

The `latex-lab-math` code*

Frank Mittelbach, Joseph Wright, L^AT_EX Project

v0.6j 2024-11-26

Abstract

This is an experimental prototype. It captures math material (basically okay, but the interfaces for packages aren't yet there) and tags the material (which is not yet anywhere near the final state). That part is provided for experimentation and to gather feedback, etc.

Contents

1	Introduction	2
2	Math capture	3
2.1	Code level interfaces	3
2.2	Document level interfaces	3
3	Math tagging	3
3.1	Code requirements	3
3.2	Inline math	4
3.3	Display math	5
3.4	Associated Files	5
3.5	Automatic mathml creation with <code>luamml</code>	6
3.6	Summary of math options	6
4	Known current bugs, etc.	8
4.1	Capture/grabbing problems	8
4.2	Fake math	9
4.2.1	Open problems	10
4.3	Processor	10
4.4	Other problems	10
4.5	Other ToDos	10

*

5	The Implementation	11
5.1	File declaration	11
5.2	Setup	11
5.3	Data structures	11
5.4	Tagging tools	12
5.5	Code related to AF	12
5.6	Mathstyle detection	22
5.7	Tagging options	23
5.8	Sockets	23
5.8.1	Main inline math sockets	23
5.8.2	Main display math sockets	24
5.8.3	Internal sockets	25
5.9	Interface commands	30
5.10	Content grabbing	30
5.11	Token-by-token inline grabbing	33
5.12	Marking math environments	35
5.13	Document commands	39
5.14	<code>\everymath</code> and <code>\everydisplay</code>	40
5.15	Modifying kernel environments	41
5.16	Disable math grabbing in the <code>begindocument</code> hook	42
5.17	Modifying <code>amsmath</code>	42
	Index	48

1 Introduction

Todo: update all the documentation! Both here and (what little there is!) in the implementation section.

Tagging math involves a variety of tasks that require that math is captured before the typesetting

- When typesetting the math MC-tags and structure commands must be inserted at the begin and the end, and perhaps also around lines or other subparts of the equation.
- The source and/or a mathml-representation of the source must be available so that it can be (perhaps after some preprocessing) be used in an associated file or in an alternate text
- It must be possible to measure the math for, e.g., a `bbox` setting.

This file implements capture of all math mode material at the outer level, i.e., a formula is captured in its entirety with inner text blocks (possibly containing further math) absorbed as part of the formula. For example,

$$\left[a \in A \text{ for all } a \right]$$

would only result in a single capture of the tokens “`a \in A \text{ for all } a`”.

2 Math capture

In the current setup

- $\$, \backslash(\dots\backslash)$ and $\$\$$ grab (through a command in $\everymath/cseverydisplay$) if the boolean $\l_@@_collected_bool$ is false. If the boolean is true they behave normally and can for example contain verbatim.
- All (registered) environments grab their body regardless of the state of the boolean. For equation , equation* and math this is a change as they no longer can contain verbatim.
- BUG: $\llbracket \dots \rrbracket$ grabs if $\l_@@_collected_bool$ is false. If it is true it falls back to equation* and then errors because this can't find the end.

2.1 Code level interfaces

$\math_register_env:n$	$\math_register_env:n \{<env>\}$
$\math_register_env:nn$	$\math_register_env:nn \{<env>\} \{<options>\}$

Registers the $\langle env \rangle$ as a math environment which should be captured and made available. This is necessary for all top-level math mode environments: low-level errors may result if these are not correct set up. One or more key-value $\langle options \rangle$ may also be given:

arg-spec The argument specification taken by the beginning of the environment; this is used to remove non-mathematical material.

$\math_processor:n$	$\math_processor:n \{<tokens>\}$
----------------------	-----------------------------------

Declares that the captured math content should be passed to the $\langle tokens \rangle$, which will receive the environment type as #1 and the content as #2. The processing is done before the typesetting. It is not applied if $\l\ifmeasuring@$ is true.

2.2 Document level interfaces

\RegisterMathEnvironment	$\RegisterMathEnvironment [<options>] \{<env>\}$
----------------------------	--

Registers the $\langle env \rangle$ as a math environment which should be captured and made available. This is necessary for all top-level math mode environments: low-level errors may result if these are not correct set up. One or more key-value $\langle options \rangle$ may also be given:

arg-spec The argument specification taken by the beginning of the environment; this is used to remove non-mathematical material.

3 Math tagging

3.1 Code requirements

The tagging code has to handle

- the embedding into the surrounding. This means

- closing and reopening MC-chunks
- closing and reopening text/P-structures
- handling interferences of the tagging code with penalties and spacing.
- the actual tagging which means to do some or all of the following tasks:
 - setup content for an associated source file
 - setup content for an associated mathml file
 - setup content for the /Alt key
 - setup content for the /ActualText key
 - setup attributes
 - add associated files
 - add a Formula structure
 - surround subparts (e.g., lines) with Formula sub structures (perhaps with their own set of additional content)
 - surround elements of the equation with mathml structure elements (currently only luatex with luamml)

3.2 Inline math

The embedding code is added through the sockets

- `tagsupport/math/inline/begin`
- `tagsupport/math/inline/end`

The sockets simply push and pop the MC currently. Without tagging they use the noop-plug.

The actual tagging is in done through the sockets

- `tagsupport/math/inline/formula/begin` This socket takes the math as second argument and its code should output it for typesetting. The `default` plug of the socket calls these three internal sockets for the tagging support:
 - `tagsupport/math/content` This should set up the various content variables (empty variables are ignored by the structure code and so can be used to suppress a setting).
 - `tagsupport/math/struct/begin` This calls `\tag_struct_begin:n`. It should also write the associated files if needed.
 - `tagsupport/math/substruct/begin` this handles subparts. TODO: does it really make sense in inline math to have that??
- `tagsupport/math/inline/formula/end` This socket ends the formula structure(s). The `default` plug calls these internal sockets:
 - `tagsupport/math/substruct/end`
 - `tagsupport/math/struct/end`

3.3 Display math

to be written

3.4 Associated Files

The current code allows the attachment of two types of associated file to the Formula structure: the L^AT_EX source and a MathML representation. Technically both can be attached—AF is an array of file references—in practice there can be problems with PDF consumers: e.g., ngpdf used both and so showed the equation twice (this has been corrected in the newest version) and Foxit seems to see only the first AF in the array (so we attach the mathml as first file).

The L^AT_EX source can be (and is) attached automatically. It can be suppressed by an option with `math/tex/AF=false`, see below.

The MathML is attached if the files `\jobname-mathml.html` and/or `\jobname-luamml-mathml.html` are found and if they contains a suitable MathML snippet for the current formula. If the files contain more than one suitable snippet (as identified by the hash) the first one is used. `\jobname-luamml-mathml.html` is automatically generated (see below section 3.5) and read after `\jobname-mathml.html`. This means that `\jobname-mathml.html` can contain improved versions of a formula.

The MathML processing can be suppressed globally by emptying the list of mathml files with `math/mathml/sources=`. Locally for a formula `math/mathml/AF=false` can be used.

For a MathML representation a file with such representations must be provided. If the equation is numbered the numbering should be part of the MathML as the L_bl sub-structure is ignored if an MathML is used (see https://github.com/foxiitsoftware/PDF__UA-2).

The MathML representation is given in a special format. It is meant to be a valid html file that can be viewed in a browser. For this it can start with `<!DOCTYPE html><html>` and end with `</html>` It should have the extension `.html`. The `<mathml>` content is read with special catcodes, so can contain ambersands, hashes, comment chars and unmatched braces such as `<mo>{</mo>`

The file should contain a number of representations in this format:

```
<div>
  <h2>\mml {key}</h2>
  <p>{source}</p>
  <p>{hash}</p>
  <math {attributes} >
    {mathml}
  </math>
</div>
```

The keywords `<div>`, `<h2>\mml`, `<p>`, `<math`, `</math>` `</div>` are required as they are used to delimit the arguments by the L^AT_EX code.

`<key>` and `<source>` are only used for debugging, they help to identify the equation referred by this representation. The source should be used correctly escaped `&` and `<` so that it gives valid html!

`<attributes>` is not required either, but can, e.g., contain attributes to improve the display in a browser:

```
<math alttext="\mathbf{G}" class="ltx_Math" display="inline">
```

It can also contain the name space declaration: `xmlns="http://www.w3.org/1998/Math/MathML"`¹

By default the code tries at the begin of the document to read a file `\jobname-mathml.html` in the `html`-format. The file name can be changed with `mathml/setfiles={filename1,filename2}` (without extension, `html` is added automatically). If there is a list, all files are loaded. If a file doesn't exist it is ignored, only an info is written to the log.

Currently every MathML-snippet from a file is embedded into the PDF, it is not checked first if it is actually used (simply writing everything to the PDF is a bit easier than keeping everything in memory and also means that the snippets are one after the other in the PDF).

As mentioned above the MathML-AF can be suppressed for the equations in a group with `math/mathml/AF=false`, or completely by setting `math/mathml/sources=` in the preamble.

Files embedded in a PDF can be listed in the attachments panel of a PDF viewer. This is probably not so useful for lots of small files (but one could create collections), but as long as PDF editors or viewers don't offer proper support to access the AF it can help so have them there. The MathML are added by default, but the \LaTeX source not. This can be changed with `viewer/pane/mathsource=true` (anywhere in the document) and `viewer/pane/mathml=false` (in the preamble, before the external file is read).

3.5 Automatic mathml creation with luamml

If `lualatex` and the package `unicode-math` is used the package `luamml` is loaded and will automatically generate the file `\jobname-luamml-mathml.html` with `mathml` representations of all math formulas. This file is then used in subsequent compilations and works also with `pdflatex`.

The generation of the file can be suppressed (in the preamble) with `math/mathml/luamml/write=false`.

If the package `unicode-math` is not used, the loading of `luamml` and with it the generation of the file can be forced with `math/mathml/luamml/load=true` or `math/mathml/luamml/write=true` but be aware that it is then possible that various symbols are mapped to the wrong Unicode code points.

The package `luamml` is still quite experimental and the output should be checked. The `\jobname-luamml-mathml.html` file may be previewed in a browser although you may need to add additional `css` or `javascript` declarations to enable browser support for all `mathml` constructs.

3.6 Summary of math options

The following options exist to make math more accessible:

ActualText An **ActualText** can be placed on structure elements, but can also be added in the stream on a **BDC** marker with a **Span** tag (normally an independant marker without an **MCID** number, it is not clear yet if it can be used on a **MC-chunk**). The content is a text string, typically one or a few Unicode characters. **ActualText** is meant to replace the content and should only be used on small entities, e.g., to define the semantic or the Unicode code point of a symbol. **ActualText** is not supported by all PDF reader. It is also unknown where it should be used at best

¹But it is probably not needed and only blows up the PDF.

(in a structure element, or on an independent Span-BDC) and what happens if it is used in more than one place.

enabled by default? False

how to enable/disable No interface yet. `ActualText` can only be added on the `Formula` structure element by changing the `tagssupport/math/content` or some other socket. For a BDC marker one can, e.g., use

```
\pdf_string_from_unicode:nnN{utf16/hex}{€}\l_tmpa_tl
\pdf_bdc:ee{Span}{/ActualText\l_tmpa_tl}content\pdf_emc:
```

There should be no pagebreak in the `<content>` and the BDC should be correctly nested into tagging, so, e.g., a `\leavevmode` should be issued before the `bdc` command.

Consumer support in part and in part buggy, needs tests ...

Alt Like `ActualText` the `Alt` key can be used on structure elements and on `Span` in the stream. It should contain a description of the content and is mainly meant for images. PDF/UA-1, which views math formulas as illustrations, mandates the key also for `Formula` structure elements.

enabled by default? false unless PDF/UA-1 is detected, then it is enabled in the `begindocument/end` hook (this will be reconsidered when it is clear, that the use of `Alt` does not shadow `mathml`). It can be enabled for all engines and PDF versions.

enable/disable `\tagpdfsetup{math/alt/use}` (local boolean, so can be used on individual equations)

default value A template text (stored in `\l_@@_content_template_tl`) starting with `LaTeX formula starts`.

user value No interface currently provided. This needs optional arguments or an external setup command. See <https://github.com/latex3/tagging-project/discussions/717>.

source-AF The \LaTeX -source of the equation can be attached as an associated file with mime-type `application/Fx-tex`. The `AFRelationship` is `Source`. The source is embedded without expansion. This means that targets of references and macros are not resolved. The files are by default not shown in the `EmbeddedFiles` pane, this can be enabled with `viewer/pane/mathsource=true`. If an A-standard is used, it must be one that allows embedded files, e.g., A-4f.

enabled by default? true for all engines and PDF versions

enable/disable `\tagpdfsetup{math/tex/AF}` (local boolean, so can be used on individual equations)

default value source code including dollars or environment name.

consumer support Currently only `ngpdf` makes use of it: if there is no `mathml` it passes the source to `mathjax`.

luamml The following options make (with `lualatex`) use of the `luamml` package. `luamml` is currently automatically loaded (at the end of the preamble) if `unicode-math` has been detected. The loading can be forced or suppressed with `\tagpdfsetup{math/mathml/luamml/}` `luamml` affects all `math`, locally it can be stopped with `math/mathml/ignore`, or by using the commands described in the package.

mathml-AF A mathml representation of the equation can be attached to the structure. The configuration possibilities are rather complex as the keys have to control three different tasks: The *generation* of the file with the mathml fragments, the *reading* and *embedding* of the mathml fragments, and the *association* of a mathml fragment to a specific equation.

generation With pdfL^AT_EX mathml fragments can not be generated automatically, but a file with dummy fragments for every equation will be written if `\tagpdfsetup{math/mathml/write-dummy}` is issued in the preamble.

With luaL^AT_EX a file with mathml fragments will be created automatically if the package `luamml` has been loaded (see above).

reading and embedding By default the code will read and embed mathml from `\jobname-mathml.html` and `\jobname-luamml-mathml.html` in this order and the first fragment with a new hash value will be inserted. The list of sources and their order can be changed with the key `math/mathml/sources`, setting that to an empty value suppresses the loading mathml associated files completely. For efficiency reasons it embeds math fragments directly, there is no check yet if the fragment is actually used.

The files are by default shown in the EmbeddedFiles pane, this can be disabled with `viewer/pane/mathml=false`.

attaching A mathml fragment is currently attached as an associated file to an Formula if the hash of the source matches the hash of the fragment. This is not a perfect test: equations with the same source and so the same hash can have different mathml representation, e.g., if there are references or commands or counters in the equation. This will change in a feature version. The attachment can be suppressed locally with `math/mathml/AF=false`. The mathml fragment will still be embedded in the PDF!

TODO: adapt test

mathml structure elements Mathml structure elements can be used in PDF 2.0 directly. In PDF 1.7. one could theoretically use them if one declares a role mapping first, (this can be done with `\tagpdfsetup{role/mathml-tags}`) which maps all to `Span`. But such a role mapping currently breaks reading, e.g. in Adobe, and so it is not recommended.

Automatic generation of structure elements is only possible with `lualatex`. It requires that the packages `luamml` and `tagpdf` have been loaded.

enabled by default? false

enable/disable `\tagpdfsetup{math/mathml/structelem}` (local setting, so can be used with grouping on individual equations).

consumer support Needs more tests.

4 Known current bugs, etc.

4.1 Capture/grabbing problems

1. Incorrect grabbing of $-$-math when there is also explicit $-$-math within a *text environment* that is itself within the math that should all be grabbed. For example,$$

`$a\begin{minipage}{1cm}$b$\end{minipage}$`

would only result in the capture of the tokens “`a\begin_{minipage}{1cm}`”. This can be avoided by an additional brace group:

`$a{\begin{minipage}{1cm}$b$\end{minipage}}$`

2. Similar incorrect grabbing with `$$` also.
3. The grabbing, for all the display environments (and `\` `\]`), needs to deal with nesting: `amsmath` contains code for this.
4. The math can't contain verbatim and verbatim-like commands. This is nothing new for the `amsmath` environments but changes `$` and `\[` `\]` and `equation*` (see, e.g., tagging-project issue #30).
5. Begin and end of the math or math environment can not be hidden in commands. For example `>{${}1<{${}` in a `tabular` would lead to errors. Defining `\[` to fall back to `equation*` doesn't work if `equation*` is a grabbing environment.
6. The behaviour of `\[...]` is faulty. See above.

