The `mathcommand` package for LaTeX

[version v1.04–2021/06/07]

Thomas Colcombet

June 7, 2021

Abstract

The `mathcommand` package provides functionalities for defining macros
1. that have different behaviors depending on whether in math or text mode,
2. that absorb Primes, Indices and Exponents (PIE) as extra parameters usable in the code,
3. offers some iteration facilities for defining macros with similar code,
4. and deactivating macros.

The primary objective of this package is to be used together with the `knowledge` package for a proper handling of mathematical notations.

1 History of the package

2019-05-12 First version of the package. V1.01 on CTAN,

2019-05-14 New macros `\IfEmptyTF`, `\GetExponent`, and `\GetIndex`.

2019-07-03 Corrects bug of functions declaring PIE's issuing an error when already existing. Version 1.02 on CTAN.

2019-12-06 Added disabling commands, and package options. Version 1.03 on CTAN.

2021-06-07 Corrected code that was broken due to an update in `xparse`. Version 1.04 on CTAN.

2 Defining text and math commands

The principle is that the package will maintain, for a macro `\macro`, two concurrent version of the code: a *math variant* (technically it is stored in a macro `\Math macro`) and a *text variant* (technically stored in a macro `\Text macro`[1]). The macro `\macro` itself will execute one or the other depending on whether it is executed in math or text mode. Note that all the macros are non-expandable for avoiding problems with mathematics that would be sent, for instance, to the table of contents. The list of commands is described at the end of the section.

---

[1] These cannot be used accidentally by the user since these control sequences contain a space.

For instance after executing:

```
\newmathcommand\macro[1]{\mathit{math}^\mathrm{code}_{(#1)}}
\newtextcommand\macro[1]{text code (#1)}
```

when executing `\macro` in math mode, the math code will be executed, and in text mode similarly:

`\macro{a}` yields 'text code (a)'   while `$\macro{a}$` yields '$math^{\mathrm{code}}_{(a)}$'.

If the macro `\macro` already exists, it is stored under the name `\LaTeXmacro`, and then everything happens as if it had already been defined both in math and text mode. This is useful for redefining known macros. For instance `\c` is a convenient way to producing cedillas in LaTeX, as in `\c a` which yields 'ą'. However, one may want `\c` to reprensent a variable c in math mode. This is done using, e.g.:

```
\renewmathcommand\c{c}
```

Then, the macro `\c` still works in text mode, and using `\c` in math mode does display simply 'c'.

The name of the macros offered by the `mathcommand` package are mere adaptations of the standard macros of LaTeX and of the package `xparse`[2]. Their syntax is the same (in particular in terms of parameter definitions):

`\newmathcommand` is similar to `\newcommand` and creates a math variant,

`\newtextcommand` is similar to `\newcommand` and creates a text variant,

`\renewmathcommand` is similar to `\renewcommand` and creates a math variant,

`\renewtextcommand` is similar to `\renewcommand` and creates a text variant,

`\declaremathcommand` is similar to `\newcommand` but defines the macro even if it exists before; it creates a math variant,

`\declaretextcommand` is similar to `\newcommand` but defines the macro even if it exists before; it creates a text variant,

`\NewDocumentMathCommand` is like `\NewDocumentCommand` of the `xparse` package, but creates a math variant,

`\NewDocumentTextCommand` is like `\NewDocumentCommand` of the `xparse` package, but creates a text variant,

`\RenewDocumentMathCommand` is like `\RenewDocumentCommand` of the `xparse` package, but creates a math variant,

`\RenewDocumentTextCommand` is like `\RenewDocumentCommand` of the `xparse` package, but creates a text variant,

`\DeclareDocumentMathCommand` is like `\DeclareDocumentCommand` of the `xparse` package, but creates a math variant,

`\DeclareDocumentTextCommand` is like `\DeclareDocumentCommand` of the `xparse` package, but creates a text variant,

`\ProvideDocumentMathCommand` is like `\ProvideDocumentCommand` of the `xparse` package, but creates a math variant,

---

[2]The package `xparse` offers a very convenient way to define macros with complicated parameter signatures.

`\ProvideDocumentTextCommand` is like `\ProvideDocumentCommand` of the xparse package, but creates a text variant.

The package offers also the following commands:

`\declarecommand` which is similar to `\newcommand` but defines the macro even if it exists before,
`\storecommand[optional-prefix]\macro` which copies the content of the macro `\macro` to `\optional-prefixmacro`. By default, the optional prefix is LaTeX. (Hence, it does what is automatically made by commands such as `\declarecommand`).

3 Defining Prime/Indices/Exponents absorbing commands (PIE commands)

Another feature offered by the mathcommand package is to permit the definitions of macros that would absorb the primes, subscript and superscript that follow them. The three pieces of information are abbreviated as *PIE* (for "Primes-Indices-Exponents"). This terminology serves as a help for remembering the order prime-index-exponent. A *PIE command* is similar to a normal macro/command, but for the fact that the PIEs that follow are absorbed and can be used in the macro as three extra parameters. The list of macros usable for for defining PIE commands can be found at the end of this section.

This is best explained through an example. After writing:

`\newcommandPIE\macro[1]{([#1]#3)#2#4}`

one obtains that

$$\texttt{\$\textbackslash macro\{A\}\_2'\$} \text{ yields } ([A]_2)' \ .$$

Indeed, in the body of the definition of macro, `#1` represents the normal parameter of the command, while the three following parameters (`#2`,`#3`,`#4` in this case) contain respectively the primes (either empty or a sequence of $'$ symbols), the index (either empty if there is no subscript or of the form `_{index}` if there is an index), and the exponent (either empty if there is no superscript or of the form `^{exponent}` if there is one). In the case of the above definition, the index (parameter `#3`) is written inside the parenthesis, while primes and exponents are put outside.

The are furthermore some helper functions:

`\IfEmptyTF` takes a string and two codes, and expands to the first one if the string is empty, and the second otherwise,
`\GetIndex` takes a string that is an index as in PIE commands, and expands to its content: it maps the empty string to the empty string, and strings of the form `_{sthg}` to sthg,
`\GetExponent` takes a string that is an exponent as in PIE commands, and expands to its content: it maps the empty string to the empty string, and strings of the form `^{sthg}` to sthg.

For instance:

```
\newmathcommandPIE\F{#2F#1\IfEmptyTF{#3}{}{^{(\GetExponent{#3})}}}
```

displays `$\F_2^3$` as '$_2F^{(3)}$': the index is placed before, and the exponent is surrounded by parentheses.

List of macros of defining PIE commands. Once more, appart from the specificity of PIE commands, the syntax is as the original corresponding commands these are based on.

