

XASSOCCNT

Associated counters stepping simultaneously

Documentation

Version 0.9

06/19/2016

Author: Christian Hupfer[†]

[†]christian.hupfer@yahoo.de

Contents

1	Introduction	6
2	Requirements, loading and incompatibilities	7
2.1	Required packages and T _E X engine	7
2.2	Loading of the package	7
2.3	Incompatibilities	7
2.4	Package options	8
3	Documentation of Macros	8
3.1	Association macros	9
3.2	Driver macros	13
3.3	Query macros	13
3.3.1	\IfIsDocumentCounter-Queries	16
3.4	Macros about the reset list	17
3.5	Information macros	18
4	Suspending and Resuming	20
5	Additions and extensions to standard commands	23
5.1	Extension of L ^A T _E X 2 _ε commands	23
5.2	Additions to L ^A T _E X 2 _ε commands	23
6	Loops on multiple counter	26
7	Coupled counters	29
7.1	Common options for coupled counters	29
7.2	Macros for coupled counters	29
8	Periodic counters	32
8.1	Commands related to periodic counters setup	32
8.2	Commands to query for periodic counter feature	33
9	Total counters	34
10	Super total counters	35
10.1	The  numberofruns counter	37
11	Counter backup/restoration	37
11.1	Macros for backup/restoration	37
11.2	Some notes on the backup features	40

12 Counter output	40
13 To - Do list	42
14 Acknowledgments	43
15 Version history	44
A Total number of sections	46
B Subsection with suspension	46
B.1 First dummy subsection	46
B.2 Second dummy subsection	46
B.3 Third dummy subsection after removing the associated counter	46
B.4 Suspension of a non-associated counter	47
Index	48

Typographical conventions

Throughout this documentation following symbols and conventions are used:

-  **foo** means a the class `foo`
-  **foo** names a package `foo`
-  **foo** indicates a counter named `foo`
-  `foo` will indicate either a file named `foo` or a file extension `foo`
-  `foo` will indicate some files
-  `foo` names a special feature or tag `foo`
-  **foo** deals with a command or package option named `foo`

Preface

This package is the successor and a complete rewrite of  `assoccnt`. Not all features of that package are implemented yet – if some functionality of your document depends on  `assoccnt`, continue using the older version and shift gradually to  `xassoccnt` please.

 Most times class and package authors will benefit of this package, but there might be usual documents that need the features of `|xassoccnt|`

1 Introduction

The aim of this package is to provide some additional support for example for a package like  `totcount`.

For example, the total number of pages in a document could be achieved by using

```
%
\regtotcounter{page}
...
The number of pages in the document is \number\totvalue{page} page(s) -- but in ↵
    {fact it has \total{totalpages} pages.
```

... The number of pages in the document is 50 page(s) – but in fact it has 49 pages.

This will work, as long there is no reset of the page counter, as it might happen in the case of `\setcounter` or `\pagenumbering` being applied in the document. The result is a false page counter total value.

This package provides associate counters, i.e. counters that are increased simultaneously with a driver counter and are not influenced by a resetting of the driver counter, as long as not being added to the reset list by definition of the counter or explicitly by `\@addtoreset`.

This package defines some macros to handle associated counters. The only interception to the standard behaviour is within the redefined commands `\addtocounter` and `\stepcounter`. The usual commands still work, as there is code added to their definition. In a previous version `\refstepcounter` was redefined, but since these use `\addtocounter` effectively, it was decided to use the basic command.

Internally, the associated counters are stored in one list per counter – it is not recommended to operate on those lists directly.

Please note that this package does not provide means for simultaneous stepping of counters defined by plain T_EX `\newcount` command.



2 Requirements, loading and incompatibilities

2.1 Required packages and T_EX engine

The package does not require features from Xe_LT_EX or Lua_LT_EX but can be run with those features as well as with _LT_EX or pdf_LT_EX. The compilation documentation requires however pdf_LT_EX as of version 0.9.

-  [xcolor](#)
-  [xparse](#)
-  [l3keys2e](#)

The documentation file requires some more packages such as  [tcolorbox](#) but those packages are widely available on CTAN, MikTeX and TeXLive as well.

2.2 Loading of the package

Loading is done with

```
\usepackage[options]{xassoccnt}
```

For the relevant options see section 2.4

Concerning the packages  [hyperref](#) and  [cleveref](#) : The preferred loading order is the usually recommended on:

- other packages
-  [xassoccnt](#)
-  [hyperref](#)
-  [cleveref](#)

For potential problems see section 2.3

2.3 Incompatibilities

- This package cannot be used with plain T_EX and will not provide support for counters (or better counter registers) that have defined with the T_EX primitive `\countdef` directly in a _LT_EX 2_ε document nor will it hook into plain T_EX `\advance` commands used otherwise than in the usual _LT_EX 2_ε wrappers `\addtocounter` etc.

- This package does not work really well with the  **calc** package if that one is loaded after  **xassoccnt** . Load  **calc** before  **xassoccnt** !

! As of version 0.9  **xassoccnt** will abort compilation if  **calc** is loaded after this package, but will issue a warning only if  **calc** is loaded first.

! Of course any package other that loads  **calc** must be loaded before **xassoccnt**, i.e.  **mathtools** .

-  **xassoccnt** and  **perpage** are not compatible completely. As of version 0.9 it is not possible to use the command `\AddAbsoluteCounter` from  **perpage** – this feature is provided already by this package.

! It is not recommended to use counters under control of  **perpage** with the association method!



2015-11-10

2.4 Package options

As of version 0.9  **xassoccnt** supports the package options

`autodefinecounters`= $\langle true, false \rangle$ (initially false)

Setting this package option to `true`, all counters used with the special package macros will be autodefined, unless disabled locally. See the commands

- `\DeclareAssociatedCounters`^{→P.9}
- `\AddAssociatedCounters`^{→P.10}

`nonumberofruns` (initially not set)

Using this package option the super total counter  **numberofruns** counter will not be defined. See section 10.1 about this feature.

v0.6
2016-03-05

3 Documentation of Macros

The main purpose of this package is co-stepping of counters, but there are some helper commands in addition to macros provided $\text{\LaTeX}2_{\epsilon}$ already, see section Additions and extensions to standard commands.

- Section Association macros describes the most important macros for setting up associated counters
- Section Driver macros informs about the macros for setting up, removing or clearing driver counters

- Section Query macros deals with query command sequences about counters being a driver or an associated counters
- Section Information macros contains routines that show which counters have been changed last

3.1 Associated counters commands

All macros have the general rule, that the driver counter is specified as 1st mandatory argument to the macro, which is in almost all cases the 2nd argument of the macro.

`\DeclareAssociatedCounters` [*options*] {*driver counter*} {*associated counters list*}

This command is the main macro of the package. It declares the counter names being specified in comma - separated - list (CSV) which should be stepped simultaneously when the driver counter is increased by `\stepcounter`. If only counter is to be associated, omit a trailing `,"!`

#1 [*options*]:

`autodefine`=*choice* (initially none)

This choice - key can be specified if the specified counters should be defined if they not already available. Possible values are

- `none` – no counter is autodefined
- `all` – all counters will be autodefined
- `driver` – only driver counters will be autodefined
- `associated` – only associated counters will be autodefined

Default is `none`

`sloppy`

If `autodefine` key is used, the `sloppy` key disables the check whether a counter is defined already.

#2 {*driver counter*}

Holds the name of the driver counter to which the list of counters should be associated

#3 {*associated counters list*}

A comma separated list of counter names that should be associated to the driver counter

- This command is a preamble command, i.e. it can be used in the preamble of the document or within other packages or class files only.
- This command should be used as early as possible, i.e. in the preamble of the document, since the driven counters are not increased as long as they are not associated to the driver counter. On the hand, it is possible or may be required to control the starting point of

the association at any position in the body of the document, when the association should start later on. Use the command `\AddAssociatedCounters`^{→P.10} if counters should be associated within the document body.

```
%%% The association of another total pages in this example just takes place
here, so the stepping of the counter will start from here and providing
a 'wrong' value.
```

```
%%%
\DeclareAssociatedCounters{page}{totalpages,another totalpages}%
This document has \number\totvalue{totalpages} (note: )
(\number\totvalue{another totalpages}) pages.
```

This document has 49 (note: 10) pages.

