The didec package

Manual for version 1.1.1 (2026/02/24)

Thomas F. Sturm[1]

https://www.ctan.org/pkg/didec

https://github.com/T-F-S/didec

Abstract

The `didec` package provides fixed-point arithmetic with two decimal places ("*di-decimal*"), as typically used in financial transactions for many currencies. Its primary intended use is (personal) bookkeeping.

Contents

[1]Prof. Dr. Dr. Thomas F. Sturm, Institut für Mathematik und Informatik, University of the Bundeswehr Munich, D-85577 Neubiberg, Germany; email: thomas.sturm@unibw.de

1 Quick start

For readers who prefer a brief summary: this package provides fixed-point arithmetic with two decimal places. It can be used in any context where exactly two decimal places are required or sufficient; its primary application, however, is (personal) bookkeeping.

Suppose John wants to keep track of his money. With

```
\didecnew{John}
```

a so-called *didec variable* is created to store currency values.

Now, lets fill in some money:

```
\didecset{John}{1000}
```

How much money does John have now?

```
\didecuse{John}
```

1.000,00 €

The amount is given in euros and formatted according to German conventions. Of course, this can be adapted to your liking, see \didecsetup → P. 13. But now, let us spend some money.

```
\didecsub{John}{19.75}
\didecuse{John}
```

980,25 €

Preferably, the package functions are used inside convenient user commands. You can choose between LaTeX2e and L3 programming layer functions.

For this quick start, we make some convenience commands for John. `\transaction` shall be one purchase made in cash and `\transaction*` one made by credit card.

```
%\usepackage{booktabs}
\didecsetup{english,
  currency          = {\pounds}{},
  currency-negative = {-\pounds}{},
}

%\didecnew{John}
\didecnew{cash}
\didecnew{credit}
\didecnew{transaction}

% Let's keep record in a table
\NewDocumentEnvironment{householdbook}{}{%
    \begin{center}
    \begin{tabular}{lp{9cm}rr}\toprule%
    & Transaction & Expenses & Budget\\\midrule
  }{%
    \midrule%
    \multicolumn{3}{r}{Cash} & \dideccoluse{cash}\\%
    \multicolumn{3}{r}{Credit card} & \dideccoluse{credit}\\\bottomrule%
    \end{tabular}%
    \end{center}
```

```
  }

% One transaction
\NewDocumentCommand \transaction{ s m m }
  {
    \didecset{transaction}{#3}%
    \didecsub{John}{transaction}%
    \IfBooleanTF {#1}%
      {\didecsub{credit}{transaction}CC}%
      {\didecsub{cash}{transaction}}%
    & #2
    & \dideccolinvuse{transaction}
    & \dideccoluse{John}\\%
  }

% Ready to start our tiny accountancy

\didecset{John}{1000}        % John's money
\didecsetequal{cash}{John}   % in cash
\didecset{credit}{0}         % blank credit card

\begin{householdbook}
  \transaction{Coffee break with snack}{19.75}
  \transaction*{Refuel}{62.87}
  \transaction{Gift from Aunt Mary for helping her}{-30}
  \transaction{Parking meter}{4.50}
  \transaction*{Shopping for weekend}{147.23}
  \transaction*{Fancy thing on the Internet}{270}
  \transaction*{Cinema}{17.70}
\end{householdbook}
```

	Transaction	Expenses	Budget
	Coffee break with snack	£19.75	£980.25
CC	Refuel	£62.87	£917.38
	Gift from Aunt Mary for helping her	-£30.00	£947.38
	Parking meter	£4.50	£942.88
CC	Shopping for weekend	£147.23	£795.65
CC	Fancy thing on the Internet	£270.00	£525.65
CC	Cinema	£17.70	£507.95
		Cash	£1,005.75
		Credit card	-£497.80

2 Didec kernels and didec expressions

All calculations are done on cent basis as integer operations, but all displayed figures have two decimal places which give the name for the package (*di-dec*imal).