`\newcommandPIE` is similar to `\newcommand` (but defines a non-expandable macro)
`\renewcommandPIE` is similar to `\renewcommand` (but defines a non-expandable macro)
`\declarecommandPIE` is similar to `\newcommand` and works even if the macro already exists (and defines a non-expandable macro)
`\NewDocumentCommandPIE` is similar to `\NewDocumentCommand` of the xparse package,
`\RenewDocumentCommandPIE` is similar to `\RenewDocumentCommand` of the xparse package,
`\DeclareDocumentCommandPIE` is similar to `\DeclareDocumentCommand` of the xparse package,
`\ProvideDocumentCommandPIE` is similar to `\ProvideDocumentCommand` of the xparse package.

Finally, a bunch of macros are used to define math variants that are PIE commands:

`\newmathcommandPIE` is like `\newcommandPIE` and creates a math variant,
`\renewmathcommandPIE` is like `\renewcommandPIE` and creates a math variant,
`\declaremathcommandPIE` is like `\declarecommandPIE` and creates a math variant,
`\NewDocumentMathCommandPIE` is like `\NewDocumentCommandPIE`, but creates a math variant,
`\RenewDocumentMathCommandPIE` is like `\RenewDocumentCommandPIE`, but creates a math variant,
`\DeclareDocumentMathCommandPIE` is like `\DeclareDocumentCommandPIE`, but creates a math variant,
`\ProvideDocumentMathCommandPIE` is like `\ProvideDocumentCommandPIE`, but creates a math variant,

4 Looping for defining commands

The `mathcommand` package offer also some capabilities for automatically defining multiple similar macros. This is done using only one command:

`\LoopCommands`{list on which to iterate}[name 1][name 2]...[name 7]{code}

The *list on which to iterate* is a list of letters or braced sequences of letters. the *name 1*, *name 2* up to *name 7* optional parameters are expandable pieces of code that are to be evaluated and then converted into control sequences; they may use the extra parameter `#1`. Finally, *code* is the code to be executed that can use the parameters `#1`, `#2`, up to `#8`.

The result of executing this macro is that each of the letters or sequences of letters in the list on which to iterate will be taken one after the other. For each of

them, the code is executed, taking as value of the parameter #1 the element in the list, and as parameters #2 to #8 control sequences constructed from the evaluation of name 1 up to name 7 (using as parameters #1 the element of the sequence).

For instance, imagine one easily wants to denote vectors simply as '\vx' instead of '\vec x' or '\vec{x}', it is sufficient to write:

```
\LoopCommands{abcdefghijklmnopqrstuvwxyz}[v#1]
    {\newcommand#2{\vec #1}}
```

It will result in the successive execution of `\newcommand\va{\vec a}` and so on up to `\newcommand\vz{\vec z}`.

Note also that the list on which to iterate is automatically expanded, and if a non-expandable control sequence is met, then it is replaced by its the text defining the control sequence. Hence using `{\alpha\beta}` is equivalent to `{{alpha}{beta}}`.

Some extra remarks may be helpful:

- As usual in TeX/LaTeX, the code may have to use its own internal parameters, for instance for defining macros: such parameters should use double #'s, i.e., `##1`, `##2` up to `##9`.

  For instance:

  ```
  \LoopCommands{abcdefghijklmnopqrstuvwxyz}[o#1]
      {\declarecommandPIE#2{\overline{#1##2}##1##3}
  ```

  will result in \ou to be declared as the PIE command defined with as main body `\overline{u#2}#1#3` (note the translation of parameters, which is the standard way to proceed for TeX). In our case `\ou_1^2` yields '$\overline{u_1}^2$' (the subscript gets to be inside the bar, and the superscript and primes outside), and so on...

- When defining multiple commands, some may already exist. To avoid conflicts, one should use the 'declare' version of the defining commands. These will work independently of the context. Is it also good to define only the math variants using the appropriate commands of the package.

- The following strings are predefined for the user to loop on:

  `\lettersUppercase` stands for ABCDEFGHIJKLMNOPQRSTUVWXYZ

  `\lettersLowercase` stands for abcdefghijklmnopqrstuvwxyz

  `\lettersAll` stands for abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ.

  `\lettersGreekLowercase` stands for $\alpha\beta\gamma\delta\epsilon\varepsilon\zeta\eta\theta\vartheta\iota\kappa\lambda\mu\nu\xi\pi\varpi\rho\varrho\sigma\varsigma\tau\upsilon\phi\varphi\chi\psi\omega$.

  `\lettersGreekUppercase` stands for $\Gamma\Delta\Theta\Lambda\Xi\Pi\Sigma\Upsilon\Phi\Psi\Omega$.

  `\lettersGreekAll` stands for $\alpha\beta\gamma\delta\epsilon\varepsilon\zeta\eta\theta\vartheta\iota\kappa\lambda\mu\nu\xi\pi\varpi\rho\varrho\sigma\varsigma\tau\upsilon\phi\varphi\chi\psi\omega\Gamma\Delta\Theta\Lambda\Xi\Pi\Sigma\Upsilon\Phi\Psi\Omega$.

Hence, for instance:

```
\LoopCommands\lettersUppercase[bb#1]
      {\newmathcommand#2{\mathbb#1}}
\LoopCommands\lettersGreekLowercase[#1][LaTeX#1]
      {\renewmathcommand#2{\textcolor{blue}{#3}}}
```

configures the macros `\bbA`, ..., `\bbZ` to display the letters in blackboard bold alphabet (e.g. with the `amsfonts` package), and the lowercase greek letters `\alpha`,... to be displayed in blue (with the `xcolor` package loaded). Note in the last case the use of the third parameter for accessing the macros `\LaTeXalpha`,... that are automatically generated by the `\renewmathcommand` macro.

5 Disabling command

The package `mathcommand` has some facilities for deactivating commands, and suggesting replacements. This can be useful when working with coauthors and help them using the same macros. Disabling a command is achieved using

$$\texttt{\textbackslash disablecommand}\{\text{sequence of control sequences}\}$$

The result of this command is that the control sequences appearing in the sequence are deactivated. *Deactivating* a macro `\macro` means that:

- The original macro is stored as `\LaTeXmacro`, and thus can still be used.

- Using `\macro` now displays an error message explaining that it has been disabled, that `\LaTeXmacro` can be used instead, and also provides a list suggestions of replacement that are defined by the command `\suggestcommand`.