4.2 Fake math

In a number of places in L^AT_EX math commands (mainly `$`) is used only for technical reason, e.g., to access a math font, to setup a symbol or to use `\vcenter`.

The code identifies such fake math mostly by making use of the `\m@th` command where two methods are used for the automatic detection:

- After grabbing math content the code checks if the content contains the token `\m@th` and if yes it doesn't call the processor before reinserting the content and perhaps adding tagging code. This method requires that the math can be grabbed (e.g. that the end dollar is visible) and that the `\m@th` is visible. It applies for example in `\@iiiparbox` where the code from `$_\vcenter` to `\m@th$` is grabbed and put back. It does not work for example for `tabular` where the dollars and the `\m@th` token are spread around over three commands. `tabular` needs therefore manual intervention. A look in the list of usages (in `usage-of-m@th.md`) justifies this approach. All usages are either not math at all, or related to small elements that probably shouldn't be grabbed and processed on their own.
- `\m@th` is redefined so that it sets the boolean `\l_@@_collected_bool` to true. If `\m@th` is used inside math that has been grabbed this doesn't change much as the boolean is set by the grabbing anyway. For usages outside math the benefit is not so clear: The setting avoids that in L^AT_EX_ε the epsilon is processed as math, but it also prevents that the content of the `amsmath` command `\boxed` is processed as math. It means that if one wants to reenable math processing inside some (fake) math one has to do it after `\m@th` calls.

4.2.1 Open problems

1. The grabbing code doesn't pass the info that it detected a `\m@th` token. This means that the tagging code has to do the same check (and doesn't do this in all cases yet).
2. Commands are missing to locally disable the grabbing and processing, e.g., to handle `tabular`.
3. It must be checked if setting the boolean in `\m@th` really makes sense or if commands like `\LaTeXe` should be handled manually.

4.3 Processor

The grabbed math is at first passed to the processor. The processor is not called in a measuring phase (from the `amsmath \ifmeasuring@`) and if the `\m@th` token is detected. It is not quite clear what purpose the processor has. As it is a public interface it can't be used for internal code. And typesetting happens later and the processor can't really change this. Currently it is mostly used for debugging and messages. If the `\m@th` is found the `\l_@@_fakemath_bool` is set, so if the code is changed this must be preserved.

4.4 Other problems

1. The presence of `\m@th` in association with `\ensuremath` does not necessarily indicate fakemath. This is because wanting `mathsurround` to be zero is very reasonable and common, *even when the math is genuine* (and hence needs to be collected).
TODO: this claim needs some examples.
2. User-defined environments can create problems; but this area, of new, copied and changed environments, has not yet been developed.

Joseph wrote, inter alia:
My thinking [regarding] `\RegisterMathEnvironment`
- (New) Math environments should not be created-then-patched, but only generated by a [(future)] dedicated command (`\DeclareMathEnvironment`, presumably)
- Math environments created with `ltxcmd` [commands] should not be copied, . . .
- Package authors should be able to manually set up math environments with a public boolean.

4.5 Other ToDos

1. Add (some of) the math display commands that were "lifted from plain", e.g., `\displaylines \eqalign{??}`.
2. The `breqn` packages changes catcodes and that isn't yet covered by our mechanism.
3. `\intertext` is not correctly taken into account by the code splitting multiline math into subformulas.

`\MaybeStop` (temporarily) not executed, as it is unknown on Chris' system.

5 The Implementation

```
1 <@@=math>
2 <*kernel>
```

5.1 File declaration

```
3 \ProvidesFile{latex-lab-math.ltx}
4     [\ltlabmathdate\space
5     v\ltlabmathversion\space
6     Grab all the math(s) and tag it (experiments)]
7
8 Temp loading ...
9 \AddToHook{begindocument/before}{\RequirePackage{latex-lab-testphase-block}}
10 \ExplSyntaxOn
```

5.2 Setup

Loading `amsmath` is an absolute requirement: this avoids needing to have conditional definitions and deals with how to define `\[/\]` neatly.

```
9 \AddToHook{begindocument/before}{\RequirePackage { amsmath } }
```

5.3 Data structures

`\l__math_collected_bool` Tracks whether math mode material has been collected, which happens inside `amsmath` environments as well as those handled directly here. If true following math will not grab and/or process. See 2 for details.

```
10 \bool_new:N \l__math_collected_bool
```

`\l__math_fakemath_bool` Tracks whether math mode material has been identified as fake math during the grabbing phase, which happens currently if the grabbed contents contains the `\m@th` token.

```
11 \bool_new:N \l__math_fakemath_bool
```

Change first tl name below: 'env' => 'info'?

Or do we need an extra

`\g__math_grabbed_env_tl`
`\g__math_grabbed_math_tl`

`\g__math_grabbed_env_tl` contains the name of the math environment (`math` in the case of inline math, `\g__math_grabbed_math_tl` the math content.

```
12 \tl_new:N \g__math_grabbed_env_tl
13 \tl_new:N \g__math_grabbed_math_tl
```

`\l__math_tmpa_tl` Temporary variables

`\l__math_tmpa_skip`
`\l__math_tmpa_str`

```
14 \tl_new:N \l__math_tmpa_tl
15 \skip_new:N \l__math_tmpa_skip
16 \str_new:N \l__math_tmpa_str
```

`\l__math_content_alt_tl` Temporary variables to hold math content that should be used in actual or alt text and
`\l__math_content_actual_tl` stored as AF.
`\l__math_content_AF_tl`

```

17 \tl_new:N \l__math_content_alt_tl
18 \tl_new:N \l__math_content_actual_tl
19 \tl_new:N \l__math_content_AF_source_tl
20 \tl_new:N \l__math_content_AF_source_tmpa_tl
21 \tl_new:N \l__math_content_AF_mathml_tl

```

5.4 Tagging tools

The following commands implement small tagging code chunks. This should probably be collected and moved into tagpdf later.

`__tag_tool_close_P:` This closes a P/text-chunk, both the MC and the structure and increases the counter manually.

```

22 \cs_new_protected:Npn \__tag_tool_close_P:
23 {
24   \tag_if_active:T
25   {
26     \tag_mc_end: %end P-chunk, should perhaps be \tag_mc_end_push: ...
27     \__tag_gincr_para_end_int:
28     \__tag_check_para_end_show:mn{red}{} %debug: show para
29     \tag_struct_end:
30   }
31 }

```

(End of definition for __tag_tool_close_P:.)

We add also an attribute.

```

32 \tl_new:N\l__math_attribute_class_tl
33 \tagpdfsetup
34   {role/new-attribute = {inline}    {/O /Layout /Placement/Inline},
35   role/new-attribute = {display}   {/O /Layout /Placement/Block},
36 }

```

5.5 Code related to AF

Booleans to handle the options.

```

\l__tag_math_texsource_AF_bool
\l__tag_math_texsource_pane_bool
\l__tag_math_mathml_AF_bool
\g__tag_math_mathml_AF_bool
\l__tag_math_mathml_pane_bool
\l__tag_math_alt_bool
\g__tag_math_luamml_tl

```

The variable `\g__tag_math_luamml_tl` is initially 0 and the user key can set it to -1 or 1. This allows to distinguish the unset case from a value set by the user.

```

37 \bool_new:N\l__tag_math_texsource_AF_bool
38 \bool_new:N\l__tag_math_texsource_pane_bool
39 \bool_new:N\l__tag_math_mathml_AF_bool
40 \bool_new:N\g__tag_math_mathml_AF_bool
41 \bool_new:N\l__tag_math_mathml_pane_bool
42 \bool_new:N\l__tag_math_alt_bool
43 \tl_new:N\g__tag_math_luamml_tl
44 \tl_gset:Nn\g__tag_math_luamml_tl {0}

```

```

\g__math_mathml_total_int
\g__math_mathml_int
\g__math_math_total_int
\g__math_mathml_AF_found_int
\g__math_mathml_AF_attached_int

```

`\g__math_mml_total_int` records the mathml fragments read in. `\g__math_mml_int` records the mathml fragments read in with a different hash. `\g__math_AF_total_int` records the number of math structures that try to attach a mathml AF. `\g__math_AF_found_int` records the number of math structures for which a fitting mathml is found. `\g__math_AF_attached_int` records the number of math structures which got a mathml fragment (if mathml-AF are not disabled locally this should be the equal to the previous number).

```

45 \int_new:N\g__math_mathml_total_int
46 \int_new:N\g__math_mathml_int
47 \int_new:N\g__math_math_total_int
48 \int_new:N\g__math_mathml_AF_found_int
49 \int_new:N\g__math_mathml_AF_attached_int

```

```

\l__tag_math_mathml_files_clist

```

A sequence to store the file list for the mathml. We also check the luamml file.

```

50 \clist_new:N\l__tag_math_mathml_files_clist
51 \clist_put_right:Ne\l__tag_math_mathml_files_clist
52   {\c_sys_jobname_str-mathml,\c_sys_jobname_str-luamml-mathml}

```

This is the internal variant of the `\mml` command.

```

\__math_AF_mml:nnnn

```

```

53 \cs_new_protected:Npn \__math_AF_mml:nnnn #1 #2 #3 #4
54   {%#1 number, #2 tex source for debugging, #3 hash, #4 mathml
55   {
56     \int_gincr:N \g__math_mathml_total_int

```

mathml with the same hash should be included only once:

```
57 \tl_if_exist:cF { g__math_mathml_#3_tl }
58 {
59   \int_gincr:N \g__math_mathml_int
```

a simple Desc key, take care that it is a valid string!

```
60   \pdfdict_put:nne {l_pdffile/Filespec} {Desc}{(mathml-#1)}
61   \pdffile_embed_stream:nnN {#4}{mathml-#1.xml}\l__math_tmpa_tl
```

not strictly necessary but makes the files visible in the file attachment page

```
62   \bool_if:NT \l__tag_math_mathml_pane_bool
63     {\pdfmanagement_add:nne {Catalog/Names}{EmbeddedFiles}{\l__math_tmpa_tl}}
64   \tl_new:c{g__math_mathml_#3_tl}
65   \tl_gset_eq:cN{g__math_mathml_#3_tl}\l__math_tmpa_tl
66   }
67 }
```

(End of definition for __math_AF_mml:nnnn.)

The html reader.

```
68 \cs_new_protected:Npn \__math_AF_html_reader:w#1</h2>#2<p>#3</p>#4<p>#5</p>#6<math{
69   \begingroup
70   \char_set_catcode_other:N\{
71   \char_set_catcode_other:N\}
72   \char_set_catcode_other:N\#
73   \char_set_catcode_other:N\%
74   \__math_AF_html_reader_verb:w{#1}{#3}{#5}<math
75   }
76 \cs_new_protected:Npn \__math_AF_html_reader_verb:w#1#2#3#4~</div>{
77   \endgroup
78   \__math_AF_mml:nnnn{#1}{#2}{#3}{#4}
79   }
```

As with luatex we write two files we define a few constants for the shared texts.

```
\c__math_mathml_write_init_tl
\l__math_mathml_write_before_tl
\c__math_mathml_write_after_tl
\c__math_mathml_write_final_tl
80 \tl_const:Nn \c__math_mathml_write_init_tl
81 {
82   <!DOCTYPE~html>
83   \iow_newline:
84   <html>
85   \iow_newline:
86   }
87 \tl_new:N \l__math_mathml_write_before_tl
88 \tl_const:Nn \c__math_mathml_write_after_tl
89 {
90   \iow_newline:
91   </div>
92   \iow_newline:
93   }
94 \tl_const:Nn \c__math_mathml_write_final_tl
95 {
96   </html>
97   }
```

(End of definition for \c__math_mathml_write_init_tl and others.)

`h/mathml/write/prepare (socket)` To prepare the hash and the starting command we use a socket, so that both the dummy and luamml can make use of it.

```
98 \socket_new:nn {tagsupport/math/mathml/write/prepare}{0}
```

On (*plug*)

```
99 \socket_new_plug:nnn{tagsupport/math/mathml/write/prepare}{0n}
100 {
101   \str_set:NV\l__math_tmpa_str\l__math_content_AF_source_tl
102   \str_replace_all:Nnn\l__math_tmpa_str{&}{&amp;}
103   \str_replace_all:Nnn\l__math_tmpa_str{<}{&lt;}
104   \tl_set:Nn \l__math_mathml_write_before_tl
105   {
106     <div>
107     \iow_newline:
108     <h2>\c_backslash_str mml\c_space_tl \int_use:N \g__math_math_total_int </h2>
109     \iow_newline:
110     <p>\l__math_tmpa_str</p>
111     \iow_newline:
112     <p>\l__math_content_hash_tl </p>
113     \iow_newline:
114   }
115 }
```

With luatex we automatically generate mathml with luamml if the package can be loaded and unicode-math is detected. We start the process in the `begindocument/end` hook so that the reading from a previous compilation can happen before!

For other engines, for future name changes and in case luamml is not loaded we provide some commands

```
116 \cs_new_protected:Npn\__math_provide_luamml_commands:
117 {
118   \providecommand\luamml_flag_structelem:{}
119   \cs_if_free:NT \luamml_structelem:
120   {
121     \cs_set_eq:NN\luamml_structelem:\luamml_flag_structelem:
122   }
123   \providecommand\luamml_flag_process:{}
124   \cs_if_free:NT \luamml_process:
125   {
126     \cs_set_eq:NN\luamml_process:\luamml_flag_process:
127   }
128   \providecommand\luamml_flag_ignore:{}
129   \cs_if_free:NT \luamml_ignore:
130   {
131     \cs_set_eq:NN\luamml_ignore:\luamml_flag_ignore:
132   }
133 }
134 \sys_if_engine luatex:TF
135 {
```

Temporary (!) fixes for endarray

```
136 \cs_new_protected:Npn \__math_correct_luamml_array_patches:
137 {
138   \AddToHook{package/array/after}
```

```

139 {
140   \cs_set:Npn \endarray
141   {
142     \tbl_crcl:n{endarray}
143     \__luamml_array_save_array:
144     \egroup
145     \UseTaggingSocket{tbl/finalize}
146     \tbl_restore_outer_cell_data:
147     \egroup
148     \mode_if_math:T { \__luamml_array_finalize_array: }
149     \@arrayright
150     \gdef \@preamble {}
151   }
152   \cs_set:Npn \@classz
153   {
154     \@classx
155     \@tempcnta \count@
156     \prepnext@tok
157     \@addtopreamble {
158       \ifcase \@chnum
159         \hfil
160         \hskip 1sp
161         \d@llarbegin
162         \cs_if_eq:NNTF \d@llarbegin \begingroup {
163           \insert@column
164           \d@llarend
165         } {
166           \__luamml_array_init_col:
167           \insert@column
168           \luamml_flag_save:nn {} {mtd}
169           \d@llarend
170           \__luamml_array_finalize_col:w 0~
171         }
172         \do@row@strut
173         \hfil
174       \or
175         \hskip 1sp
176         \d@llarbegin
177         \cs_if_eq:NNTF \d@llarbegin \begingroup {
178           \insert@column
179           \d@llarend
180         } {
181           \__luamml_array_init_col:
182           \insert@column
183           \luamml_flag_save:nn {} {mtd}
184           \d@llarend
185           \__luamml_array_finalize_col:w 1~
186         }
187         \do@row@strut
188         \hfil
189       \or
190         \hfil
191         \hskip 1sp
192         \d@llarbegin

```



```

193         \cs_if_eq:NNTF \d@llarbegin \begingroup {
194             \insert@column
195             \d@llarend
196         } {
197             \__luamml_array_init_col:
198             \insert@column
199             \luamml_flag_save:nn {} {mtd}
200             \d@llarend
201             \__luamml_array_finalize_col:w 2~
202         }
203         \do@row@strut
204     \or
205         \setbox \ar@mcclbox \vbox \@startpbox { \@nextchar }
206         \insert@pcolumn
207         \@endpbox
208         \ar@align@mccl
209         \do@row@strut
210     \or
211         \vtop \@startpbox { \@nextchar }
212         \insert@pcolumn
213         \@endpbox
214         \do@row@strut
215     \or
216         \vbox \@startpbox { \@nextchar }
217         \insert@pcolumn
218         \@endpbox
219         \do@row@strut
220     \fi
221 }
222 \prepnext@tok
223 }
224 }
225 }
226 \AddToHook{begindocument/before}
227 {
228     \str_case:on \g__math_luamml_load_tl
229     {
230         { 1 } {
231             \RequirePackage { luamml }
232             \__math_correct_luamml_array_patches:
233             \AddToHook{begindocument/end}
234             {
235                 \__math_luamml_activate_write:
236             }
237         }
238         {-1 } {
239             \AddToHook{begindocument/end}
240             {
241                 \msg_note:nnnn { tag }
242                 { luamml-status }{ disabled }{ not~create }
243             }
244         }
245         { 0 }
246         {

