

CHEMMACROS

v6.0 2022/01/16

comprehensive support for typesetting chemistry documents

Clemens NIEDERBERGER¹ Sonja K.²

<https://github.com/cgnieder/chemmacros>

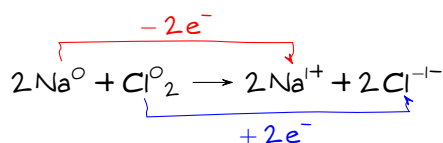


Table of Contents

I. Preliminaries	3	6.2. Macros Defined (Not Only) For Usage in <code>\iupac</code>	12
1. License	3	6.3. One-letter Macros	12
2. Motivation and Background	3	6.4. Greek Letters	13
3. The Structure of CHEMMACROS	3	6.5. Hetero Atoms and added Hydrogen	13
3.1. General Structure	3	6.6. Cahn-Ingold-Prelog	14
3.2. CHEMMACROS ' Options . . .	4	6.7. Fischer	14
3.3. Support Package CHEMFORMULA	5	6.8. cis/trans, zusammen/entgegen, syn/anti & tert	15
3.4. Upgrading from version 5.x . .	5	6.9. ortho/meta/para	15
		6.10. Absolute Configuration	15
		6.11. Coordination Chemistry . . .	15
		6.12. Examples	16
II. Main Modules	6	6.13. Own <code>\iupac</code> Macros And Shorthands	16
4. The acid-base Module	6	6.14. Latin Phrases	17
5. The charges Module	8	7. The particles Module	18
5.1. Charge Symbols	8	7.1. Provided Particle Macros . . .	18
5.2. Ion Charges	8	7.2. Defining Own Particle Macros	19
5.3. Partial Charges and Similar Stuff	9	8. The phases Module	20
5.4. Charge Options	9	8.1. Basics	20
5.5. Own Charge Macros	9	8.2. Define Own Phases	21
		8.3. Language Dependencies . . .	21
6. The nomenclature Module	10	9. The symbols Module	22
6.1. The <code>\iupac</code> Command	10		

1. contact@mychemistry.eu

2. SonjaK@mein.gmx

10. The chemformula Module	22	21. The spectroscopy Module	49
10.1. For Users	22	21.1. The \NMR Command	49
10.2. Using the chemformula Package	23	21.2. Short Cuts	50
10.3. Using the mhchem Package .	23	21.3. An Environment to Typeset Experimental Data	50
10.4. Using the chemfig Package . .	23	21.4. Customization	51
10.5. Using the chemist Package . .	24	21.5. An Example	53
10.6. For Module Writers	24	21.6. Nearly Standard	54
11. The greek Module	24	21.7. Formatted List	55
		21.8. Crazy	55
III. Additional Modules	25	22. The thermodynamics Module	56
12. The isotopes Module	25	22.1. The \state Macro	56
13. The mechanisms Module	26	22.2. Thermodynamic Variables . .	57
		22.3. Create New Variables or Re- define Existing Ones	58
14. The newman Module	27	23. The units Module	60
15. The orbital Module	28		
16. The polymers Module	30	IV. Core Modules	61
16.1. Nomenclature	30	24. The base Module	61
16.2. Copolymers	30	25. The errorcheck Module	63
16.3. Non-linear (Co) Polymers and Polymer Assemblies	31	26. The lang Module	63
16.4. Polymer Denotations in chemfig's Molecules	31	26.1. Information For Users	63
17. The reactions Module	32	26.2. Available Translation Keys . .	64
17.1. Predefined Environments . . .	32	26.3. Information For Module Writers	66
17.2. Own Reactions	34	27. The tikz Module	67
17.3. List of Reactions	35	27.1. For Users	67
18. The reactants Module	37	27.2. For Module Writers	67
18.1. Idea and Getting Started . . .	37	28. The xfrac Module	68
18.2. Basic Commands	37		
18.3. Options	39	V. Appendix	69
18.3.1. Data and Units	39	A. Own Modules	69
18.3.2. Output Styles	41	A.1. How To	69
18.4. Use in Section Headings . . .	42	A.2. Submitting a Module	70
18.5. Acronyms as Reactant/Sol- vent Names	43	B. Support, Suggestions and Bug Reports	71
18.6. List of Reactants	43	B.1. Support	71
19. The redox Module	44	B.2. Suggestions	71
19.1. Oxidation Numbers	44	B.3. Bug reports	71
19.2. Redox Reactions	46	C. References	71
19.3. Examples	47		
20. The scheme Module	48		

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 package³ 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 ϵ .
- 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 71.
- 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 71. Don't forget to add references for the corresponding IUPAC recommendation.

3. The Structure of **CHEMMACROS**

3.1. General Structure

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

Introduced
in version 5.0
(2015/09/11)

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

3. The Structure of CHEMMACROS

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:

```
1 % first possibility:
2 \chemsetup[module]{
3   option1 = value ,
```

3. The Structure of **CHEMMACROS**

```
4 option2 = value
5 }
6 % second possibility:
7 \chemsetup{
8   module/option1 = value ,
9   module/option2 = value
10 }
```

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:

```
1 \chemsetup{
2   option1 = value ,
3   option2 = value
4 }
```

Some internal modules may also define core options, e. g., the `lang` module, see section 26 starting on page 63.

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 have 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 option `minimal` allows to load **CHEMMACROS** with smallest subset necessary. Then additional modules can be added with the `modules`.
- 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, pK_a ...

\pH
pH

\pOH
pOH

\Ka
 K_a , depends on language settings, see section 26 starting on page 63. The translations can be adapted.

\Kb
 K_b

\Kw
 K_w

\pKa[*num*]
\pKa: pK_a , **\pKa**[1]: pK_{a1} , depends on language settings, see section 26 starting on page 63. The translations can be adapted.

\pKb[*num*]
\pKb: pK_b , **\pKb**[1]: pK_{b1}

\p{*anything*}
e. g. **\p**{**\Kw**} pK_w

$\text{\pKa}\text{\pKb}\text{\pKa}\text{\pKa}[1]\text{\pKb}\text{\pKb}$	$K_a K_b pK_a pK_{a1} pK_b pK_{b1}$
---	-------------------------------------

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*
Set the style of the p operator.

Default: upright

acid-base » **K-acid** = {*text*}
The subscript to **\Ka** and **\pKa**.

Default: **\ChemTranslate**{K-acid}

4. The acid-base Module

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.

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

$\text{pH}, \text{p}K_{\text{a}}$
 $\textit{pH}, \textit{p}K_{\text{a}}$
 $\textit{pH}, \textit{p}K_{\text{a}}$

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:

```
1 % this works only in the preamble:
2 % \DeclareTranslation{English}{K-acid}{\mathrm{A}}% use your language here
3 % alternative:
4 \chemsetup{acid-base/K-acid=\mathrm{A}}% overwrites language dependent
   settings
5 \pKa
```

$\text{p}K_{\text{A}}$

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 63 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

`\scrip`

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

`\scrm`

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

`\fscrip`

⊕ 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).

`\fsscrip`

⊕ 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


```
1 A\pch\ B\mch[3] C\fpch[2] D\fmch      A+ B3- C2⊕ D⊖
```

5.3. Partial Charges and Similar Stuff

The next ones probably are seldomly needed but nevertheless useful:

`\delp`

δ^+ partial positive charge

`\delm`

δ^- partial negative charge

`\fdelp`

δ^\oplus partial formal positive charge

`\fdelm`

δ^\ominus partial formal negative charge

These macros for example can be used with the `\ox` command (see section 19 starting on page 44) or with the chemfig package:

```
1 \chemsetup{
2   charges/circled = all,
3   redox/parse     = false,
4   redox/pos       = top
5 }
6 \ch{"\ox{\delp,H}" -{\} "\ox{\delm,Cl}" } \hspace*{1cm}
7 \chemfig{\chemabove[3pt]{\charge{90=\|,180=\|,270=\|}{Br}}{\delm}-\chemabove
   [3pt]{H}{\delp}}
```



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 (\oplus/\ominus) 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` \oplus /`\fminus` \ominus (chem) and `\oplus` \oplus /`\ominus` \ominus (math).

`charges` » `partial-format` = { $\langle\text{L}^{\text{A}}\text{T}_{\text{E}}\text{X}\text{ code}\rangle$ } 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:

```
1 \NewChemCharge\fpch{\fplus}
2 \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:

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

6. The nomenclature Module

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

6.1. The `\iupac` Command

Similar to the bpchem package [Ped17] `CHEMMACROS` provides a command⁴ 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].

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

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

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 `]`.
- The character `'` is printed as `\chemprime`.
- The character `=` is printed as `\nonbreakinghyphen`.

```
1 \huge\iupac{2,4-Di|chlor|pentan} \par
2 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
<code>\L</code>	L	L	L
<code>\iupac{\L}</code>	L	L	L
<code>\D</code>	D	—	D
<code>\iupac{\D}</code>	D	D	D

`nomenclature` » `hyphen-post-space = {\langle dim \rangle}` Default: -.03em
 Set the space that is inserted after the hyphen set with -.

`nomenclature` » `break-space = {\langle dim \rangle}` 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` `L`) 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	<code>\a</code>	<code>\b</code>	<code>\g</code>	<code>\d</code>	<code>\k</code>	<code>\m</code>	<code>\n</code>	<code>\w</code>
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.

```

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

5 α -androstan-3 β -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**.

1	\iupac {\nitrogen-methyl benz amide}	<i>N</i> -methylbenzamide
2		
3	\iupac {3\hydrogen-pyrrole}	3 <i>H</i> -pyrrole
4		
5	\iupac {\oxygen-ethyl hexanethioate}	<i>O</i> -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: **\upshape**

Changed in
version 6.0
(2022/01/16)

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⟩}` μ_3^-
 Denote bridging ligand connection.

`\hapto{⟨num⟩}` η^5-
 Denote hapticity.

`\dento{⟨num⟩}` κ^2-
 Denote denticity.

```
1 Ferrocene = \iupac{bis(\hapto{5}cyclopentadienyl)iron} \par
2 \iupac{tetra-\bridge{3}iodido-tetrakis[trimethylplatinum(IV)]}
```

Ferrocene = bis(η^5 -cyclopentadienyl)iron
 tetra- μ_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

```

1 \iupac{\dexter-Wein|s"aure} =
2 \iupac{\cip{2S,3S}-Wein|s"aure} \par
3 \iupac{\dexter-($-)-Threose} =
4 \iupac{\cip{2S,3R}-(($-)-2,3,4-Tri|hydroxy|butanal} \par
5 \iupac{\cis-2-Butene} =
6 \iupac{\zusammen-2-Butene}, \par
7 \iupac{\cip{2E,4Z}-Hexa|diene} \par
8 \iupac{\meta-Xylol} =
9 \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.

```
1 \NewChemIUPAC\endo{\textsc{endo}}
2 \RenewChemIUPAC\anti{\textsc{anti}}
3 \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.

```
1 \iupac{\meta-Xylol} \par m-Xylol
2 \RenewChemIUPAC\meta{\textup{m}} m-Xylol
3 \iupac{\meta-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*

7. The particles Module

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.

```
1 \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`.

