SUBSTANCES

v0.2 2015/09/28

A Chemical Database

Clemens NIEDERBERGER

http://www.mychemistry.eu/forums/forum/substances/

contact@mychemistry.eu

The SUBSTANCES package allows you to create a database like file that contains data of various chemicals. These data can be retrieved in the document. An index creation of the chemicals used in the document is directly supported.

Table of Contents

1 Licence and Requirements

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

SUBSTANCES loads and needs the following packages: expl3, xparse, xtemplate and l3keys2e. It also needs the chemistry package chemmacros.

2 About

The SUBSTANCES package allows you to create a database like file that contains data of various chemicals. These data can be retrieved in the document. An index creation of the chemicals used in the document is directly supported.

3 Options

SUBSTANCES has only a few options:

`draft` = `true`|`false` Default: `false`
If set to true all warnings will be errors.

`final` = `true`|`false` Default: `true`
The opposite of `draft`.

`index` = `true`|`false` Default: `false`
Add index entries when `\chem` is called, see section 7.

`style` = {⟨*style*⟩} Default: `default`
Load specific style, see section 4.3.

`strict` = `true`|`false` Default: `false`
If set to true all warnings will be errors. This option overwrites any `draft` or `final` option that is passed on by the document class.

4 The Database

4.1 Declaring the Chemicals

The data about substances are stored via the command

`\DeclareSubstance`{⟨*id*⟩}{⟨*list of properties*⟩}
This declares substance ⟨*id*⟩.

An entry could look like this:

```
\DeclareSubstance{NaCl}{
  name    = Sodiumchloride ,
  sum     = NaCl ,
  CAS     = 7647-14-5,
  mass    = 58.44 ,
  mp      = 801 ,
  bp      = 1465 ,
  phase   = solid ,
  density = 2.17
```

```
}
```

Such entries can either be declared in the document preamble or probably more useful in a file with the ending `.sub`. Such a file can be input in the document via

`\LoadSubstances{`⟨*filename*⟩`}`
Input the file ⟨*filename*⟩ *without* specifying the file ending.

Suppose you have the file `mysubstances.sub` then you input it in the document preamble via `\LoadSubstances{mysubstances}`.

4.2 Available Fields

4.2.1 Always Defined Fields

Below all fields defined by SUBSTANCES are listed.[1]

`name` = {⟨*name*⟩} (required)
The IUPAC name of the substance. This is the only field that *has* to be used. The field's input is parsed with chemmacros' command `\iupac`.

`sort` = {⟨*sort name*⟩}
If you plan to use the `index` option you should specify this field to get the sorting of the index right. This then creates index entries `\index{`⟨*sort field*⟩`@`⟨*name field*⟩`}`.

`alt` = {⟨*alt name*⟩}
An alternative name. The field's input is parsed with chemmacros' command `\iupac`.

`altsort` = {⟨*sort alt name*⟩}
This is the same as the `sort` field but for the alternative name.

`CAS` = {⟨*CAS number*⟩}
The Chemical Abstract Service (CAS) number. The input needs to be input in the form ⟨*num*⟩-⟨*num*⟩-⟨*num*⟩.

`PubChem` = {⟨*PubChem number*⟩}
The PubChem number.

The `CAS` field processes the number using the macro `\CAS{`⟨*number*⟩`}` which is defined like this:

```
\def\@CAS#1-#2-#3\relax{\iupac{#1-#2-#3}}
```

1. Look in the file `substances-examples.sub` which is part of this package and should be in the same place as this documentation for example uses.

```
\NewDocumentCommand\CAS{m}{\@CAS#1\relax}
```

You're free to redefine it to your needs.

4.2.2 Style-dependend Fields

SUBSTANCES defines the style 'default' which is loaded if no other style has been specified. It defines the following additional fields and loads the packages chemfig and siunitx.

`formula` = {⟨*formula*⟩}
The molecular formula of the substance. The field's input is parsed with chemmacros' command `\ch`.

