# CHEMMACROS

v6.obeta-2 2021/02/13

comprehensive support for typesetting chemistry documents

Clemens NIEDERBERGER [1] Sonja K. [2]

https://github.com/cgnieder/chemmacros

## Table of Contents



1. contact@mychemistry.eu
2. SonjaK@mein.gmx

# Part I.
# Preliminaries

## 1. License

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

## 2. Motivation and Background

This package grew from a small collection of personal helper macros back in 2010 into a rather big package supporting various different chemical typesetting tasks. I hope I have achieved the following points with this package:

- Intuitive usage as far as the syntax of the commands is concerned.
- A comprehensive set of macros! If there are any needs you might have with respect to typesetting of chemistry which is not supported by this package[3] then let me know so CHEMMACROS can be extended.
- The commands shall not only make typesetting easier and faster but also the document source more readable with respect to semantics (`\ortho-dichlorobenzene` is easier to read and understand than `\textit{o}-dichlorobenzene`); the first variant in my opinion also is more in the spirit of LaTeX $2_\varepsilon$.
- As much customizability as I could think of so every user can adapt the commands to his or her own wishes. Every now and then users have wishes which can't be solved with the available options. Almost always I'll add options then. If you find something please contact me, see section B starting on page 68.
- Default settings that are compliant with the recommendations of the International Union of Pure and Applied Chemistry (IUPAC).

Especially the last point in the past needed some pushing from users to get things right in many places. If you find anything not compliant with IUPAC recommendations please contact me, see section B starting on page 68. Don't forget to add references for the corresponding IUPAC recommendation.

3. Not including needs already solved by other packages such as chemnum or chemfig.

# 3. The Structure of CHEMMACROS

## 3.1. General Structure

Introduced in version 5.0 (2015/09/11)

Since version 5.0 the CHEMMACROS package has a strictly modular structure. On the one hand this eases maintenance but it will also allow for easy and quick extension in the future. In a way it is a logical consequence from CHEMMACROS' history: since version 2.0, *i. e.*, since the fall of 2011 CHEMMACROS already had modular options.

Since version 6.0 the different modules of CHEMMACROS are divided into three groups:

1. Core modules which provide underlying functionality or basic functionality which is not of direct interest from a user perspective but might be if you plan to write a module yourself (see section A for details).
2. Main modules which provide all the stuff for typesetting and which are always loaded.
3. Additional modules which are also loaded in the default setup. They are not loaded if CHEMMACROS is loaded with the minimal setup: `\usepackage[minimal]{chemmacros}`.

## 3.2. CHEMMACROS' Options

Prior to v5.0 CHEMMACROS had quite a number of package options. CHEMMACROS v6.0 has only two:

`minimal` = `true`|`false` Default: `false`

Loads CHEMMACROS with the basic preset of modules.

`modules` = {⟨*comma separated list of module names*⟩} (initially empty)

When `minimal` is used this option allows to load additional modules.

These are load-time option that only can be used in the optional argument of `\usepackage`. *All* other of CHEMMACROS' options are set using the command

`\chemsetup[`⟨*module*⟩`]{`⟨*option list*⟩`}`

CHEMMACROS' setup command.

When an option is described then in the left margin the module the option belongs to is denoted. This looks something like this:

`module` » `option` = {⟨*value*⟩} (initially empty)

Description of `option`. The module is printed in the left margin. The default value to the right is the setting the option has when CHEMMACROS is loaded. This can be an explicit setting but the option can also be empty.

`module` » `choice-option` = `list`|`of`|`choices` Default: `list`

Description of `choice-option`. A choice option can only be used with a predefined list of values. If one of the values is underlined it means that the option can be used without value in which case the underlined value is chosen. If no value is underlined then a value *has* to be given by the user.

`module` » `boolean-option` = `true`|`false` Default: `true`

Description of `boolean-option`. A boolean option is a choice option with exactly the two values `true` and `false`. If the option is called without value then the underlined value is chosen (which is always `true` for a boolean option).

An option or list of options belonging to a module `module` can be set in two ways:

```
% first possibility:
\chemsetup[module]{
  option1 = value ,
  option2 = value
}
% second possibility:
\chemsetup{
  module/option1 = value ,
  module/option2 = value
}
```

The second way allows to set options belonging to different modules with one call of \chemsetup.

CHEMMACROS has some core options which don't belong to any of the modules described in parts II and III. Those options have no module denoted in the left margin next to their descriptions and are also set without specifying a module:

```
\chemsetup{
  option1 = value ,
  option2 = value
}
```

Some internal modules may also define core options, *e. g.*, the lang module, see section 26 starting on page 61.

## 3.3. Support Package CHEMFORMULA

CHEMFORMULA provides means of typesetting chemical formulas and reactions. You will see its macros \ch and \chcpd every now and then in this manual. When using CHEMMACROS you can consider the CHEMFORMULA package [Nie19] to be loaded as CHEMMACROS makes use of it in various places. CHEMMACROS and CHEMFORMULA are tightly intertwined. Nevertheless you should be able to use the mhchem [Hen18] package with CHEMMACROS without problems. Please see section 10.3 starting on page 23 for details and *caveats*. *The recommendation is to use CHEMFORMULA.*

A historical note: CHEMFORMULA started as a part of CHEMMACROS in January 2012. Since July 2013 it is a completely independent package – from CHEMFORMULA's point of view. It is maintained independently and has a manual of its own.

## 3.4. Upgrading from version 5.*x*

People upgrading from versions < 6.0 will find that almost everything they know from earlier versions is the same in versions 6.*x*. But there are important and *breaking* differences:

- The compatibility mode and all its commands has been dropped.
- The option modules now is a load-time option and cannot be set through \chemsetup any more. The command \usechemmodule has been dropped.
- Per default *all* modules are now loaded.
- A new module reactants has been added, thanks to Sonja K..

# Part II.
# Main Modules

The modules described in this part are always loaded by CHEMMACROS, even in the minimal setup.

## 4. The `acid-base` Module

Easy representation of pH, p$K_a$…

`\pH`
pH

`\pOH`
pOH

`\Ka`
$K_a$, depends on language settings, see section 26 starting on page 61. The translations can be adapted.

`\Kb`
$K_b$

`\Kw`
$K_w$

`\pKa[`⟨*num*⟩`]`
`\pKa`: p$K_a$, `\pKa[1]`: p$K_{a1}$, depends on language settings, see section 26 starting on page 61. The translations can be adapted.

`\pKb[`⟨*num*⟩`]`
`\pKb`: p$K_b$, `\pKb[1]`: p$K_{b1}$

`\p{`⟨*anything*⟩`}`
*e. g.* `\p{\Kw}` p$K_w$

```
1 \Ka\ \Kb\ \pKa\ \pKa[1] \pKb\ \pKb
  [1]
```

$K_a$ $K_b$ p$K_a$ p$K_{a1}$ p$K_b$ p$K_{b1}$

> The operator p […] shall be printed in Roman type.
>
> The IUPAC Green Book *[Coh+08, p. 103]*

There is one option which changes the style the p is typeset, other options allow to change the subscript of the constants:

`acid-base` » `p-style` = `italics|slanted|upright` Default: `upright`
Set the style of the p operator.

`acid-base` » `K-acid` = {⟨*text*⟩} Default: `\ChemTranslate{K-acid}`
The subscript to `\Ka` and `\pKa`.

acid-base » K-base = {⟨text⟩} Default: \ChemTranslate{K-base}
The subscript to \Kb and \pKb.

acid-base » K-water = {⟨text⟩} Default: \ChemTranslate{K-water}
The subscript to \Kw.

acid-base » eq-constant = {⟨text⟩} Default: K
Introduced in version 5.4 (2016/02/10)
The symbol of the constants.

```
\pH, \pKa \par
\chemsetup[acid-base]{p-style=slanted} \pH, \pKa \par
\chemsetup[acid-base]{p-style=italics} \pH, \pKa
```

pH, $pK_a$
*p*H, *p*$K_a$
*p*H, *p*$K_a$

As you can see the default subscripts of \Kw, \Ka and \Kb are lowercase letters. The literature is inconclusive about if this is the right way or if uppercase letters should be preferred. In textbooks the uppercase variant usually seems to be used while journals seem to prefer the lowercase variant. CHEMMACROS' default follows the usage in *The IUPAC Green Book* [Coh+08]. If you want to change this you have two possibilities:

```
% this works only in the preamble:
% \DeclareTranslation{English}{K-acid}{\mathrm{A}}% use your language here
% alternative:
\chemsetup{acid-base/K-acid=\mathrm{A}}% overwrites language dependent
  settings
\pKa
```

p$K_A$

Introduced in version 5.4
The constants $K_a$, $K_b$, and $K_w$ were defined using the following commands:

\NewChemEqConstant{⟨cs⟩}{⟨name⟩}{⟨subscript⟩}
Define the constant ⟨*cs*⟩ with the name ⟨*name*⟩ and the subscript ⟨*subscript*⟩. This also defines the default translation with the key ⟨*name*⟩ using ⟨*subscript*⟩ as fallback translation (see section 26 starting on page 61 for details). It also defines the option ⟨*name*⟩ for setting the subscript.

\RenewChemEqConstant{⟨cs⟩}{⟨name⟩}{⟨*default appearance*⟩}
The same as \NewChemEqConstant but renews an existing command.

\DeclareChemEqConstant{⟨cs⟩}{⟨name⟩}{⟨*default appearance*⟩}
The same as \NewChemEqConstant but overwrites existing commands.

\ProvideChemEqConstant{⟨cs⟩}{⟨name⟩}{⟨*default appearance*⟩}
The same as \NewChemEqConstant but doesn't throw an error if ⟨*cs*⟩ already exists.

This is how \Ka is defined:

```
\NewChemEqConstant\Ka{K-acid}{\mathrm{a}}
```

# 5. The charges Module

The charges module loads the module chemformula.

## 5.1. Charge Symbols

\fplus
⊕ formal positive charge

\fminus
⊖ formal negative charge

\scrp
+ scriptstyle positive charge (*e. g.*, for usage in chemfig's [Tel19] formulas).

\scrm
- scriptstyle negative charge (*e. g.*, for usage in chemfig's formulas).

\fscrp
⊕ scriptstyle formal positive charge (*e. g.*, for usage in chemfig's formulas).

\fscrm
⊖ scriptstyle formal negative charge (*e. g.*, for usage in chemfig's formulas).

\fsscrp
⊕ scriptscriptstyle formal positive charge (*e. g.*, for usage in chemfig's formulas).

\fsscrm
⊖ scriptscriptstyle formal negative charge (*e. g.*, for usage in chemfig's formulas).

## 5.2. Ion Charges

Simple displaying of (real) charges. It is worth noting that these commands really are relicts from a time when CHEMMACROS tried hard to be compliant with mhchem and CHEMFORMULA didn't exist, yet. **They are still provided for backwards compatibility but *my recommendation is to use* \ch (see the documentation of the CHEMFORMULA package [Nie19]) *and forget about these commands:***

\pch[⟨*number*⟩]
positive charge

\mch[⟨*number*⟩]
negative charge

\fpch[⟨*number*⟩]
formal positive charge

\fmch[⟨*number*⟩]
formal negative charge

```
A\pch\ B\mch[3] C\fpch[2] D\fmch
```

$A^{+}\ B^{3-}\ C^{2\oplus}\ D^{\ominus}$

## 5.3. Partial Charges and Similar Stuff

The next ones probably are seldomly needed but nevertheless useful:

`\delp`
$\delta+$ partial positive charge

`\delm`
$\delta-$ partial negative charge

`\fdelp`
$\delta\oplus$ partial formal positive charge

`\fdelm`
$\delta\ominus$ partial formal negative charge

These macros for example can be used with the `\ox` command (see section 19 starting on page 41) or with the chemfig package:

```
\chemsetup{
  charges/circled = all,
  redox/parse     = false,
  redox/pos       = top
}
\ch{"\ox{\delp,H}" -{} "\ox{\delm,Cl}"} \hspace*{1cm}
\chemfig{\chemabove[3pt]{\lewis{246,Br}}{\delm}-\chemabove[3pt]{H}{\delp}}
```

$\overset{\delta\oplus}{\mathrm{H}}-\overset{\delta\ominus}{\mathrm{Cl}}$ $\quad$ $\overset{\delta\ominus}{\underline{\underline{\mathrm{Br}}}}\!\!-\!\!-\overset{\delta\oplus}{\mathrm{H}}$

## 5.4. Charge Options

`charges » circled = formal|all|none` Default: `formal`

CHEMMACROS uses two different kinds of charges which indicate the usage of real (+/−) and formal (⊕/⊖) charges. The option `formal` distinguishes between them, option `none` displays them all without circle, option `all` circles all.

`charges » circletype = chem|math` Default: `chem`

This option switches between two kinds of circled charge symbols: `\fplus` ⊕/`\fminus` ⊖ (`chem`) and `$\oplus$` ⊕/`$\ominus$` ⊖ (`math`).

`charges » partial-format = {⟨LaTeX code⟩}` Default: `\tiny`

Code which formats the macros defined with `\NewChemPartialCharge` (see section 5.5).

## 5.5. Own Charge Macros

Just in case the existing macros don't fit you needs there are commands for defining new ones or modifying the existing ones. These commands define macros like those described in section 5.2 on the preceding page.

\NewChemCharge{⟨cs⟩}{⟨charge symbol⟩}
Defines a new macro ⟨cs⟩. Raises an error if ⟨cs⟩ already exists.

\RenewChemCharge{⟨cs⟩}{⟨charge symbol⟩}
Redefines a new macro ⟨cs⟩. Raises an error if ⟨cs⟩ doesn't exist.

\DeclareChemCharge{⟨cs⟩}{⟨charge symbol⟩}
Defines a macro ⟨cs⟩. Silently overwrites ⟨cs⟩ if it exists.

\ProvideChemCharge{⟨cs⟩}{⟨charge symbol⟩}
Defines a new macro ⟨cs⟩. Does nothing if ⟨cs⟩ already exists.

An example of usage is the definition of the existing ion charge macros:

```
\NewChemCharge\fpch{\fplus}
\NewChemCharge\fmch{\fminus}
```

These commands define macros like those described in section 5.3 on the previous page.

\NewChemPartialCharge{⟨cs⟩}{⟨charge symbol⟩}
Defines a new macro ⟨cs⟩. Raises an error if ⟨cs⟩ already exists.

\RenewChemPartialCharge{⟨cs⟩}{⟨charge symbol⟩}
Redefines a new macro ⟨cs⟩. Raises an error if ⟨cs⟩ doesn't exist.

\DeclareChemPartialCharge{⟨cs⟩}{⟨charge symbol⟩}
Defines a macro ⟨cs⟩. Silently overwrites ⟨cs⟩ if it exists.

\ProvideChemPartialCharge{⟨cs⟩}{⟨charge symbol⟩}
Defines a new macro ⟨cs⟩. Does nothing if ⟨cs⟩ already exists.

An example of usage is the definition of the existing partial charge macros:

```
\NewChemPartialCharge\fdelp{\fplus}
\NewChemPartialCharge\fdelm{\fminus}
```

# 6. The nomenclature Module

The nomenclature module loads the tikz module. It also loads the package scrlfile which is part of the KOMA-Script bundle [Koh19].

## 6.1. The \iupac Command

Similar to the bpchem package [Ped17] CHEMMACROS provides a command[4] for typesetting IUPAC names. Why is that useful? IUPAC names can get very long. So long indeed that they span over more than two lines, especially in two-column documents. This means they must be allowed to be broken more than one time. This is what the following command does.

4. The idea and initial implementation is shamelessly borrowed from bpchem by Bjørn PEDERSEN.

`\iupac{`⟨*IUPAC name*⟩`}`

Inside this command use `|` indicate a breaking point `^` as a shortcut for `\textsuperscript`. `-`, `(` and `)` allow words to be broken while still allow the rest of word to be hyphenated, likewise `[` and `]`.

```
\begin{minipage}{.4\linewidth}
  \iupac{%
    Tetra|cyclo[2.2.2.1^{1,4}]-un|decane-2-dodecyl-%
    5-(hepta|decyl|iso|dodecyl|thio|ester)%
  }
\end{minipage}
```

Tetracyclo[2.2.2.1$^{1,4}$]-undecane-2-dodecyl-5-(heptadecylisododecyl-thioester)

The `\iupac` command is more of a semantic command. In many cases you can achieve (nearly) the same thing by using `\-` instead of `|`, and `\textsuperscript` instead of `^` without `\iupac`. There are some important differences, though:

- The character `-` inserts a small space before the hyphen and removes a small space after it. Also usually words with hyphens are only allowed to break at the hyphen. Inside `\iupac` the hyphen will not prevent further hyphenation. The amount of inserted space can be customized.
- The character `|` not only prevents ligatures but also inserts a small space. The amount of inserted space can be customized.
- The characters `(` and `)` allow the word to be hyphenated and don't prevent further hyphenation, likewise `[` and `]`.

Introduced in version 5.3 (2016/12/04)

- The character `'` is printed as `\chemprime`.

Introduced in version 5.8c (2018/03/02)

- The character `=` is printed ad `\nonbreakinghyphen`.

```
\huge\iupac{2,4-Di|chlor|pentan} \
  par
2,4-Dichlorpentan
```

2,4-Dichlorpentan
2,4-Dichlorpentan

`\chemprime`

Introduced in version 5.3

Prints a prime character in superscript position. It is defined as `\ensuremath{{}^{\prime}}`.

`\nonbreakinghyphen`

Introduced in version 5.8c

Prints a hyphen which doesn't allow a linebreak after it. It is defined as `\mbox{-}\nobreak\hspace{0pt}`.

The spaces inserted by `-` and `|` can be customized.

`nomenclature » hyphen-pre-space = {`⟨*dim*⟩`}` Default: `.01em`

Set the space that is inserted before the hyphen set with `-`.

Table 1: Demonstration of iupac's modes.

| | auto | restricted | strict |
|---|---|---|---|
| \L | Ł | Ł | L |
| \iupac{\L} | L | L | L |
| \D | D | — | D |
| \iupac{\D} | D | D | D |

nomenclature » hyphen-post-space = {⟨*dim*⟩} Default: -.03em

Set the space that is inserted after the hyphen set with -.

nomenclature » break-space = {⟨*dim*⟩} Default: .01em

Set the space inserted by |.

