The **alphalph** package

Heiko Oberdiek

<heiko.oberdiek at googlemail.com>

2016/05/16 v2.5

Abstract

The package provides methods to represent numbers with a limited set of symbols. Both \LaTeX{} and \TeX{} are supported.

Contents

1 Documentation 2

1.1 Introduction ........................................ 2

1.2 Use cases ........................................... 3

1.2.1 Number system based on symbols ................. 3

1.2.2 Wrap symbols around .............................. 3

1.2.3 Multiple symbols ................................ 3

1.3 Glossary ............................................. 4

1.4 Package usage ....................................... 4

1.5 User commands ...................................... 5

1.6 Programmer commands ............................... 5

1.7 Design principles ................................... 6

1.7.1 Number presentation commands ................. 6

1.7.2 General usability ............................... 6

2 Implementation 6

2.1 Begin of package ................................... 6

2.2 Catcodes .......................................... 7

2.3 Package loading .................................. 8

2.4 \epsilon-\TeX{} detection ........................ 8

2.5 Help macros ....................................... 9

2.6 Symbol provider ................................... 10

2.6.1 Alphabet ....................................... 10

2.7 Finding number of symbols ....................... 10

2.8 Methods ........................................... 12

2.8.1 Common methods ............................... 12

2.8.2 Method 'alph' .................................. 12

2.8.3 Method 'wrap' .................................. 13

2.8.4 Method 'mult' .................................. 14

2.9 User interface .................................... 15

3 Test 16

3.1 Catcode checks for loading ....................... 16

4 Macro tests ........................................... 18

*Please report any issues at https://github.com/ho-tex/oberdiek/issues
5 Installation

5.1 Download .................................................. 21
5.2 Bundle installation .................................... 22
5.3 Package installation ................................. 22
5.4 Refresh file name databases ...................... 22
5.5 Some details for the interested ............... 22

6 Catalogue

7 History ...................................................... 23

[1999/03/19 v0.1] ........................................ 23
[1999/04/12 v1.0] ........................................ 24
[1999/04/13 v1.1] ........................................ 24
[1999/06/26 v1.2] ........................................ 24
[2006/02/20 v1.3] ........................................ 24
[2006/05/30 v1.4] ........................................ 24
[2007/04/11 v1.5] ........................................ 24
[2007/09/09 v2.0] ........................................ 24
[2008/08/11 v2.1] ........................................ 24
[2010/03/01 v2.2] ........................................ 24
[2010/04/18 v2.3] ........................................ 24
[2011/05/13 v2.4] ........................................ 25
[2016/05/16 v2.5] ........................................ 25

8 Index ............................................................. 25

1 Documentation

1.1 Introduction

\LaTeX{} counters can be represented in different ways by using presentation commands:

\begin{itemize}
  \item \texttt{\arabic{}}
  \item \texttt{\roman{}}
  \item \texttt{\Roman{}}
  \item \texttt{\alph{}}
  \item \texttt{\Alph{}}
  \item \texttt{\fnsymbol{}}
\end{itemize}

The ranges of supported counter values are more or less restricted. Only \texttt{\arabic{}} can be used with any counter value \LaTeX{} supports.

<table>
<thead>
<tr>
<th>Presentation command</th>
<th>Supported domain</th>
<th>Ignored values</th>
<th>Error message</th>
</tr>
</thead>
<tbody>
<tr>
<td>\texttt{\arabic{}}</td>
<td>-MAX..MAX</td>
<td>-MAX..0</td>
<td>“Counter too large”</td>
</tr>
<tr>
<td>\texttt{\roman{}}</td>
<td>1..MAX</td>
<td>-MAX..0</td>
<td></td>
</tr>
<tr>
<td>\texttt{\Roman{}}</td>
<td>1..MAX</td>
<td>-MAX..0</td>
<td></td>
</tr>
<tr>
<td>\texttt{\alph{}}</td>
<td>1..26</td>
<td>0</td>
<td>-MAX..1, 27..MAX</td>
</tr>
<tr>
<td>\texttt{\Alph{}}</td>
<td>1..9</td>
<td>0</td>
<td>-MAX..1, 10..MAX</td>
</tr>
</tbody>
</table>

$MAX = 2147483647$

Ordinal numbers are often used in documents: numbering of chapters, sections, figures, footnotes and so on. The layouter chooses \texttt{\Alph{}} for chapter numbers and \texttt{\fnsymbol{}} for footnotes. But what can be done if there are more than 26 chapters or more than 10 footnotes? This package \texttt{alphalph} allows to define new presentation commands. They rely on a existing command and define presentations for values greater the limits. Three different methods are provided by the package. In the following use cases they are presented.
1.2 Use cases

1.2.1 Number system based on symbols

Assume you are writing a book and your lecturer demands that chapter numbers must be letters. But you have already 30 chapters and you have only 26 letters? In the decimal system the situation would be clear. If you run out of digits, you are using more digits to represent a number. This method can be also be used for letters. After chapter 26 with Z we use AA, AB, AC, and AD for the remaining chapters.

Happily this package already defines this presentation command:

\usepackage{alphalph}
\renewcommand*{\thechapter}{\AlphAlph{\value{chapter}}}
\AlphAlph generates: A, B, C, ..., Z, AA, AB, ...

The other presentation command is \alphalph for lowercase letters.

1.2.2 Wrap symbols around

Nine footnote symbols are quite a few. Too soon the symbols are consumed and \LaTeX{} complains with the error “Counter too large”. However, it could be acceptable to start again with the symbols from the beginning, especially if there are less than nine symbols on a page. This could be achieved by a counter reset. But finding the right place can be difficult or needs manual actions. Also a unique counter value can be desirable (e.g. for generating unique anchor/link names). Package alphalph allows you to define a macro that implements a “wrap around”, but letting the value of the counter untouched:

\usepackage{alphalph}
\makeatletter
\newalphalph{\fnsymbolwrap}{\@fnsymbol}\{\wrap\}
\makeatother
\renewcommand*{\thefootnote}{\fnsymbolwrap{\value{footnote}}}
\fnsymbolwrap generates: ∗ (1), † (2), ‡ (3), ..., ‡‡ (9), ∗ (10), † 11, ...

