texdimens 1.1

Jean-François Burnol

2021/11/17

Contents

Copyright and License

Usage

Utilities and documentation related to TEX dimensional units, usable:

- with Plain TEX: `\input texdimens`
- with LATEX: `\usepackage{texdimens}`

For reporting issues, use the package repository.

Aim of this package

The aim of this package is to provide facilities to express dimensions (or dimension expressions evaluated by `\dimexpr`) using the various available TeX units, to the extent possible.

Macros of this package (summary)

This package provides expandable macros:

- `\texdimenUU` with `UU` standing for one of `pt`, `bp`, `cm`, `mm`, `in`, `pc`, `cc`, `nc`, `dd` and `nd`,
- `\texdimenUUup` and `\texdimenUUdown` with `UU` as above except `pt`,
- `\texdimenbothincm` and relatives,
- `\texdimenbothbpmm` and relatives,
- `\texdimenwithunit`.

`\texdimenbp` takes on input some dimension or dimension expression and produces on output a decimal `D` such that `D bp` is guaranteed to be the same dimension as the input, *if* it admits any representation as `E bp`; else it will be either the closest match from above or from below. For this unit, as well as for `nd` and `dd` the difference is at most `1sp`. For other units (not `pt` of course) the distance will usually be larger than `1sp` and one does not know if the approximant from the other direction would have been better or worst.

The variants `\texdimenbpup` and `\texdimenbpdown` expand slightly less fast than `\texdimenbp` but they allow to choose the direction of approximation (in absolute value).

The macros associated to the other units have the same descriptions.

`\texdimenbothincm`, respectively `\texdimenbothbpmm`, find the largest (in absolute value) dimension not exceeding the input and exactly representable both with the `in` and `cm` units, respectively exactly representable both with the `bp` and `mm` units.

`\texdimenwithunit{<dimen1>}{<dimen2>}` produces a decimal `D` such that `D \dimexpr dimen2\relax` is parsed by TeX into the same dimension as `dimen1` if this is at all possible. If `dimen2<1pt` all TeX dimensions `dimen1` are attainable. If `dimen2>1pt` not all `dimen1` are attainable. If not attainable, the decimal `D` will ensure a closest match from below or from above but one does not know if the approximation from the other direction is better or worst.

In a sense, this macro divides `<dimen1>` by `<dimen2>`, see additional details in the complete macro description.

Quick review of basics: TEX points and scaled points

TEX dimensions are represented internally by a signed integer which is in absolute value at most `0x3FFFFFFF`, i.e. `1073741823`. The corresponding unit is called the "scaled point", i.e. `1sp` is `1/65536` of one TEX point `1pt`, or rather `1pt` is represented internally as `65536`.

If `\foo` is a dimen register:

- `\number\foo` produces the integer `N` such as `\foo` is the same as `Nsp`,
- inside `\numexpr`, `\foo` is replaced by `N`,
- `\the\foo` produces a decimal `D` (with at most five places) followed with `pt` (catcode 12 tokens) and this output `Dpt` can serve as input in a dimen assignment to produce the same dimension as `\foo`. One can also use the catcode 11 characters `pt` for this. Digits and decimal mark must have their standard catcode 12.

When TEX encounters a dimen denotation of the type `Dpt` it will compute `N` in a way equivalent to `N = round(65536 D)` where ties are rounded away from zero. Only 17 decimal places of `D` are kept as it can be shown that going beyond can not change the result.

When `\foo` has been assigned as `Dpt`, `\the\foo` will produce some `Ept` where `E` is not necessarily the same as `D`. But it is guaranteed that `Ept` defines the same dimension as `Dpt`.

Further units known to TEX on input

TEX understands on input further units: `bp`, `cm`, `mm`, `in`, `pc`, `cc`, `nc`, `dd` and `nd`. It also understands font-dependent units `ex` and `em`, and PDFTEX adds the `px` dimension unit. Japanese engines also add specific units.

The `ex`, `em`, and `px` units are handled somewhat differently by (pdf)TEX than `bp`, `cm`, `mm`, `in`, `pc`, `cc`, `nc`, `dd` and `nd` units. For the former (let's use the generic notation `uu`), the exact same dimensions are obtained from an input `D uu` where `D` is some decimal or from `D <dimen>` where `<dimen>` stands for some dimension register which records `1uu` or `\dimexpr1uu\relax`. In contrast, among the latter, i.e. the core TEX units, this is false except for the `pc` unit.

TEX associates (explicitly for the core units, implicitly for the units corresponding to internal dimensions) to each unit `uu` a fraction `phi` which is a conversion factor. For the internal dimensions `ex`, `em`, `px` or in the case of multiplying a dimension by a decimal, this `phi` is morally `f/65536` where `f` is the integer such that `1 uu=f sp`. For core units however, the hard-coded ratio `n/d` never has a denominator `d` which is a power of `2`, except for the `pc` whose associated ratio factor is `12/1` (and arguably for the `sp` for which morally `phi` is `1/65536` but we keep it separate from the general discussion; as well as `pt` with its unit conversion factor).

Here is a table with the hard-coded conversion factors:

uu	phi	reduced	real approximation (Python output)	1uu in sp= [65536phi]	\the<1uu>
--	----------	-------	------------------	---------	----------
bp	7227/7200	803/800	1.00375	65781	1.00374pt
nd	685/642	same	1.0669781931464175	69925	1.06697pt
dd	1238/1157	same	1.070008643042351	70124	1.07pt
mm	7227/2540	same	2.8452755905511813	186467	2.84526pt
pc	12/1	12	12.0	786432	12.0pt
nc	1370/107	same	12.80373831775701	839105	12.80373pt
cc	14856/1157	same	12.84010371650821	841489	12.8401pt
cm	7227/254	same	28.45275590551181	1864679	28.45274pt
in	7227/100	same	72.27	4736286	72.26999pt

The values of `1uu` in the `sp` and `pt` units are irrelevant and even misleading regarding the TEX parsing of `D uu` input. Notice for example that `\the\dimexpr1bp\relax` gives `1.00374pt` but the actual conversion factor is `1.00375` (and `1.00375pt=65782sp>1bp`...). Similarly `\the\dimexpr1in\relax` outputs `72.26999pt` and is represented internally as `4736286sp` but the actual conversion factor is `72.27=7227/100`, and `72.27pt=4736287sp>1in`... And for the other units except the `pc`, the conversion factors are not decimal numbers, so even less likely to match `\the<1uu>` as listed in the last column. Their denominators are not powers of `2` so they don't match exactly either `(1uu in sp)/65536` but are only close.

When TEX parses an assignment `U uu` with a decimal `U` and a unit `uu`, be it a core unit, or a unit corresponding to an internal dimension, it first handles `U` as with the `pt` unit. This means that it computes `N = round(65536*U)`. It then multiplies this `N` by the conversion factor `phi` and truncates towards zero the mathematically exact result to obtain an integer `T`: `T=trunc(N*phi)`. The assignment `Uuu` is concluded by defining the value of the dimension to be `Tsp`.

Regarding the core units, we always have `phi>1`. The increasing sequence `0<=trunc(phi)<=trunc(2phi)<=...` is thus *strictly increasing* and, as `phi` is never astronomically close to `1`, **it always has jumps**: not all TEX dimensions can be obtained from an assignment using a core unit distinct from the `pt` (and `sp` of course, but we already said it was kept out of the discussion here).

On the other hand when `phi<1`, then the sequence `trunc(N phi)` is not strictly increasing, already because `trunc(phi)=0` and besides here `phi=f/65536`, so the `65536` integers `0..65535` are mapped to `f` integers `0..(f-1)` inducing non one-to-oneness. But all integers in the `0..(2**30-1)` range will be attained for some input, so there is surjectivity.

The “worst” unit is the largest i.e. the `in` whose conversion factor is `72.27`. The simplest unit to understand is the `pc` as it corresponds to an integer ratio `12`: only dimensions which in scaled points are multiple of `12` are exactly representable in the `pc` unit.

This also means that some dimensions expressible in one unit may not be available with another unit. For example, and perhaps surprisingly, there is no decimal `D` which would achieve `1in==Dcm`: the “step” between attainable dimensions is `72--73sp` for the `in` and `28--29sp` for the `cm`, and as `1in` differs internally from `2.54cm` by only `12sp` it is impossible to adjust either the `in` side or the `cm` side to obtain equality.

In particular `1in==2.54cm` is **false** in TEX, but it is true that `100in==254cm`... (it is already true that `50in==127cm`). It is also false that `10in==25.4cm` but it is true that `10in==254mm`... It is false though that `1in==25.4mm`!

```
>>> (\dimexpr1in, \dimexpr2.54cm);
@_1     4736286, 4736274
>>> (\dimexpr10in, \dimexpr25.4cm);
@_2     47362867, 47362855
>>> (\dimexpr100in, \dimexpr254cm);
@_3     473628672, 473628672

>>> (\dimexpr1in, \dimexpr25.4mm);
@_4     4736286, 4736285
>>> (\dimexpr10in, \dimexpr254mm);
@_5     47362867, 47362867
```

`\maxdimen` can be expressed only with `pt`, `bp`, and `nd`. For the other core units the maximal attainable dimensions in `sp` unit are given in the middle column of the next table.

```
maximal allowed      the corresponding      minimal \TeX{} dimen denotation
(with 5 places)   maximal attainable dim.   causing "Dimension too large"
---------------   -------------------------  --------------------------
16383.99999pt     1073741823sp (=\maxdimen)  16383.99999237060546875pt
16322.78954bp     1073741823sp (=\maxdimen)  16322.78954315185546875bp
15355.51532nd     1073741823sp (=\maxdimen)  15355.51532745361328125nd
15312.02584dd     1073741822sp               15312.02584075927734375dd
 5758.31742mm     1073741822sp                5758.31742095947265625mm
 1365.33333pc     1073741820sp                1365.33333587646484375pc
 1279.62627nc     1073741814sp                1279.62627410888671875nc
 1276.00215cc     1073741821sp                1276.00215911865234375cc
  575.83174cm     1073741822sp                 575.83174896240234375cm
  226.70540in     1073741768sp                 226.70540618896484375in
```

Perhaps for these various peculiarities with dimensional units, TEX does not provide an output facility for them similar to what `\the` achieves for the `pt`.

