# The **l3ctr2e** package*†

Simon Sigurdhsson [sigurdhsson@gmail.com]

Version 0.1d

Abstract A reimplementation of some of the LaTeX $2_\varepsilon$ counter functionality in LaTeX3 code. It should be backwards-compatible with LaTeX $2_\varepsilon$ counters.

## Contents

## 1 Documentation

This package is intended to complement the LaTeX $2_\varepsilon$ counter macros with LaTeX3 counterparts. The new macros use the same internal counters and macros as the LaTeX $2_\varepsilon$ ones, and as such they should be fully backwards-compatible. The package is not terribly complicated but could be useful in adapting (or rewriting) LaTeX $2_\varepsilon$ packages or document styles to LaTeX3 standards.

**Warning:** The package is not backwards-compatible as it is now: the resetting of child counters is not implemented in a backwards-compatible way.

### 1.1 Options

`replace` `true,false` (`false`)

*Available on `http://www.ctan.org/pkg/l3ctr2e`.
†Development version available on `https://github.com/urdh/l3ctr2e`.

**Table 1:** *LaTeX3 counterparts of the LaTeX 2ε counter macros*

| LaTeX 2ε | LaTeX3 |
|---|---|
| \newcounter | \ctr_new:n |
| \stepcounter | \ctr_gincr:n |
| \refstepcounter | \ctr_ref_gincr:n |
| \addtocounter | \ctr_gadd:nn |
| \setcounter | \ctr_gset:nn |
| \value | \ctr_use:n |
| \arabic | \ctr_use_arabic:n |
| \alph | \ctr_use_alph:n |
| \Alph | \ctr_use_Alph:n |
| \roman | \ctr_use_roman:n |
| \Roman | \ctr_use_Roman:n |
| \fnsymbol | \ctr_use_fn:n |

There is only one option, `replace`, which will make the package replace the original LaTeX 2ε public API macros with new, LaTeX3-based ones (with the same API). Its default value is `false`.

## 1.2 Public API

The public LaTeX3 API has macros corresponding to the macros of the LaTeX 2ε API according to **??**. In addition to the direct replacements there are also a number of new macros which are missing in the original LaTeX 2ε API.

### 1.2.1 Creating counters

`\ctr_new:n` {⟨*counter name*⟩}
`\ctr_new:nn` {⟨*counter name*⟩}{⟨*parent counter*⟩}

This macro creates a new counter, and is the direct counterpart of the `\newcounter` LaTeX 2ε macro. It has two variants, one of which also sets

a parent counter (*i.e.* a counter which will reset the new counter when incremented), corresponding to using `\newcounter` with an optional argument.

### 1.2.2 Setting counters

`\ctr_gzero:n` {⟨*counter name*⟩}

This macro has no LaTeX $2_\varepsilon$ counterpart, but could be useful (and is used internally by other l3ctr2e macros). It sets the counter value to zero.

`\ctr_gincr:n` {⟨*counter name*⟩}

This macro increments the counter by one, and is the LaTeX3 equivalent of `\stepcounter`.

`\ctr_ref_gincr:n` {⟨*counter name*⟩}

In addition to incementing the counter by one, this macro sets the internal LaTeX $2_\varepsilon$ macro `\@currentlabel`, so that references and labels work properly. It is the LaTeX3 variant of `\refstepcounter`.

`\ctr_gadd:nn` {⟨*counter name*⟩}{⟨*value*⟩}

The LaTeX3 counterpart of `\addtocounter`. Simply adds ⟨*value*⟩ to the counter.

`\ctr_gset:nn` {⟨*counter name*⟩}{⟨*new value*⟩}

Corresponds to `\setcounter`. Sets the counter to ⟨*new value*⟩.

### 1.2.3 Displaying counters

LaTeX $2_\varepsilon$ provides a couple of different ways to express counters in typeset text. These macros correspond to (and extend) the repertoire of counter value macros provided by LaTeX $2_\varepsilon$ using functionality from the l3int package.

`\ctr_use:n` {⟨*counter name*⟩}

This is simply `\value`. It will simply use the value of the counter (using `\int_use:c`) and as such could be used in a l3int context whenever an integer value of the counter is needed.

