The counterz package*

Christopher McClain†

Released 2023/05/19

Abstract

The counterz package provides additional tools for manipulating counters. The package facilitates the use of stealth prefixes for counter names in order to help distinguish between counters from multiple input files. The package also provides a means to generate random counters and save such counter values for future typesetting.

Contents

1 Introduction

1.1 About

This project emerged from the author's frequent use of LaTeX counters as traditional integer type variables when generating mathematics documents with random elements. While pdfTeX primitives such as `\pdfuniformdeviate` may be used to generate random integers, these integer values will be randomized with every typesetting. The counterz package provides a way to save the values of counters. Another .tex file is created so that, if desired, it can be inputted upon a subsequent typesetting in order to initialize the counters with the previously generated values. A boolean variable and accompanying commands allow an author to toggle between reusing and rerandomizing counters.

*This file describes version v1.0.0, last revised 2023/05/19.

†E-mail: christopher.mcclain@mail.wvu.edu

One of the consequences of preloading counter values in large projects with multiple source files is that one must take care to use distinct counter names throughout all of the different files. If the file Main.tex inputs File1.tex and File2.tex, and both input files define the counter *mycounter*, then this could result in typesetting errors. One way to address this problem is to prefix every counter name with the file name or some other marker so that the counter names will actually be distinct. For example, *File1mycounter* is distinct from *File2mycounter*. Very long counter names, however, can make code difficult to read and hinder consistent application of this practice. The counterz package provides a way to stealthily define and recall such prefixes so that the shorter non-prefixed names can be used for the manipulation, recall, and typesetting of counters.

1.2 License

1.3 Installation

This package may be installed by copying the file counterz.sty to your local texmf directory. The code and documentation may also be generated from counterz.dtx. Typesetting the documentation requires the package hypdoc which is included in TEX distributions and at The Comprehensive TeX Archive Network.

2 User Guide

To use this package, include the following line in the preamble of your document:

```
\usepackage{counterz}
```

The package counterz loads the packages etoolbox and makecmds, both of which are included in TEX distributions and at The Comprehensive TeX Archive Network.

2.1 Counter Prefixes

`\setcounterprefix`
`\clearcounterprefix`

Counter prefixes are stored in an internal macro whose default value is an empty string. The command `\setcounterprefix{`⟨*prefix*⟩`}` is used to change this value. For example, to change the prefix to *PurpleMonkey*, use

```
\setcounterprefix{PurpleMonkey}
```

and to change it from *PurpleMonkey* to *Dishwasher*, use

```
\setcounterprefix{Dishwasher}
```

The command `\clearcounterprefix` returns the prefix to its empty default:

```
\clearcounterprefix
```

2.2 Manipulating Counters

\xnewcounter
\xprovidecounter
\xsetcounter
\xaddtocounter
\xvalue

The command `\xnewcounter{`⟨*countername*⟩`}` creates a counter with a prefixed name. The command `\xsetcounter{`⟨*countername*⟩`}{`⟨*integer*⟩`}` assigns the specified value to the counter with the prefixed name. For example, suppose that the file BoringFile1.tex contains the following:

```
\xnewcounter{bestcounterever}
\xsetcounter{bestcounterever}{100}
```

and suppose that the file BoringFile2.tex contains the following:

```
\xnewcounter{bestcounterever}
\xsetcounter{bestcounterever}{-29}
```

and, finally, suppose that the file Main.tex contains (in part) the following:

```
\setcounterprefix{PurpleMonkey}
\input{BoringFile1}
\setcounterprefix{Dishwasher}
\input{BoringFile2}
```

Then typesetting Main.tex will create a counter *PurpleMonkeybestcounterever* with the value 100 and a counter *Dishwasherbestcounterever* with the value −29. By using commands `\xnewcounter` and `\xsetcounter` instead of `\newcounter` and `\setcounter`, BoringFile1.tex and BoringFile2.tex may be written independently without considering any counter name conflicts. The distinction between the counters is determined by the prefixes defined in the file Main.tex. By changing prefixes, Main.tex can even input the same file multiple times without conflict.

The commands `\xprovidecounter`, `\xaddtocounter`, and `\xvalue` are likewise prefix versions of commands `\providecounter`, `\addtocounter`, and `\value`, respectively. When the prefix is empty, the commands expand like their standard counterparts. (Note: `\providecounter` defines a counter if it has not already been defined. See the documentation for the package makecmds for details.)