`structure` = {⟨*structure*⟩}
The structural formula of the substance. The field's input is parsed with chemfig's command `\chemfig`.

`mp` = {⟨*melting point*⟩}
The melting point. The field's entry is input into the siunitx command `\SI` in the following way:
`\SI{`⟨*field*⟩`}{\celsius}`.

`bp` = {⟨*boiling point*⟩}
The boiling point. The field's entry is input into the siunitx command `\SI` in the following way:
`\SI{`⟨*field*⟩`}{\celsius}`.

`density` = {⟨*density*⟩}
The density. The field's entry is input into the siunitx command `\SI` in the following way:
`\SI{`⟨*field*⟩`}{\gram\per\cmc}`.

`phase` = {⟨*phase*⟩}
The state of aggregation.

`pKa` = {⟨pK_a⟩}
The pK_a value. The field's entry is input into the siunitx command `\num`.

`pKa1` = {⟨p$K_{\mathrm{a}1}$⟩}
The first of several pK_a values. The field's entry is input into the siunitx command `\num`.

`pKa2` = {⟨p$K_{\mathrm{a}2}$⟩}
The second of several pK_a values. The field's entry is input into the siunitx command `\num`.

`pKb` = {⟨pK_b⟩}
The pK_b value. The field's entry is input into the siunitx command `\num`.

`pKb1` = {⟨p$K_{\mathrm{b}1}$⟩}
The first of several pK_b values. The field's entry is input into the siunitx command `\num`.

pKb2 = {⟨pK_{b2}⟩}
The second of several pK_b values. The field's entry is input into the siunitx command \num.

pictograms = {⟨*csv list of pictograms*⟩}
The GHS pictograms. This field takes a list of pictogram names as they're input into ghsystems' command \ghspic.

H = {⟨*csv list of hazard statements*⟩}
The H statements. This field takes a list of numbers as they're input into ghsystems' command \ghs{h}{⟨*number*⟩}.

P = {⟨*csv list of precautionary statements*⟩}
The P statements. This field takes a list of pictogram names as they're input into ghsystems' command \ghs{p}{⟨*number*⟩}.

EUH = {⟨*csv list of EUH statements*⟩}
The EUH statements. This field takes a list of pictogram names as they're input into ghsystems' command \ghs{euh}{⟨*number*⟩}.

LD50 = {⟨*Median Lethal Dose*⟩}
The LD50 in mg kg^{-1}. The field's entry is input into the siunitx command \SI in the following way:
\SI{⟨*field*⟩}{\milli\gram\per\kilo\gram}.

4.3 Define Custom Styles

4.3.1 Background

You might have other needs for fields than the ones defined by SUBSTANCES and the 'default' style. *All* fields except the required name field are defined by the 'default' style.

You can easily define your own style which means that you save a file with the name substances-⟨*style*⟩.def. In it you both define the commands you need and you declare substance properties with the command \DeclareSubstanceProperty which is explained in the next section, to declare your own fields.

Such a style file typically should start with a declaration as follows:

\SubstancesStyle*{⟨*style name*⟩}
This declares the style ⟨*style name*⟩. This also switches to the expl3 programming environment. The starred version doesn't switch to the expl3 programming environment. Either way @ has category code 11 in a style file.

4.3.2 Declare New Fields or Change Existing Fields

You might want other fields or change the definition of the predefined ones. For this there's

\DeclareSubstanceProperty*{⟨*field name*⟩}[⟨*pre code*⟩][⟨*post code*⟩]
This command declares a new property field for a substance. The star makes the property a

required one which means an error will be issued if a substance is declared without it. The optional arguments ⟨*pre code*⟩ and ⟨*post code*⟩ specify any code that should be input directly before or after the field entry, respectively. The ⟨*pre code*⟩ may end with a command that takes one mandatory argument. In this case the field entry will be its argument.

