ACRO

v2.7c 2017/08/28

Typeset Acronyms and other Abbreviations

Clemens NIEDERBERGER

https://bitbucket.org/cgnieder/acro/

contact@mychemistry.eu

ACRO allows you to define and use abbreviations in a simple way. Abbreviations can be divided into different classes of abbreviations. Lists of abbreviations can be created (also of separate classes of abbreviations) and printed wherever you want the lists to appear.

ACRO provides an option `single` which ignores abbreviations that are used only once in the whole document.

As an experimental feature ACRO also offers the option `sort` which automatically sorts the list created by `\printacronyms`.

ACRO also has the feature of creating *local* lists

Table of Contents

1. Licence and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3 or later (`http://www.latex-project.org/lppl.txt`). The software has the status "maintained."

ACRO loads and needs the following packages: expl3,[1] xparse, xtemplate, l3keys2e,[2] zref-abspage[3] and translations[4] [Nie15].

2. Basics

2.1. Creating New Acronyms

Acronyms are created with the command `\DeclareAcronym`.

`\DeclareAcronym{`⟨*id*⟩`}{`⟨*list of keys*⟩`}`
The basic command for declaring an acronym.

This command understands a number of keys which are listed below. Some of them are not described immediately but at appropriate places in the documentation.

1. on CTAN as `l3kernel`: `http://mirrors.ctan.org/macros/latex/contrib/l3kernel/`
2. on CTAN as `l3packages`: `http://mirrors.ctan.org/macros/latex/contrib/l3packages/`
3. on CTAN as `oberdiek`: `http://mirrors.ctan.org/macros/latex/contrib/oberdiek/`
4. on CTAN as `translations`: `http://mirrors.ctan.org/macros/latex/contrib/translations/`

`short` = {⟨*text*⟩} (required)
The short form of the acronym. This option is required: an acronym must have a short form. If this is set it *must* be set as first option! If another option is set first and notices the `short` option missing it assumes that the ID should be used as short version and sets it accordingly. A warning will be written to the log then.

`long` = {⟨*text*⟩} (required)
The long form of the acronym. This option is required: an acronym must have a description.

`short-plural` = {⟨*text*⟩} Default: `s`
The plural ending appended to the short form.

`short-plural-form` = {⟨*text*⟩}
Introduced in version 2.0
The plural short form of the acronym; replaces the short form when used instead of appending the plural ending.

`long-plural` = {⟨*text*⟩} Default: `s`
The plural ending appended to the long form.

`long-plural-form` = {⟨*text*⟩}
Plural long form of the acronym; replaces the long form when used instead of appending the plural ending.

`alt-plural` = {⟨*text*⟩} Default: `s`
Introduced in version 2.0
The plural ending appended to the alternative form.

`alt-plural-form` = {⟨*text*⟩}
Introduced in version 2.0
The plural alternative form of the acronym; replaces the alternative form when used instead of appending the plural ending.

`list` = {⟨*text*⟩}
If specified this will be written in the list as description instead of the long form.

`short-indefinite` = {⟨*text*⟩} Default: `a`
Indefinite article for the short form.

`long-indefinite` = {⟨*text*⟩} Default: `a`
Indefinite article for the long form.

`long-pre` = {⟨*text*⟩}
⟨*text*⟩ is prepended to the long form in the text but not in the list of acronyms.

`long-post` = {⟨*text*⟩}
⟨*text*⟩ is appended to the long form in the text but not in the list of acronyms.

`alt` = {⟨*text*⟩}
Alternative short form.

`alt-indefinite` = {⟨*text*⟩} Default: `a`
Indefinite article for the alternative form.

`extra` = {⟨*text*⟩}
Extra information to be added in the list of acronyms.

`foreign` = {⟨*original long form*⟩}
Can be useful when dealing with acronyms in foreign languages, see section 2.7 on page 11 for details.

`foreign-lang` = {⟨*language*⟩}
Introduced in version 2.3
The babel [Bra16] or polyglossia [Cha15] language of the foreign form. This language is used to wrap the entry with `\foreignlanguage`{⟨*language*⟩} if either babel or polyglossia is loaded. You'll need to take care that the corresponding language is loaded by babel or polyglossia.

`single` = {⟨*text*⟩}
Introduced in version 2.3
If provided ⟨*text*⟩ will be used instead of the long form if the acronym is only used a single time *and* the option `single` = `{true}` is active.

`sort` = {⟨*text*⟩}
If used the acronym will be sorted according to this key instead of its ID.

`class` = {⟨*csv list*⟩}
Changed in version 2.4
The class(es) the acronym belongs to.

`cite` = {[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*citation keys*⟩}}
A citation that is printed to the acronym according to an option explained later.

`short-format` = {⟨*TEX code*⟩}
The format used for the short form of the acronym.

`long-format` = {⟨*TEX code*⟩}
The format used for the long form of the acronym.

`first-long-format` = {⟨*TEX code*⟩}
The format used for the first long form of the acronym as set with `\ac`, `\acf` or `\acflike` and their uppercase, plural and indefinite forms.

`single-format` = {⟨*TEX code*⟩}
Introduced in version 2.3
The format used for the acronym if the acronym is only used a single time.

`first-style` = `default` | `empty` | `square` | `short` | `long` | `reversed` | `footnote` | `sidenote` | `footnote-reversed` | `sidenote-reversed`
Introduced in version 2.3
The style of the first appearance of the acronym, see also section 5.2.

`pdfstring` = {⟨*text*⟩/⟨*plural ending*⟩}
Changed in version 2.4b
Used as PDF string replacement in bookmarks when used together with the hyperref package. The appended plural ending is optional. If you leave it (*and* the `/`) the default ending is used. ⟨*text*⟩ is expanded before it is saved.

`accsupp` = {⟨*text*⟩}
Sets the `ActualText` key as presented by the accsupp package for the acronym.

`tooltip` = {⟨*text*⟩}
Introduced in version 2.1
Sets the tooltip description for an acronym. For actually getting tooltips you also need an appropriate setting of the `tooltip-cmd` option or to set the package option `tooltip`.

`index-sort` = {⟨*text*⟩}
If you use the package option `index` every occurrence of an acronym is recorded to the index and sorted by its ID or (if set) by the value of the `sort` key. This key allows to set an individual sorting option for the index. See section 3.8 on page 17 for details.

`index` = {⟨*text*⟩}
This key allows to overwrite the automatic index entry with an arbitrary one. See section 3.8 on page 17 for details.

`index-cmd` = {⟨*control sequence*⟩}
This key let's you set an individual index creating command for this acronym. It should be a command that takes one mandatory argument. See section 3.8 on page 17 for details.

In its simplest form an acronym needs a short and a long form. Please note that both keys *must* be set and that the `short` key *must* always be the *first* key that is set.

```
% preamble:
\DeclareAcronym{test}{
  short = ST ,
  long  = Some Test
}
```

This creates the acronym "ST" with the ID "test" and the long form "Some Test."
The `format` key allows you to choose a specific format for the short form of an acronym:

```
% preamble:
\DeclareAcronym{ot}{
  short        = ot ,
  long         = Other Test ,
  short-format = \scshape
}
```

The short form now looks like this: OT.

The `cite` key needs a bit explaining. It expects arguments like the standard `\cite` command, *i.e.*, two optional arguments setting the ⟨*prenote*⟩ and ⟨*postnote*⟩ and one mandatory argument setting the citation key.

```
% preamble:
\DeclareAcronym{ny}{
  short        = NY ,
  short-plural = ,
  long         = New York ,
  long-plural  = ,
  cite         = {NewYork}
}
```

```
% bib file for use with biber/biblatex:
@online{NewYork,
  author  = {Wikipedia},
  title   = {New York City},
  urldate = {2012-09-27},
  url     = {http://en.wikipedia.org/wiki/New_York_City},
  year    = {2012}
}
```

The first appearance now looks as follows[5]: New York (NY) [Wik12].

2.2. Logging of Acronyms

Introduced in version 2.5

When you activate ACRO's option `log` ACRO' writes information about the acronyms it defines to the log file.

`log` = `true`|`false`|`silent`|`verbose` Default: `false`

When set to `true`/`silent` ACRO writes the main properties of an acronym to the log file. When set to `verbose` ACRO' writes *all* properties of an acronym to the log file.