```
1 \ba[elpair=dots] \Nuc[elpair=dash]      ba:⁻ NuI⁻
2
3 \chemsetup[particles]{elpair=false}    ba⁻ Nu⁻
4 \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:

```
1 \NewChemParticle\el {e-}
2 \NewChemParticle\prt {p+}
3 \NewChemParticle\ntn {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:

```
1 \NewChemNucleophile\Nuc{Nu}
2 \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)

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

$\text{C(s)} + 2 \text{H}_2\text{O(l)} \longrightarrow \text{CO}_2\text{(g)} + 2 \text{H}_2\text{(g)}$
To make it complete: 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 $\text{T}_{\text{E}}\text{X}$ dimension.

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

$\text{C}_{\text{(s)}} + 2 \text{H}_2\text{O}_{\text{(l)}} \longrightarrow \text{CO}_{2\text{(g)}} + 2 \text{H}_{2\text{(g)}}$
To make it complete: $\text{NaCl}_{\text{(aq)}}$.

All those phase commands have an optional argument:

```
1 \ch{H2O "\lqd[\qty{5}{\celsius}]}      H2O(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.

```
1 % preamble:
2 \NewChemPhase\aqi{aq,$\infty$} % aqueous solution at infinite dilution
3 \NewChemPhase\cd {cd} % condensed phase
4 \NewChemPhase\lc {lc} % liquid crystal
5 \ch{NaOH\aqi} \ch{H2O\cd} \ch{U\phase{cr}} \ch{A\lc}\par
6 \chemsetup[phases]{pos=sub}
7 \ch{NaOH\aqi} \ch{H2O\cd} \ch{U\phase{cr}} \ch{A\lc}
```

NaOH(aq, ∞) H₂O(cd) U(cr) A(lc)

NaOH_(aq, ∞) H₂O_(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

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

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

```
1 \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 [Niezob] 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 63 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: \ddagger

`\standardstate`

Again self-explaining: \ominus

`\changestate`

The uppercase delta used in ΔH 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.
- \XTeX [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 = {\langle method \rangle}`

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
- chemist (from the \XTeX bundle)
- chemfig

Introduced
in version 5.6
(2016/05/02)
Introduced in
version 5.6

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 L^AT_EX 2_ε 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 L^AT_EX 2_ε 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

```
1 \usepackage{upgreek}
2 \usepackage{chemmacros}
```

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

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


```

1 \usepackage{upgreek}
2 \usepackage{chemmacros}
3 \usepackage{textgreek}

```

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

```

1 \usepackage{upgreek}
2 \usepackage{chemmacros}
3 \usepackage{textgreek}
4 \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.

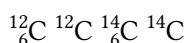
<input> can also be comma separated list: **\isotope**{*<nuc>*,*<symbol>*}. If you leave *<nuc>* out then **\isotope** will display the most common isotope. Otherwise *<nuc>* will be used. If *<nuc>* is an isotope unknown to the elements package **\isotope** will write a warning to the log file.

The starred variant omits the element number.

```

1 \isotope{C}
2 \isotope*{C}
3 \isotope{14,C}
4 \isotope*{14,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}`.

```

1 \isotope{C}
2 \chemsetup{isotopes}{format=side}
3 \isotope{C}
4 \chemsetup{isotopes}{side-connect=}
5 \isotope{C}

```

¹²₆C 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 E_{1cb}

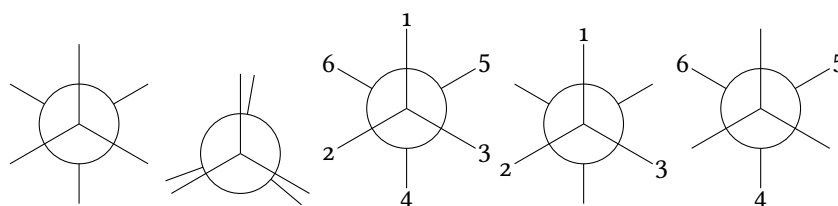
14. The newman Module

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 TikZ internally. $\langle angle \rangle$ rotates the back atoms counter clockwise with respect to the front atoms and is an optional argument. $\langle 1 \rangle$ to $\langle 6 \rangle$ are the positions, the first three are the front atoms, the last three the back atoms.

```
1 \newman{} \newman(170){}
2 \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.

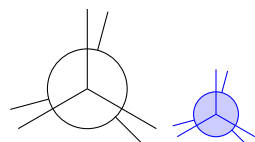
`newman` » `scale = {⟨factor⟩}` Default: 1
Scale the whole projection by factor $\langle factor \rangle$.

`newman` » `ring = {⟨tikz⟩}` (initially empty)
Customize the ring with TikZ keys.

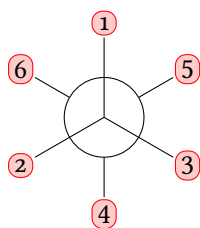
`newman` » `atoms = {⟨tikz⟩}` (initially empty)
Customize the nodes within which the atoms are set with TikZ keys.

`newman` » `back-atoms = {⟨tikz⟩}` (initially empty)
Explicitly customize the nodes of the back atoms with TikZ keys.

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



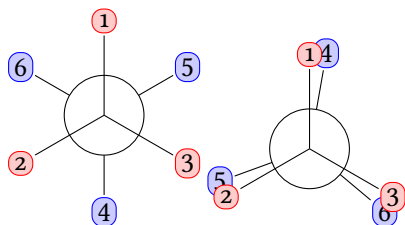
```
1 \chemsetup[newman]{atoms={draw=red,fill=red!20,inner sep=2pt,rounded corners}}
2 \newman{1,2,3,4,5,6}
```



```

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

```

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

```



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

- | | |
|--|----------------|
| <code>orbital</code> » <code>phase = + -</code> | Default: + |
| changes the phase of the orbital (all types) | |
| <code>orbital</code> » <code>scale = {⟨factor⟩}</code> | Default: 1 |
| changes the size of the orbital (all types) | |
| <code>orbital</code> » <code>color = {⟨color⟩}</code> | Default: black |
| changes the color of the orbital (all types) | |

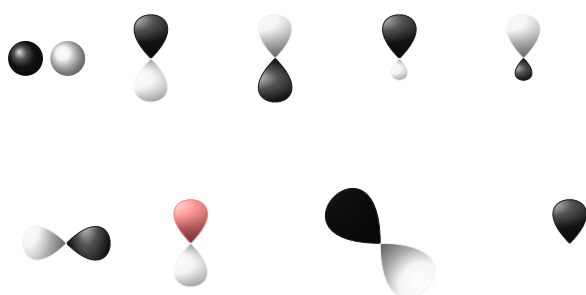
`orbital` » `angle = {⟨angle⟩}` Default: 0
 rotates the orbitals with a p contribution counter clockwise (all types except s)

`orbital` » `half = true|false` Default: false
 displays only half an orbital (only p)

```

1 \orbital{s} \orbital[phase=-]{s}
2 \orbital{p} \orbital[phase=-]{p}
3 \orbital{sp3} \orbital[phase=-]{sp3}
4
5 \orbital[angle=0]{p} \orbital[color=red!50]{p}
6 \orbital[angle=135,scale=1.5]{p} \orbital[half]{p}

```



Additionally there are two options, with which the TikZ behaviour can be changed.

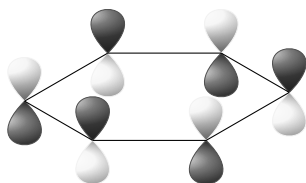
`orbital` » `overlay = true|false`
 The orbital “doesn’t need space”; it is displayed with the TikZ option `overlay`.

`orbital` » `opacity = {⟨num⟩}`
 The orbital becomes transparent; `⟨value⟩` can have values between 1 (fully opaque) to 0 (invisible).

```

1 \vspace{7mm}
2 \chemsetup{orbital}{
3   overlay,
4   p/color = black!70
5 }
6 \setchemfig{bond offset=0pt}
7 \chemfig{
8   ?\orbital{p}
9   -[,1.3]{\orbital[phase=-]{p}}
10  -[:30,1.1]\orbital{p}
11  -[:150,.9]{\orbital[phase=-]{p}}
12  -[4,1.3]\orbital{p}
13  -[: -150,1.1]{\orbital[phase=-]{p}}?
14 }
15 \vspace{7mm}

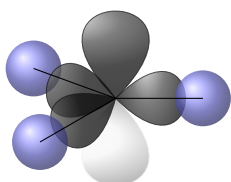
```



```

1 \vspace{7mm}
2 \setchemfig{bond offset = 0pt}
3 \chemsetup[orbital]{
4   overlay ,
5   opacity = .75 ,
6   p/scale = 1.6 ,
7   s/color = blue!50 ,
8   s/scale = 1.6
9 }
10 \chemfig{
11   \orbital{s}
12   -[: -20]{\orbital[scale=2]{p}}
13           {\orbital[half,angle=0]{p}}
14           {\orbital[angle=170,half]{p}}
15           {\orbital[angle=-150,half]{p}}
16   (-[: -150]\orbital{s})-\orbital{s}
17 }
18 \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 TikZ' nodes where the delimites are placed.

`polymers` » `delimiters = {\langle left \rangle \langle right \rangle}` 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.

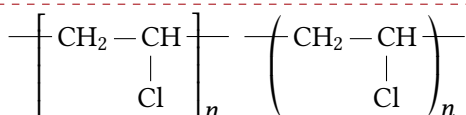
`polymers` » `subscript = {\langle subscript \rangle}` Default: \$n\$
 Subscript to the right delimiter.

`polymers` » `superscript = {\langle superscript \rangle}`
 Superscript to the right delimiter.

```

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


```

1 Reaction with counter:
2 \begin{reaction}
3   A -> B
4 \end{reaction}

```

Reaction with counter:

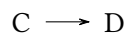


```

1 Reaction without counter:
2 \begin{reaction*}
3   C -> D
4 \end{reaction*}

```

Reaction without counter:



```

1 Several aligned reactions with counter:
2 \begin{reactions}
3   A      &-> B + C \\
4   D + E &-> F
5 \end{reactions}

```

Several aligned reactions with counter:

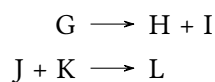


```

1 Several aligned reactions without counter:
2 \begin{reactions*}
3   G      &-> H + I \\
4   J + K &-> L
5 \end{reactions*}

```

Several aligned reactions without counter:



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>}

Introduced in
version 5.6

The left delimiter.

Default: {

`reactions` » `tag-close` = $\{\langle right\ delimiter\rangle\}$

Default: }

Introduced in version 5.6 The right delimiter.

`reactions` » `before-tag` = $\{\langle format\rangle\}$

(initially empty)

Introduced in version 5.6 Code inserted before the tags.

```

1 \chemsetup[reactions]{
2   before-tag = R \textbf ,
3   tag-open = [ ,
4   tag-close = ]
5 }
6 \begin{reaction}
7   H2O + CO2 <=> H2CO3
8 \end{reaction}

```



The use of $\mathcal{A}\mathcal{M}\mathcal{S}$ math's `\intertext` is possible:

```

1 \begin{reactions}
2   A + 2 B &-> 3 C + D "\label{rxn:test}"
3   \intertext{Some text in between aligned reactions}
4   3 E + F &<=> G + 1/2 H
5 \end{reactions}
6 See reaction~\ref{rxn:test}.

```



Some text in between aligned reactions



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` $\{\langle name\rangle\}[\langle number\ of\ arguments\rangle][\langle math\ name\rangle]$

$\langle name\rangle$ will be the name of the new chem environment. $\langle math\ name\rangle$ is the underlying math environment. Gives an error if $\langle name\rangle$ already exists.