1.2.3 Multiple symbols

\LaTeX{}’s standard set of footnote symbols contains doubled symbols at the higher positions. Could this principle be generalized? Yes, but first we need a clean footnote symbol list without doubled entries, example:

\usepackage{alphalph}
\makeatletter
\newcommand*{\fnsymbolsingle}[1]{\ensuremath{\ifcase#1\or *\or \dagger\or \ddagger\or \mathsection\or \mathparagraph\else\@ctrerr\fi}}
\makeatother
\renewcommand*{\fnsymbolwrap}{\@fnsymbol\\value{footnote}}
\fnsymbolwrap generates: * (1), † (2), ‡ (3), ..., ‡‡ (9), * (10), † 11, ...

1.2.3 Multiple symbols

\LaTeX{}'s standard set of footnote symbols contains doubled symbols at the higher positions. Could this principle be generalized? Yes, but first we need a clean footnote symbol list without doubled entries, example:

\usepackage{alphalph}
\makeatletter
\newcommand*{\fnsymbolsingle}[1]{\ensuremath{\ifcase#1\or *\or \dagger\or \ddagger\or \mathsection\or \mathparagraph\else\@ctrerr\fi}}
\makeatother
\renewcommand*{\fnsymbolwrap}{\@fnsymbol\\value{footnote}}
\fnsymbolwrap generates: * (1), † (2), ‡ (3), ..., ‡‡ (9), * (10), † 11, ...

1.2.3 Multiple symbols

\LaTeX{}’s standard set of footnote symbols contains doubled symbols at the higher positions. Could this principle be generalized? Yes, but first we need a clean footnote symbol list without doubled entries, example:
The own definition of \fnsymbol{single} has the advantage that this list can easily be modified. Otherwise you can use \@fnsymbol directly, because it uses the same first five symbols.

\usepackage{alphalph}
\makeatletter
\newalphalph{\fnsymbol{mult}}{\@fnsymbol}{5}
\makeatother
\renewcommand*{\thefootnote}{\fnsymbol{mult}{\value{footnote}}}%

\fnsymbol{mult} generates: ∗ (1), † (2), ‡ (3), § (4), ¶ (5), ∗∗ (6), ..., **** 16, †††† 17, ...

The same method can also be used for the chapter problem in the first discussed use case:

\usepackage{alphalph}
\makeatletter
\newalphalph{\Alph{Mult}}{\@Alph}{26}
\makeatother
\renewcommand*{\chapter}{\Alph{Mult}{\value{chapter}}}%

\Alph{Mult} then generates AA, BB, CC, and DD for chapters 27–30.

1.3 Glossary

Counter presentation command is a macro that expects a \LaTeX{} counter name as argument. Numbers cannot be used. Examples: \arabic, \alph, \fnsymbol.

Number presentation command is a macro that expects a number as argument. A number is anything that \LaTeX{} accepts as number including \value. Examples: \alphalph, \AlphAlph, \alphalph@alph

However, \alph or \fnsymbol are not number presentation commands because they expect a counter name as argument. Happily \LaTeX{} counter presentation commands internally use number presentation commands with the same name, but prefixed by ‘@’. Thus \@alph, \@fnsymbol are number presentation commands.

Symbols provider is a command that can be used to get a list of symbols. For example, \@Alph provides the 26 uppercase letters from ‘A’ to ‘Z’. Basically a symbol provider is a number presentation command, usually with a limited range.

Number of symbols is the number of the last symbol slot of a symbol provider. Thus \@Alph generates 26 symbols, \@fnsymbol provides 9 symbols.

1.4 Package usage

The package alphalph can be used with both plain \TeX{} and \LaTeX{}:

plain \TeX{}: \input alphalph.sty
\LaTeX{} 2.0: \usepackage{alphalph}

There aren’t any options.
1.5 User commands

\AlphAlph \{\langle number\rangle\}
\alphalph \{\langle number\rangle\}

Both macros are number presentation commands that expects a number as argument. \LaTeX counters are used with \texttt{\value{}}.

The macros represents a number by letters. First single letters A..Z are used, then two letters AA..ZZ, three letters AAA...ZZZ, ...follow.

Macro \AlphAlph uses uppercase letters, \alphalph generates the lowercase variant.

\begin{tabular}{|c|c|c|}
\hline
\langle number\rangle & \AlphAlph\{\langle number\rangle\} & \alphalph\{\langle number\rangle\} \\
\hline
1 & A & a \\
2 & B & b \\
26 & Z & z \\
27 & AA & aa \\
30 & AD & ad \\
2000 & BXX & bxx \\
3752127 & HELLO & hello \\
10786572 & WORLD & world \\
2147483647 & FXSHRXW & fxshrxw \\
\hline
\end{tabular}

\newalphalph \{\langle cmd\rangle\} \{\langle method\rangle\} \{\langle symbols provider\rangle\} \{\langle number of symbols\rangle\}

Macro \newalphalph defines \langle cmd\rangle as new number presentation command. Like \texttt{\newcommand} an error is thrown, if macro \langle cmd\rangle already exists.

The \langle method\rangle is one of \texttt{alph}, \texttt{wrap}, or \texttt{mult}. The default is \texttt{alph}.

As symbol provider a number presentation command can be used, e.g. \texttt{\@fn-symbol}, \texttt{\@Alph}, or \texttt{\alphalph@alph}.

The last argument is the number of symbols. If the argument is empty, then \newalphalph tries to find this number itself. \LaTeX’s number presentation commands throw an error message, if the number is too large. This error message is put in a macro \texttt{\@ctrerr}. Thus \newalphalph calls the symbol provider and tests a number by typesetting it in a temporary box. The error macro \texttt{\@ctrerr} is catched, it proofs that the number is not supported. Also if the width of the result is zero the number is considered as unavailable.

The empty argument is useful for potentially variable lists. However if the end cannot be detected, then the number of symbols must be given. This is also a lot faster. Therefore don’t let the argument empty without reason.