`\ctr_use_arabic:n` {⟨*counter name*⟩}

This macro typesets the counter value as arabic numbers, and corresponds to the `\arabic` macro.

`\ctr_use_alph:n` {⟨*counter name*⟩}
`\ctr_use_Alph:n` {⟨*counter name*⟩}

Typesets the counter value as lower-case or upper-case alphabetic characters — the corresponding LaTeX $2_\varepsilon$ macros are `\alph` and `\Alph`.

`\ctr_use_roman:n` {⟨*counter name*⟩}
`\ctr_use_Roman:n` {⟨*counter name*⟩}

Typesets the counter value as lower-case or upper-case roman numerals. Corresponds to `\roman` and `\Roman`.

`\ctr_use_fn:n` {⟨*counter name*⟩}

Typesets the counter value as footnote symbols, and is intended for use in footnote marks only. Corresponds to `\fnsymbol`.

`\ctr_use_binary:n` {⟨*counter name*⟩}
`\ctr_use_octal:n` {⟨*counter name*⟩}
`\ctr_use_hex:n` {⟨*counter name*⟩}

Typesets the counter value as binary, octal or hexadecimal numbers. These macros have no LaTeX $2_\varepsilon$ counterparts.

`\ctr_use_custom:nn` {⟨*counter name*⟩}{⟨*item*⟩,⟨*item*⟩,...}

Typesets the counter value according to the comma-separated list provided (containing a list of items). Values larger than the number of items in the list will cause an error.

## 2 Known issues

A list of current issues is available in the Github repository of this package[1], but as of the release of v0.1d, there are no known issues.

If you discover any bugs in this package, please report them to the issue tracker in the l3ctr2e Github repository.

## 3 Implementation

Start by including expl3 and well as declaring the class.

```
⟨package⟩ 1 \RequirePackage{expl3,l3keys2e,xparse}
          2 \ProvidesExplClass{l3ctr2e}%
          3     {2013/05/25}{0.1d}{LaTeX2e counters in LaTeX3 ways}
```

### 3.1 Options

```
⟨package⟩ 4 \keys_define:nn{l3ctr2e}{
```

`replace` (option)

```
⟨package⟩ 5     replace .bool_set:N = \g__ctr_replace_latexe_bool,
          6     replace .default:n = true,
          7     replace .initial:n = false

⟨package⟩ 8 }
          9 \ProcessKeysPackageOptions{l3ctr2e}
```

### 3.2 Messages

One message is declared for future use.

```
⟨package⟩ 10 \msg_new:nnn{l3ctr2e}{no-counter}{No~counter~`#1'~defined!}
```

[1] https://github.com/urdh/l3ctr2e/issues

## 3.3 Private macros

`\__ctr_exists:nT` `#1`: Counter name

⟨package⟩
```
\prg_new_conditional:Npnn\__ctr_exists:n#1{T}{
    \int_if_exist:cTF{c@#1}{
        \prg_return_true:
    }{
        \msg_error:nnn{l3ctr2e}{no-counter}{#1}
        \prg_return_false:
    }
}
```

⟨package⟩
```
\tl_new:N\g__ctr_children__ctr_global_tl
```

`\__ctr_define:n` `#1`: Counter name

This does not yet fully correspond to `\@definecounter`.

⟨package⟩
```
\cs_new:Npn\__ctr_define:n#1{
    \int_new:c{c@#1} % actually \newcount, incorrect! FIX!
    \ctr_gzero:n{#1}
    \tl_new:c{g__ctr_children_#1_tl}
    \cs_gset:cpn{cl@#1}{}
    \cs_gset:cpn{p@#1}{}
    \cs_gset:cpn{the#1}{\ctr_use_arabic:n{#1}}
}
```

`\__ctr_add_child:nn` `#1`: Parent counter name

`#2`: Counter name

This probably fully corresponds to `\@addtoreset`.