2.3 Conditional Statements

\ifctrequal
\ifctrless
\ifctrmore

The command `\ifctrequal{`⟨*counter1*⟩`}{`⟨*counter2*⟩`}{`⟨*foo*⟩`}{`⟨*bar*⟩`}` uses the command `\xvalue` to compare the values of the (prefixed) counters and then executes ⟨*foo*⟩ if the values are equal and otherwise executes ⟨*bar*⟩. The commands `\ifctrless` and `\ifctrmore` work analogously, based on whether the value of prefixed ⟨*counter1*⟩ is less than that of of prefixed ⟨*counter2*⟩ or more than that of prefixed ⟨*counter2*⟩, respectively. Consider the example code

```
\setcounterprefix{TigerTiger}
\xnewcounter{Small}
\xsetcounter{Small}{7}
\xnewcounter{Large}
\xsetcounter{Large}{11}
\ifctrequal{Small}{Large}{January}{February}
\ifctrless{Small}{Large}{March}{April}
\ifctrmore{Small}{Large}{May}{June}
```

which produces the output

February March June

because the value of the counter *TigerTigerSmall* is 7 which is less than 11, the value of the counter *TigerTigerLarge*.

\ifctrzero
\ifctrneg
\ifctrpos

The command `\ifctrzero{`⟨*counter*⟩`}{`⟨*foo*⟩`}{`⟨*bar*⟩`}` executes ⟨*foo*⟩ if the value of the (prefixed) counter is zero and otherwise executes ⟨*bar*⟩. The commands `\ifctrneg` and `\ifctrpos` work analogously based on whether the value is negative or positive, respectively. The example code

```
\setcounterprefix{TigerTiger}
\xprovidecounter{Small}
\xsetcounter{Small}{7}
\ifctrzero{Small}{January}{February}
\ifctrneg{Small}{March}{April}
\ifctrpos{Small}{May}{June}
```

produces the output

February April May

because the value of the counter *TigerTigerSmall* is 7 which is positive (and thus nonzero, as well).

2.4 Displaying Counters

\xarabic
\xroman
\xRoman
\xalph
\xAlph
\xfnsymbol

The command `\xarabic{`⟨*counter*⟩`}` is simply a prefix version of the standard display command `\arabic`. The commands `\xroman`, `\xRoman`, `\xalph`, `\xAlph`, and `\xfnsymbol` are likewise prefix versions of the standard display commands `\roman`, `\Roman`, `\alph`, `\Alph`, and `\fnsymbol`, inheriting the restrictions of their parent commands. Note that the code

```
\setcounterprefix{Sneaky}
\xprovidecounter{Pete}
\xsetcounter{Pete}{42}
\arabic{Pete}
```

produces an error because the counter *Pete* is not defined, but the code

```
\setcounterprefix{Sneaky}
\xprovidecounter{Pete}
\xsetcounter{Pete}{42}
\xarabic{Pete}
```

produces the output

42

which is the value of the counter *SneakyPete*. The code

```
\setcounterprefix{Sneaky}
\xprovidecounter{Pete}
\xsetcounter{Pete}{42}
\clearcounterprefix
\xarabic{Pete}
```

also generates error because the final line is trying to use the undefined counter *Pete* after the prefix was returned to its default value.

In addition to prefix versions of the standard display commands, the package counterz defines some variants of `\xarabic` that are useful in the display of mathematical expressions. For example, consider the following code:

```
\xprovidecounter{a}
\xsetcounter{a}{5}
\xprovidecounter{b}
\xsetcounter{b}{0}
\xprovidecounter{c}
\xsetcounter{c}{-7}
$\xarabic{a}+\xarabic{b}+\xarabic{c}$
```

which produces

$5 + 0 + -7$

Using `\arabicx` causes the expression to contain the consecutive pair $+-$. The

`\xsigned` command `\xsigned{`⟨*counter*⟩`}` is like `\xarabic` except that nonnegative values are preceded by a plus sign "+". The code

```
$\xarabic{a}\xsigned{b}\xsigned{c}$
```

produces

$5 + 0 - 7$

`\xsignednz` If we wish to suppress the 0, we can instead use the command `\xsignednz{`⟨*counter*⟩`}`

`\xarabicnz` which is a nonzero version of `\xsigned` and, if desired or necessary, the command `\xarabicnz{`⟨*counter*⟩`}` which is a nonzero version of `\xarabic`. The code

```
$\xarabicnz{a}\xsignednz{b}\xsignednz{c}$
```

produces

$5 - 7$

`\xnegof` The package also contains variants of these commands for displaying the negatives of counters, as demonstrated by the following code:

`\xnegofnz`

`\xnegsigned`

`\xnegsignednz`

```
\xprovidecounter{d}
\xsetcounter{d}{-2}
$\xarabic{a}\xsigned{b}\xsigned{c}=\xarabic{d}$\\
$\xnegof{d}=\xnegof{a}\xnegsigned{b}\xnegsigned{c}$\\
$\xnegofnz{d}=\xnegofnz{a}\xnegsignednz{b}\xnegsignednz{c}$
```

which produces

$5 + 0 - 7 = -2$
$2 = -5 - 0 + 7$
$2 = -5 + 7$

The preceding commands for displaying values related to counters were created by using some other commands that we make available in case they prove useful.