This is an example of the logging info with `log` = `{true}` or `log` = `{silent}`.

```
================================================
| acro info -- defining new acronym:
```

5. The appearance of the citation of course depends on the citation style you're using.

```
|   ID = {jpg}
|   short = {JPEG}
|   long = {Joint Photographic Experts Group}
|   alt = {JPG}
|   sort = {jpeg}
|   class = {}
|   list = {}
|   extra = {}
|   foreign = {}
|   pdfstring = {}
|   cite = {}
==============================================
```

2.3. Using the Acronyms – the Commands

Acronyms are used with one of the following commands:

`\ac*{`⟨*id*⟩`}`
basic command; the first output is different from subsequent ones.

`\Ac*{`⟨*id*⟩`}`
same as `\ac` but capitalizes the first letter of the long form.

`\acs*{`⟨*id*⟩`}`
short form; the actual acronym.

`\acl*{`⟨*id*⟩`}`
long form; the meaning of the acronym.

`\Acl*{`⟨*id*⟩`}`
same as `\acl` but capitalizes first letter.

`\aca*{`⟨*id*⟩`}`
alternative short form as specified in the `alt` key of `\DeclareAcronym`; if it hasn't been specified this is identical to `\acs`.

`\acf*{`⟨*id*⟩`}`
first form; output like the first time `\ac` is output.

`\Acf*{`⟨*id*⟩`}`
same as `\acf` but capitalizes first letter of the long form.

`\acp*{`⟨*id*⟩`}`
plural form of `\ac`;

`\Acp*{`⟨*id*⟩`}`
same as `\acp` but capitalizes first letter of the long form.

`\acsp*{`⟨*id*⟩`}`
plural form of `\acs`;

`\aclp*{`⟨*id*⟩`}`
plural form of `\acl`;

`\Aclp*{`⟨*id*⟩`}`
same as `\aclp` but capitalizes first letter.

`\acap*{`⟨*id*⟩`}`
plural form of `\aca`;

`\acfp*{`⟨*id*⟩`}`
plural form of `\acf`;

`\Acfp*{`⟨*id*⟩`}`
same as `\acfp` but capitalizes first letter of the long form.

If an acronym is used the first time with `\ac` its output is different from subsequent uses. To be clear on this: the first time! If the acronym has been used with *any* of the output commands before it is *not* the first time any more.

If you use the starred variant an acronym will not be marked as used. This proves useful if an acronym is typeset in a section title, for example, since then the appearance in the table of contents won't mark it as used.

```
% preamble:
% \DeclareAcronym{cd}{
%   short        = cd ,
%   long         = Compact Disc ,
%   short-format = \scshape
% }
first time: \ac{cd} \\
second time: \ac{cd} \\
short: \acs{cd} \\
alternative: \aca{cd} \\
first again: \acf{cd} \\
long: \acl{cd} \\
short plural: \acsp{cd} \\
long plural: \aclp{cd}
```

first time: Compact Disc (CD)
second time: CD
short: CD
alternative: CD
first again: Compact Disc (CD)
long: Compact Disc
short plural: CDs
long plural: Compact Discs

2.4. Plural Forms

If an acronym is defined in the standard way ACRO uses an 's' that's appended to both the short and the long form when one of the plural commands is used. However, that is not always the best solution. For one thing not all acronyms may have a plural form. Second, the plural form

especially of the long forms may be formed differently. And third, other languages can have other plural endings.

For these reasons `\DeclareAcronym` can get the following keys:

`short-plural` = {⟨*text*⟩} Default: `s`

The plural ending of the short form.

`long-plural` = {⟨*text*⟩} Default: `s`

The plural ending of the long form.

`long-plural-form` = {⟨*text*⟩}

An alternative plural form for the long form.

These keys are optional. If they're not used, the default setting is `s`. If you use `long-plural-form` the long form will be replaced by the specified plural form when necessary.

Suppose we define the following acronyms:

```
\DeclareAcronym{cd}{
  short        = cd ,
  long         = Compact Disc ,
  short-format = \scshape
}
\DeclareAcronym{ny}{
  short        = NY ,
  short-plural = ,
  long         = New York ,
  long-plural  =
}
\DeclareAcronym{sw}{
  short       = SW ,
  long        = Sammelwerk ,
  long-plural = e
}
\DeclareAcronym{MP}{
  short            = MP ,
  long             = Member of Parliament ,
  long-plural-form = Members of Parliament
}
```

These acronyms now have the following plural appearances:

```
\acsp{cd}, \aclp{cd} \\
\acsp{ny}, \aclp{ny} \\
\acsp{sw}, \aclp{sw} \\
\acsp{MP}, \aclp{MP}
```

CDs, Compact Discs
NY, New York
SWs, Sammelwerke
MPs, Members of Parliament

2.5. Alternative Short Forms

For some acronyms it might be useful to have alternative forms. For this `\DeclareAcronym` has another key:

`alt` = {⟨*text*⟩}
Alternative short form.

```
% preamble:
% \DeclareAcronym{jpg}{
%   short = JPEG ,
%   sort  = jpeg ,
%   alt   = JPG ,
%   long  = Joint Photographic Experts Group
% }
default: \acs{jpg} \\
alt.: \aca{jpg}
```

---

default: JPEG
alt.: JPG

The alternative form uses the same plural ending as the default short form and is formatted in the same way.

2.6. Extra Information for the List Entry

Of course you can print a list of acronyms where their meaning is explained. Sometimes it can be useful to add additional information there. This is done with another key to `\DeclareAcronym`:

`extra` = {⟨*text*⟩}
Additional information for the list of acronyms.

These information will only be displayed in the list. See section 4 on page 17 for the impact of the following example.

```
% preamble:
% \DeclareAcronym{nato}{
%   short        = nato ,
%   long         = North Atlantic Treaty Organization ,
%   extra        = \textit{deutsch}: Organisation des Nordatlantikvertrags ,
%   short-format = \scshape
% }
The \ac{nato} is an intergovernmental military alliance based on the
North Atlantic Treaty which was signed on 4~April 1949. \ac{nato}
headquarters are in Brussels, Belgium, one of the 28 member states
across North America and Europe, the newest of which, Albania and
Croatia, joined in April 2009.
```

The North Atlantic Treaty Organization (NATO) is an intergovernmental military alliance based on the North Atlantic Treaty which was signed on 4 April 1949. NATO headquarters are in Brussels, Belgium, one of the 28 member states across North America and Europe, the newest of which, Albania and Croatia, joined in April 2009.

2.7. Foreign Language Acronyms

I repeatedly read the wish for being able to add translations to acronyms when the acronyms stem from another language than the document language, *i.e.*, something like the following in a German document:

```
\ac{ecu}\\
\ac{ecu}
```

Steuergerät (Electronic Control Unit, ECU)
ECU

That's why I decided to add the following properties:

`foreign` = {⟨*original long form*⟩}
A description for an acronym originating in another language than the document language.

`foreign-lang` = {⟨*language*⟩}
Introduced in version 2.3
The babel [Bra16] or polyglossia [Cha15] language of the foreign form. This language is used to wrap the entry with `\foreignlanguage`{⟨*language*⟩}.

Here is the definition of the above mentioned ECU acronym:

```
\DeclareAcronym{ecu}{
  short   = ECU ,
```

```
  long    = Steuerger\"at ,
  foreign = Electronic Control Unit ,
  foreign-lang = english
}
```

As you have seen this adds the `foreign` entry to the first appearance of an acronym. It is also added in parentheses to the list of acronyms after the `long` entry. Actually the entry there is the argument to the following command:

`\acroenparen{`⟨*argument*⟩`}`
Places ⟨*argument*⟩ in parentheses: `\acroenparen{example}`: (example). See page 22 for a way to customize this other than redefining it.

3. Additional Commands and Possibilities

3.1. Indefinite Forms

Unlike many other languages[6] in English the indefinite article is not determined by the grammatical case, gender or number but by the pronounciation of the following word. This means that the short and the long form of an acronym can have different indefinite articles. For these cases ACRO offers the keys `short-indefinite`, `alt-indefinite` and `long-indefinite` whose default is `a`. For every lowercase singular command two alternatives exist, preceded by `i` and `I`, respectively, which output the lowercase and uppercase version of the corresponding indefinite article.

```
% preamble:
% \DeclareAcronym{ufo}{
%   short           = UFO ,
%   long            = unidentified flying object ,
%   long-indefinite = an
% }
\Iac{ufo}; \iacs{ufo}; \iacl{ufo}
```

An unidentified flying object (UFO); a UFO; an unidentified flying object

3.2. Uppercasing

`\acfirstupper{`⟨*token list*⟩`}`
This command uppercases the first token in ⟨*token list*⟩. The command is less powerful than `\makefirstuc` that is provided by the mfirstuc package [Tal15] but it is expandable. Obvious downsides are for example that it does not uppercase accented letters.