`\RenewChemReaction` $\{\langle name\rangle\}[\langle number\ of\ arguments\rangle][\langle math\ name\rangle]$

$\langle name\rangle$ is the name of the renewed chem environment. $\langle math\ name\rangle$ is the underlying math environment. Gives an error if $\langle name\rangle$ 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.

```
1 \NewChemReaction{reaction} {equation}
2 \NewChemReaction{reaction*} {equation*}
3 \NewChemReaction{reactions} {align}
4 \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:

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

With this the reactionsat environment is defined.

```
1 \NewChemReaction{reactionsat}[1]{alignat}
2 \NewChemReaction{reactionsat*}[1]{alignat*}
3 \begin{reactionsat}{3}
4   A      &-> B      &&-> C      &&-> D \\\
5   aaaaa &-> bbbbbb &&-> ccccc &&-> ddddd
6 \end{reactionsat}
7 \begin{reactionsat*}{2}
8   A      &-> B      & C      &-> D \\\
9   aaaaa &-> bbbbbb &\quad{} & ccccc &-> ddddd
10 \end{reactionsat*}
```

$$\begin{array}{ccccccc} A & \longrightarrow & B & & \longrightarrow & C & & \longrightarrow & D & \{7\} \\ aaaaa & \longrightarrow & bbbbbb & & \longrightarrow & ccccc & & \longrightarrow & ddddd & \{8\} \end{array}$$

$$\begin{array}{ccccccc} A & \longrightarrow & B & & C & \longrightarrow & D \\ aaaaa & \longrightarrow & bbbbbb & & ccccc & \longrightarrow & ddddd \end{array}$$

17.3. List of Reactions

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

`\listoreactions`

Print a list of reactions.

```
1 \listofreactions
```

List of Reactions

Reaction {1}	33
Reaction {2}	33
Reaction {3}	33
Reaction [R 4]	34
Reaction {5}	34
Reaction {6}	34
Reaction {7}	35
Reaction {8}	35
Reaction {9}: Autoprotolyse	36
Reaction {10}: first step of chain	37
Reaction {11}: second step of chain	37

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 63.

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 63.

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.

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



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

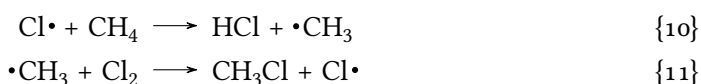
`\AddRxnDesc{<description>}`

Add a description to a reaction.

```

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

```



18. The reactants Module

☆ New

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 and will be explicitly mentioned in the corresponding sections of this manual.

18.2. Basic Commands

`\DeclareChemReactant{<ID>}{<properties>}`

This command defines the reactant `<ID>` with the properties `<properties>`. *Should be used in the preamble. Must be used in the preamble when `acro` is used as acronym support package.*

`\DeclareChemReactant{<main ID>.<sub ID>}{<properties>}`

Analogously to `chemnum`'s `\compd` command, both the `\DeclareChemReactant` and `\reactant` commands accept a combined `<ID>` consisting of a `<main ID>` and `<sub ID>` part. The default separator is a `.` here, but this can be changed using `chemnum`'s `main-sub-sep = {token}` option.

Valid `<properties>` include the following key-value pairs:

`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, see section 18.5 starting on page 43.

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, see section 18.4 starting on page 42.

upper-name = {<upper case version of the name>}

Optional property: The upper case version of a compound's name, e.g. for the use in the beginning of a sentence.

upper-bookmark = {<upper case version of the bookmark text>}

Optional property: The upper case version of the <name> in a PDF bookmark.

Common declarations will look like this:

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

\reactant[<data and units>]{<ID>}

This command is used to insert name, number, and, if present, data of a predefined reactant with the <ID> in the text. The order of the information in the output can be controlled through the **reactant-output-style** option, see section 18.3.2 starting on page 41. The upper case version of this command **\Reactant** can be used in order to start a sentence with an upper case version of a compound's name. The corresponding text must be defined through **\DeclareChemReactant**'s **upper-name** option. Further variants of **\reactant** 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. Format and order depend on the **solvent-output-style** option. The upper case version of this command **\Solvent** can be used in order to start a sentence with an upper case version of a solvent's name. The corresponding text must be defined through **\DeclareChemReactant**'s **upper-name** option. s and l suffixed variants 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 starting on page 40.

Typical uses will look like this:

```
1 \reactant{dichloropentane}
2 \reactant[volume=5]{dichloropentane}
3 \reactant[volume=0.5, volume-unit=L]{dichloropentane}
4 \solvent{thf}
5 \solvent[volume=200]{thf}
6 \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 $\langle ID \rangle$ in the list of reactants. Using `printreactants-style` different styles can be selected. (See section 18.6 starting on page 43).

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 a 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` commands offer an optional argument that can be used to easily input this data:

`\reactant` [*$\langle data and units \rangle$*] { $\langle ID \rangle$ }

`\solvent` [*$\langle data and units \rangle$*] { $\langle ID \rangle$ }

$\langle data and units \rangle$ accepts a comma separated list of key-value pairs with the available keys and their default units/values listed in table 3 on the following page. Key-value pairs can be input in any order as they are categorized and rearranged internally according to the order in which they are listed in table 3 on the next page. Customization of this order is thus far somewhat limited. The available customization possibilities are described in section 18.3.2 starting on page 41. 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 surround a number with a set of `{}` if you use a comma as input decimal separator. Otherwise the decimal places will be truncated without a warning.

`solution` here refers to the text that links concentration and solvent. This text automatically adapts to the document language set via `babel` or `polyglossia`. Currently, the English fallback, as well as the German translation are included in the package. If you write

in a different language (or just don't like the predefined text), you can use the command `\DeclareChemTranslation{<key>}{<language>}{<translation>}` (with `<key> = {solution}`) as described in section 26 starting on page 63 in order to supply your own translation.

```

1 % in the preamble:
2 % \DeclareChemReactant{nBuLi}{name={\iupac{\textit{n}}=butyllithium}}}
3 % \DeclareChemReactant{Br2benzene}{name={\iupac{1,4=di|bromo|benzene}}}
4 % \DeclareChemReactant{HBr}{name={\ch{HBr\aq}}}
5
6 \reactant[volume=5.00, amount=12.5, equiv=1.00, concentration=2.5, solvent=
   hexane]{nBuLi}\par
7 \reactant[mass=3.9, amount=15.6, equiv=1.3, purity=95]{Br2benzene}\par
8 \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`. Accepted units are units defined by the `siunitx` package or by the `units` module.

`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 » fraction-unit = {<unit>}` Default: w/w `\percent`

Change the unit of the fraction property.

`reactants » amount-unit = {<unit>}` Default: `\milli\mole`

Change the unit of the amount property.

`reactants » equiv-unit = {<unit>}` Default: `eq`

Change the unit of the equiv property.

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	<code>mass-unit</code>
volume	mL	<code>volume-unit</code>
fraction	w/w%	<code>fraction-unit</code>
amount	mmol	<code>amount-unit</code>
equiv	eq	<code>equiv-unit</code>
purity	%	<code>purity-unit</code>
concentration	M	<code>concentration-unit</code>
solvent	n.a.	
solution-name	solution in	<code>solution</code>

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

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

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

tetrahydrofuran **4** (5.5 mL)
 tetrahydrofuran **4** (5 cm³)

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 following predefined output styles:

`reactants` » `reactant-output-style = name-main-other|main-name-other|main-other-name` Default: `name-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 `solvent-output-style` choice options refer to the order in which the contents of the categories are typeset.

```
1 \chemsetup[reactants]{reactant-output-style=name-main-other}
2 \reactant[volume=5, amount=4]{dichloropentane}\par
3 \chemsetup[reactants]{reactant-output-style=main-name-other}
4 \reactant[volume=5, amount=4]{dichloropentane}\par
5 \chemsetup[reactants]{reactant-output-style=main-other-name}
6 \reactant[volume=5, amount=4]{dichloropentane}
7
8 \chemsetup[reactants]{solvent-output-style=name-main}
9 \solvent[volume=5]{thf}\par
10 \chemsetup[reactants]{solvent-output-style=main-name}
11 \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`

By default, only mass and volume are assigned to the main category. Using the `main` option, `equiv` or `amount` can be added to the main category.

```
1 \chemsetup[reactants]{main=amount}
2 \reactant[equiv=2.0, amount=5]{dichloropentane}\par
3 \chemsetup[reactants]{main=equiv}
4 \reactant[equiv=2.0, amount=5]{dichloropentane}\par
5 \chemsetup[reactants]{main=default}
6 \reactant[equiv=2.0, amount=5]{dichloropentane}
```

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

Using the `\reactants` command inside of section headings or captions can mess up the order in which the molecules are numbered, especially when also using a table of contents and/or a list of figures/tables. To prevent this, the `reactants` module offers the + suffixed variant of `\reactants`, comparably to `chemnum`'s `\cmpd+` command.

`\reactant+[\langle data and units \rangle]{\langle ID \rangle}`

This command is used to insert name, number, and, if present, data of a predefined reactant with the `\langle ID \rangle` in a section heading or caption.

If you also use the `hyperref` package in combination with PDF bookmarks, you might want to use the optional `bookmark` property of `\DeclareChemReactant` to supply an alternative text to `name` to be displayed inside of the PDF bookmarks. To later use such a predefined solvent or reactant, use one of the following three commands, that are defined analogously to `chemnum`'s `\cmpdplain`. All three commands also exist in the upper case variant (`\Reactantplain`, `\Sumbainreactantplain` and `\Solventplain`) which can be used to display the upper case version of a reactant or solvent's name. The upper case version of the name must be declared previously through `\DeclareChemReactant`'s `upper-name` and `upper-bookmark` options.

`\reactantplain{\langle ID \rangle}`

Outputs the value of `bookmark` inside of the PDF bookmark, while using the reactant's `name` inside of the section headings.

`\submainreactantplain{\langle mainID \rangle}{\langle subID \rangle}`

Outputs the value of `bookmark` inside of the PDF bookmark, while using the reactant's `name` inside of the section headings. Must be used if your `\langle ID \rangle` consists of a `\langle mainID \rangle` and a `\langle subID \rangle` part.

`\solventplain{⟨ID⟩}`

Outputs the value of `bookmark` inside of the PDF bookmark, while using the solvent's `name` inside of the section headings.

18.5. Acronyms as Reactant/Solvent Names

In order to integrate solvent/reactant acronyms into one combined list of acronyms, the reactants module offers two different options. While using either of these two options, the user can also explicitly decide if the `name` or the `short` version of the reactant/solvent should be used in the text. Inspired by the `\acs` and `\acl` commands from the `acro` or the `glossaries-extra` package, the `reactants` module also offers the following `s` and `l` suffixed variants:

`\reactants{⟨ID⟩}`

Output the `short` version of the reactant's name.

`\reactantl{⟨ID⟩}`

Output the `name` version of the reactant's name.

`\solvents{⟨ID⟩}`

Output the `short` version of the solvent's name.

`\solventl{⟨ID⟩}`

Output the `name` version of the solvent's name.