The package provides two numerical kernels which can be selected mutually by package options `didec/int` or `didec/fp`. The `didec/int` kernel is faster, but provides a smaller number range, while the `didec/int` kernel is slower with a larger number range. For ordinary people like you and me, the `didec/int` kernel will suffice to do all personal financial calculations. Upgrading later from `didec/int` to `didec/fp` is a matter of just switching the package option setting.

didec/int (no value, initially set)

Selects the ⟨*int*⟩ (integer) based numerical kernel with up to 9.33[2] significant figures and fast computation. The number range for valid figures n is:

$$-21\,474\,836.47 \leq n \leq 21\,474\,836.47$$

```
\usepackage[int]{didec}
```

Using figures outside the valid range will result in LaTeX errors complaining about too large numbers.

didec/fp (no value, initially unset)

Selects the ⟨*fp*⟩ (floating-point) based numerical kernel with up to 16 significant figures and somewhat slower computation. The number range for valid figures n is:

$$-99\,999\,999\,999\,999.99 \leq n \leq 99\,999\,999\,999\,999.99$$

```
\usepackage[fp]{didec}
```

Using figures outside the valid range will result in *silent calculation errors*, because LaTeX3 ⟨*fp*⟩ can use much larger numbers but is restricted to 16 significant figures.

In the following, a ⟨*didec expr*⟩ (didec expression) denotes one of the following:

- a number in floating-point notation, e.g. `123.45`
- a number in floating-comma notation, e.g. `123,45`
- a ⟨*didec var*⟩ (didec variable), e.g. `expenses`.

Note that the notation ⟨*didec expr*⟩ is inspired by ⟨*int expr*⟩ and ⟨*fp expr*⟩ from LaTeX3 but is in comparison very restricted and allows only the three choices above. A ⟨*didec expr*⟩ is always expanded and spaces are trimmed.

Many provided commands or functions of the package come in three flavors, for example:

- `\didecadd` → P. 9: This is a user command where arguments are space trimmed and some check on variable existence is done. Not existing variables are reported by speaking error messages (not in all cases!).
- `\didec_gadd_check:nn` → P. 9: This a programming layer function with no space trimming for arguments, but some check on variable existence is done. Not existing variables are reported by speaking error messages (not in all cases!).
- `\didec_gadd:nn` → P. 9: This a programming layer function with no space trimming for arguments and no check on variable existence. Not existing variables give strange errors. This is the fasted function and base of the others above.

[2] joke for the mathematicians

The following tables compare the computation time for selected functions of the package for the two numerical kernels. Time values will differ on other computers and also depend on selected values for the examples calculations. Nevertheless, you get an impression of the differences.

kernel: `int`, engine: `pdftex`	
`\didec_gset:nn`	7 µs
`\didec_gset_check:nn`	10 µs
`\didecset`	11 µs
`\didec_gset_eq:nn`	1 µs
`\didec_gset_eq_check:nn`	6 µs
`\didecsetequal`	8 µs
`\didec_gset_fp:nn`	105 µs
`\didec_gset_fp_check:nn`	107 µs
`\didecsetfp`	110 µs
`\didec_gadd:nn`	6 µs
`\didec_gadd_check:nn`	9 µs
`\didecadd`	29 µs
`\didec_gadd_to:nnn`	11 µs
`\didec_gadd_to_check:nnn`	14 µs
`\didecadd`	51 µs
`\didec_gmul_fp:nn`	112 µs
`\didec_gmul_fp_check:nn`	115 µs
`\didecmulfp`	142 µs
`\didec_if_positive:nTF`	1 µs
`\didecifpositive`	2 µs
`\didec_compare:nNnTF`	9 µs
`\didecifgreaterthan`	10 µs
`\didec_to_fp:n`	13 µs
`\didec_to_fp_check:n`	16 µs
`\didectofp`	17 µs
`\didec_use:n`	15 µs
`\didec_use_check:n`	17 µs
`\didecuse`	37 µs
`\didecformat`	43 µs
`\didec_color_use:n`	58 µs
`\didec_color_use_check:n`	60 µs
`\dideccoluse`	84 µs
`\dideccolformat`	90 µs

