# The dcounter package*

Alexander I. Rozhenko
rozhenko@oapmg.sscc.ru

2005/04/25

This package implements a concept of *dynamic counters*. The counter declared as dynamic is really created at the first use and receives at that moment the *count style* which was established by the `\countstyle` command. For example, if `\countstyle{section}` is established, all dynamically created counters will be subordinated to `section` counter (i.e., reset to zero when `section` counter is stepped) and their typesetting command `\the`*foo* will be equal to `\thesection.\arabic{`*foo*`}`. This package is compatible with `calc` package.

## 1 User Interface

`\DeclareDynamicCounter` To declare a dynamic counter ⟨*foo*⟩ you have to write

```
\DeclareDynamicCounter{⟨foo⟩}
```

If the ⟨*foo*⟩ counter does not exist, its name is added to the list of dynamic counters. This allows create a counter at the first use with one of the following commands

```
\setcounter{⟨foo⟩}{⟨number⟩}        \stepcounter{⟨foo⟩}
\addtocounter{⟨foo⟩}{⟨number⟩}     \refstepcounter{⟨foo⟩}
```

If the ⟨*foo*⟩ counter exists, it will emulate the dynamic style. I use the following trick for such counters: let `\the`⟨*foo*⟩ command empty and test it at the beginning of document; if it is empty, the count style of this counter is redefined on the base of dynamic style.[1] This allows work with existing counters by the same manner as with "true dynamic" counters.

`\countstyle` To specify a count style you have to use the command

```
\countstyle{⟨counter⟩}
```

The parameter ⟨*counter*⟩ have to be existing counter, or dynamic counter, or empty. Empty ⟨*counter*⟩ means the *plain* count style without subordination. If ⟨*counter*⟩ not exists and is dynamic it is created here within the previously specified count style. The default count style is the plain style.

*This file has version number v1.2, last revised 2005/04/25.

[1] This trick was added in version 1.2 of the package.

The `\counstyle` command has an optional parameter useful for special purposes. If you want to create some counters in another style that is specified by `\countstyle` command, you can write

`\countstyle[`⟨*list of counters*⟩`]{`⟨*another counter*⟩`}`

Here ⟨*list of counters*⟩ is the list of comma separated counters whose count style you want to subordinate to ⟨*another counter*⟩. This command creates all undefined counters of the list. The list may contain not only undefined counters but also existing counters. If counter in the list exists, its count style will be modified to be subordinated to ⟨*another counter*⟩. Note, that if this counter was subordinated before to any other counter, *the previous subordination will be rejected*. For example, let you want to use the `book` document class and set `\Roman` enumeration of chapters, independent arabic enumeration of sections and to subordinate enumeration of figures, tables and equations to the *section* counter. You can write

```
\documentclass{book}
\usepackage{dcounter}
\renewcommand\thechapter{\Roman{chapter}}
\countstyle[section]{}
\countstyle[figure,table,equation]{section}
```

After that the *chapter* counter will not affect on *section* counter, and all figure, table, and equation numbers will typeset as `\thesection.\arabic{`*foo*`}`.

`\DynamicCount` The command

`\DynamicCount{`⟨*counter*⟩`}`

sets the count style for ⟨*counter*⟩ exactly the same as for dynamically created counters and creates this counter if it is undefined. This command is internally used in emulation of dynamic counters and in the `\countstyle` command with optional parameter. Since version 1.2, this command is obsolete, but it is saved for backward compatibility.

