anyt
anytanyt

CURRENCY

v0.3 2017/07/04

Print monetary units

Antoine Lejay

https://github.com/antoinelejay/currency.git

antoine.lejay@univ-lorraine.fr

This packages aims at typesetting monetary units in a consistent way.

Table of Contents

1 Introduction

Strangely enough, it seems that no package deals with a conveninent, normalized way to print monetary units and values. Built on the top of the siunitx, this package then aims at providing a consistant and coherent way to format such quantities. In particular, we consider printing values and unit

- In the ISO format (ISO 4217),
- In their usual name (dollar, euro, ...), singular and plurals,

- Using their ususal symbols ($, €,...).

The currency code ISO 4217 specifies the code of the currency as a three-letters code. The first two ones are the code of the country according to ISO 3166. The last one is the name of the currency name.

This package creates macros for defined currencies which follow the ISO 4217 codes.

This package is then useful for current monetary units, using the decimal systems and cents, written in western format. Non decimal systems such as the *pound shilling pennies* are not supported.

2 Licence and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3 or later (`http://www.latex-project.org/lppl.txt`). The software has the status "maintained."

The currency package needs and thus loads the packages siunitx, pgfkeys, etoolbox, xparse, expl3, textcomp and eurosym.

3 Currency definition

`\DefineCurrency{`⟨*ISO*⟩`}{`⟨*maps*⟩`}`

defines a new currency (there is no warning if a currency is being redefined).

A new currency is defined with this command. The argument `(ISO)` is the ISO format (a unique code in three letters based on country codes [**iso**; **iso:wikipedia**], *e. g.*, USD for *United States Dollar*), or any suitable name from which commands will be created. Its characteristics are defined through key-values pairs, the so-called `(map)`.

When `XXX` is the ISO code (or any other code), it defines two commands `\dXXX` and `\cXXX` for using these units in the text. These commands are defined globally. See Section 4.

For example, to define the US dollars, we set

```
1  \DefineCurrency{USD}{name={dollar},plural={dollars},iso={USD},kind=iso,
   symbol={\$}}
```

Other styles may be defined through `\pgfkeys` by following this example:

```
1  \pgfkeys{/currency/my style/.style={locale=US,kind=plural}}
2  \dUSD[my style]{100.10}
```

100.10 dollars

Key-values when defining or using a currency. Several key-values pair are defined. Since this package relies on siunitx, it is also possible to use any key-values from this latest package, for example to control the ways the values are printed (separators, ...).

`name` = {⟨*name*⟩} Default: `zzz`
The name of the currency (*e. g.*, dollar, yen, ...).

`plural` = {⟨*plural*⟩} Default: *`name`* `s`
The plural form of the name of the currency.

`iso` = {⟨*iso*⟩}
The ISO code of the currency.

`symbol` = {⟨*symbol*⟩} Default: ¤
The symbol for the currency, if any (*e. g.*, €, $, ...).

`pre-between` = {⟨*tokens*⟩} Default: `no break space`
The tokens that are placed between the name and the value when the name is printed before.

`post-between` = {⟨*tokens*⟩} Default: `no break space`
The tokens that are placed between the name and the value when the name is printed after.

`before` = {⟨*tokens*⟩} Default:
What is printed before.

`before+` = {⟨*tokens*⟩}
Append the content to `before`.

`before<` = {⟨*tokens*⟩}
Prepend the content to `before`.

`font` = {⟨*tokens*⟩}
For setting up the font which is used for both the unit and the amount (previous uses of `font` are overrided).

`font+` = {⟨*tokens*⟩}
Add the content to `font`.

`after` = {⟨*tokens*⟩}
What is printed after.

`after+` = {⟨*tokens*⟩}
Append to what is print after.

after< = {⟨*tokens*⟩}
Prepend to what is print after.

prefix = {⟨*tokens*⟩} (initially empty)
What is printed before the name (*e. g.*, k, M, ...).

kind = iso|plural|name|symbol
The representation of the monetary unit.

cents = true|false|always
Control the way the cents are printed.

pre = true|false
Select if the unit should be print before or after the value (only for ISO code and symbols).

number = true|false
Parse the values (interface to parse-numbers from siunitx).

base = {⟨*integer*⟩} Default: 2
Number of digits for the cents.