Macros of this package (full list)

This project requires the `\dimexpr`, `\numexpr` ε-TeX extensions. It also requires the `\expanded` primitive (available in all engines since TeXLive 2019).

The macros provided by the package are all expandable, even f-expandable. They parse their arguments via `\dimexpr` so can be nested (with appropriate units added, as the outputs always are bare decimal numbers).

The notation `<dim. expr.>` in the macro descriptions refers to a *dimensional expression* as accepted by `\dimexpr`. The syntax has some peculiarities: among them the fact that `-(...)` (for example `-(3pt)`) is illegal, one must use alternatives such as `0pt-(...)` or a sub-expression `-\dimexpr...\relax` for example.

Negative dimensions behave as if replaced by their absolute value, then at last step the sign (if result is not zero) is applied (so "down" means "towards zero", and "up" means "away from zero").

Remarks about "Dimension too large" issues:

1. For input `X` equal to `\maxdimen` (or differing by a few `sp`'s) and those units `uu` for which `\maxdimen` is not exactly representable (i.e. all core units except `pt`, `bp` and `nd`), the output `D` of the "up" macros `\texdimen<uu>up{X}`, if used as `Duu` in a dimension assignment or expression, will (as is logical) trigger a "Dimension too large" error.
2. For `dd`, `nc` and `in`, it turns out that `\texdimen<uu>{X}` chooses the "up" approximant for `X` equal to or very near `\maxdimen` (check the respective macro documentations), i.e. the output `D` is such that `Duu` is the first virtually attainable dimension beyond `\maxdimen`. Hence `Duu` will trigger on use a "Dimension too large error". With the other units for which `\maxdimen` is not attainable exactly, `\texdimen<uu>{\maxdimen}` output is by luck the "down" approximant.
3. Similarly the macro `\texdimenwithunit{D1pt}{D2pt}` covers the entire dimension range, but its output `F` for `D1pt` equal to or very close to `\maxdimen` may be such that `F<D2pt>` represents a dimension beyond `\maxdimen`, if the latter is not exactly representable. Hence `F<D2pt>` would trigger "Dimension too large" on use. This can only happen if `D2pt>1pt` and (roughly) `D1pt>\maxdimen-D2sp`. As `D2sp` is less than `0.25pt`, this is not likely to occur in real life practice except if deliberately targeting `\maxdimen`. For `D2pt<1pt`, all dimensions `D1pt` are exactly representable, in particular `\maxdimen`, and the output `F` will always be such that TeX parses `F<D2pt>` into exactly the same dimension as `D1pt`.

\texdimenpt{<dim. expr.>}

Does `\the\dimexpr <dim. expr.> \relax` then removes the `pt`.

\texdimenbp{<dim. expr.>}

Produces a decimal (with up to five decimal places) `D` such that `Dbp` represents the dimension exactly if possible. If not possible it will differ by `1sp` from the original dimension, but it is not known in advance if it will be above or below.

`\maxdimen` on input produces `16322.78954` and indeed is realized as `16322.78954bp`.

\texdimenbpdown{<dim. expr.>}

Produces a decimal (with up to five decimal places) `D` such that `Dbp` represents the dimension exactly if possible. If not possible it will be smaller by `1sp` from the original dimension.

\texdimenbpup{<dim. expr.>}

Produces a decimal (with up to five decimal places) `D` such that `Dbp` represents the dimension exactly if possible. If not possible it will be larger by `1sp` from the original dimension.

\texdimennd{<dim. expr.>}

Produces a decimal (with up to five decimal places) `D` such that `Dnd` represents the dimension exactly if possible. If not possible it will differ by `1sp` from the original dimension, but it is not known in advance if it will be above or below.

`\maxdimen` on input produces `15355.51532` and indeed is realized as `15355.51532nd`.

\texdimennddown{<dim. expr.>}

Produces a decimal (with up to five decimal places) `D` such that `Dnd` represents the dimension exactly if possible. If not possible it will be smaller by `1sp` from the original dimension.

\texdimenndup{<dim. expr.>}

Produces a decimal (with up to five decimal places) `D` such that `Dnd` represents the dimension exactly if possible. If not possible it will be larger by `1sp` from the original dimension.

`\texdimendd{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Ddd` represents the dimension exactly if possible. If not possible it will differ by `1sp` from the original dimension, but it is not known in advance if it will be above or below.

Warning: the output for `\maxdimen` is `15312.02585` but `15312.02585dd` will trigger on use “Dimension too large” error. `\maxdimen-1sp` is the maximal input for which the output remains less than `\maxdimen` (max attainable dimension: `\maxdimen-1sp`).

`\texdimendddown{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Ddd` represents the dimension exactly if possible. If not possible it will be smaller by `1sp` from the original dimension.

`\texdimenddup{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Ddd` represents the dimension exactly if possible. If not possible it will be larger by `1sp` from the original dimension.

If input is `\maxdimen`, then `Ddd` virtually represents `\maxdimen+1sp` and will trigger on use “Dimension too large”.

`\texdimenmm{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dmm` represents the dimension exactly if possible. If not possible it will either be the closest from below or from above, but it is not known in advance which one (and it is not known if the other choice would have been closer).

`\maxdimen` as input produces on output `5758.31741` and indeed the maximal attainable dimension is `5758.31741mm` (`\maxdimen-1sp`).

`\texdimenmmdown{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dmm` represents the dimension exactly if possible. If not possible it will be largest representable dimension smaller than the original one.

`\texdimenmmup{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dmm` represents the dimension exactly if possible. If not possible it will be smallest representable dimension larger than the original one.

If input is `\maxdimen`, then `Dmm` virtually represents `\maxdimen+2sp` and will trigger on use “Dimension too large”.

`\texdimenpc{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dpc` represents the dimension exactly if possible. If not possible it will be the closest representable one (in case of tie, the approximant from above is chosen).

`\maxdimen` as input produces on output `1365.33333` and indeed the maximal attainable dimension is `1365.33333pc` (`\maxdimen-3sp`).

`\texdimenpcdown{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dpc` represents the dimension exactly if possible. If not possible it will be largest representable dimension smaller than the original one.

`\texdimenpcup{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dpc` represents the dimension exactly if possible. If not possible it will be smallest representable dimension larger than the original one.

If input is `>\maxdimen-3sp`, then `Dpc` virtually represents `\maxdimen+9sp` and will trigger on use “Dimension too large”.

`\texdimennc{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dnc` represents the dimension exactly if possible. If not possible it will either be the closest from below or from above, but it is not known in advance which one (and it is not known if the other choice would have been closer).

Warning: the output for `\maxdimen-1sp` is `1279.62628` but `1279.62628nc` will trigger on use “Dimension too large” error. `\maxdimen-2sp` is the maximal input for which the output remains less than `\maxdimen` (max attainable dimension: `\maxdimen-9sp`).

`\texdimenncdown{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dnc` represents the dimension exactly if possible. If not possible it will be largest representable dimension smaller than the original one.

`\texdimenncup{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dnc` represents the dimension exactly if possible. If not possible it will be smallest representable dimension larger than the original one.

If input is `>\maxdimen-9sp`, then `Dnc` virtually represents `\maxdimen+4sp` and will trigger on use "Dimension too large".

`\texdimencc{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dcc` represents the dimension exactly if possible. If not possible it will either be the closest from below or from above, but it is not known in advance which one (and it is not known if the other choice would have been closer).

`\maxdimen` as input produces on output `1276.00215` and indeed the maximal attainable dimension is `1276.00215cc` (`\maxdimen-2sp`).

`\texdimenccdown{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dcc` represents the dimension exactly if possible. If not possible it will be largest representable dimension smaller than the original one.

`\texdimenccup{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dcc` represents the dimension exactly if possible. If not possible it will be smallest representable dimension larger than the original one.

If input is `>\maxdimen-2sp`, then `Dcc` virtually represents `\maxdimen+11sp` and will trigger on use "Dimension too large".

`\texdimencm{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dcm` represents the dimension exactly if possible. If not possible it will either be the closest from below or from above, but it is not known in advance which one (and it is not known if the other choice would have been closer).

`\maxdimen` as input produces on output `575.83174` and indeed the maximal attainable dimension is `575.83174cm` (`\maxdimen-1sp`).

`\texdimencmdown{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dcm` represents the dimension exactly if possible. If not possible it will be largest representable dimension smaller than the original one.

`\texdimencmup{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dcm` represents the dimension exactly if possible. If not possible it will be smallest representable dimension larger than the original one.

If input is `\maxdimen`, then `Dcm` virtually represents `\maxdimen+28sp` and will trigger on use “Dimension too large”.