The following example would define a field `EC` which uses a custom command to parse the field entry. The European Commission Number (EC) is assigned to chemical substances for regulatory purposes within the European Union by the regulatory authorities.

```
\makeatletter
\def\@EC#1-#2-#3\relax{#1-#2-#3}
\newcommand*\EC[1]{\@EC#1\relax}
\makeatother
\DeclareSubstanceProperty{EC}[\EC]
```

For further examples of the usage of pre and post code look at the definition of the `name` and the `mp` field:

```
\DeclareSubstanceProperty*{name}[\iupac]
\DeclareSubstanceProperty{mp}[\SI][{\celsius}]
```

5 Retrieving the Data

There are two commands defined by SUBSTANCES that allow the retrieving of the data. The command `\chem` is intended as user command, the command `\GetSubstanceProperty` can be used to define your own user command (perhaps in your own style file, see section 4.3).

`\chem*[`⟨*pre*⟩`][`⟨*post*⟩`]{`⟨*id*⟩`}[`⟨*property*⟩`]`
If the command `\chem` is called without the optional ⟨*property*⟩ argument the `name` entry will be called. The starred version calls the `alt` entry if it is defined and the `name` entry otherwise. The arguments ⟨*pre*⟩ and ⟨*post*⟩ add arbitrary input before or after the output, respectively.

`\GetSubstanceProperty{`⟨*id*⟩`}{`⟨*property*⟩`}`
Retrieves ⟨*property*⟩ for substance ⟨*id*⟩.

All of the next examples use the data defined in the file `substances-examples.sub` that is part of this package, see section 8.

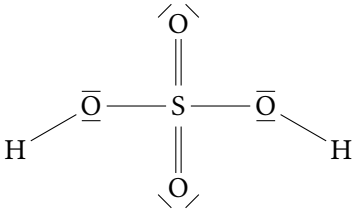
```
\chem{H2SO4}[structure] \newline
\chem{H2SO4} has the boiling point $\chem[T_b =]{H2SO4}[bp]$ and a
density of $\chem[\rho =]{H2SO4}[density]$.

Compare the melting points of methane and ethane,
$\chem[T_m=]{methane}[mp]$ and $\chem[T_m=]{ethane}[mp]$,
with the boiling points $\chem[T_b=]{methane}[bp]$ and
$\chem[T_b=]{ethane}[bp]$.

\chem{NaCl} has the \ac{CAS} number \chem{NaCl}[CAS].

\chem{acetone} (\chem*{acetone}) is the most simple ketone:

\chem{acetone}[structure]
```

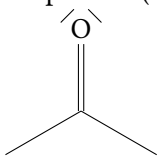


Sulfuric Acid has the boiling point $T_b = 279.6\,°C$ and a density of $\rho = 1.8356\,g\,cm^{-3}$.

Compare the melting points of methane and ethane, $T_m = -182\,°C$ and $T_m = -183\,°C$, with the boiling points $T_b = -162\,°C$ and $T_b = -89\,°C$.

Sodiumchloride has the CAS number 7647-14-5.

Propanone (Acetone) is the most simple ketone:



The following code creates table 1.

```
\begin{table}[htp]
  \centering
  \ghssetup{hide}
  \sisetup{scientific-notation=fixed,fixed-exponent=0,per-mode=symbol}
  \begin{tabular}{l>{\raggedright\arraybackslash}p{.6\linewidth}}
    \toprule
      name              & \chem{methane} \\
      formula           & \chem{methane}[formula] \\
```

```
                        & \chem{methane}[structure] \\
    \midrule
      CAS               & \chem{methane}[CAS] \\
      PubChem           & \chem{methane}[PubChem] \\
    \midrule
      boiling point     & \chem{methane}[bp] \\
      melting point     & \chem{methane}[mp] \\
      density           & \chem{methane}[density] \\
      molar mass        & \chem{methane}[mass] \\
    \midrule
                        & \chem{methane}[pictograms] \\
      H statements      & \chem{methane}[H] \\
      P statements      & \chem{methane}[P] \\
    \bottomrule
  \end{tabular}
  \caption{\label{tab:methane}All properties of \chem{methane} that have
    been saved in the example database.}
\end{table}
```