```

```

247     \@ifpackageloaded { unicode-math }
248     {
249         \RequirePackage { luamml }
250         \__math_correct_luamml_array_patches:
251         \AddToHook{begindocument/end}
252         {
253             \__math_luamml_activate_write:
254         }
255     }
256     { \msg_warning:nn { tag }{ unicode-math-missing } }
257 }
258 }
259 \__math_provide_luamml_commands:
260 }
261 }
262 {
263     \__math_provide_luamml_commands:
264 }
265 \msg_new:nnn { tag }{ luamml-status }
266 {
267     luamml~has~been~#1~and~will~#2~an~MathML~file.
268 }
269
270 \msg_new:nnn { tag }{ unicode-math-missing }
271 {
272     The~package~unicode-math~is~missing\\
273     luamml~will~not~create~an~MathML~file.\\
274     To~avoid~this~warning~load~unicode-math~\\
275     or~disable~luamml~with~\\
276     \tl_to_str:n{\tagpdfsetup{math/mathml/luamml/load=false}}\\
277     or~force~luamml~with~\\
278     \tl_to_str:n{\tagpdfsetup{math/mathml/luamml/load=true}}
279 }
280 \cs_new_protected:Npn \__math_luamml_activate_write:
281 {
282     \bool_if:NT \g__math_luamml_write_bool
283     {

```

to avoid that nothing is written in the first run, we must activate the sockets:

```

284     \bool_gset_true:N\g__tag_math_mathml_AF_bool
285     \AssignSocketPlug{tagsupport/math/struct/begin}{mathml-AF}
286     \AssignSocketPlug{tagsupport/math/struct/end}{mathml-AF}
287     \AssignSocketPlug{tagsupport/math/substruct/begin}{single}
288     \AssignSocketPlug{tagsupport/math/substruct/end}{single}
289     \int_set:Nn \l__luamml_pretty_int { 7 }
290     \RegisterFamilyMapping\symsymbols{oms}
291     \RegisterFamilyMapping\symletters{oml}
292     \AssignSocketPlug{tagsupport/math/mathml/write/prepare}{On}
293     \iow_new:N \g__math_luamml_iow
294     \iow_open:Nn \g__math_luamml_iow {\c_sys_jobname_str-luamml-mathml.html}
295     \iow_now:Ne \g__math_luamml_iow { \c__math_mathml_write_init_tl }
296     \cs_new:Npn \__math_luamml_output_hook:n ##1
297     {
298         \tl_if_empty:NF \l__math_mathml_write_before_tl

```

```

299         {
300         \iow_now:Ne \g__math_luamml_iow
301         {
302         \l__math_mathml_write_before_tl
303         ##1
304         \c__math_mathml_write_after_tl
305         }
306         }
307     }
308     \__luamml_register_output_hook:N \__math_luamml_output_hook:n

```

At the end of the document we must finish and close the file:

```

309     \AddToHook{enddocument/afterlastpage}
310     {
311         \iow_now:Ne \g__math_luamml_iow
312         { \c__math_mathml_write_final_tl }
313         \iow_close:N \g__math_luamml_iow
314     }
315     \msg_note:nnnn { tag }
316     { luamml-status }{ enabled }{ create }
317 }
318 }

```

And now keys to activate/deactivate luamml feature

`\g__math_luamml_load_tl` This variable will be used to suppress the loading of luamml altogether.

```

319 \tl_new:N \g__math_luamml_load_tl
320 \tl_gset:Nn \g__math_luamml_load_tl {0}

```

`\g__math_luamml_write_bool` This variable decides if luamml writes a mathml altogether.

```

321 \bool_new:N \g__math_luamml_write_bool
322 \bool_gset_true:N \g__math_luamml_write_bool

```

`__math_luamml_ignore:` Internal variants of the luamml commands, that can be remapped if needed.
`__math_luamml_structelem:`

```

323 \cs_new:Npn \__math_luamml_structelem: {}
324 \cs_new:Npn \__math_luamml_ignore: {}

```

(End of definition for `__math_luamml_ignore:` and `__math_luamml_structelem:.`)

```

325 \msg_new:nnn { tag }{ PDF-2.0-recommended }
326 {
327     The~key~#1~will~not~work~properly~with~PDF~#2.\\
328     Switching~to~PDF~2.0~is~recommended.
329 }
330 \keys_define:nn { __tag / setup }
331 {

```

At first a key to suppress the loading altogether

```

332     math/mathml/luamml/load .choice: ,
333     math/mathml/luamml/load/true .code:n = {\tl_gset:Nn \g__math_luamml_load_tl{1}},
334     math/mathml/luamml/load/false .code:n = {\tl_gset:Nn \g__math_luamml_load_tl{-1}},
335     math/mathml/luamml/load .default:n = true,
336     math/mathml/luamml/load .usage:n=preamble,

```

A key to activate math structure elements.

```

337     math/mathml/structelem .choice:,
338     math/mathml/structelem/true .code:n =
339     {
340         \pdf_version_compare:NnT < {2.0}
341         {
342             \msg_warning:nnne { tag }{ PDF-2.0-recommended }
343             { math/mathml/structelem }{ \pdf_version: }
344         }
345         \cs_set:Npn\__math_luamml_structelem:{\luamml_structelem;}
346         \cs_set:Npn\__math_luamml_ignore:{\luamml_ignore;}
347     },
348     math/mathml/structelem/false .code:n =
349     {
350         \cs_set_eq:NN\__math_luamml_structelem:\prg_do_nothing:
351         \cs_set_eq:NN\__math_luamml_ignore:\prg_do_nothing:
352     },
353     math/mathml/structelem .default:n = true,

```

and a key to call the ignore flag. This should only be used locally.

```

354     math/mathml/ignore .code:n = {\luamml_ignore:},
355     math/mathml/luamml/write .choice:,
356     math/mathml/luamml/write/true .code:n =
357     {
358         \tl_gset:Nn \g__math_luamml_load_tl{1}
359         \bool_gset_true:N \g__math_luamml_write_bool
360     },
361     math/mathml/luamml/write/false .code:n =
362     {
363         \bool_gset_false:N \g__math_luamml_write_bool
364     },
365     math/mathml/luamml/write .default:n = true,
366     math/mathml/luamml/write .usage:n=preamble,

```

alias keys for compatibility

```

367     math/mathml/luamml .bool_gset:N = \g__math_luamml_write_bool,
368     math/mathml/luamml .usage:n=preamble
369 }

```

`port/math/mathml/write (socket)` This writes a html-dummy with the hash and the math content. This should be optional, so it uses a socket that can be disabled

```

370 \socket_new:nn {tagssupport/math/mathml/write}{0}

```

`On (plug)`

```

371 \socket_new_plug:nnn{tagssupport/math/mathml/write}{On}
372 {
373     \iow_now:Ne \g__math_writedummy_iow
374     {
375         \l__math_mathml_write_before_tl
376         <math></math>
377         \c__math_mathml_write_after_tl
378     }
379 }

```

And now a key to activate the socket.

```

380
381 \keys_define:nn { __tag / setup }
382   {
383     math/mathml/write-dummy .code:n =
384     {
385       \bool_gset_true:N \g__tag_math_mathml_AF_bool
386       \tl_if_exist:NF\g__math_writedummy_iow
387       {
388         \iow_new:N \g__math_writedummy_iow
389         \iow_open:Nn \g__math_writedummy_iow
390         {
391           \c_sys_jobname_str-mathml-dummy.html
392         }
393         \iow_now:Ne \g__math_writedummy_iow
394         {
395           \c__math_mathml_write_init_tl
396         }
397         \AssignSocketPlug {tagsupport/math/mathml/write/prepare}{0n}
398         \AssignSocketPlug {tagsupport/math/mathml/write}{0n}
399         \AddToHook{enddocument/afterlastpage}
400         {
401           \iow_now:Ne \g__math_writedummy_iow
402             { \c__math_mathml_write_final_tl }
403           \iow_close:N \g__math_writedummy_iow
404         }
405       }
406     },
407     math/mathml/write-dummy .usage:n=preamble
408   }

\_math_AF_process_mathml_files:
409 \box_new:N\l__math_tmpa_box
410 \cs_new_protected:Npn \__math_AF_process_mathml_files:
411   {
412     \hbox_set:Nn \l__math_tmpa_box
413     {
414       \pdfdict_put:nnn { l_pdffile/Filespec }{AFRelationship} { /Supplement }
415       \pdfdict_put:nne
416         { l_pdffile }{Subtype}
417         { \pdf_name_from_unicode_e:n{application/mathml+xml} }
418       \char_set_catcode_other:N \#
419       \cs_set_eq:NN\mml \__math_AF_html_reader:w
420       \clist_map_inline:Nn \l__tag_math_mathml_files_clist
421       {
422         \file_if_exist:nTF {##1.html}
423         {
424           \typeout{Info:~reading~mathml~file~##1}
425           \file_input:n {##1.html}
426           \bool_gset_true:N\g__tag_math_mathml_AF_bool
427         }
428         {
429           \typeout{Info:~mathml~file~##1~does~not~exist}%info message
430         }

```

```

431     }
432   }
433   \bool_if:NT\g__tag_math_mathml_AF_bool
434     {
435     \typeout{Info:~Activating~mathml~support}
436     \AssignSocketPlug{tagsupport/math/struct/begin}{mathml-AF}
437     \AssignSocketPlug{tagsupport/math/struct/end}{mathml-AF}
mathml handling doesn't like subparts, so we disable them for now:
438     \AssignSocketPlug{tagsupport/math/substruct/begin}{single}
439     \AssignSocketPlug{tagsupport/math/substruct/end}{single}
440     \AddToHook{enddocument/info}
441     {
442     \iow_term:n{MathML~statistic}
443     \iow_term:n{=====}
444     \iow_term:e{==>~\int_use:N\g__math_mathml_total_int\c_space_tl
MathML~fragments~read}
445     \iow_term:e{==>~\int_use:N\g__math_mathml_int\c_space_tl
different~MathML~fragments}
446     \iow_term:e{==>~\int_use:N\g__math_math_total_int\c_space_tl
math~fragments~found}
447     \iow_term:e{==>~\int_use:N\g__math_mathml_AF_found_int\c_space_tl
fitting~MathML~AF~found}
448     \iow_term:e{==>~\int_use:N\g__math_mathml_AF_attached_int\c_space_tl
MathML~AF~attached}
449     }
450   }
451 }
452 \AddToHook{begindocument}{\__math_AF_process_mathml_files:}
453

```

(End of definition for `__math_AF_process_mathml_files:.`)

5.6 Mathstyle detection

In some cases we need to detect the mathstyle used in a `\mathchoice` command and to disable/enable tagging in the unused branches. This is currently only used in the `amstext` command `\text` but is perhaps also needed in other cases, so we create a general command.

```

\l__math_mathstyle_int
\g__math_mathchoice_int
mathstyle
458 \int_new:N \l__math_mathstyle_int
459 \int_new:N \g__math_mathchoice_int
460 \property_new:n{mathstyle}{now}{-1}{\int_use:N \l__math_mathstyle_int }

```

(End of definition for `\l__math_mathstyle_int`, `\g__math_mathchoice_int`, and `mathstyle`. This function is documented on page ??.)

For now internal, but perhaps will need a public version. The command should be used in every branch of a `\mathchoice` (with the correct mathstyle number) and with an unique label (which should be the same in every branch). `\g__math_mathchoice_int` can be, e.g., increased before the mathchoice and then used.

```

\__math_tag_if_mathstyle:nn

```

```

461 \cs_new_protected:Npn \__math_tag_if_mathstyle:nn #1 #2
462   {%#1 refers to label

```

```

463   %#2 is a number for the mathstyle (typically 0,2,4,6)
464   {
465     \int_set:Nn \l__math_mathstyle_int {#2}
466     \property_record:nn {#1} { mathstyle }
467     \int_compare:nNnTF { \property_ref:nn {#1}{ mathstyle} } = { #2 }
468       { \tag_resume:n{\mathchoice} }{ \tag_suspend:n{\mathchoice} }
469   }
470 \cs_generate_variant:Nn \__math_tag_if_mathstyle:nn {en}

```

(End of definition for `__math_tag_if_mathstyle:nn`.)

5.7 Tagging options

```

471 \keys_define:nn { __tag / setup }
472   {
473     math/mathml/sources .clist_set:N = \l__tag_math_mathml_files_clist,
474     math/alt/use        .bool_set:N = \l__tag_math_alt_bool,
475     viewer/pane/mathml .bool_set:N = \l__tag_math_mathml_pane_bool,
476     viewer/pane/mathml .initial:n = true,
477     viewer/pane/mathsource .bool_set:N = \l__tag_math_texsource_pane_bool,
478     math/mathml/AF .bool_set:N = \l__tag_math_mathml_AF_bool,
479     math/mathml/AF .initial:n = true,
480     math/tex/AF .bool_set:N = \l__tag_math_texsource_AF_bool,
481     math/tex/AF .initial:n = true
482   }

```

alt is required for pdf/UA-1. TODO: l3pdfmeta should support this test.

```

483 \AddToHook{begindocument/end}
484   {
485     \str_if_eq:eeT
486       {1}
487       {
488         \exp_last_unbraced:Ne\use_i:nn
489           {\GetDocumentProperties{document/pdfstandard-UA}}
490         \c_empty_tl\c_empty_tl
491       }
492     {
493       \bool_if:NF \l__tag_math_alt_bool
494         {
495           \typeout{PDF/UA-1~detected.~Enabling~alt~text~on~Formula}
496         }
497       \bool_set_true:N\l__tag_math_alt_bool
498     }
499   }

```

5.8 Sockets

5.8.1 Main inline math sockets

`support/math/inline/begin` (*socket*) The first two sockets are meant to embed inline math into the surrounding (so to
`support/math/inline/end` (*socket*) close/reopen, e.g., MC-chunks). The other two implement the actual formula struc-
`h/inline/formula/begin` (*socket*) ture. The formula sockets are despite their naming not symmetric: the begin socket is
`math/inline/formula/end` (*socket*) issued after the math has started, while the end socket is after the math!

```

500 \socket_new:nn {tagsupport/math/inline/begin}{0}
501 \socket_new:nn {tagsupport/math/inline/end}{0}

```

```

502 \socket_new:nn {tagsupport/math/inline/formula/begin}{2} %
503 \socket_new:nn {tagsupport/math/inline/formula/end}{0}

```

MC (*plug*)

```

504 \socket_new_plug:nnn
505   {tagsupport/math/inline/begin}
506   {MC}
507   {\tag_mc_end_push:}
508 \socket_new_plug:nnn
509   {tagsupport/math/inline/end}
510   {MC}
511   {\tag_mc_begin_pop:n{}}

```

We probably will want to test different tagging recipes.

default (*plug*)

```

512 \socket_new_plug:nnn
513   {tagsupport/math/inline/formula/begin}
514   {default}

515   { \tagpdfparaOff
516     \_math_luamml_structelem:
517     \tag_socket_use:n{math/content}
518     \tag_socket_use:n{math/struct/begin}

```

TODO: does inline math need subformula handling?

```

519     % inner formula if multiple parts (not really implemented yet)
520     \tag_socket_use:n{math/substruct/begin}
521     #2
522     \tag_socket_use:n{math/end}
523   }
524 \socket_new_plug:nnn
525   {tagsupport/math/inline/formula/end}
526   {default}
527   {
528     \socket_use:n{tagsupport/math/substruct/end}
529     \socket_use:n{tagsupport/math/struct/end}
530   }