6. Let's better say: unlike the other languages where I know at least the basics.

3.3. Simulating the First Appearance

Users told me[7] that there are cases when it might be useful to have the the acronym typeset according to the `first-style` but with another text than the long form. For such cases ACRO offers the following commands.

`\acflike*{`⟨*id*⟩`}{`⟨*instead of long form*⟩`}`
Write some alternative long form for acronym with ID ⟨*id*⟩ as if it were the first time the acronym was used.

`\acfplike*{`⟨*id*⟩`}{`⟨*instead of long form*⟩`}`
Plural form of `\acflike`.

```
1 \acsetup{first-style=footnote}
2 \acflike{ny}{the big apple}
```

NY[a]

a. the big apple [Wik12]

The plural ending in `\acfplike` is only appended to the short form. It makes no sense to append it to the text that is inserted manually anyway. Note that whatever text you're inserting might be gobbled depending on the `first-style` you're using.

3.4. Fetching the Single Appearance

Introduced in version 2.3

There are macros that fetch the *single* appearance of an acronym even if it has been used more than once and the `single` option is active.

`\acsingle*{`⟨*id*⟩`}`
Write acronym as if it were used only a single time.

`\Acsingle*{`⟨*id*⟩`}`
Uppercase form of `\acsingle`.

```
1 \acsingle{ny}
```

New York [Wik12]

3.5. Using Classes

The acronyms of ACRO can be divided into different classes. This doesn't change the output but allows different acronym lists, see section 4 on page 17. For this `\DeclareAcronym` has an additional key:

7. Well – one, to be precise ;)

`class` = {⟨*csv list*⟩}

Changed in version 2.4

Associated class(es) for an acronym.

This might be useful if you can and want to divide your acronyms into different types, technical and grammatical ones, say, that shall be listed in different lists. Since every acronym can get a list of associated classes those classes can effectively be used like tags for filtering acronyms.

```
% preamble:
% \DeclareAcronym{la}{
%   short        = LA ,
%   short-plural = ,
%   long         = Los Angeles ,
%   long-plural  = ,
%   class        = city
% }
% \DeclareAcronym{ny}{
%   short        = NY ,
%   short-plural = ,
%   long         = New York ,
%   long-plural  = ,
%   class        = city ,
%   cite         = NewYork
% }
\acl{la} (\acs{la}) \\
\acl{ny} (\acs{ny})
```

Los Angeles (LA)
New York (NY)

3.6. Reset or Mark as Used, Test if Acronym Has Been Used

If you want for some reason to fool ACRO into thinking that an acronym is used for the first time you can call one of these commands:

`\acreset`{⟨*comma separated list of ids*⟩}

This will reset a used acronym such that the next use of `\ac` will again print it as if it were used the first time. This will *not* remove an acronym from being printed in the list if it actually *has* been used before.

`\acresetall`

Reset all acronyms.

`\acifused`{⟨*id*⟩}{⟨*true*⟩}{⟨*false*⟩}

This command tests if the acronym with ID ⟨*id*⟩ has already been used and either puts `true` or `false` in the input stream.

```
1 \acreset{ny}\ac{ny}
```
New York (NY) [Wik12]

Beware that both commands act *globally*! There are also commands that effectively do the opposite of `\acreset`, *i.e.*, mark acronyms as used:

`\acuse{`⟨*comma separated list of ids*⟩`}`
This has the same effect as if an acronym had been used twice, that is, further uses of `\ac` will print the short form and the acronym will in any case be printed in the list (as long as its class is not excluded).

`\acuseall`
Mark all acronyms as used.

Then there are two further commands related to using acronyms:

`\acswitchoff`
Introduced in version 2.6
This command is for patching in certain situations. For example some table environments like `tabularx` or `tabu` pass their content two or more times for determining the width of the table columns. Those can be patched to add `\acswitchoff` to their trial phase.

`\acswitchon`
Introduced in version 2.6
Effectively the opposite of `\acswitchoff` – this command should probably never be needed.

3.7. `\ac` and Friends in PDF Bookmarks, Accessibility Support, Tooltips

3.7.1. PDF Bookmarks

ACRO's commands usually are not expandable which means they'd leave unallowed tokens in PDF bookmarks. hyperref offers `\texorpdfstring` to circumvent that issue manually but that isn't really a nice solution. What's the point of having macros to get output for you if you have to specify it manually after all?

That is why ACRO offers a preliminary solution for this. In a bookmark every `\ac` like command falls back to a simple text string typesetting what `\acs` would do (or `\acsp` for plural forms). These text strings both can accessed manually and can be modified to an output reserved for PDF bookmarks.

`\acpdfstring{`⟨*id*⟩`}`
Access the text string used in PDF bookmarks.

`\acpdfstringplural{`⟨*id*⟩`}`
Access the plural form of the text string used in PDF bookmarks.

`pdfstring` = {{⟨*pdfstring*⟩/⟨*plural ending*⟩}}
Key for `\DeclareAcronym` to declare a custom text string for PDF bookmarks. The plural ending can be set optionally.

For example the PDF acronym used in the title for this section is defined as follows:

```
\DeclareAcronym{pdf}{
  short     = pdf ,
  long      = Portable Document Format ,
  format    = \scshape ,
  pdfstring = PDF ,
  accsupp   = PDF
}
```

3.7.2. Accessibility Support

The last example also demonstrates the `accsupp` key. The idea is to have something different visible in the PDF file compared to what you get when you select and copy the corresponding string. In the example visible string is a lowercase `pdf` in small caps while the string copied is an uppercase `PDF`.

For this to work you need to use the *package option* `accsupp`, too, which will load the package accsupp if it isn't loaded by the user already. Then the key `accsupp` will set the `ActualText` property of `\BeginAccSupp`. Please refer to accsupp's documentation for details. To see its effect copy PDF and paste it into a text file. You should get uppercase letters instead of lowercase ones.

`accsupp` = {⟨*text*⟩}

Key for `\DeclareAcronym` to set the `ActualText` property of `\BeginAccSupp` (see accsupp's documentation for details) to be used for an acronym. It only has an effect when the package option `accsupp` is used, too.

3.7.3. Tooltips

The idea of a tooltip is to have some text shown when you hover with the mouse over the short form of an acronym. This is only available in some PDF viewers, though. On possibility for such tooltips is loading the pdfcomment package [Kle12] and using its `\pdftooltip` macro.

`tooltip` = `true`|`false` Default: `false`

Introduced in version 2.1

This options loads the pdfcomment package and sets the command for creating tooltips to `\pdftooltip`.

`tooltip-cmd` = {⟨*control sequence*⟩} Default: `\@firstoftwo`

Introduced in version 2.1

This allows users using another macro for tooltips – maybe one provided by another package or some own macro. It needs to be a macro with two mandatory arguments, the first being the string typeset in the PDF, the second being the tooltip description text.

For using this with acronyms they have a property `tooltip` which can be used inside `\DeclareAcronym` for specifying the description text of the tooltip. If the `tooltip` package option is used but the property is not set for an acronym then the `long` property is used instead.

If an acronym is used inside of another acronym then the tooltips of the "inner" acronyms are disabled.

3.8. Adding Acronyms to the Index

ACRO has the package option `index`. If it is used an index entry will be recorded every time an *unstarred* acronym command is used. The index entry will be ⟨*id*⟩`@`⟨*short*⟩, ⟨*sort*⟩`@`⟨*short*⟩ if the `sort` key has been set, ⟨*index-sort*⟩`@`⟨*short*⟩ if the `index-sort` has been set, or ⟨*index*⟩ if the key `index` has been set for the specific acronym. The short versions appearing there are formatted according to the chosen format of the corresponding acronym, of course.

This document demonstrates the feature. You can find every acronym that has been declared in the index. In order to allow flexibility the indexing command can be chosen both globally via package option and individually for every acronym. This would allow to add acronyms to a specific index if more than one index is used, for example with help of the imakeidx package.

I'm not yet convinced this is a feature many people if anyone needs and if they do if it is flexible enough. If you have any thoughts on this I'd appreciate an email.

4. Printing the List

Printing the whole list of acronyms is easy: just place `\printacronyms` where ever you want the list to be.