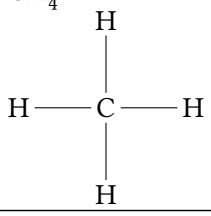
name	Methane
formula	CH_4 
CAS	74-82-8
PubChem	297
boiling point	−162 °C
melting point	−182 °C
density	0.000 72 g/cm^3
molar mass	16.04 g/mol
	
H statements	H220
P statements	P210 P377 P381 P410 + P403

Table 1: All properties of Methane that have been saved in the example database.

6 Additional Commands

SUBSTANCES provides a few commands that maybe are useful in building custom macros for styles. A field exists if it has been defined with `\DeclareSubstanceProperty` regardless if it has been used or not. A substance exists if it has been defined with `\DeclareSubstance`.

`\GetSubstanceProperty{`⟨*id*⟩`}{`⟨*field*⟩`}`
Retrieve the property specified in ⟨*field*⟩ for substance ⟨*id*⟩. This command is *not* expandable.

* `\RetrieveSubstanceProperty{`⟨*id*⟩`}{`⟨*field*⟩`}`
The same as `\GetSubstanceProperty` but expandable.

* `\ForAllSubstancesDo{`⟨*code*⟩`}`
Loops through all existing substances. Inside ⟨*code*⟩ `#1` may be used to refer to the ⟨*id*⟩ of the current substance. This command is expandable.

* `\AllSubstancesSequence`
A sequence of all substances. This is a sequence of balanced groups each containing the ⟨*id*⟩ of a substance. This command is expandable.

* `\AllSubstancesClist`
A comma separated list of all substances. Every ⟨*id*⟩ is separated from the next with a comma. This command is expandable.

* `\IfSubstancePropertyTF{`⟨*id*⟩`}{`⟨*field*⟩`}{`⟨*true code*⟩`}{`⟨*false code*⟩`}`
Tests if the property ⟨*field*⟩ is defined for the substance ⟨*id*⟩ and returns either ⟨*true code*⟩ or ⟨*false code*⟩. This command is expandable.

* `\IfSubstancePropertyT{`⟨*id*⟩`}{`⟨*field*⟩`}{`⟨*true code*⟩`}`
Tests if the property ⟨*field*⟩ is defined for the substance ⟨*id*⟩ and returns ⟨*true code*⟩ if it is. This command is expandable.

* `\IfSubstancePropertyF{`⟨*id*⟩`}{`⟨*field*⟩`}{`⟨*false code*⟩`}`
Tests if the property ⟨*field*⟩ is defined for the substance ⟨*id*⟩ and returns ⟨*false code*⟩ if it isn't. This command is expandable.

* `\IfSubstanceFieldTF{`⟨*field*⟩`}{`⟨*true code*⟩`}{`⟨*false code*⟩`}`
Tests if the property ⟨*field*⟩ exists and returns either ⟨*true code*⟩ or ⟨*false code*⟩. This command is expandable.

* `\IfSubstanceFieldT{`⟨*field*⟩`}{`⟨*true code*⟩`}`
Tests if the property ⟨*field*⟩ exists and returns ⟨*true code*⟩ if it does. This command is expandable.

* `\IfSubstanceFieldF{`⟨*field*⟩`}{`⟨*false code*⟩`}`
Tests if the property ⟨*field*⟩ exists and returns ⟨*false code*⟩ if it doesn't. This command is expandable.

* `\IfSubstanceExistTF{`⟨*id*⟩`}{`⟨*true code*⟩`}{`⟨*false code*⟩`}`
  Tests if the substance ⟨*id*⟩ exists and returns either ⟨*true code*⟩ or ⟨*false code*⟩. This command is expandable.