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

Default: none

Can be used to select, which of the two packages `acro` or `glossaries-extra` is used in the background in order to format and sort acronyms.

```
1 % in the preamble:
2 % \DeclareChemReactant{dcm}{name={dichloromethane}, short={DCM}}
3 \solvent{dcm}\par
4 \solventl{dcm}\par
5 \solvents{dcm}
```

```
dichloromethane
DICHLOROMETHANE
DCM
```

18.6. List of Reactants

As mentioned before, `\printreactants` can be used to print a list of all used reactants and their numbers. The `reactants` module internally uses either `longtable` or `xltabular` to typeset this list:

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

Default: none

Can be used to switch between `longtable` and `xltabular` which are responsible for formatting the list of reactants. Be aware that with `longtable`, the column widths are hard coded, thus you could experience overfull box warnings if you use exceptionally long `⟨ID⟩`s in combination with the starred variant `\printreactants*`, which is responsible for adding the `⟨ID⟩` in resulting list, as well.

19. The redox Module

The redox module loads the modules tikz and xfrac. It also loads the packages math-tools [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), $\text{O}^{-\text{II}}$, 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).

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

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

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 $\text{X}^{1.2}$.

`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-fraction** = {<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-fraction** = {<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.

1 <code>\ox[roman=false]{2,Ca} \ox{2,Ca} \</code>	Ca^2 Ca^{II}
2 <code>\ox[pos=top]{3,Fe}-Oxide \</code>	Fe^{III} -Oxide
3 <code>\ox[pos=side]{3,Fe}-Oxide \</code>	Fe (III)-Oxide
4 <code>\ox[parse=false]{?,Mn} \</code>	$\text{Mn}^?$
5 <code>\ox[pos=top,align=right]{2,Ca}</code>	Ca^{II}

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

1 <code>\ox{3,Fe} \ox*{3,Fe}</code>	Fe^{III} Fe^{III}
-------------------------------------	---

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

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

Na^{+1} , Ca^{+2} , S^{-2} , F_2^0

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

Compare $\text{O}_2^{-1 2-}$ to $\text{O}_2^{-1 2-}$

Sometimes one might want to use formal oxidation numbers like 0.5 or $\frac{1}{3}$:

1 <code>\chemsetup[redox]{pos=top}</code>	
2 <code>\ox{.5,Br2}</code>	
3 <code>\ch{"\ox{1/3,I}" {3}+}</code>	$\text{Br}_2^{\text{0.5}}$ $\text{I}_3^{+\frac{1}{3}}$
4	$\text{I}_3^{+(\frac{1}{3})}$
5 <code>\chemsetup[redox]{pos=side}</code>	
6 <code>\ox{1/3,I3+}</code>	

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 68.

19.2. Redox Reactions

CHEMMACROS provides two commands to visualize the transfer of electrons in redox reactions. Both commands are using TikZ.

`\OX{⟨name⟩,⟨atom⟩}`

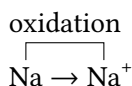
Label $\langle atom \rangle$ with the label $\langle name \rangle$.

`\redox(⟨name1⟩,⟨name2⟩)[⟨tikz⟩][⟨num⟩]{⟨text⟩}`

Connect two $\langle atom \rangle$ s previously labelled with `\OX`. Only the first argument ($\langle name1 \rangle, \langle name2 \rangle$) is required, the others are all optional.

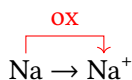
`\OX` places $\langle atom \rangle$ into a node, which is named with $\langle name \rangle$. If you have set two `\OX`, they can be connected with a line using `\redox`. To do so the names of the two nodes that are to be connected are written in the round braces. Since `\redox` draws a tikzpicture with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

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



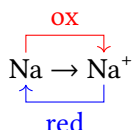
This line can be customized using TikZ keys in `[⟨tikz⟩]`:

```
1 \vspace{7mm}
2 \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 $\langle num \rangle$. If you use a negative value the line is placed *below* the text.

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



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

`redox » dist = {⟨dim⟩}`

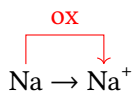
Default: `.6em`

A TeX dimension.

```

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

```



`redox » sep = {<dim>}`

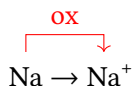
Default: .2em

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

```

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

```

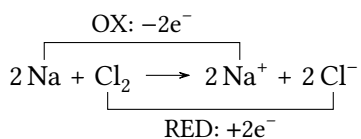


19.3. Examples

```

1 \vspace{7mm}
2 \ch{
3   2 "\OX{o1,Na}" + "\OX{r1,Cl}" {}2
4   ->
5   2 "\OX{o2,Na}" {}+ + 2 "\OX{r2,Cl}" {}-
6 }
7 \redox(o1,o2){\small OX: $- 2\text{el}$}
8 \redox(r1,r2)[][-1]{\small RED: $+ 2\text{el}$}
9 \vspace{7mm}

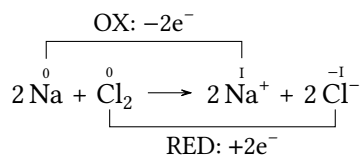
```



```

1 \vspace{7mm}
2 \ch{
3   2 "\OX{o1,\ox*{0,Na}}" + "\OX{r1,\ox*{0,Cl}}" {}2
4   ->
5   2 "\OX{o2,\ox*{+1,Na}}" {}+ + 2 "\OX{r2,\ox*{-1,Cl}}" {}-
6 }
7 \redox(o1,o2){\small OX: $- 2\text{el}$}
8 \redox(r1,r2)[][-1]{\small RED: $+ 2\text{el}$}
9 \vspace{7mm}

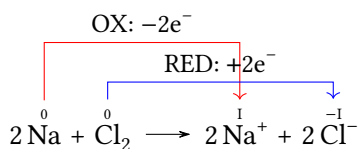
```



```

1 \vspace{14mm}
2 \ch{
3   2 "\OX{o1,\ox*{0,Na}}" + "\OX{r1,\ox*{0,Cl}}" {}2
4   ->
5   2 "\OX{o2,\ox*{+1,Na}}" {}+ + 2 "\OX{r2,\ox*{-1,Cl}}" {}-
6 }
7 \redox(o1,o2)[draw=red,->][3.33]{\small OX: $- 2\text{e}l$}
8 \redox(r1,r2)[draw=blue,->]{\small RED: $+ 2\text{e}l$}

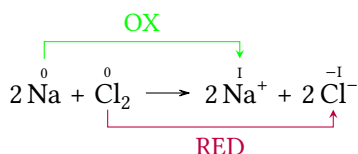
```



```

1 \vspace{7mm}
2 \ch{
3   2 "\OX{o1,\ox*{0,Na}}" + "\OX{r1,\ox*{0,Cl}}" {}2
4   -> 2 "\OX{o2,\ox*{+1,Na}}" {}+ + 2 "\OX{r2,\ox*{-1,Cl}}" {}-
5 }
6 \redox(o1,o2)[green,-stealth]{\small OX}
7 \redox(r1,r2)[purple,-stealth][-1]{\small RED}
8 \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 63 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*`

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

The first argument specifies the kind of NMR:

1 `\NMR{13,C}`

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

The second argument sets the frequency (in MHz):

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

^1H -NMR (400 MHz): δ

You can choose another unit:

```
1 \NMR(4e8,\hertz)
```

^1H -NMR (4×10^8 Hz): δ

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

```
1 \sisetup{exponent-product=\cdot}
2 \NMR(4e8,\hertz)
```

^1H -NMR ($4 \cdot 10^8$ Hz): δ

The third argument specifies the solvent:

```
1 \NMR[CDCl3]
```

^1H -NMR (CDCl_3): δ

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 resonance data. Gives an error if <cs> already exists.

`\DeclareChemNMR{<cs>}{<num>,<atom>}`

Define a new shortcut macro for typesetting a certain type of magnetic resonance data. Overwrites an existing macro.

`\RenewChemNMR{<cs>}{<num>,<atom>}`

Redefine an existing shortcut macro for typesetting a certain type of magnetic resonance 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 resonance 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>}:

```
1 \NewChemNMR\HNMR{1,H}%
2 \NewChemNMR\CNMR{13,C}%
3 \CNMR*(100) \par
4 \HNMR*(400)
```

^{13}C -NMR (100 MHz)

^1H -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]}` $^{13}\text{C}\{^1\text{H}\}$ -NMR: δ

`\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>}`.

```
1 \begin{experimental}
2   \data{type1} Data.
3   \data{type2}[specifications] More data.
4   \data*{type3} Even more data.
5 \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>}`

The used default unit.

Default: `\mega\hertz`

`spectroscopy` » `nucleus = {<num>,<atom>}`

The used default nucleus.

Default: `{1,H}`

`spectroscopy` » `connector = {<code>}`

Places `<code>` between the nucleus and the method.

Default: -

<code>spectroscopy</code> » <code>method = {⟨code⟩}</code>	Default: NMR
The measuring method.	
<code>spectroscopy</code> » <code>format = {⟨commands⟩}</code>	(initially empty)
For example <code>\bfseries</code> .	
<code>spectroscopy</code> » <code>nmr-base-format = {⟨commands⟩}</code>	(initially empty)
Introduced in version 5.8	Formatting instructions for the NMR base.
<code>spectroscopy</code> » <code>pos-number = side sub super</code>	Default: side
Position of the number next to the atom.	
<code>spectroscopy</code> » <code>coupling-symbol = {⟨code⟩}</code>	Default: J
The symbol used for the coupling constant.	
<code>spectroscopy</code> » <code>coupling-unit = {⟨unit⟩}</code>	Default: <code>\hertz</code>
A siunitx unit.	
<code>spectroscopy</code> » <code>coupling-pos = side sub</code>	Default: side
Placement of the coupling nuclei next to the symbol <i>J</i> (or rather the symbol specified with option <code>coupling-symbol</code>).	
<code>spectroscopy</code> » <code>coupling-nuclei-pre = {⟨code⟩}</code>	Default: (
Code inserted before the coupling nuclei when <code>coupling-pos = {side}</code> .	
<code>spectroscopy</code> » <code>coupling-nuclei-post = {⟨code⟩}</code>	Default:)
Code inserted after the coupling nuclei when <code>coupling-pos = {side}</code> .	
<code>spectroscopy</code> » <code>coupling-bonds-pre = {⟨code⟩}</code>	(initially empty)
Code inserted before the coupling bonds.	
<code>spectroscopy</code> » <code>coupling-bonds-post = {⟨code⟩}</code>	Default: \!
Code inserted after the coupling bonds.	
<code>spectroscopy</code> » <code>coupling-pos-cs = {⟨cs⟩}</code>	Default: <code>\@firstofone</code>
Set the macro that prints the number set with the <code>\pos</code> macro. This needs to be a command with one mandatory argument.	
<code>spectroscopy</code> » <code>atom-number-cs = {⟨cs⟩}</code>	Default: <code>\@firstofone</code>
Set the macro that prints the number set with the <code>\#</code> macro. This needs to be a command with one mandatory argument.	
<code>spectroscopy</code> » <code>atom-number-space = {⟨dim⟩}</code>	Default: <code>.16667em</code>
Introduced in version 5.3	Horizontal space inserted between number and atom (printed by <code>\#</code>).
<code>spectroscopy</code> » <code>parse = true false</code>	Default: true
Treat the solvent as <code>CHEMFORMULA</code> formula (this depends on the setting of the <code>formula</code> option, see 10 starting on page 22) or not.	
<code>spectroscopy</code> » <code>delta = {⟨tokens⟩}</code>	(initially empty)
The <code>⟨tokens⟩</code> are added after δ .	
<code>spectroscopy</code> » <code>list = true false</code>	Default: false
The environment <code>experimental</code> 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 equal sign after `\NMR` and `\data`.

The default setup of the list:

```
1 \topsep\z@skip \partopsep\z@skip
2 \itemsep\z@ \parsep\z@ \itemindent\z@
3 \leftmargin\z@
```

```
1 \begin{experimental}[format=\bfseries]
2   \data{type1} Data.
3   \data{type2}[specifications] More data.
4   \data*{type3} Even more data.
5 \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.