⟨package⟩
```
\cs_new:Npn\__ctr_add_child:nn#1#2{
    \tl_gput_right:cn{g__ctr_children_#1_tl}{{#2}}
}
```

## 3.4 Public API

### 3.4.1 Creating counters

`\ctr_new:n` #1: Counter name

```
⟨package⟩ 31 \cs_new:Npn\ctr_new:n#1{
          32     \__ctr_define:n{#1}
          33     \__ctr_add_child:nn{_ctr_global}{#1}
          34 }
```

`\ctr_new:nn` #1: Counter name
#2: Parent counter

```
⟨package⟩ 35 \cs_new:Npn\ctr_new:nn#1#2{
          36     \__ctr_define:n{#1}
          37     \__ctr_add_child:nn{#2}{#1}
          38 }
```

### 3.4.2 Setting counters

`\ctr_gzero:n` #1: Counter name

```
⟨package⟩ 39 \cs_new:Npn\ctr_gzero:n#1{
          40     \__ctr_exists:nT{#1}{\int_gzero:c{c@#1}}
          41 }
```

`\ctr_gset:nn` #1: Counter name
#2: New value

```
⟨package⟩ 42 \cs_new:Npn\ctr_gset:nn#1#2{
          43     \__ctr_exists:nT{#1}{\int_gset:cn{c@#1}{#2}}
          44 }
```

`\ctr_gadd:nn` #1: Counter name
#2: Value to add

```
⟨package⟩ 45 \cs_new:Npn\ctr_gadd:nn#1#2{
          46     \__ctr_exists:nT{#1}{\int_gadd:cn{c@#1}{#2}}
```

```
}
```

`\ctr_gincr:n` #1: Counter name

```
\cs_new:Npn\ctr_gincr:n#1{
    \ctr_gadd:nn{#1}{\c_one}
    \tl_if_exist:cT{g__ctr_children_#1_tl}{
        \tl_map_inline:cn{g__ctr_children_#1_tl}{
            \ctr_gincr:n{##1} % To reset children of the child
            \ctr_gzero:n{##1}
        }
    }
}
```

`\ctr_ref_gincr:n` #1: Counter name

```
\cs_new:Npn\ctr_ref_gincr:n#1{
    \ctr_gincr:n{#1}
    \cs_set:Npx\@currentlabel{
        \use:c{p@#1}
        \use:c{the#1}
    }
}
```

### 3.4.3 Displaying counters

`\ctr_value:n` #1: Counter name

```
\cs_new:Npn\ctr_value:n#1{
    \use:c{c@#1}
}
```

`\ctr_use:n` #1: Counter name

```
\cs_new:Npn\ctr_use:n#1{
    \int_use:c{c@#1}
}
```

`\ctr_use_arabic:n`#1: Counter name

⟨package⟩
```
\cs_new:Npn\ctr_use_arabic:n#1{
    \int_to_arabic:n{\ctr_use:n{#1}}
}
```

`\ctr_use_alph:n`#1: Counter name

⟨package⟩
```
\cs_new:Npn\ctr_use_alph:n#1{
    \int_to_alph:n{\ctr_use:n{#1}}
}
```

`\ctr_use_Alph:n`#1: Counter name

⟨package⟩
```
\cs_new:Npn\ctr_use_Alph:n#1{
    \int_to_Alph:n{\ctr_use:n{#1}}
}
```

`\ctr_use_binary:n`#1: Counter name

⟨package⟩
```
\cs_new:Npn\ctr_use_binary:n#1{
    \int_to_binary:n{\ctr_use:n{#1}}
}
```

`\ctr_use_hex:n`#1: Counter name

⟨package⟩
```
\cs_new:Npn\ctr_use_hex:n#1{
    \int_to_hexadecimal:n{\ctr_use:n{#1}}
}
```

`\ctr_use_octal:n`#1: Counter name

⟨package⟩
```
\cs_new:Npn\ctr_use_octal:n#1{
    \int_to_octal:n{\ctr_use:n{#1}}
}
```

`\ctr_use_roman:n`#1: Counter name

⟨package⟩
```
\cs_new:Npn\ctr_use_roman:n#1{
```

```
    \int_to_roman:n{\ctr_use:n{#1}}
}
```

`\ctr_use_Roman:n`#1: Counter name

⟨package⟩
```
\cs_new:Npn\ctr_use_Roman:n#1{
    \int_to_Roman:n{\ctr_use:n{#1}}
}
```