However, be careful: disabling a command which is used in the system or in the the style may cause problems. Hence, it is possible to disable all math symbols but macros like `\dagger` would yield problem if used in footnotes or for the affiliation of the authors. The syntax of `\suggestcommand` is as follows:

$$\texttt{\textbackslash suggestcommand}\texttt{\textbackslash macro}\{\text{suggestion to be displayed when } \texttt{\textbackslash macro} \text{ is used}\}$$

For instance:
```
\disablecommand{\leq\geq}
\suggestcommand{\leq}{Use the better looking \leqslant.}
\suggestcommand{\geq}{Use the better looking \geqslant.}
```

Now, executing `\leq` yields:

```
./filename.tex:linenumber: Package mathcommand Error:
(mathcommand)                The command \geq is disabled. Instead:
(mathcommand)                Use \LaTeXgeq for the original macro.
(mathcommand)                Use the better looking \geqslant .
```

6   Options

Options can be triggered either when the package is loaded, or using:

$$\texttt{\textbackslash mathcommandconfigure}\{\text{option list}\}$$

The available options are:

- *disabled=error* makes disabled commands produce errors,

- *disabled=warning* makes disabled commands produce warnings,

- *disabled=silent* makes disabled commands work as before being disabled (useful when including code that cannot be modified).

7 Implementation

7.1 README.md

<*readme>

```
 1 This directory contains the package
 2
 3   name: mathcommand
 4   license: LaTeX Project Public License version 1.3 or above
 5   version: v1.04
 6   date: 2021/06/07
 7   author: Thomas Colcombet
 8   mail: thomas.colcombet@irif.fr
 9   web: https://www.ctan.org/pkg/mathcommand
10
11 Purpose:
12 The mathcommand package provides functionalities for defining macros:
13   - that have different behaviors depending on whether in math or text mode,
14   - absorb Primes, Indices, Exponents (PIE) following LaTeX notations and
15     have them as extra parameters usable in the code.
16 The primary objective of this code is to be used together with the knowledge
17 package for a proper handling of mathematical notations.
18
19 Install:
20 It is sufficient to have the file mathcommand.sty accessible by LaTeX.
21 It can be produced by 'make mathcommand.sty' if necessary.
22 The documentation is in the file mathcommand.pdf.
23
24 Content of the file mathcommand-ctan.zip:
25 - README.md: this file generated while compiling mathcommand.ins,
26 - mathcommand.sty: the package file (generated using mathcommand.ins)
27 - mathcommand.pdf: the user documentation (generated by compiling
28   mathcommand.dtx)
29 - makefile: the makefile. Use 'make all' to generate mathcommand.sty
30   and mathcommand.pdf. It can also: clean the directory, make zip
31   version of the sources, or ready for CTAN.
32 - mathcommand.ins: is the file generating mathcommand.sty and
33   README.md from mathcommand.dtx (using docstrip).
34 - mathcommand.dtx: code and documentation.
35
```

</readme>

7.2 Code preparation

<*package>

```
36 \NeedsTeXFormat{LaTeX2e}[1994/06/01]
37 \RequirePackage{expl3,l3keys2e}
38 \RequirePackage{etoolbox}
39 \RequirePackage{xparse}
```

```
40
41 \ExplSyntaxOn
42 \bool_if_exist:NTF\mathcommand_package_loaded_bool
43  \endinput
44 {\bool_new:N\mathcommand_package_loaded_bool
45  \bool_set_true:N\mathcommand_package_loaded_bool}
```

7.3 Absorbing primes, indices and exponents (PIE)

7.3.1 Parsing pies

We start by defining the code used to absorb PIEs from the input stream. The main function defined in this context is

> __mathcommand_absorb_PIE:nw

which takes some code as first parameter, then absorbs primes, indices and exponents, and finally reinserts the code in the input stream, followed with three braces containing respectively the primes, the index, and the exponent.

It works by storing the code to be executed in __mathcommand_absorb_finished_tl, preparing __mathcommand_primes_tl, __mathcommand_index_t, and __mathcommand_exponent_tl to contain the PIEs. Then the core of the parsing is performed by __mathcommand_absorb:w.

```
46 \cs_new:Npn\__mathcommand_absorb_PIE:nw#1{
47  \tl_set:Nn\__mathcommand_absorb_finished_tl{#1}
48  \tl_set:Nn\__mathcommand_primes_tl{}
49  \tl_set:Nn\__mathcommand_index_tl{}
50  \tl_set:Nn\__mathcommand_exponent_tl{}
51  \__mathcommand_absorb:w
52 }
```

When the parsing is finished, __mathcommand_absorb_finished: is executed, which inserts the original code stored in __mathcommand_absorb_finished_tl followed by the PIEs in the input stream.

```
53 \cs_new:Nn\__mathcommand_absorb_finished:{
54  \exp_args:NV\__mathcommand_absorb_finished_:\__mathcommand_exponent_tl
55 }
56 \cs_new:Nn\__mathcommand_absorb_finished_:{
57  \exp_args:NV\__mathcommand_absorb_finished__:\__mathcommand_index_tl
58 }
59 \cs_new:Nn\__mathcommand_absorb_finished__:{
60  \exp_args:NV\__mathcommand_absorb_finished_tl\__mathcommand_primes_tl
61 }

62 \cs_new:Npn\peek_subscript_remove:TFw
63     {\peek_charcode_remove:NTF _}
64 \cs_new:Npn\peek_superscript_remove:TFw
65     {\peek_charcode_remove:NTF ^}
66 \cs_new:Npn\peek_prime_remove:TFw
67     {\peek_charcode_remove:NTF '}
```

```
68 \cs_new:Nn\__mathcommand_absorb_add_prime:{
69   \tl_put_right:Nn\__mathcommand_primes_tl{'}
70 }

71 \ExplSyntaxOff
72 \expandafter\def\csname g_tmpa_tl\endcsname{_}
73 \ExplSyntaxOn
74 \cs_new:Nx\__mathcommand_absorb_add_index_after:Nn{
75   \exp_not:N\tl_set:Nn\exp_not:N\__mathcommand_index_tl
76     {\g_tmpa_tl{#2}}
77   #1
78 }

79 \cs_new:Nn\__mathcommand_absorb_add_exponent_after:Nn{
80   \tl_set:Nn\__mathcommand_exponent_tl{^{#2}}
81   #1
82 }

83 \cs_new:Npn\__mathcommand_absorb:w{
84     \peek_prime_remove:TFw
85       {\__mathcommand_absorb_add_prime:
86        \__mathcommand_absorb_p:w}
87       \__mathcommand_absorb_:w}
88 \cs_new:Npn\__mathcommand_absorb_:w{
89     \peek_subscript_remove:TFw
90       {\__mathcommand_absorb_add_index_after:Nn
91        \__mathcommand_absorb_i:w}
92       \__mathcommand_absorb__:w}
93 \cs_new:Npn\__mathcommand_absorb__:w{
94     \peek_superscript_remove:TFw
95       {\__mathcommand_absorb_add_exponent_after:Nn
96        \__mathcommand_absorb_e:w}
97       \__mathcommand_absorb_finished:}

98 \cs_new:Npn\__mathcommand_absorb_p:w{
99     \peek_prime_remove:TFw
100       {\__mathcommand_absorb_add_prime:
101        \__mathcommand_absorb_p:w}
102       \__mathcommand_absorb_p_:w}
103 \cs_new:Npn\__mathcommand_absorb_p_:w{
104     \peek_subscript_remove:TFw
105       {\__mathcommand_absorb_add_index_after:Nn
106        \__mathcommand_absorb_pi:w}
107       \__mathcommand_absorb_finished:}

108 \cs_new:Npn\__mathcommand_absorb_pi:w{
109     \peek_prime_remove:TFw
110       {\__mathcommand_absorb_add_prime:
111        \__mathcommand_absorb_pi:w}
112       \__mathcommand_absorb_finished:}

113 \cs_new:Npn\__mathcommand_absorb_e:w{
114     \peek_subscript_remove:TFw
```

```
115        {\__mathcommand_absorb_add_index_after:Nn
116          \__mathcommand_absorb_finished:}
117          \__mathcommand_absorb_finished:}
118 \cs_new:Npn\__mathcommand_absorb_i:w{
119    \peek_prime_remove:TFw
120        {\__mathcommand_absorb_add_prime:
121          \__mathcommand_absorb_pi:w}
122          \__mathcommand_absorb_i_:w}
123 \cs_new:Npn\__mathcommand_absorb_i_:w{
124    \peek_superscript_remove:TFw
125        {\__mathcommand_absorb_add_exponent_after:Nn
126          \__mathcommand_absorb_finished:}
127          \__mathcommand_absorb_finished:}
```