**Note**. All described commands are used in the preamble.

# 2 The Basic Implementation Part

`\DCNT@list` `\DCNT@elist` We begin from the initialization of the list of dynamic counters. `\DCNT@list` contains a list of undeclared counters and `\DCNT@elist` contains a list of existing counters that are declared as dynamic counters.

```
⟨*package⟩
\def\DCNT@list{}
\def\DCNT@elist{}
\@onlypreamble\DCNT@elist
```

Their items will have the form `\@elt{`⟨*counter*⟩`}`

`\DCNT@in` The macro `\DCNT@in{`⟨*list*⟩`}{`⟨*yes*⟩`}{`⟨*no*⟩`}` tests the list of counters ⟨*list*⟩ to contain the counter `\DCNT@foo` and after testing executes ⟨*yes*⟩-sequence if `\DCNT@foo` found or ⟨*no*⟩-sequence if not. To restrict the scope of internal modifications made by this macro we always enclose it into a group. While processing the list of counters the command executes `\DCNT@noteq{`⟨*counter*⟩`}` hook for every counter which name is distinct from the tested name.

```
\def\DCNT@in#1#2#3{\@tempswafalse
  \let\@elt\DCNT@elt #1%
  \if@tempswa #2\else #3\fi
}
\def\DCNT@elt#1{\def\DCNT@name{#1}%
  \ifx\DCNT@name\DCNT@foo \@tempswatrue \else \DCNT@noteq{#1}\fi
}
```

`\DCNT@define` The command `\DCNT@define{`⟨*command*⟩`}{`⟨*foo*⟩`}` tests the counter ⟨*foo*⟩ to be undefined and, if it is true, tries to create it dynamically. After that it executes ⟨*command*⟩ with the ⟨*foo*⟩ parameter.

```
\def\DCNT@define#1#2{%
  \@ifundefined{c@#2}%
    {{\edef\DCNT@foo{#2}\let\DCNT@noteq\@gobble
      \DCNT@in\DCNT@list{\newcounter{#2}\DCNT@the{#2}}{}%
    }}{}%
  #1{#2}%
}
```

`\DCNT@the` The command `\DCNT@the{`⟨*foo*⟩`}` redefines `\the`⟨*foo*⟩ command to typeset it in the count style subordinated to `\DCNT@main` counter. It also adds the name ⟨*foo*⟩ to the resetting list of `\DCNT@main` counter.

```
\def\DCNT@the#1{%
  \ifx\DCNT@main\@empty
    \expandafter\xdef\csname the#1\endcsname
      {\noexpand\@arabic \expandafter\noexpand \csname c@#1\endcsname}%
  \else
    \expandafter\xdef\csname the#1\endcsname
      {\expandafter\noexpand \csname the\DCNT@main\endcsname
      .\noexpand\@arabic \expandafter\noexpand \csname c@#1\endcsname}%
    \@addtoreset{#1}\DCNT@main
  \fi
}
\let\DCNT@main\@empty
```

# 3 The Preamble Only Commands

`\DeclareDynamicCounter` The following command tests the counter and adds it to the list of dynamic counters if it does not exist or to the list of emulated counters if it already exists. In the last case, `\the`⟨*counter*⟩ command is defined as empty command. It will be redefined later at the beginning of document.

```
\newcommand*{\DeclareDynamicCounter}[1]{%
  \begingroup
    \edef\DCNT@foo{#1}%
    \ifx\DCNT@foo\@empty
      \PackageError{dcounter}%
        {Cannot declare a dynamic counter with empty name}{}%
    \fi
    \let\DCNT@noteq\@gobble
    \@ifundefined{c@#1}%
      {\DCNT@in\DCNT@list{}{\@cons\DCNT@list{{#1}}}}%
      {\DCNT@in\DCNT@elist{}{\@cons\DCNT@elist{{#1}}}%
       \expandafter\global\expandafter\let
         \csname the#1\endcsname\@empty}%
  \endgroup
}
\@onlypreamble\DeclareDynamicCounter
```