`\xabsof` The command `\xabsof{`⟨*counter*⟩`}` prints the absolute value of ⟨*counter*⟩. The `\xsignof` command `\xsignof{`⟨*counter*⟩`}` prints a minus sign "-" if ⟨*counter*⟩ is negative `\xnegsignof` and otherwise prints a plus sign "+". (Note that the latter case includes the value zero.) The command `\xnegsignof{`⟨*counter*⟩`}` prints a plus sign "+" if ⟨*counter*⟩ is negative and otherwise prints a minus sign "-". (Note that the former case includes the value zero.)

`\xabsofcoef` Additional variants of these commands suppress certain output, as is con- `\xsignofcoef` ventional when using integers as coefficients in algebraic expressions. The com- `\xnegsignofcoef` mand `\xabsofcoef{`⟨*counter*⟩`}` prints the absolute value of ⟨*counter*⟩ except that it suppresses the values of 1 and 0. The command `\xsignofcoef{`⟨*counter*⟩`}` prints the sign of ⟨*counter*⟩ if the value of ⟨*counter*⟩ is nonzero. The command `\xnegsignofcoef{`⟨*counter*⟩`}` prints the opposite sign of ⟨*counter*⟩ if the value of ⟨*counter*⟩ is nonzero. These commands are used to build versions of `\xarabic` and `\xsigned` specific to typesetting coefficients, as we now illustrate.

Consider the following code

```
\xprovidecounter{a0}
\xsetcounter{a0}{-10}
\xprovidecounter{a1}
\xsetcounter{a1}{1}
\xprovidecounter{a2}
\xsetcounter{a2}{-5}
\xprovidecounter{a3}
\xsetcounter{a3}{-1}
\xprovidecounter{a4}
\xsetcounter{a4}{0}
\xprovidecounter{a5}
\xsetcounter{a5}{11}
$\xarabic{a5}x^5 + \xarabic{a4}x^4 + \xarabic{a3}x^3 + \xarabic{a2}x^2
                 + \xarabic{a1}x + \xarabic{a0} = 42$
```

and its output

$11x^5 + 0x^4 + -1x^3 + -5x^2 + 1x + -10 = 42$

\xcoef
\xsignedcoef

We seek a better way to handle the coefficients, especially 1 and −1. The command `\xcoef{`⟨*counter*⟩`}` prints the value of ⟨*counter*⟩ except that it suppresses the values of 1, 0, and -1, printing a minus sign "-" in the latter case. The command `\xsignedcoef{`⟨*counter*⟩`}` is like `\xcoef` except that positive values are preceded by a plus sign "+". We use these to write the code

```
$\xarabic{a5}x^5 + \xarabic{a4}x^4 + \xarabic{a3}x^3 + \xarabic{a2}x^2
                 + \xarabic{a1}x + \xarabic{a0} = 42$\\
$\xcoef{a5}\ifctrzero{a5}{}{x^5} \xsignedcoef{a4}\ifctrzero{a4}{}{x^4}
   \xsignedcoef{a3}\ifctrzero{a3}{}{x^3} \xsignedcoef{a2}\ifctrzero{a2}{}{x^2}
   \xsignedcoef{a1}\ifctrzero{a1}{}{x} \xsignednz{a0} = 42$
```

whose output is

$11x^5 + 0x^4 + -1x^3 + -5x^2 + 1x + -10 = 42$
$11x^5 - x^3 - 5x^2 + x - 10 = 42$