`\texdimenin{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Din` represents the dimension exactly if possible. If not possible it will either be the closest from below or from above, but it is not known in advance which one (and it is not known if the other choice would have been closer).

Warning: the output for `\maxdimen-18sp` is `226.70541` but `226.70541in` will trigger on use “Dimension too large” error. `\maxdimen-19sp` is the maximal input for which the output remains less than `\maxdimen` (max attainable dimension: `\maxdimen-55sp`).

`\texdimenindown{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Din` represents the dimension exactly if possible. If not possible it will be largest representable dimension smaller than the original one.

`\texdimeninup{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Din` represents the dimension exactly if possible. If not possible it will be smallest representable dimension larger than the original one.

If input is `>\maxdimen-55sp`, then `Din` virtually represents `\maxdimen+17sp` and will trigger on use “Dimension too large”.

`\texdimenbothcmin{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Din` is the largest dimension not exceeding the original one (in absolute value) and exactly representable both in the `in` and `cm` units.

`\texdimenbothincm{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dcm` is the largest dimension not exceeding the original one (in absolute value) and exactly representable both in the `in` and `cm` units. Thus both expressions `\texdimenbothcmin{<dim. expr.>}in` and `\texdimenbothincm{<dim. expr.>}cm` represent the same dimension.

`\texdimenbothcminpt{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dpt` is the largest dimension not exceeding the original one (in absolute value) and exactly representable both in the `in` and `cm` units. It thus represents the same dimension as the one determined by `\texdimenbothcmin` and `\texdimenbothincm`.

`\texdimenbothincmpt{<dim. expr.>}`

Alias for `\texdimenbothcminpt`.

`\texdimenbothcminsp{<dim. expr.>}`

Produces an integer (explicit digit tokens) `N` such that `Nsp` is the largest dimension not exceeding the original one in absolute value and exactly representable both in the `in` and `cm` units.

`\texdimenbothincmsp{<dim. expr.>}`

Alias for `\texdimenbothcminsp`.

`\texdimenbothbpmm{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dmm` is the largest dimension smaller (in absolute value) than the original one and exactly representable both in the `bp` and `mm` units.

`\texdimenbothmmbp{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dbp` is the largest dimension smaller (in absolute value) than the original one and exactly representable both in the `bp` and `mm` units. Thus `\texdimenbothmmbp{<dim. expr.>}bp` is the same dimension as `\texdimenbothbpmm{<dim. expr.>}mm`.

`\texdimenbothbpmmpt{<dim. expr.>}`

Produces a decimal (with up to five decimal places) `D` such that `Dpt` is the largest dimension not exceeding the original one and exactly representable both in the `bp` and `mm` units.

`\texdimenbothmmbppt{<dim. expr.>}`

Alias for `\texdimenbothbpmmpt`.

`\texdimenbothbpmmsp{<dim. expr.>}`

Produces an integer (explicit digit tokens) `N` such that `Nsp` is the largest dimension not exceeding the original one and exactly representable both in the `bp` and `mm` units.

`\texdimenbothmmbpsp{<dim. expr.>}`

Alias for `\texdimenbothbpmmsp`.

`\texdimenwithunit{<dim. expr. 1>}{<dim expr. 2>}`

Produces a decimal `D` such that `D\dimexpr <dim expr. 2>\relax` is considered by TeX the same as `<dim. expr.` 1> if at all possible. If the (assumed non zero) second argument <dim2> is at most `1pt` (in absolute value), then this is always possible. If the second argument <dim2> is >`1pt` then this is not always possible and the output `D` will ensure for `D`<dim2> to be a closest match to the first argument `dim1` either from above or below, but one does not know if the other direction would have given a better or worst match.

`\texdimenwithunit{<dim>}{1bp}` and `\texdimenbp{<dim>}` are not the same: The former produces a decimal `D` such that `D\dimexpr 1bp\relax` is represented internally as is <dim> if at all possible, whereas the latter produces a decimal `D` such that `D bp` is the one aiming at being the same as <dim>. Using `D\dimexpr 1bp\relax` implies a conversion factor equal to `65781/65536`, whereas `D bp` involves the `803/800` conversion factor.

`\texdimenwithunit{D1pt}{D2pt}` output is close to the mathematical ratio `D1/D2`. But notwithstanding the various unavoidable "errors" arising from conversion of decimal inputs to binary internals, and from the latter to the former, the output `R` will tend to be on average slightly larger (in its last decimal) than mathematical `D1/D2`. The root cause being that the specification for `R` is that `R<D2pt>` must be exactly `<D1pt>` after TEX parsing, if at all possible; and it turns out this is always possible for `D2pt<1pt`. The final step in the TEX parsing of a multiplication of a dimension by a scalar is a *truncation* to an integer multiple of the `sp=1/65536pt` unit, not a rounding. So `R` is basically (i.e. before conversion to a decimal) `ceil(D1/D2,16)`, or to be more precise it is obtained as `ceil(N1/N2,16)` with `D1pt->N1sp`, `D2pt->N2sp` and the second argument of `ceil` means that `16` binary places are used. This formula is the one used for `D2pt<1pt`, for `D2pt>1pt` the mathematics is different, but the implication that `R` has a (less significant) bias to be "shifted upwards" (in its last decimal place) compared to the (rounded) value `D1/D2` or rather `N1/N2` still stands.

Change log

1.1 (2021/11/17)

- internal refactorings across the entire code base aiming at (small) efficiency gains from optimized TEX token manipulations
- in particular, the algorithm for `\texdimenwithunit{<dim1>}{<dim2>}` in the "`dim2<1pt`" branch got modified (output unchanged)
- all macros now f-expandable (this was already the case at `1.0` except for `\texdimenwithunit` with arguments of opposite signs, the second one not exceeding `1pt` in absolute value)
- the `\expanded` primitive is required (present in all engines since TEXLive 2019)
- the usual batch of documentation additions or fix-ups, also in code comments (fix in particular issues #21, #22)
- addition of this Change log to the pdf documentation
- addition of the highlighted commented source code to the pdf documentation

1.0 (2021/11/10)

- new: `\texdimenbothbpmm` and relatives (feature request #10)
- breaking: `\texdimenwithunit` output for second argument `<1pt` still obeys specs but is closer to mathematical ratio (feature request #16)
- enhanced: all `up/down` macros (i.e. also for the `dd`, `nc`, `in` units) accept the full range of dimensions (feature request #18)
- enhanced: `\texdimenwithunit`'s second argument is now allowed to be negative (feature request #13)

0.99a-d (2021/11/04)

- documentation in TEX/LATEX installations available in pdf format
- the usual batch of documentation additions or fix-ups
- let the CTAN `README.md` be much shortened, and provide `texdimens.md` as the one matching the repo `README.md`
- fix bugs of `\texdimenwithunit{<dim1>}{<dime2>}` for `dim1=0pt` or `dim2=1pt` (#3, #4, #6, #8)

0.99 (2021/11/02)

- new: `\texdimenwithunit{<dim1>}{<dim2>}` (feature request #2)

0.9 (2021/07/21)

- new: `\texdimenbothincm` and relatives
- breaking: use `\texdimen` prefix for all macros

0.9delta (2021/07/15)

- internal refactorings

0.9gamma (2021/07/14)

- new: `\texdiminbpdown` (now `\texdimenbpdown`), `\texdiminbpup` (now `\texdimenbpup`) and similar named macros associated with the other units

0.9beta (2021/06/30)

- initial release: provides `\texdiminbp` (now `\texdimenbp`) and similar named macros for the units `nd`, `dd`, `mm`, `pc`, `nc`, `cc`, `cm`, `in`

Acknowledgements

Thanks to Denis Bitouzé for raising an issue on the LATEX3 tracker which became the initial stimulus for this package.

Thanks to Ruixi Zhang for reviving the above linked-to thread and opening up on the package issue tracker the issue #2 asking to add handling of the `ex` and `em` cases. This was done at release `0.99` via the addition of `\texdimenwithunit`.

Renewed thanks to Ruixi Zhang for analyzing at issue #10 what is at stake into finding dimensions exactly representable both in the `bp` and `mm` units. Macros `\texdimenbothbpmm` and `\texdimenbothmmbp` now address this (release `1.0`).