* `\IfSubstanceExistT{`⟨*id*⟩`}{`⟨*true code*⟩`}`
  Tests if the substance ⟨*id*⟩ exists and returns ⟨*true code*⟩ if it does. This command is expandable.

* `\IfSubstanceExistF{`⟨*id*⟩`}{`⟨*false code*⟩`}`
  Tests if the substance ⟨*id*⟩ exists and returns ⟨*false code*⟩ if it doesn't. This command is expandable.

```
Just to demonstrate how these commands can be used. And to get
our demonstration index filled.\par
\newcounter{substances}
\ForAllSubstancesDo{%
  \ifnum0=\value{substances}\relax
  \else,
  \fi
  \stepcounter{substances}%
  \chem{#1}%
  \IfSubstancePropertyT{#1}{alt}{ (\chem*{#1})}%
}
```

Just to demonstrate how these commands can be used. And to get our demonstration index filled.

Sodiumchloride, Hydrochloric Acid, Nitric Acid, Sulfuric Acid, Methane, Ethane, Propane, Butane (*n*-Butane), Pentane (*n*-Pentane), Hexane (*n*-Hexane), Heptane (*n*-Heptane), Octane (*n*-Octane), Nonane (*n*-Nonane), Decane (*n*-Decane), Propanone (Acetone)

7 Create an Index

When SUBSTANCES is called with `index = {true}` the command `\chem` will add index entries each time it is used. In this case the entries of the fields `name`, `sort`, `alt` and `altsort` will be expanded during the process. You should keep that in mind if some error arises. It might be due to a `\textbf` or similar in your database. In this case you either need to replace it with some robust command or put a `\noexpand` in front of it.

Alternative names as specified in the `alt` also get an index entry with a reference to the one of the corresponding `name` field. The entry of the `name` field in this case gets the `alt` name appended in braces.

This behaviour is not customizable for the time being. It is planned for future versions of this package, though.

As a demonstration an index for all chemicals used in this documentation is created with the help of the package imakeidx.

7.1 Formatting Commands

The index entries are formatted with the following commands. You can redefine them to your needs. If you do make sure they have the same number of required arguments and are expandable!

* `\SubstanceIndexNameEntry{⟨sort⟩}{⟨name⟩}`
  Formats the name if no `alt` field is given. The default definition is `#1@#2`.

* `\SubstanceIndexNameAltEntry{⟨sort⟩}{⟨name⟩}{⟨alt⟩}`
  Formats the name if the `alt` field is given. The default definition is `#1@#2 (#3)`.

* `\SubstanceIndexAltEntry{⟨alt sort⟩}{⟨name⟩}{⟨alt⟩}`
  Formats the entry for the `alt` field. The default definition is `#1@#3|see#2`

7.2 Using makeidx

Using the option `index = {true}` with the standard way to create an index will add the entries `\index{⟨name⟩}` to the index. This means you would mix them with other entries if you have any. Below a sample document is shown.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[index]{substances}
\LoadSubstances{substances-examples}

\usepackage{makeidx}
\makeindex
\begin{document}

\newcounter{substances}
\ForAllSubstancesDo{%
  \ifnum0=\value{substances}\relax
  \else, \fi
  \stepcounter{substances}\chem{#1}
}

\printindex
\end{document}
```

7.3 Using splitidx

Maybe a seperate index for the chemicals will make more sense. In this case you could use the package splitidx. SUBSTANCES will recognize this and create `\sindex[\jobname-chem]{⟨name⟩}` entries each time `\chem` is used.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[index]{substances}
\LoadSubstances{substances-examples}

\usepackage{splitidx}
\makeindex
\newindex[Chemicals]{\jobname-chem}
\begin{document}

\newcounter{substances}
\ForAllSubstancesDo{%
  \ifnum0=\value{substances}\relax
  \else, \fi
  \stepcounter{substances}\chem{#1}
}

\printindex[\jobname-chem]
\end{document}
```