kernel: `fp`, engine: `pdftex`	
`\didec_gset:nn`	38 µs
`\didec_gset_check:nn`	40 µs
`\didecset`	42 µs
`\didec_gset_eq:nn`	1 µs
`\didec_gset_eq_check:nn`	6 µs
`\didecsetequal`	8 µs
`\didec_gset_fp:nn`	128 µs
`\didec_gset_fp_check:nn`	129 µs
`\didecsetfp`	130 µs
`\didec_gadd:nn`	62 µs
`\didec_gadd_check:nn`	63 µs
`\didecadd`	87 µs
`\didec_gadd_to:nnn`	81 µs
`\didec_gadd_to_check:nnn`	84 µs
`\didecadd`	126 µs
`\didec_gmul_fp:nn`	154 µs
`\didec_gmul_fp_check:nn`	157 µs
`\didecmulfp`	181 µs
`\didec_if_positive:nTF`	16 µs
`\didecifpositive`	17 µs
`\didec_compare:nNnTF`	80 µs
`\didecifgreaterthan`	80 µs
`\didec_to_fp:n`	27 µs
`\didec_to_fp_check:n`	29 µs
`\didectofp`	31 µs
`\didec_use:n`	28 µs
`\didec_use_check:n`	31 µs
`\didecuse`	53 µs
`\didecformat`	96 µs
`\didec_color_use:n`	73 µs
`\didec_color_use_check:n`	75 µs
`\dideccoluse`	98 µs
`\dideccolformat`	142 µs

If needed, the selected kernel can be questioned by the following:

`\c_didec_kernel_str`

The current kernel is given as a lower case string: one of `int` or `fp`.

```
\ExplSyntaxOn
\c_didec_kernel_str
\ExplSyntaxOff
```

```
int
```

`\didec_if_kernel_int_p:`
`\didec_if_kernel_int:T` {⟨*true code*⟩}
`\didec_if_kernel_int:TF` {⟨*true code*⟩}{⟨*false code*⟩}
`\didec_if_kernel_fp_p:`
`\didec_if_kernel_fp:T` {⟨*true code*⟩}
`\didec_if_kernel_fp:TF` {⟨*true code*⟩}{⟨*false code*⟩}

Conditionals which allow kernel-specific code to be used. The names follow naturally from those of the kernels.

```
\ExplSyntaxOn
\didec_if_kernel_int:T { Integer~kernel~used. }\par
Floating~point~kernel
\didec_if_kernel_fp:TF { ~used. }{ ~not~used. }
\ExplSyntaxOff
```

```
Integer kernel used.
Floating point kernel not used.
```

3 Creating didec variables

\didecnew{⟨didec var⟩}
\didec_new:n{⟨didec var⟩}

Creates a new ⟨didec var⟩ or raises an error if the name is already taken. \didecnew trims spaces while \didec_new:n does not.

```
\didecnew{konto}
\didecset{konto}{99.75}
\didecuse{konto}
```

99,75 €

4 Setting didec variables

\didecset{⟨didec var⟩}{⟨didec expr⟩}
\didec_gset:nn{⟨didec var⟩}{⟨didec expr⟩}
\didec_gset_check:nn{⟨didec var⟩}{⟨didec expr⟩}

Sets ⟨didec var⟩ to the value of ⟨didec expr⟩ which can be

- a number in floating-point notation,
- a number in floating-comma notation,
- another ⟨didec var⟩.

\didecset trims spaces and performs an existence check for ⟨didec var⟩. \didec_gset_check:nn performs an existence check for ⟨didec var⟩. Decimals places 3 and beyond are cut not rounded. If rounding is an issue, use \didecsetfp → P. 8 instead.

```
\didecset{A}{1234.56}
\didecuse{A}

\didecset{A}{2345,6789}
\didecuse{A}

\didecset{B}{-3500}
\didecset{A}{B}
\didecuse{A}
```

1.234,56 €
2.345,67 €
-3.500,00 €

\didecsetequal{⟨didec var$_1$⟩}{⟨didec var$_2$⟩}
\didec_gset_eq:nn{⟨didec var$_1$⟩}{⟨didec var$_2$⟩}
\didec_gset_eq_check:nn{⟨didec var$_1$⟩}{⟨didec var$_2$⟩}

