The chemsec package*

Justin Finnerty
jfinn24985@gmail.com

March 9, 2018

1 Abstract

A cross-reference system designed for an arbitrary set of items. Generates a document-order set of consecutive number labels that can be cross-referenced anywhere in the document. The motivating example is labeling chemical structures in a scientific document.

2 Usage

This package solves the problem of providing a sequence of numeric labels to represent entities in a document. The motivating example is labels for chemical structures in a scientific document. In such a document you want the full chemical name and a label to appear in the typeset document for each chemical structure the first time the structure is referenced in the document text. Subsequent references to the structure should only use the label. These labels should form a numeric sequence, with the first chemical structure being labeled **1**, the second **2** and so on.

These labels share similar features to numeric bibliographic reference styles, both use numbers to represent something, the numbers form a single sequence starting from one and incrementing as references or structures appear in the document. Furthermore these labels or citations can appear in multiple places in the text but must retain a single unique numeric label.

However there are a couple of differences to a bibliography. Firstly the name of the chemical structure generally appears where it is first used in the document along with its label rather than in a bibliography or footnote. Secondly it is common to use a single numeric label for closely related structures, appending a sub-label to denote individual structures, usually lower-case letters. For example we might have "methyl acetate (**1a**)" and "ethyl acetate (**1b**)" being specific examples of an "acetic acid ester (**1**)" structure. Lastly, you may need to separate the logical and actual first occurrence of a label, allowing use of labels in abstracts

*This document corresponds to chemsec v1.12, dated 2018/02/01.

and captions before a defining macro call that sets the actual number of the label in the main document sequence.

The solution presented here uses keys to identify structures, in a similar way to the BibTeX, with a special format that allows associating related structures to an exemplar structure. This is achieved by using keys that contain an exclamation mark (`main-key!sub-key`) to separate the key for an exemplar structure from a sub-key for the related structures. Following our example above we might have keys `acetate!methyl`, `acetate!ethyl` and `acetate` as the keys for "methyl acetate", "ethyl acetate" and "acetic acid ester" respectively. Using such keys with this package would automatically give the labels "**1a**", "**1b**" and "**1**". Alternatively in chemistry we want the sub-labels to be more meaningful, for example denoting isomeric structures or an abbreviation for different substituents. This package allows you to manually provide such a more meaningful sub-label for each sub-key, for example leading to "methyl acetate (**1-Me**)" and "ethyl acetate (**1-Et**)" and "acetic acid ester (**1**)" in the output document.

However, using sub-keys is entirely optional and you can write a complete document without using the sub-key feature. Keys with sub-keys can be freely mixed in a document with keys that do not have sub-keys. We suggest you generally want to provide an exemplar key without sub-key for each set of keys which have sub-keys. This allows you to refer in the document to the group of related chemicals as well as the specific structures, "acetic acid ester (**1**)" in our example above. Hopefully the most difficult part of using the chemsec package is choosing keys for your structures.

2.1 Known Issues

2.2 Capitalization of names

The package cannot capitalize the name of an entity, for example if an entity name appears at the beginning of a sentence. For example the capitalization of a chemical name requires parsing and understanding the complete chemical nomenclature, something well beyond the scope of this package. It is also hard to automatically detect when capitalization should occur. Currently, the solutions are to avoid using the macros that might produce entity names at the start of a sentence or to manually type the entity name and use a macro that only outputs a label. A possible future extension of this package may allow the specification of two names for a compound, one capitalized and one not, as well as macros to select between them.

2.2.1 Failure to issue warning

Note that any labeling issue arising from this package is always resolved when the document is processed twice in a row. It is therefore recommended that, after making an edit, the user always processes the document twice to ensure correct labeling.

The user should get a warning if the for some reason the labels typeset in the document are incorrect and the document processing needs to be repeated. This can happen when using a macro that requests the value of a label but does not set it. These macros use current values or values set in the *aux* file. If the values from the *aux* file need to be used they may either not be present or differ from values set later in the document. When this happens the user should get a warning message, however a number of issues may prevent this and it is always recommended to process the document twice after editing to ensure the correct labeling.

Known cases of failure to get a warning message are:

- Use of macros before the document begins, for example in the abstract environment, may fail to issue a warning about incorrect labels. (I think this is due to the boolean variable that signals an error being reset at the beginning of the document.)
- Automatic sub-labels. Currently we cannot check to see if sub-labels change during document processing. This is because sub-labels are stored as text rather than numbers and text comparison is surprisingly difficult in LaTeX.
- User-defined sub-labels. We do not detect if a user-defined sub-label is defined after a request for the sub-label. This is only an issue the first time a document is processed or when a `\DefineChemical` macro command is added to the document.

2.3 Defining structure names and sub-labels

`\DefineChemical` You define the names and optional sub-label of a compound or entity using the `\DefineChemical {`⟨*key*⟩`} {`⟨*name*⟩`} {`⟨*sub-label*⟩`}` macro. This command does not display any output in the document nor does it define the first part of the label. This means you can define all the chemicals used in the document in a single place rather than as they appear in the document. The list can even be placed in a separate file included in the main file, facilitating the reuse of keys and names in other documents. No matter where this macro is placed in the document the chemsec package will use the information wherever needed.