1.6 Programmer commands

\alphalph@Alph \{\langle number\rangle\}
\alphalph@alph \{\langle number\rangle\}

They are basically the same as \texttt{\@Alph} and \texttt{\@alph}. Some languages of package \texttt{babel} redefine \LaTeX’s macros to include some font setup that breaks expandibility. Therefore \texttt{\AlphAlph} and \texttt{\alphalph} are based on \texttt{\alphalph@Alph} and \texttt{\alphalph@alph} to get the letters. The behaviour of these symbol providers for numbers outside the range 1..26 is undefined.
1.7 Design principles

1.7.1 Number presentation commands

All number presentation commands that this package defines (including \alphalph and \AlphAlph) have the following properties:

- They are fully expandable. This means that they can safely
  - be written to a file,
  - used in moving arguments (\LaTeX{}: they are robust),
  - used in a \csname \endcsname pair.

- If the argument is zero or negative, the commands expand to nothing like \romannumeral.

- The argument is a \TeX{} number. Anything that would be accepted by \number is a valid argument:
  - explicite constants,
  - macros that expand to a number,
  - count registers, \LaTeX{} counter can used via \value, e. g.:
    \alphalph{\value{page}}
  - ...

- \eTeX{}’s numeric expressions are supported, if \eTeX{} is available. Then \numexpr is applied to the argument. Package \calc’s expressions are not supported. That would violate the expandibility.

1.7.2 General usability

\TeX{} format: The package does not depend on \LaTeX{}, it can also be used by plain \TeX{}, for example.

\eTeX{}: \eTeX{} is supported, the macros are shorter and faster. But \eTeX{}’s extensions are not requirements. Without \eTeX{}, just the implementation changes. The properties remain unchanged.

2 Implementation

2.1 Begin of package

1 (*package*)

Reload check, especially if the package is not used with \LaTeX{}.

2 \begingroup\catcode61\catcode48\catcode32=10\relax%
3 \catcode13=5 % ^^M
4 \endlinechar=13 %
5 \catcode35=6 % #
6 \catcode39=12 % '
7 \catcode44=12 % ,
8 \catcode45=12 % -
9 \catcode46=12 % .
10 \catcode58=12 % :
11 \catcode64=11 % @
12 \catcode123=1 % {
13 \catcode125=2 % }
14 \expandafter\let\expandafter\x\csname ver@alphalph.sty\endcsname
15 \ifx\x\relax % plain-\TeX{}, first loading
16 \else
17 \def\empty{}%
2.2 Catcodes

\begingroup\catcode13=10\relax% 
\catcode13=9 % 🔼
\def\AlPh@@AtEnd{%
\expandafter\edef\csname AlPh@AtEnd\endcsname{%

\expandafter\ifx\csname ProvidesPackage\endcsname%
2.3 Package loading

2.4 $\varepsilon$-TEX detection
2.5 Help macros

\AlPh@Error
133 \def\AlPh@Error#1{%
134 \begingroup
135 \escapechar=92 % backslash
136 \@PackageError{alphalph}{#1}\@ehc
137 \endgroup
138 }

\AlPh@IfDefinable
139 \begingroup\expandafter\expandafter\expandafter\endgroup
140 \expandafter\ifx\csname @ifdefinable\endcsname\relax
141 \def\AlPh@IfDefinable#1#2{%
142 \ifcase#1\@undefined\else#2\relax\else1\fi\fi0 %
143 #2%
144 \else
145 \AlPh@Error{%
146 Command \string#1 already defined%
147 }%
148 \fi
149 }%
150 \else

\AlPh@IfDefinable
151 \let\AlPh@IfDefinable@ifdefinable
152 \fi

\@ReturnAfterElseFi
153 \long\def\@ReturnAfterElseFi#1\else#2\fi{
154 \fi#1}

\@ReturnAfterFi
155 \long\def\@ReturnAfterFi#1\fi{
156 \fi#1}

\@gobblefour
157 \@gobblefour
158 \AlPh@IfOptArg
159 \begingroup\expandafter\expandafter\expandafter\endgroup
160 \expandafter\ifx\csname kernel@ifnextchar\endcsname\relax
161 \def\AlPh@IfOptArg#1#2{%
162 \def\AlPh@TempA{#1}%
163 \def\AlPh@TempB{#2}%
164 \futurelet\AlPh@Token\AlPh@IfOptArgNext
165 %
166 }%
167 \let\AlPh@BracketLeft=[%}
168 \def\AlPh@IfOptArgNext{%
169 \ifx\AlPh@Token\AlPh@BracketLeft
170 \expandafter\AlPh@TempA
171 \else
172 \expandafter\AlPh@TempB
173 \fi
174 }%
2.6 Symbol provider

2.6.1 Alphabet

The output of `\alphalph` and `\AlphAlph` should be usable as part of command names (see `\@namedef`, `\csname`, ...). Unhappily some languages of package `babel` redefine LaTeX's `\@alph` and `\@Alph` in a manner that they cannot be used in expandable context any more. Therefore package `alphalph` provides its own commands.

The two commands `\AlPh@Alph` and `\AlPh@alph` convert a number into a letter (uppercase and lowercase respectively). The character `@` is used as an error symbol, if the number isn’t in the range of 1 until 26. Here we need no space after the number `#1`, because the error symbol `@` for the zero case stops scanning the number. This error symbol should not appear anywhere (except for bugs).

```
\def\alphalph@Alph#1{\ifcase#1\or A\or B\or C\or D\or E\or F\or G\or H\or I\or J\or K\or L\or M\or N\or O\or P\or Q\or R\or S\or T\or U\or V\or W\or X\or Y\or Z\else\AlPh@ctrerr\@\fi}
\def\alphalph@alph#1{\ifcase#1\or a\or b\or c\or d\or e\or f\or g\or h\or i\or j\or k\or l\or m\or n\or o\or p\or q\or r\or s\or t\or u\or v\or w\or x\or y\or z\else\AlPh@ctrerr\@\fi}
```

Macro `\AlPh@ctrerr` is used as hook for the algorithm to get the available number of symbols.