Sets the ⟨didec var$_1$⟩ to the current value of ⟨didec var$_2$⟩

```
\didecset{A}{1234.56}
\didecsetequal{B}{A}
\didecuse{A}
\didecuse{B}
```

1.234,56 € 1.234,56 €

\didecsetnegative{⟨didec var⟩}{⟨didec expr⟩}
\didec_gset_negative:nn{⟨didec var⟩}{⟨didec expr⟩}
\didec_gset_negative_check:nn{⟨didec var⟩}{⟨didec expr⟩}

Sets ⟨didec var⟩ to the negated (opposite) value of ⟨didec expr⟩. \didecsetnegative trims spaces and performs an existence check for ⟨didec var⟩.

```
\didecsetnegative{A}{1234.56}
\didecuse{A}

\didecsetnegative{A}{-42.55}
\didecuse{A}

\didecset{B}{-3500}
\didecsetnegative{A}{B}
\didecuse{A}
```

-1.234,56 €
42,55 €
3.500,00 €

\didecsetfp{⟨didec var⟩}{⟨fp expr⟩}
\didec_gset_fp:nn{⟨didec var⟩}{⟨fp expr⟩}
\didec_gset_fp_check:nn{⟨didec var⟩}{⟨fp expr⟩}

Sets ⟨didec var⟩ to the value of ⟨fp expr⟩ which can by any LaTeX3 floating-point expression. \didecsetfp trims spaces and performs an existence check for ⟨didec var⟩. Other didec variables can be used inside ⟨fp expr⟩ if guarded with \didectofp → P. 11. The result is rounded to 2 decimal places. \didec_gset_fp_check:nn performs an existence check for ⟨didec var⟩.

```
\didecsetfp{A}{2345.6789}
\didecuse{A}

\didecsetfp{A}{ ln( 12345678 ) }
\didecuse{A}

\didecset{A}{123456,78}
\didecsetfp{B}{ \didectofp{A} * 2.35 / 100 }
2.35\% of \didecuse{A} are \didecuse{B}.
```

2.345,68 €
16,33 €
2.35% of 123.456,78 € are 2.901,23 €.

\didecadd[⟨didec var⟩]{⟨didec expr$_1$⟩}{⟨didec expr$_2$⟩}
\didec_gadd:nn{⟨didec var⟩}{⟨didec expr⟩}
\didec_gadd_check:nn{⟨didec var⟩}{⟨didec expr⟩}
\didec_gadd_to:nnn{⟨didec var⟩}{⟨didec expr$_1$⟩}{⟨didec expr$_2$⟩}
\didec_gadd_to_check:nnn{⟨didec var⟩}{⟨didec expr$_1$⟩}{⟨didec expr$_2$⟩}

- Adds the result of computing the ⟨didec expr⟩ to the ⟨didec var⟩
- or sets ⟨didec var⟩ to the sum of ⟨didec expr$_1$⟩ and ⟨didec expr$_2$⟩.
- For \didecadd, if the optional ⟨didec var⟩ is not available, the sum is stored into ⟨didec expr$_1$⟩ which has to be a ⟨didec var⟩ in this case.

```
\didecset{A}{123}
\didecset{B}{5,88}
\didecadd{A}{B}
\didecuse{A}

\didecadd[A]{B}{1000}
\didecuse{A}

\didecadd{A}{-2750}
\didecuse{A}
```

128,88 €
1.005,88 €
-1.744,12 €

\didecsub[⟨didec var⟩]{⟨didec expr$_1$⟩}{⟨didec expr$_2$⟩}
\didec_gsub:nn{⟨didec var⟩}{⟨didec expr⟩}
\didec_gsub_check:nn{⟨didec var⟩}{⟨didec expr⟩}
\didec_gsub_to:nnn{⟨didec var⟩}{⟨didec expr$_1$⟩}{⟨didec expr$_2$⟩}
\didec_gsub_to_check:nnn{⟨didec var⟩}{⟨didec expr$_1$⟩}{⟨didec expr$_2$⟩}