7.3.2   Definition of high level commands

```
128 \NewDocumentCommand\newcommandPIE{ m o  o m }{
129    \__mathcommand_check_definable:nNT {#1} \newcommandPIE
130        {
131          \cs_if_exist:NTF #1
132            {
133              \__kernel_msg_error:nnxx { mathcommand } { command-already-defined }
134                { \use:nnn \token_to_str:N #1 { } }
135                { \token_to_str:N \newcommandPIE }
136            }
137            { \__mathcommand_declarecommandPIE:Nnnn #1{#2}{#3}{#4} }
138        }
139 }
140 \NewDocumentCommand\renewcommandPIE{ m o  o m }{
141    \__mathcommand_check_definable:nNT {#1} \renewcommandPIE
142        {
143          \cs_if_exist:NTF #1
144            { \__mathcommand_declarecommandPIE:Nnnn #1{#2}{#3}{#4} }
145            {
146              \__kernel_msg_error:nnxx { mathcommand } { command-not-yet-defined }
147                { \use:nnn \token_to_str:N #1 { } }
148                { \token_to_str:N \renewcommandPIE }
149            }
150        }
151 }
152 \NewDocumentCommand\declarecommandPIE{ m o  o m }{
153    \__mathcommand_check_definable:nNT {#1} \declarecommandPIE
154        { \__mathcommand_declarecommandPIE:Nnnn #1{#2}{#3}{#4} }
155 }
156 \cs_new:Nn\__mathcommand_declarecommandPIE:Nnnn{
157    \use:x{
158    \exp_not:N\__mathcommand_declarePIE_generic:Nnnn
159        \exp_not:N#1
160        {\IfNoValueTF{#2}{0}{#2}}
```

```
161      {\cs_if_exist:NTF#1
162           {\exp_not:N\renewrobustcmd}
163           {\exp_not:N\newrobustcmd}
164         \exp_not:N#1
165         \IfNoValueTF{#2}{}{[#2]}
166         \IfNoValueTF{#3}{}{[{\exp_not:n{#3}}]}}}
167      {\exp_not:n{#4}}
168 }}
169 \cs_new_protected:Npn\NewDocumentCommandPIE#1#2#3{
170    \__mathcommand_check_definable:nNT {#1} \NewDocumentCommandPIE
171      {
172        \cs_if_exist:NTF #1
173          {
174            \__kernel_msg_error:nnxx { mathcommand } { command-already-defined }
175              { \use:nnn \token_to_str:N #1 { } }
176              { \token_to_str:N \NewDocumentCommandPIE }
177          }
178          { \__mathcommand_DeclareDocumentCommandPIE:Nnn #1 {#2} {#3} }
179      }
180 }
181 \cs_new_protected:Npn\RenewDocumentCommandPIE#1#2#3{
182    \__mathcommand_check_definable:nNT {#1} \RenewDocumentCommandPIE
183      {
184        \cs_if_exist:NTF #1
185          { \__mathcommand_DeclareDocumentCommandPIE:Nnn #1 {#2} {#3} }
186          {
187            \__kernel_msg_error:nnxx { mathcommand } { command-not-yet-defined }
188              { \use:nnn \token_to_str:N #1 { } }
189              { \token_to_str:N \RenewDocumentCommandPIE }
190          }
191      }
192 }
193 \cs_new_protected:Npn\DeclareDocumentCommandPIE#1#2#3{
194    \__mathcommand_check_definable:nNT {#1} \DeclareDocumentCommandPIE
195      {
196            \__mathcommand_DeclareDocumentCommandPIE:Nnn #1 {#2} {#3}
197      }
198 }
199 \cs_new_protected:Npn\ProvideDocumentCommandPIE#1#2#3{
200    \__mathcommand_check_definable:nNT {#1} \ProvideDocumentCommandPIE
201      {
202        \cs_if_exist:NTF #1{}
203          {
204            \__mathcommand_DeclareDocumentCommandPIE:Nnn #1 {#2} {#3}
205          }
206      }
207 }
208 \cs_set:Nn\__mathcommand_DeclareDocumentCommandPIE:Nnn{
```

```
209    \group_begin:
210        \DeclareDocumentCommand#1{#2}{}
211        \int_gset_eq:NN\g_tmpa_int\l__xparse_current_arg_int
212    \group_end:
213    \__mathcommand_declarePIE_generic:Nnnn
214        #1
215        {\g_tmpa_int}
216        {\DeclareDocumentCommand#1{#2}}
217        {#3}
218 }
```