2.7 Finding number of symbols

```
\def\AlPh@GetNumberOfSymbols{\ifAlPh@Unavailable\def\AlPh@Number{0}\AlPh@Error{No symbols found}\else\def\AlPh@Number{1}\AlPh@ExpSearch2\fi}
```

`\AlPh@GetNumberOfSymbols` returns the number of symbols provided by the package.
\ifAlPh@Unavailable
212 \let\ifAlPh@Unavailable\iffalse
213 \def\AlPh@Unavailabletrue{%
214 \global\let\ifAlPh@Unavailable\iftrue
215 }% 
216 \def\AlPh@Unavailablefalse{%
217 \global\let\ifAlPh@Unavailable\iffalse
218 }
\fi
\AlPh@TestNumber
#1: number to be tested
#2: symbols provider
219 \def\AlPh@TestNumber#1!#2{%  
220 \AlPh@Unavailablefalse
221 \begingroup
222 \setbox0=\hbox{\begingroup % color
223 \let\@ctrerr\AlPh@Unavailabletrue
224 \let\AlPh@ctrerr\AlPh@Unavailabletrue
225 \let\AlPh@next\relax
226 #2(#1)%
227 \endgroup
228 \endgroup}
229 \ifdim\wd0=0pt
230 \AlPh@Unavailabletrue
231 \else
232 \AlPh@Error{% Maximal symbol number not found%
233 }
234 \fi
235 \AlPh@Next
236 \fi
\AlPh@ExpSearch
#1: number to be tested
#2: symbols provider
238 \expandafter\AlPh@ProcessExpSearch\number\intcalcShl{#1}!#2!{#3}%
239 \else
240 \def\AlPh@Number{#1}%
241 \ifnum#1>1073741823%
242 \AlPh@TestNumber2147483647!{#2}%
243 \ifAlPh@Unavailable
244 \AlPh@BinSearch#1!2147483647!{#2}%
245 \else
246 \def\AlPh@Number{0}%
247 \AlPh@Error{% Maximal symbol number not found%
248 }
249 \fi
250 \fi
251 \else
252 \def\AlPh@Next{}%
253 \expandafter\AlPh@BinSearch\number\intcalcShr{#1}!{#2}!
254 \fi
255 \fi
256 \AlPh@Next
257 \fi
\AlPh@BinSearch
#1: available number
#2: unavailable number, #2 > #1
#3: symbols provider
259 \def\AlPh@BinSearch#1!#2!#3{%  
260 \expandafter\AlPh@ProcessBinSearch
261 \number\intcalcShr{\intcalcAdd(#1)(#2)}% 
262 #1!#2!#3%
\AlPh@ProcessBinSearch \#1: number to be tested, \#2 ≤ \#1 ≤ \#3
\#2: available number
\#3: unavailable number
\#4: symbols provider
\def\AlPh@ProcessBinSearch{\ifnum\#1>\#2 %
  \ifnum\#1<\#3 %
    \AlPh@TestNumber\#1{\#4}%
    \if\AlPh@Unavailable
      \def\AlPh@Next{\AlPh@BinSearch\#2!\#1!{\#4}}%
    \else
      \def\AlPh@Next{\AlPh@BinSearch\#1!\#3!{\#4}}%
    \fi
  \else
    \def\AlPh@Number{\#2}%
  \fi
\else
  \def\AlPh@Number{\#2}%
\fi
\AlPh@Next}

2.8 Methods

The names of method macros start with \AlPh@Method. These macros do the main job in converting a number to its representation. A method command is called with three arguments. The first argument is the number of symbols. The second argument is the basic macro for converting a number with limited number range. The last parameter is the number that needs converting.

2.8.1 Common methods
\AlPh@CheckPositive \#1: number to be checked \#2: continuation macro
\#3: number of symbols (hidden here)
\#4: symbol provider (hidden here)
\def\AlPh@CheckPositive{\ifnum\#1<1 %
  \expandafter\@gobblefour
\fi
\#2{\#1}%
\}

2.8.2 Method ‘alph’
\AlPh@Method@alph \#1: number of symbols
\#2: symbols provider
\#3: number to be converted
\def\AlPh@Method@alph{\expandafter\AlPh@CheckPositive
  | \number\#3%
  + \the\numexpr\#3%
  \AlPh@ProcessAlph
  \{\#1\}(\#2)%
  \}

12
\AlPh@ProcessAlph
\#1: current number
\#2: number of symbols
\#3: symbols provider
\AlPh@Method@wrap
\#1: number of symbols
\#2: symbols provider
\#3: number to be converted
\AlPh@ProcessWrap
\#1: number to be converted
\#2: number of symbols
\#3: symbols provider
2.8.4 Method ‘mult’

After the number of symbols is exhausted, repetitions of the symbol are used.

\[ x := \text{number to be converted} \]
\[ n := \text{number of symbols} \]
\[ r := \text{repetition length} \]
\[ s := \text{symbol slot} \]
\[ r = \left( \frac{x - 1}{n} \right) + 1 \]
\[ s = \left( x - 1 \right) \mod n + 1 \]
2.9 User interface

Macro \newalphalph had three arguments in versions below 2.0. For the new method argument we use an optional argument an first position.

#1: cmd
[#2]: method name: alph (default), wrap, mult
hash-ok #3: symbols provider
#4: number of symbols

\AlPh@ifDefinable\newalphalph{%}
  \def\newalphalph#1{%}
  \AlPh@ifOptArg{%
    \AlPh@newalphalph{#1}%
  }{%
    \AlPh@newalphalph{#1}[alph]%
  }%
}%
}

\AlPh@newalphalph #1: cmd #2: method name
#3: symbols provider
#4: number of symbols
\def\AlPh@newalphalph#1[#2]#3#4{%
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname AlPh@Method@#2\endcsname\relax
    \AlPh@Error{%
      Unknown method %
    }%
  \else
    \ifx\#4\%
      \AlPh@GetNumberOfSymbols{#3}%
    \fi
    \expandafter\AlPh@NewAlphAlph
    \csname AlPh@Method@#2\expandafter\endcsname
    \AlPh@Number!{#1}{#3}%
  \fi
}%

