

The `latex-lab-math` code*

Frank Mittelbach, Joseph Wright, L^AT_EX Project

v0.6i 2024-10-25

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,

$$\l[a \in A \text{ for all } a \in A \r]$$

would only result in a single capture of the tokens “`a \in A \text{ for all } a \in 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 argument and its code should output it for typesetting. It is not *used* as a tagging socket as the math argument should not be lost without tagging, so without tagging the socket uses the identity plug. 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/foxitsoftware/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 \LaTeX 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 `\@iiparbox` 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 \LaTeXe 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 {}

```

`\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

323 \msg_new:nnn { tag }{ PDF-2.0-recommended }
324 {
325     The~key~#1~will~not~work~properly~with~PDF~#2.\\
326     Switching~to~PDF~2.0~is~recommended.
327 }
328 \keys_define:nn { __tag / setup }
329 {

```

At first a key to suppress the loading altogether

```

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

```

A key to activate math structure elements. It shouldn't be issued in the preamble as luamml is not yet loaded.

```

335     math/mathml/structelem .code:n =
336     {

```

```

337     \pdf_version_compare:NnT < {2.0}
338     {
339         \msg_warning:nne { tag }{ PDF-2.0-recommended }
340         { math/mathml/structelem }{ \pdf_version: }
341     }
342     \AddToHook{begindocument/end}{\luamml_structelem:}
343 },

```

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

```

344     math/mathml/ignore .code:n = {\luamml_ignore:},
345     math/mathml/luamml/write .choice:,
346     math/mathml/luamml/write/true .code:n =
347     {
348         \tl_gset:Nn \g__math_luamml_load_tl{1}
349         \bool_gset_true:N \g__math_luamml_write_bool
350     },
351     math/mathml/luamml/write/false .code:n =
352     {
353         \bool_gset_false:N \g__math_luamml_write_bool
354     },
355     math/mathml/luamml/write .default:n = true,
356     math/mathml/luamml/write .usage:n=preamble,

```

alias keys for compatibility

```

357     math/mathml/luamml .bool_gset:N = \g__math_luamml_write_bool,
358     math/mathml/luamml .usage:n=preamble
359 }

```

`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

```

360 \socket_new:n {tagsupport/math/mathml/write}{0}

```

On (*plug*)

```

361 \socket_new_plug:nnn{tagsupport/math/mathml/write}{On}
362 {
363     \iow_now:Ne \g__math_writedummy_iow
364     {
365         \l__math_mathml_write_before_tl
366         <math></math>
367         \c__math_mathml_write_after_tl
368     }
369 }

```

And now a key to activate the socket.

```

370
371 \keys_define:nn { __tag / setup }
372 {
373     math/mathml/write-dummy .code:n =
374     {
375         \bool_gset_true:N \g__tag_math_mathml_AF_bool
376         \tl_if_exist:NF\g__math_writedummy_iow
377         {
378             \iow_new:N \g__math_writedummy_iow
379             \iow_open:Nn \g__math_writedummy_iow

```

```

380     {
381       \c_sys_jobname_str-mathml-dummy.html
382     }
383     \iow_now:Ne \g__math_writedummy_iow
384     {
385       \c__math_mathml_write_init_tl
386     }
387     \AssignSocketPlug {tagsupport/math/mathml/write/prepare}{On}
388     \AssignSocketPlug {tagsupport/math/mathml/write}{On}
389     \AddToHook{enddocument/afterlastpage}
390     {
391       \iow_now:Ne \g__math_writedummy_iow
392       { \c__math_mathml_write_final_tl }
393       \iow_close:N \g__math_writedummy_iow
394     }
395   }
396 },
397 math/mathml/write-dummy .usage:n=preamble
398 }