\xnegcoef
\xnegsignedcoef

The command `\xnegcoef{`⟨*counter*⟩`}` prints the negative of the value of ⟨*counter*⟩ except that it suppresses the values of 1, 0, and -1, printing a "-" in the latter case. The command `\xnegsignedcoef{`⟨*counter*⟩`}` is like `\xnegcoef` except that positive values are preceded by a plus sign "+". We use these to write the code

```
$\xcoef{a5}\ifctrzero{a5}{}{x^5} \xsignedcoef{a4}\ifctrzero{a4}{}{x^4}
   \xsignedcoef{a3}\ifctrzero{a3}{}{x^3} \xsignedcoef{a2}\ifctrzero{a2}{}{x^2}
   \xsignedcoef{a1}\ifctrzero{a1}{}{x} \xsignednz{a0} = 42$\\
$\xcoef{a5}\ifctrzero{a5}{}{x^5} \xsignedcoef{a4}\ifctrzero{a4}{}{x^4}
   \xsignedcoef{a2}\ifctrzero{a2}{}{x^2} \xsignednz{a0}
   = \xnegcoef{a3}\ifctrzero{a3}{}{x^3} \xnegsignedcoef{a1}\ifctrzero{a1}{}{x}
   +42$
```

whose output is

$11x^5 - x^3 - 5x^2 + x - 10 = 42$
$11x^5 - 5x^2 - 10 = x^3 - x + 42$

As the reader has probably already observed in the code above, these display commands appear to be less efficient than a manual adjustment of signs and numbers. For fixed, known values of counters, this assessment is correct. The real utility of these commands is not apparent until they are combined with randomly generated counter values.

2.5 Random Counters

\xrandsetcounter
\xrandaddtocounter

We first define random versions of `\xsetcounter` and `\xaddtocounter`. The command `\xrandsetcounter{`⟨*counter*⟩`}{`⟨*min*⟩`}{`⟨*max*⟩`}` assigns to the prefixed ⟨*counter*⟩ a random integer value between ⟨*min*⟩ and ⟨*max*⟩. Analogously, the command `\xrandaddtocounter{`⟨*counter*⟩`}{`⟨*min*⟩`}{`⟨*max*⟩`}` adds to the prefixed ⟨*counter*⟩ a random integer value between ⟨*min*⟩ and ⟨*max*⟩. The following code

may be used to produce an expression in the form $ax + b$, where a and b are random integers between -10 and 10:

```
\xprovidecounter{a}
\xprovidecounter{b}
\xrandsetcounter{a}{-10}{10}
\xrandsetcounter{b}{-10}{10}
$\xcoef{a}\ifctrzero{a}{\xarabic{b}}{x \xsignednz{b}}$
```

Organized in the following table are sixty instances of output that are randomly generated by the typesetting of this document:

$-7x - 3$	$6x$	$6x - 4$	$-x + 5$	7	$4x - 6$
$2x - 10$	$8x + 2$	$-6x - 7$	$2x + 5$	$-2x + 1$	$-8x$
$9x + 8$	$9x + 7$	0	$-3x + 1$	$-2x + 4$	$x - 2$
$-4x - 4$	$x + 6$	$-8x - 10$	$3x - 6$	$6x + 8$	$x - 9$
$x - 10$	$-9x - 2$	$7x$	$3x + 3$	$9x - 6$	$4x + 1$
$-6x - 3$	$-3x - 1$	$9x + 3$	$2x + 2$	$-8x - 8$	$-4x + 9$
$7x - 7$	$-8x + 2$	$2x - 6$	$3x + 1$	$-2x + 3$	$8x - 5$
$-8x - 8$	$-2x - 9$	$9x - 10$	$3x - 10$	$8x + 7$	$-4x - 4$
$-3x - 1$	$3x + 7$	$-10x + 10$	$8x - 2$	$-x - 4$	$-10x - 4$
$-2x - 10$	$4x + 7$	$6x - 5$	$-9x - 4$	$8x + 3$	$-4x - 6$

If our document contains randomly generated counters, but we wish to typeset the document again without changing those values, then we need a way to save those values. The counterz package offers the following solution: a file ⟨*jobname*⟩.counters.tex may be created during the typesetting process to store the necessary information. For example, if the document is named Yellowdog.tex, then the previously generated counters and their assigned values will be stored the file Yellowdog.counters.tex. The command `\opencountersfile` creates and opens the write stream to this file. The author only has to include this command once, prior to any commands used to save the counter values.

`\opencountersfile`

`\xsavecounter` The command `\xsavecounter{`⟨*counter*⟩`}` "saves" the value of *counter* by writing to the file ⟨*jobname*⟩.counters.tex the relevant `\providecounter` and `\setcounter` commands. The commands written to the file explicitly include the necessary counter prefixes, and consequently an author can, if necessary, manually find in the file the specific assignment for any counter. The counters file may then be inputted near the beginning of a subsequent typesetting to preassign all of the values.