\AlPh@NewAlphAlph #1: method macro
#2: number of symbols
#3: cmd
#4: symbols provider
\def\AlPh@NewAlphAlph#1#2!#3#4{%
  \AlPh@ifDefinable#3{%
    \ifnum#2>0 %
      \def#3{#1{#2}{#4}}%
    \fi
  }%
}
\else
  \AlPh@Error{%
  Definition of \string#3 failed,\MessageBreak
  because number of symbols (#2) is not positive%
}\fi}
}

\AlphAlph
\newalphalph\AlphAlph\alphalph@Alph{26}

\alphalph
\newalphalph\alphalph\alphalph@alph{26}
\AlPh@AtEnd%
(/package)

3 Test

3.1 Catcode checks for loading
\(*test1*
\catcode\{=1 
\catcode\}=2 
\catcode\#=6 
\catcode\@=11 
\expandafter\ifx\csname count@\endcsname\relax
\countdef\count@=255 
\fi
\expandafter\ifx\csname @gobble\endcsname\relax
\long\def\@gobble#1{}% 
\fi
\expandafter\ifx\csname @firstofone\endcsname\relax
\long\def\@firstofone#1{#1}% 
\fi
\expandafter\ifx\csname loop\endcsname\relax
\expandafter\@firstofone
\else
\expandafter\@gobble
\fi
\else
\expandafter\@gobble
\fi
(%
\def\loop#1\repeat{%
  \def\body{#1}%
\iterate
}%
\def\iterate{%
  \body
\let\next\iterate
\else
\let\next\relax
\fi
\next
}%
\let\repeat=\fi
}%
\def\RestoreCatcodes{}
\count@=0 
\loop
\edef\RestoreCatcodes{%
\RestoreCatcodes
}
\catcode\the\count@=\the\catcode\count@\relax
\ifnum\count@<255
\advance\count@ 1
\repeat
\def\RangeCatcodeInvalid#1#2{\count@=#1\relax
\loop\catcode\count@=15
\ifnum\count@<#2\relax
\advance\count@ 1
\repeat
\def\RangeCatcodeCheck#1#2#3{\count@=#1\relax
\loop
\ifnum#3=\catcode\count@
\else
\errmessage{Character \the\count@ space with wrong catcode \the\catcode\count@ space instead of \number#3}%\fi
\ifnum\count@<#2\relax
\advance\count@ 1
\repeat
\def\space{ }\expandafter\ifx\csname LoadCommand\endcsname\relax
\def\LoadCommand{\input alphalph.sty}\relax\fi
\def\Test{\RangeCatcodeInvalid{0}{47}\
\RangeCatcodeInvalid{58}{64}\
\RangeCatcodeInvalid{91}{96}\
\catcode`\@=12 \catcode`\\=0 \catcode`\%=14 \LoadCommand
\RangeCatcodeCheck{0}{36}{15}\
\RangeCatcodeCheck{37}{37}{14}\
\RangeCatcodeCheck{38}{47}{15}\
\RangeCatcodeCheck{48}{57}{12}\
\RangeCatcodeCheck{58}{63}{15}\
\RangeCatcodeCheck{64}{64}{12}\
\RangeCatcodeCheck{65}{90}{11}\
\RangeCatcodeCheck{91}{91}{15}\
\RangeCatcodeCheck{92}{92}{0}\
\RangeCatcodeCheck{93}{96}{15}\
\RangeCatcodeCheck{97}{122}{11}\
\RangeCatcodeCheck{123}{255}{15}\
\RestoreCatcodes
}\Test\csname @@end\endcsname\relax
\end ⟨/test1⟩
4  Macro tests

\NeedsTeXFormat{LaTeX2e}
\nofiles
\documentclass{article}
\makeatletter
\let\saved@numexpr=\numexpr
\newcommand*{\DisableNumexpr}{\let\numexpr\@undefined}
\newcommand*{\RestoreNumexpr}{\let\numexpr\saved@numexpr}
\DisableNumexpr
\usepackage{alphalph}\[2016/05/16\]
\usepackage{qstest}
\IncludeTests{*}
\LogTests{log}{*}{*}
\newcommand*{\TestCmd}[3]{\setbox0=\hbox{\DisableNumexpr\edef\TestString{#1{#2}}\expandafter\Expect\expandafter{\TestString}{#3}\edef\TestString{#1{#2} }\expandafter\Expect\expandafter{\TestString}{#3 }}\Expect*{\the\wd0}{0.0pt}}
\makeatletter
\newalphalph\LaTeXAlphAlph\@Alph{26}
\newalphalph\LaTeXalphalph\@alph{26}
\newalphalph\AlphWrap[wrap]\alphalph@Alph{26}
\newalphalph\alphwrap[wrap]\alphalph@alph{26}
\newalphalph\LaTeXAlphWrap[wrap]\@Alph{26}
\newalphalph\LaTeXalphwrap[wrap]\@alph{26}
\def\LastSymbol#1{\ifx\#1\%\else\@LastSymbol#1\@nil\fi}
\def\@LastSymbol#1#2\@nil{\ifx\#2\%\#1\else\@LastSymbol#2\@nil\fi}
\makeatother
\newcommand*{\TestAlph}[2]{\uppercase{\TestCallCmd\AlphAlph{#2}}{#1}\lowercase{\TestCallCmd\alphalph{#2}}{#1}\uppercase{\TestCallCmd\LaTeXAlphAlph{#2}}{#1}\lowercase{\TestCallCmd\LaTeXalphalph{#2}}{#1}\edef\WrapString{\LastSymbol{#2}}\expandafter\TestAlphWrap\expandafter{\WrapString}{#1}}
\makeatletter
\newcommand*{\TestAlph}[2]{\Ifx\#1\%\else\@LastSymbol\#1\@nil\fi\def\@LastSymbol#1#2\@nil{\ifx\#2\%\#1\else\@LastSymbol#2\@nil\fi}\\makeatother
\newcommand*{\TestAlph}[2]{\upper\csname TestAlph\endcsname{#2}{#1}\lower\csname TestAlph\endcsname{#2}{#1}\upper\csname TestAlph\endcsname{#2}{#1}\lower\csname TestAlph\endcsname{#2}{#1}\edef\WrapString{\LastSymbol{#2}}\expandafter\TestAlphWrap\expandafter{\WrapString}{#1}}
\makeatletter
\newcommand*{\TestAlph}[2]{\Ifx\#1\%\else\@LastSymbol\#1\@nil\fi\def\@LastSymbol#1#2\@nil{\ifx\#2\%\#1\else\@LastSymbol#2\@nil\fi}\\makeatother
\newcommand*{\TestAlph}[2]{\upper\csname TestAlph\endcsname{#2}{#1}\lower\csname TestAlph\endcsname{#2}{#1}\upper\csname TestAlph\endcsname{#2}{#1}\lower\csname TestAlph\endcsname{#2}{#1}\edef\WrapString{\LastSymbol{#2}}\expandafter\TestAlphWrap\expandafter{\WrapString}{#1}}
\newcommand*{\TestAlphWrap}[2]{%
  \uppercase{\TestCallCmd{AlphWrap}{#1}{#2}}%
  \lowercase{\TestCallCmd{alphwrap}{#1}{#2}}%
\}
\newcommand*{\TestCallCmd}[3]{%
  \TestCmd{#1}{#3}{#2}%
\}