`\printacronyms`[⟨*options*⟩]
Print the list of acronyms.

The commands takes a few options, namely the following ones:

`include-classes` = {⟨*list of classes*⟩}
Takes a comma-separated list of the classes of acronyms that should be in the list.

`exclude-classes` = {⟨*list of classes*⟩}
Takes a comma-separated list of the classes of acronyms that should *not* be in the list. *Note that this list overwrites any entries in `include-classes`!* If a class is both included and excluded then the corresponding acronyms will not be added to the list.

`name` = {⟨*name of the list*⟩}
sets the name for the list.

`heading` = {⟨*sectioning command without leading backslash*⟩} Default: `section*`
Sets the sectioning command for the heading of the list. A special value is `none` which suppresses the heading.

`sort` = `true`|`false` Default: `true`
Set sorting for this list only.

`local-to-barriers` = `true`|`false` Default: `false`

Introduced in version 2.4

This option can be used to create a list of only the acronyms of the current “barrier group”, see section 8.

```
\acsetup{extra-style=comma}
\printacronyms[exclude-classes=city]

\printacronyms[include-classes=city,name={City Acronyms}]
```

Acronyms

cd Compact Disc

ctan Comprehensive TEX Archive Network

ECU Steuergerät (Electronic Control Unit)

id identification string

JPEG Joint Photographic Experts Group

lppl LATEX Project Public License

MP Member of Parliament

nato North Atlantic Treaty Organization, *deutsch*: Organisation des Nordatlantikvertrags

pdf Portable Document Format

SW Sammelwerk

ST Some Test

UFO unidentified flying object

City Acronyms

LA Los Angeles

NY New York

You can see that the default layout is a `description` list with a `\section*` title. Both can be

changed, see section 5.

The command \printacronyms needs two LaTeX runs. This is a precaution to avoid error messages with a possibly empty list. But since almost all documents need at least two runs and often are compiled much more often than that, this fact shouldn't cause too much inconvenience.

5. Options and Customization

5.1. General Options

There are a few options which change the general behaviour of ACRO. Underlined values are used if no value is given.

`messages` = `silent|loud` Default: `loud`

Setting `messages` = `{silent}` will turn all of ACRO's error messages into warnings and all of ACRO's warnings into info messages. Be sure to check the log file carefully if you decide to set this option.

`single` = `true|false` Default: `false`

If set to `true` an acronym that's used only once (with \ac) in a document will only print the acronym in a specified form and will not be printed in the list.

`single-form` = `long|short|alt|first` Default: `long`

Introduced in version 2.0

Determines how a single appearance of an acronym is printed if `single` = `{true}` has been chosen.

`hyperref` = `true|false` Default: `false`

If set to `true` the short forms of the acronyms will be linked to their list entry.

`label` = `true|false` Default: `false`

If set to `true` this option will place \label{⟨*prefix*⟩⟨*id*⟩} the first time the acronym with ID ⟨*id*⟩ is used.

`label-prefix` = {⟨*text*⟩} Default: `ac:`

The prefix for the \label that is placed when option `label` = `{true}` is used.

`only-used` = `true|false` Default: `true`

This option is `true` as default. It means that only acronyms that are actually used in the document are printed in the list. If `false`, all acronyms defined with \DeclareAcronym will be written to the list.

`mark-as-used` = `first|any` Default: `any`

This option determines wether an acronym is mark as used when the *first* form is used the first time (with \ac, \acf or \acflike and their uppercase, plural and indefinite forms) or when any of the \ac-like commands is used.

`macros` = `true|false` Default: `false`

If set to `true` this option will create a macro \⟨*id*⟩ for each acronym as a shortcut for \ac{⟨*id*⟩}. Already existing macros will *not* be overwritten.

`xspace` = `true`|`false` Default: `false`
If set to `true` this option will append `\xspace` from the xspace package to the commands created with the `macros` option.

`strict` = `true`|`false` Default: `false`
If set to `true` and the option `macros` = `{true}` is in effect then already existing macros will be overwritten.

`sort` = `true`|`false` Default: `true`
If set to `true` the acronym list will be sorted automatically. The entries are sorted by their ID ignoring upper and lower case. This option needs the experimental package l3sort (from the l3experimental bundle) and can only be set in the preamble.

`cite` = `all`|`first`|`none` Default: `first`
Changed in version 2.4b
This option decides whether citations that are added via `cite` are added to each first, every or no appearance of an acronym. If `first` is chosen, the option `single` = `{true}` is active and an acronym appears only once it still will get the citation.

`cite-cmd` = `{`⟨*control sequence*⟩`}` Default: `\cite`
This option determines which command is used for the citation. Each citation command that takes the cite key as argument is valid, for example biblatex's `\footcite`.

`cite-connect` = `{`⟨*code*⟩`}` Default: `\nobreakspace`
Depending on the citation command in use a space should be inserted before the citation or maybe not (e.g. `\footcite`…). This option allows you to set this. Actually it can be used to place arbitrary code right before the citation.

`group-citation` = `true`|`false` Default: `false`
Introduced in version 2.0
If set to true the short form (or the long form) and the citation of an acronym will be printed together in parentheses when an acronym is used the first time.

`group-cite-cmd` = `{`⟨*control sequence*⟩`}` Default: `\cite`
Introduced in version 2.0
This option determines which command is used for the citation when an acronym is used the first time *and* `group-citation` = `{true}`. Each citation command that takes the cite key as argument is valid, for example biblatex's `\footcite`.

`index` = `true`|`false` Default: `false`
If set to `true` an index entry will be recorded every time an *unstarred* acronym command is used for the corresponding acronym.

`index-cmd` = `{`⟨*control sequence*⟩`}` Default: `\index`
Chooses the index command that is used when option `index` has been set to `true`.

`accsupp` = `true`|`false` Default: `false`
Activates the access support as provided by the accsupp package.

`tooltip` = `true`|`false` Default: `false`
Introduced in version 2.1
Activates tooltip support for ACRO using the pdfcomment package.

`tooltip-cmd` = {⟨*control sequence*⟩} Default: `\@firstoftwo`

Introduced in version 2.1

A macro taking two mandatory arguments, the first being the short form of the acronym and the second being some tooltip description.

`uc-cmd` = {⟨*control sequence*⟩} Default: `\acfirstupper`

The command that is used to capitalize the first word in the `\Ac` and the like commands. You can change it to another one like for example `\makefirstuc`[8] or `\MakeTextUppercase`.[9]

All options of this and the following sections can be set up either as package options or via the setup command:

`\acsetup`{⟨*options*⟩}

Set up ACRO anywhere in the document. Or separate package loading from setup.

```
% with \acsetup{macros}
we could have used these before: \nato, \ny
```

---

we could have used these before: NATO, NY

5.2. Options Regarding Acronyms

The options described in this section all influence the layout of one of the possible output forms of the acronyms.

`short-format` = {⟨*format*⟩} (initially empty)

Sets a format for all short forms. For example `short-format` = {`\scshape`} would print all short forms in small caps.

`long-format` = {⟨*format*⟩} (initially empty)

The same for the long forms.

`foreign-format` = {⟨*format*⟩} (initially empty)

The format for the `foreign` entry when it appears as part of the first appearance of an acronym.

`single-format` = {⟨*format*⟩} (initially empty)

Introduced in version 2.3

The format for the acronym when it is used only once. If not specified the formatting according to `single-form` is used.

`first-long-format` = {⟨*format*⟩} (initially empty)

The format for the long form on first usage (with `\ac`, `\acf` or `\acflike` and their uppercase, plural and indefinite forms).

8. from the mfirstuc package
9. from the textcase package

`list-short-format` = {⟨*format*⟩} (initially empty)
An extra format for the short entries in the list. If not used this is the same as `short-format`. Please be aware that a call of `short-format` after this one will overwrite it again.

`list-short-width` = {⟨*dim*⟩} Default: `3em`
Introduced in version 2.1
This option controls the width reserved for the short forms of the acronyms in the `lof` list style.

`list-long-format` = {⟨*format*⟩} (initially empty)
An extra format for the long entries in the list. If not used this is the same as `long-format`. Please be aware that a call of `long-format` after this one will overwrite it again.

`list-foreign-format` = {⟨*format*⟩} Default: `\acroenparen`
The format for the `foreign` entry as it appears in the list. This may be code that ends with a macro that takes a mandatory argument.

`extra-format` = {⟨*format*⟩} (initially empty)
The same for the extra information.