```

5.8.2 Main display math sockets

`support/math/display/begin (socket)` The first two sockets are meant to embed display math into the surrounding (so to `support/math/display/end (socket)` close/reopen, e.g., MC-chunks and P-structure). The other two implement the actual `math/display/formula/begin (socket)` formula structure. The formula sockets are despite their naming not symmetric: the `math/display/formula/end (socket)` begin socket is issued after the math has started, while the end socket is after the math!

```

531 \socket_new:nn {tagsupport/math/display/begin}{0}
532 \socket_new:nn {tagsupport/math/display/end}{0}
533 \socket_new:nn {tagsupport/math/display/formula/begin}{2} %
534 \socket_new:nn {tagsupport/math/display/formula/end}{0}

```

default (*plug*)

```

535 \socket_new_plug:nnn
536   {tagsupport/math/display/begin}
537   {default}

```



```

538 { \_tag_tool_close_P: }
539 \socket_new_plug:nnn
540 {tagsupport/math/display/end}
541 {default}
542 {
543 }

```

default (*plug*)

```

544 \socket_new_plug:nnn
545 {tagsupport/math/display/formula/begin}
546 {default}
547 {
548   \tagpdfparaOff
549   \_math_luamml_structelem:
550   \tag_socket_use:n{math/content}
551   \tag_socket_use:n{math/struct/begin}
552   \tag_socket_use:n{math/substruct/begin}
553   #2
554   \tag_socket_use:n{math/end}
555 }
556 \socket_new_plug:nnn
557 {tagsupport/math/display/formula/end}
558 {default}
559 {
560   \socket_use:n{tagsupport/math/substruct/end}
561   \socket_use:n{tagsupport/math/struct/end}
562 }

```

5.8.3 Internal sockets

\l__math_content_template_tl

The default text used as alt or actual text.

```

563 \tl_new:N\l__math_content_template_tl
564 \tl_set:Nn \l__math_content_template_tl
565 {
566   LaTeX~ formula~ starts~
567   \exp_not:N\begin{\g__math_grabbed_env_tl}
568   \c_space_tl
569   \exp_not:V\g__math_grabbed_math_tl
570   \c_space_tl
571   \exp_not:N\end{\g__math_grabbed_env_tl}
572   \c_space_tl LaTeX~ formula~ ends~
573 }

```

\l__math_texsource_template_tl

The default text used as texsource

```
574 \tl_new:N\l__math_texsource_template_tl
575 \tl_const:Nn\c__math_inline_env_tl {math}
576 \tl_set:Nn \l__math_texsource_template_tl
577   {
578     \tl_if_eq:NNTF\g__math_grabbed_env_tl\c__math_inline_env_tl
579     {
580       $
581       \exp_not:V\g__math_grabbed_math_tl
582       $
583     }
584     {
585       \exp_not:N\begin{\g__math_grabbed_env_tl}
586       \exp_not:V\g__math_grabbed_math_tl
587       \exp_not:N\end{\g__math_grabbed_env_tl}
588     }
589   }
```

`tagsupport/math/content` (*socket*) The math content is stored in associated files and used for actual and alternative text. As the exact text is still unclear we use a socket to be able to test variants. The socket should set all four tl vars above, if needed to identical values. It can use the two variables `\g__math_grabbed_env_tl` and `\g__math_grabbed_math_tl`

```
590 \socket_new:nn {tagsupport/math/content}{0}
```

Some default sockets to set the contents. TODO: think about naming convention. TODO: think how this should be organized so that one has options to change from the outside and so that there are less repetitions.

`actual+source` (*plug*)

```
591 \socket_new_plug:nnn
592   {tagsupport/math/content}
593   {actual+source}
594   {
595     \tl_set:Ne\l__math_content_actual_tl
596     {
597       \l__math_content_template_tl
598     }
599     \tl_set:Ne \l__math_content_AF_source_tl
600     {
601       \l__math_texsource_template_tl
602     }
603     \tl_set:Nn \l__math_content_AF_mathml_tl {}
604     \tl_set:Nn \l__math_content_alt_tl {}
605   }
```

`alt+source` (*plug*)

```
606 \socket_new_plug:nnn
607   {tagsupport/math/content}
608   {alt+source}
609   {
610     \tl_set:Ne\l__math_content_alt_tl
```

```

611     {
612       \l__math_content_template_tl
613     }
614   \tl_set:Nc \l__math_content_AF_source_tl
615     {
616       \l__math_texsource_template_tl
617     }
618   \tl_set:Nn \l__math_content_AF_mathml_tl {}
619   \tl_set:Nn \l__math_content_actual_tl {}
620 }

621 \socket_assign_plug:nn {tagsupport/math/content}{alt+source}

```

port/math/struct/begin (*socket*) For the main structure we use a socket too. This allows, e.g., to create a special one
support/math/struct/end (*socket*) for luamml which setups additional objects. The begin socket can use the two variables

```

\g__math_grabbed_env_tl and \g__math_grabbed_math_tl
622 \socket_new:nn {tagsupport/math/struct/begin}{0}
623 \socket_new:nn {tagsupport/math/struct/end}{0}

```

default (*plug*) TODO: think about some naming convention ...

```

624 \socket_new_plug:nnn
625   {tagsupport/math/struct/begin}
626   {default}
627   {
628     \bool_if:NTF\l__tag_math_texsource_AF_bool
629     { \tl_set_eq:NN \l__math_content_AF_source_tmpa_tl \l__math_content_AF_source_tl }
630     { \tl_clear:N \l__math_content_AF_source_tmpa_tl }
631     \tl_if_eq:NnTF\g__math_grabbed_env_tl {math}
632       {
633         \tl_set:Nn\l__math_attribute_class_tl{inline}
634       }
635       {
636         \tl_set:Nn\l__math_attribute_class_tl{display}
637       }
638     \bool_if:NF\l__tag_math_alt_bool
639     { \tl_set:Nn \l__math_content_alt_tl{} }
640     \tag_struct_begin:n
641     {
642       tag=Formula,
643       attribute-class=\l__math_attribute_class_tl,
644       texsource      = \l__math_content_AF_source_tmpa_tl,
645       title-o        = \g__math_grabbed_env_tl,
646       actualtext     = \l__math_content_actual_tl,
647       alt            = \l__math_content_alt_tl
648     }
649   }
650 \socket_new_plug:nnn
651   {tagsupport/math/struct/end}
652   {default}
653   { \tag_struct_end: }
654
655 \socket_assign_plug:nn {tagsupport/math/struct/begin}{default}
656 \socket_assign_plug:nn {tagsupport/math/struct/end}{default}

```

`mathml-AF (plug)` This socket tries to add a mathml-AF to formula. It is activated if a mathml.html has been found and loaded. As it disturbs the reading of the AF it currently deactivates the /Alt key, unless it has been reenabled with `math/alt/use=true`

```
657 \cs_generate_variant:Nn \str_mdfive_hash:n {o}
658 \tl_new:N\l__math_content_hash_tl
```

we need to save the grabbed math:

```
659 \tl_new:N\l__math_grabbed_math_tl
```

the socket definition

```
660 \socket_new_plug:nnn
661 {tagsupport/math/struct/begin}
662 {mathml-AF}
663 {
664   \int_gincr:N\g__math_math_total_int
665   \tl_set:Ne\l__math_content_hash_tl
666   {\str_mdfive_hash:o { \l__math_content_AF_source_tl }}
667   \tl_set_eq:NN\l__math_grabbed_math_tl\g__math_grabbed_math_tl
668   \tl_if_eq:NnTF\g__math_grabbed_env_tl {math}
669   {
670     \tl_set:Nn\l__math_attribute_class_tl{inline}
671   }
672   {
673     \tl_set:Nn\l__math_attribute_class_tl{display}
674   }
675   \bool_if:NF\l__tag_math_alt_bool
676   { \tl_set:Nn \l__math_content_alt_tl{} } }
```

debugging option. TODO: hide in debug key.

```
677   \tl_if_exist:cTF { g__math_mathml_ \l__math_content_hash_tl _tl }
678   {
679     \int_gincr:N\g__math_mathml_AF_found_int
680     \bool_if:NTF \l__tag_math_mathml_AF_bool
681     {
682       \int_gincr:N\g__math_mathml_AF_attached_int
683       \typeout {Inserting~mathml~with~Hash~\l__math_content_hash_tl}
684     }
685     {
686       \typeout {Ignoring~mathml~with~Hash~\l__math_content_hash_tl}
687     }
688   }
689   {
690     \bool_if:NT \l__tag_math_mathml_AF_bool
691     {
692       \typeout {WARNING:~mathml~missing~for~hash~\l__math_content_hash_tl}
693     }
694   }
695   \socket_use:n {tagsupport/math/mathml/write/prepare}
696   \socket_use:n {tagsupport/math/mathml/write} % write hash if request
697   \bool_if:NTF\l__tag_math_texsource_AF_bool
698   { \tl_set_eq:NN \l__math_content_AF_source_tmpa_tl \l__math_content_AF_source_tl }
699   { \tl_clear:N \l__math_content_AF_source_tmpa_tl }
700   \tag_struct_begin:n
701   {
702     tag=Formula,
```

```

703     attribute-class=\l__math_attribute_class_tl, %
704     AFref          =
705     \bool_if:NT\l__tag_math_mathml_AF_bool
706     {
707         \cs_if_exist_use:c {g__math_mathml_ \l__math_content_hash_tl _tl}
708     },
709     texsource      = \l__math_content_AF_source_tmpa_tl, % should be after mathml AF!
710     title-o        = \g__math_grabbed_env_tl,      %
711     alt            = \l__math_content_alt_tl
712 }
713 }

```

not really needed but looks more symmetric:

```

714 \socket_new_plug:nnn
715   {tagsupport/math/struct/end}
716   {mathml-AF}
717   {
718     \tag_struct_end:
719   }

```

`t/math/substruct/begin (socket)` This holds the code to handle subparts of the formula.

```

\port/math/substruct/end (socket) 720 \socket_new:nn {tagsupport/math/substruct/begin}{0}
721 \socket_new:nn {tagsupport/math/substruct/end}{0}

```

`default (plug)`

```

722 \socket_new_plug:nnn
723   {tagsupport/math/substruct/begin}
724   {default}
725   { \grabaformulapartandstart }
726 \socket_new_plug:nnn
727   {tagsupport/math/substruct/end}
728   {default}
729   {
730     \tagmcend
731     \if@subformulas
732     \tagstructend
733     \fi
734   }
735 \socket_assign_plug:nn {tagsupport/math/substruct/begin}{default}
736 \socket_assign_plug:nn {tagsupport/math/substruct/end}{default}

```

`single (plug)` We need an option to disable subparts as it is unclear if consumers can handle them:

```

737 \socket_new_plug:nnn
738   {tagsupport/math/substruct/begin}
739   {single}
740   {
741     \typeout{====>subpart~splitting~deactivated}
742     \typeout{====>grabbed~math=\meaning\g__math_grabbed_math_tl}
743     \tag_mc_begin:n{ }
744   }
745 \socket_new_plug:nnn
746   {tagsupport/math/substruct/end}
747   {single}
748   { \tag_mc_end: }

```

`tagsupport/math/end` (*socket*) A socket used at the end of the math (before the closing dollar(s)) which can, e.g., set a flag for luamml.

```

749 \socket_new:nn {tagsupport/math/end}{0}

750 \socket_assign_plug:nn {tagsupport/math/inline/begin}{MC}
751 \socket_assign_plug:nn {tagsupport/math/inline/end}{MC}
752 \socket_assign_plug:nn {tagsupport/math/inline/formula/begin}{default}
753 \socket_assign_plug:nn {tagsupport/math/inline/formula/end}{default}
754 \socket_assign_plug:nn {tagsupport/math/display/begin}{default}
755 \socket_assign_plug:nn {tagsupport/math/display/end}{default}
756 \socket_assign_plug:nn {tagsupport/math/display/formula/begin}{default}
757 \socket_assign_plug:nn {tagsupport/math/display/formula/end}{default}

```

5.9 Interface commands

`_math_process:nn` A no-op place-holder; the internal wrapper means that it does not need to be concerned with internals.

```

\_math_process:Vn
\_math_process_auxi:nn
\_math_process_auxii:nn
758 \cs_new_protected:Npn \_math_process:nn #1#2
759 {
760   \legacy_if:nF { measuring@ }
761   {
762     \tl_if_in:nnTF {#2} { \m@th }
763     { \bool_set_true:N\l_math_fakemath_bool }
764     { \tl_trim_spaces_apply:nN {#2} \_math_process_auxi:nn {#1} }
765   }
766 }
767 \cs_generate_variant:Nn \_math_process:nn { V }
768 \cs_new_protected:Npn \_math_process_auxi:nn #1#2
769 {
770   \tl_gset:Nn \g__math_grabbed_env_tl {#2}
771   \tl_gset:Nn \g__math_grabbed_math_tl {#1}
772   \_math_process_auxii:nn {#2} {#1}
773 }
774 \cs_new_protected:Npn \_math_process_auxii:nn #1#2 { }

```

(End of definition for `_math_process:nn`, `_math_process_auxi:nn`, and `_math_process_auxii:nn`.)

`\math_processor:n` A simple installer

```

775 \cs_new_protected:Npn \math_processor:n #1
776 { \cs_set_protected:Npn \_math_process_auxii:nn ##1##2 {#1} }

```

(End of definition for `\math_processor:n`. This function is documented on page 3.)

5.10 Content grabbing

`_math_grab_dollar:w` Top-level function to handle grabbing of inline math mode delimited by \$ tokens. We provide two different ways to do that: a token-by-token one that can be used everywhere, and a fast delimited one that does not work anywhere that the end \$ token may be hidden, most obviously in tabulars. The function here is therefore set up as a variable starting point.

```

777 \cs_new_protected:Npn \_math_grab_dollar:w { \_math_grab_dollar_delim:w }

```

After grabbing inline math material, there is again common processing independent of mechanism of collection.

```
778 \cs_new_protected:Npn \__math_grab_dollar:n #1
779 {
```

We need to do processing first as this picks up “fake” math mode: that information is needed below.

```
780 \__math_process:nn { math } {#1}
```

We do not want math tagging in fakemath or when measuring, We also do not want math tagging if tagging has been suspended.

```
781 \bool_lazy_any:nTF
782 {
783   {\legacy_if_p:n { measuring@ }}
784   { \l__math_fakemath_bool }
785   { \tl_if_blank_p:n {#1} }
786 }
787 {
788   \__math_luamml_ignore:
789   #1 $ % $
790 }
791 {
792   \tag_socket_use:n {math/inline/begin} %end P-MC
```

We do not use a tagging socket here, so that the argument (the math) is not lost, tagging-project issue 661.

```
793 \tag_socket_use:nnn {math/inline/formula/begin}{}{#1}
794 $ % $
795 \tag_socket_use:n {math/inline/formula/end}
796 \tag_socket_use:n {math/inline/end} % restart P-MC
797 }
798 }
```

(End of definition for __math_grab_dollar:w and __math_grab_dollar:n.)

`__math_grab_dollar_delim:w` Grab up to a single \$, for inline math mode, suppressing any processing if the token is `\m@th` found in the content.

```
799 \cs_new_protected:Npn \__math_grab_dollar_delim:w #1 $ % $
800 { \__math_grab_dollar:n {#1} }
```

(End of definition for __math_grab_dollar_delim:w.)

`__math_grab_dollardollar:w` And for the classical T_EX display structure.

```
801 \cs_new_protected:Npn \__math_grab_dollardollar:w % $$
802 #1 $$
803 {
804   \tl_if_blank:nF {#1}
805   {
806     \__math_process:nn { equation* } {#1}
807     \tag_socket_use:n {math/display/begin}
808     \tag_socket_use:nn{math/display/formula/begin}{}{#1}
809   }
810   $$
811 }
```

The end code is added through a `\aftergroup` so we store it inside a command.

```
812 \cs_new_protected:Npn \__math_tag_dollardollar_display_end:
813 {
814   % \typeout{== tag dollar\dollar display end}
815   % \ShowTagging{struct-stack}
816   \para_raw_end:
```

The `\postdisplaypenalty` was temporarily set to 10000 inside the display and the `\belowdisplayskip` and the `\belowdisplaysshortskip` was negated, so whatever was inserted it should have been a negative skip. Whatever skip was added we pick it up value up here, so that we can correct the spacing after the tagging code was inserted.

```
817   \l__math_tmpa_skip \lastskip
818   \tag_socket_use:n{math/display/formula/end}
```

Now we add a skip without introducing a page break possibility, that should bring the current vertical position back to the point where \TeX would add the penalty and the “below skip”.

```
819   \nobreak
820   \skip_vertical:n { -\l__math_tmpa_skip } % remove the negative belowdisplayskip
```

Then we finally add the real stuff:

```
821   \penalty \postdisplaypenalty
822   \skip_vertical:n { -\l__math_tmpa_skip } % insert the correct skip
823   \@doendpe           % this has no \end{...} to take care of it
824 }
825
```

(End of definition for __math_grab_dollardollar:w.)