Leaving the `{`⟨*sub-label*⟩`}` argument empty for a `main!sub` key turns on automatic sub-labeling for this key. By default this generates sequential alphabetic labels for each `sub` key with the same `main` key for which there is no user defined sub-label. This means that you can manually supply labels for some `main!sub` keys and also have `main!sub` keys without a defined label. This is advanced usage and this package does not check to see if the automatically generated labels match any of the user-defined labels.

The *name* parameter can contain many LaTeX formatting commands, including the use of `$$` math-mode.

2.4 ChemCite Macros

The package provides a range of macros that tailor what information about a structure is output into the typeset document. These are the `ChemCite` family

of macros. All of these macros can have a "*" (star) suffix. This suffix indicates this use of the macro should act as a cross-reference for the label and not as a possible defining reference. This means that the star form can be used in abstracts, captions and anywhere in the document you need to use a label where it might appear before it appears in the main document text where you want the numeric label to be defined.

`\ChemCite{`⟨*key*⟩`}` Output the label for the given key. If this is the first time the key is used and the non-star form is used the full name precedes the label, with a star no full name is used. For example `\ChemCite{acetate!methyl}` could give "methyl acetate (**1a**)" the first time it appeared and "**1a**" elsewhere. The non-star form also sets the value of the numeric label at the point it first appears in the document. The star form uses in order; the current label if defined or a value stored in the *aux* file from the previous time the document was processed or a dummy value. The star form also never outputs the full name. For example `\ChemCite*{acetate!methyl}` could give "**1a**" if it can determine the labels or output a dummy value "**0??**" (or less likely "**0a**" or "**1??**").

`\ChemFCite{`⟨*key*⟩`}` This macro always displays the structure name as well as the label for the given key. The `F` is a mnemonic for full, indicating the full entry of the name and label is always used. For example `\ChemFCite{acetate!methyl}` would give "methyl acetate (**1a**)" the first time it appeared as well as elsewhere. The non-star form also sets the value of the numeric label at the point it first appears in the document. The star form uses in order; the current label if defined or a value stored in the *aux* file from the previous time the document was processed or a dummy value. For example `\ChemFCite*{acetate!methyl}` could give "methyl acetate (**1a**)" if it can determine the labels or output a dummy value "?? (**0??**)" or similar.

`\ChemSCite{`⟨*key*⟩`}` In counterpoint to the `\ChemFCite` macro, this always only outputs the label for the given key. The `S` is a mnemonic for short, indicating the short entry of just the label is always used. For example `\ChemSCite{acetate!methyl}` would give "**1a**" the first time it appeared as well as "**1a**" elsewhere. The non-star form also sets the value of the numeric label at the point it first appears in the document. The star form uses in order; the current label if defined or a value stored in the *aux* file from the previous time the document was processed or a dummy value. For example `\ChemSCite*{acetate!methyl}` could give "**1a**" if it can determine the labels or output a dummy value "**0??**" (or less likely "**0a**" or "**1??**").

`\ChemMFCite{`⟨*key*⟩`}` This macro always displays the structure name of the main part of the key with the label for the given key. The `MF` is a mnemonic for main full or mixed full, indicating the full entry of the *main* (exemplar) name is used with the label and sub-label for the key. For example `\ChemMFCite{acetate!methyl}` would give "acetic acid ester (**1a**)" wherever it was used. The non-star form also sets the value of the numeric label at the point it first appears in the document. The star form uses in order; the current label if defined or a value stored in the *aux* file from the previous time the document was processed or a dummy value. For example `\ChemMFCite*{acetate!methyl}` could give "acetic acid ester (**1a**)" if it can determine the labels or output a dummy value "?? (**0??**)" or similar.

`\ChemMSCite{`⟨*key*⟩`}` This macro always displays just the label from the main part of the key. The `MS`

is a mnemonic for main short or mixed short, indicating the short entry of just the *main* (exemplar) label for the key. For example `\ChemMSCite{acetate!methyl}` would give "**1**" wherever it was used. While this macro always gives exactly the same result as `\ChemSCite` with just the main part of a key (e.g. `\ChemSCite{acetate}`), it is included here because it allows the document writer to include more context in the document as it is being written. The non-star form also sets the value of the numeric label at the point it first appears in the document. The star form uses in order; the current label if defined or a value stored in the *aux* file from the previous time the document was processed or a dummy value. For example `\ChemMSCite*{acetate!methyl}` could give "**1**" if it can determine the label or output a dummy value "**0**".

`\NoCite{`⟨*key*⟩`}` This macro outputs nothing in the document but sets the numeric value of the label for the key in the document ordered sequence.