Implementation

```
% This is file texdimens.tex, part of texdimens package, which
% is distributed under the LPPL 1.3c. Copyright (c) 2021 Jean-François Burnol
% 2021/11/17 v1.1
\edef\texdimensendinput{\endlinechar\the\endlinechar%
\catcode`\noexpand _=\the\catcode`\_%
\catcode`\noexpand @=\the\catcode`\@\relax\noexpand\endinput}%
\endlinechar13\relax%
% only for using \p@ (also \z@ now) of Plain. Check if \p@, \z@ exists?
\catcode`\_=11 \catcode`\@=11
% so tempted to do \input xintkernel.sty to have some utilities...
% not even a \@gobble in Plain...
\def\texdimenfirstofone#1{#1}%
\def\texdimengobtilminus#1-{}%
\def\texdimenzerominusfork #10-#2#3\krof {#2}%
%
% \texdimenuu, \texdimenuudown, \texdimenuuup
% ===========================================
%
% Mathematics
% -----------
%
% In the entire discussion here, "uu" stands for some core unit,
% or some unit corresponding to an internal dimension > 1pt.
%
% Main question at the origin of this file was:
%     Is T sp attainable from unit "uu"?.
%     If not, what is largest dimension < Tsp which is?
%
% Here we suppose T>0. TeX parsing of D uu is equivalent to:
%
% D uu --> N = round(D * 65536) --> T = trunc (N * phi)
%
% phi>1 is the conversion factor associated to "uu"
% psi=1/phi, psi<1. Define U(N, phi) = trunc (N * phi)
%
%     U(N,phi) is thus the strictly increasing sequence,
%     indexed by non-negative integers, of non-negative
%     attainable dimensions. (in sp unit)
%
% T>0, then:
%
%     U(N)<= T <  U(N+1)    iff    N = ceil((T+1)psi) - 1
%     U(M)<  T <= U(M+1)    iff    M = ceil(T psi)    - 1
%
% In other words:
%
% - the largest attainable dimension not exceeding T sp
%   is obtained via the integer "Zd = ceil((T+1)psi) - 1 = N",
%   (i.e. find D with Zd=round(65536 D) then "D uu" is "down"
%    approximation)
%
% - the smallest attainable dimension at least equal to T sp
%   is obtained from the integer "Zu = ceil(T psi) = M + 1"
%
% - the two "Z"'s are either equal (i.e. T is attained) or Zu=Zd+1.
%
% \texdimenUU macros use round((T+0.5)*psi)
% -----------------------------------------
%
% case1:  M = N, i.e. Zd<Zu, i.e. T is not attainable:
%         M=N=Zd < T psi < (T+1) psi <= N+1=Zu
%
%         Then clearly R = round((T+0.5)psi) is either Zd or Zu.
%         We will not know which one before computing trunc(R phi)
%         and check if it is < T or > T.
%
%         As will be explained later trunc(R phi) can be computed very
%         easily by hijacking TeX's handling of dimensions, no \numexpr
%         chains is needed.
%
% case2:  M = N - 1, i.e. T = Zd = Zu is attained:
%         T psi <= N < (T+1) psi, T = trunc(N phi)
%
%         Let v=(T+0.5)psi. As v = T psi + 0.5 psi it is < N+0.5
%         And as v = (T+1)psi - 0.5psi it is > N - 0.5.
%         So R = round(v) = N.
%
% We thus have the initial observation which was at the core of this
% package initial release:
%
% - compute R = round((T+0.5) psi)
%
%   - if T is attained, then T = trunc(R * phi)
%
%   - if T is not attained then either { Zd = R and Zu = R+1 }  or
%     {Zd = R-1 and Zu = R}.
%
% How do we check if R = Zd or Zu? We need to evaluate trunc(R phi) and
% compare it with T. This trunc(R phi) can be computed the following way:
%
% - obtain D pt from \the\dimexpr R sp. Knuth's algorithm guarantees
%   that R = round(D * 65536)
%
% - then D uu where uu is the unit with conversion factor phi is
%   converted by TeX into "trunc(R phi) sp", i.e.  trunc(R phi) =
%   \number\dimexpr Duu\relax, where D pt = \the\dimexpr Rsp\relax.
%
% Conclusion:
%
% 1. the macro \texdimenuu does the one-liner R=round((T+0.5) psi)
%    then \the\dimexpr Rsp\relax gives "Dpt", the "pt" is removed,
%    we have a decimal D such that "Duu" does what one wants.
%
% 2. to get Zd (resp. Zu) one can use the D obtained in 1. and check
%    if "D uu" is at most (or at least) the user input dimension.
```

```
%
% For units with conversion factor phi>2, a simplification is possible.
% In that case let X = round(T psi) (it has the advantage compared to
% R that we can apply the formula without checking the sign of T).
%
% Going back to our earlier analyis, now with psi < 0.5 (1uu>2pt)
%
% case1: T is not attainable
%        M=N=Zd < T psi < (T+1) psi <= N+1=Zu
%        As Zd < T psi < Zu, we have round(T psi) = Zd or Zu
%
% case2: T is attained, i.e. T psi <= N < (T+1) psi.
%        As psi<0.5, and T psi + psi > N, we have T psi > N - 0.5.
%        And T psi <= N so N = round(T psi).
%
% So, for psi < 0.5, the X=round(T psi) can play the same role as
% R=round((T+0.5)psi). If T is attained, we get the decimal D from this
% X and if T is not attained we know that X is either Zd or Zu.
%
% The computations of X and Y=trunc(X phi) can be done independently of
% sign of T.  But the final test has to be changed to Y < T if T < 0 and
% then one must replace X by X+1. So we must filter out the sign of the
% input.
%
% Going back to the 1<phi<2 case, psi>0.5, then it would be slightly
% less costly to compute X = round(T psi) than R = round((T + 0.5) psi),
% but if we then realize that trunc(X phi) < T we do not yet know if
% trunc((X+1) phi) = T or is > T, i.e. we don't know if Zd =X or X+1,
% and we can not tell yet if T is attained or not.
%
% In contrast if we find out that trunc(R phi)<T, we then know for sure
% that Zd=R, Zu=R+1 and that T is not attained.
%
% Problems with \maxdimen in the obtention of Zu and Zd
% -----------------------------------------------------
%
% Obtaining R = round((T+0.5)psi) has no risk of overflow.
% But checking as described above which one of Zd or Zu (or both)
% is R goes via a test computation which will cause overflow
% if by bad luck R = Zu and Zu will give rise to a decimal D
% such that D uu > \maxdimen.
%
% For T=\maxdimen (or very close) this is what happens for the units
% "dd", "nc", and "in".
%
% Besides, it turns out that this test which is done to decide whether
% R=Zu or R=Zd, and on which the initial implementation of the macros
% "up" and "down" was done at 0.9 gamma release is a bit costly.
%
% At 1.0 release, all the "up" and "down" macros were re-implemented
% via a more stubborn usage of the ceil() based formulae for Zd and Zu.
% This made all usable even with \maxdimen input and besided, proved
% on average slightly faster.
```

```
%
% Overcoming the ceil() stumbling block for \texdimenUU{up,down}
% ------------------------------------------------------------
%
% I will in what follows refer to trunc(), floor() or ceil() only for
% positive arguments, obtained as ratios x/y or sometimes as a numexpr
% "scaling" operation" x*y/z which uses temporarily use doubled
% precision.
%
% As \numexpr's x/y is round(x/y), with rounding away from zero, we have
% access to floor(t) for t>=0 as round(t+0.5)-1 and for t>0 also as
% round(t-0.5). The former may cause overflow as it involves
% (2x+y)/(2y) but the latter (2x-y)/(2y) will not overflow if x comes
% from a dimension as 2x<2**31 then.
%
% ceil(t) is more complex as it is floor(t)+1 only for t not an integer.
% Let's explain how to overcome the challenge for Zd and the "in" unit,
% i.e. a conversion factor of 7227/100.
%
% We want Zd = ceil((T+1)*100/7227) - 1, with T assumed positive.
%
% Let T = k*7227 + r with 0<= r < 7227, 0<=k, and r>0 if k=0.
%
% (T+1)*100/7227 becomes 100*k + (r+1)*100/7227 and thus
%
% Zd = 100 * k + ceil(x) - 1
%
% with  x = n*100/7227, and n = 1+r, so 0<n<=7227
%
% Here we have a nice situation 0 < x <= 100. Then:
%
% ceil(x) = 100 - floor(100 - x)
%         = 100 - (round(100 - x + 0.5) - 1)
%         = 101 - round(100 * (1 - n/7227) + 0.5)
%         = 101 - round((200 * (7227 - n) + 7227)/14454)
%
% We can thus achieve the computation of Zd = ceil((T+1)*100/7227) - 1
% for T>0 without overflow in \numexpr this way:
%
%     k = floor(T/7227) = round(T/7227 - 0.5)
%                       = round((2*T - 7227) / 14454)  (T>0 used here)
%
%     r = T - 7227 * k  = T modulo 7227
%
%     Zd = 100 * k + 100 - round( (201*7227 - 200*(r+1))/14454 )
%
% Everything here is computable within \numexpr and has absolutely no