```

_math_AF_process_mathml_files:

```

399 \box_new:N\l__math_tmpa_box
400 \cs_new_protected:Npn \_math_AF_process_mathml_files:
401 {
402   \hbox_set:Nn \l__math_tmpa_box
403   {
404     \pdfdict_put:nnn { l_pdffile/Filespec }{AFRelationship} { /Supplement }
405     \pdfdict_put:nne
406     { l_pdffile }{Subtype}
407     { \pdf_name_from_unicode_e:n{application/mathml+xml} }
408     \char_set_catcode_other:N \#
409     \cs_set_eq:NN\mml \_math_AF_html_reader:w
410     \clist_map_inline:Nn \l__tag_math_mathml_files_clist
411     {
412       \file_if_exist:nTF {##1.html}
413       {
414         \typeout{Info:~reading~mathml~file~##1}
415         \file_input:n {##1.html}
416         \bool_gset_true:N\g__tag_math_mathml_AF_bool
417       }
418       {
419         \typeout{Info:~mathml~file~##1~does~not~exist}%info message
420       }
421     }
422   }
423   \bool_if:NT\g__tag_math_mathml_AF_bool
424   {
425     \typeout{Info:~Activating~mathml~support}
426     \AssignSocketPlug{tagsupport/math/struct/begin}{mathml-AF}
427     \AssignSocketPlug{tagsupport/math/struct/end}{mathml-AF}
428     \AssignSocketPlug{tagsupport/math/substruct/begin}{single}
429     \AssignSocketPlug{tagsupport/math/substruct/end}{single}

```

mathml handling doesn't like subparts, so we disable them for now:

```

430 \AddToHook{enddocument/info}
431 {
432   \iow_term:n{MathML~statistic}
433   \iow_term:n{=====}
434   \iow_term:e{==>~\int_use:N\g__math_mathml_total_int\c_space_tl
435   MathML~fragments~read}
436   \iow_term:e{==>~\int_use:N\g__math_mathml_int\c_space_tl
437   different~MathML~fragments}
438   \iow_term:e{==>~\int_use:N\g__math_math_total_int\c_space_tl
439   math~fragments~found}
440   \iow_term:e{==>~\int_use:N\g__math_mathml_AF_found_int\c_space_tl
441   fitting~MathML~AF~found}
442   \iow_term:e{==>~\int_use:N\g__math_mathml_AF_attached_int\c_space_tl
443   MathML~AF~attached}
444 }
445 }
446 }
447 \AddToHook{begindocument}{\__math_AF_process_mathml_files:}

```

(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
448 \int_new:N \l__math_mathstyle_int
449 \int_new:N \g__math_mathchoice_int
450 \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
451 \cs_new_protected:Npn \__math_tag_if_mathstyle:nn #1 #2
452 %#1 refers to label
453 %#2 is a number for the mathstyle (typically 0,2,4,6)
454 {
455   \int_set:Nn \l__math_mathstyle_int {#2}
456   \property_record:nn {#1} { mathstyle }
457   \int_compare:nNnTF { \property_ref:nn {#1}{ mathstyle } } = { #2 }
458   { \tag_resume:n{\mathchoice} } { \tag_suspend:n{\mathchoice} }
459 }
460 \cs_generate_variant:Nn \__math_tag_if_mathstyle:nn {en}

```

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

5.7 Tagging options

```

461 \keys_define:nn { __tag / setup }
462 {
463   math/mathml/sources .clist_set:N = \l__tag_math_mathml_files_clist,
464   math/alt/use        .bool_set:N = \l__tag_math_alt_bool,
465   viewer/pane/mathml .bool_set:N = \l__tag_math_mathml_pane_bool,
466   viewer/pane/mathml .initial:n = true,
467   viewer/pane/mathsource .bool_set:N = \l__tag_math_texsource_pane_bool,
468   math/mathml/AF .bool_set:N = \l__tag_math_mathml_AF_bool,
469   math/mathml/AF .initial:n = true,
470   math/tex/AF .bool_set:N = \l__tag_math_texsource_AF_bool,
471   math/tex/AF .initial:n = true
472 }

```

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

```

473 \AddToHook{begindocument/end}
474 {
475   \str_if_eq:eeT
476     {1}
477     {
478       \exp_last_unbraced:Ne\use_i:nn
479         {\GetDocumentProperties{document/pdfstandard-UA}}
480       \c_empty_tl\c_empty_tl
481     }
482     {
483       \bool_if:NF \l__tag_math_alt_bool
484         {
485           \typeout{PDF/UA-1~detected.~Enabling~alt~text~on~Formula}
486         }
487       \bool_set_true:N\l__tag_math_alt_bool
488     }
489 }

```

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-
`\math/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!