- Subtracts the result of computing the ⟨didec expr⟩ to the ⟨didec var⟩
- or sets ⟨didec var⟩ to the difference of ⟨didec expr$_1$⟩ and ⟨didec expr$_2$⟩.
- For \didecsub, if the optional ⟨didec var⟩ is not available, the difference is stored into ⟨didec expr$_1$⟩ which has to be a ⟨didec var⟩ in this case.

```
\didecset{A}{123}
\didecset{B}{5,88}
\didecsub{A}{B}
\didecuse{A}

\didecsub[A]{B}{1000}
\didecuse{A}

\didecsub{A}{-2750}
\didecuse{A}
```

117,12 €
-994,12 €
1.755,88 €

\didecmulfp[⟨didec var2⟩]{⟨didec var⟩}{⟨fp expr⟩}
\didec_gmul_fp:nn{⟨didec var⟩}{⟨fp expr⟩}
\didec_gmul_fp_check:nn{⟨didec var⟩}{⟨fp expr⟩}
\didec_gmul_fp_to:nnn{⟨didec var2⟩}{⟨didec var⟩}{⟨fp expr⟩}
\didec_gmul_fp_to_check:nnn{⟨didec var2⟩}{⟨didec var⟩}{⟨fp expr⟩}

- Multiplies ⟨didec var⟩ with the result of computing the ⟨fp expr⟩
- and sets ⟨didec var⟩ or respectively ⟨didec var2⟩ to the result.

```
\didecset{A}{123}
\didecmulfp{A}{0.9675}
\didecuse{A}

\didecmulfp[B]{A}{ln(42)}
\didecuse{B}
```

119,00 €
444,78 €

\didecdivfp[⟨didec var2⟩]{⟨didec var⟩}{⟨fp expr⟩}
\didec_gdiv_fp:nn{⟨didec var⟩}{⟨fp expr⟩}
\didec_gdiv_fp_check:nn{⟨didec var⟩}{⟨fp expr⟩}
\didec_gdiv_fp_to:nnn{⟨didec var2⟩}{⟨didec var⟩}{⟨fp expr⟩}
\didec_gdiv_fp_to_check:nnn{⟨didec var2⟩}{⟨didec var⟩}{⟨fp expr⟩}

- Divides ⟨didec var⟩ by the result of computing the ⟨fp expr⟩
- and sets ⟨didec var⟩ or respectively ⟨didec var2⟩ to the result.

```
\didecset{A}{123}
\didecdivfp{A}{0.9675}
\didecuse{A}

\didecdivfp[B]{A}{ln(42)}
\didecuse{B}
```

127,13 €
34,01 €

\didecsetsum[⟨didec var⟩]{⟨sum of didec exp⟩}

Sets ⟨didec var⟩ to the result of computing the given ⟨sum of didec exp⟩.
Here, ⟨sum of didec exp⟩ = ⟨didec exp1⟩ + ⟨didec exp2⟩ + . . . + ⟨didec expn⟩

```
\didecset{A}{123}
\didecset{B}{-32.15}
\didecsetsum{A}{ A + B + -22.5 }
\didecuse{A}
```

68,35 €

5 Using didecs

\didectoint{⟨didec var⟩}
\didec_to_int:n{⟨didec var⟩}
\didec_to_int_check:n{⟨didec var⟩}

Expresses the ⟨didec var⟩ as Cent integer value, i.e. 100 times the value. All functions are expandable.

```
\didecset{A}{27123.45}
\didectoint{A}

\didecset{A}{-17}
\didectoint{A}
```

```
2712345
-1700
```

\didectofp{⟨didec var⟩}
\didec_to_fp:n{⟨didec var⟩}
\didec_to_fp_check:n{⟨didec var⟩}

Expresses the ⟨didec var⟩ as floating-point value. All functions are expandable.

```
\didecset{A}{27123.45}
\didectofp{A}

\didecset{A}{-17}
\didectofp{A}
```

```
27123.45
-17.00
```

\didectofc{⟨didec var⟩}
\didec_to_fc:n{⟨didec var⟩}
\didec_to_fc_check:n{⟨didec var⟩}

Expresses the ⟨didec var⟩ as floating-comma value. All functions are expandable.

```
\didecset{A}{27123.45}
\didectofc{A}

\didecset{A}{-17}
\didectofc{A}
```

```
27123,45
-17,00
```