% potential overflow problem at all. The same analysis can be done for
% Zu = ceil(T*100/7227) and for all core TeX units. See the comments
% below for all obtained formulae and some additional details.
%
{\catcode`p 12\catcode`t 12
 \csname expandafter\endcsname\gdef\csname texdimenstrippt\endcsname#1pt{#1}}%
```

```
%
% pt
%
\def\texdimenpt#1{\expandafter\texdimenstrippt\the\dimexpr#1\relax}%
%
% bp 7227/7200 = 803/800
%
\def\texdimenbp#1{\expandafter\texdimenstrippt\the\dimexpr\numexpr(%
                  \expandafter\texdimen_bpnddd_signcheck
                  \the\numexpr2*\dimexpr#1\relax\relax)*400/803sp\relax}%
\def\texdimen_bpnddd_signcheck#1{\texdimengobtilminus#1-1+#1}%
%
% nd 685/642
%
\def\texdimennd#1{\expandafter\texdimenstrippt\the\dimexpr\numexpr(%
                  \expandafter\texdimen_bpnddd_signcheck
                  \the\numexpr2*\dimexpr#1\relax\relax)*321/685sp\relax}%
%
% dd 1238/1157
%
\def\texdimendd#1{\expandafter\texdimenstrippt\the\dimexpr\numexpr(%
                  \expandafter\texdimen_bpnddd_signcheck
                  \the\numexpr2*\dimexpr#1\relax\relax)*1157/2476sp\relax}%
%
% mm 7227/2540 phi now >2, use from here on the X = round(T psi) approach
%
\def\texdimenmm#1{\expandafter\texdimenstrippt\the\dimexpr(#1)*2540/7227\relax}%
%
% pc 12/1
%
\def\texdimenpc#1{\expandafter\texdimenstrippt\the\dimexpr(#1)/12\relax}%
%
% nc 1370/107
%
\def\texdimennc#1{\expandafter\texdimenstrippt\the\dimexpr(#1)*107/1370\relax}%
%
% cc 14856/1157
%
\def\texdimencc#1{\expandafter\texdimenstrippt\the\dimexpr(#1)*1157/14856\relax}%
%
% cm 7227/254
%
\def\texdimencm#1{\expandafter\texdimenstrippt\the\dimexpr(#1)*254/7227\relax}%
%
% in 7227/100
%
\def\texdimenin#1{\expandafter\texdimenstrippt\the\dimexpr(#1)*100/7227\relax}%
%
% "up and down macros"
% --------------------
%
% The notation <u/v> means u/v in numexpr, which does rounding
% away from zero. It is essential that the argument be >-0.5 else <x+1>
% not same as <x>+1. All formulae are overflow free.
%
% The comments are for T > 0.
%
% Roughly such an approach works for phi = a/b > 1, such that:
%
%     a*(2b+1)<2**31 if a is odd, <2**32 if a is even
%
% This is true for all core units with quite some margin, the one with
% largest a*b being phi=7227/2540 for "mm".
%
% Note: for a unit such as "ex" or "em" where morally b=65536=2**16,
% this limits to a<=16383 if a is odd and to a<=32766 if a is even.
% Thus the general \texdimenwithunit{dim1}{dim2} (which for dim2<1pt
% computes basically an "up" value) can *not imitate fully* this scheme.
%
% The macros and formulas in the comments were obtained from a template
% (see file generateupdownmacros.py at the project repository),
% and we could actually combine them into a generic macro handling
% general a/b (assuming above bounds are verified).
% But for the the sake of efficiency, this is "rolled-out" here unit per unit.
%
\def\texdimenuudownup_zero#1;{\z@\relax}%
\def\texdimenuudownup_neg#1-{-#1}%
% bp 803/800
% T = 803 k + r
% Zd = 800 k + 800 - <(1284003 - 1600 r)/1606>
\def\texdimenbpdown#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenbpdown_a\the\numexpr\dimexpr#1;%
}%
\def\texdimenbpdown_a#1{\texdimenzerominusfork
                        #1-\texdimenuudownup_zero
                        0#1\texdimenuudownup_neg
                         0-{}%
                        \krof \texdimenbpdown_b#1}%
\def\texdimenbpdown_b#1;{\expandafter\texdimenbpdown_c\the\numexpr(2*#1-803)/1606;#1;}%
\def\texdimenbpdown_c#1;#2;{\expandafter\texdimenbpdown_d\the\numexpr#2-803*#1;#1;}%
\def\texdimenbpdown_d#1;#2;{\numexpr800*#2+800-(1284003-1600*#1)/1606sp\relax}%
% Zu = 800 k + 800 + 1 - <(1285603 - 1600 r)/1606>
\def\texdimenbpup#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenbpup_a\the\numexpr\dimexpr#1;%
}%
\def\texdimenbpup_a#1{\texdimenzerominusfork
                        #1-\texdimenuudownup_zero
                        0#1\texdimenuudownup_neg
                         0-{}%
                      \krof \texdimenbpup_b#1}%
\def\texdimenbpup_b#1;{\expandafter\texdimenbpup_c\the\numexpr(2*#1-803)/1606;#1;}%
\def\texdimenbpup_c#1;#2;{\expandafter\texdimenbpup_d\the\numexpr#2-803*#1;#1;}%
\def\texdimenbpup_d#1;#2;{\numexpr800*#2+801-(1285603-1600*#1)/1606sp\relax}%
% nd 685/642
% T = 685 k + r
% Zd = 642 k + 642 - <(878941 - 1284 r)/1370>
```

```
\def\texdimennddown#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimennddown_a\the\numexpr\dimexpr#1;%
}%
\def\texdimennddown_a#1{\texdimenzerominusfork
                        #1-\texdimenuudownup_zero
                        0#1\texdimenuudownup_neg
                         0-{}%
                        \krof \texdimennddown_b#1}%
\def\texdimennddown_b#1;{\expandafter\texdimennddown_c\the\numexpr(2*#1-685)/1370;#1;}%
\def\texdimennddown_c#1;#2;{\expandafter\texdimennddown_d\the\numexpr#2-685*#1;#1;}%
\def\texdimennddown_d#1;#2;{\numexpr642*#2+642-(878941-1284*#1)/1370sp\relax}%
% Zu = 642 k + 642 + 1 - <(880225 - 1284 r)/1370>
\def\texdimenndup#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenndup_a\the\numexpr\dimexpr#1;%
}%
\def\texdimenndup_a#1{\texdimenzerominusfork
                        #1-\texdimenuudownup_zero
                        0#1\texdimenuudownup_neg
                         0-{}%
                      \krof \texdimenndup_b#1}%
\def\texdimenndup_b#1;{\expandafter\texdimenndup_c\the\numexpr(2*#1-685)/1370;#1;}%
\def\texdimenndup_c#1;#2;{\expandafter\texdimenndup_d\the\numexpr#2-685*#1;#1;}%
\def\texdimenndup_d#1;#2;{\numexpr642*#2+643-(880225-1284*#1)/1370sp\relax}%
% dd 1238/1157
% T = 1238 k + r
% Zd = 1157 k + 1157 - <(1431828 - 1157 r)/1238>
\def\texdimendddown#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimendddown_a\the\numexpr\dimexpr#1;%
}%
\def\texdimendddown_a#1{\texdimenzerominusfork
                        #1-\texdimenuudownup_zero
                        0#1\texdimenuudownup_neg
                         0-{}%
                        \krof \texdimendddown_b#1}%
\def\texdimendddown_b#1;{\expandafter\texdimendddown_c\the\numexpr(#1-619)/1238;#1;}%
\def\texdimendddown_c#1;#2;{\expandafter\texdimendddown_d\the\numexpr#2-1238*#1;#1;}%
\def\texdimendddown_d#1;#2;{\numexpr1157*#2+1157-(1431828-1157*#1)/1238sp\relax}%
% Zu = 1157 k + 1157 + 1 - <(1432985 - 1157 r)/1238>
\def\texdimenddup#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenddup_a\the\numexpr\dimexpr#1;%
}%
\def\texdimenddup_a#1{\texdimenzerominusfork
                        #1-\texdimenuudownup_zero
                        0#1\texdimenuudownup_neg
                         0-{}%
                      \krof \texdimenddup_b#1}%
\def\texdimenddup_b#1;{\expandafter\texdimenddup_c\the\numexpr(#1-619)/1238;#1;}%
\def\texdimenddup_c#1;#2;{\expandafter\texdimenddup_d\the\numexpr#2-1238*#1;#1;}%
\def\texdimenddup_d#1;#2;{\numexpr1157*#2+1158-(1432985-1157*#1)/1238sp\relax}%
% mm 7227/2540
% T = 7227 k + r
% Zd = 2540 k + 2540 - <(36715307 - 5080 r)/14454>
\def\texdimenmmdown#1{\expandafter\texdimenstrippt\the\dimexpr
```

```
    \expandafter\texdimenmmdown_a\the\numexpr\dimexpr#1;%
}%
\def\texdimenmmdown_a#1{\texdimenzerominusfork
                       #1-\texdimenuudownup_zero
                       0#1\texdimenuudownup_neg
                        0-{}%
                       \krof \texdimenmmdown_b#1}%
\def\texdimenmmdown_b#1;{\expandafter\texdimenmmdown_c\the\numexpr(2*#1-7227)/14454;#1;}%
\def\texdimenmmdown_c#1;#2;{\expandafter\texdimenmmdown_d\the\numexpr#2-7227*#1;#1;}%
\def\texdimenmmdown_d#1;#2;{\numexpr2540*#2+2540-(36715307-5080*#1)/14454sp\relax}%
% Zu = 2540 k + 2540 + 1 - <(36720387 - 5080 r)/14454>
\def\texdimenmmup#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenmmup_a\the\numexpr\dimexpr#1;%
}%
\def\texdimenmmup_a#1{\texdimenzerominusfork
                       #1-\texdimenuudownup_zero
                       0#1\texdimenuudownup_neg
                        0-{}%
                     \krof \texdimenmmup_b#1}%
\def\texdimenmmup_b#1;{\expandafter\texdimenmmup_c\the\numexpr(2*#1-7227)/14454;#1;}%