The command \iupac serves another purpose, too, however. Regardless of the setting of the iupac option (see below) all the commands presented in this section are always defined *inside* \iupac. Quite a number of the naming commands have very general names: \meta, \D, \E, \L, \R, \S, \trans and so forth.[5] This means they either are predefined already (\L Ł) or are easily defined by another package or class (the cool package defines both \D and \E, for example). In order to give you control which commands are defined in which way, there is the option iupac:

nomenclature » iupac = auto|restricted|strict Default: auto

Take care of how IUPAC naming commands are defined.

It has three modes:

- iupac = {auto}: if the commands are *not* defined by any package or class you're using they are available generally, otherwise only *inside* \iupac.
- iupac = {restricted}: all naming commands are *only* defined inside \iupac. If the commands are defined by another package they of course have that meaning outside. They're not defined outside otherwise.
- iupac = {strict}: CHEMMACROS overwrites any other definition and makes the commands available throughout the document. Of course the commands can be redefined (but only in the document body). They will still be available inside \iupac then.

Table 1 demonstrates the different modes.

## 6.2. Macros Defined (Not Only) For Usage in \iupac

## 6.3. One-letter Macros

For some of the macros explained in this section one-letter commands are defined – with a *caveat* in mind, though: they are not actively recommended. One-letter commands seldomly have meaningful names and often they've also been defined by other packages. This means they make collaboration more difficult than it needs to be and are a source for package conflicts. CHEMMACROS solves the latter problem by only providing them inside the argument of \iupac. The one exception CHEMMACROS makes is the command \p (for things like pH) which is and will remain an official command (see section 4 starting on page 6). For all other one-letter macros alternatives with more meaningful names exist.

5. Please read section 6.3 before you consider using the one-letter commands

Table 2: IUPAC shortcuts for Greek letters.

| macro | \a | \b | \g | \d | \k | \m | \n | \w |
|---|---|---|---|---|---|---|---|---|
| letter | α | β | γ | δ | κ | μ | η | ω |

## 6.4. Greek Letters

Greek letters in compound names are typeset upright. Here are a few examples for the existing macros:

`\chemalpha` α
Upright lowercase alpha

`\chembeta` β
Upright lowercase alpha

`\chemgamma` γ
Upright lowercase alpha

`\chemdelta` δ
Upright lowercase alpha

There exist two commands for each of the twenty-four Greek letters: a lowercase and an uppercase version (`\chemalpha` and `\chemAlpha`). Those commands are actually provided by the CHEMGREEK package. For more details read section 11 starting on page 24 and also refer to CHEMGREEK's documentation.

There are a number of one-letter commands that some people may find convenient to use which use above mentioned commands to print Greek letters inside `\iupac`. They're listed in table 2.

```
\iupac{5\chemalpha-androstan-3\chembeta-ol} \par
\iupac{\chemalpha-(tri|chloro|methyl)-\chemomega
  -chloro|poly(1,4-phenylene|methylene)}
```

5α-androstan-3β-ol
α-(trichloromethyl)-ω-chloropoly(1,4-phenylenemethylene)

## 6.5. Hetero Atoms and added Hydrogen

Attachments to hetero atoms and added hydrogen atoms are indicated by italic letters [Coh+08]. CHEMMACROS defines a few macros for the most common ones.

`\hydrogen` *H*
The italic H for hydrogen. An alias for this command is `\H`.

`\oxygen` *O*
The italic O for oxygen. An alias for this command is `\O`.

`\nitrogen` *N*
The italic N for nitrogen. An alias for this command is `\N`.

\sulfur *S*
The italic S for sulfur. An alias for this command is \Sf.

\phosphorus *P*
The italic P for phosphorus. An alias for this command is \P.

```
\iupac{\nitrogen-methyl|benz|amide}

\iupac{3\hydrogen-pyrrole}

\iupac{\oxygen-ethyl hexanethioate}
```

*N*-methylbenzamide
3*H*-pyrrole
*O*-ethyl hexanethioate

## 6.6. Cahn-Ingold-Prelog

\cip{⟨*conf*⟩}
Typeset Cahn-Ingol-Prelog descriptors, *e. g.*: \cip{R,S} (*R*,*S*). ⟨*conf*⟩ may be a csv list of entries.

\rectus (*R*)
The rectus descriptor. An alias for this command is \R.

\sinister (*S*)
The sinister descriptor. An alias for this command is \S.

Both these commands and the entgegen/zusammen descriptors get a small additional amount of kerning after the closing parenthesis. This amount can be changed through the following option:

nomenclature » cip-kern = {⟨*dim*⟩} Default: .075em
Set the amount of kerning after the closing parenthesis.

The entries typeset by and implemented with \cip can be customized further:

cip-outer-format = {⟨*format*⟩} Default: \upshape
Introduced in version 5.8 (2017/04/17)
The format of parentheses and commas typeset by \cip.

cip-inner-format = {⟨*format*⟩} Default: \itshape
Introduced in version 5.8
The format of the entries in \cip. This format works additive to the outer format.

cip-number-format = {⟨*format*⟩} Default: \itshape
Introduced in version 5.8
The format of numbers in \cip. This format works additive to the outer format and is applied to arabic figures only.

## 6.7. Fischer

\dexter D
The dexter descriptor. An alias for this command is \D.

\laevus L
The laevus descriptor. An alias for this command is \L.

## 6.8. cis/trans, zusammen/entgegen, syn/anti & tert

- \cis *cis* \trans *trans*
- \fac *fac* \mer *mer*
- \sin *sin* \ter *ter*
- \zusammen (*Z*) \entgegen (*E*)
- \syn *syn* \anti *anti*
- \tert *tert*

An alias for \entgegen is \E and an alias for \zusammen is \Z.

## 6.9. ortho/meta/para

\ortho *o* \meta *m* \para *p*

Although these commands are provided I like to cite *The IUPAC Blue Book* [PPR04]:

> The letters *o*, *m*, and *p* have been used in place of *ortho*, *meta*, and *para*, respectively, to designate the 1,2-, 1,3-, and 1,4- isomers of disubstituted benzene. This usage is strongly discouraged and is not used in preferred IUPAC names. *[PPR04, p. 90]*

## 6.10. Absolute Configuration

\Rconf[⟨*letter*⟩]

\Rconf:  \Rconf[]: 

\Sconf[⟨*letter*⟩]

\Sconf:  \Sconf[]: 

## 6.11. Coordination Chemistry

CHEMMACROS provides a few commands useful in coordination chemistry:

\bridge{⟨*num*⟩} $\mu_3$-
Denote bridging ligand connection.

\hapto{⟨*num*⟩} $\eta^5$-
Denote hapticity.

\dento{⟨*num*⟩} $\kappa^2$-
Denote denticity.

```
Ferrocene = \iupac{bis(\hapto{5}cyclo|penta|dienyl)iron} \par
\iupac{tetra-\bridge{3}iodido-tetrakis[tri|methyl|platinum(IV)]}
```

Ferrocene = bis($\eta^5$-cyclopentadienyl)iron
tetra-$\mu_3$-iodido-tetrakis[trimethylplatinum(IV)]

Two options allow customization:

nomenclature » bridge-number = sub|super Default: sub

Appends the number as a subscript or superscript, depending on the choice. The IUPAC recommendation is the subscript [Con+05].

nomenclature » coord-use-hyphen = true|false Default: true

Append a hyphen to \hapto, \dento and \bridge or don't.

Introduced in version 5.8

The default behaviour of \hapto and \dento has changed with version 5.8 to follow IUPAC recommendations.

## 6.12. Examples

```
\iupac{\dexter-Wein|s\"aure} =
\iupac{\cip{2S,3S}-Wein|s\"aure} \par
\iupac{\dexter-($-$)-Threose} =
\iupac{\cip{2S,3R}-($-$)-2,3,4-Tri|hydroxy|butanal} \par
\iupac{\cis-2-Butene} =
\iupac{\zusammen-2-Butene}, \par
\iupac{\cip{2E,4Z}-Hexa|diene} \par
\iupac{\meta-Xylol} =
\iupac{1,3-Di|methyl|benzene}
```

---

D-Weinsäure = (*2S,3S*)-Weinsäure
D-(−)-Threose = (*2S,3R*)-(−)-2,3,4-Trihydroxybutanal
*cis*-2-Butene = (*Z*)-2-Butene,
(*2E,4Z*)-Hexadiene
*m*-Xylol = 1,3-Dimethylbenzene

## 6.13. Own \iupac Macros And Shorthands

If you find any commands missing you can define them using

\NewChemIUPAC{⟨*cs*⟩}{⟨*declaration*⟩}
Define a new IUPAC command that is in any case defined inside of \iupac regardless if ⟨*cs*⟩ is defined elsewhere already.

\ProvideChemIUPAC{⟨*cs*⟩}{⟨*declaration*⟩}
Define a new IUPAC command that is in any case defined inside of \iupac regardless if ⟨*cs*⟩ is defined elsewhere already only if the corresponding IUPAC macro is not defined, yet.

\RenewChemIUPAC{⟨*cs*⟩}{⟨*declaration*⟩}
Redefine an existing IUPAC command that is in any case defined inside of \iupac regardless if ⟨*cs*⟩ is defined elsewhere already.

\DeclareChemIUPAC{⟨*cs*⟩}{⟨*declaration*⟩}
Define a new IUPAC command that is in any case defined inside of \iupac regardless if ⟨*cs*⟩ is defined elsewhere already. This silently overwrites an existing IUPAC macro definition.

\LetChemIUPAC{⟨*cs1*⟩}{⟨*cs2*⟩}
Defines ⟨*cs1*⟩ to be an alias of ⟨*cs2*⟩.

A command defined in this way will obey the setting of the option iupac. This means any existing command is only overwritten with iupac = {strict}. However, \NewChemIUPAC will *not* change the definition of an existing IUPAC naming command but issue an error if the IUPAC naming command already exists. \DeclareChemIUPAC *will* overwrite an existing IUPAC command.

```
\NewChemIUPAC\endo{\textsc{endo}}
\RenewChemIUPAC\anti{\textsc{anti}}
\iupac{(2-\endo,7-\anti)-2-bromo-7-fluoro|bicyclo[2.2.1]heptane}
```

(2-ENDO,7-ANTI)-2-bromo-7-fluorobicyclo[2.2.1]heptane

\RenewChemIUPAC allows you to redefine the existing IUPAC naming commands.

```
\iupac{\meta-Xylol} \par
\RenewChemIUPAC\meta{\textup{m}}
\iupac{\meta-Xylol}
```

*m*-Xylol
m-Xylol

There's also a way for defining new IUPAC shorthands or changing the existing ones:

\NewChemIUPACShorthand⟨*shorthand token*⟩⟨*control sequence*⟩
Defines a new IUPAC shorthand. Inside \iupac it will be equal to using ⟨*control sequence*⟩. This throws an error if ⟨*shorthand token*⟩ is already defined.

\RenewChemIUPACShorthand⟨*shorthand token*⟩⟨*control sequence*⟩
Redefines an existing IUPAC shorthand. This throws an error if ⟨*shorthand token*⟩ is not defined, yet.

\DeclareChemIUPACShorthand⟨*shorthand token*⟩⟨*control sequence*⟩
Defines a new IUPAC shorthand or redefines an existing one.

\ProvideChemIUPACShorthand⟨*shorthand token*⟩⟨*control sequence*⟩
Provides a new IUPAC shorthand. Does nothing if ⟨*shorthand token*⟩ is already defined.

\RemoveChemIUPACShorthand⟨*shorthand token*⟩
Deletes an existing IUPAC shorthand.

## 6.14. Latin Phrases

CHEMMACROS provides a command for typesetting latin phrases:

\latin[⟨*options*⟩]{⟨*phrase*⟩}
Typesets ⟨*phrase*⟩ according to the option format described below.

\insitu *in situ*

\invacuo *in vacuo*

\abinitio *ab initio*

If you additionally load chemstyle [Wri13] said package will *not* define its own \latin. The last three commands mentioned above are defined through

\NewChemLatin{⟨*cs*⟩}{⟨*phrase*⟩}
Define a new latin phrase. Gives an error if ⟨*cs*⟩ already exists.

\DeclareChemLatin{⟨*cs*⟩}{⟨*phrase*⟩}
Define a new latin phrase. Silently redefined existing macros.

\RenewChemLatin{⟨*cs*⟩}{⟨*phrase*⟩}
Redefine an existing latin phrase. Gives an error if ⟨*cs*⟩ doesn't exist.

\ProvideChemLatin{⟨*cs*⟩}{⟨*phrase*⟩}
Define a new latin phrase only if ⟨*cs*⟩ doesn't exist.

```
\NewChemLatin\ltn{latin text}\ltn
```
*latin text*

You can change the appearance with this option:

nomenclature » format = {⟨*definition*⟩} Default: \emph
Changed in version 5.7 (2016/06/07)
Sets the format for the latin phrases.

# 7. The particles Module

The particles module loads the modules charges and chemformula.

## 7.1. Provided Particle Macros

The particles defines a number of macros which can be used for typesetting common particles in the running text. Most of them don't make much sense in chemformula [Nie19]'s \ch, though, which doesn't mean that they can't be used there, of course:

\el $e^-$ \prt $p^+$ \ntr $n^0$ \Hyd $OH^-$ \Oxo $H_3O^+$ \water $H_2O$ \El $E^+$ \Nuc $Nu^-$ \ba $ba^-$

All of these macros are defined using chemformula's \chcpd. The details are explained in section 7.2 on the next page.

The macros \Nuc and \ba are special: they have an optional argument for the following options:

particles » elpair = dots|dash|false Default: false
Determine how the electron pair of the nucleophiles is displayed. The electron pair is drawn using CHEMFORMULA's \chlewis macro.

particles » space = {⟨*dim*⟩} Default: .1em
Introduced in version 5.3
Sets the space that is inserted between the electron pair and the negative charge sign.

Both options can of course also be set with \chemsetup.

```
\ba[elpair=dots] \Nuc[elpair=dash]
                                        ba:⁻ Nu|⁻
\chemsetup[particles]{elpair=false}     ba⁻ Nu⁻
\ba\ \Nuc
```

## 7.2. Defining Own Particle Macros

There are two sets of macros, one for defining particles and one for defining nucleophiles.

\NewChemParticle{⟨*cs*⟩}{⟨*formula*⟩}
Defines a new macro ⟨*cs*⟩. ⟨*formula*⟩ is any valid CHEMFORMULA input (this depends on the setting of the formula option, see 10 starting on page 22). Raises an error if ⟨*cs*⟩ already exists.

\RenewChemParticle{⟨*cs*⟩}{⟨*formula*⟩}
Redefines a new macro ⟨*cs*⟩. ⟨*formula*⟩ is any valid CHEMFORMULA input (this depends on the setting of the formula option, see 10 starting on page 22). Raises an error if ⟨*cs*⟩ doesn't exist.

\DeclareChemParticle{⟨*cs*⟩}{⟨*formula*⟩}
Defines a macro ⟨*cs*⟩. ⟨*formula*⟩ is any valid CHEMFORMULA input (this depends on the setting of the formula option, see 10 starting on page 22). Silently overwrites ⟨*cs*⟩ if it exists.

\ProvideChemParticle{⟨*cs*⟩}{⟨*formula*⟩}
Defines a new macro ⟨*cs*⟩. ⟨*formula*⟩ is any valid CHEMFORMULA input (this depends on the setting of the formula option, see 10 starting on page 22). Does nothing if ⟨*cs*⟩ already exists.

An example of usage is the definition of the existing particle macros:

```
\NewChemParticle\el {e-}
\NewChemParticle\prt{p+}
\NewChemParticle\ntr{n^0}
```

The following set defines macros like \Nuc

\NewChemNucleophile{⟨*cs*⟩}{⟨*formula*⟩}
Defines a new macro ⟨*cs*⟩. ⟨*formula*⟩ is any valid CHEMFORMULA input (this depends on the setting of the formula option, see 10 starting on page 22). Note that ⟨*formula*⟩ will get a trailing negative charge! Raises an error if ⟨*cs*⟩ already exists.

\RenewChemNucleophile{⟨*cs*⟩}{⟨*formula*⟩}
Redefines a new macro ⟨*cs*⟩. ⟨*formula*⟩ is any valid CHEMFORMULA (this depends on the setting of the formula option, see 10 starting on page 22). Note that ⟨*formula*⟩ will get a trailing negative charge! Raises an error if ⟨*cs*⟩ doesn't exist.

\DeclareChemNucleophile{⟨*cs*⟩}{⟨*formula*⟩}
Defines a macro ⟨*cs*⟩. ⟨*formula*⟩ is any valid CHEMFORMULA (this depends on the setting of the formula option, see 10 starting on page 22). Note that ⟨*formula*⟩ will get a trailing negative charge! Silently overwrites ⟨*cs*⟩ if it exists.

\ProvideChemNucleophile{⟨*cs*⟩}{⟨*formula*⟩}
Defines a new macro ⟨*cs*⟩. ⟨*formula*⟩ is any valid CHEMFORMULA (this depends on the setting of the formula option, see 10 starting on page 22). Note that ⟨*formula*⟩ will get a trailing negative charge! Does nothing if ⟨*cs*⟩ already exists.

An example of usage is the definition of the existing nucleophile macros:

```
\NewChemNucleophile\Nuc{Nu}
\NewChemNucleophile\ba {ba}
```

A macro defined this way will have an optional argument for the elpair option.

# 8. The phases Module

The phases module loads the chemformula modul.

## 8.1. Basics

These commands are intended to indicate the phase of a compound.

\sld (s)  \lqd (l)  \gas (g)  \aq (aq)

```
\ch{C\sld{} + 2 H2O\lqd{} -> CO2\gas{} + 2 H2\gas}\par
To make it complete: NaCl\aq.
```

$\mathrm{C(s) + 2\,H_2O(l) \longrightarrow CO_2(g) + 2\,H_2(g)}$
To make it complete: $\mathrm{NaCl(aq)}$.

The IUPAC recommendation to indicate the state of aggregation is to put it in parentheses after the compound [Coh+08]. However, you might want to put it as a subscript which is also very common.

> The [...] symbols are used to represent the states of aggregation of chemical species. The letters are appended to the formula in parentheses and should be printed in Roman (upright) type without a full stop (period). The IUPAC Green Book *[Coh+08, p. 54]*

There are two options to customize the output:

`phases` » `pos` = `side|sub` Default: `side`
Switch the position of the phase indicator.

`phases` » `space` = {⟨*dim*⟩} Default: `.1333em`
Change the default spacing between compound a phase indicator if `pos = {side}`. A TEX dimension.

```
\chemsetup[phases]{pos=sub}
\ch{C\sld{} + 2 H2O\lqd{} -> CO2\gas{} + 2 H2\gas}\par
To make it complete: NaCl\aq.
```

$\mathrm{C_{(s)} + 2\,H_2O_{(l)} \longrightarrow CO_{2(g)} + 2\,H_{2(g)}}$
To make it complete: $\mathrm{NaCl_{(aq)}}$.