`__math_grab_inline:w` Collect inline math content and deal with the need to move to math mode.

```
826 \cs_new_protected:Npn \__math_grab_inline:w % \langle
827   #1 \rangle
828 {
829   \tl_if_blank:nF {#1}
830   {
831     $ #1 $
832   }
833   \bool_set_false:N \l__math_collected_bool
834 }
```

(End of definition for __math_grab_inline:w.)

`__math_grab_eqn:w` For the most common use of `\[/\]`: turn into an environment.

```
835 \cs_new_protected:Npn \__math_grab_eqn:w % \[
836   #1 \]
837 {
838   % \typeout{collected? = \bool_if:NTF \l__math_collected_bool {true}{false}}
839   \begin { equation* } #1 \end { equation* }
840 }
```

(End of definition for __math_grab_eqn:w.)

5.11 Token-by-token inline grabbing

Grabbing inline math token-by-token is more involved. The mechanism here is essentially a simplified version of that originally seen in `collcell` and refined in `siunitx`. We make use of the fact that in math mode spaces are ignored, so we have to deal with only N-type tokens and groups. Furthermore, there is no need to look inside groups, so the only special cases are a small selection of N-type tokens.

`\l__math_grabbed_tl` For collection of the material piecewise.

```
841 \tl_new:N \l__math_grabbed_tl
```

`\l__math_grab_env_int` Needed to count up the number of nested environments encountered.

```
842 \int_new:N \l__math_grab_env_int
```

`__math_grab_dollar_loop:` The lead-off here establishes a group: we need that as we will have to be careful in the way `\cr` is handled and ensure this is only manipulated whilst grabbing. The main loop is then started.

`__math_grab_loop:`

```
843 \cs_new_protected:Npn \__math_grab_dollar_loop:
844 {
845   \group_begin:
846   \tl_clear:N \l__math_grabbed_tl
847   \__math_grab_loop:
848 }
849 \cs_new_protected:Npn \__math_grab_loop:
850 {
851   \peek_remove_spaces:n
852   {
853     \peek_meaning:NTF \c_group_begin_token
854     { \__math_grab_loop_group:n }
855     { \__math_grab_loop_token:N }
856   }
857 }
```

(End of definition for `__math_grab_dollar_loop:` and `__math_grab_loop:.`)

`__math_grab_loop_group:n` Handling of grabbed groups is pretty easy.

`__math_grab_loop_store:n`

```
858 \cs_new_protected:Npn \__math_grab_loop_group:n #1
859 { \__math_grab_loop_store:n { #1 } }
860 \cs_new_protected:Npn \__math_grab_loop_store:n #1
861 {
862   \tl_put_right:Nn \l__math_grabbed_tl {#1}
863   \__math_grab_loop:
864 }
```

(End of definition for `__math_grab_loop_group:n` and `__math_grab_loop_store:n.`)

`__math_grab_loop_token:N` Filter out the special cases: for performance reasons, use a hash table approach rather than a loop (*cf.* `collcell`).

`__math_grab_loop_$:`

`__math_grab_loop_\backslash:`

```
865 \cs_new_protected:Npn \__math_grab_loop_token:N #1
```

`__math_grab_loop_\begin:`

```
866 {
```

`__math_grab_loop_\end:`

`__math_grab_loop_\ignorespaces:`

`__math_grab_loop_\unskip:`

`__math_grab_loop_\textonly@unskip:`

```

867 \cs_if_exist_use:cF
868 { __math_grab_loop_ \token_to_str:N #1 : }
869 { \__math_grab_loop_store:n {#1} }
870 }
871 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N $ : }
872 { \__math_grab_loop_end: }
873 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N \ : }
874 {
875 \int_compare:nNnTF \l__math_grab_env_int = 0
876 { \__math_grab_loop_newline: }
877 { \__math_grab_loop_store:n { \ } }
878 }

```

In contrast to `collcell`, nesting is tracked by counting `\begin/\end` pairs: this is needed in case there is a tabular-like construct containing `\` inside a cell. As a result, the end-of-tabular can be detected without checking the name argument: if `\end` is encountered at nesting level 0, we've hit the end of a cell. In that case, end the row and leave the environment to clean up.

```

879 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N \begin : }
880 {
881 \int_incr:N \l__math_grab_env_int
882 \__math_grab_loop_store:n { \begin }
883 }
884 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N \end : }
885 {
886 \int_compare:nNnTF \l__math_grab_env_int = 0
887 {
888 \__math_grab_loop_newline:
889 \end
890 }
891 {
892 \int_decr:N \l__math_grab_env_int
893 \__math_grab_loop_store:n { \end }
894 }
895 }
896 \tl_map_inline:nn { \ignorespaces \unskip \textonly@unskip }
897 {
898 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N #1 : }
899 { \__math_grab_loop: }
900 }

```

(End of definition for `__math_grab_loop_token:N` and others.)

`__math_grab_loop_newline:` To allow collection of tokens in the part of the `\halign` template after `#`, we need \TeX to see the primitive with the loop token in the right place. That is done by re-defining `\cr` at present. Ideally there would be a socket in the definition of `tabular`, etc., to handle this: there is also the need to examine in interaction with `longtable`, which also redefines `\cr`.

```

901 \cs_new_protected:Npn \__math_grab_loop_newline:
902 {
903 \if_false: { \fi:
904 \cs_set_protected:Npn \cr
905 {
906 \__math_grab_loop:

```

```

907     \tex_cr:D
908   }
909   \if_false: } \fi:
910   \\\
911 }

```

(End of definition for `_math_grab_loop_newline:.`)

`_math_grab_loop_end:` Clean up and pass on.

```

912 \cs_new_protected:Npn \_math_grab_loop_end:
913 {
914   \exp_args:NNV \group_end:
915   \_math_grab_dollar:n \l\_math_grabbed_tl
916 }

```

(End of definition for `_math_grab_loop_end:.`)

5.12 Marking math environments

A general mechanism for math mode environments that do not grab their content (*cf.* most `amsmath` environments).

`\l_math_env_name_tl` To allow us to carry out “special effects”

```

917 \tl_new:N \l\_math_env_name_tl

```

Here we set up specialised handling of environments. The idea for the `arg-spec` key is that if an environment takes arguments, we don’t worry during the main grabbing. Rather, we remove the arguments from the grabbed content and forward only the payload. That is done by (ab)using `ltxcmd`.

```

918 \keys_define:nn { \_math }
919 {
920   arg-spec .code:n =
921     {
922       \ExpandArgs { c } \DeclareDocumentCommand
923       { \_math_env \l\_math_env_name_tl _aux: }
924       {#1}
925       { \_math_env_forward:w }
926     }
927 }

```

`\math_register_env:nn` Set up to capture environment content and make available.
`\math_register_env:n`
`\RegisterMathEnvironment`

```

928 \cs_new_protected:Npn \math_register_env:nn #1#2
929 {
930   \tl_set:Nn \l\_math_env_name_tl {#1}
931   \keys_set:nn { \_math } {#2}
932   \cs_gset_eq:cc { \_math_env_ #1 _begin: } {#1}
933   \cs_gset_eq:cc { \_math_env_ #1 _end: } { end #1 }
934   %
935   \ExpandArgs { nne } \RenewDocumentEnvironment {#1} { b }
936   {
937     \exp_not:N \bool_if:NTF \exp_not:N \l\_math_collected_bool
938     {

```

```

939 %           \typeout{===>B1}
940         }
941       {
942 %           \typeout{===>B2}
943       \cs_if_exist:cTF { __math_env #1 _aux: }
944         {
945           \exp_not:c { __math_env #1 _aux: }
946             ##1 \exp_not:N \__math_env_end: {#1}
947         }
948         { \exp_not:N \__math_process:nn {#1} {##1} }
949       \exp_not:n { \@kernel@math@registered@begin }
950       \bool_set_true:N \exp_not:N \l__math_collected_bool
951     }
952 %       \exp_not:N \tracingall
953     \exp_not:c { __math_env_ #1 _begin: }
954     ##1
955     \exp_not:c { __math_env_ #1 _end: }
956 %     \exp_not:N \tracingnone
957   }
958   {
959   }
960 }
961
962 \cs_new_protected:Npn \math_register_halign_env:nn #1#2
963 {
964   \tl_set:Nn \l__math_env_name_tl {#1}
965   \keys_set:nn { __math } {#2}
966   \cs_gset_eq:cc { __math_env_ #1 _begin: } {#1}
967   \cs_gset_eq:cc { __math_env_ #1 _end: } { end #1 }
968 %
969   \ExpandArgs { nnee } \RenewDocumentEnvironment {#1} { b }
970   {
971     \exp_not:N \bool_if:NTF \exp_not:N \l__math_collected_bool
972     {
973 %       \typeout{===>B1}
974     }
975     {
976 %       \typeout{===>B2}
977     \cs_if_exist:cTF { __math_env #1 _aux: }
978       {
979         \exp_not:c { __math_env #1 _aux: }
980           ##1 \exp_not:N \__math_env_end: {#1}
981       }
982       { \exp_not:N \__math_process:nn {#1} {##1} }
983     \exp_not:n { \@kernel@math@registered@begin }
984     \bool_set_true:N \exp_not:N \l__math_collected_bool
985   }
986 %     \exp_not:N \tracingall
987   \exp_not:c { __math_env_ #1 _begin: }
988   ##1
989 %     \exp_not:N \tracingnone
990   }
991   {
992     \exp_not:c { __math_env_ #1 _end: }

```

```

993     }
994 }

```

TODO: the following command is neither documented nor used. Is it needed?

```

995 \cs_new_protected:Npn \math_register_odd_env:nn #1#2
996 {
997   \tl_set:Nn \l__math_env_name_tl {#1}
998   \keys_set:nn { __math } {#2}
999   \cs_gset_eq:cc { __math_env_ #1 _begin: } {#1}
1000  \cs_gset_eq:cc { __math_env_ #1 _end: } { end #1 }
1001 %
1002  \ExpandArgs { nnee } \RenewDocumentEnvironment {#1} { b }
1003  {
1004    \exp_not:N \bool_if:NTF \exp_not:N \l__math_collected_bool
1005    {
1006      % \typeout{===>B1}
1007    }
1008    {
1009      % \typeout{===>B2}
1010      \cs_if_exist:cTF { __math_env_ #1 _aux: }
1011      {
1012        \exp_not:c { __math_env_ #1 _aux: }
1013        ##1 \exp_not:N \__math_env_end: {#1}
1014      }
1015      { \exp_not:N \__math_process:nn {#1} {##1} }
1016      \exp_not:n { \@kernel@math@registered@begin }
1017      \bool_set_true:N \exp_not:N \l__math_collected_bool
1018    }
1019    % \exp_not:N \tracingall
1020    \exp_not:c { __math_env_ #1 _begin: }
1021    ##1
1022  }
1023  {
1024    \exp_not:c { __math_env_ #1 _end: }
1025    % needed if we don't have $$...$$
1026    \exp_not:n { \typeout{---> @kernel@math@registered@end } }
1027    \exp_not:n { \@kernel@math@registered@end }
1028  }
1029 }
1030
1031
1032 % FMi: compare with block change!
1033 %
1034 % \DeclareRobustCommand*\begin[1]{%
1035 % \UseHook{env/#1/before}%
1036 % \ifundefined{#1}%
1037 %   {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
1038 %   {\def\reserved@a{\def\@currenvir{#1}%
1039 %     \edef\@currenvline{\on@line}%
1040 %     \@execute@begin@hook{#1}%
1041 %     \csname #1\endcsname}}%
1042 % \@ignorefalse
1043 % \begin@group
1044 % \end@pefalse % tmp!!! is it ok to drop this here?

```

```

1045 % \reserved@a}
1046
1047
1048 \cs_new:Npn \@kernel@math@registered@begin {
1049 % \ShowTagging{struct-stack}
1050 %\typeout{==>A1}\ShowTagging{struct-stack,mc-current}
1051 \mode_if_vertical:TF
1052 {
1053 % \legacy_if:nTF { @endpe }
1054 % { \legacy_if_set_false:n { @endpe } }
1055 % { \__block_list_beginpar_vmode: }
1056 %
1057 % \typeout{==>~ at:~ \g__tag_struct_tag_tl}
1058 %
1059 \tag_if_active:T
1060 {
1061 \exp_args:Noo\str_if_eq:nnF \g__tag_struct_tag_tl { \l__tag_para_main_tag_tl }
1062 {
1063 % \typeout{==>A2}
1064 \__block_beginpar_vmode:
1065 } % needs correction!
1066 }
1067 }
1068 {
1069 % \typeout{==>A3}
1070 \__tag_tool_close_P:
1071 }
1072 \tag_socket_use:nn{math/display/formula/begin}{}{}
1073 % \typeout{==>MC1}\ShowTagging{mc-current}
1074 }
1075
1076 \cs_new:Npn \@kernel@math@registered@end {
1077 % \typeout{==>MC2}\ShowTagging{mc-current}
1078 \para_raw_end:
1079 \tagpdfparaOn
1080 \tag_socket_use:n{tagsupport/math/display/formula/end}
1081 % \typeout{==>MC3}\ShowTagging{mc-current}
1082 \@endpetrue
1083 }
1084
1085 \cs_new_protected:Npn \math_register_env:n #1
1086 { \math_register_env:nn {#1} { } }
1087
1088 \NewDocumentCommand \RegisterMathEnvironment { 0{} m }
1089 { \math_register_env:nn {#2} {#1} }

```

(End of definition for `\math_register_env:nn`, `\math_register_env:n`, and `\RegisterMathEnvironment`.
These functions are documented on page 3.)

`__math_env_forward:w`

```

1090 \cs_new_protected:Npn \__math_env_forward:w #1 \__math_env_end: #2
1091 { \__math_process:nn {#2} {#1} }

```

(End of definition for `__math_env_forward:w`.)

5.13 Document commands

Add one more here: `displaymath`, which is equivalent to `\[, \]` and hence to the basic `equation*`.
Added in more recent branch.

```

\equation
\_math_equation_begin:
\equation*
\_math_equation_star_begin:
\endequation
\_math_equation_end:
\endequation*
\_math_equation_star_end:

```

These environments are not set up by `amsmath` to collect their body, so we do that here. This has to be done *after* we can be sure `amsmath` is loaded.

Note that with `amsmath` loaded, `equation*` and `equation` are the two basics: they are used to define the other single-row display environments, etc.

```

1092 \tl_gput_right:Nn \@kernel@before@begindocument
1093 {
1094   \math_register_env:n { equation }
1095   \math_register_env:n { equation* }
1096   % at the moment register_env can only do display math
1097   % \math_register_env:n { math }
1098   \RenewDocumentEnvironment{math} {b}{\${#1$}}{}
1099   % and this one doesn't work either
1100   % \math_register_env:n { displaymath }
1101   \RenewDocumentEnvironment{displaymath} {b}{\[#1\]}{}
1102 }

```

(End of definition for `\equation` and others. These functions are documented on page ??.)

`\(` If math mode has not been collected, we need to do that; otherwise, worry about whether `\)` we are in math mode or not. The closing command here can only occur inside a collected math block: otherwise it will be simply used as a delimiter.

```

1103 \cs_gset_protected:Npn \( % \)
1104 {
1105   \bool_if:NTF \l_math_collected_bool
1106   {
1107     \mode_if_math:TF
1108     { \badmath }
1109     { $ }
1110   }
1111   {
1112     \__math_grab_inline:w
1113   }
1114 } % \(
1115 \cs_gset_protected:Npn \)
1116 {
1117   \mode_if_math:TF
1118   { $ }
1119   { \badmath }
1120 }

```

(End of definition for `\(` and `\)`. These functions are documented on page ??.)