\def\texdimenmmup_c#1;#2;{\expandafter\texdimenmmup_d\the\numexpr#2-7227*#1;#1;}%
\def\texdimenmmup_d#1;#2;{\numexpr2540*#2+2541-(36720387-5080*#1)/14454sp\relax}%
% pc 12/1
% T = 12 k + r
% Zd = 1 k + 1 - <(17 - 1 r)/12>
\def\texdimenpcdown#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenpcdown_a\the\numexpr\dimexpr#1;%
}%
\def\texdimenpcdown_a#1{\texdimenzerominusfork
                       #1-\texdimenuudownup_zero
                       0#1\texdimenuudownup_neg
                        0-{}%
                       \krof \texdimenpcdown_b#1}%
\def\texdimenpcdown_b#1;{\expandafter\texdimenpcdown_c\the\numexpr(#1-6)/12;#1;}%
\def\texdimenpcdown_c#1;#2;{\expandafter\texdimenpcdown_d\the\numexpr#2-12*#1;#1;}%
\def\texdimenpcdown_d#1;#2;{\numexpr#2+1-(17-#1)/12sp\relax}%
% Zu = 1 k + 1 + 1 - <(18 - 1 r)/12>
\def\texdimenpcup#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenpcup_a\the\numexpr\dimexpr#1;%
}%
\def\texdimenpcup_a#1{\texdimenzerominusfork
                       #1-\texdimenuudownup_zero
                       0#1\texdimenuudownup_neg
                        0-{}%
                     \krof \texdimenpcup_b#1}%
\def\texdimenpcup_b#1;{\expandafter\texdimenpcup_c\the\numexpr(#1-6)/12;#1;}%
\def\texdimenpcup_c#1;#2;{\expandafter\texdimenpcup_d\the\numexpr#2-12*#1;#1;}%
\def\texdimenpcup_d#1;#2;{\numexpr#2+2-(18-#1)/12sp\relax}%
% nc 1370/107
% T = 1370 k + r
% Zd = 107 k + 107 - <(147168 - 107 r)/1370>
\def\texdimenncdown#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenncdown_a\the\numexpr\dimexpr#1;%
```

```
}%
\def\texdimenncdown_a#1{\texdimenzerominusfork
                        #1-\texdimenuudownup_zero
                        0#1\texdimenuudownup_neg
                         0-{}%
                        \krof \texdimenncdown_b#1}%
\def\texdimenncdown_b#1;{\expandafter\texdimenncdown_c\the\numexpr(#1-685)/1370;#1;}%
\def\texdimenncdown_c#1;#2;{\expandafter\texdimenncdown_d\the\numexpr#2-1370*#1;#1;}%
\def\texdimenncdown_d#1;#2;{\numexpr107*#2+107-(147168-107*#1)/1370sp\relax}%
% Zu = 107 k + 107 + 1 - <(147275 - 107 r)/1370>
\def\texdimenncup#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenncup_a\the\numexpr\dimexpr#1;%
}%
\def\texdimenncup_a#1{\texdimenzerominusfork
                        #1-\texdimenuudownup_zero
                        0#1\texdimenuudownup_neg
                         0-{}%
                      \krof \texdimenncup_b#1}%
\def\texdimenncup_b#1;{\expandafter\texdimenncup_c\the\numexpr(#1-685)/1370;#1;}%
\def\texdimenncup_c#1;#2;{\expandafter\texdimenncup_d\the\numexpr#2-1370*#1;#1;}%
\def\texdimenncup_d#1;#2;{\numexpr107*#2+108-(147275-107*#1)/1370sp\relax}%
% cc 14856/1157
% T = 14856 k + r
% Zd = 1157 k + 1157 - <(17194663 - 1157 r)/14856>
\def\texdimenccdown#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenccdown_a\the\numexpr\dimexpr#1;%
}%
\def\texdimenccdown_a#1{\texdimenzerominusfork
                        #1-\texdimenuudownup_zero
                        0#1\texdimenuudownup_neg
                         0-{}%
                        \krof \texdimenccdown_b#1}%
\def\texdimenccdown_b#1;{\expandafter\texdimenccdown_c\the\numexpr(#1-7428)/14856;#1;}%
\def\texdimenccdown_c#1;#2;{\expandafter\texdimenccdown_d\the\numexpr#2-14856*#1;#1;}%
\def\texdimenccdown_d#1;#2;{\numexpr1157*#2+1157-(17194663-1157*#1)/14856sp\relax}%
% Zu = 1157 k + 1157 + 1 - <(17195820 - 1157 r)/14856>
\def\texdimenccup#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenccup_a\the\numexpr\dimexpr#1;%
}%
\def\texdimenccup_a#1{\texdimenzerominusfork
                        #1-\texdimenuudownup_zero
                        0#1\texdimenuudownup_neg
                         0-{}%
                      \krof \texdimenccup_b#1}%
\def\texdimenccup_b#1;{\expandafter\texdimenccup_c\the\numexpr(#1-7428)/14856;#1;}%
\def\texdimenccup_c#1;#2;{\expandafter\texdimenccup_d\the\numexpr#2-14856*#1;#1;}%
\def\texdimenccup_d#1;#2;{\numexpr1157*#2+1158-(17195820-1157*#1)/14856sp\relax}%
% cm 7227/254
% T = 7227 k + r
% Zd = 254 k + 254 - <(3678035 - 508 r)/14454>
\def\texdimencmdown#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimencmdown_a\the\numexpr\dimexpr#1;%
}%
```

```
\def\texdimencmdown_a#1{\texdimenzerominusfork
                        #1-\texdimenuudownup_zero
                        0#1\texdimenuudownup_neg
                         0-{}%
                        \krof \texdimencmdown_b#1}%
\def\texdimencmdown_b#1;{\expandafter\texdimencmdown_c\the\numexpr(2*#1-7227)/14454;#1;}%
\def\texdimencmdown_c#1;#2;{\expandafter\texdimencmdown_d\the\numexpr#2-7227*#1;#1;}%
\def\texdimencmdown_d#1;#2;{\numexpr254*#2+254-(3678035-508*#1)/14454sp\relax}%
% Zu = 254 k + 254 + 1 - <(3678543 - 508 r)/14454>
\def\texdimencmup#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimencmup_a\the\numexpr\dimexpr#1;%
}%
\def\texdimencmup_a#1{\texdimenzerominusfork
                        #1-\texdimenuudownup_zero
                        0#1\texdimenuudownup_neg
                         0-{}%
                      \krof \texdimencmup_b#1}%
\def\texdimencmup_b#1;{\expandafter\texdimencmup_c\the\numexpr(2*#1-7227)/14454;#1;}%
\def\texdimencmup_c#1;#2;{\expandafter\texdimencmup_d\the\numexpr#2-7227*#1;#1;}%
\def\texdimencmup_d#1;#2;{\numexpr254*#2+255-(3678543-508*#1)/14454sp\relax}%
% in 7227/100
% T = 7227 k + r
% Zd = 100 k + 100 - <(1452427 - 200 r)/14454>
\def\texdimenindown#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenindown_a\the\numexpr\dimexpr#1;%
}%
\def\texdimenindown_a#1{\texdimenzerominusfork
                        #1-\texdimenuudownup_zero
                        0#1\texdimenuudownup_neg
                         0-{}%
                        \krof \texdimenindown_b#1}%
\def\texdimenindown_b#1;{\expandafter\texdimenindown_c\the\numexpr(2*#1-7227)/14454;#1;}%
\def\texdimenindown_c#1;#2;{\expandafter\texdimenindown_d\the\numexpr#2-7227*#1;#1;}%
\def\texdimenindown_d#1;#2;{\numexpr#200+100-(1452427-2*#100)/14454sp\relax}%
% Zu = 100 k + 100 + 1 - <(1452627 - 200 r)/14454>
\def\texdimeninup#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimeninup_a\the\numexpr\dimexpr#1;%
}%
\def\texdimeninup_a#1{\texdimenzerominusfork
                        #1-\texdimenuudownup_zero
                        0#1\texdimenuudownup_neg
                         0-{}%
                      \krof \texdimeninup_b#1}%
\def\texdimeninup_b#1;{\expandafter\texdimeninup_c\the\numexpr(2*#1-7227)/14454;#1;}%
\def\texdimeninup_c#1;#2;{\expandafter\texdimeninup_d\the\numexpr#2-7227*#1;#1;}%
\def\texdimeninup_d#1;#2;{\numexpr#200+101-(1452627-2*#100)/14454sp\relax}%
%
% "both in and cm"
% ===============
%
% Mathematics
% -----------
%
```

```
% Let a and b be two non-negative integers such that U = floor(a 7227/100) =
% floor(b 7227/254).  It can be proven that a=50k, b=127k for some integer k.
% The proof is left to reader.  So U = floor(7227 k /2) for some k.
%
% Let's now find the largest such U <= T. So U = floor(k 7227/2)<= T which is
% equivalent (as k is integer) to k 7227/2 <= T + 1/2, i.e.
%
%     kmax = floor((2T+1)/7227)
%
% If we used for x>0 the formula floor(x)=round(x-1/2)=<x-1/2> we would end
% up basically with some 4T hence overflow problems even in \numexpr.
% Here I used <.> to denote rounding in the sense of \numexpr. It is not
% 1-periodical due to how negative inputs are handled, but here x-1/2>-1/2.
%
% The following lemma holds: let T be a non-negative integer then
%
%     floor((2T+1)/7227) = <(2T - 3612)/7227>
%
% So we can compute this k, hence get a=50k, b=127k, all within \numexpr and
% avoiding overflow.
%
% Implementation
% --------------
%
% Regarding the output in pt or sp, we seem to need floor(k 7227/2).
% The computation of floor(k 7227/2) as <(7227 k - 1)/2> would require to
% check if k==0 so we do it rather as <(7227 k + 1)/2> - 1.  No overflow
% can arise as k = 297147 for \maxdimen, and then 7227 k = 2**31 - 2279 and
% there is ample room for 7227k+1 using \numexpr.
%
% But this step, as well as initial step to get kmax will require to separate
% handling of negative input from positive one.
%