`\didecuse[⟨key list⟩]{⟨didec var⟩}`
`\didec_use:n{⟨didec var⟩}`
`\didec_use_check:n{⟨didec var⟩}`

Expresses the ⟨*didec var*⟩ as formatted value. With `\didecsetup` → P. 13, the standard format can be set. This standard format can be overwritten by ⟨*key list*⟩.

```
\didecset{A}{123456.78}
\didecuse{A}

\didecuse[ english, currency={\pounds}{} ]{A}
```

123.456,78 €
£123,456.78

`\dideccoluse[⟨key list⟩]{⟨didec var⟩}`
`\didec_color_use:n{⟨didec var⟩}`
`\didec_color_use_check:n{⟨didec var⟩}`

Expresses the ⟨*didec var*⟩ as colorized formatted value. With `\didecsetup` → P. 13, the standard format can be set. This standard format can be overwritten by ⟨*key list*⟩.

```
\didecset{A}{123456.78}
\dideccoluse{A}

\didecset{A}{-125}
\dideccoluse{A}

\dideccoluse[color-negative=didec-blue]{A}
```

123.456,78 €
-125,00 €
-125,00 €

`\dideccolinvuse[⟨key list⟩]{⟨didec var⟩}`
`\didec_color_inverse_use:n{⟨didec var⟩}`
`\didec_color_inverse_use_check:n{⟨didec var⟩}`

Expresses the ⟨*didec var*⟩ as colorized formatted value. The coloring is switched between positive and negative. The standard coloring and format can be overwritten by ⟨*key list*⟩.

```
\didecset{A}{123456.78}
\dideccolinvuse{A}

\didecset{A}{-125}
\dideccolinvuse{A}
```

123.456,78 €
-125,00 €

`\didecformat[`⟨*key list*⟩`]{`⟨*didec expr*⟩`}`
`\dideccolformat[`⟨*key list*⟩`]{`⟨*didec expr*⟩`}`
`\dideccolinvformat[`⟨*key list*⟩`]{`⟨*didec expr*⟩`}`

Like `\didecuse` → P. 12, `\dideccoluse` → P. 12, `\dideccolinvuse` → P. 12, but accepts a ⟨*didec expr*⟩ instead of a ⟨*didec var*⟩. If the ⟨*didec expr*⟩ is a ⟨*didec var*⟩, `\didecuse` → P. 12, `\dideccoluse` → P. 12, `\dideccolinvuse` → P. 12 are more efficient.

```
\didecformat{123456.78}\\
\didecformat{A}\\
\didecformat[english]{123456.78}\\
\dideccolformat{123456.78}\\
\dideccolinvformat{123456.78}
```

123.456,78 €
-125,00 €
123,456.78 €
123.456,78 €
123.456,78 €

`\didecsetup{`⟨*key list*⟩`}`

Sets all keys of the given ⟨*key list*⟩. See the following documentation for available settings.

```
\didecsetup{
    currency           = {\pounds}{},
    decimal-separator  = {.},
    grouping-separator = {,},
}
\didecset{A}{123456.78}
\didecuse{A}
```

£123,456.78

didec/decimal-separator={⟨*separator*⟩} (initially ,)

Sets some ⟨*separator*⟩ as decimal separator.

```
\didecset{A}{123456.78}
\didecuse[ decimal-separator={\#} ]{A} \par
```

123.456#78 €

didec/grouping-separator={⟨*separator*⟩} (initially .)

Sets some ⟨*separator*⟩ as grouping separator.

```
\didecset{A}{123456.78}
\didecuse[ grouping-separator={'} ]{A}
```

123’456,78 €

U 2026-02-20 **didec/currency**={⟨*prefix*⟩}{⟨*postfix*⟩} (initially {}{\;€})

Sets some ⟨*prefix*⟩ and ⟨*postfix*⟩ to denote the currency of the didec variable. This also sets didec/currency-negative{⟨*prefix*⟩-}{⟨*postfix*⟩}

```
\didecset{A}{123456.78}
\didecuse[ currency = {\pounds}{} ]{A} \par
\didecuse[ currency = {}{\:Gulden} ]{A}
```