```

490 \socket_new:nn {tagsupport/math/inline/begin}{0}
491 \socket_new:nn {tagsupport/math/inline/end}{0}
492 \socket_new:nn {tagsupport/math/inline/formula/begin}{1} %
493 \socket_new:nn {tagsupport/math/inline/formula/end}{0}

```

MC (*plug*)

```

494 \socket_new_plug:nnn
495   {tagsupport/math/inline/begin}
496   {MC}
497   {\tag_mc_end_push:}
498 \socket_new_plug:nnn
499   {tagsupport/math/inline/end}
500   {MC}
501   {\tag_mc_begin_pop:n{}}

```

We probably will want to test different tagging recipes.

default (*plug*)

```

502 \socket_new_plug:nnn
503   {tagsupport/math/inline/formula/begin}
504   {default}
505   { \tagpdfparaOff
506     \tag_socket_use:n{math/content}
507     \tag_socket_use:n{math/struct/begin}
508     % inner formula if multiple parts (not really implemented yet)
509     \tag_socket_use:n{math/substruct/begin}
510     #1
511     \tag_socket_use:n{math/end}
512   }
513 \socket_new_plug:nnn
514   {tagsupport/math/inline/formula/end}
515   {default}
516   {
517     \socket_use:n{tagsupport/math/substruct/end}
518     \socket_use:n{tagsupport/math/struct/end}
519   }

```

TODO: does inline math need subformula handling?

5.8.2 Main display math sockets

`tagsupport/math/display/begin` (*socket*) The first two sockets are meant to embed display math into the surrounding (so to close/reopen, e.g., MC-chunks and P-structure). The other two implement the actual formula structure. The formula sockets are despite their naming not symmetric: the `begin` socket is issued after the math has started, while the `end` socket is after the math! The socket `tagsupport/math/display/formula/begin` should be similar to the inline version not be used as tagging socket so that the argument, the math, is not lost.

```

520 \socket_new:nn {tagsupport/math/display/begin}{0}
521 \socket_new:nn {tagsupport/math/display/end}{0}
522 \socket_new:nn {tagsupport/math/display/formula/begin}{1} %
523 \socket_new:nn {tagsupport/math/display/formula/end}{0}

```

default (*plug*)

```

524 \socket_new_plug:nnn
525   {tagsupport/math/display/begin}
526   {default}
527   { \_tag_tool_close_P: }
528 \socket_new_plug:nnn
529   {tagsupport/math/display/end}
530   {default}
531   {
532   }

```

default (*plug*)

```

533 \socket_new_plug:nnn
534   {tagsupport/math/display/formula/begin}
535   {default}
536   {

```

```

537 \tag_socket_use:n{math/content}
538 \tag_socket_use:n{math/struct/begin}
539 \tag_socket_use:n{math/substruct/begin}
540 #1
541 \tag_socket_use:n{math/end}
542 }
543 \socket_new_plug:nnn
544 {tagsupport/math/display/formula/end}
545 {default}
546 {
547 \socket_use:n{tagsupport/math/substruct/end}
548 \socket_use:n{tagsupport/math/struct/end}
549 }

```

5.8.3 Internal sockets

\l__math_content_template_tl

The default text used as alt or actual text.

```

550 \tl_new:N\l__math_content_template_tl
551 \tl_set:Nn \l__math_content_template_tl
552 {
553 LaTeX~ formula~ starts~
554 \exp_not:N\begin{\g__math_grabbed_env_tl}
555 \c_space_tl
556 \exp_not:V\g__math_grabbed_math_tl
557 \c_space_tl
558 \exp_not:N\end{\g__math_grabbed_env_tl}
559 \c_space_tl LaTeX~ formula~ ends~
560 }

```

\l__math_texsource_template_tl

The default text used as texsource

```

561 \tl_new:N\l__math_texsource_template_tl
562 \tl_const:Nn\c__math_inline_env_tl {math}
563 \tl_set:Nn \l__math_texsource_template_tl
564 {
565 \tl_if_eq:NNTF\g__math_grabbed_env_tl\c__math_inline_env_tl
566 {
567 $
568 \exp_not:V\g__math_grabbed_math_tl
569 $
570 }
571 {
572 \exp_not:N\begin{\g__math_grabbed_env_tl}
573 \exp_not:V\g__math_grabbed_math_tl
574 \exp_not:N\end{\g__math_grabbed_env_tl}
575 }
576 }

```

`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`

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

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

`actual+source` (*plug*)