7.4 Using imakeidx

Another way to create multiple indexes is the package imakeidx. SUBSTANCES recognizes its usage and creates index entries `\index[\jobname-chem]{⟨name⟩}`.

```
\documentclass{article}
\usepackage[T1]{fontenc}
\usepackage[index]{substances}
\LoadSubstances{substances-examples}

\usepackage{imakeidx}
\makeindex[name=\jobname-chem,title=Chemicals]
\begin{document}

\newcounter{substances}
\ForAllSubstancesDo{%
  \ifnum0=\value{substances}\relax
  \else, \fi
  \stepcounter{substances}\chem{#1}
}

\printindex[\jobname-chem]
\end{document}
```

8 The Example Database

The following code shows the example database `substances-examples.sub` that is part of this package.

```
% ------------------------------------------------------------------------
% the SUBSTANCES package
%
%   A Chemical Database
%
% ------------------------------------------------------------------------
% Clemens Niederberger
% Web:    https://bitbucket.org/cgnieder/substances/
% E-Mail: contact@mychemistry.eu
% ------------------------------------------------------------------------
% Copyright 2012--2015 Clemens Niederberger
%
% This work may be distributed and/or modified under the
% conditions of the LaTeX Project Public License, either version 1.3
% of this license or (at your option) any later version.
% The latest version of this license is in
%   http://www.latex-project.org/lppl.txt
% and version 1.3 or later is part of all distributions of LaTeX
% version 2005/12/01 or later.
%
% This work has the LPPL maintenance status `maintained'.
%
% The Current Maintainer of this work is Clemens Niederberger.
% ------------------------------------------------------------------------
% The substances package consists of the files
%  - substances.sty, substances-default.def, substances-examples.sub,
%    substances_en.tex, substances_en.pdf, README
% ------------------------------------------------------------------------
% If you have any ideas, questions, suggestions or bugs to report, please
% feel free to contact me.
% ------------------------------------------------------------------------
%
% example database to the package `substances'
%
\ProvideChemIUPAC\normal{\textit{n}}
\DeclareSubstance{NaCl}{
  name       = Sodium|chloride ,
  sort       = Sodiumchloride ,
  formula    = NaCl ,
  CAS        = 7647-14-5,
  mass       = 58.44 ,
  mp         = 801 ,
  bp         = 1465 ,
  phase      = solid ,
  density    = 2.17
}

\DeclareSubstance{HCl}{
  name       = Hydro|chloric Acid ,
```

```
  sort       = Hydrochloric Acid ,
  formula    = HCl ,
  CAS        = 7647-01-0 ,
  pictograms = {acid,exclam} ,
  H          = {314,335} ,
  P          = {260,301+330+331,303+361+353,305+351+338,405,501} ,
  mass       = 36.46 ,
  density    = 1.19 ,
  mp         = -30
}

\DeclareSubstance{HNO3}{
  name       = Nitric Acid ,
  sort       = Nitric Acid ,
  formula    = HNO3 ,
  CAS        = 7697-37-2 ,
  PubChem    = 944 ,
  mass       = 63.01 ,
  density    = 1.51 ,
  mp         = -42 ,
  bp         = 86 ,