All those phase commands have an optional argument:

```
\ch{H2O "\lqd[\SI{5}{\celsius}]"}
```

$\mathrm{H_2O}$(l, 5 °C)

There is also a generic phase command:

\phase{⟨*phase*⟩}
If you need a phase indicator just once or twice. You can use it to denote a phase for which there is no phase command, yet.

## 8.2. Define Own Phases

Depending on the subject of your document you might need to indicate other states of aggregation. You can easily define them.

`\NewChemPhase{`⟨*cs*⟩`}{`⟨*symbol*⟩`}`
Define a new phase command. See section 8.3 for a way to define language dependent settings. Gives an error if ⟨*cs*⟩ already exists.

`\DeclareChemPhase{`⟨*cs*⟩`}{`⟨*symbol*⟩`}`
Define a new phase command. See section 8.3 for a way to define language dependent settings. Overwrites previous definitions of ⟨*cs*⟩.

`\RenewChemPhase{`⟨*cs*⟩`}{`⟨*symbol*⟩`}`
Redefine an existing phase command. See section 8.3 for a way to define language dependent settings. Gives an error if ⟨*cs*⟩ is not defined.

`\ProvideChemPhase{`⟨*cs*⟩`}{`⟨*symbol*⟩`}`
Define a new phase command. See section 8.3 for a way to define language dependent settings. Does nothing if ⟨*cs*⟩ is already defined.

```
% preamble:
\NewChemPhase\aqi{aq,$\infty$} % aqueous solution at infinite dilution
\NewChemPhase\cd {cd} % condensed phase
\NewChemPhase\lc {lc} % liquid crystal
\ch{NaOH\aqi} \ch{H2O\cd} \ch{U\phase{cr}} \ch{A\lc}\par
\chemsetup[phases]{pos=sub}
\ch{NaOH\aqi} \ch{H2O\cd} \ch{U\phase{cr}} \ch{A\lc}
```

NaOH(aq,∞) $H_2O$(cd) U(cr) A(lc)
$NaOH_{(aq,\infty)}$ $H_2O_{(cd)}$ $U_{(cr)}$ $A_{(lc)}$

## 8.3. Language Dependencies

For each phase command a translation into the custom language can be defined. If a phase is declared with `\NewChemPhase` no translation exists and for every babel language the literal string is used that was provided as a definition. Let's say you define the phase

```
\NewChemPhase\liquid{l}
```

and want to add the German translation "fl". Then you could do

```
\DeclareTranslation{German}{phase-liquid}{f\/l}
```

This way, when you use it in a German document using the appropriate babel option using `\liquid` would correctly translate. For this the package translations [Nie20b] is used. The ID always is `phase-`⟨*csname*⟩ where ⟨*csname*⟩ is the name of the phase command you defined without leading backslash.

See section 26 starting on page 61 for predefined translations and general language options of CHEMMACROS.

# 9. The symbols Module

The symbols module defines a few symbols chemists need now and then. It loads the package amstext [MSoo].

`\transitionstatesymbol`
This is self-explaining: ⇌

`\standardstate`
Again self-explaining: ⦵

`\changestate`
The uppercase delta used in $\Delta H$ for example.

# 10. The chemformula Module

The chemformula module loads the amstext package [MSoo] and the charges module.

## 10.1. For Users

There are different packages which provide means for typesetting chemical formulas:

- chemformula [Nie19]. This is probably well known to users of CHEMMACROS.
- mhchem [Hen18]. This is the "older brother" of CHEMFORMULA.
- chemfig [Tel19]. The easiest and most complete of the packages for drawing skeletal formulas.
- XƎTEX [Fuj13]. A very comprehensive alternative for typesetting chemistry.

In order to help authors getting a consistent layout CHEMMACROS does not make a choice which package to use for typesetting formulas. Although CHEMFORMULA is well tested and preferred users can choose other packages if they like.

this is done with the following general option:

`formula` = {⟨*method*⟩} Default: `chemformula`

Introduced in version 5.1 (2015/09/23)

This option let's you choose how chemical formulas are typeset. Available methods are

- chemformula
- mhchem

Introduced in version 5.6 (2016/05/02)

- chemist (from the XƎTEX bundle)

Introduced in version 5.6

- chemfig

The corresponding package with the same name is loaded.

If you explicitly set this option the corresponding package is loaded immediately and the method is set up. Otherwise the option will be set by CHEMMACROS at the end of the preamble.

Introduced in version 5.2 (2015/10/14)

If you load a method package in a way that a unique choice is possible then CHEMMACROS will set the method accordingly if you haven't set the option by yourself. If *no* unique choice is possible CHEMMACROS will raise a warning and choose chemformula regardless if the package is loaded or not. In this case if you want to use another method you'll have to choose manually. *All automatic choices only happen at the end of the preamble.*

## 10.2. Using the chemformula Package

If you set formula = {chemformula} the chemformula module makes it possible that you can set all CHEMFORMULA options via the \chemsetup command using the module chemformula, for example:

```
\chemsetup[chemformula]{format=\sffamily}
```

Everywhere where CHEMMACROS typesets chemical formulas CHEMFORMULA's macros \chcpd or \ch are used, for example in the reaction environments provided by the reactions module.

*This method is the recommended choice!*

## 10.3. Using the mhchem Package

Introduced in version 5.1

If you set formula = {mhchem} the chemformula module makes it possible that you can set all of mhchem's options via the \chemsetup command using the module mhchem, for example:

```
\chemsetup[mhchem]{format=\sffamily}
```

Everywhere where CHEMMACROS typesets chemical formulas mhchem's macro \ce is used, for example in the reaction environments provided by the reactions module.

There are some *caveats* if you use this method:

- This method has not been extensively tested, yet. There may be errors and wrong output at unexpected places.
- Using this method effectively disables the different values of the particles option elpair (see section 7).
- The different kinds of formal charges provided by the charges module (see section 5.2) are disabled. Formal charges always use the math method now.
- There may also be other incompatibilities (*e. g.*, mhchem has it's own method of setting upright Greek letters so it may or may not disable CHEMMACROS' mechanism).

## 10.4. Using the chemfig Package

Introduced in version 5.6

Everywhere where CHEMMACROS typesets chemical formulas chemfig's macro \printatom is used, for example in the reaction environments provided by the reactions module.

There are some *caveats* if you use this method:

- This method has not been extensively tested, yet. There may be errors and wrong output at unexpected places.
- Using this method effectively disables the different values of the particles option elpair (see section 7).
- The different kinds of formal charges provided by the charges module (see section 5.2) are disabled. Formal charges always use the math method now.
- The reaction environments by the reactions module may work only to a limited respect. If you plan to use them consider using methods chemformula or mhchem instead.

## 10.5. Using the chemist Package

Introduced in version 5.6

Everywhere where CHEMMACROS typesets chemical formulas chemist's macro `\ChemForm` is used, for example in the reaction environments provided by the `reactions` module.

There are some *caveats* if you use this method:

- This method has not been extensively tested, yet. There may be errors and wrong output at unexpected places.
- Using this method effectively disables the different values of the `particles` option `elpair` (see section 7).
- The different kinds of formal charges provided by the `charges` module (see section 5.2) are disabled. Formal charges always use the math method now.
- The reaction environments by the `reactions` module may work only to a limited respect. If you plan to use them consider using methods chemformula or mhchem instead.[6]

## 10.6. For Module Writers

There are two macros for module writers:

`\chemmacros_chemformula:n` {⟨*formula*⟩}
This is only a wrapper for `\chcpd` or `\ce`. It is recommended that module writers use this macro (or a variant thereof) inside of CHEMMACROS' macros whenever they want to display a chemical formula. Writers who prefer traditional LaTeX 2ε programming over expl3 should use `\chemmacros@formula`.

`\chemmacros_reaction:n` {⟨*reaction*⟩}
This is only a wrapper for `\ch` or `\ce`. It is recommended that module writers use this macro (or a variant thereof) inside of CHEMMACROS' macros whenever they want to display a chemical reaction. Writers who prefer traditional LaTeX 2ε programming over expl3 should use `\chemmacros@reaction`.

# 11. The greek Module

The `greek` module loads the chemgreek package [Nie16a].

This module provides one option:

`greek` = {⟨*mapping*⟩}
A valid value is any valid CHEMGREEK ⟨*mapping*⟩. CHEMMACROS will warn you if no mapping has been chosen or if you are using the `default` or the `var-default` mapping because this means that no upright Greek letters are available.

If you load a CHEMGREEK support package which allows an unambiguous choice of a mapping CHEMGREEK will make this choice automatically. This means if you say

```
\usepackage{upgreek}
\usepackage{chemmacros}
```

then CHEMMACROS will use upgreek's upright Greek letters. If you have

6. On the other hand XeTeX (and especially the chemist package) provides quite a number of chemical reaction environments itself.

```
\usepackage{upgreek}
\usepackage{chemmacros}
\usepackage{textgreek}
```

then no unambiguous choice is possible and you should choose a mapping yourself, for example:

```
\usepackage{upgreek}
\usepackage{chemmacros}
\usepackage{textgreek}
\chemsetup{greek=textgreek}
```

For further details on mappings please refer to CHEMGREEK's manual.

# Part III.
# Additional Modules

The modules described in this part are not part of CHEMMACROS' minimal setup.

## 12. The isotopes Module

The isotope module loads the elements package [Nie15]. This module defines one user command:

`\isotope*{`⟨*input*⟩`}`

⟨*input*⟩ can either be the *symbol* of an element or the *name* of an element. Be aware that *the name is language dependent*, refer to the manual of the elements package for details. To be on the safe side use the element symbol.

⟨*input*⟩ can also be comma separated list: `\isotope{`⟨*nuc*⟩`,`⟨*symbol*⟩`}`. If you leave ⟨*nuc*⟩ out then `\isotope` will display the most common isotope. Otherwise ⟨*nuc*⟩ will be used. If ⟨*nuc*⟩ is an isotope unknown to the elements package `\isotope` will write a warning to the log file.

The starred variant omits the element number.

```
\isotope{C}
\isotope*{C}
\isotope{14,C}
\isotope*{14,C}
```

$^{12}_{6}\mathrm{C}$ $^{12}\mathrm{C}$ $^{14}_{6}\mathrm{C}$ $^{14}\mathrm{C}$

As input for the element symbol you can choose any of the elements known to the elements package.

There are options which allow you to determine how the isotope is printed:

`isotopes » format = super|side` Default: `super`

Either print the isotope number as superscript or to the right of the element symbol.

`isotopes` » `side-connect` = {⟨*input*⟩} Default: -

Determine what is printed between the element symbol and the isotope number if `format` = `{side}`.

```
\isotope{C}
\chemsetup[isotopes]{format=side}
\isotope{C}
\chemsetup[isotopes]{side-connect=}
\isotope{C}
```

$^{12}_{6}\mathrm{C}$ C-12 C12

## 13. The mechanisms Module

The module `mechanisms` loads the package amstext [MSoo]. It provides one macro:

`\mech[`⟨*type*⟩`]`
Allows to specify the most common reaction mechanisms.

⟨*type*⟩ can have one of the following values:

`\mech`
(empty, no opt. argument) nucleophilic substitution $S_N$

`\mech[1]`
unimolecular nucleophilic substitution $S_N1$

`\mech[2]`
bimolecular nucleophilic substitution $S_N2$

`\mech[se]`
electrophilic substitution $S_E$

`\mech[1e]`
unimolecular electrophilic substitution $S_E1$

`\mech[2e]`
bimolecular electrophilic substitution $S_E2$

`\mech[ar]`
electrophilic aromatic substitution Ar-$S_E$

`\mech[e]`
elimination E

`\mech[e1]`
unimolecular elimination E1

`\mech[e2]`
bimolecular elimination E2

`\mech[cb]`
unimolecular elimination “conjugated base”, *i. e.*, via carbanion $E1_{cb}$

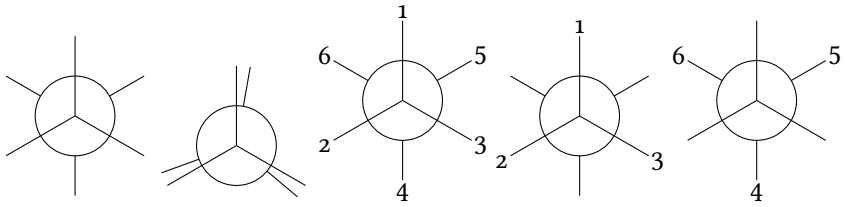
# 14. The newman Module

The newman module provides a command for drawing Newman projections. It loads the tikz module.

`\newman`[⟨*options*⟩](⟨*angle*⟩){⟨*1*⟩,⟨*2*⟩,⟨*3*⟩,⟨*4*⟩,⟨*5*⟩,⟨*6*⟩}

Create Newman projections. This command uses Ti*k*Z internally. ⟨*angle*⟩ rotates the back atoms counter clockwise with respect to the front atoms and is an optional argument. ⟨*1*⟩ to ⟨*6*⟩ are the positions, the first three are the front atoms, the last three the back atoms.

```
\newman{} \newman(170){}
\newman{1,2,3,4,5,6} \newman{1,2,3} \newman{,,,4,5,6}
```




Several options allow customization:

`newman » angle = {`⟨*angle*⟩`}` Default: `0`

Default angle.

`newman » scale = {`⟨*factor*⟩`}` Default: `1`

Scale the whole projection by factor ⟨*factor*⟩.

`newman » ring = {`⟨*tikz*⟩`}` (initially empty)

Customize the ring with Ti*k*Z keys.

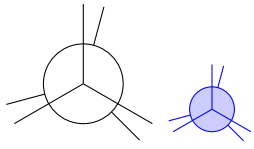
`newman » atoms = {`⟨*tikz*⟩`}` (initially empty)

Customize the nodes within which the atoms are set with Ti*k*Z keys.

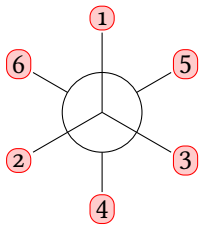
`newman » back-atoms = {`⟨*tikz*⟩`}` (initially empty)

Explicitly customize the nodes of the back atoms with Ti*k*Z keys.

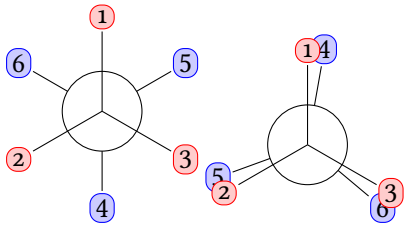
```
\chemsetup[newman]{angle=45} \newman{}
\newman[scale=.75,ring={draw=blue,fill=blue!20}]{}
```



```
\chemsetup[newman]{atoms={draw=red,fill=red!20,inner sep=2pt,rounded corners
  }}
\newman{1,2,3,4,5,6}
```




```
\chemsetup[newman]{
  atoms = {draw=red,fill=red!20,inner sep=2pt,rounded corners},
  back-atoms = {draw=blue,fill=blue!20,inner sep=2pt,rounded corners}
}
\newman{1,2,3,4,5,6} \newman(170){1,2,3,4,5,6}
```




## 15. The orbital Module

The orbital module loads the tikz module. It provides the following command to create orbitals:

`\orbital[`⟨*options*⟩`]{`⟨*type*⟩`}`
Draw an orbital shape of type ⟨*type*⟩. This command uses TikZ internally.

There are the following types available for ⟨*type*⟩:

`s` `p` `sp` `sp2` `sp3`

```
\orbital{s} \orbital{p} \orbital{sp} \orbital{sp2} \orbital{sp3}
```

Depending on the type you have different options to modify the orbitals:

`orbital` » `phase` = `+|-` Default: `+`
changes the phase of the orbital (all types)

`orbital` » `scale` = {⟨*factor*⟩} Default: `1`
changes the size of the orbital (all types)

`orbital` » `color` = {⟨*color*⟩} Default: `black`
changes the color of the orbital (all types)

orbital » angle = {⟨*angle*⟩} Default: 0

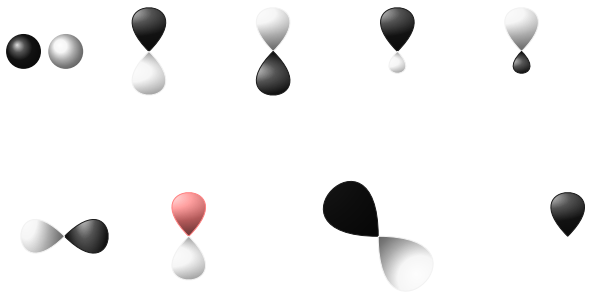
rotates the orbitals with a p contribution counter clockwise (all types except s)

orbital » half = true|false Default: false

displays only half an orbital (only p)

```
\orbital{s} \orbital[phase=-]{s}
\orbital{p} \orbital[phase=-]{p}
\orbital{sp3} \orbital[phase=-]{sp3}

\orbital[angle=0]{p} \orbital[color=red!50]{p}
\orbital[angle=135,scale=1.5]{p} \orbital[half]{p}
```



Additionally there are two options, with which the Ti*k*Z behaviour can be changed.

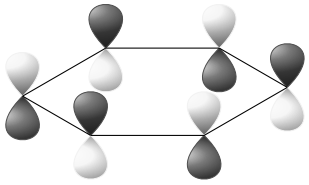
orbital » overlay = true|false

The orbital "doesn't need space"; it is displayed with the Ti*k*Z option overlay.

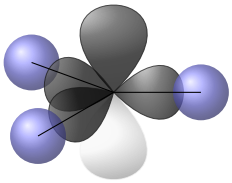
orbital » opacity = {⟨*num*⟩}

The orbital becomes transparent; ⟨*value*⟩ can have values between 1 (fully opaque) to 0 (invisible).

```
\vspace{7mm}
\chemsetup[orbital]{
  overlay,
  p/color = black!70
}
\setchemfig{bond offset=0pt}
\chemfig{
  ?\orbital{p}
  -[,1.3]{\orbital[phase=-]{p}}
  -[:30,1.1]\orbital{p}
  -[:150,.9]{\orbital[phase=-]{p}}
  -[4,1.3]\orbital{p}
  -[:-150,1.1]{\orbital[phase=-]{p}}?
}
\vspace{7mm}
```



```
\vspace{7mm}
\setchemfig{bond offset = 0pt}
\chemsetup[orbital]{
  overlay ,
  opacity = .75 ,
  p/scale = 1.6 ,
  s/color = blue!50 ,
  s/scale = 1.6
}
\chemfig{
  \orbital{s}
  -[:-20]{\orbital[scale=2]{p}}
         {\orbital[half,angle=0]{p}}
         {\orbital[angle=170,half]{p}}
         {\orbital[angle=-150,half]{p}}
  (-[:-150]\orbital{s})-\orbital{s}
}
\vspace{1cm}
```



# 16. The polymers Module

Introduced in version 5.5 (2016/03/08)

The polymers module loads the nomenclature and the tikz modules.