Control token, number parameters, defining command, code

```
219 \cs_new:Nn\__mathcommand_declarePIE_generic:Nnnn{
220    \int_compare:nNnTF{#2}>{6}
221      {\PackageError{mathcommand}
222            {At~most~6~parameters~in~PIE~commands~when~defining~'\token_to_str:N#1'}
223            {PIE~commands~(mathcommand~package)~do~not~accept~more~than~six~parameters.}}
224      {\int_case:nn{#2}
225       {{0} {\cs_set:cpn{\cs_to_str:N#1~PIE~code}##1##2##3}
226        {1} {\cs_set:cpn{\cs_to_str:N#1~PIE~code}##1##2##3##4}
227        {2} {\cs_set:cpn{\cs_to_str:N#1~PIE~code}##1##2##3##4##5}
228        {3} {\cs_set:cpn{\cs_to_str:N#1~PIE~code}##1##2##3##4##5##6}
229        {4} {\cs_set:cpn{\cs_to_str:N#1~PIE~code}##1##2##3##4##5##6##7}
230        {5} {\cs_set:cpn{\cs_to_str:N#1~PIE~code}##1##2##3##4##5##6##7##8}
231        {6} {\cs_set:cpn{\cs_to_str:N#1~PIE~code}##1##2##3##4##5##6##7##8##9}}
232       {#4}
233    \use:x{
234        \exp_not:n{#3}
235        {\exp_not:N\__mathcommand_absorb_PIE:nw
236            {\exp_not:c{\cs_to_str:N#1~PIE~code}
237            \int_case:nn{#2}
238            {{0}{}
239             {1}{\exp_not:n{{##1}}}
240             {2}{\exp_not:n{{##1}{##2}}}
241             {3}{\exp_not:n{{##1}{##2}{##3}}}
242             {4}{\exp_not:n{{##1}{##2}{##3}{##4}}}
243             {5}{\exp_not:n{{##1}{##2}{##3}{##4}{##5}}}
244             {6}{\exp_not:n{{##1}{##2}{##3}{##4}{##5}{##6}}}}}}}}
245    }
246 }
```

7.3.3   Auxiliary functions

```
247 \def\lettersUppercase{ABCDEFGHIJKLMNOPQRSTUVWXYZ}
248 \def\lettersLowercase{abcdefghijklmnopqrstuvwxyz}
249 \xdef\lettersAll{\lettersLowercase\lettersUppercase}
250 \def\lettersGreekLowercase{\alpha\beta\gamma\delta\epsilon\varepsilon\zeta\eta\theta\vartheta\i
251 \def\lettersGreekUppercase{\Gamma\Delta\Theta\Lambda\Xi\Pi\Sigma\Upsilon\Phi\Psi\Omega}
252 \xdef\lettersGreekAll{\lettersGreekLowercase\lettersGreekUppercase}

253 \cs_set_eq:NN\IfEmptyTF\tl_if_empty:nTF
```

```
254 \cs_new:Npn\EmptyContent#1{
255     \tl_if_empty:nTF{#1}{{}}{
256         \__mathcommand_EmptyContent:w #1*\end_marker:
257     }
258 }
259 \cs_new:Npn\__mathcommand_EmptyContent:w #1#2\end_marker:{
260     #1*
261 }

262 \cs_new:Npn\GetExponent#1{
263     \tl_if_empty:nTF{#1}{}{
264         \__mathcommand_GetIndexOrExponent:w #1\__end_marker__
265     }
266 }
267 \cs_new:Npn\GetIndex#1{
268     \tl_if_empty:nTF{#1}{}{
269         \__mathcommand_GetIndexOrExponent:w #1\__end_marker__
270     }
271 }
272 \cs_new:Npn\__mathcommand_GetIndexOrExponent:w #1#2#3\__end_marker__{
273     #2
274 }
```

7.4   Separating math and text macros

```
275 \tl_const:Nn\__mathcommand_prefix_math_tl{Math~}
276 \tl_const:Nn\__mathcommand_prefix_text_tl{Text~}
277 \tl_const:Nn\__mathcommand_prefix_store_tl{LaTeX}

278 \cs_new:Nn\__mathcommand_to_mathtl:N{\__mathcommand_prefix_math_tl\cs_to_str:N#1}
279 \cs_new:Nn\__mathcommand_to_texttl:N{\__mathcommand_prefix_text_tl\cs_to_str:N#1}
280 \cs_new:Nn\__mathcommand_to_storetl:N{\__mathcommand_prefix_store_tl\cs_to_str:N#1}
281 \cs_new:Nn\__mathcommand_coretl:N
282     {\expandafter\__command_coretl:w\string#1\end_mark}
283 \cs_new:Npn\__command_coretl:w#1~#2\end_mark{#2}

284 \cs_new:Npn\__mathcommand_if_exist:NTF
285   {\cs_if_exist:NTF}
286 \cs_new:Npn\__mathcommand_if_exist_math:NTF#1
287   {\cs_if_exist:cTF{\__mathcommand_to_mathtl:N#1}}
288 \cs_new:Npn\__mathcommand_if_exist_text:NTF#1
289   {\cs_if_exist:cTF{\__mathcommand_to_texttl:N#1}}
290 \cs_new:Npn\__mathcommand_if_exist_text_or_math:NTF#1
291   {\__mathcommand_if_exist_math:NTF#1
292       \use_i:nn{\__mathcommand_if_exist_text:NTF#1}}

293 \cs_new:Nn\__mathcommand_error_if_exist_math:NF{
294     \cs_if_exist:cTF{\__mathcommand_to_mathtl:N#1}
295     {\exp_args:Nnx\PackageError{}
296         {Command~'\token_to_str:N#1'~already~exists~in~math~mode}
297         {}
298     }{#2}
299 }
```

14

```
300 \cs_new:Nn\__mathcommand_error_if_exist_text:NF{
301     \cs_if_exist:cTF{\__mathcommand_to_texttl:N#1}
302     {\exp_args:Nnx\PackageError{}
303         {Command~'\token_to_str:N#1'~already~exists~in~text~mode}
304         {}
305     }{#2}
306 }

307 \cs_new:Nn\__mathcommand_error_if_not_exist_math:NF{
308     \cs_if_exist:cTF{\__mathcommand_to_mathtl:N#1}
309        {#2}
310        {\exp_args:Nnx\PackageError{}
311         {Command~'\token_to_str:N#1'~does~not~exist~in~math~mode}
312         {}
313     }
314 }

315 \cs_new:Nn\__mathcommand_error_if_not_exist_text:NF{
316     \cs_if_exist:cTF{\__mathcommand_to_texttl:N#1}
317        {#2}
318        {\exp_args:Nnx\PackageError{}
319         {Command~'\token_to_str:N#1'~does~not~exist~in~text~mode}
320         {}
321     }
322 }

323 \cs_new:Nn\__mathcommand_error_unknownmath:N{
324     \exp_args:Nnx\PackageError{}
325         {Command~'\token_to_str:c{\__mathcommand_coretl:N #1}'~does~not~exist~in~math~mode}
326         {}
327 }

328 \cs_new:Nn\__mathcommand_error_unknowntext:N{
329     \exp_args:Nnx\PackageError{}
330         {Command~'\token_to_str:c{\__mathcommand_coretl:N #1}'~does~not~exist~in~text~mode}
331         {}
332 }

333 \cs_new:Nn\__mathcommand_try_math:N{
334     \cs_if_exist:NTF#1#1{\__mathcommand_error_unknownmath:N#1}
335 }
336 \cs_new:Nn\__mathcommand_try_text:N{
337     \cs_if_exist:NTF#1#1{\__mathcommand_error_unknowntext:N#1}
338 }
```