2.5 Possible Future extensions

`\ChemEntityTable` Output a table of entity names, labels and keys. This is intended to be used for creating a useful reference table when creating and editing a document rather than for final output.

2.6 Internal macros that control style

Style macros can be overridden by the user to control the style of the label and accompanying text.

`\ChemLabelStyle` Internal macro that formats a label. It is used by the other two style macros for this purpose. Users may redefine this macro to change the label formatting.

`\ChemFullLabelStyle` Internal macro used to format a label and its full name text. Users may redefine this macro to change the name and label formatting, particularly when you wish to format the label and sub-label separately.

`\ChemShortLabelStyle` Internal macro used to formats a label without name text. Users may redefine this macro to change the name and label formatting, particularly when you wish to format the label and sub-label separately.

`\ChemMainCounterStyle` Internal macro that converts the counter used for main key label into a number or letter. The default implementation uses `\arabic` to generate a number. Users may redefine this macro to output letters instead of numbers.

`\ChemSubCounterStyle` Converts the counter used for the sub-key label into a number or letter. The default implementation uses `\alph` to generate a letter. Users may redefine this macro to output numbers instead of letters.

3 Design brief

This section contains a description of what this package sets out to achieve. The package should be able to consistently label an entity throughout the document based on a user-defined key. The label can be either a consecutive number or a

user-defined label. These labels can be combined with a sub-label, which can also be consecutive or user-defined.

When sub-labels are used a user-supplied key and sub-key are required. Any entity associated with the key without a sub-key is considered to be a generic entity for any entities defined with the same key and a sub-key.

3.1 Limits

- The package can provide consecutive labels or user-defined labels but not both.
- Unless user-defined labels are used, consecutive labels are numeric for labels and alphabetic for sub-labels.
- The package can provide only a single set of consecutive labels.
- The package can provide consecutive sub-labels or user-defined sub-labels but not both for each label.

3.2 Actions

- The first time the `\ChemCite` macros appear in the latex document with a given key is where the consecutive number label for the key is defined (unless user-defined label are being used).
- Using macros `\ChemCite*` provides the label without defining the consecutive number label. This allows the label to appear, for example in a table, abstract or figure, before the desired defining location of the label.
- The names of entities can be defined anywhere in the document and appear in a user-controlled manner.
- The basic macros typeset the entity name and label at the first defining reference position and only the label in other places.
- A range of advanced macros allow greater control over the display of the entity name and label. For example the user can output a label and sub-label but use the name of the generic entity. A chemistry example of this with "ethyl acetate (1b)" and its generic entity "acetic acid ester (1)" appearing as "... gives the second acetic acid ester (1b)...".
- The label will be typeset in a particular, user configurable, style. The label should be consistently formatted, even in math-mode.
- The label can be used transparently by other packages. For example it can be used with the psfrag package to inject the correct labels into figures at documentation preparation time rather than having to recreate new figures whenever the label numbering changes.

4 Package options

The package has two options. Both options are intended to be used when debugging the package or its usage. Control of the style used when typesetting names and labels is optionally done manually by redefining the style control macros described elsewhere.

`draft` Output structure/entity key as a superscript before any name and/or label output. This can be useful during document preparation.

`debug` Output extra information in the processing log to do with the operation of the package and turn on `draft` mode.

5 Implementation

The package creates a virtual mapping of keys to a name and a label. The key is stored as `Main!Sub` if the key has a sub-key. This mapping is stored into the `aux` file where it is read at the start of each document processing. If the mapping is altered during processing the user gets a warning to reprocess the document, similar to other automatically created indices.

The package has several sets of macros. The public macros are used to define the key to name mapping and to lookup and output the names and labels based on the key during document processing. One set of internal macros defines the style used to display the name and labels. These macros can be redefined by the user. A second set of internal macros controls the definition and management of the map data. A third set controls logging information and extra output that might be used during testing.

This uses a simple first cited first numbered algorithm to generate a unique sequential label for a set of compounds. It relies on a set of chemical definitions that can appear anywhere in the text and are stored in the "aux" file at program termination and read in at program start up.

Note on programming style. All the functions are declared using LaTeX commands unless using TeX functions `edef`, `gdef` and `xdef` were required. The public functions have been defined using LaTeX2e's `DeclareRobustCommand*{}`.

5.1 Output Style Macros

`\ChemLabelStyle` `{`⟨*label*⟩`}` The default implementation formats the label in a sloping bold font.

```
\DeclareRobustCommand*{\ChemLabelStyle}[1]{%
  \textsl{\bfseries{}#1}}
```

`\ChemFullLabelStyle` `{`⟨*full name*⟩`}` `{`⟨*main label*⟩`}` `{`⟨*sub-label*⟩`}` Formats the full entity name and label. The default implementation does not format the entity name, simply forwarding the argument as-is. The default implementation appends the sub-key label to the main label and uses the `\ChemLabelStyle` to format them together.

```
\DeclareRobustCommand*{\ChemFullLabelStyle}[3]{%
```

```
  \ifthenelse{\equal{}{#3}}{%
    #1 (\ChemLabelStyle{#2})%
  }{%
    #1 (\ChemLabelStyle{#2#3})%
  }%
}%
```