£123.456,78
123.456,78 Gulden

U 2026-02-20 **didec/currency-negative**={⟨*prefix*⟩}{⟨*postfix*⟩} (initially {-}{\;€})

Sets some ⟨*prefix*⟩ and ⟨*postfix*⟩ to denote the currency of the didec variable, if the resulting value is negative. Otherwise, the settings of didec/currency are used. Note that you need to set a minus sign - explicitly, if you want to see it. Also note that setting didec/currency overwrites values given by didec/currency-negative.

```
\didecset{A}{-123456.78}
\didecuse{A}\par
\didecuse[ currency-negative = {$-$}{\;€} ]{A} \par
\didecuse[ currency-negative = {(}{\;€)} ]{A} \par
\didecuse[ currency-negative = {(}{)\;€} ]{A} \par
\didecuse[ currency-negative = {-€}{} ]{A} \par
```

-123.456,78 €
−123.456,78 €
(123.456,78 €)
(123.456,78) €
-€123.456,78

U 2026-02-20 **didec/german** (style)
U 2026-02-20 **didec/english** (style)
U 2026-02-20 **didec/french** (style)
U 2026-02-20 **didec/float** (style)

Styles to set some format preferences combined. Note that \didectofp → P. 11 is more efficient than \didecuse → P. 12 with style didec/float.

	german	english	french	float
decimal-separator	,	.	,	.
grouping-separator	.	,	\;	

```
\didecset{A}{12345678.90}
\didecuse[german]{A}  \par
\didecuse[english]{A} \par
\didecuse[french]{A}  \par
\didecuse[float]{A}   \par
```

12.345.678,90 €
12,345,678.90 €
12 345 678,90 €
12345678.90 €

didec/color-positive=⟨*positive color*⟩ (initially didec-green)
didec/color-negative=⟨*negative color*⟩ (initially didec-red)

Sets ⟨*positive color*⟩ to denote positive (and zero) values and ⟨*negative color*⟩ to denote negative values. Any valid l3color ⟨*color expression*⟩ can be used. The package defines additional colors

- didec-green
- didec-red
- didec-blue

```
\didecset{A}{123456.78}
\dideccoluse[ color-positive = magenta ]{A}
```

123.456,78 €

\didecwrite{⟨*didec var*⟩}{⟨*stream*⟩}
\didec_write:nn{⟨*didec var*⟩}{⟨*stream*⟩}
\didec_write_check:nn{⟨*didec var*⟩}{⟨*stream*⟩}

Writes \didecset → P. 7{⟨*didec var*⟩}{⟨*current value*⟩} to the given already opened output ⟨*stream*⟩.

```
\didecwrite{A}{output}
% writes to output:
% \didecset{A}{VALUE}
```

6 Didec conditionals

\didecifpositive{⟨*didec var*⟩}{⟨*true code*⟩}{⟨*false code*⟩}
\didec_if_positive_p:n{⟨*didec var*⟩}
\didec_if_positive:nTF{⟨*didec var*⟩}{⟨*true code*⟩}{⟨*false code*⟩}
\didec_if_positive:nT{⟨*didec var*⟩}{⟨*true code*⟩}
\didec_if_positive:nF{⟨*didec var*⟩}{⟨*false code*⟩}

Evaluates the ⟨*didec var*⟩ and returns `true` or executes the ⟨*true code*⟩ if the value is positive, otherwise returns `false` or executes the ⟨*false code*⟩.

```
\didecset{A}{2799.50}
\didecuse{A} is \didecifpositive{A}{positive}{not positive}.

\didecset{B}{-584}
\didecuse{B} is \didecifpositive{B}{positive}{not positive}.

\didecset{A}{0}
\didecuse{A} is \didecifpositive{A}{positive}{not positive}.
```

2.799,50 € is positive.
-584,00 € is not positive.
0,00 € is not positive.