`\[` Again, we need to watch for when `amsmath` is loaded after this code. The flag usage here `\]` is to cover the case where `\[/\]` is hidden inside another environment. In this case the grabbing happens on the outer level and should not be repeated.

```

1121 \tl_gput_right:Nn \@kernel@before@begindocument

```

```

1122 {
1123   \cs_gset_protected:Npn \[ % \]
1124   {
1125     \__math_grab_eqn:w
1126 %     \bool_if:NTF \l__math_collected_bool
1127 %     { \begin { equation* } }
1128 %     { \__math_grab_eqn:w }
1129   } % \[
1130   \cs_gset_protected:Npn \]
1131   {
1132     \@badmath
1133 %     \bool_if:NTF \l__math_collected_bool
1134 %     { \end{ equation* } }
1135 %     { \@badmath }
1136   }
1137 }

```

(End of definition for \[and \]. These functions are documented on page ??.)

why does ensuremath need handling at all?

Indeed! Currently, this is setup to process the math that it has anyways already captured as its argument; thus it is more efficient than leaving the capture to be repeated by the \everymath

A bit of nesting fun to make sure we collect only if required.

```

1138 %\cs_gset_protected:Npn \ensuremath #1
1139 % {
1140 %   \mode_if_math:TF
1141 %   {#1}
1142 %   {
1143 %     \bool_if:NTF \l__math_collected_bool
1144 %     { \@ensuredmath {#1} }
1145 %     {
1146 %       \bool_set_true:N \l__math_collected_bool
1147 %       \__math_process:nn { math } {#1}
1148 %       \@ensuredmath {#1}
1149 %       \bool_set_false:N \l__math_collected_bool
1150 %     }
1151 %   }
1152 % }

```

(End of definition for \ensuremath. This function is documented on page ??.)

5.14 \everymath and \everydisplay

The business end for grabbing inline math and “raw” TeX display. Most display math mode is actually handled elsewhere, as we have macro control.

```

1153
1154 \exp_args:No \tex_everymath:D
1155 {
1156   \tex_the:D \tex_everymath:D
1157   \bool_if:NF \l__math_collected_bool
1158   {
1159     \bool_set_true:N \l__math_collected_bool
1160     \__math_grab_dollar:w
1161   }
1162 }
1163
1164 \exp_args:No \tex_everydisplay:D

```



```

1165 {
1166   \tex_the:D \tex_everydisplay:D
1167   \iftrue % this may have to be a settable flag!
1168 %       \typeout{==>~ in~ everydisplay}

```

flipping the `\belowdisplay` values is done so that we get (assumption) a negative skip and not make the page bigger then we take that out, then we add the tagging code (in `__math_tag_dollardollar_display_end`) and then we put a real `\postdisplaypenalty` in and the right skip (of which we don't know if it is short or a normal `\belowdisplayskip`). This might need some refinement if that skip is actually negative from the start (not sure it ever is and is worth bothering about)

```

1169     \skip_set:Nn \belowdisplayskip      {-\belowdisplayskip}
1170     \skip_set:Nn \belowdisplayshortskip {-\belowdisplayshortskip}
1171     \int_set:Nn \postdisplaypenalty {10000}
1172     \group_insert_after:N \__math_tag_dollardollar_display_end:
1173   \fi
1174   \bool_if:NF \l__math_collected_bool
1175   {
1176     \bool_set_true:N \l__math_collected_bool
1177     \__math_grab_dollardollar:w
1178   }
1179 }

```

5.15 Modifying kernel environments

We need to cover this even though it is, of course, not encouraged.

```

1180 \math_register_env:n { eqnarray }
1181 \math_register_env:n { eqnarray* }

```

Tabulars currently contain a `$` that shouldn't trigger math tagging.

```

1182 \RequirePackage{array}
1183 \tl_if_in:NnT\@tabular{$}
1184 {
1185   \def\@tabular{%
1186     \leavevmode
1187     \UseTaggingSocket{tbl/hmode/begin}%
1188     \hbox \bgroup
1189     \bool_set_true:N \l__math_collected_bool
1190     $
1191     \bool_set_false:N \l__math_collected_bool
1192     \col@sep\tabcolsep \let\d@llarbegin\begin\group
1193     \let\d@llarend\end\group

```

A proper switching mechanism is needed: for the present, do directly.

```

1194   \cs_set_protected:Npn \__math_grab_dollar:w { \__math_grab_dollar_loop: }
1195   \@tabarray}
1196 }

```

`__math_m@th:` Handle non-math use of math mode. At present nesting isn't supported as `\m@th` pops `\m@th` up in a few places that *are* math mode!

```

1197 \cs_new_eq:NN \__math_m@th: \m@th
1198 \cs_gset_protected:Npn \m@th
1199 {
1200   \bool_set_true:N \l__math_collected_bool

```

```

1201     \_math_m@th:
1202   }

```

(End of definition for `_math_m@th:` and `\m@th`. This function is documented on page ??.)

5.16 Disable math grabbing in the begindocument hook

For example `amsart` uses `math` to measure text there.

```

1203 \tl_gput_right:Nn\@kernel@before@begindocument
1204 {
1205   \bool_set_true:N\l__math_collected_bool
1206 }
1207 \tl_gput_right:Nn\@kernel@after@begindocument
1208 {
1209   \bool_set_false:N\l__math_collected_bool
1210 }

```

5.17 Modifying `amsmath`

Mark up all of the display environments as the content is captured anyway. We then use an internal macro in each environment type to insert the processing code. Each of these is slightly different, so we cannot use a simple loop here. The test for `\split@tag` is required as the `split` environment internally uses `gather` *when not within an `amsmath` environment*, for example inside `equation`. Without the precaution, we'd get two copies of the grabbed math, the second of which would start with `\split@tag`.

```

\_math_amsmath_align@:nn
\_math_amsmath_gather@:n
\_math_amsmath_multline@:n
  \align@
  \gather@
  \multline@

```

```

1211
1212
1213
1214 \tl_gput_right:Nn \@kernel@before@begindocument {
1215 %
1216 \renewenvironment{gather*}{%
1217   \start@gather\st@rredtrue
1218 }
1219 {%
1220 % this redirection doesn't work if we alter "gather"!
1221 % \endgather
1222 % so replace it with its real meaning
1223 \math@cr \black@\totwidth@ \egroup
1224 $$\ignorespacesafterend
1225 }
1226 \def\common@align@ending {
1227   \math@cr \black@\totwidth@
1228   \egroup
1229   \ifingather@
1230     \restorealignstate@
1231     \egroup
1232     \nonumber
1233     \ifnum0='{ \fi\iffalse}\fi
1234   \else
1235     $$$
1236   \fi
1237   \ignorespacesafterend

```

```

1238 }
1239 \renewenvironment{alignat}{%
1240   \start@align@z@\st@rredfalse
1241 }{%
1242   \common@align@ending
1243 }
1244 \renewenvironment{alignat*}{%
1245   \start@align@z@\st@rredtrue
1246 }{%
1247   \common@align@ending
1248 }
1249 \renewenvironment{xalignat}{%
1250   \start@align@ne\st@rredfalse
1251 }{%
1252   \common@align@ending
1253 }
1254 \renewenvironment{xalignat*}{%
1255   \start@align@ne\st@rredtrue
1256 }{%
1257   \common@align@ending
1258 }
1259 \renewenvironment{xxalignat}{%
1260   \start@align@tw@\st@rredtrue
1261 }{%
1262   \common@align@ending
1263 }
1264 \renewenvironment{align}{%
1265   \start@align@ne\st@rredfalse\m@ne
1266 }{%
1267   \common@align@ending
1268 }
1269 \renewenvironment{align*}{%
1270   \start@align@ne\st@rredtrue\m@ne
1271 }{%
1272   \common@align@ending
1273 }
1274 \renewenvironment{flalign}{%
1275   \start@align@tw@\st@rredfalse\m@ne
1276 }{%
1277   \common@align@ending
1278 }
1279 \renewenvironment{flalign*}{%
1280   \start@align@tw@\st@rredtrue\m@ne
1281 }{%
1282   \common@align@ending
1283 }
1284 %
1285 \renewenvironment{multline*}{\start@multline\st@rredtrue}
1286 {%
1287   \iftagsleft@ \exp\lendmultline@ \else \exp\rendmultline@ \fi
1288   \ignorespacesafterend
1289 }

```

Also for false?

```

1290 \def\measuring@true{\let\ifmeasuring@\iftrue\tag_suspend:n{\measuring}}
1291 %
1292 \math_register_halign_env:nn {align}{}
1293 \math_register_halign_env:nn {align*}{}
1294 \math_register_halign_env:nn {alignat}{}
1295 \math_register_halign_env:nn {alignat*}{}
1296 \math_register_halign_env:nn {flalign}{}
1297 \math_register_halign_env:nn {flalign*}{}
1298 \math_register_halign_env:nn {gather}{}
1299 \math_register_halign_env:nn {gather*}{}
1300 \math_register_halign_env:nn {multline}{}
1301 \math_register_halign_env:nn {multline*}{}
1302 \math_register_halign_env:nn {xalignat}{}
1303 \math_register_halign_env:nn {xalignat*}{}
1304 \math_register_halign_env:nn {xxalignat}{}
1305 %
1306 \@namedef{maketag @ @ @} #1{%
1307 % \typeout{--->maketag @ @ @}
1308 \ifmeasuring@
1309 \hbox{\m@th\normalfont#1}%
1310 \else
1311 \tagmccend \tagstructbegin{tag=Lbl}%
1312 \tagmccbegin{tag=Lbl}%
1313 \hbox{\m@th\normalfont#1}%
1314 \tagmccend \tagstructend \tagmccbegin{}%
1315 \fi
1316 }
1317 \@namedef{math@cr @ @ @ gather}{%
1318 \ifst@rred\nonumber\fi
1319 &\relax
1320 \make@display@tag
1321 %
1322 \maybestartnewformulatag
1323 %
1324 \ifst@rred\else\global\@eqnswtrue\fi
1325 \global\advance\row@\@ne
1326 \cr
1327 }
1328 \@namedef{math@cr @ @ @ align}{%
1329 \ifst@rred\nonumber\fi
1330 \if@eqnsw \global\tag@true \fi
1331 \global\advance\row@\@ne
1332 \add@amps\maxfields@
1333 \omit
1334 \kern-\alignsep@
1335 \iftag@
1336 \setboxz@h{\@lign\strut@{\make@display@tag}}%
1337 \place@tag
1338 \fi
1339 %
1340 \maybestartnewformulatag
1341 %
1342 \ifst@rred\else\global\@eqnswtrue\fi

```

```

1343 \global\lineht@z@
1344 \cr
1345 }
1346 \def\restore@math@cr{\@namedef{math@cr @ @ @}{
1347 %
1348 \maybestartnewformulatag
1349 %
1350 \cr}}
1351 \restore@math@cr
1352 }

```

(End of definition for `_math_amsmath_align@:nn` and others. These functions are documented on page ??.)

`_math_split_at_nl:NN` This splits grabbed math at newlines.

```

1353 \cs_new:Npn \_math_split_at_nl:NN #1#2 {
1354 \tl_set:Nf \l_math_tmpa_tl {
1355 \exp_after:wN \_math_split_at_nl_first:w #1 \ \ \q_nil \ \ \s_stop }
1356 \exp_after:wN \_math_split_at_nl_aux:nnNN \l_math_tmpa_tl #1 #2
1357 }

```

and the auxiliary commands

```

1358 \cs_new:Npn \_math_split_at_nl_first:w #1 \ \ #2 \ \ #3 \s_stop
1359 {
1360 \quark_if_nil:nTF {#2}
1361 { {#1} { } }
1362 {
1363 \_math_split_chk_if_begin:ww #1 \begin \q_nil \s_mark
1364 #2 \ \ #3 \s_stop
1365 }
1366 }

```

```

1367
1368 \cs_new_protected:Npn \_math_split_at_nl_aux:nnNN #1 #2 #3 #4
1369 {
1370 \tl_gset:Nn #4 {#1}
1371 \tl_gset:Nn #3 {#2}
1372 }

```

```

1373
1374 \cs_new:Npn \_math_split_chk_if_begin:ww
1375 #1 \begin #2 #3 \s_mark #4 \ \ \q_nil \ \ \s_stop
1376 {
1377 \quark_if_nil:nTF {#2}
1378 { {#1} {#4} }
1379 {
1380 \exp_after:wN \_math_split_collect_one_end:w
1381 \_math_split_cleanup_begin_q_nil:w #1 \begin{#2} #3 \ \ #4 \s_stop
1382 { } { 1 }
1383 }
1384 }

```

```

1385 \cs_new:Npn \_math_split_cleanup_begin_q_nil:w #1 \begin \q_nil {#1}

```

```

1386
1387 \cs_new:Npn \_math_split_collect_one_end:w #1 \end #2 #3 \s_stop #4 #5
1388 {

```

```

1390     \exp_args:Nf \_math_split_check_count_begins:nmmm
1391     { \_math_split_count_begins:n { #4 #1 } } {#5}
1392     { #4 #1 \end{#2} } {#3}
1393   }
1394 \cs_new:Npn \_math_split_count_begins:n #1
1395   { \int_eval:n { 0 \_math_split_count_begins:w #1 \begin \q_nil } }
1396
1397 \cs_new:Npn \_math_split_count_begins:w #1 \begin #2
1398   { \quark_if_nil:nF {#2} { +1 \_math_split_count_begins:w } }
1399
1400 \cs_new:Npn \_math_split_check_count_begins:nmmm #1 #2 #3 #4
1401   {
1402     \int_compare:nNnTF {#1} = {#2}
1403     {
1404       \exp_last_unbraced:Nf \_math_split_final_cleanup:nn
1405       { \_math_split:n { \_math_split_guard:n {#3} #4 } }
1406     }
1407     {
1408       \exp_args:No \use_ii_i:nn
1409       { \exp_after:wN { \int_value:w \int_eval:n { #2 + 1 } } }
1410       { \_math_split_collect_one_end:w #4 \s_stop {#3} }
1411     }
1412   }
1413 \cs_new:Npn \_math_split_final_cleanup:nn #1 #2
1414   {
1415     \exp:w \_math_split_final_cleanup:w #1
1416     \_math_split_guard:n \q_nil \s_mark { }
1417     {#2}
1418   }
1419 \cs_new:Npn \_math_split_final_cleanup:w #1 \_math_split_guard:n #2 #3 \s_mark #4
1420   {
1421     \quark_if_nil:nTF {#2}
1422     { \exp_end: { #4 #1 } }
1423     { \_math_split_final_cleanup:w #3 \s_mark { #4 #1 #2 } }
1424   }
1425
1426 \cs_new:Npn \_math_split:n #1 {
1427   \_math_split_at_nl_first:w #1 \\ \q_nil \\ \s_stop }
1428
1429 % this looks unused.
1430 %\NewDocumentCommand \splitnl { mm +m }
1431 % {
1432 %   \tl_set:Nf \l__math_tmpa_tl { \split:n {#3} }
1433 %   \show \l__math_tmpa_tl
1434 %   \exp_after:wN \_splitnl_aux:nnNN \l__math_tmpa_tl #1 #2
1435 % }

```

(End of definition for _math_split_at_nl:NN.)