The macro `\__mathcommand_create_fork:N` takes a control sequence, and creates the forking code that executes either the math branch of the text branch. If this forking code is already present, the command does nothing. However, if some macro was already associated with this control sequence, then it is copied to the math variant, the text variant as well as the stored.

```
339 \cs_set:Nn\__mathcommand_create_fork:N{
340     \let\__mathcommand_tmp_cs\undefined
341     \exp_args:NNx
```

```
342    \cs_new_protected:Npn\__mathcommand_tmp_cs{
343        \exp_not:N\mode_if_math:TF
344          {\exp_not:N\__mathcommand_try_math:N\exp_not:c{\__mathcommand_to_mathtl:N#1}}
345          {\exp_not:N\__mathcommand_try_text:N\exp_not:c{\__mathcommand_to_texttl:N#1}}
346      }
347    \cs_if_exist:NTF#1{
348      \cs_if_eq:NNTF#1\__mathcommand_tmp_cs
349        { }
350        { \cs_set_eq:cN{\__mathcommand_to_storetl:N#1}#1
351          \cs_set_eq:cN{\__mathcommand_to_mathtl:N#1}#1
352          \cs_set_eq:cN{\__mathcommand_to_texttl:N#1}#1
353          \cs_set_eq:NN#1\__mathcommand_tmp_cs
354        }
355    }{ \cs_set_eq:NN#1\__mathcommand_tmp_cs }
356 }
```

7.5   Definition of the high level commands

```
357 \NewDocumentCommand\declarecommand{m}{
358    \__mathcommand_check_definable:nNT {#1} \declarecommand
359    {
360      \cs_if_exist:NTF#1
361        {\renewcommand#1}
362        {\newcommand#1}
363    }
364 }
365 \newrobustcmd\storecommand[2][\__mathcommand_prefix_store_tl]{
366    \__mathcommand_check_definable:nNT {#2} \storecommand
367    {
368      \cs_if_exist:NTF#2{
369          \cs_set_eq:cN{#1\cs_to_str:N#2}#2
370      }
371      {
372          \PackageError{mathcommand}
373        {The~command~'\token_to_str:N#2'~does~not~exist~(in~\token_to_str:N\storecommand)}
374        {}
375      }
376    }
377 }
378
379 \NewDocumentCommand\NewDocumentMathCommand{m}{
380    \__mathcommand_check_definable:nNT {#1} \NewDocumentMathCommand
381    {
382      \__mathcommand_create_fork:N#1
383      \__mathcommand_error_if_exist_math:NF#1
384      {\exp_args:Nc\DeclareDocumentCommand{\__mathcommand_to_mathtl:N#1}}
385    }
386 }
387 \NewDocumentCommand\NewDocumentTextCommand{m}{
388    \__mathcommand_check_definable:nNT {#1} \NewDocumentTextCommand
389    {
```

```
390      \__mathcommand_create_fork:N#1
391      \__mathcommand_error_if_exist_text:NF#1
392      {\exp_args:Nc\DeclareDocumentCommand{\__mathcommand_to_textttl:N#1}}
393    }
394 }
395 \NewDocumentCommand\newmathcommand{m}{
396    \__mathcommand_check_definable:nNT {#1} \newmathcommand
397    {
398      \__mathcommand_create_fork:N#1
399      \__mathcommand_error_if_exist_math:NF#1
400      {\exp_args:Nc\newcommand{\__mathcommand_to_mathtl:N#1}}
401    }
402 }
403 \NewDocumentCommand\newtextcommand{m}{
404    \__mathcommand_check_definable:nNT {#1} \newtextcommand
405    {
406      \__mathcommand_create_fork:N#1
407      \__mathcommand_error_if_exist_text:NF#1
408      {\exp_args:Nc\newcommand{\__mathcommand_to_textttl:N#1}}
409    }
410 }
411 \NewDocumentCommand\RenewDocumentMathCommand{m}{
412    \__mathcommand_check_definable:nNT {#1} \RenewDocumentMathCommand
413    {
414      \__mathcommand_create_fork:N#1
415      \__mathcommand_error_if_not_exist_math:NF#1
416      {\exp_args:Nc\DeclareDocumentCommand{\__mathcommand_to_mathtl:N#1}}
417    }
418 }
419 \NewDocumentCommand\RenewDocumentTextCommand{m}{
420    \__mathcommand_check_definable:nNT {#1} \RenewDocumentMathCommand
421    {
422      \__mathcommand_create_fork:N#1
423      \__mathcommand_error_if_not_exist_text:NF#1
424      {\exp_args:Nc\DeclareDocumentCommand{\__mathcommand_to_textttl:N#1}}
425    }
426 }
427 \NewDocumentCommand\renewmathcommand{m}{
428    \__mathcommand_check_definable:nNT {#1} \renewmathcommand
429    {
430      \__mathcommand_create_fork:N#1
431      \__mathcommand_error_if_not_exist_math:NF#1
432      {\exp_args:Nc\renewcommand{\__mathcommand_to_mathtl:N#1}}
433    }
434 }
435 \NewDocumentCommand\renewtextcommand{m}{
436    \__mathcommand_check_definable:nNT {#1} \renewtextcommand
437    {
438      \__mathcommand_create_fork:N#1
439      \__mathcommand_error_if_not_exist_text:NF#1
```

```
440        {\exp_args:Nc\renewcommand{\__mathcommand_to_textttl:N#1}}
441    }
442 }
443 \NewDocumentCommand\declaremathcommand{m}{
444    \__mathcommand_check_definable:nNT {#1} \renewmathcommand
445    {
446        \__mathcommand_create_fork:N#1
447        \exp_args:Nc\declarecommand{\__mathcommand_to_mathtl:N#1}
448    }
449 }
450 \NewDocumentCommand\declaretextcommand{m}{
451    \__mathcommand_check_definable:nNT {#1} \renewtextcommand
452    {
453        \__mathcommand_create_fork:N#1
454        \exp_args:Nc\declarecommand{\__mathcommand_to_textttl:N#1}
455    }
456 }
457
458 \NewDocumentCommand\DeclareDocumentMathCommand{m}{
459    \__mathcommand_check_definable:nNT {#1} \DeclareDocumentMathCommand
460    {
461        \__mathcommand_create_fork:N#1
462        \exp_args:Nc\DeclareDocumentCommand{\__mathcommand_to_mathtl:N#1}
463    }
464 }
465 \NewDocumentCommand\DeclareDocumentTextCommand{m}{
466    \__mathcommand_check_definable:nNT {#1} \DeclareDocumentTextCommand
467    {
468        \__mathcommand_create_fork:N#1
469        \exp_args:Nc\DeclareDocumentCommand{\__mathcommand_to_textttl:N#1}
470    }
471 }
472 \NewDocumentCommand\ProvideDocumentMathCommand{mmm}{
473    \__mathcommand_check_definable:nNT {#1} \ProvideDocumentMathCommand
474    {
475        \__mathcommand_create_fork:N#1
476        \exp_args:Nc\ProvideDocumentCommand{\__mathcommand_to_mathtl:N#1}{#2}{#3}
477    }
478 }
479 \NewDocumentCommand\ProvideDocumentTextCommand{m}{
480    \__mathcommand_check_definable:nNT {#1} \ProvideDocumentTextCommand
481    {
482        \__mathcommand_create_fork:N#1
483        \exp_args:Nc\ProvideDocumentCommand{\__mathcommand_to_textttl:N#1}
484    }
485 }
```