Examples No currency are actually defined in currency. Euros, US dollars, yens and pounds could be defined by

```
1  \DefineCurrency{EUR}{name={euro}, plural={euros}, symbol={\euro}, iso={
   EUR}, kind=iso}
2  \DefineCurrency{USD}{name={dollar}, plural={dollars}, symbol={\$}, iso
   ={USD}, kind=iso}
3  \DefineCurrency{JPY}{name={yen}, plural={yens}, symbol={\textyen}, iso
   ={JPY}, kind=iso,cents=false}
4  \DefineCurrency{GBP}{name={pound},pre=true,plural={pounds}, symbol={\
   pounds}, iso={GBP}, kind=iso}
```

4 Using currencies

4.1 Using currencies

Currencies are used with or without amounts. They could also be changed locally.

\CurrencySetup{⟨*maps*⟩}
This command defines a *style* in the sense of pgfkeys, that is a series of keys-values pairs. These maps are exectued after the ones defining the format of the currency but before the optional argument passed to \dXXX or \cXXX where XXX is the ISO 4217

code of the currency. It could be used to change locally the setting of a currency. Using this command overrides the previous settings of \CurrencySetup. The command \CurrencySetupAppend append to the current style.:w The style is stored in /currency /currency/.style.

\CurrencySetupAppend{⟨*maps*⟩}

Documentation corrected in version 0.2

Similar to \CurrencySetup but append the style.

\cXXX[⟨*maps*⟩]

Print only the monetary unit with currency code XXX (mnemonic *c* stands for *currency*).

\dXXX[⟨*maps*⟩]{⟨*value*⟩}

Print the value with the monetary unit with currency code XXX (mnemonic *d* stands for *display*).

\vXXX[⟨*maps*⟩]{⟨*value*⟩}

Introduced in version 0.2

alias for \dXXX (mnemonic *v* stands for *value*).

4.2 How currencies are composed?

The commands \cXXX and \dXXX are expanded inside a group. The argument ⟨*value*⟩ for \cXXX is stored into \value. Besides, the unit is stored into \currencyunit according to the value of kind.

When using \dXXX, the order in which the elements are composed is

font before prefix \currencyunit pre-between \value after

when the currency unit is printed before (pre=false), and

font before \value post-betwen prefix \currencyunit after

otherwise (pre=true).

The rules are

- The value (mandatory argument) specified by ⟨*value*⟩ is printed using \num{\value} using the \num command from siunitx, and the value is stored locally into \value.
- Both prefix and unit are enclosed into a \text command so that they could safely be used in math mode.

When using \cXXX, the order in which the elements are composed is

font before prefix \currencyunit after

where prefix and \currencyunit are enclosed in a \text command. The boolean option pre is useless in this case.

4.3 The hierarchy of keys definitions

The order in which the keys are defined (and then overwritten) is

- Default options values.
- Command \DefineCurrency.
- Command \CurrencySetup, which is a shorthand for defining the pgfkey /currency /currency/.style.
- Command \cXXX ou \dXXX (the commands are executed in a group).

4.4 Examples

4.4.1 Punctuation.

The commands \dXXX and \cXXX are defined using the xparse package. The space is preserved after the command so that there is no need to use \ after a command.

```
1  The total gross salary is \dEUR{2000}. A part of \dEUR{1500} forms the
   net salary.
```

The total gross salary is 2000 EUR. A part of 1500 EUR forms the net salary.

Using a prefix

```
1  The total cost of the project is \dEUR[prefix=M,kind=symbol]{0.5}.
```

The total cost of the project is 0.50 M€.