`\ChemShortLabelStyle` `{`⟨*main label*⟩`}` `{`⟨*sub-label*⟩`}` Formats the labels when they appear without the entity name. The default implementation appends the sub-key label to the main label and uses the `\ChemLabelStyle` to format them together.

```
\DeclareRobustCommand*{\ChemShortLabelStyle}[2]{%
  \ifthenelse{\equal{}{#2}}{%
    \ChemLabelStyle{#1}%
  }{%
    \ChemLabelStyle{#1#2}%
  }%
}%
```

`\ChemMainCounterStyle` `{`⟨*counter*⟩`}` Converts the counter used for the main key label into a number or letter. The default implementation uses `\arabic` to generate a number.

```
\DeclareRobustCommand*{\ChemMainCounterStyle}[1]{%
  \arabic{#1}}
```

`\ChemSubCounterStyle` `{`⟨*counter*⟩`}` Converts the counter used for the sub-key label into a number or letter. The default implementation uses `\alph` to generate a letter.

```
\DeclareRobustCommand*{\ChemSubCounterStyle}[1]{%
  \alph{#1}}
```

5.2 Macros for Debugging and Draft Output

Several macros are defined to implement the `draft` and `debug` package options. When these options are not used these macros fall back on default implementations that mostly do nothing. These macros are redefined when the options are used.

`\CN@DEBUG` Internal macro used to suppress debugging messages when not debugging. This is redefined if the `debug` package option is used to output debugging messages in the processing log.

```
\makeatletter
\gdef\CN@DEBUG #1#2{}
```

`\CN@WARN` Internal macro used to output errors and warnings as information messages when not debugging. This is redefined if the `debug` option is used to promote these messages to warning messages in the processing log.

```
\gdef\CN@WARN #1#2{\PackageInfo{#1}{#2}}
```

`\CN@DRAFT` Internal macro used to suppress draft only output when not in draft mode. This is redefined if the `debug` or `draft` option is used to add superscript text in the output document.

```
\gdef\CN@DRAFT #1{}
```

5.3 Options for Debugging and Draft Output

The package has several options to help with using the package.

- The debug option generates more error and warning messages as well as outputting draft data. It does this by redefining the \CN@DEBUG, \CN@WARN and \CN@DRAFT macros.

```
\DeclareOption{debug}{%
  \gdef\CN@DEBUG #1#2{\PackageInfo{#1}{#2}}%
  \gdef\CN@WARN #1#2{\PackageWarning{#1}{#2}}%
  \gdef\CN@DRAFT #1{$^{[#1]}$}%
}
```

- The draft option outputs some draft data. It does this by redefining the \CN@DRAFT macro.

```
\DeclareOption{draft}{%
  \gdef\CN@DRAFT #1{$^{[#1]}$}%
} \ProcessOptions
```

5.4 Counters

Several counters are used internally for generating the label values. They should not be manually altered by the user.

- CN@MaxLabelIndex tracks the maximum label index.

```
\newcounter{CN@MaxLabelIndex}
\setcounter{CN@MaxLabelIndex}{0}
```

- CN@LabelIndex is the number of the current label.

```
\newcounter{CN@LabelIndex}
```

- CN@SubLabelIndex is the number of the current sub-key label

```
\newcounter{CN@SubLabelIndex}
```

5.5 Internal Variables

- \CN@@label is a dummy value used when no label is found for a particular key. This is a guard value for erroneous usage.

```
\expandafter\def\csname CN@@label\endcsname{999}
```

- \CN@@name is a dummy name used when no name is found for a particular key. This is a guard value for erroneous usage.

```
\expandafter\def\csname CN@@name\endcsname{X}
```

- `\CN@@sublabel` is a dummy sub-label used when no label is found for a particular key/sub-key. This is a guard value for erroneous usage.

```
\expandafter\def\csname CN@@sublabel\endcsname{X}
```

- `CN@KeyFound` is a boolean to signal whether the key was found in the key list. Failure to be found indicates that this is the first time this key was cited.

```
\newboolean{CN@KeyFound}
```

- `CN@ScanList` is a boolean to signal whether the key was found in the last loop of the map scan.

```
\newboolean{CN@ScanList}
```

- `CN@Star` is a boolean that signals that a macro was called using the star form of the command. This alters `CN@getLabel` and `CN@getSubLabel` to use predefined labels (if any) instead of labels defined in the regular document flow.

```
\newboolean{CN@Star}
\setboolean{CN@Star}{false}
```

- `CN@Warning` is a boolean that signals that the document has possible incorrect labels and needs to be reprocessed. This is similar to other places where LaTeX generates warnings for automatically generated labels.

```
\newboolean{CN@Warning}
\setboolean{CN@Warning}{false}
\AtEndDocument{%
  \ifthenelse{\boolean{CN@Warning}}{%
    \PackageWarningNoLine{chemsec}{Chemsec-label(s) were used before they were
      defined or were never defined.  Rerun to get chemsec-labels right. If
      this warning message persists check that all names and labels have been
      defined in the text}%
  }{%else
    % do nothing
  }%end if
}%end def
```