% Alternative
% -----------
%
% For non-negative T we can compute U = ((T+1)/7227)*7227. If U <= T keep it,
% else if U > T, replace it by U - 3614. This is alternative road to the maximal
% floor(k 7227/2) at most equal to T.
%
\def\texdimenbothincm#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenboth_a
    \the\numexpr\dimexpr#1\relax\relax-3612)/7227)*127sp\relax}%
\def\texdimenbothcmin#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenboth_a
    \the\numexpr\dimexpr#1\relax\relax-3612)/7227)*50sp\relax}%
\def\texdimenboth_a#1{\texdimengobtilminus#1\texdimenboth_neg-\numexpr((2*#1}%
\def\texdimenboth_neg-\numexpr((2*-{-\numexpr((2*}%
%
\def\texdimenbothincmsp#1{\number
    \expandafter\texdimenbothsp_a\the\numexpr\dimexpr#1\relax\relax
     -3612)/7227)*7227+1)/2-1\relax}%
\def\texdimenbothincmpt#1{\expandafter\texdimenstrippt\the\dimexpr
```

```
    \expandafter\texdimenbothsp_a\the\numexpr\dimexpr#1\relax\relax
     -3612)/7227)*7227+1)/2-1sp\relax}%
\def\texdimenbothsp_a#1{\texdimengobtilminus#1\texdimenbothsp_neg-\numexpr(((2*#1}%
\def\texdimenbothsp_neg-\numexpr(((2*-{-\numexpr(((2*}%
%
\let\texdimenbothcminpt\texdimenbothincmpt
\let\texdimenbothcminsp\texdimenbothincmsp
%
% "both mm and bp"
% ================
%
% Mathematics and Algorithm
% -------------------------
%
% We start from a dimension expressed in sp unit, "T sp". Assume T positive.
% We know how to get largest "X sp <= T sp" which is exactly expressible
% in mm unit
% i.e. can be written X=trunc(a 7227/2540) for some non-negative integer a.
% We want to achieve X=trunc(b 803/800) for some b.
%
%    Only the congruence of X modulo 803 matters for this.
%    It turns out that the mod 803 impossible values are 267, 535, 802.
%    As pointed out by Ruixi Zhang on the package repo issue #10,
%    when a<--a+2540, X increases by 7227=9*803 hence the value
%    modulo 803 does not change. Thus only "a modulo 2540" matters
%    to check if X(a) is attainable with bp unit. Ruixi Zhang found by
%    brute force that there are modulo 2540 nine excluded a-values
%
% Rather than checking if "a mod. 2540" avoids the 9 Ruixi Zhang values
% or if "X mod. 803" avoids  267, 535, 802, we will simply basically
% check if X sp = \texdimenbp{X sp}bp, as this approach is probably
% about the same cost or even less than computing "X mod. 803" and
% correspondingly branching.
%
% The key is that if "a" is bad, then "a-1" is automatically good as
% pointed out by R.Z. on #10, which can be seen without knowing the 9
% bad congruences, simply by noticing that a<--a-1 modifies X either to
% X-2 or X-3, so if X was bad certainly the new one is not.
%
% Once "a" has gotten its final value, we apply "\the\dimexpr a sp
% = D pt" trick to recover the D such that "D mm" gives rise to the found
% dimension.  We go via this "Dmm" intermediary also to express the final
% result as "X sp", because anyhow the "X" we worked with and had in
% our token stream has to be recomputed if a<--a-1, so lets always
% recompute it from final "a", and this goes via "D mm" (but see
% the paragraph MEMO for alternative for this trunc(a 7227/2540) step).
%
% I will copy here the style I used for bothincm expansion triggering
% via an already positioned \dimexpr waiting to output final result.
\def\texdimenbothbpmm#1{\expandafter\texdimenstrippt\the\dimexpr
                        \expandafter\texdimenbothbpmm_fork\the\numexpr\dimexpr#1;}%
\def\texdimenbothbpmm_fork#1{\texdimenzerominusfork
                             #1-\texdimenbothbpmm_zero
                             0#1\texdimenbothbpmm_neg
                             0-\texdimenbothbpmm_a
                             \krof#1}%
% because this is *inside* a pre-positioned \dimexpr, we don't have
% to worry about zero output ending up as -0.0
\def\texdimenbothbpmm_neg-{-\texdimenbothbpmm_a}%
\def\texdimenbothbpmm_zero#1;{\z@\relax}%
% now, find X sp <= T sp maximal and expressible in mm unit
% it will be X=trunc(a 7227/2540), we first get a candidate for "a"
\def\texdimenbothbpmm_a#1;%
    {\expandafter\texdimenbothbpmm_b\the\numexpr#1*2540/7227;#1;}%
% we get in a single line the X from this candidate, hijacking TeX's
% built-in *7227/2540... the "MEMO" above explains one could do this
% purely within \numexpr, working around its division rounds, and
% avoiding overflow, but I suspect this would be more costly.
\def\texdimenbothbpmm_b#1;{\expandafter\texdimenbothbpmm_c
    \the\numexpr\dimexpr\expandafter\texdimenstrippt\the\dimexpr#1spmm;#1;}%
% now we have X;a;T;
\def\texdimenbothbpmm_c#1;#2;#3;{%
% If X>T, our candidate "a=#2" must be decreased by 1 and we go to _ca
% The original #3 is not needed anymore
    \ifnum#1>#3 \expandafter\texdimenbothbpmm_ca\fi
% Else we decide whether it is "a" or "a-1" we must use. I preferred
% to induce a re-grabbing cost here, rather than have \texdimenbothbpmm_ca
% re-grab its arguments from \texdimenbothbpmm_d replacement text.
    \texdimenbothbpmm_d#1;#2;%
}%
% Here, dynamically at the time of the concluding \dimexpr, we
% check if X sp is expressible in bp unit and then use "a" or "a-1"
% accordingly
\def\texdimenbothbpmm_d#1;#2;{#2sp%
    \ifnum\dimexpr
    \expandafter\texdimenstrippt\the\dimexpr\numexpr(2*#1+1)*400/803spbp=#1
    \else-1sp\fi
% and a \relax to stop the concluding \dimexpr
    \relax
}%
% Here we must decrease "a=#2" by 1, recompute X=#1, then loop
% back to \texdimenbothbpmm_d. Hesitation between forcing a
% re-grab or doing it in one step with the subtraction of 1 done twice
\def\texdimenbothbpmm_ca\texdimenbothbpmm_d#1;#2;%
   {\expandafter\texdimenbothbpmm_cb\the\numexpr#2-1;}%
\def\texdimenbothbpmm_cb#1;{%
    \expandafter\texdimenbothbpmm_d
    \the\numexpr\dimexpr\expandafter\texdimenstrippt\the\dimexpr#1spmm;#1;%
}%
% done...
% now the lazy way for \texdimenbothmmbp
\def\texdimenbothmmbp#1{\expandafter\texdimenstrippt\the\dimexpr
    \expandafter\texdimenbothmmbp_a\the\numexpr\dimexpr\texdimenbothbpmm{#1}mm;}%
% If zero at this stage, we will correctly get 0.0 in the end
\def\texdimenbothmmbp_a#1#2;{\numexpr(2*#1#2+\texdimengobtilminus#1-1)*400/803sp\relax}%
% \texdimenbothbpmmpt and its alias \texdimenbothmmbppt
```

```
\def\texdimenbothbpmmpt#1{\texdimenpt{\texdimenbothbpmm{#1}mm}}%
\let\texdimenbothmmbppt\texdimenbothbpmmpt
% \texdimenbothbpmmsp and its alias \texdimenbothmmbpsp
\def\texdimenbothbpmmsp#1{\the\numexpr\dimexpr\texdimenbothbpmm{#1}mm\relax\relax}%
\let\texdimenbothmmbpsp\texdimenbothbpmmsp
%
% \texdimenwithunit
% =================
%
% Mathematics
% -----------
%
% The ex and em units are handled by TeX as if multiplying by a
% conversion factor f/65536 (here f sp = 1ex resp. = 1em).
%
% In particular, for any decimal D, input "D em" is handled the exact
% same way as input "D\dimexpr 1em\relax"; this is not
% the case for the core units except for pt and pc (and sp), whose
% conversion factors are the sole ones with a power of 2 denominator
% (respectively 1, 1, and 65536).  The further difference is that
% for the core units apart from sp, the conversion factor is >1.
%
% We assume for this discussion T is non-negative.
% If f/65536 > 1, the analysis is as above : some dimensions T sp
% are not attainable as D uu, but the formula
%     N=round((2T+1)*32768/f)
% will give a suitable decimal D via \the\dimexpr N sp\relax.
% (if T=0, we get N=0 as 32768/f<0.5)
% This D will let TeX convert D uu into T sp, if the dimension
% is attainable else it will be a closest match
% either from above or below (not necessarily nearest overall).
%
% If f/65536=1, attention that above formula would give N=1 for
% T=0 (was bug #4).
%
% If f/65536<1, all dimensions Tsp are attainable as D uu. Indeed
% D uu is parsed by TeX via N=round(D*65536), then T=trunc(N*phi),
% with phi=f/65536. Starting from T we need to find an N such that
% T/phi <= N< (T+1)/phi.
%
% This is equivalent to ceil(T/phi)<= N < ceil((T+1)/phi)
%
% Now obsolete remark: let v=(T+0.5)/phi. As its
% distance to the extremities is 0.5/phi>0.5, (phi>1) its rounding M
% to an integer verifies automatically T/phi < M < (T+1)/phi, so