- Current version (0.9) rules:
 - No checking whether the 2nd and 3rd arguments hold counter names is applied.
 - Mutually cross - association of two counters is not supported! The compilation will stop on this!

A driver counter, say, `\cfoo` of, say, `\cfoobar` can not be an associated counter of `\cfoobar`, which in turn can be a driver counter of other counters, of course.

A contrary feature are the **coupled counters** – If some counters should share a common base, i.e. increasing one arbitrary member counter of a group of counters then all should be increased, this called coupling of counters – all group members are on an equal footing. See section 7 about this feature.

On the other side, **associated counters** belong to a hierarchy. The driver counter dominates the associated counters.
- A self-association of the driver counter to itself is ignored internally as this would lead to inconsistent counter values.
- The order of the specification of associated counters in the 2nd arguments is of no importance.
- Specifying an associated counter name multiple times has no effect, only the first occurrence of the name will be used.

`\AddAssociatedCounters` [*options*] {*driver counter*} {*associated counters list*}

The usage of this macro is similar to `\DeclareAssociatedCounters`^{→P.9}; if it is called in the document preamble (or in package file), `\AddAssociatedCounters` falls back to

$$\backslash\text{DeclareAssociatedCounters}^{\rightarrow\text{P.9}},$$

having the same optional argument functionality with `autodefine`^{→P.9} and `sloppy`^{→P.9}; if it is called in the document body, this command adds some counters to the associated counter list

for a specific driver counter – if this list does not exist, the \LaTeX run will issue a warning, but add the driver counter to the driver list and the associated counters analogously.

Using `\AddAssociatedCounters` in the document body automated generation of counters is disabled.



Description of arguments of command `\AddAssociatedCounters`

#1 [*options*]: As of version 0.9, the optional argument [*options*] are the same as for `\DeclareAssociatedCounters`^{→P.9}, see `autodefine`^{→P.9} and `sloppy`^{→P.9}.

#2 {*driver counter*}

Holds the name of the driver counter to which the list of counters should be associated

#3 {*associated counters list*}

A comma separated list of counter names that should be associated to the driver counter

`\RemoveAssociatedCounter`{*driver counter*}{*associated counter*}

This command removes a counter from the existing list for a driver counter, i.e. the counter will not be increased any longer by `\stepcounter`. It can be increased however manually, of course.

```
\RemoveAssociatedCounter{page}{anothertotalpages}
This document has \number\totvalue{totalpages} (beware: 2
  \number\totvalue{anothertotalpages}) pages.
```

This document has 49 (beware: 10) pages.

`\RemoveAssociatedCounters`{*driver counter*}{*list of associated counters*}

This command removes the comma-separated-value list of counters from the existing list for a driver counter, i.e. the counters will not be increased any longer by `\stepcounter`. They can be increased however manually, of course.

Take care not to confuse the commands `\RemoveAssociatedCounters` and `\RemoveAssociatedCounter`

`\ClearAssociatedCounters`[*options*]{*driver counter*}

This command clears the internal list for all counters associated to the {*driver counter*}. The counters will not be increased automatically any longer.

The optional argument is not used as of version 0.9.

Please note that the driver counter is not removed from the list of driver counters – this simplifies reassociating of (other) counters to this one later on with the macro `\AddAssociatedCounters`^{→P.10} and suppress the relevant warning.

If the driver counter and all its associated counters should be removed, use `\RemoveDriverCounter`^{→P.13} instead.

3.2 Driver counter commands

`\AddDriverCounter` [*<options>*] {*<driver counter name>*}

Description of arguments of command `\AddDriverCounter`

- #1 [*<options>*]: As of 0.9, the optional argument [*<options>*] is not used so far, but is reserved for later purposes.
- #2 {*<driver counter name>*}
Holds the name of the driver counter that should be added to the list of driver counters.

`\RemoveDriverCounter` [*<options>*] {*<driver counter>*}

This command clears the internal list for all counters associated to the {*<driver counter>*}. The counters will not be increased automatically any longer.
The optional argument is not used as of version 0.9.
If all driver counters should be unregistered, use `\ClearDriverCounters` instead!

`\ClearDriverCounters` [*<options>*]

This clears completely the list of driver counters, such that no counters are regarded as being associated – i.e. no driver is hold as being a driver counter.
The optional argument is not used as of version 0.9.

3.3 Commands for queries

Sometimes it might be necessary to get information, whether a counter is regarded as a driver or as an associated counter. This section describes some query macros in order to obtain this information.

`\IsAssociatedToCounter` {*<driver counter>*} {*<associated counter>*} {*<True branch>*} {*<False branch>*}

This macro checks, whether a counter is associated to a particular given driver counter and expands the corresponding branch. If the internal driver counter list does not exist, the false branch will be used, since this also means, that the possibly associated counter is not associated at all.

Description of arguments of command `\IsAssociatedToCounter`

#1 `{\langle driver counter \rangle}`

Holds the name of the driver counter to which `{\langle associated counter \rangle}` the could possibly be associated.

#2 `{\langle associated counter \rangle}`

Contains the name of the possibly associated counter.

#3 `{\langle True branch \rangle}`

This code is expanded if the counter is associated to the driver, otherwise it is ignored.

#4 `{\langle True branch \rangle}`

This code is expanded if the counter is **not** associated to the driver, otherwise it is ignored.

```
% Remove associated counter first for demonstration purposes
\RemoveAssociatedCounter{page}{anothertotalpages}
\IsAssociatedToCounter{page}{totalpages}{Yes, totalpages is associated}{No, }
  {totalpages is not associated}

\IsAssociatedToCounter{page}{anothertotalpages}{Yes, anothertotalpages is }
  {associated}{No, anotherpages is not associated}

-----
Yes, totalpages is associated
No, anotherpages is not associated
```

See also

- `\IsAssociatedCounter`^{→P.15} for checking whether a counter is associated
- `\IsDriverCounter`^{→P.15} in order to check whether a counter is a driver.
- `\GetDriverCounter` returns the driver counter name for a given associated counter name

`\GetDriverCounter``{\langle counter name \rangle}`

This commands returns the driver counter to which the counter name of the first argument is connected to. If the counter is not defined, the macro returns nothing.

- No check whether the counter name is defined is performed
- No check whether the counter is associated at all is performed. Usage of this command in conjunction with `\IsAssociatedCounter`^{→P.15} is strongly encouraged.

```

%
totalpages is associated to the ↵
  {\textcolor{blue}{\textbf{\GetDriverCounter{totalpages}}}} counter.
% Try with an undefined counter name
humptydumpty is associated to the ↵
  {\textcolor{blue}{\textbf{\GetDriverCounter{humptydumpty}}}} counter.

```

totalpages is associated to the **page** counter. humptydumpty is associated to the – counter.

`\IsAssociatedCounter`{*<counter name>*}{*<True branch>*}{*<False branch>*}

This commands tests, whether a given counter name is an associated counter and expands correspondingly the true or the false branch. The command does not tell to which driver the counter it is associated – this information can be obtained by `\GetDriverCounter`^{→ P.14}.

Description of arguments of command `\IfAssociatedCounter`

#1 {*<counter name>*}

Contains the name of the possibly associated counter

#2 {*<True branch>*}

This code is expanded if the counter is associated to a driver, otherwise it is ignored

#3 {*<True branch>*}

This code is expanded if the counter is **not** associated a driver, otherwise it is ignored

```

\IsAssociatedCounter{section}{Yes, section is an associated counter}{No, ↵
  {\section counter does not have the associated counter properties}
\IsAssociatedCounter{totalpages}{Yes, totalpages is an associated counter}{No, ↵
  {\totalpages counter does not have the associated counter properties}

```

No, section counter does not have the associated counter properties Yes, totalpages is an associated counter

`\IsDriverCounter`{*<driver counter name>*}{*<True branch>*}{*<False branch>*}

This commands tests, whether a given counter name is a driver counter and expands correspondingly the true or the false branch.