`\randomizectr` `\norandomizectr` `\ifrandomizectr` In order to effectively manage the options of randomizing counter values or reusing counter values, the counterz package offers the commands `\randomizectr` and `\norandomizectr` that toggle an internal boolean variable, and a conditional `\ifrandomizectr{`⟨*foo*⟩`}{`⟨*bar*⟩`}` that executes ⟨*foo*⟩ when the boolean is true and otherwise executes ⟨*bar*⟩. For example, a document named Yellowdog.tex might include the code

```
\ifrandomizectr{}{input{Yellowdog.counters}}
```

to determine whether to preload previously stored counter values.

\xrandprovidecounter
\xrandprovidecounternz

The command \xrandprovidecounter{⟨*counter*⟩}{⟨*min*⟩}{⟨*max*⟩} combines the four commands \xprovidecounter, \ifrandomizectr, \xrandsetcounter, and \xsavecounter to define a command that creates ⟨*counter*⟩ if it has not already been defined and, if the document is randomized, assigns to ⟨*counter*⟩ a random integer value between ⟨*min*⟩ and ⟨*max*⟩ and saves this value to the counters file. The command \xrandprovidecounternz is like \xrandprovidecounter except that the generated value is nonzero. Suppose that Neverending.tex contains the code

```
\randomizectr
\ifrandomizectr{\opencountersfile}{}
\setcounterprefix{Southern}
\xrandprovidecounternz{Oracle}{-10}{10}
\xcoef{Oracle}x+42
```

After typesetting once, the resulting document might display an expression such as $-9x + 42$ and print to Neverending.counters.tex the line

```
\providecounter {SouthernOracle} \setcounter {SouthernOracle}{-9}
```

After typesetting a second time, the resulting document might display $4x+42$ and print to Neverending.counters.tex the line

```
\providecounter {SouthernOracle} \setcounter {SouthernOracle}{4}
```

If, however, the command \randomizectr is replaced by \norandomizectr, then a third typesetting will leave both the displayed text and the counters file unchanged. (Tip: Users who are concerned about accidental randomization might create a terminal prompt with the commands \typeout and \typein to input the randomization preference, as an added layer of security.)

3 Implementation

The counterz package loads the two packages etoolbox and makecmds for the use of conditional tests (boolean and numerical) and the macro \providecounter.

```
⟨*package⟩
\ProvidesPackage{counterz}[2023/05/19 v1.0.0 Additional tools for counters]
\RequirePackage{etoolbox,makecmds}
```

3.1 Counter Prefixes

\@counterz@counterprefix
\setcounterprefix
\clearcounterprefix

The default expansion of \@counterz@counterprefix is null, but it can be changed with the commands \setcounterprefix and \clearcounterprefix.

```
\newcommand{\@counterz@counterprefix}{}
\newcommand{\setcounterprefix}[1]{\renewcommand{\@counterz@counterprefix}{#1}}
\newcommand{\clearcounterprefix}{\setcounterprefix{}}
```

3.2 Manipulating Counters

\xnewcounter
\xprovidecounter
\xsetcounter
\xaddtocounter
\xvalue

These commands are prefix versions of commands `\newcounter`, `\providecounter`, `\setcounter`, `\addtocounter`, and `\value`, respectively. The creation, modification, or use of the counters is carried out on a prefixed version of the specified counter name. When `\@counterz@counterprefix` is null, the commands expand like their standard counterparts.

```
\newcommand{\xnewcounter}[1]{\newcounter{\@counterz@counterprefix #1}}
\newcommand{\xprovidecounter}[1]{\providecounter{\@counterz@counterprefix #1}}
\newcommand{\xsetcounter}[2]{\setcounter{\@counterz@counterprefix #1}{#2}}
\newcommand{\xaddtocounter}[2]{\addtocounter{\@counterz@counterprefix #1}{#2}}
\newcommand{\xvalue}[1]{\value{\@counterz@counterprefix #1}}
```

3.3 Conditional Statements

The following commands provide if-then-else constructs analogous to those in the package etoolbox. The notable difference is that the arguments are counter names. The command `\xvalue` is used to determine the values of the counters, so that the stored prefix is applied to the specified counter names before execution.

\ifctrequal

`\ifctrequal{`⟨*counter1*⟩`}{`⟨*counter2*⟩`}{`⟨*foo*⟩`}{`⟨*bar*⟩`}` executes ⟨*foo*⟩ if the value of ⟨*counter1*⟩ is equal to the value of ⟨*counter2*⟩ and otherwise executes ⟨*bar*⟩.

```
\newcommand{\ifctrequal}[4]{\ifnumequal{\xvalue{#1}}{\xvalue{#2}}{#3}{#4}}
```