## 16.1. Nomenclature

The polymers module defines a number of IUPAC macros for usage inside \iupac which are used in polymer chemistry.

## 16.2. Copolymers

\copolymer *co*
unspecified copolymer. An alias for this command is \co.

\statistical *stat*
statistical copolymer. An alias for this command is \stat.

\random *ran*
random copolymer. An alias for this command is \ran.

\alternating *alt*
alternating copolymer. An alias for this command is \alt.

\periodic *per*
periodic copolymer. An alias for this command is \per.

\block *block*
block copolymer.

\graft *graft*
graft copolymer.

## 16.3. Non-linear (Co) Polymers and Polymer Assemblies

\blend *blend*
The blend qualifier.

\comb *comb*
The comb qualifier.

\complex *compl*
The complex qualifier. An alias for this command is \compl.

\cyclic *cyclo*
The cyclic qualifier. An alias for this command is \cyclo.

\branch *branch*
The branch qualifier.

\network *net*
The network qualifier. An alias for this command is \net.

\ipnetwork *ipn*
The interpenetrating network qualifier. An alias for this command is \ipn.

\sipnetwork *sipn*
The semi-interpenetrating network qualifier. An alias for this command is \sipn.

\star *star*
The star qualifier.

## 16.4. Polymer Denotations in chemfig's Molecules

The chemfig manual proposes some code defining the macros \setpolymerdelim and \makebraces which make it possible to add delimiters to chemfig molecules. The polymers module implements the following macro based on the same idea:

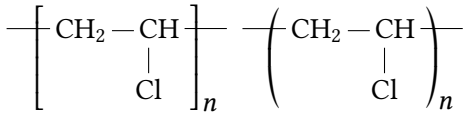
\makepolymerdelims[⟨*options*⟩]{⟨*height*⟩}[⟨*depth*⟩]{⟨*opening node*⟩}{⟨*closing node*⟩}
The value of ⟨*depth*⟩ is the same as ⟨*height*⟩ unless it is specified explicitly. ⟨*opening node*⟩ and ⟨*closing node*⟩ are the names of Ti*k*Z' nodes where the delimites are placed.

polymers » delimiters = {⟨*left*⟩⟨*right*⟩} Default: []
This option demands two tokens as argument, the first being the opening brace, the second the closing brace. A dot (.) denotes an empty delimiter.

polymers » subscript = {⟨*subscript*⟩} Default: $n$
Subscript to the right delimiter.

polymers » superscript = {⟨*superscript*⟩}
Superscript to the right delimiter.

```
\setchemfig{atom sep=2em}
\chemfig{-[@{op,.75}]CH_2-CH(-[6]Cl)-[@{cl,0.25}]}
\makepolymerdelims{5pt}[27pt]{op}{cl}
\chemfig{-[@{op,.75}]CH_2-CH(-[6]Cl)-[@{cl,0.25}]}
\makepolymerdelims[delimiters=()]{5pt}[27pt]{op}{cl}
```



# 17. The reactions Module

The reactions module loads the chemformula module and the mathtools package [MRW19].

## 17.1. Predefined Environments

You can use these environments for numbered…

\begin{reaction}
A single reaction where CHEMFORMULA code is placed directly in the environment body. A wrapper around the equation environment. The environment body is parsed with \ch or \ce depending on the value of the formula option, see section 10 starting on page 22.

\begin{reactions}
Several aligned reactions. A wrapper around amsmath's align environment. The environment body is parsed with \ch or \ce depending on the value of the formula option, see section 10 starting on page 22.

…and their starred versions for unnumbered reactions.

\begin{reaction*}
A wrapper around the equation* environment. The environment body is parsed with \ch or \ce depending on the value of the formula option, see section 10 starting on page 22.

\begin{reactions*}
A wrapper around amsmath's align* environment. The environment body is parsed with \ch or \ce depending on the value of the formula option, see section 10 starting on page 22.

With those environments you can create (un)numbered reaction equations similar to mathematical equations.

Theses environments use the equation/equation* environments or the align/align* environments, respectively, to display the reactions.

```
Reaction with counter:
\begin{reaction}
  A -> B
\end{reaction}
```

Reaction with counter:

$$\mathrm{A} \longrightarrow \mathrm{B} \qquad \{1\}$$

```
Reaction without counter:
\begin{reaction*}
  C -> D
\end{reaction*}
```

Reaction without counter:

$$\mathrm{C} \longrightarrow \mathrm{D}$$

```
Several aligned reactions with counter:
\begin{reactions}
  A     &-> B + C \\
  D + E &-> F
\end{reactions}
```

Several aligned reactions with counter:

$$\begin{aligned} \mathrm{A} &\longrightarrow \mathrm{B} + \mathrm{C} && \{2\} \\ \mathrm{D} + \mathrm{E} &\longrightarrow \mathrm{F} && \{3\} \end{aligned}$$

```
Several aligned reactions without counter:
\begin{reactions*}
  G     &-> H + I \\
  J + K &-> L
\end{reactions*}
```

Several aligned reactions without counter:

$$\begin{aligned} \mathrm{G} &\longrightarrow \mathrm{H} + \mathrm{I} \\ \mathrm{J} + \mathrm{K} &\longrightarrow \mathrm{L} \end{aligned}$$

If you want to change the layout of the counter tags, you can use

`\renewtagform{`⟨*tagname*⟩`}[`⟨*format*⟩`]{`⟨*left delimiter*⟩`}{`⟨*right delimiter*⟩`}`

Provided by the mathtools package.

or use the following options:

`reactions` » `tag-open` = {⟨*left delimiter*⟩} Default: {

Introduced in version 5.6

The left delimiter.

reactions » tag-close = {⟨*right delimiter*⟩} Default: }

Introduced in version 5.6

The right delimiter.

reactions » before-tag = {⟨*format*⟩} (initially empty)

Introduced in version 5.6

Code inserted before the tags.

```
\chemsetup[reactions]{
  before-tag = R \textbf ,
  tag-open = [ ,
  tag-close = ]
}
\begin{reaction}
  H2O + CO2 <<=> H2CO3
\end{reaction}
```

$$H_2O + CO_2 \rightleftharpoons H_2CO_3 \qquad [R\,\mathbf{4}]$$

The use of $\mathcal{AMS}$math's \intertext is possible:

```
\begin{reactions}
  A + 2 B &-> 3 C + D "\label{rxn:test}"
  \intertext{Some text in between aligned reactions}
  3 E + F &<=> G + 1/2 H
\end{reactions}
See reaction~\ref{rxn:test}.
```

$$A + 2\,B \longrightarrow 3\,C + D \qquad \{5\}$$

Some text in between aligned reactions

$$3\,E + F \rightleftharpoons G + \frac{1}{2}\,H \qquad \{6\}$$

See reaction 5.

Introduced in version 5.6

If you are using either cleveref or fancyref the reaction counter is supported already. For fancyref use the prefix rct.

## 17.2. Own Reactions

You can create new types of reactions with the command:

\NewChemReaction{⟨*name*⟩}[⟨*number of arguments*⟩]{⟨*math name*⟩}

⟨*name*⟩ will be the name of the new chem environment. ⟨*math name*⟩ is the underlying math environment. Gives an error if ⟨*name*⟩ already exists.

\RenewChemReaction{⟨*name*⟩}[⟨*number of arguments*⟩]{⟨*math name*⟩}

⟨*name*⟩ is the name of the renewed chem environment. ⟨*math name*⟩ is the underlying math environment. Gives an error if ⟨*name*⟩ does not exist.

\DeclareChemReaction{⟨*name*⟩}[⟨*number of arguments*⟩]{⟨*math name*⟩}
⟨*name*⟩ will be the name of the chem environment. ⟨*math name*⟩ is the underlying math environment.

\ProvideChemReaction{⟨*name*⟩}[⟨*number of arguments*⟩]{⟨*math name*⟩}
⟨*name*⟩ will be the name of the new chem environment. ⟨*math name*⟩ is the underlying math environment. The new environment is only defined if it doesn't exist, yet.

```
\NewChemReaction{reaction}  {equation}
\NewChemReaction{reaction*} {equation*}
\NewChemReaction{reactions} {align}
\NewChemReaction{reactions*}{align*}
```

Let's suppose, you'd like to have the alignment behaviour of the alignat environment for CHEMFORMULA reactions. You could do the following:

```
\NewChemReaction{reactionsat}[1]{alignat}
```

With this the reactionsat environment is defined.

```
\NewChemReaction{reactionsat}[1]{alignat}
\NewChemReaction{reactionsat*}[1]{alignat*}
\begin{reactionsat}{3}
  A     &-> B     &&-> C     &&-> D \\
  aaaaa &-> bbbbb &&-> ccccc &&-> ddddd
\end{reactionsat}
\begin{reactionsat*}{2}
  A     &-> B     & C            &-> D  \\
  aaaaa &-> bbbbb &\quad{} ccccc &-> ddddd
\end{reactionsat*}
```

$$\begin{aligned} A &\longrightarrow B &&\longrightarrow C &&\longrightarrow D && \{7\} \\ aaaaa &\longrightarrow bbbbb &&\longrightarrow ccccc &&\longrightarrow ddddd && \{8\} \end{aligned}$$

$$\begin{aligned} A &\longrightarrow B & C &\longrightarrow D \\ aaaaa &\longrightarrow bbbbb & ccccc &\longrightarrow ddddd \end{aligned}$$

## 17.3. List of Reactions

The reactions module also provides a command to display a list of the reactions created with the reaction environment.

\listofreactions
Print a list of reactions.

```
\listofreactions
```

**List of Reactions**

The output of this list can be modified by two options:

`reactions` » `list-name` = {⟨*name of the list*⟩} Default: `\ChemTranslate{list-of-reactions}`

Let's you set the name of the list manually. The default name is language dependent, see section 26 starting on page 61.

`reactions` » `list-entry` = {⟨*prefix to each entry*⟩} Default: `\ChemTranslate{reaction}`

Let's you set a prefix to each list entry. The default name is language dependent, see section 26 starting on page 61.

`reactions` » `list-heading-cmd` = {⟨*code*⟩} Default: `\section*{#1}`

Introduced in version 5.2

The macro that is called at the beginning of the list. Inside of ⟨*code*⟩ `#1` refers to the actual heading of the list. The default setting is not entirely true: if a macro `\chapter` is defined `\chapter*{#1}` is used.

`reactions` » `tocbasic` = `true`|`false` Default: `false`

Introduced in version 5.6

If you use a KOMA-Script class *or* if you load the tocbasic package *or* if you set this option to `true` the list of reactions will be set up using the tocbasic package. This *disables* the `list-heading-cmd` option. For a KOMA-Script class this means that the list of reactions obeys KOMA-Script's `listof` option.

Instead of using the option `list-name` you also could redefine `\reactionlistname`.

The list lists all reactions with a number and disregards reactions without number. All reaction environments without star have an optional argument which let's you add a description (or caption) for the entry in the list.

```
\begin{reaction}[Autoprotolyse]
  2 H2O <=> H3O+ + OH-
\end{reaction}
```

$$2\,H_2O \rightleftharpoons H_3O^+ + OH^- \qquad \{9\}$$

If you use the `reactions` environment this will not work, though. In this case you can use

`\AddRxnDesc{`⟨*description*⟩`}`

Add a description to a reaction.

```
\begin{reactions}
  "\chlewis{0.}{Cl}" + CH4  &
    -> HCl + "\chlewis{180.}{C}" H3 \AddRxnDesc{first~step~of~chain} \\
  "\chlewis{180.}{C}" H3 + Cl2 &
    -> CH3Cl + "\chlewis{0.}{Cl}" \AddRxnDesc{second~step~of~chain}
\end{reactions}
```

$$\mathrm{Cl^\bullet + CH_4 \longrightarrow HCl + {^\bullet CH_3}} \quad \{10\}$$
$$\mathrm{{^\bullet CH_3} + Cl_2 \longrightarrow CH_3Cl + Cl^\bullet} \quad \{11\}$$

# 18. The reactants Module

Introduced in version 6.0 (2021/02/13)

Idea for this module is by Sonja K., who also does the main development of the module. Many thanks for all her work!

## 18.1. Idea and Getting Started

The reactants module offers a simplified input syntax for chemical reactants in the description of reaction procedures. Reactant and solvent names are declared in the preamble removing the need to repeat the same IUPAC names multiple times throughout the document. With the help of module options the output style (order of name number and data) can be altered globally (or locally) to suit your needs, while the data itself is input using an easy to use key-value approach and processed by the siunitx package. The reactants module responds to the language declared with the babel package and also offers methods to integrate the acronyms of used reactants or solvents into the list of acronyms.

The module requires and loads the packages chemnum [Nie16b] and siunitx [Wri18]. Depending on the selected options the packages acro [Nie20a], glossaries-extra [Tal20], hyperref [ORT20], longtable [Car19] and/or xltabular [VN20] might be needed for this module. Unlike other needed packages, these have to be loaded by the user and will be explicitly mentioned in the corresponding sections of this manual.

## 18.2. Commands

\DeclareChemReactant
{⟨*ID*⟩}{⟨*properties*⟩} This command defines the reactant ⟨*ID*⟩ with the properties ⟨*properties*⟩.

\DeclareChemReactant
{⟨*main ID*⟩.⟨*sub ID*⟩}{⟨*properties*⟩} Analogously to chemnum's \cmpd the \DeclareChemReactant and \reactant commands accept a combined ⟨*ID*⟩ consisting of a ⟨*main ID*⟩ and ⟨*sub ID*⟩ part.

Valid properties are

name = {⟨*name*⟩}
Mandatory property: the name of the substance.

short = {⟨*abbreviation*⟩}
Optional property: a short form of the name, used when the reactants module is used in combination with the acronym-support option.

bookmark = {⟨*replacement in PORTABLE DOCUMENT FILE (PDF) bookmarks*⟩}
Optional property: replaces ⟨*name*⟩ in a PDF bookmark. This might be advisable when reactants are used in section titles and the hyperref package is used as well.

Common declarations will look like this:

```
\DeclareChemReactant{thf}{name={tetrahydrofuran}, short={THF}}
\DeclareChemReactant{H2SO4}{name={\ch{H2SO4}}}
\DeclareChemReactant{dichloropentane}{name={\iupac{2,4-di|chloro|pentane}}}
```

`\reactant`
`[`⟨*data and units*⟩`]{`⟨*ID*⟩`}` This command is used to insert a predefined reactant with the ⟨*ID*⟩ in the text. Depending on the settings and the `reactant-output-style` option, this command will display name and/or number of the reactant in combination with relevant data. Variants of this command with different suffixes, such as `*`, `+`, `l`, `s` or `plain` will be described later.

`\solvent`
`[`⟨*data and units*⟩`]{`⟨*ID*⟩`}` Analogous to `\reactants`. Can be used to insert solvent names and corresponding data in the text, depending on the `solvent-output-style` option. `s` and `l` suffixed variant exist and are discussed later.

⟨*data and units*⟩ accepts a comma separated list of key-value pairs. Valid keys acceptable values as well as their defaults are listed in table 3 on the next page.

Typical uses will look like this:

```
\reactant{dichloropentane}
\reactant[volume=5]{dichloropentane}
\reactant[volume=0.5, volume-unit=\L]{dichloropentane}
\solvent{thf}
\solvent[volume=200]{thf}
\solvent[volume=1.5, volume-unit=\L]{thf}
```

`\printreactants`
Prints a list of number and name of all reactants used throughout the document. The resulting list is sorted by number and also includes compounds numbered with chemnum's `\cmpd` command. The starred variant also includes the ⟨*ID*⟩ in the list of reactants. Using `printreactants-style` different styles can be selected.

## 18.3. Options

`reactants » initiate = true|false` Default: `false`
The chemnum package that is internally used for numbering the reactants offers two ways of initiating a new label: either when `\cmpd` is first used or through `\initcmpd`. The `reactants` module also offers these two methods with initiating a new label upon the first use of `reactant` being the default. If you prefer to initiate a new label through the `\DeclareChemReactant` command set this option to true.

Reactants are automatically numbered in the order of their first appearance, while `initiate` numbers the compounds in the order in which they were declared in the preamble or in an external document.

`reactants » switch = true|false` Default: `false`
While `\reactants` will output name and number of a reactant, its starred variant `\reactant*`, will by default result in the name without the corresponding number. Setting `switch = {true}`, globally or locally, reverses this behavior and outputs a reactant's number without its name.

Other options are described at later places when the corresponding behavior is described.

### 18.3.1. Data and Units

Describing synthetic procedures often requires adding a lot of data with the corresponding units to each reactant/solvent that is used. In order to allow for a uniform representation of numbers and units, as well as making the code more readable, the `\reactant` and `\solvent` command offers an optional argument that can be used to easily input this data: `\reactant[`⟨*data and units*⟩`]{`⟨*ID*⟩`}` `\solvent[`⟨*data and units*⟩`]{`⟨*ID*⟩`}`.

⟨*list of data*⟩ accepts a comma separated list of key-value pairs with the available keys and their default units/values listed in table 3. Key-value pairs can be input in any order as they are categorized and rearranged internally. Since numbers and their corresponding units are processed using siunitx, the usual `\sisetup` command can be used to alter for example the output decimal separator according to your needs. Be aware, though, that you must not use a comma as the the input decimal separator.