Description of arguments of command `\IfDriverCounter`

#1 `{\langle driver counter name \rangle}`

Contains the name of the possible driver counter

#2 `{\langle True branch \rangle}`

This code is expanded if the counter is a driver, otherwise it is ignored

#3 `{\langle True branch \rangle}`

This code is expanded if the counter is **not** a driver, otherwise it is ignored

```
\IsDriverCounter{section}{Yes, section is a driver counter}{No, section counter }
  {does not have driver properties}
```

Yes, section is a driver counter

`\IsSuspendedCounter``{\langle counter name \rangle}{\langle true branch \rangle}{\langle false branch \rangle}`

See Suspending and Resuming on this topic.

This command checks, whether a counter is suspended, i.e. not updated at all and expands the corresponding branches.

#1 `{\langle counter name \rangle}`

Contains the name of counter presumed to be suspended

#2 `{\langle True branch \rangle}`

This code is expanded if the counter is suspended, otherwise it is ignored

#3 `{\langle True branch \rangle}`

This code is expanded if the counter is **not** suspended, otherwise it is ignored

3.3.1 Commands checking whether a name refers to a counter

✉ **xassoccnt** provides three commands that are quite similar – all check whether `{\langle name \rangle}` is an already defined $\text{\LaTeX} 2_{\epsilon}$ counter (name), in good tradition with the ✉ **xparse** - syntax:

- `\IfIsDocumentCounterTF[\langle \rangle]{\langle name \rangle}{\langle true branch \rangle}{\langle false branch \rangle}`

This macro performs the full branching

- `\IfIsDocumentCounterT[\langle \rangle]{\langle name \rangle}{\langle long true branch \rangle}`

This command executes only if the name is a counter.

`\IfIsDocumentCounterF[⟨⟩]{⟨name⟩}{⟨true branch⟩}`

This command executes only if the name is not a counter.

The optional argument is not used as of version 0.9 for none of those three commands.

3.4 Information macros about the counter reset list

Sometimes it might be necessary or convenient to know how many counters are on a reset list of some other counters, i.e. added by `\newcounter{counter}[resetting counter]` or `\NewDocumentCounter`^{→ P.23}. There are some macros that provide this information:

`\countersresetlistcount{⟨counter name⟩}`

This macro determines the number of counters being in the reset list of the counter specified as mandatory argument.

Please note: This command isn't expandable. The number is stored internally to another macro, which can be accessed with `\getresetlistcount`, which returns a pure integer number.

`\getresetlistcount`

This macro returns the number of counters being in the reset list of the counter specified as mandatory argument. It needs a previous call of `\countersresetlistcount` first!

If the counter has no other counters in its reset list, the value of 0 is returned.

`\IfInResetListTF[⟨⟩]{⟨resetting counter⟩}{⟨reset counter⟩}{⟨true branch⟩}{⟨false branch⟩}`

This command sequence tests whether the counter `⟨reset counter⟩` is in the reset list of `⟨resetting counter⟩` and expands the relevant branch then. See the short-circuit commands `\IfInResetListT` and `\IfInResetListF` as well.

`\IfInResetListT[⟨⟩]{⟨resetting counter⟩}{⟨reset counter⟩}{⟨true branch⟩}`

This command sequence tests whether the counter `⟨reset counter⟩` is in the reset list of `⟨resetting counter⟩` and expands to the true branch. See the related commands `\IfInResetListTF` and `\IfInResetListF` as well.

`\IfInResetListF[⟨⟩]{⟨resetting counter⟩}{⟨reset counter⟩}{⟨false branch⟩}`

This command sequence tests whether the counter `⟨reset counter⟩` is not in the reset list of `⟨resetting counter⟩` and expands to the false branch. See the related commands `\IfInResetListTF` and `\IfInResetListT` as well.

`\DisplayResetList` [*separator=,*] {*resetting counter*}

This command displays the reset list of a counter as a separated list. If the counter has no resetting list, nothing is shown.

v0.8
2016-06-10

Description of arguments of command `\DisplayResetList`

- #1 [*separator*] This separator is used for display, it defaults to a comma character.
- #2 {*resetting counter*}
- Contains the name of counter whose resetting list should be displayed.

`\ShowResetList` {*resetting counter*}

This command displays the reset list of a counter on the terminal as the `\show` command would do. This is rather useful for debugging purposes only.

v0.8
2016-06-10

3.5 Information on counters

On occasions it might be important to have some information which counter has been changed last. Since there are four commands manipulating counter values, there are four corresponding routines for this:

`\LastAddedToCounter`

This command has no arguments and expands to the name of the counter which was used last in `\addtocounter`. There is no further typesetting done with the countername.

```
\newcounter{SomeCounter}
```

```
\addtocounter{SomeCounter}{10}
```

```
The last counter something added to was \LastAddedToCounter.
```

```
The last counter something added to was SomeCounter.
```

! Please note that `\LastAddedToCounter` might fail!

`\LastSteppedCounter`

This command has no arguments and expands to the name of the counter which was stepped last using `\stepcounter`. There is no further typesetting done with the countername.

```
\stepcounter{SomeCounter}
```

The last counter being stepped was \LastSteppedCounter.

The last counter being stepped was SomeCounter.

`\LastRefSteppedCounter`

```
\begin{equation}
  E = mc^2 \label{eq::einstein}
\end{equation}
% \stepcounter{SomeCounter}
```

The last counter being refstepped was \LastRefSteppedCounter.

$$E = mc^2 \tag{1}$$

The last counter being refstepped was equation.

`\LastSetCounter`

This command has no arguments and expands to the name of the counter which was set last using `\setcounter`. There is no further typesetting done with the countername.

```
\setcounter{SomeCounter}{21}%
```

The last counter being set was \LastSetCounter.

The last counter being set was SomeCounter.

`\LastCounterValue`

This command has no arguments and expands to the value of the very last change of a counter, i.e. using `\setcounter` etc.

```
\setcounter{SomeCounter}{100}%
```

The last counter being set was \LastSetCounter and it had the value `\LastCounterValue{}` then, where as `\stepcounter{equation}` will yield `\fbox{\LastSteppedCounter}` and `\LastCounterValue!`

The last counter being set was SomeCounter and it had the value 100 then, where as will yield equation and 2!

The usage of `\LastSetCounter` is best together with one of the other `\Last...` macros.

! All of the `\Last...` macros are expandable, i.e. it is possible to store the value to a macro defined with `\edef`

```
\setcounter{SomeCounter}{50}%

\edef\lastcounterset{\LastSetCounter}
\edef\lastcountervalue{\LastCounterValue}

\setcounter{equation}{81}%
```

The last counter being set was `\fbox{\LastSetCounter}` and it had the value `\LastCounterValue` then, but we changed `\lastcounterset` earlier and it had the value `\lastcountervalue` then.

The last counter being set was `equation` and it had the value 81 then, but we changed `SomeCounter` earlier and it had the value 50 then.

! Please note, that all of these commands are only working in the current run of compilation, i.e. after there has been some operation on the counters. They can't be used for information on the last changed counter in a previous run.

4 Suspending and resuming (associated) counters

Rather than removing an associated counter from the list, it is possible to suspend the automatic stepping for a while and then resume it (or completely drop it), for example, if the value of a counter should not be stepped within a specific chapter etc.

! Suspension and resuming counters can cause wrong hyper links if [hyperref](#) is used.

v0.8
2016-06-10

`\SuspendCounters` [*options*] {*counters list*}

Description of arguments of command `\SuspendCounters`

#1 [*options*]

Not used so far, reserved for later usage.

#2 {*counters list*}

Contains the name of counters to be suspended, separated by commas (CSV - list)

`\CascadeSuspendCounters` [*<options>*] {*<counters list>*}

This macro is more powerful than `\SuspendCounters`^{→ P.20}, since it tries to detect whether a counter has a reset list and 'mutes' the counters on this list as well and checks whether those counters themselves have reset lists and cascades down to the final state.

! Stated differently: All counters anyhow connected to a counter named `foo` will be suspended, e.g. for the `book` class and `chapter`, this means in a standard setup, that `section,figure,table,equation,footnote` will be suspended, as well as in consequence `subsection,subsubsection,paragraph,subparagraph`, assuming hereby no other counters have been added to the reset lists.

Description of arguments of command `\CascadeSuspendCounters`

#1 [*<options>*]

Not used so far, reserved for later usage.