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

type1 = Data. **type2 (specifications)** = More data. ¹H-NMR: δ = 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`.

```
1 \sisetup{separate-uncertainty,per-mode=symbol,detect-all,range-phrase=- -}
2 \begin{experimental}[<optionen>]
3   \data*{yield} \qty{17}{\milli\gram} yellow needles (\qty{0.04}{\milli\mole}
4     \qty{13}{\percent}).
5   %
6   \data{mp.} \qty{277}{\celsius} (DSC).
7   %
8   \NMR(600)[CDCl3] \val{2.01} (s, \#{24}, \pos{5}), \val{2.31} (s, \#{12},
9     \pos{1}), \val{6.72--6.74} (m, \#{2}, \pos{11}), \val{6.82} (s, \#{8},
10    \pos{3}), \val{7.05--7.07} (m, \#{2}, \pos{12}), \val{7.39--7.41} (m,
    \#{4},
```

```

11 \pos{9}), \val{7.48--7.49} (m, \#{4}, \pos{8}).
12 %
13 \NMR{13,C}(150)[CDCl3] \val{21.2} ($+$, \#{4}, \pos{1}), \val{23.4} ($+$,
14 \#{8}, \pos{5}), \val{126.0} ($+$, \#{4}, \pos{9}), \val{128.2} ($+$,
15 \#{8},
16 \pos{3}), \val{130.8} ($+$, \#{2}, \pos{12}), \val{133.6} ($+$, \#{2},
17 \pos{11}), \val{137.0} ($+$, \#{4}, \pos{8}), \val{138.6} (q, \#{4},
18 \pos{2}), \val{140.6} (q, \#{2}, \pos{10}), \val{140.8} (q, \#{8}, \pos{4})
19 ,
20 \val{141.8} (q, \#{4}, \pos{6}), \val{145.6} (q, \#{2}, \pos{7}).
21 %
22 \data{MS}[DCP, EI, \qty{60}{\electronvolt}] \val{703} (2, \ch{M+}), \val
23 {582}
24 (1), \val{462} (1), \val{249} (13), \val{120} (41), \val{105} (100).
25 %
26 \data{MS}[\ch{MeOH + H2O + KI}, ESI, \qty{10}{\electronvolt}] \val{720}
27 (100,
28 \ch{M+ + OH-}), \val{368} (\ch{M+ + 2 OH-}).
29 %
30 \data{IR}[KBr] \val{3443} (w), \val{3061} (w), \val{2957} (m), \val{2918}
31 (m), \val{2856} (w), \val{2729} (w), \val{1725} (w), \val{1606} (s),
32 \val{1592} (s), \val{1545} (w), \val{1446} (m), \val{1421} (m), \val{1402}
33 (m), \val{1357} (w), \val{1278} (w), \val{1238} (s), \val{1214} (s),
34 \val{1172} (s), \val{1154} (m), \val{1101} (w), \val{1030} (w), \val{979}
35 (m), \val{874} (m), \val{846} (s), \val{818} (w), \val{798} (m), \val{744}
36 (w), \val{724} (m), \val{663} (w), \val{586} (w), \val{562} (w), \val{515}
37 (w).
38 %
39 \data*{UV-Vis} \qty{386}{\nano\metre} ($\varepsilon = \val{65984}$),
40 \qty{406}{\nano\metre} ($\varepsilon = \val{65378}$).
41 %
42 \data*{quantum yield} $\Phi = \val{0.74+-0.1}$\,,.
43 \end{experimental}

```

21.6. Nearly Standard

Output with these options:

```
\delta=ppm, \pos=number, \use-equal
```

yield: 17 mg yellow needles (0.04 mmol, 13 %). mp. = 277 °C (DSC). ¹H-NMR (600 MHz, CDCl₃): δ (ppm) = 2.01 (s, 24 H, H₅), 2.31 (s, 12 H, H₁), 6.72–6.74 (m, 2 H, H₁₁), 6.82 (s, 8 H, H₃), 7.05–7.07 (m, 2 H, H₁₂), 7.39–7.41 (m, 4 H, H₉), 7.48–7.49 (m, 4 H, H₈). ¹³C-NMR (150 MHz, CDCl₃): δ (ppm) = 21.2 (+, 4 C, C₁), 23.4 (+, 8 C, C₅), 126.0 (+, 4 C, C₉), 128.2 (+, 8 C, C₃), 130.8 (+, 2 C, C₁₂), 133.6 (+, 2 C, C₁₁), 137.0 (+, 4 C, C₈), 138.6 (q, 4 C, C₂), 140.6 (q, 2 C, C₁₀), 140.8 (q, 8 C, C₄), 141.8 (q, 4 C, C₆), 145.6 (q, 2 C, C₇). MS (DCP, EI, 60 eV) = 703 (2, M⁺), 582 (1), 462 (1), 249 (13), 120 (41), 105 (100). MS (MeOH + H₂O + 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 (ε = 65 984), 406 nm (ε = 65 378). quantum yield: Φ = 0.74 ± 0.10.

21.7. Formatted List

Output with these options:

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

yield: 17 mg yellow needles (0.04 mmol, 13 %).**mp.** = 277 °C (DSC).**¹H-NMR (600 MHz, CDCl₃):** δ (ppm) = 2.01 (s, 24 H, H₅), 2.31 (s, 12 H, H₁), 6.72–6.74 (m, 2 H, H₁₁), 6.82 (s, 8 H, H₃), 7.05–7.07 (m, 2 H, H₁₂), 7.39–7.41 (m, 4 H, H₉), 7.48–7.49 (m, 4 H, H₈).**¹³C-NMR (150 MHz, CDCl₃):** δ (ppm) = 21.2 (+, 4 C, C₁), 23.4 (+, 8 C, C₅), 126.0 (+, 4 C, C₉), 128.2 (+, 8 C, C₃), 130.8 (+, 2 C, C₁₂), 133.6 (+, 2 C, C₁₁), 137.0 (+, 4 C, C₈), 138.6 (q, 4 C, C₂), 140.6 (q, 2 C, C₁₀), 140.8 (q, 8 C, C₄), 141.8 (q, 4 C, C₆), 145.6 (q, 2 C, C₇).**MS (DCP, EI, 60 eV)** = 703 (2, M⁺), 582 (1), 462 (1), 249 (13), 120 (41), 105 (100).**MS (MeOH + H₂O + 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 (ϵ = 65 984), 406 nm (ϵ = 65 378).**quantum yield:** Φ = 0.74 ± 0.10 .**21.8. Crazy**

Output for these options:

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

yield: 17 mg yellow needles (0.04 mmol, 13 %).**mp.** = 277 °C (DSC).**¹H-NMR (600 MHz, CDCl₃):** δ M⁺ + H₂O = 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).**¹³C-NMR (150 MHz, CDCl₃):** δ M⁺ + H₂O = 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 + H₂O + 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 ($\epsilon = 65\,984$), 406 nm ($\epsilon = 65\,378$).

quantum yield: $\Phi = 0.74 \pm 0.10$.

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.

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

ΔA° , $\Delta_f G^\circ$, $\Delta E_{\text{Na}}^\circ$, $\Delta H^{1000^\circ\text{C}}$

These options are available:

<code>thermodynamics » pre = {⟨text⟩}</code>	Default: <code>\changestate</code>
Code inserted before the variable. Inserted in text mode.	
<code>thermodynamics » post = {⟨text⟩}</code>	(initially empty)
Code inserted after the variable. Inserted in text mode.	
<code>thermodynamics » superscript-left = {⟨text⟩}</code>	(initially empty)
The left superscript. Inserted in text mode.	
<code>thermodynamics » superscript-right = {⟨text⟩}</code>	Default: <code>\standardstate</code>
The right superscript. Inserted in text mode.	
<code>thermodynamics » superscript = {⟨text⟩}</code>	
An alias of <code>superscript-right</code> .	
<code>thermodynamics » subscript-left = {⟨text⟩}</code>	(initially empty)
The left subscript. Inserted in text mode.	
<code>thermodynamics » subscript-right = {⟨text⟩}</code>	(initially empty)
The right subscript. Inserted in text mode.	
<code>thermodynamics » subscript = {⟨text⟩}</code>	
An alias of <code>subscript-left</code> .	

22.2. Thermodynamic Variables

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

`\enthalpy*[\langle options \rangle](\langle subscript \rangle)\{\langle value \rangle\}`
Typeset the amount of enthalpy.

`\entropy*[\langle options \rangle](\langle subscript \rangle)\{\langle value \rangle\}`
Typeset the amount of entropy.

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

Their usage is pretty much self-explaining:

1 <code>\enthalpy{123} \par</code>	$\Delta H^\circ = 123 \text{ kJ mol}^{-1}$
2 <code>\entropy{123} \par</code>	$S^\circ = 123 \text{ J K}^{-1} \text{ mol}^{-1}$
3 <code>\gibbs{123}</code>	$\Delta G^\circ = 123 \text{ kJ mol}^{-1}$

The argument `(\langle subscript \rangle)` adds a subscript for specification, `*` hides number and unit:

1 <code>\enthalpy(r){123} \par</code>	$\Delta_r H^\circ = 123 \text{ kJ mol}^{-1}$
2 <code>\enthalpy*{123} \par</code>	ΔH°

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

`thermodynamics » post = {\langle text \rangle}` (initially empty)
Code inserted after the variable. Inserted in text mode.

`thermodynamics » superscript-left = {\langle text \rangle}` (initially empty)
The left superscript. Inserted in text mode.

`thermodynamics » superscript-right = {\langle text \rangle}` Default: `\standardstate`
The right superscript. Inserted in text mode.

`thermodynamics » superscript = {\langle text \rangle}`
An alias of `superscript-right`.

`thermodynamics » subscript-left = {\langle text \rangle}` (initially empty)
The left subscript. Inserted in text mode.

`thermodynamics » subscript-right = {\langle text \rangle}` (initially empty)
The right subscript. Inserted in text mode.

`thermodynamics » subscript = {\langle text \rangle}`
An alias of `subscript-left`.

`thermodynamics » subscript-pos = left|right` Default: `left`
Determines whether the subscript given in `(\langle subscript \rangle)` is placed to the left or the right of the variable.

thermodynamics » **symbol** = { $\langle symbol \rangle$ } (initially empty)
 The symbol of the variable. Inserted in math mode.

thermodynamics » **unit** = { $\langle unit \rangle$ } (initially empty)
 A valid siunitx unit.