\begin{qstest}{AlphSymbols}{alphalph, AlphAlph, symbols}
\TestAlph{1}{a}%
\TestAlph{2}{b}%
\TestAlph{3}{c}%
\TestAlph{4}{d}%
\TestAlph{5}{e}%
\TestAlph{6}{f}%
\TestAlph{7}{g}%
\TestAlph{8}{h}%
\TestAlph{9}{i}%
\TestAlph{10}{j}%
\TestAlph{11}{k}%
\TestAlph{12}{l}%
\TestAlph{13}{m}%
\TestAlph{14}{n}%
\TestAlph{15}{o}%
\TestAlph{16}{p}%
\TestAlph{17}{q}%
\TestAlph{18}{r}%
\TestAlph{19}{s}%
\TestAlph{20}{t}%
\TestAlph{21}{u}%
\TestAlph{22}{v}%
\TestAlph{23}{w}%
\TestAlph{24}{x}%
\TestAlph{25}{y}%
\TestAlph{26}{z}%
\end{qstest}

\begin{qstest}{AlphRange}{alphalph, range}
\TestAlph{0}{}
\TestAlph{-1}{}
\TestAlph{-2147483647}{}
\TestAlph{27}{aa}%
\TestAlph{28}{ab}%
\TestAlph{52}{az}%
\TestAlph{53}{ba}%
\TestAlph{78}{bz}%
\TestAlph{79}{ca}%
\TestAlph{702}{zz}%
\TestAlph{703}{aaa}%
\TestAlph{2147483647}{fxshrxw}%
\end{qstest}

\makeatletter
\newcommand*{\myvocals}[1]{%
  \ifcase#1X\or A\or E\or I\or O\or U\else Y\fi
}\makeatother
\newalphalph{vocalsvocals}{\myvocals}{5}
\newcommand*{\TestVocals}{%
  \TestCmd{vocalsvocals}%
}\begin{qstest}{vocals}{vocals}
\end{qstest}

\makeatletter
\newcommand*{\myvocals}[1]{%
  \ifcase#1X\or A\or E\or I\or O\or U\else Y\fi
}\makeatother
\newalphalph{vocalsvocals}{\myvocals}{5}
\newcommand*{\TestVocals}{%
  \TestCmd{vocalsvocals}%
}\begin{qstest}{vocals}{vocals}
\end{qstest}
\TestVocals{} \TestVocals{1}{A} \TestVocals{2}{E} \TestVocals{3}{I} \TestVocals{4}{O} \TestVocals{5}{U} \TestVocals{6}{AA} \TestVocals{7}{AE} \TestVocals{8}{AI} \TestVocals{9}{AO} \TestVocals{10}{AU} \TestVocals{11}{EA} \TestVocals{12}{OU} \TestVocals{13}{UA} \TestVocals{14}{UO} \TestVocals{15}{UU} \TestVocals{16}{AAA} \TestVocals{17}{AAAA} \TestVocals{18}{AIIOOEEIOIIUOE} \\end{qstest}\def\myvocalsB#1{%\ifcase#1\or A\or E\or I\or O\or U\fi\makeatletter\newalphalph\AlphMult[mult]{\alphalph@Alph}{26}\newalphalph\alphmult[mult]{\alphalph@alph}{26}\newalphalph\LaTeXAlphMult[mult]{\@Alph}{26}\newalphalph\LaTeXalphmult[mult]{\@alph}{26}\makeatother\newcommand*{\Test#1#2}{%\let\TestCmd\relax\newalphalph\TestCmd{#1}{#2}\Expect*{\AlPh@Number}{#2}}\\begin{qstest}{symbolnum}{symbolnum}\begin{qtest}{mult}{mult}\begin{qtest}{}{}%
5 Installation

5.1 Download

Package. This package is available on CTAN\(^1\):


Bundle. All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

\(^1\)http://ctan.org/pkg/alphalph
TDS refers to the standard “A Directory Structure for \TeX Files” (CTAN:tds/tds.pdf). Directories with \texttt{texmf} in their name are usually organized this way.

5.2 Bundle installation

Unpacking. Unpack the \texttt{oberdiek.tds.zip} in the TDS tree (also known as \texttt{texmf} tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

Script installation. Check the directory TDS:scripts/oberdiek/ for scripts that need further installation steps. Package \texttt{attachfile2} comes with the Perl script \texttt{pdfatfi.pl} that should be installed in such a way that it can be called as \texttt{pdfatfi}. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

5.3 Package installation

Unpacking. The \texttt{.dtx} file is a self-extracting \texttt{docstrip} archive. The files are extracted by running the \texttt{.dtx} through plain \TeX:

```
tex alphalph.dtx
```

TDS. Now the different files must be moved into the different directories in your installation TDS tree (also known as \texttt{texmf} tree):

```
alphalph.sty  → tex/generic/oberdiek/alphalph.sty
alphalph.pdf → doc/latex/oberdiek/alphalph.pdf
test/alphalph-test1.tex → doc/latex/oberdiek/test/alphalph-test1.tex
test/alphalph-test2.tex → doc/latex/oberdiek/test/alphalph-test2.tex
test/alphalph-test3.tex → doc/latex/oberdiek/test/alphalph-test3.tex
alphalph.dtx → source/latex/oberdiek/alphalph.dtx
```

If you have a \texttt{docstrip.cfg} that configures and enables \texttt{docstrip}’s TDS installing feature, then some files can already be in the right place, see the documentation of \texttt{docstrip}.