#2 {*<counters list>*}

Contains the name of counters to be suspended, separated by commas (CSV - list)

`\ResumeSuspendedCounters` [*<options>*] {*<counters list>*}

As of version 0.9 the optional argument is not used and reserved for later purposes. This command revokes the suspension of the counters in the {*<counters>*} list.

`\ResumeAllSuspendedCounters` [*<options>*]

As of version 0.9 the optional argument is not used and reserved for later purposes. This command revokes all suspended counters.

! If a driver counter is suspended, all counters associated to it are suspended too!

```
\textbf{This example shows 4 equations, but only two of them are counted}
```

```
\begin{equation}
E_{0} = mc^2
\end{equation}
```

Now suspend the equations:

```
\SuspendCounters{equation}
\begin{equation}
E^2 = \left( pc \right)^2 + E_{0}^2
\end{equation}
```

```
\begin{equation}
m(v) = \frac{m_{0}}{\sqrt{1-\frac{v^2}{c^2}}}
\end{equation}
```

And resume it: `\ResumeSuspendedCounters{equation}`

```
\begin{equation}
E = h \nu
\end{equation}
```

There are `\number\totvalue{totalequations}`~equations in here!

This example shows 4 equations, but only two of them are counted

$$E_0 = mc^2 \tag{1}$$

Now suspend the equations:

$$E^2 = (pc)^2 + E_0^2 \tag{1}$$

$$m(v) = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{1}$$

And resume it:

$$E = h\nu \tag{2}$$

There are 2 equations in here!

5 Additions and extensions to standard counter related commands

5.1 Extension of $\LaTeX 2_{\epsilon}$ commands

`\addtocounter`{*counter*}{*increment value*}[*options*]

The `\addtocounter` macro behaves like the usual `\addtocounter` counter, but takes care to specific counter features such as  **periodic counters** and has an optional argument in order to perform special settings.

As of 0.9, there is only one option used:

`wrap`=*true/false* (initially true)

This key determines whether addition of values to a periodic counter (see Periodic counters) will lead to a modulo part addition.

v0.9
2016-06-18

5.2 Additions to $\LaTeX 2_{\epsilon}$ commands

`\NewDocumentCounter`[*options*]{*counter*}[*resetting counter*]

This command is a new interface to `\newcounter` and behaves effectively the same.

`initial`=*integer value* (initially 0)

This is used for the start value of the new counter.

`\DeclareDocumentCounter`[*options*]{*counter*}[*resetting counter*]

This command is the preamble-only version of `\NewDocumentCounter`.

`\SetDocumentCounter`[*options*]{*counter*}{*counter value*}

This command behaves like the standard macro `\setcounter`, but has an additional optional 1st argument.

Description of arguments of command `\SetDocumentCounter`**#1** [*options*]:**associatedtoo**=*true/false*

If enabled (*true*), `\SetDocumentCounter` will use the counter value for all counters associated to this driver counter as well. Initially, this option is set to *false*.

onlycounters=*comma separated list of counters*

If this key is used, only those associated counters are set as well that are given in the comma separated list.

Names, that are either not referring to counters at all or to counters that are not associated to the given driver counter will be ignored silently.

#2 {*counter*} Holds the name of the (driver) counter to be set.**#3** {*counter value*} Holds the value to be setSome notes on `\SetDocumentCounter` ^{→ P.23}

- The option keys `associatedtoo` and `onlycounters` are mutually exclusive!
- The counter to be set can be either a driver counter or an otherwise associated counter.

`\StepDownCounter` [*options*] {*counter*}This macro subtracts the value of 1 from the counter and is the counterpart of `\stepcounter`.v0.4
2016-01-26Description of arguments of command `\StepDownCounter`**#1** [*options*]: As of version 0.9, this option is not used**#2** {*counter*} Holds the name of the first counter.`\SubtractFromCounter` [*options*] {*counter*} {*delta value*}This macro subtracts the (positive) delta value from the counter and is the counterpart of `\addtocounter`v0.4
2016-01-26

Description of arguments of command `\SubtractFromCounter`

- #1 [*options*]: As of version 0.9, this option is not used
- #2 {*counter 1*} Holds the name of the first counter.
- #3 {*delta value*} Holds the (positive) value to be subtracted from the counter value.

`\CopyDocumentCounters` [*options*] {*source counter*} {*target counter*}

This document copies the counter value from the source counter in argument 2 to the target counter in argument 3.

Description of arguments of command `\CopyDocumentCounters`

- #1 [*options*]: As of version 0.9, this option is not used
- #2 {*source counter*} Holds the name of the source counter.
- #3 {*target counter*} Holds the name of the target counter.

`\SwapDocumentCounters` [*options*] {*counter 1*} {*counter 2*}

This macro swaps the values of the counters given in arguments 2 and 3

Description of arguments of command `\SwapDocumentCounters`

- #1 [*options*]: As of version 0.9, this option is not used
- #2 {*counter 1*} Holds the name of the first counter.
- #3 {*counter 2*} Holds the name of the second counter.

`\SyncCounters` [*options*] {*driver counter*}

This document synchronizes the driver counter value to the associated values. It has the same options as `\SetDocumentCounter`^{→ P.23}. If the given counter is no driver counter, nothing is done.

Description of arguments of command `\SyncCounters`

- #1 [*options*]: see `\SetDocumentCounter`^{→ P.23}
- #2 {*source counter*} Holds the name of the source counter.

```

%[breakable=true]
\SetDocumentCounter{foocntr}{17}
\SetDocumentCounter{foobarcntr}{20}

\begin{itemize}
\item Displaying counters

\thefoocntr\ and \thefoobarcntr
\item Swapping counters

\SwapDocumentCounters{foocntr}{foobarcntr}

\thefoocntr\ and \thefoobarcntr

\item Step down counters

\StepDownCounter{foocntr}
\StepDownCounter{foobarcntr}

\thefoocntr\ and \thefoobarcntr

\item Subtracting some value from the counters
\SubtractFromCounter{foocntr}{5}
\SubtractFromCounter{foobarcntr}{10}

\thefoocntr\ and \thefoobarcntr
\end{itemize}

```

- Displaying counters
17 and 20
- Swapping counters
20 and 17
- Step down counters
19 and 16
- Subtracting some value from the counters
14 and 6

6 Performing the same action for many counters

Sometimes it might be necessary to set the values of many counters at once. This can be done with consecutive `\setcounter` statements, for example. This poses no problem, but might become tedious if there are more than three counters or if this task occurs more than once. [✉ xassocnt](#) provides some macros that can do the usual operations like stepping, refstepping, adding to, resetting

or setting counter values.

All macros concerning this feature use the first macro argument having a comma-separated list of counters. Whether there's a second argument depends on the specific nature of the operation that should be performed.

- As of version 0.9 `xassoccnt` does not check whether the names given in the first argument refer to counters.
- All macros use the extended counter macros, i.e. are aware of associated counters and step them too if their driver counter is given in the argument list. If an associated counter itself is given in the list, this one is stepped or operated on too!

`\LoopAddtoCounters`{*<counter1, counter2, ...>*}{*<counter increment/decrement>*}

Description of arguments of command `\LoopAddtoCounters`

The 2nd argument value is added (or subtracted) to the counters given in the list of the 1st argument using the `\addtocounter`.

- #1 {*<counter1, counter2, ...>*} Holds the comma separated list of counter names
- #2 {*<counter increment/decrement>*} Specifies the value to be added or subtracted
No check is performed whether **is** or **expands** to an integer value.

v0.7
2016-05-10

`\LoopResetCounters`{*<counter1, counter2, ...>*}

Description of arguments of command `\LoopResetCounters`

All counters given in the first argument are set to zero using the regular `\setcounter`. This is a shorthand version of `\LoopSetCounters`^{→P.28} for this specific case.

- #1 {*<counter1, counter2, ...>*} Holds the comma separated list of counter names

v0.7
2016-05-10

`\LoopRefstepCounters`{*<counter1, counter2, ...>*}

Description of arguments of command `\LoopStepCounters`

All counters given in the first argument are stepped using the regular `\refstepcounter` to allow labels – however, only the last counter will have the correct label reference.

! This macro is meant only to complete the number of `\Loop...Counters` but is not regarded as being really useful.