\didecifnegative{⟨*didec var*⟩}{⟨*true code*⟩}{⟨*false code*⟩}
\didec_if_negative_p:n{⟨*didec var*⟩}
\didec_if_negative:nTF{⟨*didec var*⟩}{⟨*true code*⟩}{⟨*false code*⟩}
\didec_if_negative:nT{⟨*didec var*⟩}{⟨*true code*⟩}
\didec_if_negative:nF{⟨*didec var*⟩}{⟨*false code*⟩}

Evaluates the ⟨*didec var*⟩ and returns `true` or executes the ⟨*true code*⟩ if the value is negative, otherwise returns `false` or executes the ⟨*false code*⟩.

```
\didecset{A}{2799.50}
\didecuse{A} is \didecifnegative{A}{negative}{not negative}.

\didecset{B}{-584}
\didecuse{B} is \didecifnegative{B}{negative}{not negative}.

\didecset{A}{0}
\didecuse{A} is \didecifnegative{A}{negative}{not negative}.
```

2.799,50 € is not negative.
-584,00 € is negative.
0,00 € is not negative.

\didecifzero{⟨didec var⟩}{⟨true code⟩}{⟨false code⟩}
\didec_if_zero_p:n{⟨didec var⟩}
\didec_if_zero:nTF{⟨didec var⟩}{⟨true code⟩}{⟨false code⟩}
\didec_if_zero:nT{⟨didec var⟩}{⟨true code⟩}
\didec_if_zero:nF{⟨didec var⟩}{⟨false code⟩}

Evaluates the ⟨didec var⟩ and returns `true` or executes the ⟨true code⟩ if the value is zero, otherwise returns `false` or executes the ⟨false code⟩.

```
\didecset{A}{2799.50}
\didecuse{A} is \didecifzero{A}{zero}{not zero}.

\didecset{B}{-584}
\didecuse{B} is \didecifzero{B}{zero}{not zero}.

\didecset{A}{0}
\didecuse{A} is \didecifzero{A}{zero}{not zero}.
```

2.799,50 € is not zero.
-584,00 € is not zero.
0,00 € is zero.

\dideciflowerthan{⟨didec expr$_1$⟩}{⟨didec expr$_2$⟩}{⟨true code⟩}{⟨false code⟩}
\didecifequal{⟨didec expr$_1$⟩}{⟨didec expr$_2$⟩}{⟨true code⟩}{⟨false code⟩}
\didecifgreaterthan{⟨didec expr$_1$⟩}{⟨didec expr$_2$⟩}{⟨true code⟩}{⟨false code⟩}
\didec_compare_p:nNn{⟨didec expr$_1$⟩}⟨relation⟩{⟨didec expr$_2$⟩}
\didec_compare:nNnTF{⟨didec expr$_1$⟩}⟨relation⟩{⟨didec expr$_2$⟩}{⟨true code⟩}{⟨false code⟩}
\didec_compare:nNnT{⟨didec expr$_1$⟩}⟨relation⟩{⟨didec expr$_2$⟩}{⟨true code⟩}
\didec_compare:nNnF{⟨didec expr$_1$⟩}⟨relation⟩{⟨didec expr$_2$⟩}{⟨false code⟩}

Compares the ⟨didec expr$_1$⟩ and the ⟨didec expr$_2$⟩, and returns `true` or executes the ⟨true code⟩ if the ⟨relation⟩ (given by function respectively) is obeyed, otherwise returns `false` or executes the ⟨false code⟩.

```
\didecset{A}{2799.50}
\didecset{B}{-584}
\didecuse{A} is \dideciflowerthan{A}{B}{lower}{not lower} than \didecuse{B}

\didecuse{A} is \didecifequal{A}{B}{equal}{not equal} to \didecuse{B}

\didecuse{A} is \didecifequal{A}{A}{equal}{not equal} to \didecuse{A}

\didecuse{A} is \didecifgreaterthan{A}{B}{greater}{not greater} than
↪  \didecuse{B}
```

2.799,50 € is lower than -584,00 €
2.799,50 € is equal to -584,00 €
2.799,50 € is equal to 2.799,50 €
2.799,50 € is greater than -584,00 €

7 Viewing didecs

`\didec_show:n{`⟨*didec var*⟩`}`

Displays the content of ⟨*didec var*⟩ in the terminal.

```
\didecset{A}{2799.50}
\ExplSyntaxOn
\didec_show:n{A}
\ExplSyntaxOff
```

Index