TABLE 3: Overview of available keys as well as the default units and the option to locally or globally change that default unit

| Key | default unit | option |
|---|---|---|
| mass | g | mass-unit |
| volume | mL | volume-unit |
| amount | mmol | amount-unit |
| concentration | M | concentration-unit |
| equiv | eq | equiv-unit |
| fraction | w/w% | fraction-unit |
| purity | % | purity-unit |
| solvent | n.a. | |
| solution-name | solution in | solution |

```
\DeclareChemReactant{nBuLi}{name={\iupac{\textit{n}=butyllithium}}}
\DeclareChemReactant{Br2benzene}{name={\iupac{1,4=di|bromo|benzene}}}
\DeclareChemReactant{HBr}{name={\ch{HBr\aq}}}

\reactant[volume=5.00, amount=12.5, equiv=1.00, concentration=2.5, solvent=
  hexane]{nBuLi}\par
\reactant[mass=3.9, amount=15.6, equiv=1.3, purity=95]{Br2benzene}\par
\reactant[volume=2.0, amount=43.8, equiv=3.5, fraction=65]{HBr}
```

*n*-butyllithium **1** (5.00 mL, 12.5 mmol, 1.00 eq, 2.5 M solution in hexane)
1,4-dibromobenzene **2** (3.9 g, 15.6 mmol, 1.3 eq, 95 %)
HBr (aq) **3** (2.0 mL, 65 w/w%, 43.8 mmol, 3.5 eq)

The options that change the units of the properties can be set with `\chemsetup` or in the optional argument of `\reactant`.

`reactants` » `mass-unit` = {⟨*unit*⟩} Default: `\gram`
Change the unit of the `mass` property.

`reactants` » `volume-unit` = {⟨*unit*⟩} Default: `\milli\liter`
Change the unit of the `volume` property.

reactants » amount-unit = {⟨unit⟩} Default: \milli\mole
Change the unit of the amount property.

reactants » concentration-unit = {⟨unit⟩} Default: \Molar
Change the unit of the concentration property.

reactants » equiv-unit = {⟨unit⟩} Default: eq
Change the unit of the equiv property.

reactants » fraction-unit = {⟨unit⟩} Default: w/w \percent
Change the unit of the fraction property.

reactants » purity-unit = {⟨unit⟩} Default: \percent
Change the unit of the purity property.

```
\reactant[volume=5.5]{thf} \par
\reactant[volume=5, volume-unit=\cubic\centi\metre]{thf}
```

tetrahydrofuran **4** (5.5 mL)
tetrahydrofuran **4** ($5\,\mathrm{cm}^3$)

solution here refers to the text that links concentration and solvent. This text automatically adapts to the document language. Currently, the English fallback, as well as the German translation are included in the package. If you write in a different language (or don't like the predefined text), you can use \DeclareChemTranslation{⟨key⟩}{⟨language⟩} {⟨translation⟩} (with {⟨key⟩} = solution) as described in section 26 starting on page 61.

### 18.3.2. Output Styles

The reactants module categorizes the data into different categories that are later used to determine the order in which this information is displayed. This behavior can be controlled using the predefined output styles:

reactants » reactant-output-style = name-main-other|main-name-other|main-other-name Default: main-other
Select one of the three predefined styles to determine the output style of the data and their units in the \reactant command.

reactants » solvent-output-style = main-name|name-main Default: main-name
Select one of the two predefined styles to determine the output style of the data and their units in the \solvent command.

name here refers to the combination of name and number (or if just one of them is available, to either name or number).

main here refers to the mass or volume of a reactant or solvent. If needed, equiv and/or amount can also be assigned to the main category.

other here refers to all the other data that is give to the reactant command.

The names of the reactant-output-style and olvent-output-style choice options refer to the order in which the contents of the categories are typeset.

```
\chemsetup[reactants]{reactant-output-style=name-main-other}
\reactant[volume=5, amount=4]{dichloropentane}\par
```

```
\chemsetup[reactants]{reactant-output-style=main-name-other}
\reactant[volume=5, amount=4]{dichloropentane}\par
\chemsetup[reactants]{reactant-output-style=main-other-name}
\reactant[volume=5, amount=4]{dichloropentane}

\chemsetup[reactants]{solvent-output-style=name-main}
\solvent[volume=5]{thf}\par
\chemsetup[reactants]{solvent-output-style=main-name}
\solvent[volume=5]{thf}
```

2,4-dichloropentane **5** (5 mL, 4 mmol)
5 mL 2,4-dichloropentane **5** (4 mmol)
5 mL (4 mmol) 2,4-dichloropentane **5**
tetrahydrofuran (5 mL)
5 mL tetrahydrofuran

`reactants` » `main` = `default|amount|equiv` Default: `default`

Be default, only `mass` and `volume` are assigned to the `main` category. Using the `main` option, `equiv` or `amount` can locally or globally assigned to the `main` category, as well.

```
\chemsetup[reactants]{main=equiv}
\reactant[equiv=2.0, amount=5]{dichloropentane}\par
\reactant[equiv=2.0, amount=5, main=amount]{dichloropentane}\par
\reactant[equiv=2.0, amount=5, main=default]{dichloropentane}
```

2,4-dichloropentane **5** (2.0 eq, 5 mmol)
2,4-dichloropentane **5** (5 mmol, 2.0 eq)
2,4-dichloropentane **5** (, 5 mmol, 2.0 eq)

`reactants` » `equivalents` = `true|false` Default: `true`

Can be used to prevent `equiv` from being output while still keeping the corresponding information in the input code. If you used the `main` = `equiv` option, the `equivalents` = `false` option will be ignored for the corresponding entries.

### 18.4. Use in Section Headings

### 18.5. Acronyms as Reactant/Solvent Names

`reactants` » `acronym-support` = `acro|glossaries|none` Default: `none`

### 18.6. List of Reactants

`reactants` » `printreactants-style` = `xltabular|longtable|none` Default: `none`

## 19. The redox Module

The `redox` module loads the modules `tikz` and `xfrac`. It also loads the packages mathtools [MRW19] and relsize [Ars13].

## 19.1. Oxidation Numbers

Regarding the typesetting of oxidation numbers *The IUPAC Green Book* [Coh+08] says the following:

> Oxidation numbers are denoted by positive or negative Roman numerals or by zero […]
>
> *Examples* $Mn^{VII}$, manganese (VII), $O^{-II}$, $Ni^{0}$ *[Coh+08, p. 50]*

The following command is provided to set oxidation numbers:

`\ox*[⟨options⟩]{⟨number⟩,⟨atom⟩}`
Places ⟨*number*⟩ as right superscript to ⟨*atom*⟩; ⟨*number*⟩ has to be a (rational) number! ⟨*atom*⟩ is treated as a CHEMFORMULA formula, like it would be in `\chcpd` (this depends on the setting of the `formula` option, see 10 starting on page 22).

```
1 \ox{+1,Na}, \ox{2,Ca}, \ox{-2,S}, \ox{-1,F}
```

$Na^{I}$, $Ca^{II}$, $S^{-II}$, $F^{-I}$

There are a number of options that can be used to modify the typeset result:

`redox » format = {⟨code⟩}`
Introduced in version 5.11 (2020/03/07)
Allows to apply arbitrary ⟨*code*⟩ in front of the typeset oxidation numbers. The last command may expect the oxidation number as an argument. An example might be `\textcolor{red}`.

`redox » parse = true|false` Default: `true`
When `false` an arbitrary entry can be used for `<number>`.

`redox » roman = true|false` Default: `false`
Switches from roman to arabic numbers.

`redox » pos = top|super|side` Default: `super`
`top` places ⟨*number*⟩ above ⟨*atom*⟩, `super` to the upper right as superscript and `side` to the right and inside brackets. Both `super` and `side` follow IUPAC recommendation, `top` does not!

`redox » explicit-sign = true|false` Default: `false`
Shows the + for positiv numbers and the ± for 0.

`redox » explizit-zero-sign = true|false` Default: `true`
Introduced in version 5.4
Only if both `explicit-sign` and `explicit-zero-sign` are set to `true` ±0 will be printed.

`redox » decimal-marker = comma|point` Default: `point`
Choice for the decimal marker for formal oxidation numbers like $X^{1.2}$.

`redox » align = center|right` Default: `center`
Center the oxidation number relative to the atom or right-align it.

`redox » side-connect = {⟨code⟩}` Default: `\,`
Code that is inserted between atom and oxidation number if `pos = {side}` is used.

`redox » text-frac = {⟨cs⟩}` Default: `\chemfrac[text]{#1}{#2}`
The fraction macro that is used for fractions if `pos = {side}` is used. ⟨*cs*⟩ must be a macro that takes two mandatory arguments, the first for the numerator and the second for the denominator.

redox » super-frac = {⟨*cs*⟩} Default: \chemfrac[superscript]{#1}{#2}

The fraction macro that is used for fractions if pos = {top} or pos = {super} is used. ⟨*cs*⟩ must be a macro that takes two mandatory arguments, the first for the numerator and the second for the denominator.

```
\ox[roman=false]{2,Ca} \ox{2,Ca} \\
\ox[pos=top]{3,Fe}-Oxide \\
\ox[pos=side]{3,Fe}-Oxide \\
\ox[parse=false]{?,Mn} \\
\ox[pos=top,align=right]{2,Ca}
```

$\mathrm{Ca}^{2}$ $\mathrm{Ca}^{\mathrm{II}}$
$\overset{\mathrm{III}}{\mathrm{Fe}}$-Oxide
Fe (III)-Oxide
$\mathrm{Mn}^{?}$
$\overset{\mathrm{II}}{\mathrm{Ca}}$

The pos = {top} variant also can be set with the shortcut \ox∗:

```
\ox{3,Fe} \ox*{3,Fe}
```

$\mathrm{Fe}^{\mathrm{III}}$ $\overset{\mathrm{III}}{\mathrm{Fe}}$

Using the explicit-sign option will always show the sign of the oxidation number:

```
\chemsetup[redox]{explicit-sign = true}
\ox{+1,Na}, \ox{2,Ca}, \ox{-2,S}, \ch{"\ox{0,F}" {}2}
```

---

$\mathrm{Na}^{+\mathrm{I}}$, $\mathrm{Ca}^{+\mathrm{II}}$, $\mathrm{S}^{-\mathrm{II}}$, $\mathrm{F}^{0}{}_{2}$

```
\chemsetup[redox]{pos=top}
Compare \ox{-1,O2^2-} to \ch{"\ox{-1,O}" {}2^2-}
```

---

Compare $\overset{-\mathrm{I}}{\mathrm{O}}{}_{2}^{2-}$ to $\overset{-\mathrm{I}}{\mathrm{O}}{}_{2}^{2-}$

Sometimes one might want to use formal oxidation numbers like 0.5 or ⅓:

```
\chemsetup[redox]{pos=top}
\ox{.5,Br2}
\ch{"\ox{1/3,I}" {}3+}

\chemsetup[redox]{pos=side}
\ox{1/3,I3+}
```

$\overset{0.5}{\mathrm{Br}}{}_{2}$ $\overset{⅓}{\mathrm{I}}{}_{3}^{+}$
$\mathrm{I}_{3}^{+}$ (⅓)

The fraction is displayed with the help of the xfrac package [L3P]. For more details on how CHEMMACROS uses it read section 28 starting on page 65.

## 19.2. Redox Reactions

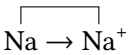
CHEMMACROS provides two commands to visualize the transfer of electrons in redox reactions. Both commands are using Ti*k*Z.

\OX{⟨*name*⟩,⟨*atom*⟩}
Label ⟨*atom*⟩ with the label ⟨*name*⟩.

`\redox(`⟨*name1*⟩`,`⟨*name2*⟩`)[`⟨*tikz*⟩`][`⟨*num*⟩`]{`⟨*text*⟩`}`
Connect two ⟨*atom*⟩s previously labelled with `\OX`. Only the first argument `(`⟨*name1*⟩`,`⟨*name2*⟩`)` is required, the others are all optional.

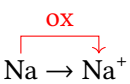
`\OX` places ⟨*atom*⟩ into a node, which is named with ⟨*name*⟩. If you have set two `\OX`, they can be connected with a line using `\redox`. To do so the names of the two nodes that are to be connected are written in the round braces. Since `\redox` draws a `tikzpicture` with options `remember picture,overlay`, the document needs to be *compiled at least two times*.

```
\vspace{7mm}
\OX{a,Na} $\rightarrow$ \OX{b,Na}\pch\redox(a,b){oxidation}
```



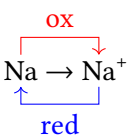

This line can be customized using Ti*k*Z keys in `[`⟨*tikz*⟩`]`:

```
\vspace{7mm}
\OX{a,Na} $\rightarrow$ \OX{b,Na}\pch\redox(a,b)[->,red]{ox}
```




With the argument `[`⟨*num*⟩`]` the length of the vertical parts of the line can be adjusted. The default length is `.6em`. This length is multiplied with ⟨*num*⟩. If you use a negative value the line is placed *below* the text.

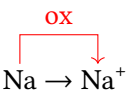
```
\vspace{7mm}
\OX{a,Na} $\rightarrow$ \OX{b,Na}\pch
\redox(a,b)[->,red]{ox}
\redox(a,b)[<-,blue][-1]{red}
\vspace{7mm}
```




The default length of the vertical lines can be customized with the option

`redox » dist = {`⟨*dim*⟩`}` Default: `.6em`
A TEX dimension.

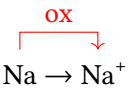
```
\vspace{7mm}
\chemsetup{redox/dist=1em}
\OX{a,Na} $\rightarrow$ \OX{b,Na}\pch\redox(a,b)[->,red]{ox}
```




redox » sep = {⟨dim⟩} Default: .2em

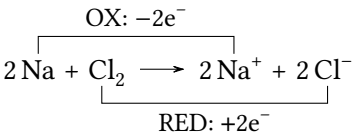
The option can be used to change the distance between the atom and the beginning of the line.

```
\vspace{7mm}
\chemsetup{redox/sep=.5em}
\OX{a,Na} $\rightarrow$ \OX{b,Na}\pch\redox(a,b)[->,red]{ox}
```

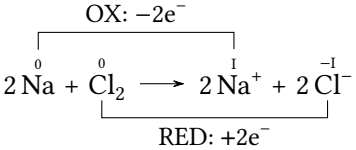



## 19.3. Examples

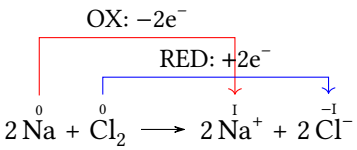
```
\vspace{7mm}
\ch{
  2 "\OX{o1,Na}" + "\OX{r1,Cl}" {}2
  ->
  2 "\OX{o2,Na}" {}+ + 2 "\OX{r2,Cl}" {}-
}
\redox(o1,o2){\small OX: $- 2\el$}
\redox(r1,r2)[][-1]{\small RED: $+ 2\el$}
\vspace{7mm}
```



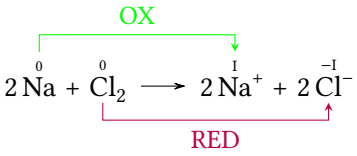

```
\vspace{7mm}
\ch{
  2 "\OX{o1,\ox*{0,Na}}" + "\OX{r1,\ox*{0,Cl}}" {}2
  ->
  2 "\OX{o2,\ox*{+1,Na}}" {}+ + 2 "\OX{r2,\ox*{-1,Cl}}" {}-
}
\redox(o1,o2){\small OX: $- 2\el$}
\redox(r1,r2)[][-1]{\small RED: $+ 2\el$}
\vspace{7mm}
```



```
\vspace{14mm}
\ch{
  2 "\OX{o1,\ox*{0,Na}}" + "\OX{r1,\ox*{0,Cl}}" {}2
  ->
  2 "\OX{o2,\ox*{+1,Na}}" {}+ + 2 "\OX{r2,\ox*{-1,Cl}}" {}-
}
\redox(o1,o2)[draw=red,->][3.33]{\small OX: $- 2\el$}
\redox(r1,r2)[draw=blue,->]{\small RED: $+ 2\el$}
```




```
\vspace{7mm}
\ch{
  2 "\OX{o1,\ox*{0,Na}}" + "\OX{r1,\ox*{0,Cl}}" {}2
  -> 2 "\OX{o2,\ox*{+1,Na}}" {}+ + 2 "\OX{r2,\ox*{-1,Cl}}" {}-
}
\redox(o1,o2)[green,-stealth]{\small OX}
\redox(r1,r2)[purple,-stealth][-1]{\small RED}
\vspace{7mm}
```




# 20. The scheme Module

The scheme module loads the chemnum package [Nie16b] and defines a floating environment \begin{scheme}. That is, it *only* defines this float if no environment scheme exists at the end of the preamble. The module checks for different available float defining methods, in *this* order:

- If the current class is a KOMA-Script class \DeclareNewTOC will be used.
- If the current class is memoir, memoir's methods are used.
- If the package tocbasic has been loaded \DeclareNewTOC will be used.
- If the package newfloat has been loaded \DeclareFloatingEnvironment will be used.

Introduced in version 5.1

- If the package floatrow has been loaded its method will be used.
- If the package float has been loaded its method will be used.
- If neither of the above the "manual" method is used. This means the environment is defined the same way like figure is defined in the article class or the book class, depending if \chapter is defined or not.

The list name and the caption name both are translated to the language specified according to the lang option and the provided translations, see section 26 starting on page 61 for details. If you want to manually change them then redefine these macros after begin document:

\listschemename
The name of the list of schemes.

\schemename
The name used in captions.

The list of schemes is printed as expected with

\listofschemes

Introduced in version 5.6

If you are using either cleveref or fancyref the scheme environment (or rather its captions) are supported already. For fancyref use the prefix sch.

# 21. The spectroscopy Module

The spectroscopy module loads the chemformula module and the siunitx package [Wri18].

## 21.1. The \NMR Command

When you're trying to find out if a compound is the one you think it is often NMR spectroscopy is used. The experimental data are typeset similar to this:

$$^{1}\text{H-NMR (400 MHz, } CDCl_3\text{): } \delta = 1.59$$

The spectroscopy module provides a command which simplifies the input.

\NMR*{⟨*num*⟩,⟨*element*⟩}(⟨*num*⟩,⟨*unit*⟩)[⟨*solvent*⟩]
Typeset nuclear magnetic resonance data. ⟨*num*⟩ is a valid siunitx number input, ⟨*unit*⟩ is a valid siunitx unit input. ⟨*solvent*⟩ is any valid CHEMFORMULA input as in \chcpd (this depends on the setting of the formula option, see 10 starting on page 22).

*All* Argument are optional! Without arguments we get:

```
1 \NMR \par
2 \NMR*
```

$^{1}$H-NMR: $\delta$
$^{1}$H-NMR

The first argument specifies the kind of NMR:

```
1 \NMR{13,C}
```

$^{13}$C-NMR: $\delta$

The second argument sets the frequency (in MHz):

```
1 \NMR(400)
```

$^{1}$H-NMR (400 MHz): $\delta$

You can choose another unit:

```
\NMR(4e8,\hertz)
```
$^{1}$H-NMR ($4 \times 10^{8}$ Hz): $\delta$

Please note that the setup of siunitx also affects this command:

```
\sisetup{exponent-product=\cdot}
\NMR(4e8,\hertz)
```
$^{1}$H-NMR ($4 \cdot 10^{8}$ Hz): $\delta$

The third argument specifies the solvent:

```
\NMR[CDCl3]
```
$^{1}$H-NMR ($CDCl_3$): $\delta$

## 21.2. Short Cuts

It is possible to define short cut commands for specific nuclei.

