaler}{example 2}{0.0.15}
\nduFormatSum{danish rigsdaler}{example 2} % = 101.4.2
```

101 Rdl. 4 ℔ 2 ß

3.4 Accessing Information About Units

`\nduName{`⟨*unit name*⟩`}{`⟨*segment*⟩`}`

Expands to the name of the the given segment of the unit.
Set by **??**[→ P. ??].

`\nduSymbol{`⟨*unit name*⟩`}{`⟨*segment*⟩`}`

Expands to the symbol of the the given segment of the unit.
Set by **??**[→ P. ??].

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

Expands to the conversion factor of the the given segment of the unit, ie. how many of the underlying segment the given segment consists of.

```
That is, 1 \nduName{danish rigsdaler}{0} consists of
\nduFactor{danish rigsdaler}{0} \nduName{danish rigsdaler}{1}.
```

That is, 1 rigsdaler consists of 6 mark.

3.5 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`.

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

Units can have up to 5 segments, numbered ⟨*0-4*⟩. The left-most segment, that is, the *top* or *root* segment, is numbered `0`.
The numeral part of the below key paths `segment 0/` can be any integer up to `4`, ie. `segment 4/`. The internal number of segments is determined by how many name keys are created.
See below for available settings.

`\nduNewMacro{`⟨*unit name*⟩`}[`⟨*key/value pairs*⟩`]{`⟨*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 **??**→P.?? macro, except without the first argument (ie. the name of the unit).

```
\nduNewMacro{danish rigsdaler}
  [segment 0/symbol={R\textsuperscript{dl}}]
  {myRdl}
\myRdl{1.2.3}
```

1 R$^{\text{dl}}$ 2 ℔ 3 ß

`\nduCommonSymbols{`⟨*key/value pairs*⟩`}`

It is possible to configure commonly used symbols using the form ⟨*name*⟩=⟨*symbol*⟩. These will be used as fallbacks if no specific symbol is configured for a segment via **??**→P.??.

3.5.1 Options

`segment separator=`⟨...⟩ (initially ~)

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

```
\nduFormatValue{danish hartkorn}[
    show=values,
    segment separator=.
  ]
  {1.2.3.4}

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

1.2.3.4
1 Rdl.—2 ℔—3 ß

`restrict segment depth=`⟨*integer*⟩ (initially no restriction)

When summing 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 2 for `danish rigsdaler`, but the older historical sub-unit penning can be included by locally setting the depth to 3 (or indeed not restricting it globally).

```
\nduFormatValue{danish rigsdaler}
  [restrict segment depth=3]
  {1.2.3.4}
```

1 Rdl. 2 ℔ 3 ß 4 §

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

Gives the proper name of the segment's unit. Used internally to determine how many segments the unit contains.
Can be accessed with by **??**[→ P. ??].

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

Configures a symbol displaying the unit. This is used in `\nduAlignedHeader` and is also available via `\nduSym` when defining the **??**[→ P. ??] (see below).
If none is configured, an attempt to look up a common symbol by its name is made. These can be configured with **??**[→ P. ??].

`segment` ⟨*n*⟩`/display={`⟨*prefix*⟩`}{`⟨*suffix*⟩`}` (initially `{}{ \nduSym}`)

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

`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 when summing values, in order to calculate the correct segment values.
Can be accessed via **??**→P. ??.

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

```
\nduFormatValue{danish rigsdaler}
  [segment 1/symbol=Mk.]
  {.9.}

\nduFormatValue{danish rigsdaler}
  [segment 0/display={}{ Rigsdaler og}]
  {1.2.3}

\nduFormatValue{danish rigsdaler}[
    segment separator={---},
    segment 0/display={(}{)},
    segment 1/display={[}{]},
    segment 2/display={\{}{\}},
  ]
  {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 **??**→P. ?? which describes the arguments of the resulting macros.

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

2 Rdl. 3 ℔

4 Included Units

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

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

```
% https://en.wikipedia.org/wiki/£sd
\nduNewUnit{british pound sterling lsd}{
        segment 0/name=pound sterling,
        segment 1/name=shilling,
        segment 2/name=penny,
        segment 0/symbol=£,
        segment 1/symbol=s,
        segment 2/symbol=d,
        segment 0/display={\nduSym}{},
        segment 1/display={}{\nduSym},
        segment 2/display={}{\nduSym},
        segment 0/factor=20,
        segment 1/factor=12,
        unit separator={. },
}
```

Listing of units loaded with the `danish` option

```
\RequirePackage{fontspec}
\newfontfamily\mufifont{Palemonas MUFI}

\RequirePackage[
        MUFI,
        fonts={
                MUFI=\mufifont,
        },
]{unicode-alphabets}

\nduCommonSymbols{%
        mark=\mufi{markflour},
        skilling=\mufi{schillgerm},
        penning=\mufi{20B0},
}

\nduNewUnit{danish rigsdaler}{
        segment 0/name=rigsdaler,
        segment 1/name=mark,
        segment 2/name=skilling,
        segment 3/name=penning,
        segment 0/symbol=Rdl.,
        segment 0/factor=6,
        segment 1/factor=16,
        segment 2/factor=12,
        create macro named=rdl,
}

\nduNewUnit{danish sletdaler}{
        segment 0/name=sletdaler,
        segment 1/name=mark,
        segment 2/name=skilling,
        segment 0/symbol=Sldl.,
        segment 0/factor=4,
        segment 1/factor=16,
        create macro named=sldl,
}

\nduNewUnit{danish rigsbankdaler}{
        segment 0/name=rigsbankdaler,
        segment 1/name=skilling,
        segment 0/symbol=Rbd.,
        segment 0/factor=96,
        create macro named=rbd,
}

\nduNewUnit{danish hartkorn}{
```

```
        segment 0/name=tønde,
        segment 1/name=skæppe,
        segment 2/name=fjerdingkar,
        segment 3/name=album,
        segment 4/name=penning,
        segment 0/symbol=Td.,
        segment 1/symbol=Skp.,
        segment 2/symbol=Fjk.,
        segment 3/symbol=Alb.,
        segment 4/symbol=Pg.,
        segment 0/factor=8,
        segment 1/factor=4,
        segment 2/factor=3,
        segment 3/factor=4,
        create macro named=hartkorn,
}
```

Listing of units loaded with the `german` option

```
\RequirePackage{fontspec}
\newfontfamily\mufifont{Palemonas MUFI}

\RequirePackage[
        MUFI,
        fonts={
                MUFI=\mufifont,
        },
]{unicode-alphabets}

\nduNewUnit{german reichsthaler}{
        segment 0/name=reichsthaler,
        segment 1/name=silbergroschen,
        segment 2/name=pfennig,
        segment 0/symbol=\mufi{reichtalold},
        segment 1/symbol=S\mufi{grosch},
        segment 2/symbol=\mufi{20B0},
        segment 0/factor=30,
        segment 1/factor=12,
        unit separator={~},
}
```

Index