  pKa        = -1.37 ,
  pictograms = {flame-O,acid} ,
  H          = {272,314} ,
  P          = {220,280,305+351+338,310}
}

\DeclareSubstance{H2SO4}{
  name       = Sulfuric Acid ,
  sort       = Sulfuric Acid ,
  formula    = H2SO4 ,
  structure  = {H-[:30]\Lewis{26,O}-S(=[2]\Lewis{13,O})(=[6]\Lewis{57,O})-\Lewis{26,O}-[:-30]
  H} ,
  CAS        = 7664-93-9 ,
  PubChem    = 1118 ,
  mass       = 98.08 ,
  density    = 1.8356 ,
  mp         = 10.38 ,
  bp         = 279.6 ,
  phase      = liquid ,
  pKa        = -3.0 ,
  pKa1       = -3.0 ,
  pKa2       = 1.9 ,
  pictograms = acid ,
  H          = 314 ,
  P          = {280,301+330+331,309,310,305+351+338} ,
  LD50       = 510
}

\DeclareSubstance{methane}{
  name       = Methane ,
  sort       = Methane ,
  formula    = CH4 ,
  structure  = H-C(-[2]H)(-[6]H)-H ,
```

```
  CAS        = 74-82-8 ,
  PubChem    = 297 ,
  pictograms = {flame,bottle} ,
  H          = 220 ,
  P          = {210,377,381,410+403} ,
  mass       = 16.04 ,
  density    = 0.72e-3 ,
  mp         = -182 ,
  bp         = -162 ,
  phase      = gaseous
}

\DeclareSubstance{ethane}{
  name       = Ethane ,
  sort       = Ethane ,
  formula    = C2H6 ,
  structure  = H-C(-[2]H)(-[6]H)-C(-[2]H)(-[6]H)-H ,
  CAS        = 74-84-0 ,
  PubChem    = 6324 ,
  pictograms = {flame,bottle} ,
  H          = 220 ,
  P          = {210,377,381,403} ,
  mass       = 30.07 ,
  density    = 0.72e-3 ,
  mp         = -183 ,
  bp         = -89 ,
  phase      = gaseous
}

\DeclareSubstance{propane}{
  name       = Propane ,
  sort       = Propane ,
  formula    = C3H8 ,
  structure  = H-C(-[2]H)(-[6]H)-C(-[2]H)(-[6]H)-C(-[2]H)(-[6]H)-H ,
  CAS        = 74-98-6 ,
  pictograms = {flame,bottle} ,
  H          = 220 ,
  P          = {201,377,381,403} ,
  mass       = 44.10 ,
  density    = 2.01e-3 ,
  mp         = -188 ,
  bp         = -42 ,
  phase      = gaseous
}

\DeclareSubstance{butane}{
  name       = Butane ,
  sort       = Butane ,
  alt        = \normal-Butane ,
  altsort    = n-Butane ,
  formula    = C4H10 ,
  structure  = H-C(-[2]H)(-[6]H)-C(-[2]H)(-[6]H)-C(-[2]H)(-[6]H)-C(-[2]H)(-[6]H)-H ,
  CAS        = 106-97-8 ,
  PubChem    = 7843 ,
```

```
  pictograms = {flame,bottle} ,
  H          = {220,280} ,
  P          = {201,377,381,403} ,
  mass       = 58.12 ,
  density    = 2.71e-3 ,
  mp         = -138.3 ,
  bp         = -0.5 ,
  phase      = gaseous
}

\DeclareSubstance{pentane}{
  name       = Pentane ,
  sort       = Pentane ,
  alt        = \normal-Pentane ,
  altsort    = n-Pentane ,
  formula    = C5H12 ,
  structure  = H-C(-[2]H)(-[6]H)-C(-[2]H)(-[6]H)-C(-[2]H)(-[6]H)-C(-[2]H)(-[6]H)-C(-[2]H)
  (-[6]H)-H ,
  CAS        = 109-66-0 ,
  PubChem    = 8003 ,
  pictograms = {flame,health,exclam,aqpol} ,
  H          = {225,304,336,411} ,
  EUH        = 066 ,
  P          = {273,301+310,331,403+235} ,
  mass       = 72.15 ,
  density    = 0.63 ,
  mp         = -130 ,
  bp         = 36 ,
  phase      = liquid
}

\DeclareSubstance{hexane}{
  name       = Hexane ,
  sort       = Hexane ,