`first-style` = `default | empty | square | short | long | reversed | footnote | sidenote | footnote-reversed|sidenote-reversed` Default: `default`
The basic style of the first appearance of an acronym. The value `sidenote` needs the command `\sidenote` to be defined (for example by the sidenotes package).

`extra-style` = `default|plain|comma|paren|bracket` Default: `default`
Defines the way the extra information is printed in the list.

`plural-ending` = {⟨*short*⟩`/`⟨*long*⟩} Default: `s/s`
Changed in version 2.4b
With this option the default plural ending can be set. The appended ⟨*long*⟩ ending is optional. If you leave it (*and* the `/`) the ⟨*short*⟩ ending is used for both short and long versions.

```
% (Keep in mind that we're in
% a minipage here!)
\acsetup{first-style=empty}
empty: \acf{ny} \\
\acsetup{first-style=footnote}
footnote: \acf{ny} \\
\acsetup{first-style=square}
square: \acf{ny} \\
\acsetup{first-style=short}
short: \acf{ny} \\
\acsetup{first-style=long}
long: \acf{ny} \\
\acsetup{first-style=reversed}
reversed: \acf{ny} \\
\acsetup{
  first-style = footnote-reversed
}
footnote-reversed: \acf{ny}
```

empty: NY
footnote: NY[a]
square: New York [NY] [Wik12]
short: NY [Wik12]
long: New York [Wik12]
reversed: NY (New York) [Wik12]
footnote-reversed: New York[b]

a. New York [Wik12]
b. NY [Wik12]

5.3. Options Regarding the List

`page-style` = `none|plain|comma|paren` Default: `none`

If this option is set to a value other than `none` the page numbers of the an acronym appeared on are printed in the list. Please note that this is an experimental feature and might fail in quite a number of cases. If you notice anything please send me an email!

`pages` = `all|first` Default: `all`

If the option `page-style` has any value other than `none` this option determines wether all usages of the acronyms are listed or only the first time. Implicitly sets `label` = `{true}`.

`page-name` = `{`⟨*page name*⟩`}` Default: `p.\@\,`

The "name" of the page label. This is automatically translated to the active language. However for the time being there are many translations missing, yet. Please notify me if you find your language missing.

`pages-name` = `{`⟨*page name plural*⟩`}` Default: `pp.\@\,`

The "name" of the page label when there are more than one page. This is automatically translated to the active language. However for the time being there are many translations missing, yet. Please notify me if you find your language missing.

`following-page` = `true|false` Default: `false`

If set to `true` a page range in the list of acronyms that consists of two pages will be written by the first page and an appended `f`. This depends on the option `next-page`.

`following-pages` = `true|false` Default: `false`

If set to `true` a page range in the list of acronyms that set consists of more than two pages will be written by the first page and an appended `ff`. This depends on the option `next-pages`.

`following-pages*` = `true`|`false` Default: `false`

Introduced in version 2.5

If set to `true` this sets both options `following-page` = {`true`} and `following-pages` = {`true`}. `false` sets `following-page` = {`false`} and `following-pages` = {`false`}.

`next-page` = {⟨*text*⟩} Default: `\,f.\@`

Appended to a page number when `following-page` is set to `true` and the range is only 2 pages long. This is automatically translated to the active language. However, for the time being there are many translations missing, yet. Please notify me if you find your language missing.

`next-pages` = {⟨*text*⟩} Default: `\,ff.\@`

Appended to a page number when `following-pages` is set to `true` and the range is more than 2 pages long. This is automatically translated to the active language. However, for the time being there are many translations missing, yet. Please notify me if you find your language missing.

`list-style` = `description` | `lof` | `longtable` | `extra-longtable` | `extra-longtable-rev` | `extra-tabular`|`extra-tabular-rev`|`tabular`|`toc` Default: `description`

Changed in version 2.2

Choose with which style the list of acronyms should be typeset. If you choose ⟨*longtable*⟩, `extra-longtable` or `extra-longtable-rev` you have to load the longtable [Car14] package in your preamble. The values `extra-`⟨*something*⟩ put the extra information in a column of it own. *Be aware that per default* all `extra-table` *styles only use* `l` *columns. Since acronym descriptions can easily get longer that a line you should probably define your own style if you want to use them.* See section 13.2 on page 35 for details.

`list-heading` = `chapter` | `chapter*` | `section` | `section*` | `subsection` | `subsection*` | `subsubsection`|`subsubsection*`|`addchap`|`addsec`|`none` Default: `section*`

Changed in version 2.0

The heading type of the list. The last two only work with a KOMA-Script class that also defines the appropriate command. A special value is `none` which suppresses the heading.

`list-name` = {⟨*list name*⟩} Default: `Acronyms`

The name of the list. This is what's written in the list-heading. This is automatically translated to the active language. However, for the time being there are many translations missing, yet. Please notify me if you find your language missing.

`list-caps` = `true`|`false` Default: `false`

Print the first letters of the long form capitalized.

6. Trailing Tokens and Special Action

Introduced in version 2.0

ACRO has the possibility to look ahead for certain tokens and switch a boolean if it finds them. Per default ACRO knows about three tokens: the "`dot`" (`.`), the "`dash`" (`-`) and the "`babel-hyphen`" (`\babelhyphen`).

A token is made known to ACRO with the following macro:

`\AcroRegisterTrailing`⟨*token*⟩{⟨*name*⟩}

This registers the token ⟨*token*⟩ so ACRO looks if it follows directly after an acronym macro. ⟨*name*⟩ is the internal name for this token.

The ACRO package already registers the above mentioned tokens:

```
\AcroRegisterTrailing . {dot}
\AcroRegisterTrailing - {dash}
\AcroRegisterTrailing \babelhyphen {babel-hyphen}
```

If a token is registered it doesn't mean that ACRO looks for it. The token must first be activated for this:

`activate-trailing-tokens` = {⟨*csv list of token names*⟩}
Tell ACRO to look for trailing tokens. This is done by giving a csv list of the internal *names* of the tokens. Per default only `dot` is activated.

`deactivate-trailing-tokens` = {⟨*csv list of token names*⟩}
Tell ACRO not to look for trailing tokens. This is done by giving a csv list of the internal *names* of the tokens.

All of the above on its own does nothing visible. However: inside of an acronym, *i. e.*, for example inside the long or the short form it can be tested for those trailing tokens:

`\aciftrailing`{⟨*csv list of token names*⟩}{⟨*true*⟩}{⟨*false*⟩}
Check if one of the tokens listed in ⟨*csv list of token names*⟩ is following and either place ⟨*true*⟩ or ⟨*false*⟩ in the input stream.

ACRO uses this to define to further macros:

`\acdot`
Inserts a `.` if no `dot` follows.

`\acspace`
Inserts a `\space` if no `dash` or `babel-hyphen` follows.

The definitions are equivalent[10] to the following code:

```
\newcommand*\acdot{\aciftrailing{dot}{}{.\@}}
\newcommand*\acspace{\aciftrailing{dash,babel-hyphen}{}{\space}}
```

This could be used to define an acronym as follows:

10. Not *quite*: ACRO's definitions are engine protected.

```
\DeclareAcronym{etc}{
  short = {\textit{etc}\acdot} ,
  long  = {\textit{et cetera}} ,
  short-plural = , long-plural =
}
```

If now you somewhere use

```
\ac{etc}.
```

there won't be two dots printed.

The command \acspace is used already in the definition of the first appearance of a macro. Let's say you're a German chemist and you have

```
\DeclareAcronym{PU}{
  long = Polyurethan ,
  long-plural = e
}
```

and you use it the first time like this:

```
\ac{PU}-Hartschaum
```

then according to German orthography and typesetting rules this should be printed as

"Polyurethan(PU)-Hartschaum"

i. e., with *no* space between long and short form. This is exactly what happens it you say

```
\acsetup{activate-trailing-tokens = {dash,babel-hyphen}}
```

in the preamble.

7. About Page Ranges

If you enable the `page-style` option ACRO adds page numbers to the list of acronyms. In version 0.* it would add a page reference for an acronym in the list of acronyms that used `\pageref` to refer to the first appearance of an acronym. This is retained using `pages = {first}`. Version 1.0 uses a different approach that doesn't use a label but instead will list *all* pages an acronym appeared on. With hyperref the pages are referenced using `\hyperpage`.