\ifctrless

`\ifctrless{`⟨*counter1*⟩`}{`⟨*counter2*⟩`}{`⟨*foo*⟩`}{`⟨*bar*⟩`}` executes ⟨*foo*⟩ if the value of ⟨*counter1*⟩ is less than the value of ⟨*counter2*⟩ and otherwise executes ⟨*bar*⟩.

```
\newcommand{\ifctrless}[4]{\ifnumless{\xvalue{#1}}{\xvalue{#2}}{#3}{#4}}
```

\ifctrmore

`\ifctrmore{`⟨*counter1*⟩`}{`⟨*counter2*⟩`}{`⟨*foo*⟩`}{`⟨*bar*⟩`}` executes ⟨*foo*⟩ if the value of ⟨*counter1*⟩ is more than the value of ⟨*counter2*⟩ and otherwise executes ⟨*bar*⟩.

```
\newcommand{\ifctrmore}[4]{\ifnumless{\xvalue{#2}}{\xvalue{#1}}{#3}{#4}}
```

\ifctrzero

`\ifctrzero{`⟨*counter*⟩`}{`⟨*foo*⟩`}{`⟨*bar*⟩`}` executes ⟨*foo*⟩ if the value of ⟨*counter*⟩ is zero and otherwise executes ⟨*bar*⟩.

```
\newcommand{\ifctrzero}[3]{\ifnumequal{\xvalue{#1}}{0}{#2}{#3}}
```

\ifctrneg

`\ifctrneg{`⟨*counter*⟩`}{`⟨*foo*⟩`}{`⟨*bar*⟩`}` executes ⟨*foo*⟩ if the value of ⟨*counter*⟩ is negative and otherwise executes ⟨*bar*⟩.

```
\newcommand{\ifctrneg}[3]{\ifnumless{\xvalue{#1}}{0}{#2}{#3}}
```

\ifctrpos

`\ifctrpos{`⟨*counter*⟩`}{`⟨*foo*⟩`}{`⟨*bar*⟩`}` executes ⟨*foo*⟩ if the value of ⟨*counter*⟩ is positive and otherwise executes ⟨*bar*⟩.

```
\newcommand{\ifctrpos}[3]{\ifnumless{\xvalue{#1}}{1}{#3}{#2}}
```

3.4 Displaying Counters

\xarabic
\xroman
\xRoman
\xalph
\xAlph
\xfnsymbol

These commands include prefix versions of the standard display commands.

```
\newcommand{\xarabic}[1]{\arabic{\@counterz@counterprefix #1}}
\newcommand{\xroman}[1]{\roman{\@counterz@counterprefix #1}}
\newcommand{\xRoman}[1]{\Roman{\@counterz@counterprefix #1}}
\newcommand{\xalph}[1]{\alph{\@counterz@counterprefix #1}}
\newcommand{\xAlph}[1]{\Alph{\@counterz@counterprefix #1}}
\newcommand{\xfnsymbol}[1]{\fnsymbol{\@counterz@counterprefix #1}}
```

The following commands likewise apply the stored prefix to the counter name. These commands are designed to aid in the typesetting of counter values within algebraic expressions while observing particular conventions about the display of numbers and their and their signs.

\xabsof

`\xabsof{`⟨*counter*⟩`}` prints the absolute value of ⟨*counter*⟩.

```
\newcommand{\xabsof}[1]{%
   \ifctrneg{#1}{%
      \the \numexpr 0 - \xvalue{#1} \relax%
   }{%
      \xarabic{#1}%
   }%
}
```

\xsignof

`\xsignof{`⟨*counter*⟩`}` prints a minus sign "-" if ⟨*counter*⟩ is negative and otherwise prints a plus sign "+". Note that the latter case includes the value zero.

```
\newcommand{\xsignof}[1]{\ifctrneg{#1}{-}{+}}
```

\xnegsignof

`\xnegsignof{`⟨*counter*⟩`}` prints a plus sign "+" if ⟨*counter*⟩ is negative and otherwise prints a minus sign "-". Note that the latter case includes the value zero.

```
\newcommand{\xnegsignof}[1]{\ifctrneg{#1}{+}{-}}
```

\xsigned

`\xsigned{`⟨*counter*⟩`}` prints the absolute value of ⟨*counter*⟩, preceded by a plus sign "+" or a minus sign "-" as defined by `\xsignof`.

```
\newcommand{\xsigned}[1]{\xsignof{#1} \xabsof{#1}}
```

\xsignednz

`\xsignednz{`⟨*counter*⟩`}` is like `\xsigned` but suppresses the number zero.

```
\newcommand{\xsignednz}[1]{\ifctrzero{#1}{}{\xsigned{#1}}}
```