7.6   Definition of the high level combined commands

```
486 \NewDocumentCommand\NewDocumentMathCommandPIE{m}{
```

```
487     \__mathcommand_check_definable:nNT {#1} \NewDocumentMathCommandPIE
488     {
489         \__mathcommand_create_fork:N#1
490         \__mathcommand_error_if_exist_math:NF#1
491         {\exp_args:Nc\DeclareDocumentCommandPIE{\__mathcommand_to_mathtl:N#1}}
492     }
493 }
494 \NewDocumentCommand\newmathcommandPIE{m}{
495     \__mathcommand_check_definable:nNT {#1} \newmathcommandPiE
496     {
497         \__mathcommand_create_fork:N#1
498         \__mathcommand_error_if_exist_math:NF#1
499         {\exp_args:Nc\newcommandPIE{\__mathcommand_to_mathtl:N#1}}
500     }
501 }
502 \NewDocumentCommand\RenewDocumentMathCommandPIE{m}{
503     \__mathcommand_check_definable:nNT {#1} \RenewDocumentMathCommandPIE
504     {
505         \__mathcommand_create_fork:N#1
506         \__mathcommand_error_if_not_exist_math:NF#1
507         {\exp_args:Nc\DeclareDocumentCommandPIE{\__mathcommand_to_mathtl:N#1}}
508     }
509 }
510 \NewDocumentCommand\renewmathcommandPIE{m}{
511     \__mathcommand_check_definable:nNT {#1} \renewmathcommandPIE
512     {
513         \__mathcommand_create_fork:N#1
514         \__mathcommand_error_if_not_exist_math:NF#1
515         {\exp_args:Nc\renewcommandPIE{\__mathcommand_to_mathtl:N#1}}
516     }
517 }
518 \NewDocumentCommand\DeclareDocumentMathCommandPIE{m}{
519     \__mathcommand_check_definable:nNT {#1} \DeclareDocumentMathCommand
520     {
521         \__mathcommand_create_fork:N#1
522         \exp_args:Nc\DeclareDocumentCommand{\__mathcommand_to_mathtl:N#1}
523     }
524 }
525 \NewDocumentCommand\declaremathcommandPIE{m}{
526     \__mathcommand_check_definable:nNT {#1} \declaremathcommandPIE
527     {
528         \__mathcommand_create_fork:N#1
529         \exp_args:Nc\declarecommandPIE{\__mathcommand_to_mathtl:N#1}
530     }
531 }
532
533 \NewDocumentCommand\ProvideDocumentMathCommandPIE{mmm}{
534     \__mathcommand_check_definable:nNT {#1} \ProvideDocumentMathCommandPIE
535     {
536         \__mathcommand_create_fork:N#1
```

```
537        \exp_args:Nc\ProvideDocumentCommandPIE{\__mathcommand_to_mathtl:N#1}{#2}{#3}
538    }
539 }
```

7.7   Looping for command definitions

```
540 \NewDocumentCommand\LoopCommands{ m ooooooo m }{
541    \IfNoValueTF{#2}
542      {\cs_set:Nn\__tmp_two:n{\exp_not:c{##1}}}
543      {\cs_set:Nn\__tmp_two:n{\exp_not:c{#2}}}
544    \IfNoValueTF{#3}
545      {\cs_set:Nn\__tmp_three:n{\exp_not:c{##1}}}
546      {\cs_set:Nn\__tmp_three:n{\exp_not:c{#3}}}
547    \IfNoValueTF{#4}
548      {\cs_set:Nn\__tmp_four:n{\exp_not:c{##1}}}
549      {\cs_set:Nn\__tmp_four:n{\exp_not:c{#4}}}
550    \IfNoValueTF{#5}
551      {\cs_set:Nn\__tmp_five:n{\exp_not:c{##1}}}
552      {\cs_set:Nn\__tmp_five:n{\exp_not:c{#5}}}
553    \IfNoValueTF{#6}
554      {\cs_set:Nn\__tmp_six:n{\exp_not:c{##1}}}
555      {\cs_set:Nn\__tmp_six:n{\exp_not:c{#6}}}
556    \IfNoValueTF{#7}
557      {\cs_set:Nn\__tmp_seven:n{\exp_not:c{##1}}}
558      {\cs_set:Nn\__tmp_seven:n{\exp_not:c{#7}}}
559    \IfNoValueTF{#8}
560      {\cs_set:Nn\__tmp_eight:n{\exp_not:c{##1}}}
561      {\cs_set:Nn\__tmp_eight:n{\exp_not:c{#8}}}
562    %
563    \cs_gset:Nn\g_tmpb_cs:nnnnnnnn{#9}
564    %
565    \cs_gset:Nn\g_tmpa_cs:n{
566      \tl_set:Nn\l_tmpa_tl{##1}
567      \use:x{
568        \exp_not:N\g_tmpb_cs:nnnnnnnn
569          {\l_tmpa_tl}
570          \__tmp_two:n{\l_tmpa_tl}
571          \__tmp_three:n{\l_tmpa_tl}
572          \__tmp_four:n{\l_tmpa_tl}
573          \__tmp_five:n{\l_tmpa_tl}
574          \__tmp_six:n{\l_tmpa_tl}
575          \__tmp_seven:n{\l_tmpa_tl}
576          \__tmp_eight:n{\l_tmpa_tl}}
577      }
578    \exp_args:Nx\tl_map_inline:nn{#1}
579      {\tl_if_blank:nTF{##1}
580        {}
581        {\g_tmpa_cs:n{\__mathcommand_getbasename:n{##1}}}
582      }
583 }
```

```
584 \cs_new:Nn\__mathcommand_getbasename:n{
585     \tl_if_single:nTF{#1}{
586         \token_if_cs:NTF#1
587             {\cs_to_str:N#1}
588             {#1}
589     }{#1}
590 }
```