`\countstyle` Now we implement `\countstyle` command which redefines `\DCNT@main` macro to new main counter. It tests the counter to be defined and tries to define it if not.

```
\newcommand{\countstyle}{\@ifnextchar[{\DCNT@lcstyle}{\DCNT@cstyle}}
\@onlypreamble\countstyle
\def\DCNT@cstyle#1{\edef\DCNT@foo{#1}%
  \ifx\DCNT@foo\@empty \else
    \DCNT@define\@gobble{#1}%
    \@ifundefined{c@#1}{\@nocounterr{#1}}{}%
  \fi
  \let\DCNT@main\DCNT@foo
}
\@onlypreamble\DCNT@cstyle
```

The special variant of this command with optional parameter locally sets the special count style and redefines all counters in list via the `\DynamicCount` command.

```
\def\DCNT@lcstyle[#1]#2{%
  {\DCNT@cstyle{#2}\@for\@tempa:=#1\do{\DynamicCount\@tempa}}%
}
\@onlypreamble\DCNT@lcstyle
```

`\DynamicCount` The macro `\DynamicCount{`⟨*foo*⟩`}` modifies the count style of the counter ⟨*foo*⟩ and defines this counter if it is undefined.

```
\newcommand*{\DynamicCount}[1]{%
  \@ifundefined{c@#1}%
    {\newcounter{#1}}%
```

If the counter is already defined, we check all resetting lists and remove all references to this counter.

```
    {{\edef\DCNT@foo{#1}\let\DCNT@noteq\DCNT@add
      \let\@elt\DCNT@remove \cl@@ckpt
    }}%
  \DCNT@the{#1}%
}
```

```
\@onlypreamble\DynamicCount
```

`\DCNT@remove` The `\DCNT@remove{`⟨*foo*⟩`}` command removes all references to `\DCNT@foo` counter from the `\cl@`⟨*foo*⟩ list of counters.

```
\def\DCNT@remove#1{\expandafter\DCNT@remlist\csname cl@#1\endcsname}
\def\DCNT@remlist#1{%
  {\let\@tempa\@empty \DCNT@in#1{\global\let#1\@tempa}{}}%
}
\@onlypreamble\DCNT@remove
\@onlypreamble\DCNT@remlist
```

`\DCNT@add` The `\DCNT@add{`⟨*counter*⟩`}` command locally adds `\@elt{`⟨*counter*⟩`}` to `\@tempa`.

```
\def\DCNT@add#1{%
  \let\@elt\relax\edef\@tempa{\@tempa\@elt{#1}}\let\@elt\DCNT@elt
}
\@onlypreamble\DCNT@add
```

`\DCNT@eltemu` The `\DCNT@emu{`⟨*counter*⟩`}` command test `\the`⟨*counter*⟩ command to be empty and redefines the counter in the dynamic style. This command is applied to all existing counters that are emulated as dynamic counters.

```
\def\DCNT@emu#1{%
  \expandafter\ifx\csname the#1\endcsname\@empty
    \DynamicCount{#1}\fi
}
\@onlypreamble\DCNT@emu
```

# 4 Final Modifications

Finally, we modify `\setcounter` and `\addtocounter` commands. We do it at the beginning of the document to avoid conflict with the `calc` package. If the list of dynamic counters is empty, we delete all commands of the package. We also define all dynamically emulated counters if their `\the` command is empty.

```
\AtBeginDocument{%
  \ifx\DCNT@list\@empty
    \@onlypreamble\DCNT@list
    \@onlypreamble\DCNT@in
    \@onlypreamble\DCNT@elt
    \@onlypreamble\DCNT@define
    \@onlypreamble\DCNT@the
    \@onlypreamble\DCNT@main
    \@onlypreamble\DCNT@name
    \@onlypreamble\DCNT@foo
    \@onlypreamble\DCNT@noteq
  \else
    \let\DCNT@setcounter\setcounter
    \def\setcounter{\DCNT@define\DCNT@setcounter}
    \let\DCNT@addtocounter\addtocounter
    \def\addtocounter{\DCNT@define\DCNT@addtocounter}
```

```
  \fi
  {\let\@elt\DCNT@emu \DCNT@elist}%
}
```
⟨/package⟩