5.6 Internal Macros

`\CN@mainpartofkey` Splits the representation of a key as `Main!Sub`, returning only the `Main` part.

```
\gdef\CN@mainpartofkey (#1!#2){#1}
```

`\CN@MainKey` Acts as a driver macro for `\CN@mainpartofkey` macro.

```
\gdef\CN@MainKey (#1){\CN@mainpartofkey(#1!)}%
```

`\CN@definenewchemical` `{`⟨*main!sub key*⟩`}{`⟨*name*⟩`}{`⟨*sub-label*⟩`}` Associate a key with a chemical name and sub-label. The instances of this macro written in the `aux` file are what define the current mapping of key to names and sub-labels. There is a check to make sure the key is unique but there is no check that the sub-label is unique.

This macro is the main macro that creates the table of key name and labels that is used by the other macros to generate output in the document.

```
\DeclareRobustCommand*{\CN@definenewchemical}[3]{%key, name, label
  % check for previous definition of name
  \ifcsname CN@#1@label\endcsname%ifdef
  % previous definition, do nothing
  \CN@DEBUG{chemsec}{Attempt to redefine key "#1" ignored.}%
  \else%
  % label not defined
  \CN@DEBUG{chemsec}{Defining new compound with key "#1". Name is "#2", sub-label is "#3".}%
  \expandafter\gdef\csname CN@#1@label\endcsname{0}%
  \expandafter\gdef\csname CN@#1@name\endcsname{#2}%
  \ifthenelse{\equal{#3}{-}}{%if no sub-label
  }{%else
    \ifthenelse{\equal{#3}{}}{%if no sub-label
    }{%else
      \expandafter\gdef\csname CN@#1@sublabel\endcsname{#3}%
    }%end if
  }%end if
  \fi%
}%
```

`\CN@defineLabelUsed` `{`⟨*main!sub key*⟩`}` `{`⟨*label index*⟩`}` Associate a label with a `main` key. The instances of this macro written in the `aux` file are what define the mapping of key to label used to typeset a label before a macro call that defines the position of the key in document order, and hence its true label.

```
\DeclareRobustCommand*{\CN@defineLabelUsed}[2]{%key, label
  % check for previous definition of name
  \ifcsname CN@#1@labelUsed\endcsname%ifdef
    \CN@DEBUG{chemsec}{Attempt to redefine label for key "#1" ignored.}%
  \else
    % label not defined
    \expandafter\gdef\csname CN@#1@labelUsed\endcsname{#2}%
    \CN@DEBUG{chemsec}{defineLabelUsed: Presetting label of key "#1" to "#2" -> "\csname CN@
  \fi%
}
```

`\CN@defineSublabelUsed` `{`⟨*main!sub key*⟩`}` `{`⟨*label value*⟩`}` Associate a sublabel with a `main!sub` key. The instances of this macro written in the `aux` file are what define the mapping of key to sublabel used to typeset a sublabel before a macro call that defines the position of the key in document order, and hence its true sublabel.

```
\DeclareRobustCommand*{\CN@defineSublabelUsed}[2]{%key, sublabel
  % check for previous definition of name
  \ifcsname CN@#1@sublabelUsed\endcsname%ifdef
```

```
    \CN@DEBUG{chemsec}{Attempt to redefine sublabel for key "#1" ignored.}%
  \else
    % label not defined
    \expandafter\gdef\csname CN@#1@sublabelUsed\endcsname{#2}%

    \CN@DEBUG{chemsec}{defineLabelUsed: Presetting sublabel of key "#1" to "#2" -> "\csname
  \fi%
}
```

`\CN@GetFullName` Check that a name is defined for the given key. If no name exists then check using only the `Main` part of a `Main!Sub` key. If still no name is found then output "??" and print an error message to the processing log.

```
\DeclareRobustCommand*{\CN@GetFullName}[1]{%
  \ifcsname CN@#1@name\endcsname%if name
  % use my own name
  \csname CN@#1@name\endcsname%
  \else%
  \ifcsname CN@\CN@MainKey(#1)@name\endcsname%if main key name
  % Use main key name
  \csname CN@\CN@MainKey(#1)@name\endcsname%
  \else%
  \CN@WARN{chemsec}{Attempt to find name for key "#1" failed.}%
  \setboolean{CN@Warning}{true}%
  ??%
  \fi%endif main
  \fi%endif full key
}%
```