\maybestartnewformulatag

```

1436
1437 \newif\if@subformulas
1438 \tl_new:N \result
1439

```

```

1440 \cs_new_protected:Npn\grabaformulapartandstart {
1441   \__math_split_at_nl:NN \g__math_grabbed_math_tl \result
1442   \typeout{====>first-result=\meaning\result}
1443   \typeout{====>first-tmpmathcontent=\meaning\g__math_grabbed_math_tl}
1444   \tl_if_empty:NTF \g__math_grabbed_math_tl
1445     {
1446       \typeout{====>formula~ has~ no~ subparts}
1447       \global\@subformulasfalse
1448     }
1449     {
1450       \typeout{====>formula~ has~ subparts}
1451       \global\@subformulastrue
1452       \edef\resulttitle{\g__math_grabbed_env_tl\space (part)}
1453       \tagstructbegin{tag=Formula,

```

For now we don't put real content in /alt or /ActualText on subformulas but we add a short text to satisfy the pdf/ua-2 validator

```

1454   %       alt=\result,
1455           alt = subformula,
1456           title-o=\resulttitle
1457       }
1458   }
1459   \tagmcbegin{}
1460 }
1461
1462 \cs_new_protected:Npn\grabaformulapartandmayberestart {
1463   \__math_split_at_nl:NN \g__math_grabbed_math_tl \result
1464   \typeout{====>result=\meaning\result}
1465   \typeout{====>tmpmathcontent=\meaning\g__math_grabbed_math_tl}
1466   % \tl_if_empty:NTF \g__math_grabbed_math_tl
1467   % {
1468   %   \typeout{====>tmpmathcontent=empty}
1469   % }
1470   % {
1471   %   \typeout{====>tmpmathcontent=not-empty}
1472   %   \edef\resulttitle{\g__math_grabbed_env_tl\space (part)}
1473   %   \tagstructbegin{tag=Formula,
1474           alt=\result,
1475           title-o=\resulttitle
1476   %   }
1477   % }
1478   \tagmcbegin{}
1479 }

```

(End of definition for \maybestartnewformulatag. This function is documented on page ??.)

```

1480 \def\maybestartnewformulatag {
1481   \if@subformulas
1482   \ifmeasuring@\else
1483   %
1484   \tl_if_empty:NF \g__math_grabbed_math_tl
1485     {
1486       \tagmcbegin{}
1487       \tagstructbegin{}
1488       \grabaformulapartandmayberestart

```

```

1489     }
1490 \fi
1491 \fi
1492 }

The breqn packages changes catcodes and that isn't yet covered by our mechanism.

1493 %\AddToHook{package/breqn/after}{
1494 % \typeout{===>~ in~ hook}
1495 % \math_register_halign_env:nn {dmath}{}
1496 % \math_register_halign_env:nn {dgroup*}{}
1497 %}

1498 \ExplSyntaxOff

1499 <@@=)

1500 </kernel>

```

Index

The italic numbers denote the pages where the corresponding entry is described, numbers underlined point to the definition, all others indicate the places where it is used.

Symbols	B
<code>\#</code> 72, 418	<code>\begin</code> 34, 567, 585, 839, 879, 882, 1034, 1127, 1363, 1375, 1381, 1386, 1395, 1397
<code>\%</code> 73	<code>\beginngroup</code> 69, 162, 177, 193, 1043, 1192
<code>\(</code> 826	<code>\belowdisplay</code> 41
<code>\(</code> <u>1103</u>	<code>\belowdisplayshortskip</code> 32, 1170
<code>\)</code> 827	<code>\belowdisplayskip</code> 32, 41, 1169
<code>\)</code> <u>1103</u>	<code>\bgroup</code> 1188
<code>@@</code> commands:	block internal commands:
<code>\l_@@_collected_bool</code> 3, 9	<code>_block_beginpar_vmode:</code> 1064
<code>\l_@@_content_template_tl</code> 7	<code>_block_list_beginpar_vmode:</code> 1055
<code>\l_@@_fakemath_bool</code> 10	bool commands:
<code>\[</code> 835, <u>1101</u>	<code>\bool_gset_false:N</code> 363
<code>\[</code> 11, 32, 39, <u>1121</u>	<code>\bool_gset_true:N</code> 284, 322, 359, 385, 426
<code>\]</code> 272, 273, 274, 275, 276, 277, 327, 873, 877, 910, 1355, 1358, 1364, 1375, 1381, 1427	<code>\bool_if:NTF</code> 62, 282, 433, 493, 628, 638, 675, 680, 690, 697, 705, 838, 937, 971, 1004, 1105, 1126, 1133, 1143, 1157, 1174
<code>\{</code> 70	<code>\bool_lazy_any:nTF</code> 781
<code>\}</code> 71	<code>\bool_new:N</code> 10, 11, 37, 38, 39, 40, 41, 42, 321
<code>\]</code> 836, 1101	<code>\bool_set_false:N</code> 833, 1149, 1191, 1209
<code>\]</code> 11, 32, 39, <u>1121</u>	<code>\bool_set_true:N</code> 497, 763, 950, 984, 1017, 1146, 1159, 1176, 1189, 1200, 1205
A	bool internal commands:
<code>actual+source</code> (plug) <u>591</u>	<code>\l_math_collected_bool</code> 11, 10, 833, 838, 937, 950, 971,
<code>\AddToHook</code> 7, 9, 138, 226, 233, 239, 251, 309, 399, 440, 457, 483, 1493	
<code>\advance</code> 1325, 1331	
<code>\aftergroup</code> 32	
<code>alt+source</code> (plug) <u>606</u>	
<code>\AssignSocketPlug</code> 285, 286, 287, 288, 292, 397, 398, 436, 437, 438, 439	

984, 1004, 1017, 1105, 1126, 1133, 1143, 1146, 1149, 1157, 1159, 1174, 1176, 1189, 1191, 1200, 1205, 1209	
<code>\l_math_fakemath_bool</code> 11, 11, 763, 784	
<code>\g_math_luamml_write_bool</code>	
. 19, 282, 321, 322, 359, 363, 367	
box commands:	
<code>\box_new:N</code> 409	
box internal commands:	
<code>\l_math_tmpa_box</code> 409, 412	
<code>\boxed</code> 9	
C	
char commands:	
<code>\char_set_catcode_other:N</code>	
. 70, 71, 72, 73, 418	
clist commands:	
<code>\clist_map_inline:Nn</code> 420	
<code>\clist_new:N</code> 50	
<code>\clist_put_right:Nn</code> 51	
<code>\cr</code> 904, 1326, 1344, 1350	
cs commands:	
<code>\cs_generate_variant:Nn</code> 470, 657, 767	
<code>\cs_gset_eq:NN</code>	
. 932, 933, 966, 967, 999, 1000	
<code>\cs_gset_protected:Npn</code>	
. 1103, 1115, 1123, 1130, 1138, 1198	
<code>\cs_if_eq:NNTF</code> 162, 177, 193	
<code>\cs_if_exist:NTF</code> 943, 977, 1010	
<code>\cs_if_exist_use:N</code> 707	
<code>\cs_if_exist_use:NTF</code> 867	
<code>\cs_if_free:NTF</code> 119, 124, 129	
<code>\cs_new:Npn</code> 296, 323, 324, 1048, 1076, 1353, 1358, 1374, 1386, 1388, 1394, 1397, 1400, 1413, 1419, 1426	
<code>\cs_new_eq:NN</code> 1197	
<code>\cs_new_protected:Npn</code>	
. 22, 53, 68, 76, 116, 136, 280, 410, 461, 758, 768, 774, 775, 777, 778, 799, 801, 812, 826, 835, 843, 849, 858, 860, 865, 871, 873, 879, 884, 898, 901, 912, 928, 962, 995, 1085, 1090, 1368, 1440, 1462	
<code>\cs_set:Npn</code> 140, 152, 345, 346	
<code>\cs_set_eq:NN</code>	
. 121, 126, 131, 350, 351, 419	
<code>\cs_set_protected:Npn</code> 776, 904, 1194	
<code>\csname</code> 1041	
D	
<code>\DeclareDocumentCommand</code> 922	
<code>\DeclareMathEnvironment</code> 10	
<code>\DeclareRobustCommand</code> 1034	
<code>\def</code> 1037, 1038, 1185, 1226, 1290, 1346, 1480	
default (plug) 512, 535, 544, 624, 722	
<code>\displaylines</code> 10	
E	
<code>\edef</code> 1039, 1452, 1472	
<code>\egroup</code> 144, 147, 1223, 1228, 1231	
<code>\else</code> 1234, 1287, 1310, 1324, 1342, 1482	
<code>\end</code> 34, 571, 587, 823, 839, 884, 889, 893, 1134, 1388, 1392	
end internal commands:	
<code>_math_tag_dollardollar_-</code> <code>display_end</code> 41	
<code>\endarray</code> 140	
<code>\endcsname</code> 1041	
<code>\endequation</code> 1092	
<code>\endequation*</code> 1092	
<code>\endgather</code> 1221	
<code>\endgroup</code> 77, 1193	
<code>\ensuremath</code> 10, 1138	
<code>\equalign</code> 10	
<code>\equation</code> 1092	
<code>\equation*</code> 1092	
<code>\everydisplay</code> 40	
<code>\everymath</code> 3, 40	
exp commands:	
<code>\exp:w</code> 1415	
<code>\exp_after:wN</code>	
. 1355, 1356, 1380, 1409, 1434	
<code>\exp_args:Nf</code> 1390	
<code>\exp_args:NNV</code> 914	
<code>\exp_args:No</code> 1154, 1164, 1408	
<code>\exp_args:Noo</code> 1061	
<code>\exp_end:</code> 1422	
<code>\exp_last_unbraced:Ne</code> 488	
<code>\exp_last_unbraced:Nf</code> 1404	
<code>\exp_not:N</code> 567, 571, 585, 587, 937, 945, 946, 948, 950, 952, 953, 955, 956, 971, 979, 980, 982, 984, 986, 987, 989, 992, 1004, 1012, 1013, 1015, 1017, 1019, 1020, 1024	
<code>\exp_not:n</code> 569, 581, 586, 949, 983, 1016, 1026, 1027	
<code>\ExpandArgs</code> 922, 935, 969, 1002	
<code>\ExplSyntaxOff</code> 1498	
<code>\ExplSyntaxOn</code> 8	
F	
<code>\fi</code> 220, 733, 1173, 1233, 1236, 1287, 1315, 1318, 1324, 1329, 1330, 1338, 1342, 1490, 1491	
fi commands:	
<code>\fi:</code> 903, 909	
file commands:	
<code>\file_if_exist:nTF</code> 422	
<code>\file_input:n</code> 425	

G	
<code>\gdef</code>	150
<code>\GetDocumentProperties</code>	489
<code>\global</code>	1324, 1325, 1330, 1331, 1342, 1343, 1447, 1451
<code>\grabaformulapartandmayberestart</code> 1462, 1488
<code>\grabaformulapartandstart</code>	725, 1440
group commands:	
<code>\group_begin:</code>	845
<code>\c_group_begin_token</code>	853
<code>\group_end:</code>	914
<code>\group_insert_after:N</code>	1172
H	
<code>\hbox</code>	1188, 1309, 1313
hbox commands:	
<code>\hbox_set:Nn</code>	412
<code>\hfil</code>	159, 173, 188, 190
<code>\hskip</code>	160, 175, 191
I	
if commands:	
<code>\if_false:</code>	903, 909
<code>\ifcase</code>	158
<code>\iffalse</code>	1233
<code>\ifnum</code>	1233
<code>\iftrue</code>	1167, 1290
<code>\ignorespaces</code>	896
<code>\ignorespacesafterend</code> ..	1224, 1237, 1288
int commands:	
<code>\int_compare:nNnTF</code> ..	467, 875, 886, 1402
<code>\int_decr:N</code>	892
<code>\int_eval:n</code>	1395, 1409
<code>\int_gincr:N</code>	56, 59, 664, 679, 682
<code>\int_incr:N</code>	881
<code>\int_new:N</code> 45, 46, 47, 48, 49, 458, 459, 842
<code>\int_set:Nn</code>	289, 465, 1171
<code>\int_use:N</code> 108, 444, 446, 448, 450, 452, 460
<code>\int_value:w</code>	1409
int internal commands:	
<code>\g_math_AF_attached_int</code>	13
<code>\g_math_AF_found_int</code>	13
<code>\g_math_AF_total_int</code>	13
<code>\l_math_grab_env_int</code> 33, 842, 875, 881, 886, 892
<code>\g_math_math_total_int</code> 13, 47, 108, 448, 664
<code>\g_math_mathchoice_int</code>	22, 458
<code>\g_math_mathml_AF_attached_int</code> 13, 49, 452, 682
<code>\g_math_mathml_AF_found_int</code> 13, 48, 450, 679
<code>\g_math_mathml_int</code> ..	13, 46, 59, 446
<code>\g_math_mathml_total_int</code> 13, 45, 56, 444
<code>\l_math_mathstyle_int</code>	458, 465
<code>\g_math_mml_int</code>	13
<code>\g_math_mml_total_int</code>	13
<code>\intertext</code>	10
iow commands:	
<code>\iow_close:N</code>	313, 403
<code>\iow_new:N</code>	293, 388
<code>\iow_newline:</code> 83, 85, 90, 92, 107, 109, 111, 113
<code>\iow_now:Nn</code> ..	295, 300, 311, 373, 393, 401
<code>\iow_open:Nn</code>	294, 389
<code>\iow_term:n</code> 442, 443, 444, 446, 448, 450, 452
iow internal commands:	
<code>\g_math_luamml_iow</code> 293, 294, 295, 300, 311, 313
<code>\g_math_writedummy_iow</code> 373, 386, 388, 389, 393, 401, 403
K	
<code>\kern</code>	1334
keys commands:	
<code>\keys_define:nn</code> ...	330, 381, 471, 918
<code>\keys_set:nn</code>	931, 965, 998
L	
<code>\lastskip</code>	817
<code>\LaTeXe</code>	9, 10
<code>\leavevmode</code>	7, 1186
legacy commands:	
<code>\legacy_if:nTF</code>	760, 1053
<code>\legacy_if_p:n</code>	783
<code>\legacy_if_set_false:n</code>	1054
<code>\let</code>	1192, 1193, 1290
<code>\ltxlabmathdate</code>	4
<code>\ltxlabmathversion</code>	5
luamml commands:	
<code>\luamml_flag_ignore:</code>	128, 131
<code>\luamml_flag_process:</code>	123, 126
<code>\luamml_flag_save:nn</code> ..	168, 183, 199
<code>\luamml_flag_structelem:</code> ...	118, 121
<code>\luamml_ignore:</code> ...	129, 131, 346, 354
<code>\luamml_process:</code>	124, 126
<code>\luamml_structelem:</code> ...	119, 121, 345
luamml internal commands:	
<code>__luamml_array_finalize_array:</code>	148
<code>__luamml_array_finalize_col:w</code> 170, 185, 201

<code>_luamml_array_init_col:</code>	<code>_math_grab_loop\textonly@unskip:</code>
..... 166, 181, 197 865
<code>_luamml_array_save_array:</code> ... 143	<code>_math_grab_loop\unskip:</code> 865
<code>\l_luamml_pretty_int</code>	<code>_math_grab_loop_end:</code> . 872, 912, 912
..... 289	<code>_math_grab_loop_group:n</code>
<code>_luamml_register_output_hook:N</code> 308 854, 858, 858
M	
math commands:	
<code>\math_processor:n</code>	<code>_math_grab_loop_newline:</code>
..... 3, 775, 775 876, 888, 901, 901
<code>\math_register_env:n</code> 3, 928, 1085,	<code>_math_grab_loop_store:n</code>
1094, 1095, 1097, 1100, 1180, 1181 858, 859, 860, 869, 877, 882, 893
<code>\math_register_env:nn</code>	<code>_math_grab_loop_token:N</code>
..... 3, 928, 928, 1086, 1089 855, 865, 865
<code>\math_register_halign_env:nn</code> ...	<code>_math_luamml_activate_write:</code> ..
..... 962, 1292, 1293, 1294, 235, 253, 280
1295, 1296, 1297, 1298, 1299, 1300,	<code>_math_luamml_ignore:</code>
1301, 1302, 1303, 1304, 1495, 1496 323, 324, 346, 351, 788
<code>\math_register_odd_env:nn</code>	<code>_math_luamml_output_hook:n</code> 296, 308
..... 995	<code>_math_luamml_structelem:</code>
math internal commands:	
<code>_math_AF_html_reader:w</code> 68, 419 323, 323, 345, 350, 516, 549
<code>_math_AF_html_reader_verb:w</code> 74, 76	<code>_math_m@th:</code>
<code>_math_AF_mml:nnnn</code> 1197, 1197, 1201
..... 53, 53, 78	<code>_math_process:nn</code> . 758, 758, 767,
<code>_math_AF_process_mathml_files:</code>	780, 806, 948, 982, 1015, 1091, 1147
..... 409, 410, 457	<code>_math_process_auxi:nn</code> 758, 764, 768
<code>_math_amsmath_align@:nn</code> 1211	<code>_math_process_auxii:nn</code>
<code>_math_amsmath_gather@:n</code> 1211 758, 772, 774, 776
<code>_math_amsmath_multline@:n</code> .. 1211	<code>_math_provide_luamml_commands:</code>
<code>_math_correct_luamml_array_-</code> 116, 259, 263
<code>patches:</code>	<code>_math_split:n</code>
..... 136, 232, 250 1405, 1426
<code>_math_env_end:</code> . 946, 980, 1013, 1090	<code>_math_split_at_nl:NN</code>
<code>_math_env_forward:w</code> 925, 1090, 1090 1353, 1353, 1441, 1463
<code>_math_equation_begin:</code>	<code>_math_split_at_nl_aux:nnNN</code> ...
..... 1092 1356, 1368
<code>_math_equation_end:</code>	<code>_math_split_at_nl_first:w</code>
..... 1092 1355, 1358, 1427
<code>_math_equation_star_begin:</code> . 1092	<code>_math_split_check_count_-</code>
<code>_math_equation_star_end:</code> ... 1092	<code>begins:nnnn</code>
<code>_math_grab_dollar:n</code> 1390, 1400
..... 777, 778, 800, 915	<code>_math_split_chk_if_begin:ww</code> ...
<code>_math_grab_dollar:w</code> 1363, 1374
..... 777, 777, 1160, 1194	<code>_math_split_cleanup_begin_q_-</code>
<code>_math_grab_dollar_delim:w</code>	<code>nil:w</code>
..... 777, 799, 799 1381, 1386
<code>_math_grab_dollar_loop:</code>	<code>_math_split_collect_one_end:w</code> .
..... 843, 843, 1194 1380, 1388, 1410
<code>_math_grab_dollardollar:w</code>	<code>_math_split_count_begins:n</code> ...
..... 801, 801, 1177 1391, 1394
<code>_math_grab_eqn:w</code> 835, 835, 1125, 1128	<code>_math_split_count_begins:w</code> ...
<code>_math_grab_inline:w</code> 826, 826, 1112 1395, 1397, 1398
<code>_math_grab_loop:</code>	<code>_math_split_final_cleanup:nn</code> ..
..... 843, 847, 849, 863, 899, 906 1404, 1413
<code>_math_grab_loop_\$:</code>	<code>_math_split_final_cleanup:w</code> ...
..... 865 1415, 1419, 1423
<code>_math_grab_loop\:</code>	<code>_math_split_guard:n</code> 1405, 1416, 1419
..... 865	<code>_math_tag_dollardollar_-</code>
<code>_math_grab_loop\begin:</code>	<code>display_end:</code>
..... 865 812, 1172
<code>_math_grab_loop\end:</code>	
..... 865	
<code>_math_grab_loop\ignorespaces:</code> 865	