There are some options that control how this list will be typeset, e.g., `following-page`, `next-pages` or the option `page-style` itself. It is important to mention that the page list will always take at least two compilation runs until changes in the options or the actual page numbers affect it. This is due to the fact that the updated sequence is first written to the `aux` file and only read in during the next run.

8. Dividing Your Document Into Pieces – Creating Local Lists

Introduced in version 2.4

ACRO introduces the concept of *barriers* which can divide the document into parts. It is possible to create lists of only those acronyms used between two such barriers.

`\acbarrier`
Sets a barrier at the point of use in the document. The begin and the end of the document mark implicit barriers.

`use-barriers = true|false` Default: `false`

Introduced in version 2.5

If you want to use barriers and local lists you have to activate the feature first. This should be set in the preamble in order to work reliably. Make sure to watch out for log file messages asking you to rerun.

`reset-at-barriers = true|false` Default: `false`
If this option is set to `true` `\acbarrier` implicitly calls `\acresetall`.

`local-to-barriers = true|false` Default: `false`
This option can *only* be used as option to the `\printacronyms` command. It then prints a list of only the acronyms of the current "barrier group".

```
\acbarrier
\printacronyms[local-to-barriers]
\ac{ctan} and \ac{lppl}
\acbarrier
```

Acronyms

ctan Comprehensive TEX Archive Network

lppl LATEX Project Public License

CTAN and LPPL

9. Language Support

ACRO detects if packages babel [Bra16] or polyglossia [Bra16] are being loaded and tries to adapt certain strings to match the chosen language. However, due to my limited language knowledge only a few translations are provided. I'll show how the English translations are defined so you can add the translations to your preamble if needed. Even better would be you'd send me a short email to `contact@mychemistry.eu` with the appropriate translations for your language and I'll add them to ACRO.

```
\DeclareTranslation{English}{acronym-list-name}{Acronyms}
\DeclareTranslation{English}{acronym-page-name}{p.}
\DeclareTranslation{English}{acronym-pages-name}{pp.}
\DeclareTranslation{English}{acronym-next-page}{f.}
\DeclareTranslation{English}{acronym-next-pages}{ff.}
```

10. hyperref Support

The option `hyperref = {true}` adds internal links from all short (or alternative) forms to their respective list entries. Of course this only works if you have loaded the hyperref package in your preamble. You should use this option with care: if you don't use `\printacronyms` anywhere this option will result in loads of hyperref warnings. Also printing several lists can result in warnings if don't clearly separate the lists into different classes. If an acronym appears in more than one list there will also be more than one hypertarget for this acronym.

Using hyperref will also add `\hyperpage` to the page numbers in the list (provided they are displayed in the style chosen). Like with an index the references will thus not point to the acronyms directly but to the page they're on.

11. Defining Own Acronym Macros

Introduced in version 2.0

The commands explained in section 2.3 on page 7 have all been defined with a dedicated command – there is a family of dedicated commands, actually:

`\NewAcroCommand{`⟨*cs*⟩`}{`⟨*code*⟩`}`
Defines a new ACRO acronym command ⟨*cs*⟩. This sets up the necessary framework needed by acronym commands and defines ⟨*cs*⟩ with an optional star argument and a mandatory argument for the acronym id using xparse's `\NewDocumentCommand`. Inside ⟨*code*⟩ one can refer to the ID ⟨*id*⟩ with `#1`.

`\RenewAcroCommand{`⟨*cs*⟩`}{`⟨*code*⟩`}`
Defines a new ACRO acronym command ⟨*cs*⟩. This sets up the necessary framework needed by acronym commands and defines ⟨*cs*⟩ with an optional star argument and a mandatory argument for the acronym id using xparse's `\RenewDocumentCommand`. Inside ⟨*code*⟩ one can refer to the ID ⟨*id*⟩ with `#1`.

`\DeclareAcroCommand{`⟨*cs*⟩`}{`⟨*code*⟩`}`
Defines a new ACRO acronym command ⟨*cs*⟩. This sets up the necessary framework needed by acronym commands and defines ⟨*cs*⟩ with an optional star argument and a mandatory argument for the acronym id using xparse's `\DeclareDocumentCommand`. Inside ⟨*code*⟩ one can refer to the ID ⟨*id*⟩ with `#1`.

`\ProvideAcroCommand{`⟨*cs*⟩`}{`⟨*code*⟩`}`
Defines a new ACRO acronym command ⟨*cs*⟩. This sets up the necessary framework needed by acronym commands and defines ⟨*cs*⟩ with an optional star argument and a mandatory argument for the acronym id using xparse's `\ProvideDocumentCommand`. Inside ⟨*code*⟩ one can refer to the ID ⟨*id*⟩ with `#1`.

Inside these macros one can use a number of low-level expl3 commands.[11]

Acronym fetching commands

`\acro_use:n {`⟨*id*⟩`}`
Fetches the acronym using either the first or the short form depending on earlier uses.

`\acro_short:n {`⟨*id*⟩`}`
Fetches the short form of the acronym.

`\acro_long:n {`⟨*id*⟩`}`
Fetches the long form of the acronym.

`\acro_alt:n {`⟨*id*⟩`}`
Fetches the alternative short form of the acronym.

`\acro_foreign:n {`⟨*id*⟩`}`
Fetches the foreign property of the acronym if available.

`\acro_extra:n {`⟨*id*⟩`}`
Fetches the extra property of the acronym if available.

11. Which is why you need to use them inside an expl3 programming environment. This means in the preamble surround the definitions with `\ExplSyntaxOn` and `\ExplSyntaxOff`.

Acronym setup commands

`\acro_first_upper:`
ACRO setup command which tells the macros above that we want to uppercase the first letter of the long version. Should be used *before* one of the acronym fetching commands.

`\acro_plural:`
ACRO setup command which tells the macros above that we want to use plural forms. Should be used *before* one of the acronym fetching commands.

`\acro_indefinite:`
ACRO setup command which tells the macros above that we want to add the indefinite article. Should be used *before* one of the acronym fetching commands.

`\acro_cite:`
ACRO setup command which tells the macros above that we want to add the citation in any case independent of the option `cite`. Should be used *before* one of the acronym fetching commands.

`\acro_no_cite:`
ACRO setup command which tells the macros above that we want to have no citation independent of the option `cite`. Should be used *before* one of the acronym fetching commands.

`\acro_index:`
ACRO setup command which tells the macros above that we want to add an index entry in any case independent of the option `index`. Should be used *before* one of the acronym fetching commands.

`\acro_reset_specials:`
Introduced in version 2.0b
This macro is called implicitly by `\NewAcroCommand` and `\NewPseudoAcroCommand`. If you plan to define an ACRO command by yourself using `\NewDocumentCommand` this should be the first macro after `\acro_begin:`. It ensures that in nested acronyms the inner acronyms don't inherit indefinite articles, uppercasing, endings…

Additional macros for further uses

`\acro_begin:`
When an acronym macro is defined "by hand", *i. e.*, *not using* `\NewAcroCommand` then this must be the first macro in the code. *Must have a matching* `\acro_end:`.

`\acro_end:`
When an acronym macro is defined "by hand", *i. e.*, *not using* `\NewAcroCommand` then this must be the last macro in the code. *Must have a matching* `\acro_begin:`.

`\acro_check_and_mark_if:nn {`⟨*boolean expression*⟩`} {`⟨*id*⟩`}`
Checks if the acronym with the ID ⟨*id*⟩ exists and marks it as used when ⟨*boolean*⟩ expression evaluates to `true`. This macro is used inside `\NewAcroCommand` and friends implicitly.

`\acro_check_acronym:nn` {⟨*id*⟩} {`true|false`}
Checks if the acronym with the ID ⟨*id*⟩ exists and marks it as used if `true` or doesn't. This macro is used inside `\acro_check_and_mark_if:nn`.

`\acro_use_acronym:n` {`true|false`}
Tell `\acro_use:n` and similar commands wether to mark the acronym as used or not. This macro is used inside `\acro_check_acronym:nn`. If this macro is used explicitly it should be used before `\acro_use:n` (or a similar command) otherwise it has no effect. An acronym marked as used cannot be unmarked.

`\acro_mark_as_used:n` {⟨*id*⟩}
Explicitly use the acronym with the ID ⟨*id*⟩. This is the expl3 macro applied to all entries in `\acuse`.

* `\acro_if_acronym_used:nTF` {⟨*id*⟩} {⟨*true*⟩} {⟨*false*⟩}
The code-level version of `\acifused`. This macro is expandable.