`\CN@GetLabel` Check that a label is defined for the given key. If no label exists then check using only the `Main` part of a `Main!Sub` key. If still no label is found then if the star form is used output the value of the label from the previous time the document was processed (actually the value from `\CN@defineLabelUsed` calls in the *aux* file) or `0` (zero). If the not a star form then the label is calculated by incrementing the CNMaxLabelIndex. This also writes a call to `\CN@defineLabelUsed` in the *aux* for the next processing run.

```
\DeclareRobustCommand*{\CN@GetLabel}[3]{% key, foundbool,
  % indexcount
  \setboolean{#2}{true}%
  \CN@DEBUG{chemsec}{GetLabel: looking for '#1'}%
  \setcounter{#3}{\csname CN@#1@label\endcsname}%
  \CN@DEBUG{chemsec}{GetLabel: Label count is \arabic{#3}}%
  \CN@DEBUG{chemsec}{GetLabel: Star form is {\ifCN@Star true\else false\fi}}%
  \ifthenelse{\value{#3} = 0}{%
     %% label has NOT been defined
     \CN@DEBUG{chemsec}{GetLabel: No local label for key '#1' is defined.}%
     \ifthenelse{\boolean{CN@Star}}{%
       %% Star form means can only use predefined value
       \ifcsname CN@#1@labelUsed\endcsname% have predefined value
         \CN@DEBUG{chemsec}{GetLabel: Predefined label for key '#1' is \csname CN@#1@labelUs
```

```
        \setcounter{#3}{\csname CN@#1@labelUsed\endcsname}%
      \else% no predefined, set warning
        \setboolean{CN@Warning}{true}%
        \CN@DEBUG{chemsec}{GetLabel: No predefined label for key '#1'.}%
      \fi%
    }{%
      %% Non-star form means calculate value
      \CN@DEBUG{chemsec}{GetLabel: MaxLabelIndex is \arabic{CN@MaxLabelIndex}}%
      \stepcounter{CN@MaxLabelIndex}%
      \CN@DEBUG{chemsec}{GetLabel: Increment MaxLabelIndex to \arabic{CN@MaxLabelIndex}}%
      \expandafter\xdef\csname CN@#1@label\endcsname{\arabic{CN@MaxLabelIndex}}%
      \write\@auxout{\string\CN@defineLabelUsed{#1}{\csname CN@#1@label\endcsname}}%
      \setcounter{#3}{\csname CN@#1@label\endcsname}%
      \setboolean{#2}{false}%
      %% Check that @labelUsed matches @label or set @Warning
      \ifcsname CN@#1@labelUsed\endcsname%
        \ifthenelse{\equal{\csname CN@#1@labelUsed\endcsname}{\csname CN@#1@label\endcsname
        }{% predefined does not match.
          \CN@DEBUG{chemsec}{GetLabel: Local label does not match predefined label}
          \setboolean{CN@Warning}{true}%
        }%
      \fi
    }%
  }{%else
    % do nothing
  }%end if def to 0
  \CN@DEBUG{chemsec}{GetLabel: Result counter \arabic{#3}}%
}%end def
```

`\CN@GetSubLabel` Check that a sublabel is defined for the given key. If the star form is used output the value of the label from the previous time the document was processed (actually the value from `\CN@defineSubLabelUsed` calls in the *aux* file) or `??`. If not a star form and no sublabel exists then try and calculate a value based on the number of keys having the same `Main` part of the `Main!Sub` key. This also writes a call to `\CN@defineSubLabelUsed` in the *aux* for the next processing run.

```
\DeclareRobustCommand*{\CN@GetSubLabel}[3]{% key, found?, return-element
  \setboolean{#2}{true}%
  \CN@DEBUG{chemsec}{GetSubLabel: looking for sublabel of '#1'}%
  \ifthenelse{\equal{\CN@MainKey(#1)}{#1}}{%if a main key
    \CN@DEBUG{chemsec}{GetSubLabel: '#1' is a main key .: no sublabel}%
    \xdef#3{}%
  }{%else
    \ifcsname CN@#1@sublabel\endcsname%if defined
      \CN@DEBUG{chemsec}{GetSubLabel: Sublabel previously defined as '\csname
      CN@#1@sublabel\endcsname'}%
      \xdef#3{\csname CN@#1@sublabel\endcsname}%
    \else% else need to define sublabel
      \setboolean{#2}{false}%
      \ifthenelse{\boolean{CN@Star}}{%
```

```
        % Star form means can only use predefined value
        \ifcsname CN@#1@sublabelUsed\endcsname% have predefined value
          \CN@DEBUG{chemsec}{GetSubLabel: Predefined sublabel for key '#1' is \csname CN@#1
          \xdef#3{\csname CN@#1@sublabelUsed\endcsname}%
        \else% no predefined, set warning
          \setboolean{CN@Warning}{true}%
          \CN@DEBUG{chemsec}{GetSubLabel: No predefined sublabel for key
            '#1'.}%
          \xdef#3{??}%
        \fi%
       }{%
        \CN@DEBUG{chemsec}{GetSubLabel: Sublabel not previously defined}%
        \ifcsname CN@\CN@MainKey(#1)@sublabel\endcsname%if
          % main key def
          \setcounter{CN@SubLabelIndex}{\csname CN@\CN@MainKey(#1)@sublabel\endcsname}%
        \else%
          \setcounter{CN@SubLabelIndex}{0}%
        \fi%
        \CN@DEBUG{chemsec}{GetSubLabel: Highest label for these keys is
          \arabic{CN@SubLabelIndex}.}%
        \stepcounter{CN@SubLabelIndex}%
        \expandafter\xdef\csname CN@\CN@MainKey(#1)@sublabel\endcsname{\arabic{CN@SubLabelIn
        \expandafter\xdef\csname CN@#1@sublabel\endcsname{\ChemSubCounterStyle{CN@SubLabelIn
        \write\@auxout{\string\CN@defineSublabelUsed{#1}{\csname CN@#1@sublabel\endcsname}}%
        \CN@DEBUG{chemsec}{GetSubLabel: Sublabel now defined as '\csname
          CN@#1@sublabel\endcsname'}%
        \xdef#3{\csname CN@#1@sublabel\endcsname}%
       }%
     \fi%end if
   }%end if
}% enddef
```