\xarabicnz

`\xarabicnz{`⟨*counter*⟩`}` is like `\xarabic` but suppresses the number zero.

```
\newcommand{\xarabicnz}[1]{\ifctrzero{#1}{}{\xarabic{#1}}}
```

\xnegsigned

`\xnegsigned{`⟨*counter*⟩`}` prints the absolute value of ⟨*counter*⟩, preceded by a plus sign "+" or a minus sign "-" as defined by `\xnegsignof`.

```
\newcommand{\xnegsigned}[1]{\xnegsignof{#1} \xabsof{#1}}
```

`\xnegsignednz` `\xnegsignednz{`⟨*counter*⟩`}` is like `\xnegsigned` but suppresses the number zero.

```
\newcommand{\xnegsignednz}[1]{\ifctrzero{#1}{}{\xnegsigned{#1}}}
```

`\xnegof` `\xnegof{`⟨*counter*⟩`}` prints the negative of the value of ⟨*counter*⟩.

```
\newcommand{\xnegof}[1]{\ifctrpos{#1}{-}{}\xabsof{#1}}
```

`\xnegofnz` `\xnegofnz{`⟨*counter*⟩`}` is like `\xnegof` but suppresses the number zero.

```
\newcommand{\xnegofnz}[1]{\ifctrzero{#1}{}{\xnegof{#1}}}
```

`\xcoef` `\xcoef{`⟨*counter*⟩`}` prints the value of ⟨*counter*⟩ except that it suppresses the values of 1, 0, and -1, printing a "-" in the latter case.

```
\newcommand{\xcoef}[1]{%
   \ifboolexpr{test {\ifnumless{\xvalue{#1}}{-1}}%
             or test {\ifnumgreater{\xvalue{#1}}{1}}}{%
      \xarabic{#1}%
   }{%
   }%
   \ifnumequal{\xvalue{#1}}{-1}{-}{}%
}
```

`\xnegcoef` `\xnegcoef{`⟨*counter*⟩`}` prints the value of ⟨*counter*⟩ except that it suppresses the values of 1, 0, and -1, printing a "-" in the former case.

```
\newcommand{\xnegcoef}[1]{%
   \ifboolexpr{test {\ifnumless{\xvalue{#1}}{-1}}%
             or test {\ifnumgreater{\xvalue{#1}}{1}}}{%
      \xnegof{#1}%
   }{%
   }%
   \ifnumequal{\xvalue{#1}}{1}{-}{}%
}
```

`\xabsofcoef` `\xabsofcoef{`⟨*counter*⟩`}` prints the absolute value of ⟨*counter*⟩ except that it suppresses the values of 1 and 0.

```
\newcommand{\xabsofcoef}[1]{%
   \ifboolexpr{test {\ifnumless{\xvalue{#1}}{-1}}%
             or test {\ifnumgreater{\xvalue{#1}}{1}}}{%
      \xabsof{#1}%
   }{%
   }
}
```

`\xsignofcoef` `\xsignofcoef{`⟨*counter*⟩`}` prints the sign of ⟨*counter*⟩ if ⟨*counter*⟩ is nonzero.

```
\newcommand{\xsignofcoef}[1]{\ifctrzero{#1}{}{\xsignof{#1}}}
```

`\xnegsignofcoef` `\xnegsignofcoef{`⟨*counter*⟩`}` prints the opposite sign of ⟨*counter*⟩ if ⟨*counter*⟩ is nonzero.

```
\newcommand{\xnegsignofcoef}[1]{\ifctrzero{#1}{}{\xnegsignof{#1}}}
```

\xsignedcoef `\xsignedcoef{`⟨*counter*⟩`}` is like `\xcoef` except that positive values are preceded by a plus sign "+".

```
\newcommand{\xsignedcoef}[1]{\xsignofcoef{#1} \xabsofcoef{#1}}
```

\xnegsignedcoef `\xnegsignedcoef{`⟨*counter*⟩`}` is like `\xsignedcoef` except using the opposite sign.

```
\newcommand{\xnegsignedcoef}[1]{\xnegsignofcoef{#1} \xabsofcoef{#1}}
```

3.5 Random Counters

The commands `\xrandsetcounter` and `\xrandaddtocounter` use the pdfTEX primitive `\pdfuniformdeviate` to provide random versions of `\xsetcounter` and `\xaddtocounter`.