`\NewChemNMR{`⟨*cs*⟩`}{`⟨*num*⟩`,`⟨*atom*⟩`}`
Define a new shortcut macro for typesetting a certain type of magnetic resonence data. Gives an error if ⟨*cs*⟩ already exists.

`\DeclareChemNMR{`⟨*cs*⟩`}{`⟨*num*⟩`,`⟨*atom*⟩`}`
Define a new shortcut macro for typesetting a certain type of magnetic resonence data. Overwrites an existing macro.

`\RenewChemNMR{`⟨*cs*⟩`}{`⟨*num*⟩`,`⟨*atom*⟩`}`
Redefine an existing shortcut macro for typesetting a certain type of magnetic resonence data. Gives an error if ⟨*cs*⟩ doesn't exist.

`\ProvideChemNMR{`⟨*cs*⟩`}{`⟨*num*⟩`,`⟨*atom*⟩`}`
Define a new shortcut macro for typesetting a certain type of magnetic resonence data. ⟨*cs*⟩ is only defined if it doesn't exist, yet.

This defines a command with the same arguments as `\NMR` *except* for `{`⟨*num*⟩`,`⟨*atom*⟩`}`:

```
\NewChemNMR\HNMR{1,H}%
\NewChemNMR\CNMR{13,C}%
\CNMR*(100) \par
\HNMR*(400)
```
$^{13}$C-NMR (100 MHz)
$^{1}$H-NMR (400 MHz)

## 21.3. An Environment to Typeset Experimental Data

The spectroscopy module provides an environment to ease the input of experimental data.

`\begin{experimental}`
Environment for the output of experimental data. Inside the environment the following commands are defined.

`\data{`⟨*type*⟩`}[`⟨*specification*⟩`]`
Type of data, *e. g.* IR, MS… The optional argument takes further specifications which are output in parentheses.

`\data*{`⟨*type*⟩`}[`⟨*specification*⟩`]`
Like `\data` but changes the `=` into a `:`, given that `use-equal = {true}` is used.

`\NMR{`⟨*num*⟩`,`⟨*elem*⟩`[`⟨*coupling core*⟩`]}(`⟨*num*⟩`,`⟨*unit*⟩`)[`⟨*solvent*⟩`]`
This command gets an additional argument: `\NMR{13,C[^1H]}` $^{13}C\{^1H\}$-NMR: $\delta$

`\J(`⟨*bonds*⟩`;`⟨*nuclei*⟩`)[`⟨*unit*⟩`]{`⟨*list of nums*⟩`}`
Coupling constant, values are input separated by `;` (NMR). The arguments `(`⟨*bonds*⟩`;`⟨*nuclei*⟩`)` and `[`⟨*unit*⟩`]` are optional and enable further specifications of the coupling.

`\#{`⟨*num*⟩`}`
Number of nuclei (NMR).

`\pos{`⟨*num*⟩`}`
Position of nuclues (NMR).

`\val{`⟨*num*⟩`}`
A number, an alias of siunitx' `\num{`⟨*num*⟩`}`.

`\val{`⟨*num1*⟩`--`⟨*num2*⟩`}`
An alias of siunitx' `\numrange{`⟨*num1*⟩`}{`⟨*num2*⟩`}`.

```
\begin{experimental}
  \data{type1} Data.
  \data{type2}[specifications] More data.
  \data*{type3} Even more data.
\end{experimental}
```

type1 Data. type2 (specifications) More data. type3 Even more data.

## 21.4. Customization

The output of the environment and of the NMR commands can be customized be a number of options. For historical reasons they all belong to the module `nmr`.

`spectroscopy` » `unit = {`⟨*unit*⟩`}` Default: `\mega\hertz`
The used default unit.

`spectroscopy` » `nucleus = {`⟨*num*⟩`,`⟨*atom*⟩`}` Default: `{1,H}`
The used default nucleus.

`spectroscopy` » `connector = {`⟨*code*⟩`}` Default: -
Places ⟨*code*⟩ between the nucleus and the method.

`spectroscopy` » `method = {`⟨*code*⟩`}` Default: `NMR`
The measuring method.

`spectroscopy` » `format = {`⟨*commands*⟩`}` (initially empty)
For example `\bfseries`.

`spectroscopy` » `nmr-base-format = {`⟨*commands*⟩`}` (initially empty)
Introduced in version 5.8
Formatting instructions for the NMR base.

spectroscopy » pos-number = side|sub|super — Default: side
Position of the number next to the atom.

spectroscopy » coupling-symbol = {⟨code⟩} — Default: J
The symbol used for the coupling constant.

spectroscopy » coupling-unit = {⟨unit⟩} — Default: \hertz
A siunitx unit.

spectroscopy » coupling-pos = side|sub — Default: side
Placement of the coupling nuclei next to the symbol $J$ (or rather the symbol specified with option coupling-symbol).

spectroscopy » coupling-nuclei-pre = {⟨code⟩} — Default: (
Code inserted before the coupling nuclei when coupling-pos = {side}.

spectroscopy » coupling-nuclei-post = {⟨code⟩} — Default: )
Code inserted after the coupling nuclei when coupling-pos = {side}.

spectroscopy » coupling-bonds-pre = {⟨code⟩} — (initially empty)
Code inserted before the coupling bonds.

spectroscopy » coupling-bonds-post = {⟨code⟩} — Default: \!
Code inserted after the coupling bonds.

spectroscopy » coupling-pos-cs = {⟨cs⟩} — Default: \@firstofone
Set the macro that prints the number set with the \pos macro. This needs to be a command with one mandatory argument.

spectroscopy » atom-number-cs = {⟨cs⟩} — Default: \@firstofone
Set the macro that prints the number set with the \# macro. This needs to be a command with one mandatory argument.

spectroscopy » atom-number-space = {⟨dim⟩} — Default: .16667em
Introduced in version 5.3
Horizontal space inserted between number and atom (printed by \#).

spectroscopy » parse = true|false — Default: true
Treat the solvent as CHEMFORMULA formula (this depends on the setting of the formula option, see 10 starting on page 22) or not.

spectroscopy » delta = {⟨tokens⟩} — (initially empty)
The ⟨tokens⟩ are added after $\delta$.

spectroscopy » list = true|false — Default: false
The environment nmr is formatted as a list

spectroscopy » list-setup = {⟨setup⟩}
Setup of the list. See below for the default settings.

spectroscopy » use-equal = true|false — Default: false
Add egual sign after \NMR and \data.

The default setup of the list:

```
\topsep\z@skip \partopsep\z@skip
\itemsep\z@ \parsep\z@ \itemindent\z@
\leftmargin\z@
```

```
\begin{experimental}[format=\bfseries]
  \data{type1} Data.
  \data{type2}[specifications] More data.
  \data*{type3} Even more data.
\end{experimental}
```

**type1** Data. **type2 (specifications)** More data. **type3** Even more data.

The command \NMR and all commands defined through \NewChemNMR can be used like \data for the NMR data.

```
\begin{experimental}[format=\bfseries,use-equal]
  \data{type1} Data.
  \data{type2}[specifications] More data.
  \NMR Even more data.
\end{experimental}
```

**type1** = Data. **type2 (specifications)** = More data. **$^1$H-NMR:** $\delta$ =Even more data.

## 21.5. An Example

The code below is shown with different specifications for ⟨*options*⟩. Of course options can also be chosen with \chemsetup.

```
\sisetup{separate-uncertainty,per-mode=symbol,detect-all,range-phrase=--}
\begin{experimental}[<optionen>]
  \data*{yield} \SI{17}{\milli\gram} yellow needles (\SI{0.04}{\milli\mole},
    \SI{13}{\percent}).
  %
  \data{mp.} \SI{277}{\celsius} (DSC).
  %
  \NMR(600)[CDCl3] \val{2.01} (s, \#{24}, \pos{5}), \val{2.31} (s, \#{12},
  \pos{1}), \val{6.72--6.74} (m, \#{2}, \pos{11}), \val{6.82} (s, \#{8},
  \pos{3}), \val{7.05--7.07} (m, \#{2}, \pos{12}), \val{7.39--7.41} (m,
  \#{4},
  \pos{9}), \val{7.48--7.49} (m, \#{4}, \pos{8}).
  %
  \NMR{13,C}(150)[CDCl3] \val{21.2} ($+$, \#{4}, \pos{1}), \val{23.4} ($+$,
  \#{8}, \pos{5}), \val{126.0} ($+$, \#{4}, \pos{9}), \val{128.2} ($+$,
  \#{8},
  \pos{3}), \val{130.8} ($+$, \#{2}, \pos{12}), \val{133.6} ($+$, \#{2},
  \pos{11}), \val{137.0} ($+$, \#{4}, \pos{8}), \val{138.6} (q, \#{4},
  \pos{2}), \val{140.6} (q, \#{2}, \pos{10}), \val{140.8} (q, \#{8}, \pos{4})
  ,
```

```
  \val{141.8} (q, \#{4}, \pos{6}), \val{145.6} (q, \#{2}, \pos{7}).
  %
  \data{MS}[DCP, EI, \SI{60}{\electronvolt}] \val{703} (2, \ch{M+}), \val
  {582}
  (1), \val{462} (1), \val{249} (13), \val{120} (41), \val{105} (100).
  %
  \data{MS}[\ch{MeOH + H2O + KI}, ESI, \SI{10}{\electronvolt}] \val{720}
  (100,
  \ch{M+ + OH-}), \val{368} (\ch{M+ + 2 OH-}).
  %
  \data{IR}[KBr] \val{3443} (w), \val{3061} (w), \val{2957} (m), \val{2918}
  (m), \val{2856} (w), \val{2729} (w), \val{1725} (w), \val{1606} (s),
  \val{1592} (s), \val{1545} (w), \val{1446} (m), \val{1421} (m), \val{1402}
  (m), \val{1357} (w), \val{1278} (w), \val{1238} (s), \val{1214} (s),
  \val{1172} (s), \val{1154} (m), \val{1101} (w), \val{1030} (w), \val{979}
  (m), \val{874} (m), \val{846} (s), \val{818} (w), \val{798} (m), \val{744}
  (w), \val{724} (m), \val{663} (w), \val{586} (w), \val{562} (w), \val{515}
  (w).
  %
  \data*{UV-Vis} \SI{386}{\nano\metre} ($\varepsilon = \val{65984}$),
  \SI{406}{\nano\metre} ($\varepsilon = \val{65378}$).
  %
  \data*{quantum yield} $\Phi = \val{0.74+-0.1}$\,.
\end{experimental}
```

## 21.6. Nearly Standard

Output with these options:

```
delta=(ppm),pos-number=sub,use-equal
```

yield: 17 mg yellow needles (0.04 mmol, 13 %). mp. = 277 °C (DSC). $^{1}$H-NMR (600 MHz, $CDCl_3$): $\delta$ (ppm) = 2.01 (s, 24 H, $H_5$), 2.31 (s, 12 H, $H_1$), 6.72–6.74 (m, 2 H, $H_{11}$), 6.82 (s, 8 H, $H_3$), 7.05–7.07 (m, 2 H, $H_{12}$), 7.39–7.41 (m, 4 H, $H_9$), 7.48–7.49 (m, 4 H, $H_8$). $^{13}$C-NMR (150 MHz, $CDCl_3$): $\delta$ (ppm) = 21.2 (+, 4 C, $C_1$), 23.4 (+, 8 C, $C_5$), 126.0 (+, 4 C, $C_9$), 128.2 (+, 8 C, $C_3$), 130.8 (+, 2 C, $C_{12}$), 133.6 (+, 2 C, $C_{11}$), 137.0 (+, 4 C, $C_8$), 138.6 (q, 4 C, $C_2$), 140.6 (q, 2 C, $C_{10}$), 140.8 (q, 8 C, $C_4$), 141.8 (q, 4 C, $C_6$), 145.6 (q, 2 C, $C_7$). MS (DCP, EI, 60 eV) = 703 (2, $M^+$), 582 (1), 462 (1), 249 (13), 120 (41), 105 (100). MS (MeOH + $H_2O$ + KI, ESI, 10 eV) = 720 (100, $M^+ + OH^-$), 368 ($M^+ + 2\,OH^-$). IR (KBr) = 3443 (w), 3061 (w), 2957 (m), 2918 (m), 2856 (w), 2729 (w), 1725 (w), 1606 (s), 1592 (s), 1545 (w), 1446 (m), 1421 (m), 1402 (m), 1357 (w), 1278 (w), 1238 (s), 1214 (s), 1172 (s), 1154 (m), 1101 (w), 1030 (w), 979 (m), 874 (m), 846 (s), 818 (w), 798 (m), 744 (w), 724 (m), 663 (w), 586 (w), 562 (w), 515 (w). UV-Vis: 386 nm ($\varepsilon$ = 65 984), 406 nm ($\varepsilon$ = 65 378). quantum yield: $\Phi = 0.74 \pm 0.10$ .

## 21.7. Formatted List

Output with these options:

```
format=\bfseries,delta=(ppm),list=true,use-equal
```

**yield:** 17 mg yellow needles (0.04 mmol, 13 %).
**mp.** = 277 °C (DSC).
**$^{1}$H-NMR (600 MHz, $CDCl_3$):** $\delta$ (ppm) = 2.01 (s, 24 H, $H_5$), 2.31 (s, 12 H, $H_1$), 6.72–6.74 (m, 2 H, $H_{11}$), 6.82 (s, 8 H, $H_3$), 7.05–7.07 (m, 2 H, $H_{12}$), 7.39–7.41 (m, 4 H, $H_9$), 7.48–7.49 (m, 4 H, $H_8$).
**$^{13}$C-NMR (150 MHz, $CDCl_3$):** $\delta$ (ppm) = 21.2 (+, 4 C, $C_1$), 23.4 (+, 8 C, $C_5$), 126.0 (+, 4 C, $C_9$), 128.2 (+, 8 C, $C_3$), 130.8 (+, 2 C, $C_{12}$), 133.6 (+, 2 C, $C_{11}$), 137.0 (+, 4 C, $C_8$), 138.6 (q, 4 C, $C_2$), 140.6 (q, 2 C, $C_{10}$), 140.8 (q, 8 C, $C_4$), 141.8 (q, 4 C, $C_6$), 145.6 (q, 2 C, $C_7$).
**MS (DCP, EI, 60 eV)** = 703 (2, $M^+$), 582 (1), 462 (1), 249 (13), 120 (41), 105 (100).
**MS (MeOH + $H_2O$ + KI, ESI, 10 eV)** = 720 (100, $M^+ + OH^-$), 368 ($M^+ + 2\,OH^-$).
**IR (KBr)** = 3443 (w), 3061 (w), 2957 (m), 2918 (m), 2856 (w), 2729 (w), 1725 (w), 1606 (s), 1592 (s), 1545 (w), 1446 (m), 1421 (m), 1402 (m), 1357 (w), 1278 (w), 1238 (s), 1214 (s), 1172 (s), 1154 (m), 1101 (w), 1030 (w), 979 (m), 874 (m), 846 (s), 818 (w), 798 (m), 744 (w), 724 (m), 663 (w), 586 (w), 562 (w), 515 (w).
**UV-Vis:** 386 nm ($\varepsilon$ = 65 984), 406 nm ($\varepsilon$ = 65 378).
**quantum yield:** $\Phi$ = 0.74 ± 0.10 .

## 21.8. Crazy

Output for these options:

```
format=\color{red}\itshape,
list=true,
delta=\textcolor{green}{\ch{M+ + H2O}},
pos-number=side,
coupling-unit=\mega\gram\per\square\second,
list-setup=,
use-equal
```

*yield:* 17 mg yellow needles (0.04 mmol, 13 %).

*mp.* = 277 °C (DSC).

*$^{1}$H-NMR (600 MHz, $CDCl_3$):* $\delta$ $M^+ + H_2O$ = 2.01 (s, 24 H, H-5), 2.31 (s, 12 H, H-1), 6.72–6.74 (m, 2 H, H-11), 6.82 (s, 8 H, H-3), 7.05–7.07 (m, 2 H, H-12), 7.39–7.41 (m, 4 H, H-9), 7.48–7.49 (m, 4 H, H-8).

*$^{13}$C-NMR (150 MHz, $CDCl_3$):* $\delta$ $M^+ + H_2O$ = 21.2 (+, 4 C, C-1), 23.4 (+, 8 C, C-5), 126.0 (+, 4 C, C-9), 128.2 (+, 8 C, C-3), 130.8 (+, 2 C, C-12), 133.6 (+, 2 C, C-11), 137.0 (+, 4 C, C-8), 138.6 (q, 4 C, C-2), 140.6 (q, 2 C, C-10), 140.8 (q, 8 C, C-4), 141.8 (q, 4 C, C-6), 145.6 (q, 2 C, C-7).

*MS (DCP, EI, 60 eV)* = 703 (2, $M^+$), 582 (1), 462 (1), 249 (13), 120 (41), 105 (100).

*MS (MeOH + $H_2O$ + KI, ESI, 10 eV)* = 720 (100, $M^+ + OH^-$), 368 ($M^+ + 2\,OH^-$).

*IR (KBr)* = 3443 (w), 3061 (w), 2957 (m), 2918 (m), 2856 (w), 2729 (w), 1725 (w), 1606 (s), 1592 (s), 1545 (w), 1446 (m), 1421 (m), 1402 (m), 1357 (w), 1278 (w), 1238 (s), 1214 (s), 1172 (s), 1154 (m), 1101 (w), 1030 (w), 979 (m), 874 (m), 846 (s), 818 (w), 798 (m), 744 (w), 724 (m), 663 (w), 586 (w), 562 (w), 515 (w).

*UV-Vis:* 386 nm ($\varepsilon$ = 65 984), 406 nm ($\varepsilon$ = 65 378).

*quantum yield:* $\Phi$ = 0.74 ± 0.10 .

# 22. The thermodynamics Module

The thermodynamics module loads the siunitx package [Wri18].

## 22.1. The \state Macro

\state[⟨*options*⟩]{⟨*symbol*⟩}
Typeset a state variable.