`\CN@RealCite` Search for the label and sublabel for a given key.

```
\DeclareRobustCommand*{\CN@RealCite}[4]{%key,
  % found, index-counter,
  % sub
  \CN@DRAFT{#1}%
  \CN@DEBUG{chemsec}{In RealCite}%
  \setcounter{#3}{0}%
  \setcounter{CN@SubLabelIndex}{0}%
  \setboolean{#2}{false}%
  \xdef#4{}%
  \CN@DEBUG{chemsec}{Real: calling GetLabel for '\CN@MainKey(#1)', the main
    part of '#1'}%
  \CN@GetLabel{\CN@MainKey(#1)}{#2}{#3}%
  \CN@DEBUG{chemsec}{Real: calling GetSubLabel for "#1"}%
  \CN@GetSubLabel{#1}{CN@ScanList}{#4}%
}%end def
```

`\NoCite` Search for the label and sublabel for a given key without producing any output.

This can be used to define the label associated with a key without anything being typeset in the document.

```
\DeclareRobustCommand*{\NoCite}[1]{%
  \CN@RealCite{#1}{CN@KeyFound}{CN@LabelIndex}{\CN@SubLabel}%
}
```

`\CN@ChemCite` Output name, label and sublabel for a given key. The name is output only if `CN@KeyFound` is `false` which occurs the first time a key is cited.

```
\DeclareRobustCommand*{\CN@ChemCite}[1]{%
  % \CN@DEBUG{chemsec}{in Chemcite}%
  \CN@RealCite{#1}{CN@KeyFound}{CN@LabelIndex}{\CN@SubLabel}%
  % \CN@DEBUG{chemsec}{Cite: called Real}%
  \ifthenelse{\boolean{CN@KeyFound}}{%
    % \CN@DEBUG{chemsec}{Cite: key found, use Short}%
    \ChemShortLabelStyle{\ChemMainCounterStyle{CN@LabelIndex}}{\CN@SubLabel}%
  }{%else
    % \CN@DEBUG{chemsec}{Cite: key not found, use Long}%
    \ChemFullLabelStyle{\CN@GetFullName{#1}}{\ChemMainCounterStyle{CN@LabelIndex}}{\CN@SubLa
  }%
}%
```

`\CN@ChemFCite` Output name, label and sublabel for a given key. The name is always output.

```
\DeclareRobustCommand*{\CN@ChemFCite}[1]{%
  % \CN@DEBUG{chemsec}{in ChemFullCite}%
  \CN@RealCite{#1}{CN@KeyFound}{CN@LabelIndex}{\CN@SubLabel}%
  % \CN@DEBUG{chemsec}{Cite: called Real}%
  \ChemFullLabelStyle{\CN@GetFullName{#1}}{\ChemMainCounterStyle{CN@LabelIndex}}{\CN@SubLabe
}%
```

`\CN@ChemSCite` Output label and sublabel for a given key. The name is never output.

```
\DeclareRobustCommand*{\CN@ChemSCite}[1]{%
  % \CN@DEBUG{chemsec}{in ChemShortCite}%
  \CN@RealCite{#1}{CN@KeyFound}{CN@LabelIndex}{\CN@SubLabel}%
  % \CN@DEBUG{chemsec}{SCite: called Real}%
  \ChemShortLabelStyle{\ChemMainCounterStyle{CN@LabelIndex}}{\CN@SubLabel}%
}%
```

`\CN@ChemMFCite` Output name associated with the `Main` part of the `Main!Sub` key as well as the label *and* sublabel.

```
\DeclareRobustCommand*{\CN@ChemMFCite}[1]{%
  % \CN@DEBUG{chemsec}{in ChemMainFullCite}%
  \CN@RealCite{\CN@MainKey(#1)}{CN@KeyFound}{CN@LabelIndex}{\CN@SubLabel}%
  % \CN@DEBUG{chemsec}{Cite: called Real}%
  \ChemFullLabelStyle{\CN@GetFullName{\CN@MainKey(#1)}}{\ChemMainCounterStyle{CN@LabelIndex}
}%
```