4.4.2 Changing the base

Most of the currencies have *cents*, that is a monetary unit equal to $1/100$ of the monetary unit. It is however possible to use another number of digits, either for special purposes, or for monetary units with other bases such as the Kuwaiti dinar which is decomposed in 1000 fils.

```
\DefineCurrency{KWD}{name={dinar},plural={dinars},symbol={KD},iso={KWD
},kind=iso,base=3}
$\dUSD{1}=\dKWD{0.29963}$
```

1 USD = 0.300 KWD

4.4.3 Changing the font

Using the `font = {⟨tokens⟩}` key, it is possible to change the font which is used for the monetary units (remember that everything is enclosed into a group). When used in the currency definition and in `\CurrencySetup`, it is however supersed by any other `font = {⟨tokens⟩}` key used in `\dXXX` (A similar result could be obtained with `before = {⟨tokens⟩}`, which aims at putting some material). To avoid this, `font+ = {⟨tokens⟩}` shall be used.

Numbers are typesetted using a upright font. The `detect-...` options of siunitx could be used to change [**siunitx**]. However, they should be passed as boolean keys.

```
\textit{It costs \dUSD{1}}.
\textit{It costs \dUSD[font=\normalfont]{1}}.
\begin{empty}
\CurrencySetup{font=\normalfont}
\textit{It costs \dUSD{1}}.
\textit{It costs \dUSD[font+=\bfseries]{1}}.
\textit{It costs \dUSD[font=\bfseries]{1}}.
\textit{It costs \dUSD[font=\bfseries,detect-weight=true]{1}}.
\textit{It costs \dUSD[font=\bfseries,detect-all=true]{1}}.
\end{empty}
```

It costs 1 *USD*. *It costs* 1 USD. *It costs* 1 USD. *It costs* 1 **USD**. *It costs* 1 ***USD***. *It costs* **1** ***USD***. *It costs* ***1*** ***USD***.

4.4.4 Using options before and after

The use of `before = {⟨tokens⟩}` is similar to the one of `font = {⟨tokens⟩}`. It is possible to append or to prepend the value to existing ones defined as a higher level through `before+ = {⟨tokens⟩}` and `before< = {⟨tokens⟩}`. Similarly, one could use `after = {⟨tokens⟩}`, `after+ = {⟨tokens⟩}` and `after< = {⟨tokens⟩}`.

```
\CurrencySetup{before=X}
\dUSD[before+={\color{red}}]{1}
\dUSD[before<={\color{red}}]{1}
```

X1 USD X1 USD

4.4.5 Using siunitx's features

Any key of the siunitx package could be used. For example, localization may change the unit separator (comma, ...).

```
This costs \dEUR{12 345.76}.
{\sisetup{locale=FR} Cela co\^ute \dEUR{12 345.76}.}
```

This costs 12 345.76 EUR. Cela coûte 12 345,76 EUR.

4.4.6 Using raw formula

A raw formula could be typeseted using the number option `number=false`. Beware, this propagate `parse-numbers=false` to `\num` so that any inner call to the latter command should specify `parse-numbers=true` if needed.

A style (see Section 4.4.7) `no-parse` is equivalent to `number=false`.

```
We get a total of \dEUR[number=false]{2\times \num[parse-numbers=true
]{10000}}=\dEUR{20000}.

We get a total of \dEUR[number=false]{2\times \num{10000}}=\dEUR
{20000}.
```

We get a total of $2 \times 10\,000$ EUR=20 000 EUR.
We get a total of 2×10000 EUR=20 000 EUR.

4.4.7 Using styles

Some styles are aldeary defined to shorten the typesetting. For example, `@iso` expands to `kind=iso`. It acts similarly for `@sy` (or `@symb`), `@na` (or `@name`) and `@pl` (or `@plural`).

```
\dEUR[@iso]{10}; \dEUR[@symb]{10}; \dEUR[@name]{10}; \dEUR[@pl]{10}

\dGBP{10} ; \dGBP[@symb]{10} ; \dGBP[@pl]{10} ; \dGBP{5}
```

10 EUR; 10 €; 10 euro; 10 euros
GBP 10 ; £ 10 ; 10 pounds ; GBP 5

Introduced in version 0.2

A style `no-parse` is also equivalent to `number=false`.

5 To Do

- Store the values to use them later.
- Automatic detection of plurals.
- Perform simple arithmetics.
- Behavior of `detect-...` keys from siunitx with default argument.
- Internationalization using the `translations` packages [**translations**].
- Non decimal systems such as pounds, shillings, and pence.
- Column definition for a table.
- ...

6 Index