This macro can be used to write the thermodynamic state variables.

```
\state{A}, \state[subscript-left=f]{G} ,
\state[subscript-right=\ch{Na}]{E},
\state[superscript-right=\SI{1000}{\celsius}]{H}
```

$\Delta A^{\ominus}$, $\Delta_{\mathrm{f}} G^{\ominus}$ , $\Delta E^{\ominus}_{\mathrm{Na}}$, $\Delta H^{1000\,^{\circ}\mathrm{C}}$

These options are available:

thermodynamics » pre = {⟨*text*⟩} Default: \changestate
Code inserted before the variable. Inserted in text mode.

thermodynamics » post = {⟨*text*⟩} (initially empty)
Code inserted after the variable. Inserted in text mode.

thermodynamics » superscript-left = {⟨*text*⟩} (initially empty)
The left superscript. Inserted in text mode.

thermodynamics » superscript-right = {⟨*text*⟩} Default: \standardstate
The right superscript. Inserted in text mode.

thermodynamics » superscript = {⟨*text*⟩}
An alias of superscript-right.

thermodynamics » subscript-left = {⟨*text*⟩} (initially empty)
The left subscript. Inserted in text mode.

thermodynamics » subscript-right = {⟨*text*⟩} (initially empty)
The right subscript. Inserted in text mode.

thermodynamics » subscript = {⟨*text*⟩}
An alias of subscript-left.

## 22.2. Thermodynamic Variables

The thermodynamics module provides a few commands for specific thermodynamic variables:

\enthalpy*[⟨*options*⟩](⟨*subscript*⟩){⟨*value*⟩}
Typeset the amount of enthalpy.

\entropy*[⟨*options*⟩](⟨*subscript*⟩){⟨*value*⟩}
Typeset the amount of entropy.

`\gibbs*[`⟨*options*⟩`](`⟨*subscript*⟩`){`⟨*value*⟩`}`
Typeset the amount of Gibbs enthalpy.

Their usage is pretty much self-explaining:

```
\enthalpy{123} \par
\entropy{123} \par
\gibbs{123}
```

$\Delta H^\ominus = 123\,\mathrm{kJ\,mol^{-1}}$
$S^\ominus = 123\,\mathrm{J\,K^{-1}\,mol^{-1}}$
$\Delta G^\ominus = 123\,\mathrm{kJ\,mol^{-1}}$

The argument `(`⟨*subscript*⟩`)` adds a subscript for specification, `*` hides number and unit:

```
  \enthalpy(r){123} \par
  \enthalpy*{123} \par
```

$\Delta_\mathrm{r} H^\ominus = 123\,\mathrm{kJ\,mol^{-1}}$
$\Delta H^\ominus$

`thermodynamics » pre = {`⟨*text*⟩`}` Default: `\changestate`
Code inserted before the variable. Inserted in text mode.

`thermodynamics » post = {`⟨*text*⟩`}` (initially empty)
Code inserted after the variable. Inserted in text mode.

`thermodynamics » superscript-left = {`⟨*text*⟩`}` (initially empty)
The left superscript. Inserted in text mode.

`thermodynamics » superscript-right = {`⟨*text*⟩`}` Default: `\standardstate`
The right superscript. Inserted in text mode.

`thermodynamics » superscript = {`⟨*text*⟩`}`
An alias of `superscript-right`.

`thermodynamics » subscript-left = {`⟨*text*⟩`}` (initially empty)
The left subscript. Inserted in text mode.

`thermodynamics » subscript-right = {`⟨*text*⟩`}` (initially empty)
The right subscript. Inserted in text mode.

`thermodynamics » subscript = {`⟨*text*⟩`}`
An alias of `subscript-left`.

`thermodynamics » subscript-pos = left|right` Default: `left`
Determines wether the subscript given in `(`⟨*subscript*⟩`)` is placed to the left or the right of the variable.

`thermodynamics » symbol = {`⟨*symbol*⟩`}` (initially empty)
The symbol of the variable. Inserted in math mode.

`thermodynamics » unit = {`⟨*unit*⟩`}` (initially empty)
A valid siunitx unit.

The default values depend on the command.

```
\enthalpy[unit=\kilo\joule]{-285} \
  par
\gibbs[pre=]{0} \par
\entropy[pre=$\Delta$,superscript
  =]{56.7}
```

$\Delta H^\ominus = -285\,\mathrm{kJ}$
$G^\ominus = 0\,\mathrm{kJ\,mol^{-1}}$
$\Delta S = 56.7\,\mathrm{J\,K^{-1}\,mol^{-1}}$

The unit is set corresponding to the rules of siunitx and depends on its settings:

```
\enthalpy{-1234.56e3} \par
\sisetup{
  per-mode=symbol,
  exponent-product=\cdot,
  output-decimal-marker={,},
  group-four-digits=true
}
\enthalpy{-1234.56e3}
```

$\Delta H^\ominus = -1234.56 \times 10^3\,\mathrm{kJ\,mol^{-1}}$
$\Delta H^\ominus = -1\,234{,}56 \cdot 10^3\,\mathrm{kJ/mol}$

## 22.3. Create New Variables or Redefine Existing Ones

`\NewChemState{`⟨*cs*⟩`}{`⟨*options*⟩`}`
Define new state commands like `\enthalpy`. Gives an error is ⟨*cs*⟩ already exists.

`\RenewChemState{`⟨*cs*⟩`}{`⟨*options*⟩`}`
Redefine existing state commands.

`\DeclareChemState{`⟨*cs*⟩`}{`⟨*options*⟩`}`
Like `\NewChemState` but gives now error if ⟨*cs*⟩ already exists.

`\ProvideChemState{`⟨*cs*⟩`}{`⟨*options*⟩`}`
Define new state commands like `\enthalpy`. Defines ⟨*cs*⟩ only if it is not defined, yet.

The argument ⟨*options*⟩ is a comma separated list of key/value options:

`thermodynamics » pre = {`⟨*text*⟩`}` Default: `\changestate`
Code inserted before the variable. Inserted in text mode.

`thermodynamics » post = {`⟨*text*⟩`}` (initially empty)
Code inserted after the variable. Inserted in text mode.

`thermodynamics » superscript-left = {`⟨*text*⟩`}` (initially empty)
The left superscript. Inserted in text mode.

`thermodynamics » superscript-right = {`⟨*text*⟩`}` Default: `\standardstate`
The right superscript.

`thermodynamics » superscript = {`⟨*text*⟩`}`
An alias of `superscript-right`.

`thermodynamics » subscript-left = {`⟨*text*⟩`}` (initially empty)
The left subscript. Inserted in text mode.

`thermodynamics » subscript-right = {`⟨*text*⟩`}` (initially empty)
The right subscript. Inserted in text mode.

thermodynamics » subscript = {⟨*text*⟩}
An alias of subscript-left.

thermodynamics » subscript-pos = left|right Default: left
Determines wether the subscript given in (⟨*subscript*⟩) is placed to the left or the right of the variable.

thermodynamics » symbol = {⟨*symbol*⟩} (initially empty)
The symbol of the variable.

thermodynamics » unit = {⟨*unit*⟩} (initially empty)
A valid siunitx unit.

```
\NewChemState\Helmholtz{ symbol=A , unit=\kilo\joule\per\mole }
\NewChemState\ElPot{ symbol=E , subscript-pos=right , superscript= , unit=\
  volt }
\Helmholtz{123.4} \par
\ElPot{-1.1} \par
\ElPot[superscript=0]($\ch{Sn}|\ch{Sn^2+}||\ch{Pb^2+}|\ch{Pb}$){0.01} \par
\RenewChemState\enthalpy{ symbol=h , unit=\joule} \par
\enthalpy(f){12.5}
```

$\Delta A^{\ominus} = 123.4\,\mathrm{kJ\,mol^{-1}}$
$\Delta E = -1.1\,\mathrm{V}$
$\Delta E^{\mathrm{o}}_{\mathrm{Sn|Sn^{2+}||Pb^{2+}|Pb}} = 0.01\,\mathrm{V}$
$\Delta_{\mathrm{f}} h^{\ominus} = 12.5\,\mathrm{J}$

The existing commands have been defined like this:

```
\NewChemState \enthalpy{ symbol = H, unit = \kilo\joule\per\mole }
\NewChemState \entropy { symbol = S, unit = \joule\per\kelvin\per\mole, pre =
  }
\NewChemState \gibbs   { symbol = G, unit = \kilo\joule\per\mole }
```

So – for following thermodynamic conventions – one could define a molar and an absolute variable:

```
\RenewChemState\enthalpy{symbol=h,superscript=,unit=\kilo\joule\per\mole}%
  molar
\NewChemState\Enthalpy{symbol=H,superscript=,unit=\kilo\joule}% absolute
\enthalpy{-12.3} \Enthalpy{-12.3}
```

$\Delta h = -12.3\,\mathrm{kJ\,mol^{-1}}$ $\Delta H = -12.3\,\mathrm{kJ}$

# 23. The units Module

The units module loads the siunitx package [Wri18].

In chemistry some non-SI units are very common. siunitx provides the command

`\DeclareSIUnit{⟨cs⟩}{⟨unit⟩}`
Define ⟨*cs*⟩ to be a valid unit command inside siunitx' macros `\SI` and `\si` which represents ⟨*unit*⟩.

to add arbitrary units. CHEMMACROS uses that command to provide some units. Like all siunitx units they're only valid inside `\SI{⟨num⟩}{⟨unit⟩}` and `\si{⟨unit⟩}`.

`\atmosphere`
atm

`\atm`
atm

`\calory`
cal

`\cal`
cal

`\cmc`
$\mathrm{cm}^3$

The units `\cmc`, `\molar`, and `\Molar` are defined by the package chemstyle as well. CHEMMACROS only defines them, if chemstyle is not loaded.

`\molar`
$\mathrm{mol\,dm}^{-3}$

`\moLar`
$\mathrm{mol\,L}^{-1}$

`\Molar`
M

`\MolMass`
$\mathrm{g\,mol}^{-1}$

`\normal`
N

`\torr`
Torr

By the way: `\mmHg` mmHg already is defined by siunitx.

# Part IV.
# Core Modules

The modules described in this part are always loaded and mainly concern module writers.

# 24. The base Module

The base module is the core module of CHEMMACROS. It defines some tools which can (and should) be used in other modules. This means this section is only interesting for you if you plan to write a module yourself (see section A starting on page 66 for details).

This module requires the packages bm [CM19], amstext [MS00], and etoolbox [Leh19].

This module also provides `\chemsetup` and the option `modules`.

It also provides a number of (expl3) macros which may be used in other modules. In the macro descriptions below *TF* denotes that a `T`, an `F` and a `TF` variant exist. In case of an expandable conditional (*) also the predicate variant is available.

* `\chemmacros_if_loaded:nnTF {package|class} {⟨name⟩} {⟨true⟩} {⟨false⟩}`
Checks if package (or class) ⟨*name*⟩ has been loaded. Also works after begin document.

* `\chemmacros_if_package_loaded:nTF {⟨name⟩} {⟨true⟩} {⟨false⟩}`
Checks if package ⟨*name*⟩ has been loaded. Also works after begin document.

* `\chemmacros_if_class_loaded:nTF {⟨name⟩} {⟨true⟩} {⟨false⟩}`
Checks if class ⟨*name*⟩ has been loaded. Also works after begin document.

`\chemmacros_nobreak:`
Inserts a penalty of 10 000.

`\chemmacros_allow_break:`
Inserts a penalty of 0.

`\chemmacros_skip_nobreak:N` ⟨*skip/length variable*⟩
Insert a horizontal skip where a linebreak is disallowed.

* `\chemmacros_if_is_int:nTF {⟨input⟩} {⟨true⟩} {⟨false⟩}`
Checks if ⟨*input*⟩ is an integer or something else.

`\chemmacros_if_bold:TF {⟨true⟩} {⟨false⟩}`
Checks if the current font weight is one of `b`, `bc`, `bm`, `bx`, `bux`, `eb`, `ebc`, `ebx`, `mb`, `sb`, `sbc`, `sbx`, `ub`, `ubc` or `ubx`.

`\chemmacros_bold:n {⟨text⟩}`
Checks if the current font weight is bold and if yes places ⟨*text*⟩ in `\textbf` if in text mode or in `\bm` if in math mode. If no ⟨*text*⟩ simply is placed in the input stream as is.

`\chemmacros_text:n {⟨text⟩}`
Ensures that ⟨*text*⟩ is placed in text mode.

`\chemmacros_math:n {⟨text⟩}`
Ensures that ⟨*text*⟩ is placed in math mode.

`\chemmacros_new_macroset:nnn {⟨name⟩} {⟨arg spec⟩} {⟨internal command call⟩}`
Changed in version 5.3b ()
A command to define a set of macros `\NewChem`⟨*name*⟩, `\RenewChem`⟨*name*⟩, `\DeclareChem`⟨*name*⟩ and `\ProvideChem`⟨*name*⟩ where the first letter of ⟨*name*⟩ is converted to uppercase, other letters are kept unchanged. ⟨*arg spec*⟩ is any valid argument specification for xparse's `\DeclareDocumentCommand` [L3P]. ⟨*internal command call*⟩ should be a macro which makes definitions *without* error checks, *i. e.*, define new macros or redefine existing ones like `\def` does. This macro just should get the arguments passed on to. Have a look at the example below.

\chemmacros_new_environment_macroset:nnn {⟨name⟩} {⟨arg spec⟩} {⟨internal command call⟩}
Like \chemmacros_new_macroset:nnn but for environments.

\NewChemMacroset*{⟨name⟩}{⟨arg spec⟩}{⟨internal command call⟩}
A non-expl3 version of \chemmacros_new_macroset:nnn for LaTeX 2ε programmers. The starred version calls \chemmacros_new_environment_macroset:nnn.

\chemmacros_add_cleveref_support:nnnnn {⟨counter⟩} {⟨singular⟩} {⟨plural⟩} {⟨uppercase singular⟩} {⟨uppercase plural⟩}
Introduced in version 5.6
A command to add suiting names for a counter for the cleveref package's \cref commands. This command acts at the end of the preamble and only if a user hasn't provided definitions with \crefname already.

\ChemCleverefSupport{⟨counter⟩}{⟨singular⟩}[⟨uppercase singular⟩]{⟨plural⟩}[⟨uppercase plural⟩]
Introduced in version 5.6
LaTeX 2ε-version of \chemmacros_add_cleveref_support:nnnnn.

\chemmacros_add_fancyref_support:nnn {⟨prefix⟩} {⟨name⟩} {⟨uppercase name⟩}
Introduced in version 5.6
A command to add suiting names for a counter for the fancyref package's \fref commands. This command acts at the end of the preamble and doesn't override definitions made by the users.

\ChemFancyrefSupport{⟨prefix⟩}{⟨name⟩}[⟨uppercase name⟩]
Introduced in version 5.6
LaTeX 2ε-version of \chemmacros_add_fancyref_support:nnnnn.

This is how the macros \NewChemParticle, \RenewChemParticle, \DeclareChemParticle and \ProvideChemParticle were defined:

```
\NewChemMacroset {Particle} {mm}
  { \chemmacros_define_particle:Nn #1 {#2} }
```

The following macros strictly speaking are not provided by the base module but this place fits best for their description.

* \chemmacros_if_module_exist:nTF {⟨module⟩} {⟨true⟩} {⟨false⟩}
Checks if a file with the correct name for a module ⟨module⟩ can be found.

* \chemmacros_if_module_loaded:nTF {⟨module⟩} {⟨true⟩} {⟨false⟩}
Checks if the module ⟨module⟩ has already been loaded or not.

\chemmacros_load_module:n {⟨module⟩}
Loads module ⟨module⟩ if it hasn't been loaded, yet.

\chemmacros_load_modules:n {⟨csv list of modules⟩}
Loads every module in ⟨csv list of modules⟩ if they haven't been loaded, yet. This is the code level variant of \usechemmodule.

\chemmacros_before_module:nn {⟨module⟩} {⟨code⟩}
Introduced in version 5.1
Saves ⟨code⟩ and inserts it right before ⟨module⟩ is loaded. If ⟨module⟩ is never loaded then ⟨code⟩ is never inserted. If ⟨module⟩ already is loaded when the command is used then ⟨code⟩ also is never inserted.

`\chemmacros_after_module:nn` {⟨*module*⟩} {⟨*code*⟩}

Introduced in version 5.1

Saves ⟨*code*⟩ and inserts it right after ⟨*module*⟩ is loaded. If ⟨*module*⟩ is never loaded then ⟨*code*⟩ is never inserted. If ⟨*module*⟩ already is loaded when the command is used then ⟨*code*⟩ is inserted immediately.

# 25. The errorcheck Module

Introduced in version 5.2

The errorcheck module provides some rudimentary support for giving users more meaningful messages when they use a command or environment provided by a module that they haven't loaded.

# 26. The lang Module

The lang module provides language support for CHEMMACROS. It loads the package translations [Nie20b].

## 26.1. Information For Users

This module defines the following option:

`language` = auto|⟨*language*⟩ Default: auto

If set to auto CHEMMACROS will detect the language used by babel [Bra19] or polyglossia [Cha19] automatically, the fallback translation is English and will be used if no translation for the actual language is available. Any language known to the translations package is a valid value for ⟨*language*⟩.

The language chosen via `language` is used for translation of certain strings in different places all over CHEMMACROS. They are mentioned in the places when the corresponding function of CHEMMACROS is explained.

Translation is done with the help of the translations package, available translation keys are listed in section 26.2.

## 26.2. Available Translation Keys

*Changed in version 5.6*

Table 4 on the next page lists all predefined translations of the available keys. *Some of the translations have changed in version 5.6.* The lang module doesn't provide the translations themselves – they are provided by the corresponding modules. A translation key is a unique string[7] of characters. Each key is used to identify a replacement text which depends on the current language or the language set through the `language` option. For each key at least the English fallback translation is provided, for most also the German translation is provided. For a few keys also other translations are provided. If you find that a translation for your language is missing you can provide it in the preamble:

`\DeclareChemTranslation`{⟨*key*⟩}{⟨*language*⟩}{⟨*translation*⟩}

Introduced in version 5.6

A command which makes an abstraction from the translations package. It should be used in documents for adding missing translations that are needed. This command can only be used in the preamble.

7. That is, a string using the definition for strings used for expl3, i.e., converted to a series of category code 12 characters..

`\DeclareChemTranslations{⟨key⟩}{⟨language⟩ = ⟨translation⟩}`

Introduced in version 5.6

A command rather meant for module writers but can be used by document authors as well, of course. It gets a csv list of key/value pairs of translations. This command can only be used in the preamble.

If you send me an email (see section B starting on page 68) with the translations for your language I'll gladly add them to the next release of CHEMMACROS!