The default values depend on the command.

```

1 \enthalpy[unit=\kilo\joule]{-285} \
  par                                 $\Delta H^\circ = -285 \text{ kJ}$ 
2 \gibbs[pre=]{0} \par               $G^\circ = 0 \text{ kJ mol}^{-1}$ 
3 \entropy[pre=\Delta$,superscript   $\Delta S = 56.7 \text{ J K}^{-1} \text{ mol}^{-1}$ 
  =]{56.7}
```

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

```

1 \enthalpy{-1234.56e3} \par
2 \sisetup{
3   per-mode=symbol,
4   exponent-product=\cdot,           $\Delta H^\circ = -1234.56 \times 10^3 \text{ kJ mol}^{-1}$ 
5   output-decimal-marker={,},       $\Delta H^\circ = -1\,234,56 \cdot 10^3 \text{ kJ/mol}$ 
6   group-four-digits=true
7 }
8 \enthalpy{-1234.56e3}
```

22.3. Create New Variables or Redefine Existing Ones

\NewChemState{ $\langle cs \rangle$ }{ $\langle options \rangle$ }

Define new state commands like **\enthalpy**. Gives an error if $\langle cs \rangle$ already exists.

\RenewChemState{ $\langle cs \rangle$ }{ $\langle options \rangle$ }

Redefine existing state commands.

\DeclareChemState{ $\langle cs \rangle$ }{ $\langle options \rangle$ }

Like **\NewChemState** but gives now error if $\langle cs \rangle$ already exists.

\ProvideChemState{ $\langle cs \rangle$ }{ $\langle options \rangle$ }

Define new state commands like **\enthalpy**. Defines $\langle cs \rangle$ only if it is not defined, yet.

The argument $\langle options \rangle$ is a comma separated list of key/value options:

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

thermodynamics » **post** = { $\langle text \rangle$ } (initially empty)
 Code inserted after the variable. Inserted in text mode.

thermodynamics » **superscript-left** = { $\langle text \rangle$ } (initially empty)
 The left superscript. Inserted in text mode.

thermodynamics » **superscript-right** = { $\langle text \rangle$ } Default: **\standardstate**
 The right superscript.

- thermodynamics » `superscript = {\langle text \rangle}`
An alias of `superscript-right`.
- thermodynamics » `subscript-left = {\langle text \rangle}` (initially empty)
The left subscript. Inserted in text mode.
- thermodynamics » `subscript-right = {\langle text \rangle}` (initially empty)
The right subscript. Inserted in text mode.
- thermodynamics » `subscript = {\langle text \rangle}`
An alias of `subscript-left`.
- thermodynamics » `subscript-pos = left|right` Default: left
Determines whether the subscript given in ($\langle subscript \rangle$) is placed to the left or the right of the variable.
- thermodynamics » `symbol = {\langle symbol \rangle}` (initially empty)
The symbol of the variable.
- thermodynamics » `unit = {\langle unit \rangle}` (initially empty)
A valid siunitx unit.

```

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

```

$\Delta A^\ominus = 123.4 \text{ kJ mol}^{-1}$
 $\Delta E = -1.1 \text{ V}$
 $\Delta E^\ominus_{\text{Sn}|\text{Sn}^{2+}||\text{Pb}^{2+}|\text{Pb}} = 0.01 \text{ V}$
 $\Delta_f h^\ominus = 12.5 \text{ J}$

The existing commands have been defined like this:

```

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

```

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

```

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

```

$$\Delta h = -12.3 \text{ kJ mol}^{-1} \quad \Delta H = -12.3 \text{ kJ}$$

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 `\qty` and `\unit` which represents `<unit>`.

to add arbitrary units. `CHEMMACROS` uses that command to provide some units. Like all siunitx units they're only valid inside `\qty{<num>}{<unit>}` and `\unit{<unit>}`.

`\atmosphere`
atm

`\atm`
atm

`\calory`
cal

`\cal`
cal

`\cmc`
cm³

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

`\molar`
mol dm⁻³

`\moLar`
mol L⁻¹

`\Molar`
M

`\MolMass`
g mol⁻¹

`\normal`
N

`\torr`
Torr

☆ New

Since some units still frequently used in chemistry were removed from version 3 of siunitx, `CHEMMACROS` also defines these in the same way, older versions of siunitx used to do. These units are:

`\angstrom`

Å

`\atomicmassunit`

u

`\bar`

bar

`\elementarycharge`

e

`\mmHg`

mmHg

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 69 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` { $\langle true \rangle$ } { $\langle false \rangle$ }

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` { $\langle text \rangle$ }

Checks if the current font weight is bold and if yes places $\langle text \rangle$ in `\textbf` if in text mode or in `\bm` if in math mode. If no $\langle text \rangle$ simply is placed in the input stream as is.

`\chemmacros_text:n` { $\langle text \rangle$ }

Ensures that $\langle text \rangle$ is placed in text mode.

`\chemmacros_math:n` { $\langle text \rangle$ }

Ensures that $\langle text \rangle$ is placed in math mode.

`\chemmacros_new_macroset:nnn` { $\langle name \rangle$ } { $\langle arg spec \rangle$ } { $\langle internal command call \rangle$ }

Changed in
version 5.3b ()

A command to define a set of macros `\NewChem` $\langle name \rangle$, `\RenewChem` $\langle name \rangle$, `\DeclareChem` $\langle name \rangle$ and `\ProvideChem` $\langle name \rangle$ where the first letter of $\langle name \rangle$ is converted to uppercase, other letters are kept unchanged. $\langle arg spec \rangle$ is any valid argument specification for xparse's `\DeclareDocumentCommand` [L3P]. $\langle internal command call \rangle$ 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` { $\langle name \rangle$ } { $\langle arg spec \rangle$ } { $\langle internal command call \rangle$ }

Like `\chemmacros_new_macroset:nnn` but for environments.

`\NewChemMacroset*` { $\langle name \rangle$ } { $\langle arg spec \rangle$ } { $\langle internal command call \rangle$ }

A non-expl3 version of `\chemmacros_new_macroset:nnn` for L^AT_EX 2_ε programmers. The starred version calls `\chemmacros_new_environment_macroset:nnn`.

`\chemmacros_add_cleveref_support:nnnn` { $\langle counter \rangle$ } { $\langle singular \rangle$ } { $\langle plural \rangle$ } { $\langle uppercase singular \rangle$ } { $\langle uppercase plural \rangle$ }

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` { $\langle counter \rangle$ } { $\langle singular \rangle$ } [$\langle uppercase singular \rangle$] { $\langle plural \rangle$ } [$\langle uppercase plural \rangle$]

Introduced in
version 5.6

L^AT_EX 2_ε-version of `\chemmacros_add_cleveref_support:nnnn`.

`\chemmacros_add_fancyref_support:nnn` { $\langle prefix \rangle$ } { $\langle name \rangle$ } { $\langle uppercase name \rangle$ }

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` { $\langle prefix \rangle$ } { $\langle name \rangle$ } [$\langle uppercase name \rangle$]

Introduced in
version 5.6

L^AT_EX 2_ε-version of `\chemmacros_add_fancyref_support:nnnn`.

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

```
1 \NewChemMacroset {Particle} {mm}
2 { \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 on the following page.

26.2. Available Translation Keys

Changed in
version 5.6

Table 4 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 string⁷ 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:

Introduced in
version 5.6

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

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.

Introduced in
version 5.6

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

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 71) 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	a
K-base	fallback	b
K-water	fallback	w
phase-sld	fallback	s
phase-lqd	fallback	l
phase-gas	fallback	g
phase-aq	fallback	aq
K-acid	German	s
K-acid	Danish	v
K-acid	Dutch	z
phase-sld	German	f
phase-lqd	German	f\l
solution	fallback	solution in
solution	English	solution in
solution	German	L"osung in
list-of-reactions	fallback	List of Reactions
list-of-reactions	English	List of Reactions
list-of-reactions	German	Reaktionsverzeichnis

continues

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

key	language	translation
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
list-of-reactions	Danish	Reaktionsliste
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
reaction	Danish	reaktion
reactions	fallback	reactions
reactions	English	reactions
reactions	German	Reaktion
reactions	Italian	reazioni
reactions	French	r\{'e}actions
reactions	Dutch	reacties
reactions	Norwegian	reaksjoner
reactions	Nynorsk	reaksjonar
reactions	Danish	reaktioner
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
Reaction	Danish	Reaktion
Reactions	fallback	Reactions
Reactions	English	Reactions
Reactions	German	Reaktion
Reactions	Italian	Reazioni
Reactions	French	R\{'e}actions
Reactions	Dutch	Reacties
Reactions	Norwegian	Reaksjoner
Reactions	Nynorsk	Reaksjonar
Reactions	Danish	Reaktioner
scheme-name	fallback	Scheme
scheme-name	English	Scheme
scheme-name	German	Schema
scheme-name	Norwegian	Skjema
scheme-name	Nynorsk	Skjema

continues

key	language	translation
scheme-name	Danish	Skema
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-list	Danish	Skemaliste
scheme	fallback	scheme
scheme	English	scheme
scheme	German	Schema
scheme	Norwegian	skjema
scheme	Nynorsk	skjema
scheme	Danish	skema
Scheme	fallback	Scheme
Scheme	English	Scheme
Scheme	German	Schema
Scheme	Norwegian	Skjema
Scheme	Nynorsk	Skjema
Scheme	Danish	Skema
schemes	fallback	schemes
schemes	English	schemes
schemes	German	Schemata
schemes	Norwegian	skjema
schemes	Nynorsk	skjema
schemes	Danish	skemaer
Schemes	fallback	Schemes
Schemes	English	Schemes
Schemes	German	Schemata
Schemes	Norwegian	Skjema
Schemes	Nynorsk	Skjema
Schemes	Danish	Skemaer
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 64 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_ϵ programming over `expl3`.

`\chemmacros_declare_translation:nn{<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 `TikZ` library `calc`.

27.1. For Users

The `tikz` module defines a few arrow tips:

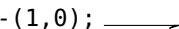
`el`

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`.

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



27.2. For Module Writers

The `tikz` module provides some macros for common `TikZ` 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 `TikZ` 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>]`.

TABLE 5: Predefined xfrac text instances.

font family	text	superscript
cmr	$\frac{2}{3}$	$\frac{2}{3}$
lmr	$\frac{2}{3}$	$\frac{2}{3}$
LinuxLibertineT-TLF	$\frac{2}{3}$	$\frac{2}{3}$
LinuxLibertineT-T0sF	$\frac{2}{3}$	$\frac{2}{3}$

`\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>`

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:

```

1 \chemsetup[redox]{pos=top}
2 \code{superscript}:
3 \ch{"\ox{1/3,I}" {}3+}
4
5 \chemsetup[redox]{pos=side}
6 \code{text}: \ox{1/3,I3+}
7
8 \huge
9 \chemsetup[redox]{pos=top}
10 \code{superscript}:
11 \ch{"\ox{1/3,I}" {}3+}
12
13 \chemsetup[redox]{pos=side}
14 \code{text}: \ox{1/3,I3+}

```

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

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

If you define instances for other families please feel free to submit them to me (see section A.2 on the next page) 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:

```
1 \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 2 ϵ methods then use the starred variant:

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

A. Own Modules

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:

```
1 \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*) please follow the `expl3` naming conventions for variables and functions, *i. e.*, use `chemmacros` as `expl3` module name:

```
1 \tl_new:N \__chemmacros_my_internal_variable_tl
2 \tl_new:N \__chemmacros_my_public_variable_tl
3 \cs_new:Npn \__chemmacros_my_internal_function:n #1 { ... }
4 \cs_new_protected:Npn \chemmacros_my_public_function:n #1 { ... }
5 \NewDocumentCommand \publicfunction {m}
6 { \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 61 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 a 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 L^AT_EX Project Public License (v1.3 or later) and that I take over maintenance (according to the “maintainer” status of the L^AT_EX Project Public License).