5.4 Refresh file name databases

If your \TeX distribution (\texttt{tex\TeX}, \texttt{mik\TeX}, ...) relies on file name databases, you must refresh these. For example, \texttt{tex\TeX} users run \texttt{texhash} or \texttt{mktexlsr}.

5.5 Some details for the interested

Unpacking with \La\TeX. The \texttt{.dtx} chooses its action depending on the format:

\texttt{\La\TeX}: Run \texttt{docstrip} and extract the files.

\texttt{\La\TeX}: Generate the documentation.

If you insist on using \La\TeX for \texttt{docstrip} (really, \texttt{docstrip} does not need \La\TeX), then inform the autodetect routine about your intention:

```
l latex \let\install=y\input{alphalph.dtx}
```

Do not forget to quote the argument according to the demands of your shell.
Generating the documentation. You can use both the .dtx or the .drv to generate the documentation. The process can be configured by the configuration file \texttt{ltxdoc.cfg}. For instance, put this line into this file, if you want to have A4 as paper format:

\PassOptionsToClass{a4paper}{article}

An example follows how to generate the documentation with \texttt{pdf\LaTeX}:

\begin{verbatim}
pdflatex alphalph.dtx
makeindex -s gind.ist alphalph.idx
pdflatex alphalph.dtx
makeindex -s gind.ist alphalph.idx
pdflatex alphalph.dtx
\end{verbatim}


6 Catalogue

The following XML file can be used as source for the \LaTeX Catalogue. The elements \texttt{caption} and \texttt{description} are imported from the original XML file from the Catalogue. The name of the XML file in the Catalogue is \texttt{alphalph.xml}.

\begin{verbatim}
<?xml version='1.0' encoding='us-ascii'?>
<!DOCTYPE entry SYSTEM 'catalogue.dtd'>
<entry datestamp='$Date$' modifier='$Author$' id='alphalph'>
  <name>alphalph</name>
  <caption>Convert numbers to letters.</caption>
  <authorref id='auth:oberdiek'/>
  <license type='lppl1.3'/>
  <version number='2.5'/>
  <description>
    Provides commands \texttt{$\alphalph$} and \texttt{$\AlphAlph.$}. They are like \texttt{$\number$} but the expansion consists of lowercase and uppercase letters respectively (1 to a, 26 to z, 27 to aa, 52 to zz, 53 to ba, 702 to zz, 703 to aaa, etc.). Can be used as a replacement for \LaTeX's \texttt{$\@alph$} and \texttt{$\@Alph$} macros.
  </description>
  <documentation details='Package documentation' href='ctan:/macros/latex/contrib/oberdiek/alphalph.pdf'/>
  <ctan file='true' path='/macros/latex/contrib/oberdiek/alphalph.dtx'/>
  <miktex location='oberdiek'/>
  <texlive location='oberdiek'/>
  <install path='/macros/latex/contrib/oberdiek/oberdiek.tds.zip'/>
</entry>
\end{verbatim}

7 History

[1999/03/19 v0.1]

- The first version was built as a response to a question\textsuperscript{2} of Will Douglas\textsuperscript{3} and the request\textsuperscript{4} of Donald Arsenau\textsuperscript{5}, published in the newsgroup \texttt{comp.text.tex}: “Re: alph counters > 26”\textsuperscript{6}

\begin{footnotesize}
\begin{itemize}
\item Url: \url{http://groups.google.com/group/comp.text.tex/msg/17a74cd721641038}
\item Will Douglas's email address: william.douglas@wolfson.ox.ac.uk
\item Url: \url{http://groups.google.com/group/comp.text.tex/msg/8f9768825640315f}
\item Donald Arsenau's email address: asnd@reg.triumf.ca
\item Url: \url{http://groups.google.com/group/comp.text.tex/msg/cee563ee8bf65d0}
\end{itemize}
\end{footnotesize}
• Copyright: LPPL (CTAN:macros/latex/base/lppl.txt)

[1999/04/12 v1.0]
• Documentation added in dtx format.
• \TeX support added.

[1999/04/13 v1.1]
• Minor documentation change.
• First CTAN release.

[1999/06/26 v1.2]
• First generic code about \ProvidesPackage improved.
• Documentation: Installation part revised.

[2006/02/20 v1.3]
• Reload check (for plain \TeX)
• New DTX framework.
• LPPL 1.3

[2006/05/30 v1.4]
• \newalphalph added.

[2007/04/11 v1.5]
• Line ends sanitized.

[2007/09/09 v2.0]
• New implementation that uses package \intcalc. This removes the dependency on \TeX.
• \newalphalph is extended to support new methods ‘wrap’ and ‘multi’.
• Documentation rewritten.

[2008/08/11 v2.1]
• Code is not changed.
• URLs updated from www.dejanews.com to groups.google.com.

[2010/03/01 v2.2]
• Compatibility with ini\TeX.

[2010/04/18 v2.3]
• Documentation fixes (Martin Münch).
[2011/05/13 v2.4]
- Documentation fixes (Jim Diamond) and using package hologo for the documentation.
- Catalogue file added.

[2016/05/16 v2.5]
- Documentation updates.

8 Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; plain numbers refer to the code lines where the entry is used.