<code>\socket_new:nn</code>	98, 370, 500, 501, 502, 503, 531, 532, 533, 534, 590, 622, 623, 720, 721, 749
<code>\socket_new_plug:nnn</code>	99, 371, 504, 508, 512, 524, 535, 539, 544, 556, 591, 606, 624, 650, 660, 714, 722, 726, 737, 745
<code>\socket_use:n</code>	528, 529, 560, 561, 695, 696
Sockets:	
<code>tagsupport/math/content</code>	590
<code>tagsupport/math/display/begin</code> ..	531
<code>tagsupport/math/display/end</code> ...	531
<code>tagsupport/math/display/formula/begin</code>	531
<code>tagsupport/math/display/formula/end</code>	531
<code>tagsupport/math/end</code>	749
<code>tagsupport/math/inline/begin</code> ..	500
<code>tagsupport/math/inline/end</code>	500
<code>tagsupport/math/inline/formula/begin</code>	500
<code>tagsupport/math/inline/formula/end</code>	500
<code>tagsupport/math/mathml/write</code> ..	370
<code>tagsupport/math/mathml/write/prepare</code>	98
<code>tagsupport/math/struct/begin</code> ..	622
<code>tagsupport/math/struct/end</code>	622
<code>tagsupport/math/substruct/begin</code> ..	720
<code>tagsupport/math/substruct/end</code> ..	720
<code>\space</code>	4, 5, 1452, 1472
split commands:	
<code>\split:n</code>	1432
<code>\splitnl</code>	1430
splitnl internal commands:	
<code>__splitnl_aux:nnNN</code>	1434
str commands:	
<code>\c_backslash_str</code>	108
<code>\str_case:nn</code>	228
<code>\str_if_eq:nnTF</code>	485, 1061
<code>\str_mdfive_hash:n</code>	657, 666
<code>\str_new:N</code>	16
<code>\str_replace_all:Nnn</code>	102, 103
<code>\str_set:Nn</code>	101
str internal commands:	
<code>\l_math_tpa_str</code>	11, 16, 101, 102, 103, 110
<code>\symletters</code>	291
<code>\symsymbols</code>	290
sys commands:	
<code>\sys_if_engine luatex:TF</code>	134
<code>\c_sys_jobname_str</code>	52, 294, 391
T	
<code>\tabcolsep</code>	1192
tag commands:	
<code>\tag_if_active:TF</code>	24, 1059
<code>\tag_mc_begin:n</code>	743
<code>\tag_mc_begin_pop:n</code>	511
<code>\tag_mc_end:</code>	26, 748
<code>\tag_mc_end_push:</code>	26, 507
<code>\tag_resume:n</code>	468
<code>\tag_socket_use:n</code>	517, 518, 520, 522, 550, 551, 552, 554, 792, 795, 796, 807, 818, 1080
<code>\tag_socket_use:nn</code>	808, 1072
<code>\tag_socket_use:nnn</code>	793
<code>\tag_struct_begin:n</code>	4, 640, 700
<code>\tag_struct_end:</code>	29, 653, 718
<code>\tag_suspend:n</code>	468, 1290
tag internal commands:	
<code>__tag_check_para_end_show:nn</code> ...	28
<code>__tag_gincr_para_end_int:</code>	27
<code>\l__tag_math_alt_bool</code>	13, 42, 474, 493, 497, 638, 675
<code>\g__tag_math_luamml_tl</code>	13, 43, 44
<code>\g__tag_math_mathml_AF_bool</code>	13, 40, 284, 385, 426, 433
<code>\l__tag_math_mathml_AF_bool</code>	13, 39, 478, 680, 690, 705
<code>\l__tag_math_mathml_files_clist</code> .	13, 50, 51, 420, 473
<code>\l__tag_math_mathml_pane_bool</code> ...	13, 41, 62, 475
<code>\l__tag_math_texsource_AF_bool</code> ..	13, 37, 480, 628, 697
<code>\l__tag_math_texsource_pane_bool</code>	13, 38, 477
<code>\l__tag_para_main_tag_tl</code>	1061
<code>\g__tag_struct_tag_tl</code> ...	1057, 1061
<code>__tag_tool_close_P:</code> ..	22, 22, 538, 1070
<code>\tagmcbegin</code>	1312, 1314, 1459, 1478
<code>\tagmchend</code>	730, 1311, 1314, 1486
<code>\tagpdfparaOff</code>	515, 548
<code>\tagpdfparaOn</code>	1079
<code>\tagpdfsetup</code>	33, 276, 278
<code>\tagstructbegin</code>	1311, 1453, 1473
<code>\tagstructend</code>	732, 1314, 1487
<code>tagsupport/math/content (socket)</code> ...	590
<code>tagsupport/math/display/begin (socket)</code>	531
<code>tagsupport/math/display/end (socket)</code>	531
<code>tagsupport/math/display/formula/begin</code> (socket)	531
<code>tagsupport/math/display/formula/end</code> (socket)	531
<code>tagsupport/math/end (socket)</code>	749

tagsupport/math/inline/begin (socket)	500	\@subformulastrue	1451
tagsupport/math/inline/end (socket)	500	\@tabarray	1195
tagsupport/math/inline/formula/begin (socket)	500	\@tabular	1183, 1185
tagsupport/math/inline/formula/end (socket)	500	\@tempcnta	155
tagsupport/math/mathml/write (socket)	370	\@xp	1287
tagsupport/math/mathml/write/prepare (socket)	98	\add@amps	1332
tagsupport/math/struct/begin (socket)	622	\align@	1211
tagsupport/math/struct/end (socket)	622	\alignsep@	1334
tagsupport/math/substruct/begin (socket)	720	\ar@align@mcCell	208
tagsupport/math/substruct/end (socket)	720	\ar@mcCellbox	205
tbl commands:		\black@	1223, 1227
\tbl_crcr:n	142	\col@sep	1192
\tbl_restore_outer_cell_data:	146	\common@align@ending	1226, 1242, 1247, 1252, 1257, 1262, 1267, 1272, 1277, 1282
T _E X and L ^A T _E X 2 _ε commands:		\count@	155
\@addtopreamble	157	\cr	33, 34
\@arrayright	149	\d@llarbegin	161, 162, 176, 177, 192, 193, 1192
\@badmath	1108, 1119, 1132, 1135	\d@llarend	164, 169, 179, 184, 195, 200, 1193
\@chnum	158	\do@row@strut	172, 187, 203, 209, 214, 219
\@classx	154	\gather@	1211
\@classz	152	\halign	34
\@currentvir	1038	\if@eqnsw	1330
\@currentvline	1039	\if@subformulas	731, 1437, 1481
\@doendpe	823	\ifingather@	1229
\@eHa	1037	\ifmeasuring@	3, 10, 1290, 1308, 1482
\@endpbox	207, 213, 218	\ifst@rred	1318, 1324, 1329, 1342
\@endpefalse	1044	\iftag@	1335
\@endpetrue	1082	\iftagsleft@	1287
\@ensuredmath	1144, 1148	\insert@column	163, 167, 178, 182, 194, 198
\@eqnswtrue	1324, 1342	\insert@pcolumn	206, 212, 217
\@execute@begin@hook	1040	\lendmultline@	1287
\@ifpackageloaded	247	\lineht@	1343
\@ifundefined	1036	\m@ne	1265, 1270, 1275, 1280
\@ignorefalse	1042	\m@th	9–11, 31, 41, 762, 1197, 1309, 1313
\@iiiparbox	9	\make@display@tag	1320, 1336
\@kernel@after@begindocument	1207	\math@cr	1223, 1227
\@kernel@before@begindocument	1092, 1121, 1203, 1214	\maxfields@	1332
\@kernel@math@registered@begin	949, 983, 1016, 1048	\measuring@true	1290
\@kernel@math@registered@end	1027, 1076	\multline@	1211
\@latex@error	1037	\on@line	1039
\@lign	1336	\place@tag	1337
\@namedef	1306, 1317, 1328, 1346	\prepnext@tok	156, 222
\@ne	1250, 1255, 1265, 1270, 1325, 1331	\rendmultline@	1287
\@nextchar	205, 211, 216	\reserved@a	1037, 1038, 1045
\@preamble	150	\restore@math@cr	1346, 1351
\@startpbox	205, 211, 216	\restorealignstate@	1230
\@subformulasfalse	1447	\row@	1325, 1331
		\setboxz@h	1336
		\split@tag	42

<code>\st@rredfalse</code>	1240, 1250, 1265, 1275	<code>\l__math_content_actual_tl</code>
<code>\st@rredtrue</code> 1217,	 12, 18, 595, 619, 646
	1245, 1255, 1260, 1270, 1280, 1285	<code>\l__math_content_AF_mathml_tl</code>	...
<code>\start@align</code> 1240, 1245, 1250,	 21, 603, 618
	1255, 1260, 1265, 1270, 1275, 1280	<code>\l__math_content_AF_source_tl</code>	...
<code>\start@gather</code> 1217	 19, 101, 599, 614, 629, 666, 698
<code>\start@multline</code> 1285	<code>\l__math_content_AF_source_tmpa_</code>	
<code>\strut@</code> 1336	<code>tl</code>	.. 20, 629, 630, 644, 698, 699, 709
<code>\tag@true</code> 1330	<code>\l__math_content_AF_tl</code> 12
<code>\textonly@unskip</code> 896	<code>\l__math_content_alt_tl</code>
<code>\totwidth@</code> 1223, 1227		.. 12, 17, 604, 610, 639, 647, 676, 711
<code>\tw@</code> 1260, 1275, 1280	<code>\l__math_content_hash_tl</code>
<code>\z@</code> 1240, 1245, 1343		112, 658, 665, 677, 683, 686, 692, 707
tex commands:		<code>\l__math_content_template_tl</code>	...
<code>\tex_cr:D</code> 907	 25, 563, 564, 597, 612
<code>\tex_everydisplay:D</code> 1164, 1166	<code>\l__math_env_name_tl</code>
<code>\tex_everymath:D</code> 1154, 1156	 35, 917, 923, 930, 964, 997
<code>\tex_the:D</code> 1156, 1166	<code>\g__math_grabbed_env_tl</code> 11,
<code>\text</code> 22		26, 27, 12, 567, 571, 578, 585, 587,
tl commands:			631, 645, 668, 710, 770, 1452, 1472
<code>\c_empty_tl</code> 490	<code>\g__math_grabbed_math_tl</code>
<code>\c_space_tl</code> 108,	 11, 26, 27, 13,
	444, 446, 448, 450, 452, 568, 570, 572		569, 581, 586, 667, 742, 771, 1441,
<code>\tl_clear:N</code> 630, 699, 846		1443, 1444, 1463, 1465, 1466, 1484
<code>\tl_const:Nn</code> 80, 88, 94, 575	<code>\l__math_grabbed_math_tl</code>	... 659, 667
<code>\tl_gput_right:Nn</code>	<code>\l__math_grabbed_tl</code>
 1092, 1121, 1203, 1207, 1214	 33, 841, 846, 862, 915
<code>\tl_gset:Nn</code> 44,	<code>\c__math_inline_env_tl</code> 575, 578
	320, 333, 334, 358, 770, 771, 1370, 1371	<code>\g__math_luaaml_load_tl</code>
<code>\tl_gset_eq:NN</code> 65	 19, 228, 319, 320, 333, 334, 358
<code>\tl_if_blank:nTF</code> 804, 829	<code>\c__math_mathml_write_after_tl</code>	..
<code>\tl_if_blank_p:n</code> 785	 80, 304, 377
<code>\tl_if_empty:NTF</code>	298, 1444, 1466, 1484	<code>\l__math_mathml_write_before_tl</code>	..
<code>\tl_if_eq:NNTF</code> 578	 80, 104, 298, 302, 375
<code>\tl_if_eq:NnTF</code> 631, 668	<code>\c__math_mathml_write_final_tl</code>	..
<code>\tl_if_exist:NTF</code> 57, 386, 677	 80, 312, 402
<code>\tl_if_in:NnTF</code> 1183	<code>\c__math_mathml_write_init_tl</code>	...
<code>\tl_if_in:nnTF</code> 762	 80, 295, 395
<code>\tl_map_inline:nn</code> 896	<code>\l__math_texsource_template_tl</code>	..
<code>\tl_new:N</code> 12, 13,	 26, 574, 576, 601, 616
	14, 17, 18, 19, 20, 21, 32, 43, 64, 87,	<code>\l__math_tmpa_tl</code> 11, 14,
	319, 563, 574, 658, 659, 841, 917, 1438		61, 63, 65, 1354, 1356, 1432, 1433, 1434
<code>\tl_put_right:Nn</code> 862	token commands:	
<code>\tl_set:Nn</code> 104,	<code>\token_to_str:N</code>
	564, 576, 595, 599, 603, 604, 610,	 868, 871, 873, 879, 884, 898
	614, 618, 619, 633, 636, 639, 665,	<code>\tracingall</code> 952, 986, 1019
	670, 673, 676, 930, 964, 997, 1354, 1432	<code>\tracingnone</code> 956, 989
<code>\tl_set_eq:NN</code> 629, 667, 698	<code>\typeout</code> 424, 429, 435, 495, 683,
<code>\tl_to_str:n</code> 276, 278		686, 692, 741, 742, 814, 838, 939,
<code>\tl_trim_spaces:apply:nN</code> 764		942, 973, 976, 1006, 1009, 1026,
tl internal commands:			1050, 1057, 1063, 1069, 1073, 1077,
<code>\l__math_attribute_class_tl</code>		1081, 1168, 1307, 1442, 1443, 1446,
 32, 633, 636, 643, 670, 673, 703		1450, 1464, 1465, 1468, 1471, 1494

U			
\unskip	896	\UseTaggingSocket	145, 1187
use commands:		V	
\use_i:nn	488	\vbox	205, 216
\use_ii_i:nn	1408	\vcenter	9
\UseHook	1035	\vtop	211