7.8 Deactivating macros

```
591 \bool_new:N\__mathcommand_disabled_error_bool
592 \bool_set_true:N\__mathcommand_disabled_error_bool
593 \bool_new:N\__mathcommand_disabled_suggest_original_bool
594 \bool_set_true:N\__mathcommand_disabled_suggest_original_bool
595 \bool_new:N\__mathcommand_force_enabled_bool
596 \bool_set_false:N\__mathcommand_force_enabled_bool

597 \cs_new:Nn\__mathcommand_to_disabled_help_tl:N
598     {mathcommand_disabled_help_\cs_to_str:N#1_tl}

599 \cs_new:Nn\__mathcommand_error:nn
600 {   \msg_new:nnn{mathcommand}{#1}{#2}
601     \msg_error:nn{mathcommand}{#1}
602 }
603 \cs_new:Nn\__mathcommand_dc_error:n
604 {   \msg_new:nnn{mathcommand}{disabled~command}{#1}
605     \msg_error:nn{mathcommand}{disabled~command}
606 }
607 \cs_new:Nn\__mathcommand_dc_warning:n
608 {   \msg_set:nnn{mathcommand}{disabled~command}{#1}
609     \msg_warning:nn{mathcommand}{disabled~command}
610 }
611 \cs_new:Nn\mathcommand_disabled_error:N
612 {   \bool_if:NTF\__mathcommand_disabled_error_bool
613     {\exp_args:Nx\__mathcommand_dc_error:n}
614     {\exp_args:Nx\__mathcommand_dc_warning:n}
615       {\exp_not:n{\\The~command~\string#1~is~disabled.~Instead:\\ }
616        \bool_if:NT\__mathcommand_disabled_suggest_original_bool
617          {Use~\exp_not:c{\__mathcommand_to_storetl:N#1} for~the~original~macro.}
618        \exp_not:v{\__mathcommand_to_disabled_help_tl:N#1}}}
619 }

620 \msg_new:nnn{mathcommand}{unknown~command}
621     {\\The~control~sequence~#1 is~not~defined \\Origin~macro:~#2}

622 \NewDocumentCommand\disablecommand{m}{
623     \tl_map_function:nN{#1}\mathcommand_disablecommand:N
624 }
625
626
627 \cs_new:Nn\mathcommand_disablecommand:N
628   { \cs_if_exist:NTF#1{
629     \__mathcommand_check_definable:nNT#1\disablecommand
```

```
630      {\tl_if_exist:cTF{\__mathcommand_to_disabled_help_tl:N#1}
631        {}
632        {\storecommand#1
633          \tl_new:c{\__mathcommand_to_disabled_help_tl:N#1}
634         \renewcommand#1
635            {\bool_if:NF
636                \__mathcommand_force_enabled_bool
637                {\mathcommand_disabled_error:N#1}
638           \use:c{\__mathcommand_to_storetl:N#1}}}}
639      }
640      }{
641        \msg_error:nnnn{mathcommand}{unknown~command}{#1}{\disablecommand}
642      }
643 }

644 \NewDocumentCommand\suggestcommand{mm}
645     { \cs_if_exist:NTF#1{
646        \cs_if_exist:cTF{\__mathcommand_to_disabled_help_tl:N#1}{
647            \tl_put_right:cx{\__mathcommand_to_disabled_help_tl:N#1}{\exp_not:N\\\tl_to_str:n{#
648          }{
649          \__mathcommand_error:nn{command~not~disabled}
650            {\\\string\suggestcommand :~Command~#1~has~not~been~disabled.}
651          }
652        }
653        {  \__mathcommand_error:nn{unknown~command}
654            {\\\string\suggestcommand :~Unknown~command~#1.}
655        }
656 }
```

7.9   Options

```
657 \keys_define:nn { mathcommand }{
658   disabled .multichoice:,
659   disabled / silent .code:n = {\bool_set_true:N\__mathcommand_force_enabled_bool },
660   disabled / error .code:n = {\bool_set_false:N\__mathcommand_force_enabled_bool
661                                \bool_set_true:N\__mathcommand_disabled_error_bool},
662   disabled / warning .code:n = {\bool_set_false:N\__mathcommand_force_enabled_bool
663                                \bool_set_false:N\__mathcommand_disabled_error_bool},
664 }
665 \ProcessKeysOptions { mathcommand } % Parses the option list
666 \NewDocumentCommand\mathcommandconfigure{ m }
667   {\keys_set:nn{ mathcommand}{ #1} }
```

7.10   Testing control sequence for definition

This part of the code is an import of a previous version of xparse (Frank Mittelbach, Chris Rowley, David Carlisle, The LaTeX Project). It is done in a very ugly hack: the code is copied and past, and `mathcommand` substituted to `xparse` wherever necessary.

```
668 \cs_new_protected:Npn \__mathcommand_check_definable:nNT #1
669   { \tl_trim_spaces_apply:nN {#1} \__mathcommand_check_definable_aux:nN }
```

```
670 \group_begin:
671   \char_set_catcode_active:n { '? }
672   \cs_new_protected:Npn \__mathcommand_check_definable_aux:nN #1#2
673     {
674       \group_begin:
675       \tl_if_single_token:nTF {#1}
676         {
677           \int_set:Nn \tex_escapechar:D { 92 }
678           \exp_args:Nx \tl_if_empty:nTF
679             { \exp_args:No \str_tail:n { \token_to_str:N #1 } }
680             {
681               \exp_args:Nx \char_set_lccode:nn
682                 { ' \str_head:n {#1} } { '? }
683               \tex_lowercase:D { \tl_if_eq:nnTF {#1} } { ? }
684                 { \group_end: \use_iii:nnn }
685                 { \group_end: \use_i:nnn }
686             }
687             { \group_end: \use_iii:nnn }
688         }
689         { \group_end: \use_ii:nnn }
690         {
691           \__kernel_msg_error:nnxx { mathcommand } { not-definable }
692             { \tl_to_str:n {#1} } { \token_to_str:N #2 }
693         }
694         {
695           \__kernel_msg_error:nnxx { mathcommand } { not-one-token }
696             { \tl_to_str:n {#1} } { \token_to_str:N #2 }
697         }
698     }
699 \group_end:
```

7.11   Closure

```
700 \ExplSyntaxOff
```