`\CN@ChemMSCite` Output the label for the `Main` part of the `Main!Sub` key.

```
\DeclareRobustCommand*{\CN@ChemMSCite}[1]{%
  % \CN@DEBUG{chemsec}{in ChemMainShortCite}%
```

```
  \CN@RealCite{\CN@MainKey{#1}}{CN@KeyFound}{CN@LabelIndex}{\CN@SubLabel}%
  % \CN@DEBUG{chemsec}{SCite: called Real}%
  \ChemShortLabelStyle{\ChemMainCounterStyle{CN@LabelIndex}}{\CN@SubLabel}%
}%
```

`\CN@ChemCiteStar` Set `CN@Star` boolean before calling base `\CN@ChemCite`.

```
\DeclareRobustCommand*{\CN@ChemCiteStar}[1]{%
  % \CN@DEBUG{chemsec}{in Chemcite}%
  \setboolean{CN@Star}{true}%
  \CN@ChemCite{#1}%
  \setboolean{CN@Star}{false}%
}
```

`\CN@ChemFCiteStar` Set `CN@Star` boolean before calling base `\CN@ChemFCite`.

```
\DeclareRobustCommand*{\CN@ChemFCiteStar}[1]{%
  % \CN@DEBUG{chemsec}{in ChemFullCite}%
  \setboolean{CN@Star}{true}%
  \CN@ChemFCite{#1}%
  \setboolean{CN@Star}{false}%
}%
```

`\CN@ChemSCiteStar` Set `CN@Star` boolean before calling base `\CN@ChemSCite`.

```
\DeclareRobustCommand*{\CN@ChemSCiteStar}[1]{%
  % \CN@DEBUG{chemsec}{in ChemShortCite}%
  \setboolean{CN@Star}{true}%
  \CN@ChemSCite{#1}%
  \setboolean{CN@Star}{false}%
}%
```

`\CN@ChemMFCiteStar` Set `CN@Star` boolean before calling base `\CN@ChemMFCite`.

```
\DeclareRobustCommand*{\CN@ChemMFCiteStar}[1]{%
  % \CN@DEBUG{chemsec}{in ChemMainFullCite}%
  \setboolean{CN@Star}{true}%
  \CN@ChemMFCite{#1}%
  \setboolean{CN@Star}{false}%
}%
```

`\CN@ChemMSCiteStar` Set `CN@Star` boolean before calling base `\CN@ChemMSCite`.

```
\DeclareRobustCommand*{\CN@ChemMSCiteStar}[1]{%
  % \CN@DEBUG{chemsec}{in ChemMainShortCite}%
  \setboolean{CN@Star}{true}%
  \CN@ChemMSCite{#1}%
  \setboolean{CN@Star}{false}%
}%
```

5.7 Public Interface Macros

`\ChemCite` Delegate to `\CN@ChemCiteStar` or `\CN@ChemCite` depending on whether "`*`" was used.

```
\DeclareRobustCommand*{\ChemCite}{%
  \@ifstar{\CN@ChemCiteStar}{\CN@ChemCite}%
}
```

`\ChemFCite` Delegate to `\CN@ChemFCiteStar` or `\CN@ChemFCite` depending on whether "`*`" was used.

```
\DeclareRobustCommand*{\ChemFCite}{%
  \@ifstar{\CN@ChemFCiteStar}{\CN@ChemFCite}%
}%
```

`\ChemSCite` Delegate to `\CN@ChemSCiteStar` or `\CN@ChemSCite` depending on whether "`*`" was used.

```
\DeclareRobustCommand*{\ChemSCite}{%
  \@ifstar{\CN@ChemSCiteStar}{\CN@ChemSCite}%
}%
```

`\ChemMFCite` Delegate to `\CN@ChemMFCiteStar` or `\CN@ChemMFCite` depending on whether "`*`" was used.

```
\DeclareRobustCommand*{\ChemMFCite}{%
  \@ifstar{\CN@ChemMFCiteStar}{\CN@ChemMFCite}%
}%
```

`\ChemMSCite` Delegate to `\CN@ChemMSCiteStar` or `\CN@ChemMSCite` depending on whether "`*`" was used.

```
\DeclareRobustCommand*{\ChemMSCite}{%
  \@ifstar{\CN@ChemMSCiteStar}{\CN@ChemMSCite}%
}%
```

`\DefineChemical` `{`⟨*main!sub key*⟩`}{`⟨*name*⟩`}{`⟨*label*⟩`}` Associate a key with a chemical name and sub-label by inserting a `\CN@definenewchemical` macro call into the `aux` file.

```
\DeclareRobustCommand*{\DefineChemical}[3]{%
  \CN@definenewchemical{#1}{#2}{#3}%
  \write\@auxout{\string\CN@definenewchemical{#1}{#2}{#3}}%
}%end def

\makeatother
```