- #1 {*<counter1, counter2, ...>*} Holds the comma separated list of counter names

v0.7
2016-05-10

`\LoopSetCounters`{*counter1, counter2,...*}{*new counter value*}

Description of arguments of command `\LoopAddToCounters`

The 2nd argument value is used as new counter value added (or subtracted) to the counters given in the list of the 1st argument using the `\addtocounter`.

#1 {*counter1, counter2,...*} Holds the comma separated list of counter names

#2 {*new counter value*} Specifies the value to be set.

No check is performed whether **is** or **expands** to an integer value.

v0.7
2016-05-10

`\LoopStepCounters`{*counter1, counter2,...*}

Description of arguments of command `\LoopStepCounters`

All counters given in the first argument are stepped using the regular `\stepcounter`.

#1 {*counter1, counter2,...*} Holds the comma separated list of counter names

v0.7
2016-05-10

A more general command for doing "arbitrary" operations with counters (and more setup, for example) is

`\LoopCountersFunction`{*counter1, counter2,...*}{*counter operation macro*}

Description of arguments of command `\LoopAddToCounters`

The 2nd argument value should hold a macro with any number of arguments, but the last mandatory argument of this macro is reserved for counter name.

#1 {*counter1, counter2,...*} Holds the comma separated list of counter names

#2 A macro name that is to be called and that operates on a counter.

v0.7
2016-05-10

```

% We assume we have the counters fooctr and foobarctr
\newcommand{\showcountervalues}[2]{%
  \textcolor{#1}{\csname the#2\endcsname}% Now, an extra empty line to show the
  \values in rows
}
% Note that the 2nd argument is not given here -- it's added by the
\LoopCountersFunction macro
\LoopCountersFunction{fooctr,foobarctr}{\showcountervalues{blue}}

```

14
6

7 Coupled counters

v0.5
2016-02-27

! The features described here are very experimental and not fully implemented so far.

Occasionally there are requests where the figure or table environment should use the same counter in the sense of using continued counter values, e.g figure 1 is followed by table 2, the next figure is numbered as 3 etc.

This can be achieved with the concept of coupled counters. As usual, those counters belonging to a 'group' should be declared first in the preamble. In some sense coupled counters are similar to associated counters.

7.1 Common options for most of the coupled counter macros

name=*(name of a list)*

This option has the name of the counter group that should be coupled, say "figuretablegroup" etc.

v0.6
2016-03-05

multiple=*(true,false)*

(initially false)

This option allows to add a counter multiple times to a counter group. In general, using this style is not recommended.

v0.6
2016-03-05

7.2 Macros for declaring, adding and removing coupled counters

`\DeclareCoupledCounters`[*(options)*]{*(counter name1, counter name2, ...)*}

#1 [*(options)*]: See section 7.1 for a explanation about available options.

v0.5
2016-02-27

#2 $\{\langle\text{counter name 1, counter name2, ...}\rangle\}$: The list of counters that should be stepped together for the given counter group.

This macro is a preamble-only command.

`\DeclareCoupledCountersGroup` $\{\langle\text{counter group name}\rangle\}$

This macro defines a name for a counter group and allocates a new group list for the counter names. If the name already exists, nothing is done.

#1 $\{\langle\text{counter group name}\rangle\}$: The name of the counter group.

This macro is a preamble-only command and does not add counters to the group container. Use `\DeclareCoupledCounters`^{→P.29} or `\AddCoupledCounters` to add counters to the relevant group.

v0.5
2016-02-27

`\RemoveCoupledCounters` $[\langle\text{options}\rangle]\{\langle\text{counter name1, counter name2, ...}\rangle\}$

This removes the comma separated counter names from the coupled counter list given in the `name`^{→P.29} option.

#1 $[\langle\text{options}\rangle]$: As of version 0.9 the only recognized option is `name`^{→P.29}.

#2 $\{\langle\text{counter name 1, counter name2, ...}\rangle\}$: The list of counters that should be removed from the given counter group.

v0.5
2016-02-27

- The list name itself is still available
- If the list given by the `name`^{→P.29} option does not exist, `\RemoveCoupledCounters` issues a warning on the console and ignores this list then.

If all counters from a group name should be removed, this is equal to clearing – just use `\ClearCoupledCounters`^{→P.31} for simpler usage of this feature.

`\AddCoupledCounters` $[\langle\text{options}\rangle]\{\langle\text{counter name1, counter name2, ...}\rangle\}$

This adds the listed counter names to coupled counter list. It acts like `\DeclareCoupledCounters` but does not setup new counter groups. Please use `\DeclareCoupledCounters`^{→P.29} first, then apply `\AddCoupledCounters` later on.

#1 $[\langle\text{options}\rangle]$: See section 7.1 for a explanation about available options.

#2 $\{\langle\text{counter name 1, counter name2, ...}\rangle\}$: The list of counters that should be stepped together for the given counter group.

→P.29 v0.6
2016-03-05

If the list given by the `name`^{→P.29} option does not exist, `\AddCoupledCounters` issues a warning on the console and ignores this list then. The counters are not added to any list at all.

`\ClearCoupledCounters{<options>}`

This removes all names from the given name of a group of coupled counters.

#1 [`<options>`]: As of version 0.9 the only recognized option is `name`^{→P.29}.

After clearing a list, the coupling stops for the counters on that list (unless they are part of another list, which is possible, but not recommended). using `\AddCoupledCounters`^{→P.30} with the relevant `name`^{→P.29} option adds counters again to the list and the coupling is active again, however, for different counters (eventually).

In order to clear all coupled counter lists, use `\ClearAllCoupledCounters` instead.

- The list name itself is still available
- If the list given by the `name`^{→P.29} option does not exist, `\ClearCoupledCounters` issues a warning on the console and ignores this list then.

`\ClearAllCoupledCounters`

This removes all coupled counter groups, but not the group names, i.e. the list names can be used later on to add counter names again. In order to clear a specific list, use `\ClearCoupledCounters`.

`\IsCoupledCounterTF{<counter name>}{<true branch>}{<false branch>}`

This macro tests if a counter is under the administration of the coupled counter commands and expands to the relevant branch then. There are two short-circuit commands `\IsCoupledCounterT` and `\IsCoupledCounterF`.

`\IsCoupledCounterT{<counter name>}{<true branch>}`

This macro tests if a counter is under the administration of the coupled counter commands and executes the true branch then. There are two related commands `\IsCoupledCounterTF` and `\IsCoupledCounterF`.

`\IsCoupledCounterF{<counter name>}{<false branch>}`

This macro tests if a counter is under the administration of the coupled counter commands and executes the false branch then if this is not the case. There are two related commands `\IsCoupledCounterTF` and `\IsCoupledCounterT`.

8 Periodic counters

It might be very convenient to have counters that are automatically reset not only by a driving counter such as `\chapter` but also periodically, i.e. after a certain amount of steps – this can be achieved with the concept of periodic counters.

8.1 Commands related to periodic counters setup

`\DeclarePeriodicCounter` [$\langle \rangle$] { \langle counter name \rangle } { \langle counter treshold value \rangle }

This defines the counter given in the first mandatory argument as a periodic counter and is automatically reset if the treshold value is reached.

! The command `\DeclarePeriodicCounter` does not define a new counter, however but is the preamble-only version of `\AddPeriodicCounter`.

Please note that in case of `\addtocounter` applied to a periodic counter the value to be added leads to a modulo division such that the counter might be reset if the addition would increase the counter beyond the treshold value, the module part will be added then. In order to prevent this wrapping, use the `wrap`^{→ P.23} option to `\addtocounter`^{→ P.23}:

```
\setcounter{foocntr}{3}%
\AddPeriodicCounter{foocntr}{8}%

Value of foocntr is: \thefoocntr % Should be 3

\addtocounter{foocntr}{20} % Is it 23? No, it is 23 % 8 = 7
Value of foocntr is \thefoocntr\ now!

Adding a value of 4 again:
\addtocounter{foocntr}{4} % Is it 11? No, it is 11 % 8 = 3
Value of foocntr is \thefoocntr\ now!

Now prevent the wrapping
\addtocounter{foocntr}{10}[wrap=false] % Is it 13? Yes, it is!
Value of foocntr is \thefoocntr\ now!
```

```
Value of foocntr is: 3
Value of foocntr is 7 now!
Adding a value of 4 again: Value of foocntr is 3 now!
Now prevent the wrapping Value of foocntr is 13 now!
```

`\AddPeriodicCounter` [$\langle \rangle$] { \langle counter name \rangle } { \langle counter treshold value \rangle }

This defines the counter given in the first mandatory argument as a periodic counter and is automatically reset if the treshold value is reached.