`\ctr_use_fn:n`#1: Counter name

⟨package⟩
```
\cs_new:Npn\ctr_use_fn:n#1{
    \ensuremath{\ctr_use_custom:nn{#1}{
        *,          \dagger,       \ddagger,
        \mathsection,\mathparagraph,\|,
        **,         \dagger\dagger,\ddagger\ddagger
    }}
}
```

`\ctr_use_custom:nn`#1: Counter name
#2: Comma-separated list of items

⟨package⟩
```
\cs_new:Npn\ctr_use_custom:nn#1#2{
    \clist_set:Nn\l_tmpa_clist{#2}
    \tl_clear:N\l_tmpa_tl
    \int_zero:N\l_tmpa_int
    \clist_map_inline:Nn\l_tmpa_clist{
            \int_incr:N\l_tmpa_int
            \tl_put_right:Nx\l_tmpa_tl{{\int_use:N\l_tmpa_int}{##1}}
    }
    \exp_args:Nnnx\int_to_symbols:nnn{\ctr_use:n{#1}}
        {\clist_count:N\l_tmpa_clist}
        {\tl_use:N\l_tmpa_tl}
}
```

## 3.5 Replacing the LaTeX 2ε API

```
⟨package⟩ 113 \bool_if:NT\g__ctr_replace_latexe_bool{
```

`\newcounter` #1: Counter name
#2: Parent counter

```
⟨package⟩ 114     \RenewDocumentCommand\newcounter{mo}{
          115         \IfNoValueTF{#2}{\ctr_new:n{#1}}
          116                         {\ctr_new:nn{#1}{#2}}
          117     }
```

`\stepcounter` #1: Counter name

```
⟨package⟩ 118     \RenewDocumentCommand\stepcounter{m}{\ctr_gincr:n{#1}}
```

`\refstepcounter` #1: Counter name

```
⟨package⟩ 119     \RenewDocumentCommand\refstepcounter{m}{\ctr_ref_gincr:n{#1}}
```

`\addtocounter` #1: Counter name
#2: Value to add

```
⟨package⟩ 120     \RenewDocumentCommand\addtocounter{mm}{\ctr_gadd:nn{#1}{#2}}
```

`\setcounter` #1: Counter name
#2: New value

```
⟨package⟩ 121     \RenewDocumentCommand\setcounter{mm}{\ctr_gset:nn{#1}{#2}}
```

`\value` #1: Counter name

```
⟨package⟩ 122     \RenewDocumentCommand\value{m}{\ctr_value:n{#1}}
```

`\arabic` #1: Counter name

```
⟨package⟩ 123     \RenewDocumentCommand\arabic{m}{\ctr_use_arabic:n{#1}}
```

`\alph` #1: Counter name

```
⟨package⟩ 124    \RenewDocumentCommand\alph{m}{\ctr_use_alph:n{#1}}
```

`\Alph` #1: Counter name

```
⟨package⟩ 125    \RenewDocumentCommand\Alph{m}{\ctr_use_Alph:n{#1}}
```

`\roman` #1: Counter name

```
⟨package⟩ 126    \RenewDocumentCommand\roman{m}{\ctr_use_roman:n{#1}}
```

`\Roman` #1: Counter name

```
⟨package⟩ 127    \RenewDocumentCommand\Roman{m}{\ctr_use_Roman:n{#1}}
```

`\fnsymbol` #1: Counter name

```
⟨package⟩ 128    \RenewDocumentCommand\fnsymbol{m}{\ctr_use_fn:n{#1}}
⟨package⟩ 129 }
⟨package⟩ 130 \endinput
```

# 4 Installation

The easiest way to install this package is using the package manager provided by your LaTeX installation if such a program is available. Failing that, provided you have obtained the package source (`l3ctr2e.tex` and `Makefile`) from either CTAN or Github, running `make install` inside the source directory works well. This will extract the documentation and code from `l3ctr2e.tex`, install all files into the TDS tree at `TEXMFHOME` and run `mktexlsr`.

If you want to extract code and documentation without installing the package, run `make all` instead. If you insist on not using `make`, remember

that packages distributed using skdoc must be extracted using `pdflatex`, *not* `tex` or `latex`.