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.

C. References

- [Fuj13] Shinsaku FUJITA. \LaTeX . 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 \LaTeX 3 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 MITTELBAACH 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 \LaTeX 3 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/>.

C. References

- [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

Symbols

<i><key></i>	40
' (symbol)	11
((symbol)	11
) (symbol)	11
- (symbol)	11 f.
= (symbol)	11
[(symbol)	11
\#	51 f.
^ (symbol)	11
(symbol)	11 f.
option	40
] (symbol)	11

A

\a	13
\abinitio	17
\acl	43
acro (package)	37, 43
acronym-support	38, 43
\acs	43
\AddRxnDesc	37
align	44
alignat (environment)	35
\alt	31
\alternating	31
amount-unit	40
amsmath (package)	32
amstext (package)	22, 26, 61
angle	27, 29
\angstrom	61
\anti	15, 17
\aq	20 f.
\aqi	21
ARSENEAU, Donald	44
\atm	60
\atmosphere	60
atom-number-cs	52
atom-number-space	52
\atomicmassunit	61
atoms	27

B

\b	13
\ba	18 f.
babel (package)	21, 37, 39, 63
back-atoms	27
\bar	61
base (module)	61, 63, 70
before-tag	34
BEZOS, Javier	63
\blend	31
\block	31
bm (package)	61
\bond	29 f.
bookmark	38, 42 f.
boolean-option	4
bpchem (package)	10

BRAAMS, Johannes	63
\branch	31
break-space	12
\bridge	15 f.
bridge-number	16

C

\c_chemmacros_other_colon_tl (expl3)	67
\cal	60
\calory	60
CARLISLE, David	37, 61
\cd	21
\ce	23 f., 32
\ch	5, 8 f., 18, 20 f., 23 f., 32, 38, 45, 47 f., 54 ff., 59, 69
\changestate	22, 56 ff.
CHARETTE, François	63
charges (module)	8, 18, 22 ff.
\chcpd	5, 18, 23 f., 44, 49
\chemabove	9
\chemAlpha	13
\chemalpha	13
\chembeta	13
\ChemCleverefSupport	62
\Chemdelta	13
\ChemFancyrefSupport	62
chemfig (package)	3, 8 f., 22 f., 31
\ChemForm	24
chemformula	23
chemformula (module)	8, 18, 20, 22 f., 32, 49
chemformula (package)	5, 8, 18, 22 ff.
\chemfrac	45, 68
\chemgamma	13
chemgreek (package)	24
chemist (package)	22, 24
\chemmacros_add_cleveref_support:nnnn (expl3)	62
\chemmacros_add_fancyref_support:nnn (expl3)	62
\chemmacros_add_fancyref_support:nnnn (expl3)	62
\chemmacros_after_module:nn (expl3)	63
\chemmacros_allow_break: (expl3)	61
\chemmacros_before_module:nn (expl3)	63
\chemmacros_bold:n (expl3)	62
\chemmacros_chemformula:n (expl3)	24
\chemmacros_declare_translation:nnn (expl3)	67
\chemmacros_declare_translations:nn (expl3)	67
\chemmacros_define_keys:nn (expl3)	70
\chemmacros_if_bold: (expl3)	62
\chemmacros_if_class_loaded:n (expl3)	61
\chemmacros_if_is_int:n (expl3)	61
\chemmacros_if_loaded:nn (expl3)	61
\chemmacros_if_module_exist:n (expl3)	63
\chemmacros_if_module_loaded:n (expl3)	63
\chemmacros_if_package_loaded:n (expl3)	61
\chemmacros_load_module:n (expl3)	63
\chemmacros_load_modules:n (expl3)	63
\chemmacros_math:n (expl3)	62
\chemmacros_new_environment_macroset:nnn (expl3)	62
\chemmacros_new_macroset:nnn (expl3)	62

INDEX

<code>\chemmacros_nobreak: (expl3)</code>	61	D	
<code>\chemmacros_reaction:n (expl3)</code>	24	<code>\D</code>	12, 14
<code>\chemmacros_set_keys:nn (expl3)</code>	70	<code>\d</code>	13
<code>\chemmacros_skip_nobreak:N (expl3)</code>	61	<code>\data</code>	51, 53 f.
<code>\chemmacros_text:n (expl3)</code>	62	<code>decimal-marker</code>	44
<code>\chemmacros_tikz:nn (expl3)</code>	67	<code>\DeclareChemCharge</code>	10
<code>\chemmacros_tikz_draw:n (expl3)</code>	67	<code>\DeclareChemEqConstant</code>	7
<code>\chemmacros_tikz_node:n (expl3)</code>	68	<code>\DeclareChemIUPAC</code>	16 f.
<code>\chemmacros_tikz_node_in_draw:n (expl3)</code>	68	<code>\DeclareChemIUPACShorthand</code>	17
<code>\chemmacros_tikz_picture:nn (expl3)</code>	67	<code>\DeclareChemLatin</code>	18
<code>\chemmacros_tikz_shade:n (expl3)</code>	68	<code>\DeclareChemNMR</code>	50
<code>\chemmacros_tikz_shadedraw:n (expl3)</code>	68	<code>\DeclareChemNucleophile</code>	19
<code>\chemmacros_translate:n (expl3)</code>	66 f.	<code>\DeclareChemPartialCharge</code>	10
<code>\ChemModule</code>	69 f.	<code>\DeclareChemParticle</code>	19, 62
<code>chemnum (package)</code>	3, 37, 39, 42, 48	<code>\DeclareChemPhase</code>	21
<code>\chemomega</code>	13	<code>\DeclareChemReactant</code>	37 ff., 42
<code>\chemprime</code>	11	<code>\DeclareChemReaction</code>	35
<code>\chemsetup</code> 4 f., 7, 9, 18, 20 f., 23, 25–30, 34, 40 ff., 45, 47, 53, 61, 69 f.		<code>\DeclareChemState</code>	58
<code>chemstyle (package)</code>	18, 60	<code>\DeclareChemTranslation</code>	40, 64, 67
<code>\ChemTranslate</code>	6 f., 67	<code>\DeclareChemTranslations</code>	64, 67
<code>\chlewis</code>	18, 37	<code>\DeclareDocumentCommand</code>	62
<code>choice-option</code>	4	<code>\DeclareTranslation</code>	64
<code>\cip</code>	14, 16	<code>delimiters</code>	32
<code>cip-inner-format</code>	14	<code>\delm</code>	9
<code>cip-kern</code>	14	<code>\delp</code>	9
<code>cip-number-format</code>	14	<code>\Delta</code>	58
<code>cip-outer-format</code>	14	<code>delta</code>	52
<code>circled</code>	9	<code>\dento</code>	15 f.
<code>circletype</code>	9	<code>\dexter</code>	14, 16
<code>\cis</code>	15 f.	<code>dist</code>	46
<code>cleveref (package)</code>	34, 49, 62	E	
<code>\cmc</code>	60	<code>\E</code>	12, 15
<code>\cmpd</code>	37, 39	<code>\El</code>	18
<code>\cmpd+</code>	42	<code>\el</code>	18 f., 47 f., 67
<code>\cmpdplain</code>	42	<code>\elementarycharge</code>	61
<code>\CNMR</code>	50	<code>elements (package)</code>	25
<code>\co</code>	30	<code>elpair</code>	18 f., 23 f.
<code>COHAN, E. Richard</code>	6 f., 13, 20, 44	<code>\ElPot</code>	59
<code>color</code>	28	<code>\endo</code>	17
<code>\comb</code>	31	<code>\entgegen</code>	15
<code>compatibility</code>	69	<code>\Enthalpy</code>	59
<code>\compl</code>	31	<code>\enthalpy</code>	57 ff.
<code>\complex</code>	31	<code>\entropy</code>	57 ff.
<code>concentration-unit</code>	40 f.	<code>eq-constant</code>	7
<code>connector</code>	51	<code>equiv-unit</code>	40
<code>CONNELLY, Neil G.</code>	16	<code>equivalents</code>	42
<code>cool (package)</code>	12	<code>errorcheck (module)</code>	63
<code>coord-use-hyphen</code>	16	<code>etoolbox (package)</code>	61
<code>\copolymer</code>	30	<code>experimental (environment)</code>	51 f.
<code>coupling-bonds-post</code>	52	<code>explicit-sign</code>	44 f.
<code>coupling-bonds-pre</code>	52	<code>explicit-zero-sign</code>	44
<code>coupling-nuclei-post</code>	52	<code>explizit-zero-sign</code>	44
<code>coupling-nuclei-pre</code>	52	F	
<code>coupling-pos</code>	52	<code>\fac</code>	15
<code>coupling-pos-cs</code>	52	<code>fancyref (package)</code>	34, 49, 62
<code>coupling-symbol</code>	52	<code>\fdelm</code>	9 f.
<code>coupling-unit</code>	52	<code>\fdelp</code>	9 f.
<code>\cyclic</code>	31	<code>float (package)</code>	49
<code>\cyclo</code>	31	<code>floatrow (package)</code>	49