`\acro_for_all_acronyms_do:n` {⟨*code*⟩}
Loops over all acronyms known when the macro is used. Inside of ⟨*code*⟩ you can refer to the ID ⟨*id*⟩ of an acronym with `#1`.

`\acro_barrier:`
The code-level version of `\acbarrier`.

`\acro_switch_off:`
Introduced in version 2.6
The expl3 version of `\acswitchoff`.

`\acro_switch_on:`
Introduced in version 2.6
The expl3 version of `\acswitchon`.

`\acro_add_action:n` {⟨*code*⟩}
Introduced in version 2.7
Adds code to `\acro_get:n`. Inside of ⟨*code*⟩ you can refer to the ID of the acronym with `#1`.

`\acro_get_property:nnTF` {⟨*id*⟩} {⟨*property*⟩} {⟨*true*⟩} {⟨*false*⟩}
Introduced in version 2.7
Fetches the property ⟨*property*⟩ of the acronym ⟨*id*⟩ and stores it in a tokenlist variable `\l__acro_`⟨*property*⟩`_tl` where all dashes in the property names are replaced with underscores. ⟨*true*⟩ is placed in the input stream if the property had been set, ⟨*false*⟩ otherwise.

`\acro_get_property:nn`{⟨*id*⟩} {⟨*property*⟩}
Introduced in version 2.7
Like `\acro_get_property:nnTF`, but without the ⟨*true*⟩ and ⟨*false*⟩ arguments.

`\acro_if_property:nnTF` {⟨*id*⟩} {⟨*property*⟩} {⟨*true*⟩} {⟨*false*⟩}
Introduced in version 2.7
Checks if the property ⟨*property*⟩ of the acronym ⟨*id*⟩ is set and places ⟨*true*⟩ is in the input stream if yes and ⟨*false*⟩ otherwise.

Examples The usage of above macros is best explained with a few examples. The following definition is done by ACRO:

```
\NewAcroCommand \ac { \acro_use:n {#1} }
```

An equivalent definition for \ac would be

```
\NewDocumentCommand \ac {sm}
  {
    \acro_begin:
      \acro_reset_specials:
      \acro_check_and_mark_if:nn {#1} {#2}
      \acro_use:n {#2}
    \acro_end:
  }
```

which should explain what the actual framework is which \NewAcroCommand adds.

Other definitions by ACRO are for example the following ones:

```
\NewAcroCommand \Ac
  {
    \acro_first_upper:
    \acro_use:n {#1}
  }
\NewAcroCommand \iac
  {
    \acro_indefinite:
    \acro_use:n {#1}
  }
\NewAcroCommand \acp
  {
    \acro_plural:
    \acro_use:n {#1}
  }
\NewAcroCommand \Acp
  {
    \acro_plural:
    \acro_first_upper:
    \acro_use:n {#1}
```

```
  }
\NewAcroCommand \Aclp
  {
    \acro_plural:
    \acro_first_upper:
    \acro_long:n {#1}
  }
```

12. About Plural Forms, Possessive Forms and Similar Constructs – the Concept of Endings

ACRO has a concept of *endings*. All of ACRO's plural options are defined by saying

```
\ProvideAcroEnding {plural} {s} {s}
```

The command's syntax and what it does is as follows:

`\ProvideAcroEnding{`⟨*name*⟩`}{`⟨*short default*⟩`}{`⟨*long default*⟩`}`

This macro defines the options

- ⟨*name*⟩`-ending`,
- `short-`⟨*name*⟩`-ending`,
- `alt-`⟨*name*⟩`-ending` and
- `long-`⟨*name*⟩`-ending`.

It also defines the acronym properties

- `short-`⟨*name*⟩,
- `short-`⟨*name*⟩`-form`,
- `alt-`⟨*name*⟩,
- `alt-`⟨*name*⟩`-form`,
- `long-`⟨*name*⟩ and
- `long-`⟨*name*⟩`-form`.

Additionally it defines a setup macro as described in section 11 on page 28, `\acro_`⟨*name*⟩`:`. If ⟨*name*⟩ contains a `-` (dash) it is replaced by `_` before `\acro_`⟨*name*⟩`:` is built. So if you choose `my-name` the corresponding macro is named `\acro_my_name:`. If you use any other non-letters you are on your own. If you use the command with the same ⟨*name*⟩ a second time the command only resets the defaults.

Changed in version 2.4b

Note that you *must use* `\ProvideAcroEnding` *before any acronym definition*!

This could be used together with the macros described in section 11 on page 28 for adding support for possessive forms:

```
\ExplSyntaxOn
% this now only works because I've used the same already in the preamble so
% it does nothing here:
\ProvideAcroEnding {possessive} {'s} {'s}

\ProvideAcroCommand \acg
  {
    \acro_possessive:
    \acro_use:n {#1}
  }
\ExplSyntaxOff
The \acg{cd} booklet says\ldots
```

The CD's booklet says…

Please note that different endings are cumulative which you probably want to avoid! Imagine a macro

```
\NewAcroCommand \acgp
  {
    \acro_possessive:
    \acro_plural:
    \acro_use:n {#1}
  }
```

This would give "CDs's" instead of "CDs'". To solve this you might want to consider

```
  \ProvideAcroEnding {possessive-singular} {'s} {'s}
  \ProvideAcroEnding {possessive-plural}   {s'} {s'}
```

13. More on Customization

13.1. Background Information

Several of ACRO's objects are customized using templates. For each of these objects it is possible to define own templates.[12] Possibly more interesting: it is easily possible to define further instances of an object using a certain template. How this works is explained in the following sections. However, the basics are always the same. There is a command

`\DeclareAcro`⟨*object type*⟩`Style{`⟨*name*⟩`}{`⟨*template*⟩`}{`⟨*options*⟩`}`
which allows to define a new style (*i. e.*, instance) for the object ⟨*object type*⟩ using the template `{`⟨*template*⟩`}`.

13.2. Lists

13.2.1. Own List Style

The different existing list styles are all built from four different templates, `list`, `list-of`, `table` and `extra-table`. Those templates are defined with the help of the xtemplate package (from [L3P]). Each of these templates has a few options which are described in table 1 on the next page. New list styles now are defined via the following macro:

`\DeclareAcroListStyle{`⟨*name*⟩`}{`⟨*template*⟩`}{`⟨*options*⟩`}`
Declares a new ACRO list style ⟨*name*⟩. ⟨*name*⟩ will be the value which can be chosen in the option `list-style`. ⟨*template*⟩ is the name of the template to be used by the style. Available templates are listed in table 1. ⟨*options*⟩ are the option settings for the corresponding template.

For defining new styles you need some information on what the different templates and options do:

- The option `list` of the `list` template sets the list environment. This must be a classic LaTeX list where items are listed with `\item`. In those lists short entries will always be fed as optional argument to `\item`:
  `\item[`⟨*short*⟩`]`⟨*long*⟩⟨*extra*⟩⟨*page*⟩
- The template `list-of` simulates a table of contents or a list of figures. This can be chosen by setting the option `style` to either `toc` or `lof`.
- The template `table` typesets the list in a table with two columns:
  ⟨*short*⟩ & ⟨*long*⟩⟨*extra*⟩⟨*page*⟩ `\tabularnewline`
- The template `extra-table` typesets the list in a table with four columns:
  ⟨*short*⟩ & ⟨*long*⟩ & ⟨*extra*⟩ & ⟨*page*⟩ `\tabularnewline`
- The option `foreign-sep` is the code inserted between long form and foreign entry (if a foreign entry is present).

12. This requires some knowledge of xtemplate and expl3. Plans are to provide a documented interface for users of ACRO in the future.

TABLE 1: Available List Templates and Their Options

Template	Option	Option Type	Default
list	list	tokenlist	description
	foreign-sep	tokenlist	\space
	reverse	boolean	false
	before	tokenlist	
	after	tokenlist	
list-of	style	tokenlist	toc
	foreign-sep	tokenlist	\space
	reverse	boolean	false
	before	tokenlist	
	after	tokenlist	
table	table	tokenlist	tabular
	table-spec	tokenlist	lp{.7\linewidth}
	foreign-sep	tokenlist	\space
	reverse	boolean	false
	before	tokenlist	
	after	tokenlist	
extra-table	table	tokenlist	tabular
	table-spec	tokenlist	llll
	foreign-sep	tokenlist	\space
	reverse	boolean	false
	before	tokenlist	
	after	tokenlist	

- The options `before` and `after` are inserted directly before and after the complete list.
- The option `reverse` switches the place of ⟨*long*⟩ with ⟨*extra*⟩.
- The option `table-spec` sets the column types for the table templates. It must correspond to the number of columns the corresponding template uses.