% is a valid candidate. This was used at 0.99 release.
% (it is funny that N=round((2T+1)*32768/f) works for all f>0
%  *except* f=65536).
%
% The 1.0 release chooses to implement the ceil(T/phi) formula rather as
% it is closer to naive expectation "dim1/dim2" of a division.
%
% It is not obvious to compute this ceil(T/phi) without overflow.
```

```
%
% Implementation
% --------------
%
% \texdimenwithunit{dim1}{dim2}
%
% First done at 0.99, then refactored at 1.0:
% - to add support for dim2<0pt
% - to handle differently the dim2<1pt case and make the output
%   closer to mathematical dim1/dim2
%
% To handle dim2<0pt, we simply simultaneously do
% dim1<-- (-dim1) and dim2<-- (-dim2).
%
% dim2=0pt is not intercepted and will cause division by zero low-level
% error.  Code comments below were not adjusted and handle only
% dim2>0pt.
%
% We first get f from dim2 and branch according to whether f>65536,
% or f<=65536.
% We will also need to check the sign of T (dim1=T sp).
% f>65536: we compute round((2T+1)*32768/f)
% f=65536: merged with f<65536 branch (as it works and avoids checking for it)
% f<65536: 0.99 release used the round((2T+1)*32768/f) formula
%          (it is funny that it works for all f except for f=65536)
%
%          But the output then diverges noticeably from mathematical
%          dim1/dim2 "=" T*65536/f, the more so the smaller the dim2.
%          See issue #16 and also the discussion at #13.
%
%          1.0 release thus opted for the ceil(T*65536/f) formula, as it is the
%          smallest allowable choice, hence the closest to naive dim1/dim2.
%
%          To avoid arithmetic overflow issues we first do the euclidean
%          division T = k f + r, 0<= r < f, 0<= k
%
%          The final result in "sp" unit would be k*65536 + C with
%          C = ceil(r * 65536/f).
%
%          We don't do this k*65536 explicitly as it may overflow and is
%          anyhow unneeded: the output will be the integer k concatenated with
%          the decimal E given by TeX from \the\dimexpr C sp, i.e. such that
%          E pt = C sp, with C = ceil(r*65536/f).
%
%          As r is at most f-1, r*65536/f is at most 65536-65536/f, and as
%          65536>=f (we use this branch also for f=65536), C<=65535. Hence
%          E is never 1.0 but always "0.<some digits>"
%
%          To compute the Euclidean quotient k in \numexpr we use there
%          <(2T-f)/(2f)> i.e. round((2T-f)/2f) = trunc(T/f)
%          as we are careful to never have T=0 in-there...
%
%          Computing C = ceil(r * 65536/f) in \numexpr is the delicate
%          part, as r can be as large as f-1 hence 65535 and 65535*65536 would
%          overflow.  Let's try anyhow to see how to compute ceil() with round():
%
%          C = 65536 - floor(65536 * (1 - r/f))
%            = 65536 - round(65536*(f-r)/f - 0.5) (as r<f so no "round(-0.5)=-1")
%            = 65536 - <(2*65536*(f-r) - f)/(2f)>
%
%          Here the problem is with small r, and large f, and naive implementation
%          of this formula can overflow...
%          Let's thus retreat to eTeX scaling operation <r*65536/f> as it
%          operates with temporary double precision.
%
%          R=round(r*65536/f)=<r*65536/f> is either C-1 or C
%          Let x = mathematical exact r*65536/f:
%          - if R < x,  C=R+1.
%          - if R >= x, C=R.
%
%          C=ceil(r*65536/f) is the smallest integer such that
%          trunc(C*f/65536)>=r, or more precisely (as f<=65536) the
%          smallest integer with trunc(C*f/65536)=r. So trunc(R*f/65536)
%          will be either r (then R=C), or r-1, then R=C-1.
%
%          Method from release 1.0: let's TeX compute P=trunc(R*f/65536) itself!
%          Via P sp = E <f sp> where E is a decimal such that E pt = R sp.
%          So
%          - if P>=r (it is then equal to r in fact) then C=R
%          - if P<r (it is then equal to r-1), then C=R+1.
%
%          New method: overflow-free pure \numexpr way to get the sign of R-x.
%
%          Write R=4*S+t, with say S=<R/4>=round(R/4), so t=-2,-1,0,+1.
%
%          Then R*f-65536*r = 4*(S*f-16384*r)+t*f
%
%          We know that R<=C<65536, so <R/4> <= 16384 and 16384*f
%          is at worst 2**(14+16)=\maxdimen+1 but we will be in \numexpr,
%          so no overflow!
%          And r<f<=65536 so also 16384*r can not overflow.
%          As |R - r*65536/f|<= 0.5, then |R*f-65536*r|<= f/2, so
%          4*|S*f-16384*r| <= 2.5*f is very far from overflow risk
%
%              T>0, 0<f<=65536
%              k = <(2*T-f)/(2*f)>
%              r = T - k*f
%              R=<r*65536/f>
%              S=<R/4>
%              t=R-4*S
%
%              IF: 4*(S*f-16384*r)+t*f < 0 THEN C=R+1 ELSE C=R.
%
%              Ept=\the\dimexpr Csp, E=0.d...d
%
%              End expansion with the contatenation k.d...d
```

```
%
\def\texdimenwithunit#1#2{\expandafter\texdimenwithunit_i
% no premultiplication of dim1 by 2 as was done for technical
% reasons when dim2<1pt branch used round((2T+1)*32768/f)
    \the\numexpr\dimexpr#2\expandafter;\the\numexpr\dimexpr#1;%
}%
\def\texdimenwithunit_i#1{%
     \texdimengobtilminus#1\texdimenwithunit_switchsigns-%
     \texdimenwithunit_j#1%
}%
\def\texdimenwithunit_switchsigns-\texdimenwithunit_j-#1;#2%
{%
% due to \texdimenwithunit_Bneg we can not simply prefix dim1
% with -, as -0 is bad there. So let's check also if #2 is 0
    \texdimenzerominusfork
      #2-\texdimenwithunit_Bzero % also used in \texdimenwithunit_B
      0#2\texdimenwithunit_j     % abusive shortcut
       0-{\texdimenwithunit_ic#2}%
    \krof
    #1;%
}%
\def\texdimenwithunit_ic#1#2;{\texdimenwithunit_j#2;-#1}%
\def\texdimenwithunit_j#1;#2{%
        \ifnum#1>\p@\texdimenwithunit_A\fi
        \texdimenwithunit_B#2#1;%
}%
% unit>1pt, handle this as for bp.
% Attention it would be wrong for unit=1pt!
\def\texdimenwithunit_A\fi\texdimenwithunit_B#1#2;#3;{\fi
    \expandafter\texdimenstrippt
    \the\dimexpr\numexpr(2*#1#3+\texdimengobtilminus#1-1)*32768/#2sp\relax
    % - fine if dim1>0, <0, or =0
    % - with *\p@ better but an early doubled dim2 would complicate 1pt
    % test and not sure if doing \p@/(2*#2) here advantageous
}%
% unit<=1pt.
% if dim1<0, simply negate result for dim1>0 as it can not possibly be 0.0
% Indeed T*65536/f will be at least 1 so its ceil also (in fact ceil
% will even be at least 2 if f<65536).
% The dim1=0 case must get filtered out due to way of calculating the
% "ceil" in \numexpr
\def\texdimenwithunit_B#1{\texdimenzerominusfork
                           #1-\texdimenwithunit_Bzero
                           0#1\texdimenwithunit_Bneg
                            0-\texdimenwithunit_Ba
                          \krof#1}%
\def\texdimenwithunit_Bzero#1;#2;{0.0}%
\def\texdimenwithunit_Ba#1#2;#3;{%
    % no overflow possible from 2*#1#3 in \numexpr
    \expanded{\expandafter\texdimenwithunit_Bb
              \the\numexpr(2*#1#3-#2)/(2*#2);#1#3;#2;}%
}%
% I could have inserted \expanded\bgroup in \texdimenwithunit_B
% but then needed to modify _Bzero (used also by \texdimenwithunit_switchsigns)
% so easiest is to simply defined Bneg explicitly here rather than
% insisting on deriving it from _Ba
\def\texdimenwithunit_Bneg-#1;#2;{%
    \expanded{-\expandafter\texdimenwithunit_Bb
               \the\numexpr(2*#2-#1)/(2*#1);#2;#1;}%
}%
% now k;T;f;. Get the remainder r=T-k*f, and abandon k in the token stream.
% the earlier \expanded maintains f-expandability
\def\texdimenwithunit_Bb#1;#2;#3;{%
    #1\expandafter\texdimenwithunit_Bc\the\numexpr#2-#1*#3;#3;%
}%
% now r;f;. Get R=<r*65536/f>
\def\texdimenwithunit_Bc#1;#2;{%
    \expandafter\texdimenwithunit_Bd\the\numexpr #1*\p@/#2;#1;#2;%
}%
% R;r;f; Is 4*(S*f-16384*r)+t*f < 0 ? with S=<R/4>, t=R-4S
\def\texdimenwithunit_Bd#1;#2;#3;{%
    \expandafter\texdimenwithunitstripzeroandpt
    \the\dimexpr\numexpr#1%
    \ifnum\numexpr 4*((#1/4)*#3-16384*#2)<\numexpr(4*(#1/4)-#1)*#3\relax
     +1\fi sp\relax
}%
{\catcode`P12\catcode`T12\lowercase{\gdef\texdimenwithunitstripzeroandpt0#1PT}{#1}}%
\texdimensendinput
```