INDEX

<code>\fmch</code>	8 ff.	<code>\Kw</code>	6 f.
<code>\fminus</code>	8 ff.	L	
<code>format</code>	17 f., 25 f., 44, 52	<code>\L</code>	12, 14
<code>formula</code>	19, 22 f., 32, 44, 49, 52	<code>l3packages (bundle)</code>	45, 62, 68
<code>\fpch</code>	8 ff.	<code>_chemmacros_language_tl (expl3)</code>	66
<code>\fplus</code>	8 ff.	<code>\laevus</code>	14
<code>fraction-unit</code>	40	<code>lang</code>	49
<code>\fscrm</code>	8	<code>lang (module)</code>	5, 63 f.
<code>\fscrp</code>	8	<code>language</code>	63 f.
<code>\fsscrm</code>	8	<code>\latin</code>	17 f.
<code>\fsscrp</code>	8	LEHMAN, Philipp	61
FUJITA, Shinsaku	22	<code>\LetChemIUPAC</code>	16
G		<code>list</code>	52
<code>\g</code>	13	<code>list-entry</code>	36
<code>\gas</code>	20	<code>list-heading-cmd</code>	36
<code>\gibbs</code>	57 ff.	<code>list-name</code>	36
<code>glossaries-extra (package)</code>	37, 43	<code>list-setup</code>	53
<code>\graft</code>	31	<code>\listofreactions</code>	35 f.
<code>\gram</code>	53, 55	<code>\listofschemes</code>	49
<code>greek</code>	24	<code>\listschemename</code>	49
<code>greek (module)</code>	24	<code>longtable (package)</code>	37, 43
H		<code>LPPL</code>	3, 70
<code>\H</code>	13	<code>\lqd</code>	20
<code>half</code>	29	M	
<code>\hapto</code>	15 f.	<code>\m</code>	13
<code>\Helmholtz</code>	59	MADSEN, Lars	32, 44
HENSEL, Martin	5, 22	<code>main</code>	42
<code>\HNMR</code>	50	<code>main-sub-sep</code>	37
<code>\Hyd</code>	18	<code>\makepolymerdelims</code>	31 f.
<code>\hydrogen</code>	13 f.	<code>mass-unit</code>	40
<code>hyperref (package)</code>	37 f., 42	<code>mathtools (package)</code>	32 f., 44
<code>hyphen-post-space</code>	12	<code>\mch</code>	8 f.
<code>hyphen-pre-space</code>	11	<code>\mech</code>	26
I		<code>mechanisms (module)</code>	26
<code>\initcmpd</code>	39	<code>\mega</code>	55
<code>initiate</code>	39	<code>memoir (class)</code>	48
<code>\insitu</code>	17	<code>\mer</code>	15
<code>\intertext</code>	34	<code>\meta</code>	12, 15 ff.
<code>\invacuo</code>	17	<code>method</code>	52
<code>\ipn</code>	31	<code>mhchem</code>	23
<code>\ipnetwork</code>	31	<code>mhchem (package)</code>	5, 8, 22 ff.
<code>\isotope</code>	25 f.	<code>minimal</code>	4 f.
<code>isotope (module)</code>	25	MITTELBACH, Frank	22, 26, 61
<code>iupac</code>	12, 17	<code>\mmHg</code>	61
<code>\iupac</code>	11–17, 30, 38	<code>module (module)</code>	4
J		<code>modules</code>	4 f., 61
<code>\J</code>	51	<code>\Molar</code>	60
K		<code>\moLar</code>	60
<code>\k</code>	13	<code>\molar</code>	60
<code>K-acid</code>	6	<code>\MolMass</code>	60
<code>K-base</code>	7	N	
<code>K-water</code>	7	<code>\N</code>	13
K., Sonja	5, 37	<code>\n</code>	13
<code>\Ka</code>	6 ff.	<code>name</code>	37, 42 f.
<code>\Kb</code>	6 f.	<code>\net</code>	31
KOHM, Markus	10	<code>\network</code>	31
KOMA-Script (bundle)	10	<code>\NewChemCharge</code>	10
		<code>\NewChemEqConstant</code>	7 f.

INDEX

<code>\NewChemIUPAC</code>	16 f.	<code>\pH</code>	6 f.
<code>\NewChemIUPACShorthand</code>	17	<code>phase</code>	28
<code>\NewChemLatin</code>	18	<code>\phase</code>	20 f., 29
<code>\NewChemMacroset</code>	62	<code>phases (module)</code>	20
<code>\NewChemNMR</code>	50, 53	<code>\phosphorus</code>	14
<code>\NewChemNucleophile</code>	19	<code>\pKa</code>	6 f.
<code>\NewChemPartialCharge</code>	9 f.	<code>\pKb</code>	6 f.
<code>\NewChemParticle</code>	19, 62	<code>\pOH</code>	6
<code>\NewChemPhase</code>	21	<code>polyglossia (package)</code>	39, 63
<code>\NewChemReaction</code>	34 f.	<code>polymers (module)</code>	30 f.
<code>\NewChemState</code>	58 f.	<code>pos</code>	20, 44 f.
<code>newfloat (package)</code>	48	<code>\pos</code>	9, 20 f., 45, 51–55, 59, 69
<code>\newman</code>	27 f.	<code>pos-number</code>	52
<code>newman (module)</code>	27	<code>post</code>	56 ff.
NIEDERBERGER, Clemens 5, 8, 18, 21 f., 24 f., 37, 48, 63		POWELL, W. H.	15
NIEPRASCHK, Rolf	37	<code>pre</code>	56 ff.
<code>\nitrogen</code>	13 f.	<code>\printatom</code>	23
<code>\NMR</code>	49 ff., 53 f.	<code>\printreactants</code>	39, 43
<code>nmr</code>	51	<code>\printreactants*</code>	43
<code>nmr-base-format</code>	52	<code>printreactants-style</code>	39, 43
<code>nomenclature (module)</code>	10, 30	<code>\ProvideChemCharge</code>	10
“Nomenclature of Inorganic Chemistry”, IUPAC Red Book		<code>\ProvideChemEqConstant</code>	7
16		<code>\ProvideChemIUPAC</code>	16
“Nomenclature of Organic Chemistry, Sections A, B, C, D,		<code>\ProvideChemIUPACShorthand</code>	17
E, F, and H”, IUPAC Blue Book	15	<code>\ProvideChemLatin</code>	18
<code>\nonbreakinghyphen</code>	11	<code>\ProvideChemNMR</code>	50
<code>\normal</code>	60	<code>\ProvideChemNucleophile</code>	19
<code>\ntr</code>	18	<code>\ProvideChemPartialCharge</code>	10
<code>\Nu</code>	19	<code>\ProvideChemParticle</code>	19, 62
<code>\Nuc</code>	18 f.	<code>\ProvideChemPhase</code>	21
<code>nucleus</code>	51	<code>\ProvideChemReaction</code>	35
		<code>\ProvideChemState</code>	58
O		<code>\prt</code>	18 f.
<code>\O</code>	13	<code>purity-unit</code>	40 f.
OBERDIEK, Heiko	37		
<code>opacity</code>	29	Q	
<code>option</code>	4	“Quantities, Symbols and Units in Physical Chemistry”,	
<code>\orbital</code>	28 ff.	IUPAC Green Book	6 f., 13, 20, 44
<code>orbital (module)</code>	28		
<code>\ortho</code>	3, 15	R	
<code>overlay</code>	29	<code>\R</code>	12, 14
<code>\OX</code>	46 ff.	RAHTZ, Sebastian	37
<code>\ox</code>	9, 44–48, 69	<code>\ran</code>	31
<code>\Oxo</code>	18	<code>\random</code>	31
<code>\oxygen</code>	13 f.	<code>\Rconf</code>	15
		<code>\Reactant</code>	38
P		<code>\reactant</code>	37–42
<code>\P</code>	14	<code>\reactant*</code>	39
<code>\p</code>	6, 12	<code>\reactant+</code>	42
<code>p-style</code>	6	<code>reactant-output-style</code>	38, 41
PANICO, R.	15	<code>\reactantl</code>	43
<code>\para</code>	15	<code>\Reactantplain</code>	42
<code>parse</code>	44, 52	<code>\reactantplain</code>	42
<code>partial-format</code>	9	<code>reactants</code>	5, 37 ff., 41 ff.
<code>particles</code>	23 f.	<code>\reactants</code>	38 f., 41 ff.
<code>particles (module)</code>	18	<code>reaction (environment)</code>	32, 34 f.
<code>\pch</code>	8 f., 46 f.	<code>reaction* (environment)</code>	32
PEDERSEN, Bjørn	10	<code>\reactionlistname</code>	36
PEDERSEN, Bjørn	10	<code>reactions (environment)</code>	32, 36
<code>\per</code>	31, 53, 55, 58 f.	<code>reactions (module)</code>	23 f., 32, 35
<code>\periodic</code>	31	<code>reactions* (environment)</code>	32

INDEX

reactionsat (environment)	35	<code>\statistical</code>	30
<code>\rectus</code>	14	<code>\submainreactantplain</code>	42
<code>\redox</code>	9, 45–48, 69	<code>subscript</code>	32, 56 f., 59
redox (module)	44	<code>subscript-left</code>	56 f., 59
resize (package)	44	<code>subscript-pos</code>	57, 59
<code>\RemoveChemIUPACShorthand</code>	17	<code>subscript-right</code>	56 f., 59
<code>\RenewChemCharge</code>	10	<code>\sulfur</code>	14
<code>\RenewChemEqConstant</code>	7	<code>\Sumbainreactantplain</code>	42
<code>\RenewChemIUPAC</code>	16 f.	<code>super-frac</code>	45
<code>\RenewChemIUPACShorthand</code>	17	<code>superscript</code>	32, 56 f., 59
<code>\RenewChemLatin</code>	18	<code>superscript-left</code>	56 ff.
<code>\RenewChemNMR</code>	50	<code>superscript-right</code>	56–59
<code>\RenewChemNucleophile</code>	19	<code>switch</code>	39
<code>\RenewChemPartialCharge</code>	10	<code>symbol</code>	58 f.
<code>\RenewChemParticle</code>	19, 62	symbols (module)	22
<code>\RenewChemPhase</code>	21	<code>\syn</code>	15
<code>\RenewChemReaction</code>	34	T	
<code>\RenewChemState</code>	58 f.	<code>tag-close</code>	34
REUTENAUER, Arthur	63	<code>tag-open</code>	33
RICHER, J-C	15	TALBOT, Nicola L. C.	37
<code>ring</code>	27	TANTAU, Till	67
ROBERTSON, Will	32, 44	TELLECHEA, Christian	8, 22
<code>roman</code>	44	<code>\ter</code>	15
S		<code>\tert</code>	15
<code>\S</code>	12, 14	<code>text-frac</code>	45
<code>scale</code>	27 f.	THE L ^A T _E X3 PROJECT TEAM	37, 45, 62, 68
scheme (environment)	48 f.	thermodynamics (module)	56 f.
scheme (module)	48	tikz (module)	10, 27 f., 30, 44, 67
<code>\schemename</code>	49	tikz (package)	67
SCHÖPF, Rainer	22, 26, 61	TikZ/pgf (package)	67
<code>\Sconf</code>	15	<code>tocbasic</code>	36
scrfile (package)	10	tocbasic (package)	36, 48
<code>\scrm</code>	8	<code>\torr</code>	60
<code>\scrp</code>	8	<code>\trans</code>	12, 15
<code>\second</code>	37, 55	<code>\transitionstatesymbol</code>	22
<code>sep</code>	47	translations (package)	21, 63 f., 66
<code>\Sf</code>	14	U	
<code>short</code>	38, 43	<code>unit</code>	51, 58 f.
<code>side-connect</code>	26, 44	<code>units</code>	40
<code>\sin</code>	15	units (module)	60
<code>\sinister</code>	14	upgreek (package)	24
<code>\sipn</code>	31	<code>upper-bookmark</code>	38, 42
<code>\sipnetwork</code>	31	<code>upper-name</code>	38, 42
<code>\sisetup</code>	39	<code>use-equal</code>	51, 53
siunitx (package)	37, 39 f., 49–52, 56, 58 ff.	<code>\usechemmodule</code>	5, 63
<code>\sld</code>	20	V	
<code>solution</code>	39 f.	<code>\val</code>	51, 53 f.
<code>\Solvent</code>	38	<code>volume-unit</code>	40
<code>\solvent</code>	38–41, 43	Voss, Herbert	37
<code>solvent-output-style</code>	38, 41	W	
<code>\solventl</code>	43	<code>\w</code>	13
<code>\Solventplain</code>	42	<code>\water</code>	18
<code>\solventplain</code>	43	wave (TikZ decoration)	67
<code>\solvents</code>	43	WRIGHT, Joseph	18, 32, 37, 44, 49, 56, 60 f.
<code>space</code>	18, 20	X	
spectroscopy (module)	49 f.	X _Y TEX (package)	22
<code>\standardstate</code>	22, 56 ff.	xfrac (module)	44, 68 f.
<code>\star</code>	31		
<code>\stat</code>	30		
<code>\state</code>	56		

INDEX

xfrac (package).....	45, 68	Z
xltable (package).....	37, 43	\Z15
xparse (package).....	62	\zusammen15 f.