`\RemovePeriodicCounter` [*options*] {*counter name*} {*counter treshold value*}

v0.9
2016-06-18

This defines the counter given in the first mandatory argument as a periodic counter and is automatically reset if the treshold value is reached.

#1 [*options*]

As of version 0.9, there is only one option:

`reset` = {*true/false*} (initially true)

v0.9
2016-06-18

Use 'false' to prevent the resetting of the relevant counter after removal!

#2 {*counter name*} – the name of the counter that should be no periodic counter any longer.

`\ChangePeriodicCounterCondition` [*options*] {*counter name*} {*new counter treshold value*}

v0.9
2016-06-18

This changes the counter treshold condition – the counter is reset automatically if not specified otherwise with the `reset` option.

#1 [*options*]

As of version 0.9, there is only one option: `reset`, which serves the same functionality as in `\RemovePeriodicCounter`.

#2 {*counter name*} – the name of the counter that should be no periodic counter any longer.

#3 {*new counter value treshold*} – the new value after which an automatic resetting will occur.

8.2 Commands to query for periodic counter feature

`\IsPeriodicCounterTF` {*counter name*} {*true branch*} {*false branch*}

v0.9
2016-06-18

This macro tests if a counter is under the administration of the periodic counter commands and expands to the relevant branch then. There are two short-circuit commands: `\IsPeriodicCounterT` and `\IsPeriodicCounterF`.

`\IsPeriodicCounterT` {*counter name*} {*true branch*}

v0.9
2016-06-18

This macro tests if a counter is under the administration of the periodic counter commands and expands to the *true* branch then. There are two related commands: `\IsPeriodicCounterTF` and `\IsPeriodicCounterF`.

`\IsPeriodicCounterF` {*counter name*} {*false branch*}

v0.9
2016-06-18

This macro tests if a counter is under the administration of the periodic counter commands and expands to the *false* branch then if this is not the case. There are two related commands: `\IsPeriodicCounterTF` and `\IsPeriodicCounterT`.

9 Total counters

Similarly to the package  **totcount** or the features of  **totalcount** by Axel Sommerfeldt this package provides the possibility of defining a counter that stores its finally value to the auxiliary file and starts from this value then, if not set otherwise to another value.

The declaration of a total counter is a preamble - only event and `\NewTotalDocumentCounter`^{→P.36} is a preamble-only command in order to prevent counter register confusion. If a certain existing counter should be treated with total counter features, use `\RegisterTotalDocumentCounter` instead.

 The standard $\text{\LaTeX} 2_{\epsilon}$ commands `\stepcounter`, `\addtocounter` and `\setcounter` support the specification of a total counter, but `\refstepcounter` will fail since the usage of a total counter for labelling purposes is most probably of no use (as of version 0.9)

`\RegisterTotalDocumentCounter` [*options*] {*total counter name*}

#1 [*options*]: As of version 0.9, only this option is used

`supertotal`=*<true,false>* (false)

Set this key to switch the super total counter on or off.

#2 {*total counter name*}: The name of the total counter. This must be the same like the name of an already existing counter. Internally another counter is defined which has a prefix to prevent name clashes with counter names defined by the package  **totalcount**. If the counter name does not exist, the compilation exits with an error message.

`\TotalCounterInternalName`{*counter name*}

This command reports the internal name of a total counter or the usual name if this counter is not a total one.

```
\TotalCounterInternalName{chapter}
```

```
\TotalCounterInternalName{foototal}
```

```
chapter
xassocnt@total@foototal
```

`\TotalValue`{*counter name*}

This command prints the value of a total counter or falls back to the value of the counter if this is not a total counter.



```
‘‘Total’’ value of the section non-total counter: \TotalValue{section}
```

```
Total value of the foototal total counter: \TotalValue{foototal}
```

```
“Total” value of the section non-total counter: 2
```

```
Total value of the foototal total counter: 1
```

`\IsTotalCounterTF`{*<counter name>*}{*<true branch>*}{*<>false branch>*}

This macro tests if a counter is under the administration of the total counter commands and expands to the relevant branch then. There are two short-circuit commands `\IsTotalCounterT` and `\IsTotalCounterF`.

`\IsTotalCounterT`{*<counter name>*}{*<true branch>*}

This macro checks if a counter is under the administration of the total counter commands and expands to the code in the second argument if this is true.

`\IsTotalCounterF`{*<counter name>*}{*<>false branch>*}

This macro checks if a counter is under the administration of the total counter commands and expands to the code in the second argument if this is not the case.

```
\IsTotalCounterTF{foototal}{Yes, this is a total counter}{No, this is no total }
  {counter}
```

```
\IsTotalCounterTF{section}{Yes, this is a total counter}{No, this is no total }
  {counter}
```

```
\IsTotalCounterT{foototal}{Yes, this is a total counter}
```

```
\IsTotalCounterF{page}{No, page isn't a total counter}
```

```
Yes, this is a total counter
```

```
Yes, this is a total counter
```

```
Yes, this is a total counter
```

```
No, page isn't a total counter
```

! The features of using other  aux files or a different external file as provided by  **totcount** is not (yet) supported as of version 0.9.

10 Super total counters

In addition to the concept of a total counter, there is also the possibility of using super total counters – those counters survive the reset at the beginning of a compilation, i.e. the value of a super total counter might be stepped in each run and as such the number of compilation runs etc. can be tracked. The values of the last run are persistent as long as the `.aux` file isn't deleted.

`\NewTotalDocumentCounter` [*options*] {*total counter name*}

This macro defines a new counter (which mustn't exist before of course) and puts it under control of the total counter features.

#1 [*options*]: As of version 0.9, only this option is used

`supertotal`^{→P.34} – this has the same meaning as in `\RegisterTotalDocumentCounter`^{→P.34} and defaults to `false`.

#2 {*total counter name*}: The name of the total counter. This must not be the same like the name of any already existing counter. Internally another counter is defined which has a prefix to prevent name clashes with counter names defined by the package `totalcount`.

If an already existing counter should be tracked with total counter features, use `\RegisterTotalDocumentCounter` instead.

`\IsSuperTotalCounterTF` {*counter name*} {*true branch*} {*false branch*}

This macro tests if a counter is under the administration of the super total counter commands and expands to the relevant branch then. There are two short-circuit commands `\IsSuperTotalCounterT` and `\IsSuperTotalCounterF`.

`\IsSuperTotalCounterT` {*counter name*} {*true branch*}

This macro checks if a counter is under the administration of the super total counter commands and expands to the code in the second argument if this is true.

`\IsSuperTotalCounterF` {*counter name*} {*false branch*}

This macro checks if a counter is under the administration of the super total counter commands and expands to the code in the second argument if this is not the case.

```
\IsSuperTotalCounterTF{numberofruns}{Yes, this is a super total counter}{No, this is no super total counter}
```

```
\IsSuperTotalCounterT{numberofruns}{Yes, this is a super total counter}
```

```
\IsSuperTotalCounterTF{chapter}{Yes, this is a total counter}{No, this is no super total counter}
```

No, this is no super total counter

No, this is no super total counter

10.1 The *numberofruns* counter

This package adds a counter of its own:  *numberofruns* which is a super total counter and is stepped each compilation run. It's added in `\AtBeginDocument` and can be retrieved with `\TotalValue`^{P.34}. Use the `nonnumberofruns`^{P.8} package option to prevent the definition of this counter.

11 Backup and restore of counter values

It might be necessary to interrupt the current sectioning, e.g. including another document's structure (an external paper, for example) such that the counting should start again and after finishing of the external structure the old values should be restored.

v0.4
2016-01-26

11.1 Description of backup and restoring macros for counter values

`\BackupCounterValues` [*options*] {*counter name1, counter name2,...*}

This macro adds counter names (separated by a comma) to a list and stores the current values of the counters to another list. The values are used from the current state where this command is used, not a previous or a later state is stored.