*Please do not use translations' `\DeclareTranslation` for declaring translations.*

TABLE 4: Translation keys predefined by CHEMMACROS (except `phase-aqi`, `phase-cd` and `phase-lc` which were defined in this document).

| key | language | translation |
|---|---|---|
| K-acid | fallback | \mathrm {a} |
| K-base | fallback | \mathrm {b} |
| K-water | fallback | \mathrm {w} |
| K-acid | German | \mathrm {s} |
| K-acid | Dutch | \mathrm {z} |
| phase-sld | fallback | s |
| phase-lqd | fallback | l |
| phase-gas | fallback | g |
| phase-aq | fallback | aq |
| phase-sld | German | f |
| phase-lqd | German | f\/l |
| list-of-reactions | fallback | List of Reactions |
| list-of-reactions | English | List of Reactions |
| list-of-reactions | German | Reaktionsverzeichnis |
| list-of-reactions | Italian | Elenco delle reazioni |
| list-of-reactions | French | Table des r\'{e}actions |
| list-of-reactions | Dutch | Lijst van reacties |
| list-of-reactions | Norwegian | Reaksjonsliste |
| list-of-reactions | Nynorsk | Reaksjonsliste |
| reaction | fallback | reaction |
| reaction | English | reaction |
| reaction | German | Reaktion |
| reaction | Italian | reazione |
| reaction | French | r\'{e}action |
| reaction | Dutch | reactie |
| reaction | Norwegian | reaksjon |
| reaction | Nynorsk | reaksjon |
| reactions | fallback | reactions |
| reactions | English | reactions |
| reactions | German | Reaktionen |
| reactions | Italian | reazioni |
| reactions | French | r\'{e}actions |
| reactions | Dutch | reacties |
| reactions | Norwegian | reaksjoner |
| reactions | Nynorsk | reaksjonar |
| Reaction | fallback | Reaction |

*continues*

| key | language | translation |
|---|---|---|
| Reaction | English | Reaction |
| Reaction | German | Reaktion |
| Reaction | Italian | Reazione |
| Reaction | French | R\'{e}action |
| Reaction | Dutch | Reactie |
| Reaction | Norwegian | Reaksjon |
| Reaction | Nynorsk | Reaksjon |
| Reactions | fallback | Reactions |
| Reactions | English | Reactions |
| Reactions | German | Reaktionen |
| Reactions | Italian | Reazioni |
| Reactions | French | R\'{e}actions |
| Reactions | Dutch | Reacties |
| Reactions | Norwegian | Reaksjoner |
| Reactions | Nynorsk | Reaksjonar |
| scheme-name | fallback | Scheme |
| scheme-name | English | Scheme |
| scheme-name | German | Schema |
| scheme-name | Norwegian | Skjema |
| scheme-name | Nynorsk | Skjema |
| scheme-list | fallback | List of Schemes |
| scheme-list | English | List of Schemes |
| scheme-list | German | Verzeichnis der Schemata |
| scheme-list | Norwegian | Skjemaliste |
| scheme-list | Nynorsk | Skjemaliste |
| scheme | fallback | scheme |
| scheme | English | scheme |
| scheme | German | Schema |
| scheme | Norwegian | skjema |
| scheme | Nynorsk | skjema |
| Scheme | fallback | Scheme |
| Scheme | English | Scheme |
| Scheme | German | Schema |
| Scheme | Norwegian | Skjema |
| Scheme | Nynorsk | Skjema |
| schemes | fallback | schemes |
| schemes | English | schemes |
| schemes | German | Schemata |
| schemes | Norwegian | skjema |
| schemes | Nynorsk | skjema |
| Schemes | fallback | Schemes |
| Schemes | English | Schemes |
| Schemes | German | Schemata |
| Schemes | Norwegian | Skjema |
| Schemes | Nynorsk | Skjema |
| solution | fallback | solution in |
| solution | English | solution in |

*continues*

| key | language | translation |
|---|---|---|
| solution | German | LÃűsung in |
| phase-aqi | fallback | aq,$\infty $ |
| phase-cd | fallback | cd |
| phase-lc | fallback | lc |

### 26.3. Information For Module Writers

In addition to the commands from section 26.2 starting on page 61 the following macros are available:

* `\chemmacros_translate:n` {⟨*translation key*⟩}
Translates the given key to the language which is detected automatically or given by the user. Should be used in CHEMMACROS' macros instead of translations' `\GetTranslation`.

`\l_chemmacros_language_tl`
A token list variable that holds the language which is used by `\chemmacros_translate:n` for translation, *after begin document*.

`\ChemTranslate`{⟨*translation key*⟩}
A version of `\chemmacros_translate:n` for those who prefer traditional LaTeX $2_\varepsilon$ programming over expl3.

`\chemmacros_declare_translation:nnn`{⟨*language*⟩}{⟨*key*⟩}{⟨*translation*⟩}
The expl3 version of `\DeclareChemTranslation`.

`\chemmacros_declare_translations:nn`{⟨*key*⟩}{⟨*language*⟩ = ⟨*translation*⟩}
The expl3 version of `\DeclareChemTranslations`.

## 27. The tikz Module

The tikz module loads the tikz package [Tan19] and the Ti*k*Z library `calc`.

### 27.1. For Users

The tikz module defines a few arrow tips:

`el`
An arrow tip: `\tikz\draw[-el](0,0)--(1,0);` 

`left el`
An arrow tip: `\tikz\draw[-left el](0,0)--(1,0);` 

`right el`
An arrow tip: `\tikz\draw[-right el](0,0)--(1,0);` 

Introduced in version 5.3

The tikz module also loads the libraries `calc` and `decorations.pathmorphing`. It uses those libraries for defining a new decoration `wave`.

```
\begin{tikzpicture}
  \draw[decorate,decoration=wave]
  (0,0) -- (2,0) ;
\end{tikzpicture}
```



### 27.2. For Module Writers

The tikz module provides some macros for common Ti*k*Z functions. This allows to use expl3's powerful function variants for expansion control.

\c_chemmacros_other_colon_tl
A constant tokenlist which contains a colon with category code 12 (other). This is useful since Ti*k*Z sometimes expects an other colon and in an expl3 programming environment : has category code 11 (letter).

\chemmacros_tikz_picture:nn {⟨*options*⟩} {⟨*code*⟩}
Defined as \tikzpicture[{#1}] #2 \endtikzpicture.

\chemmacros_tikz:nn {⟨*options*⟩} {⟨*code*⟩}
Defined as \tikz[{#1}]{#2}.

\chemmacros_tikz_draw:n {⟨*options*⟩}
Defined as \draw[{#1}].

\chemmacros_tikz_node:n {⟨*options*⟩}
Defined as \node[{#1}].

\chemmacros_tikz_shade:n {⟨*options*⟩}
Defined as \shade[{#1}].

\chemmacros_tikz_shadedraw:n {⟨*options*⟩}
Defined as \shadedraw[{#1}].

\chemmacros_tikz_node_in_draw:n {⟨*options*⟩}
Defined as node[{#1}].

## 28. The xfrac Module

The xfrac module loads the package xfrac [L3P]. For the following explanations it will be helpful if you know about said package and how it works first. This module is a support module that defines the macro

\chemfrac[⟨*type*⟩]{⟨*numerator*⟩}{⟨*denominator*⟩}
⟨*type*⟩ can either be text or superscript.

This macro calls a certain instance of the xfrac text template, depending on the option ⟨*type*⟩ and the current font family. If used \chemfrac looks if an instance

chemmacros-frac-\f@family-⟨*type*⟩

Table 5: Predefined xfrac text instances.

| font family | text | superscript |
|---|---|---|
| cmr | 2/3 | 2/3 |
| lmr | 2/3 | 2/3 |
| LinuxLibertineT-TLF | 2/3 | 2/3 |
| LinuxLibertineT-TOsF | 2/3 | 2/3 |

exists. If yes this instance is used, if no the instance chemmacros-frac-default-⟨*type*⟩ is used. The default instances are the same as the ones for cmr.

The xfrac module defines instances some font families, they are listed and demonstrated in table 5. The superscript type fractions *look* larger than the text types. The reason is that the superscript types are typically used with a smaller font size. Let's take a look at an example where both instances are used:

```
\chemsetup[redox]{pos=top}
\code{superscript}:
\ch{"\ox{1/3,I}" {}3+}

\chemsetup[redox]{pos=side}
\code{text}: \ox{1/3,I3+}

\huge
\chemsetup[redox]{pos=top}
\code{superscript}:
\ch{"\ox{1/3,I}" {}3+}

\chemsetup[redox]{pos=side}
\code{text}: \ox{1/3,I3+}
```

superscript: $\overset{1/3}{I_3^+}$
text: $I_3^+$ (1/3)

**superscript:** $\overset{1/3}{I_3^+}$
**text:** $I_3^+$ (1/3)

If you define instances for other families please feel free to submit them to me (see section A.2 starting on page 68) so they can be added to the xfrac module.

# Part V.
# Appendix

## A. Own Modules

### A.1. How To

If you have additional functionality which you think might be useful as a CHEMMACROS module then you can easily write one yourself. The module must be a file in a path where TEX can find it following a certain naming scheme. The file for a module foo *must be named* chemmacros.module.foo.code.tex.

\ChemModule*{⟨*name*⟩}{⟨*description*⟩}[⟨*minimal compatibility version*⟩]
Register module ⟨*name*⟩. The optional argument ⟨*minimal compatibility version*⟩ ensures that

this module is only loaded if the option `compatibility` has a high enough version number. If it is omitted the module can be loaded in each version 5.0 or higher.

The first line in the file then should look similar to this:

```
\ChemModule{foo}{2015/07/14 description of foo}
```

This registers module `foo` which means CHEMMACROS will accept this file as a valid module.

Since CHEMMACROS is written using expl3 `\ChemModule` starts an expl3 programming environment. If you don't want that but rather want to write your module using traditional LaTeX 2ε methods then use the starred variant:

```
\ChemModule*{foo}{2015/07/14 description of foo}
```

In both variants `@` has category code 11 (letter).

Since new modules very likely might rely on code provided first in a certain version of CHEMMACROS you might want to make sure that your module only is loaded when the compatibility mode is high enough to provide the features you want:

```
\ChemModule{foo}{2015/10/14 description of foo}[5.2]
```

If you decide to write your module `foo` using expl3 and add options you want to be able to set using `\chemsetup[foo]{`⟨*options*⟩`}` please make sure you define (and set) them with the following macros:

`\chemmacros_define_keys:nn {`⟨*module*⟩`} {`⟨*key definitions*⟩`}`
Define l3keys options for the module ⟨*module*⟩. This is a wrapper for `\keys_define:nn {chemmacros/`⟨*module*⟩`} {`⟨*key definitions*⟩`}`.

`\chemmacros_set_keys:nn {`⟨*module*⟩`} {`⟨*input*⟩`}`
Sets l3keys options for the module ⟨*module*⟩. This is a wrapper for `\keys_set:nn {chemmacros/`⟨*module*⟩`}` `{`⟨*input*⟩`}`.

Also (*especially if you consider submitting the module, see section A.2 on the following page*) please follow the expl3 naming conventions for variables and functions, *i. e.*, use `chemmacros` as expl3 module name:

```
\tl_new:N \l__chemmacros_my_internal_variable_tl
\tl_new:N \l_chemmacros_my_public_variable_tl
\cs_new:Npn \__chemmacros_my_internal_function:n #1 { ... }
\cs_new_protected:Npn \chemmacros_my_public_function:n #1 { ... }
\NewDocumentCommand \publicfunction {m}
  { \chemmacros_my_public_function:n {#1} }
```

You will find more details on the naming conventions in `interface3.pdf` which most likely is available on your system:

```
~ $ texdoc interface3
```

If you haven't read section 24 starting on page 59 about the `base` module, yet, please have a look. There some macros for module writers are described. Also other modules define macros for module writers which may be useful.

### A.2. Submitting a Module

If you have written a module and feel it might be useful for other users please feel free to contact me and submit the module. I will surely take at look at both functionality and code and if I feel that it adds value to CHEMMACROS I will add it to the package. Requirement for this is that the module is licensed with the LaTeX Project Public License (v1.3 or later) and that I take over maintenance (according to the "maintainer" status of the LPPL).

Please do *not* submit your module via pull request but send me the files directly. In the best case you also have a short piece of documentation.

## B. Support, Suggestions and Bug Reports

### B.1. Support

If you need support or help with anything regarding CHEMMACROS please use the usual support forums

- `http://www.golatex.de/` or
- `http://texwelt.de/wissen/` if you speak German,
- `http://www.latex-community.org/forum/` or
- `http://tex.stackexchange.com/` if you speak English

You can also open an issue on `https://github.com/cgnieder/chemmacros/issues/` possibly adding the label *support*.

### B.2. Suggestions

If you have any suggestions on how CHEMMACROS could be improved then please go to `https://github.com/cgnieder/chemmacros/issues/` and open a new issue possibly adding the label *suggestion*.

### B.3. Bug reports

If you find any bugs, *i. e.*, errors (something not working as described, conflicts with other packages, …) then please go to `https://github.com/cgnieder/chemmacros/issues/` and open a new issue describing the error including a minimal working example and possibly adding the label *bug*.

# C. References


[Ars13] Donald Arseneau. relsize. version 4.1, Mar. 29, 2013 (or newer).
URL: https://www.ctan.org/pkg/relsize.

[Bra19] Johannes Braams, current maintainer: Javier Bezos.
babel. version 3.33, July 19, 2019 (or newer).
URL: https://www.ctan.org/pkg/babel/.

[Car19] David Carlisle. longtable. version 4.12, Feb. 6, 2019 (or newer).
URL: https://www.ctan.org/pkg/longtable/.

[Cha19] François Charette, current maintainer: Arthur Reutenauer.
polyglossia. version 1.44, Apr. 4, 2019 (or newer).
URL: https://www.ctan.org/pkg/polyglossia/.

[CM19] David Carlisle and Frank Mittelbach.
bm. version 1.2d, July 24, 2019 (or newer). URL: https://www.ctan.org/pkg/bm/.

[Coh+08] E. Richard Cohan et al.
*"Quantities, Symbols and Units in Physical Chemistry", IUPAC Green Book.*
3rd Edition. 2nd Printing. IUPAC & RSC Publishing, Cambridge, 2008.

[Con+05] Neil G. Connelly et al. *"Nomenclature of Inorganic Chemistry", IUPAC Red Book.*
IUPAC & RSC Publishing, Cambridge, 2005. ISBN: 0-85404-438-8.

[Fuj13] Shinsaku Fujita. XǍMTEX. version 5.06, 2013 (or newer).
URL: https://www.ctan.org/pkg/xymtex/.

[Hen18] Martin Hensel. mhchem. version 4.08, June 22, 2018 (or newer).
URL: https://www.ctan.org/pkg/mhchem/.

[Koh19] Markus Kohm. KOMA-Script. version 3.25, Jan. 14, 2019 (or newer).
URL: https://www.ctan.org/pkg/koma-script/.

[L3P] The LaTeX3 Project Team. l3packages. Oct. 27, 2020 (or newer).
URL: https://www.ctan.org/pkg/l3packages/.

[Leh19] Philipp Lehman, current maintainer: Joseph Wright.
etoolbox. version 2.5h, Sept. 21, 2019 (or newer).
URL: https://www.ctan.org/pkg/etoolbox/.

[MRW19] Lars Madsen, Will Robertson, and Joseph Wright.
mathtools. version 1.22, July 31, 2019 (or newer).
URL: https://www.ctan.org/pkg/mh/.

[MSoo] Frank Mittelbach and Rainer Schöpf.
amstext. version 2.01, June 29, 2000 (or newer).
URL: https://www.ctan.org/pkg/amstext/.

[Nie15] Clemens Niederberger. elements. version 0.1, June 14, 2015 (or newer).
URL: https://www.ctan.org/pkg/elements/.

[Nie16a] Clemens Niederberger. chemgreek. version 1.1, Dec. 20, 2016 (or newer).
URL: https://www.ctan.org/pkg/chemgreek/.

[Nie16b] Clemens Niederberger. chemnum. version 1.2, Apr. 14, 2016 (or newer).
URL: https://www.ctan.org/pkg/chemnum/.

[Nie19] Clemens Niederberger. chemformula. version 4.15f, Sept. 23, 2019 (or newer).
URL: https://www.ctan.org/pkg/chemformula/.

[Nie20a] Clemens Niederberger. acro. version 3.4, Dec. 25, 2020 (or newer).
URL: https://www.ctan.org/pkg/acro/.

[Nie20b] Clemens Niederberger. translations. version 1.8, Feb. 28, 2020 (or newer).
URL: https://www.ctan.org/pkg/translations/.

[ORT20] Heiko Oberdiek, Sebastian Rahtz, and The LaTeX3 Project Team.
hyperref. version 7.00d, Jan. 14, 2020 (or newer).
URL: https://www.ctan.org/pkg/hyperref/.

[Ped17] Bjørn Pedersen. bpchem. version 1.1, Aug. 23, 2017 (or newer).
URL: https://www.ctan.org/pkg/bpchem/.

[PPR04] R. Panico, W. H. Powell, and J-C. Richer. *"Nomenclature of Organic Chemistry, Sections A, B, C, D, E, F, and H", IUPAC Blue Book.* draft. Oct. 7, 2004.
URL: http://old.iupac.org/reports/provisional/abstract04/BB-prs310305/CompleteDraft.pdf (visited on 07/07/2013).

[Tal20] Nicola L. C. Talbot. glossaries-extra. version 1.45, Apr. 1, 2020 (or newer).
URL: https://ctan.org/pkg/glossaries-extra/.

[Tan19] Till Tantau. TikZ/pgf. version 3.1.4b, Aug. 3, 2019 (or newer).
URL: https://www.ctan.org/pkg/pgf/.

[Tel19] Christian Tellechea. chemfig. version 1.41, May 21, 2019 (or newer).
URL: https://www.ctan.org/pkg/chemfig/.

[VN20] Herbert Voss and Rolf Niepraschk.
xltabular. version 0.2e, Nov. 4, 2020 (or newer).
URL: https://ctan.org/pkg/xltabular/.

[Wri13] Joseph Wright. chemstyle. version 2.0m, July 3, 2013 (or newer).
URL: https://www.ctan.org/pkg/chemstyle/.

[Wri18] Joseph Wright. siunitx. version 2.7s, May 17, 2018 (or newer).
URL: https://www.ctan.org/pkg/siunitx/.

# D. Index

**D**



**E**



**F**