\xrandsetcounter `\xrandsetcounter{`⟨*counter*⟩`}{`⟨*min*⟩`}{`⟨*max*⟩`}` assigns to (the prefixed) ⟨*counter*⟩ a random integer value between ⟨*min*⟩ and ⟨*max*⟩.

```
\newcommand{\xrandsetcounter}[3]{%
   \xsetcounter{#1}{%
      \the \numexpr #2+\pdfuniformdeviate \numexpr #3-#2+1 \relax
   }
}
```

\xrandaddtocounter `\xrandaddtocounter{`⟨*counter*⟩`}{`⟨*min*⟩`}{`⟨*max*⟩`}` adds to (the prefixed) ⟨*counter*⟩ a random integer value between ⟨*min*⟩ and ⟨*max*⟩.

```
\newcommand{\xrandaddtocounter}[3]{%
   \xaddtocounter{#1}{%
      \the \numexpr #2+\pdfuniformdeviate \numexpr #3-#2+1 \relax
   }
}
```

The following commands are designed to provide a means by which authors can generate random values for counters but also preserve those values for future typesettings. This is accomplished by storing counters and their values in an external file and then inputting the file before a subsequent typesetting.

\opencountersfile The command `\opencountersfile` creates and opens the write stream to the file ⟨*jobname*⟩.counters.tex, referenced by the macro `\countersfile`.

```
\newcommand{\opencountersfile}{%
   \newwrite\countersfile
   \immediate\openout\countersfile=\jobname.counters.tex
}
```

\@counterz@openbrace
\@counterz@closebrace The commands `\@counterz@openbrace` and `\@counterz@closebrace` facilitate the writing of the brace delimiters to `\countersfile`.

```
\begingroup
   \catcode`<=1 \catcode`>=2
   \catcode`{=12 \catcode`}=12
   \gdef\@counterz@openbrace<{>
   \gdef\@counterz@closebrace<}>
\endgroup
```

`\xsavecounter` `\xsavecounter{`⟨*counter*⟩`}` writes `\providecounter` and `\setcounter` commands to the file ⟨*jobname*⟩.counters.tex so that they may be inputted as part of a future typesetting.

```
\newcommand{\xsavecounter}[1]{%
   \immediate\write\countersfile{%
      \unexpanded{\providecounter}\@counterz@openbrace%
         \@counterz@counterprefix #1\@counterz@closebrace%
         \unexpanded{ \setcounter}\@counterz@openbrace%
         \@counterz@counterprefix #1\@counterz@closebrace%
         \@counterz@openbrace%
         \arabic{\@counterz@counterprefix #1}\@counterz@closebrace%
   }%
}%
```

`\randomizectr` `\norandomizectr` In order to assign a random value to a counter during one typesetting and avoid overwriting this value with a random assignment during another typesetting, the boolean *@counterz@random* is used to distinguish between the two typesettings. The value of *@counterz@random* may be changed by the commands `\randomizectr` and `\norandomizectr`.

```
\newbool{@counterz@random}
\newcommand{\randomizectr}{\booltrue{@counterz@random}}
\newcommand{\norandomizectr}{\boolfalse{@counterz@random}}
```

`\ifrandomizectr` `\ifrandomizectr{`⟨*foo*⟩`}{`⟨*bar*⟩`}` executes ⟨*foo*⟩ if the boolean *@counterz@random* is true and otherwise executes ⟨*bar*⟩.

```
\newcommand{\ifrandomizectr}[2]{%
   \ifbool{@counterz@random}{#1}{#2}
}%
```

`\xrandprovidecounter` `\xrandprovidecounter{`⟨*counter*⟩`}{`⟨*min*⟩`}{`⟨*max*⟩`}` creates ⟨*counter*⟩ if it does not already exist, and if the boolean *@counterz@random* is true then ⟨*counter*⟩ is assigned a random integer value between ⟨*min*⟩ and ⟨*max*⟩ and then saved.

```
\newcommand{\xrandprovidecounter}[3]{%
   \xprovidecounter{#1}
   \ifrandomizectr{%
      \xrandsetcounter{#1}{#2}{#3}
      \xsavecounter{#1}
   }{%
   }
}
```

`\xrandprovidecounternz` `\xrandprovidecounternz{`⟨*counter*⟩`}{`⟨*min*⟩`}{`⟨*max*⟩`}` does the same job as the command `\xrandprovidecounter` except that the value of ⟨*counter*⟩ is randomized until it is nonzero.

```
\newcommand{\xrandprovidecounternz}[3]{%
   \xprovidecounter{#1}
   \ifrandomizectr{%
```

```
114       \xsetcounter{#1}{0}
115       \whileboolexpr{ test {\ifnumequal{\xvalue{#1}}{0}}}{%
116          \xrandsetcounter{#1}{#2}{#3}
117       }
118       \xsavecounter{#1}
119    }{%
120    }
121 }
```

122 ⟨/package⟩

4 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.