- All counters in the list will be reset to zero (after storing the values) for the next usage, unless the `resetbackup` key is set to *false*.
- Multiple specification of the same counter name is possible, but only the first occurrence will be regarded – consecutive occurrences of the same counter name are not taken into account.

v0.5
2016-02-27

`resetbackup`=*true/false* (initially true)

This key decides whether **all** counters in the backup list should be reset to zero or should keep the current value. The default value is *true*.

Please note: If a name does not belong to a counter register the compilation aborts with an error message!

Some remarks

! If a specific counter name is suffixed with an *** at its end the resetting is disabled for this particular counter, regardless whether `resetbackup` is set to true or not.

v0.4
2016-01-26

! Strangely enough, a counter name like  *foo** is possible, but `\thefoo*` would fail. Be careful about choosing counter names for new counters – just restrict yourself to the usual letters (and if really needed, using *@*)

`\RestoreAllCounterValues` [*options*]

This macro restores all stored counter values corresponding to the counter names. As of version 0.9 the optional argument isn't used and reserved for later purposes. The backup list is cleared after the restoring has been finished.

v0.5
2016-02-27



The `\RestoreAllCounterValues` was previously called `\RestoreCounterValues` – that macro is now reserved for updating only particular counters, not all in a row.

`\RestoreCounterValues` [*options*] {*counter name1,counter name2,...*}

This macro restores only the stored counter values given by the counter names. As of version 0.9 the optional argument isn't used and reserved for later purposes.

v0.5
2016-02-27

```

\captionof{figure}{A dummy figure}
\captionof{table}{A dummy table}

\BackupCounterValues{figure,table*}

\captionof{figure}{Another dummy figure}
\captionof{table}{Another dummy table}

\captionof{figure}{Even another dummy figure}
\captionof{table}{Even another dummy table}

Before restoring: \thefigure\ and \thetable

\RestoreAllCounterValues

Restored the values: \thefigure\ and \thetable

\captionof{figure}{Yet another dummy figure}
\captionof{table}{Yet another dummy table}

```

Figure 1: A dummy figure

Table 1: A dummy table

Figure 1: Another dummy figure

Table 2: Another dummy table

Figure 2: Even another dummy figure

Table 3: Even another dummy table

Before restoring: 2 and 3

Restored the values: 1 and 1

Figure 2: Yet another dummy figure

Table 2: Yet another dummy table

`\AddBackupCounter [options] {<counter name1,counter name2,...>}`

This is similar to `\BackupCounterValues`^{→P.37}, but adds the counter names to an existing global list and can be applied after `\BackupCounterValues`^{→P.37}.

`\RemoveBackupCounters` [*options*] {*counter name1, counter name2,...*}

This macro removes the counters from the list of backup counters and immediately restores the counter value unless the starred version `\RemoveBackupCounters*` is used.

If the package  `hyperref` is used, the macro `\theH...` (see Some notes on the backup features on this) is restored to the original meaning.

As of version 0.9 the optional argument isn't used and reserved for later purposes.

v0.5
2016-02-27

`\RemoveBackupCounters*` [*options*] {*counter name*}

This command is basically similar to `\RemoveBackupCounters`, but does not restore the counter value right at the place the macro is used.

As of version 0.9 the optional argument isn't used and reserved for later purposes.

v0.5
2016-02-27

11.2 Some notes on the backup features

Principally backing up counter values and restoring them later on is not really difficult – with one exception: If the  `hyperref` package is used, the counter values form up the hypertarget anchors, for example `chapter.1` for the first chapter. If the chapter counter is reset, there would be a chapter with number one again and as well an anchor name `chapter.1` –  `hyperref` will complain 'only' about this but it will put the wrong hyperlink as well, for example for the table of contents and the bookmarks – this is an undesirable feature.

However, there is a solution to this problem: The hypertarget anchors are built up from the specifications of a macro `\theH...` where the ellipses stands for the counter name. If for example `\theHchapter` is changed after a counter was reset the hypertargets will again be correct, since this will provide a different target name. `\BackupCounterValues` does this resetting automatically in an unique way and `\RestoreCounterValues` restores as well the old `\theH...` macros of all counters that are in the backup list. It tracks the number of calls to `\BackupCounterValues` and changes the relevant `\theH...` macro definitions to use unique anchor names then – this way multiple `\BackupCounterValues` calls are possible without destroying the hyperlink facilities with  `hyperref`.

12 Counter output

Once in a while it might be necessary to provide counter output not only as integer numbers, letters or Roman figures but also using binary, octal or hexadecimal number output. The  `fmtcount` package has support for this already – here are some alternatives

v0.7
2016-05-10

 None of the commands checks whether the argument refers to counter name.

`\BinaryValue{<counter name>}`

This command will print the value of the counter using binary digits.

v0.7
2016-05-
10

`\hexValue{<counter name>}`

This command will print the value of the counter using lowercase hexadecimal digits.

v0.7
2016-05-10

`\HexValue{<counter name>}`

This command will print the value of the counter using uppercase hexadecimal digits.

v0.7
2016-05-10

`\OctalValue{<counter name>}`

This command will print the value of the counter using uppercase hexadecimal digits.

v0.7
2016-05-10

13 To - Do list

- Switch to the container support for all features – this is a major task and will be done in (tiny) steps.
- Add counter group support for the  **backup** feature, i.e. define a symbolic name for a group of counters that should be controlled by the backup feature. This will allow multiple backup groups, which might be necessary.
- Better counter definition/copy counter routines → another package perhaps
- More examples
- Some macro names might be non-intuitive
- Improve documentation
- Hooks for conditionals on  **numberofruns** (see section 10.1)

If you

- find bugs
- errors in the documentation
- have suggestions
- have feature requests

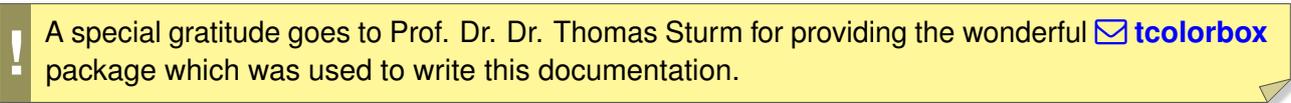
don't hesitate and contact me using my mail address: christian.hupfer@yahoo.de .

14 Acknowledgments

I would like to express my gratitudes to the developpers of fine \LaTeX packages and of course to the users at tex.stackexchange.com, especially to

- Paulo Roberto Massa Cereda
- Enrico Gregorio
- Joseph Wright
- David Carlisle
- Werner Grundlingh
- Gonzalo Medina
- Cosmo Huber (for providing the bug report with the [✉ calc](#) package.)

for their invaluable help on many questions on macros.

 A special gratitude goes to Prof. Dr. Dr. Thomas Sturm for providing the wonderful [✉ tcolorbox](#) package which was used to write this documentation.

15 Version history

- Version v0.9 2016-06-18
 - `\TotalValue`^{→P.34} is an expandable command now. v0.9
2016-06-18
 - Added the  **periodic counter** features – see Periodic counters for more information on this. v0.9
2016-06-18

- Version v0.8 2016-06-10
 - ✓ Fixed the `\SuspendCounters`^{→P.20} and `\ResumeSuspendedCounters`^{→P.21} macros – the comma separated list of counters was not used (contrary to the purpose and the documentation description). v0.8
2016-06-10
 - ✓ Additions of commands v0.8
2016-06-10
 - ▷ `\ResumeAllSuspendedCounters`^{→P.21}
 - ▷ `\CascadeSuspendCounters`^{→P.21}
 - ▷ `\DisplayResetList`^{→P.18}
 - ▷ `\ShowResetList`^{→P.18}

- Version v0.7 2016-05-10
 - ✓ Fixed a small bug in the  **xassocnt** version of `\stepcounter` v0.7
2016-05-10
 - ✓ Added some macros that support the output of binary, octal or hexadecimal (both lower/uppercase) values of counters. v0.7
2016-05-10
 - ✓ Added the `\Loop...Counters` macros that perform an action in loop on all given counter names. v0.7
2016-05-10

- Version v0.6 2016-03-05
 - ✓ The coupled counters allow to specify a counter group to which all relevant counters belong, this allows several coupled counter groups then v0.6
2016-03-05
 - ✓ Fixed a small bug within backup counter support – the resetting was not done any more v0.6
2016-03-05
 - ✓ Added the  **nonnumberofruns** package option.