<table>
<thead>
<tr>
<th>Symbols</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>#</td>
<td>435</td>
</tr>
<tr>
<td>%</td>
<td>511</td>
</tr>
<tr>
<td>@</td>
<td>436, 509</td>
</tr>
<tr>
<td>@Alph</td>
<td>564, 568, 680, 718, 778</td>
</tr>
<tr>
<td>@LastSymbol</td>
<td>573, 576, 580</td>
</tr>
<tr>
<td>@PackageError</td>
<td>136</td>
</tr>
<tr>
<td>@PackageInfo</td>
<td>399</td>
</tr>
<tr>
<td>@ReturnAfterElseFi</td>
<td>153, 301, 329, 352</td>
</tr>
<tr>
<td>@ReturnAfterFi</td>
<td>153, 311, 335, 361, 369</td>
</tr>
<tr>
<td>@alpha</td>
<td>565, 569, 681, 717, 778</td>
</tr>
<tr>
<td>@ctrerr</td>
<td>224</td>
</tr>
<tr>
<td>@ehc</td>
<td>136</td>
</tr>
<tr>
<td>@firstofone</td>
<td>444, 447</td>
</tr>
<tr>
<td>@fnsymbol</td>
<td>719</td>
</tr>
<tr>
<td>@gobble</td>
<td>441, 449</td>
</tr>
<tr>
<td>@gobblefour</td>
<td>155, 288</td>
</tr>
<tr>
<td>@ifdefinable</td>
<td>151</td>
</tr>
<tr>
<td>@ifnextchar</td>
<td>176</td>
</tr>
<tr>
<td>@nil</td>
<td>573, 576, 580</td>
</tr>
<tr>
<td>@onelevel@sanitize</td>
<td>739, 744</td>
</tr>
<tr>
<td>@undefined</td>
<td>58, 142, 539</td>
</tr>
<tr>
<td>\</td>
<td>203, 219, 230, 242, 268</td>
</tr>
<tr>
<td>\AlPh@newalphalph</td>
<td>378, 380, 384</td>
</tr>
<tr>
<td>\AlPh@Next</td>
<td></td>
</tr>
<tr>
<td>\AlPh@Number</td>
<td>205, 208, 238, 240, 246, 279, 282, 395, 400, 405, 715</td>
</tr>
<tr>
<td>\AlPh@ProcessAlph</td>
<td>296, 299, 317</td>
</tr>
<tr>
<td>\AlPh@ProcessBinSearch</td>
<td>260, 264</td>
</tr>
<tr>
<td>\AlPh@ProcessErr</td>
<td>347, 350</td>
</tr>
<tr>
<td>\AlPh@ProcessMult</td>
<td>324, 327</td>
</tr>
<tr>
<td>\AlPh@ProcessWrap</td>
<td>302, 316</td>
</tr>
<tr>
<td>\AlPh@TempA</td>
<td>163, 170</td>
</tr>
<tr>
<td>\AlPh@TempB</td>
<td>164, 172</td>
</tr>
<tr>
<td>\AlPh@TestNumber</td>
<td></td>
</tr>
<tr>
<td>\AlPh@Token</td>
<td></td>
</tr>
<tr>
<td>\AlPh@Unavailablefalse</td>
<td>216, 220</td>
</tr>
<tr>
<td>\AlPh@Unavailabletrue</td>
<td>213, 224, 225, 230</td>
</tr>
<tr>
<td>\AlphAlph</td>
<td>5, 428, 585, 774</td>
</tr>
<tr>
<td>alphalph</td>
<td>429, 586, 774</td>
</tr>
<tr>
<td>alphalph@Alph</td>
<td>5, 181, 428, 566, 678, 722</td>
</tr>
<tr>
<td>alphalph@alph</td>
<td>181, 429, 567, 679, 721</td>
</tr>
<tr>
<td>AlphMult</td>
<td>678, 684</td>
</tr>
<tr>
<td>alphmult</td>
<td>679, 685</td>
</tr>
<tr>
<td>AlphWrap</td>
<td>566, 593</td>
</tr>
<tr>
<td>alphwrap</td>
<td>567, 594</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>A</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>\advance</td>
<td>474, 482, 497</td>
</tr>
<tr>
<td>\aftergroup</td>
<td>29</td>
</tr>
<tr>
<td>\AlPh@AtEnd</td>
<td>95, 96, 116, 430</td>
</tr>
<tr>
<td>\AlPh@BinSearch</td>
<td>238, 244, 259, 271, 275</td>
</tr>
<tr>
<td>\AlPh@BracketLeft</td>
<td>167, 169</td>
</tr>
<tr>
<td>\AlPh@CheckPositive</td>
<td>286, 293, 321, 344</td>
</tr>
<tr>
<td>\AlPh@Ctterr</td>
<td>187, 197, 201, 225</td>
</tr>
<tr>
<td>\AlPh@Error</td>
<td>133, 145, 206, 247, 387, 421</td>
</tr>
<tr>
<td>\AlPh@ExpSearch</td>
<td>209, 234</td>
</tr>
<tr>
<td>\AlPh@GetNumberOfSymbols</td>
<td>202, 394</td>
</tr>
<tr>
<td>\AlPh@ifDefinable</td>
<td>139, 151, 375, 417</td>
</tr>
<tr>
<td>\AlPh@ifOptArg</td>
<td>158, 162, 176, 179, 377</td>
</tr>
<tr>
<td>\AlPh@ifOptArgNext</td>
<td>165, 168</td>
</tr>
<tr>
<td>\AlPh@Method@alph</td>
<td>292</td>
</tr>
<tr>
<td>\AlPh@Method@mult</td>
<td>343</td>
</tr>
<tr>
<td>\AlPh@Method@wrap</td>
<td>320</td>
</tr>
<tr>
<td>\AlPh@NewAlphAlph</td>
<td>403, 408, 416</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>B</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>\begin</td>
<td>601, 629, 653, 689, 710, 725, 760</td>
</tr>
<tr>
<td>\body</td>
<td>453, 457</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>C</th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>\catch</td>
<td>727, 728, 732, 736, 737, 740, 742, 747, 748, 749, 750, 751, 752, 753, 754, 755, 756, 757</td>
</tr>
<tr>
<td>\endcode</td>
<td>2, 3, 5, 6, 7, 8, 9, 10, 11, 12, 13, 33, 34, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 69, 70, 72, 73, 74, 78, 79, 80, 81, 82, 83, 84, 87, 88, 90, 91, 92, 93, 97, 99, 127, 128, 130, 131, 433, 434, 435, 436, 471, 480, 488, 492, 509, 510, 511</td>
</tr>
</tbody>
</table>