  alt        = \normal-Hexane ,
  altsort    = n-Hexane ,
  formula    = C6H14 ,
  structure  = -[:30]-[:-30]-[:30]-[:-30]-[:30] ,
  CAS        = 110-54-3 ,
  PubChem    = 8058 ,
  pictograms = {flame,health,exclam,aqpol} ,
  H          = {225,361f,304,373,315,336,411} ,
  P          = {210,240,273,301+310,331,302+352,403+235} ,
  mass       = 86.18 ,
  density    = 0.66 ,
  mp         = -95 ,
  bp         = 69 ,
  phase      = liquid
}

\DeclareSubstance{heptane}{
  name       = Heptane ,
  sort       = Heptane ,
  alt        = \normal-Heptane ,
```

```
  altsort    = n-Heptane ,
  formula    = C7H16 ,
  structure  = -[:30]-[:-30]-[:30]-[:-30]-[:30]-[:-30] ,
  CAS        = 142-82-5 ,
  PubChem    = 8900 ,
  pictograms = {flame,health,exclam,aqpol} ,
  H          = {225,304,315,336,410} ,
  P          = {210,273,301+310,331,302+352,403+235} ,
  mass       = 100.21 ,
  density    = 0.68 ,
  mp         = -91 ,
  bp         = 98 ,
  phase      = liquid
}

\DeclareSubstance{octane}{
  name       = Octane ,
  sort       = Octane ,
  alt        = \normal-Octane ,
  altsort    = n-Octane ,
  formula    = C8H18 ,
  structure  = -[:30]-[:-30]-[:30]-[:-30]-[:30]-[:-30]-[:30] ,
  CAS        = 111-65-9 ,
  PubChem    = 356 ,
  pictograms = {flame,health,exclam,aqpol} ,
  H          = {225,304,315,336,410} ,
  P          = {210,273,301+330+331,302+352} ,
  mass       = 114.23 ,
  density    = 0.70 ,
  mp         = -56.8 ,
  bp         = 126 ,
  phase      = liquid
}

\DeclareSubstance{nonane}{
  name       = Nonane ,
  sort       = Nonane ,
  alt        = \normal-Nonane ,
  altsort    = n-Nonane ,
  formula    = C9H20 ,
  structure  = -[:30]-[:-30]-[:30]-[:-30]-[:30]-[:-30]-[:30]-[:-30] ,
  CAS        = 111-84-2 ,
  PubChem    = 8141 ,
  pictograms = {flame,exclam,health} ,
  H          = {226,304,315,319,332,336,413} ,
  P          = {261,301+310,305+351+338,331} ,
  mass       = 128.26 ,
  density    = 0.72 ,
  mp         = -54 ,
  bp         = 151 ,
  phase      = liquid
}

\DeclareSubstance{decane}{
```

```
  name       = Decane ,
  sort       = Decane ,
  alt        = \normal-Decane ,
  altsort    = n-Decane ,
  formula    = C10H22 ,
  structure  = -[:30]-[:-30]-[:30]-[:-30]-[:30]-[:-30]-[:30]-[:-30]-[:30] ,
  CAS        = 124-18-5 ,
  PubChem    = 15600 ,
  pictograms = {flame,health} ,
  H          = {226,304} ,
  P          = {210,260,262,301+310,331} ,
  mass       = 142.29 ,
  density    = 0.73 ,
  mp         = -29.7 ,
  bp         = 174 ,
  phase      = liquid
}

\DeclareSubstance{acetone}{
  name       = Propanone ,
  sort       = Propanone ,
  alt        = Acetone ,
  altsort    = Acetone ,
  formula    = C3H6O ,
  structure  = {-[:30](=[2]\Lewis{13,O})-[:-30]} ,
  CAS        = 67-64-1 ,
  PubChem    = 180 ,
  mass       = 58.08 ,
  density    = 0.79 ,
  mp         = -95 ,
  bp         = 56 ,
  pictograms = {flame,exclam} ,
  H          = {225,319,336} ,
  EUH        = {066} ,
  P          = {210,233,305+351+338} ,
  LD50       = 5800
}

\endinput
```

Chemicals

Index