As an example let's define a style `longtabu` which uses the corresponding table environment from the package tabu [Che11]:

```
\usepackage{tabu,longtable}
\DeclareAcroListStyle{longtabu}{table}{
  table = longtabu ,
  table-spec = @{}>{\bfseries}lX@{}
}
\acsetup{list-style=longtabu}
```

As another example let's define a new list with the help of the enumitem package [Bez11]:

```
% preamble:
% \usepackage{enumitem}
\newlist{acronyms}{description}{1}
\newcommand*\addcolon[1]{#1:}
\setlist[acronyms]{
  labelwidth = 3em,
  leftmargin = 3.5em,
  noitemsep,
  itemindent = 0pt,
  font=\addcolon}
\DeclareAcroListStyle{mystyle}{list}{ list = acronyms }
\acsetup{ list-style = mystyle }
```

This would look as follows:

Acronyms

cd: Compact Disc
ctan: Comprehensive TEX Archive Network
ECU: Steuergerät (Electronic Control Unit)
id: identification string
JPEG: Joint Photographic Experts Group

LA: Los Angeles
lppl: LaTeX Project Public License
MP: Member of Parliament
nato: North Atlantic Treaty Organization. *deutsch*: Organisation des Nordatlantikvertrags
NY: New York
pdf: Portable Document Format
SW: Sammelwerk
ST: Some Test
UFO: unidentified flying object

13.2.2. Own List Heading Command

With the option `list-heading` you can choose which command prints the heading of the list. If you need a different choice than what's already provided you can use the following command to define a new option:

`\DeclareAcroListHeading{`⟨*name*⟩`}{`⟨*control sequence*⟩`}`
Defines a new value ⟨*name*⟩ for the option `list-heading`. ⟨*control sequence*⟩ must be a control sequence which takes one mandatory argument.

As an example here is how the value `section` is defined:

```
\DeclareAcroListHeading{section}{\section}
```

13.3. First Styles

The first styles define how an acronym is typeset when it is used for the first time. It is set with the option `first-style`. Legal values for this option are defined with the following command:

`\DeclareAcroFirstStyle{`⟨*name*⟩`}{`⟨*template*⟩`}{`⟨*options*⟩`}`
Declares a new ACRO first style ⟨*name*⟩. ⟨*name*⟩ will be the value which can be chosen in the option `first-style`. ⟨*template*⟩ is the name of the template to be used by the style. Available templates are listed in table 2. ⟨*options*⟩ are the option settings for the corresponding template.

Here are two examples of the already available styles and how they are defined:

```
\DeclareAcroFirstStyle{short}{inline}{
  only-short = true ,
  brackets   = false
}
\DeclareAcroFirstStyle{sidenote-reversed}{note}{
  note-command = \sidenote{#1} ,
  reversed     = true
```

```
}
```

Table 2: Available First Style Templates and Their Options

Template	Option	Option Type	Default
inline	brackets	boolean	true
	brackets-type	tokenlist	()
	only-short	boolean	false
	only-long	boolean	false
	reversed	boolean	false
	between	tokenlist	
	foreign-sep	tokenlist	,~
note	use-note	boolean	true
	note-command	function	\footnote{#1}
	reversed	boolean	false
	foreign-sep	tokenlist	,~

13.4. Extra Styles

The extra styles define how the extra information of an acronym is typeset in the list. It is set with the option `extra-style`. Legal values for this option are defined with the following command:

`\DeclareAcroExtraStyle{`⟨*name*⟩`}{`⟨*template*⟩`}{`⟨*options*⟩`}`

Declares a new ACRO extra style ⟨*name*⟩. ⟨*name*⟩ will be the value which can be chosen in the option `extra-style`. ⟨*template*⟩ is the name of the template to be used by the style. Available templates are listed in table 3. ⟨*options*⟩ are the option settings for the corresponding template.

Here are two examples of the already available styles and how they are defined:

```
\DeclareAcroExtraStyle{default}{inline}{
  brackets     = false ,
  punct        = true ,
  punct-symbol = .
}
\DeclareAcroExtraStyle{paren}{inline}{
  brackets     = true ,
  punct        = true ,
  punct-symbol =
```

TABLE 3: Available Extra Style Templates and Their Options

Template	Option	Option Type	Default
inline	punct	boolean	true
	punct-symbol	tokenlist	,
	brackets	boolean	true
	brackets-type	tokenlist	()

```
}
```

13.5. Page Number Styles

The page number styles define how the page numbers where acronyms have been used are typeset in the list. It is set with the option page-style. Legal values for this option are defined with the following command:

\DeclareAcroPageStyle{⟨*name*⟩}{⟨*template*⟩}{⟨*options*⟩}
Declares a new ACRO extra style ⟨*name*⟩. ⟨*name*⟩ will be the value which can be chosen in the option page-style. ⟨*template*⟩ is the name of the template to be used by the style. Available templates are listed in table 4. ⟨*options*⟩ are the option settings for the corresponding template.

Here are two examples of the already available styles and how they are defined:

```
\DeclareAcroPageStyle{default}{inline}{
  punct = true ,
  punct-symbol = .
}
\DeclareAcroPageStyle{paren}{inline}{
  brackets=true ,
  punct-symbol = ~
}
```

13.6. Configuration Files

Introduced in version 2.2

If you repeatedly have the same setup and definitions for ACRO in your preamble[13] you might want to place those in a configuration file. If ACRO finds a file named acro.cfg present it inputs it at the end of the package. The only thing to be aware of is that this file is input like a package which means that @ is treated as a letter (category code 11).

13. For example defining new endings, ACRO commands, list styles, …

A. All Acronyms Used in this Documentation

cd Compact Disc, pp. 8 f., 34

ctan Comprehensive TEX Archive Network, pp. 2, 27

ECU Steuergerät (Electronic Control Unit), p. 11

id identification string, pp. 2, 4 f., 13 f., 19 f., 28 ff.

JPEG Joint Photographic Experts Group, p. 10

LA Los Angeles, p. 14

lppl LATEX Project Public License, pp. 2, 27

MP Member of Parliament, p. 9

nato North Atlantic Treaty Organization, *deutsch*: Organisation des Nordatlantikvertrags, pp. 10, 21

NY New York, pp. 6, 9, 13 f., 21 f.

pdf Portable Document Format, pp. 4, 15 f.

SW Sammelwerk, p. 9

ST Some Test, p. 5

UFO unidentified flying object, p. 12

B. References

[Bez11] Javier Bezos. enumitem. version 3.5.2, Sept. 28, 2011 (or newer). URL: http://mirror.ctan.org/macros/latex/contrib/enumitem/.

Table 4: Available Page Number Style Templates and Their Options

Template	Option	Option Type	Default
inline	display	boolean	true
	punct	boolean	false
	punct-symbol	tokenlist	,
	brackets	boolean	false
	brackets-type	tokenlist	()
	space	skip	.333333em plus .166666em minus .111111em

[Bra16] Johannes Braams, current maintainer: Javier Bezos.
babel. version 3.9q, Feb. 24, 2016 (or newer).
URL: http://mirror.ctan.org/macros/latex/required/babel/.

[Car14] David Carlisle. longtable. version 4.11, Oct. 28, 2014 (or newer).
URL: http://mirror.ctan.org/macros/latex/required/tools/.

[Cha15] François Charette, current maintainer: Arthur Reutenauer.
polyglossia. version 1.42.0, Aug. 6, 2015 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/polyglossia/.

[Che11] Florent Chervet. tabu. version 2.8, Feb. 26, 2011 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/tabu/.

[Kle12] Josef Kleber. pdfcomment. version 2.3a, Sept. 28, 2012 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/pdfcomment/.

[L3P] The LaTeX3 Project Team.
l3packages. version SVN 6377, Jan. 19, 2016 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/l3packages/.

[Nie15] Clemens Niederberger. translations. version 1.2e, Nov. 7, 2015 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/translations/.

[Tal15] Nicola L.C. Talbot. mfirstuc. version 2.02, Dec. 17, 2015 (or newer).
URL: http://mirror.ctan.org/macros/latex/contrib/glossaries/.

[Wik12] Wikipedia. *New York City*. 2012.
URL: http://en.wikipedia.org/wiki/New_York_City (visited on 09/27/2012).

C. Index