- Version v0.5 2016-02-27
 - ✓ Added support (very experimental!) for the  **coupled counters** feature, see section 7 about this feature! v0.5
2016-02-27

✓ Added `\RegisterTotalDocumentCounter` and improved `\TotalValue` support

v0.5
2016-02-27

- Version v0.4 2016-01-26

✓ Added `\BackupCounterValues` and `\RestoreCounterValues` support

v0.4
2016-01-26

✓ Added `\StepDownCounter` and `\SubtractFromCounter` macros

v0.4
2016-01-26

- Version v0.3 2016-01-08

✓ Added the  `totalcounter` features similar to the packages  `totcount` or  `totalcount`

v0.3
2016-01-08

✓ Added the  `super total counter` features

2015-11-11

✓ Added the  `numberofruns` counter

2015-25-11

2015-25-11

2015-25-11

- Version v0.2 2016-11-14

Improved `\stepcounter` to remove some incompatibilities with the  `perpage`. This is only partially managed so far.

v0.2
2016-11-14

- Version v0.1 2016-11-07

A major bug fixed due to some error in usage together with  `calc` when the driven counters are not stepped any longer. The culprit was in  `assoccnt` that the counter reset list was not really disabled.

Thanks to this question <http://tex.stackexchange.com/questions/269731/calc-breaks-assoccnt> this bug was detected.

This however lead to some internal inconsistencies and it was decided to rewrite  `assoccnt` with  `expl3` and the features of the new \LaTeX 3 - Syntax.

Note: The `\DeclareAssociatedCounters` command have to be used in the preamble of the document. It's missing here for the sake of a compact example.

A Example: Total number of sections

In this example, all sections of this document are counted, i.e. the current one as well as all following ones.

```
This document has \total{totalsections} section(s)%
```

```
This document has 17 section(s)
```

B Example: Total number of subsections with suspension

In this example, the subsections of this document are counted but later on, the associatedcounter is removed from the list, so it is frozen.

```
\subsection{First dummy subsection}
SubSection counter: \thesubsection-- \number\totvalue{totalsubsections}
\subsection{Second dummy subsection}
SubSection counter: \thesubsection-- \number\totvalue{totalsubsections}

\RemoveAssociatedCounter{subsection}{totalsubsections}%
\subsection{Third dummy subsection after removing the associated counter}

SubSection counter: \thesubsection-- \number\totvalue{totalsubsections}
```

B.1 First dummy subsection

SubSection counter: B.1 – 20

B.2 Second dummy subsection

SubSection counter: B.2 – 20

B.3 Third dummy subsection after removing the associated counter

SubSection counter: B.3 – 20

B.4 Suspension of a non-associated counter

This example will show the suspension of a non-associated counter

```
\setcounter{equation}{0}%
\SuspendCounters{equation}%
\begin{equation}
E_{0} = mc^2
\end{equation}

\begin{equation}
E^2 = \left( pc \right)^2 + E_{0}^2
\end{equation}

\begin{equation}
m(v) = \frac{m_{0}}{\sqrt{1-\frac{v^2}{c^2}}}
\end{equation}
```

There are \number\value{equation}~equations in here!

$$E_0 = mc^2 \tag{0}$$

$$E^2 = (pc)^2 + E_0^2 \tag{0}$$

$$m(v) = \frac{m_0}{\sqrt{1 - \frac{v^2}{c^2}}} \tag{0}$$

There are 0 equations in here!

Index

- \AddAssociatedCounters, 10
- \AddBackupCounter, 39
- \AddCoupledCounters, 30
- \AddDriverCounter, 13
- \AddPeriodicCounter, 32
- \addtcounter, 23
- associatedtoo key, 24
- autodefine key, 9
- autodefinecounters key, 8
- \BackupCounterValues, 37
- \BinaryValue, 41
- \CascadeSuspendCounters, 21
- \ChangePeriodicCounterCondition, 33
- \ClearAllCoupledCounters, 31
- \ClearAssociatedCounters, 11
- \ClearCoupledCounters, 31
- \ClearDriverCounters, 13
- \CopyDocumentCounters, 25
- Counter
 - chapter, 21
 - foo, 10, 21
 - foo*, 37
 - foobar, 10
 - numberofruns, 3, 8, 37
- \countersresetlistcount, 17
- \DeclareAssociatedCounters, 9
- \DeclareCoupledCounters, 29
- \DeclareCoupledCountersGroup, 30
- \DeclareDocumentCounter, 23
- \DeclarePeriodicCounter, 32
- \DisplayResetList, 18
- Feature
 - associated counters, 10
 - backup, 42
 - coupled counters, 10, 44
 - periodic counter, 44
 - periodic counters, 23
 - super total counter, 45
 - totalcounter, 45
- \GetDriverCounter, 14
- \getresetlistcount, 17
- \HexValue, 41
- \hexValue, 41
- \IfInResetListF, 17
- \IfInResetListT, 17
- \IfInResetListTF, 17
- \IfIsDocumentCounterF, 17
- \IfIsDocumentCounterT, 16
- \IfIsDocumentCounterTF, 16
- initial key, 23
- \IsAssociatedCounter, 15
- \IsAssociatedToCounter, 13
- \IsCoupledCounterF, 31
- \IsCoupledCounterT, 31
- \IsCoupledCounterTF, 31
- \IsDriverCounter, 15
- \IsPeriodicCounterF, 33
- \IsPeriodicCounterT, 33
- \IsPeriodicCounterTF, 33
- \IsSuperTotalCounterF, 36
- \IsSuperTotalCounterT, 36
- \IsSuperTotalCounterTF, 36
- \IsSuspendedCounter, 16
- \IsTotalCounterF, 35
- \IsTotalCounterT, 35
- \IsTotalCounterTF, 35
- Keys
 - associatedtoo, 24
 - autodefine, 9
 - autodefinecounters, 8
 - initial, 23
 - multiple, 29
 - name, 29
 - nonumberofruns, 8

- onlycounters, 24
- reset, 33
- resetbackup, 37
- sloppy, 9
- supertotal, 34
- wrap, 23
- \LastAddedToCounter, 18
- \LastCounterValue, 19
- \LastRefSteppedCounter, 19
- \LastSetCounter, 19
- \LastSteppedCounter, 18
- \LoopAddtoCounters, 27
- \LoopCountersFunction, 28
- \LoopRefstepCounters, 27
- \LoopResetCounters, 27
- \LoopSetCounters, 28
- \LoopStepCounters, 28
- multiple key, 29
- name key, 29
- \NewDocumentCounter, 23
- \NewTotalDocumentCounter, 36
- nonnumberofruns key, 8
- \OctalValue, 41
- onlycounters key, 24
- Option
 - nonnumberofruns, 44
- Package
 - assoccnt, 6, 45
 - book, 21
 - calc, 8, 43, 45
 - cleveref, 7
 - expl3, 45
 - fmtcount, 40
 - hyperref, 7, 20, 40
 - l3keys2e, 7
 - mathtools, 8
 - perpage, 8, 45
 - tcolorbox, 7, 43
 - totalcount, 34, 36, 45
 - totcount, 6, 34, 35, 45
 - xassoccnt, 6–8, 16, 26, 44
 - xcolor, 7
 - xparse, 7, 16
 - \RegisterTotalDocumentCounter, 34
 - \RemoveAssociatedCounter, 11
 - \RemoveAssociatedCounters, 11
 - \RemoveBackupCounters, 40
 - \RemoveBackupCounters*, 40
 - \RemoveCoupledCounters, 30
 - \RemoveDriverCounter, 13
 - \RemovePeriodicCounter, 33
 - reset key, 33
 - resetbackup key, 37
 - \RestoreAllCounterValues, 38
 - \RestoreCounterValues, 38
 - \ResumeAllSuspendedCounters, 21
 - \ResumeSuspendedCounters, 21
 - \SetDocumentCounter, 23
 - \ShowResetList, 18
 - sloppy key, 9
 - \StepDownCounter, 24
 - \SubtractFromCounter, 24
 - supertotal key, 34
 - \SuspendCounters, 20
 - \SwapDocumentCounters, 25
 - \SyncCounters, 25
 - \TotalCounterInternalName, 34
 - \TotalValue, 34
 - wrap key, 23