```
578 \socket_new_plug:nnn
579   {tagsupport/math/content}
580   {actual+source}
581   {
582     \tl_set:Ne\l__math_content_actual_tl
583     {
584       \l__math_content_template_tl
585     }
586     \tl_set:Ne \l__math_content_AF_source_tl
587     {
588       \l__math_texsource_template_tl
589     }
590     \tl_set:Nn \l__math_content_AF_mathml_tl {}
591     \tl_set:Nn \l__math_content_alt_tl {}
592   }
```

`alt+source` (*plug*)

```
593 \socket_new_plug:nnn
594   {tagsupport/math/content}
595   {alt+source}
596   {
597     \tl_set:Ne\l__math_content_alt_tl
598     {
599       \l__math_content_template_tl
600     }
601     \tl_set:Ne \l__math_content_AF_source_tl
602     {
603       \l__math_texsource_template_tl
604     }
605     \tl_set:Nn \l__math_content_AF_mathml_tl {}
606     \tl_set:Nn \l__math_content_actual_tl {}
607   }
608 \socket_assign_plug:nm {tagsupport/math/content}{alt+source}
```

`support/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`

```
609 \socket_new:nn {tagsupport/math/struct/begin}{0}
610 \socket_new:nn {tagsupport/math/struct/end}{0}
```

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

```
611 \socket_new_plug:nnn
612   {tagsupport/math/struct/begin}
```

```

613 {default}
614 {
615   \bool_if:NTF\l__tag_math_texsource_AF_bool
616   { \tl_set_eq:NN \l__math_content_AF_source_tmpa_tl \l__math_content_AF_source_tl }
617   { \tl_clear:N \l__math_content_AF_source_tmpa_tl }
618   \tl_if_eq:NnTF\g__math_grabbed_env_tl {math}
619   {
620     \tl_set:Nn\l__math_attribute_class_tl{inline}
621   }
622   {
623     \tl_set:Nn\l__math_attribute_class_tl{display}
624   }
625   \bool_if:NF\l__tag_math_alt_bool
626   { \tl_set:Nn \l__math_content_alt_tl{} }
627   \tag_struct_begin:n
628   {
629     tag=Formula,
630     attribute-class=\l__math_attribute_class_tl,
631     texsource      = \l__math_content_AF_source_tmpa_tl,
632     title-o        = \g__math_grabbed_env_tl,
633     actualtext     = \l__math_content_actual_tl,
634     alt            = \l__math_content_alt_tl
635   }
636 }
637 \socket_new_plug:nnn
638 {tagsupport/math/struct/end}
639 {default}
640 { \tag_struct_end: }
641
642 \socket_assign_plug:nn {tagsupport/math/struct/begin}{default}
643 \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`

```

644 \cs_generate_variant:Nn \str_mdfive_hash:n {o}
645 \tl_new:N\l__math_content_hash_tl

```

we need to save the grabbed math:

```

646 \tl_new:N\l__math_grabbed_math_tl

```

the socket definition

```

647 \socket_new_plug:nnn
648 {tagsupport/math/struct/begin}
649 {mathml-AF}
650 {
651   \int_gincr:N\g__math_math_total_int
652   \tl_set:Ne\l__math_content_hash_tl
653   {\str_mdfive_hash:o { \l__math_content_AF_source_tl }}
654   \tl_set_eq:NN\l__math_grabbed_math_tl\g__math_grabbed_math_tl
655   \tl_if_eq:NnTF\g__math_grabbed_env_tl {math}
656   {
657     \tl_set:Nn\l__math_attribute_class_tl{inline}
658   }
659   {

```

```

660     \tl_set:Nn\l__math_attribute_class_tl{display}
661   }
662   \bool_if:NF\l__tag_math_alt_bool
663     { \tl_set:Nn \l__math_content_alt_tl{} }
debugging option. TODO: hide in debug key.
664   \tl_if_exist:cTF { g__math_mathml_ \l__math_content_hash_tl _tl }
665     {
666       \int_gincr:N\g__math_mathml_AF_found_int
667       \bool_if:NTF \l__tag_math_mathml_AF_bool
668         {
669           \int_gincr:N\g__math_mathml_AF_attached_int
670           \typeout {Inserting~mathml~with~Hash~\l__math_content_hash_tl}
671         }
672         {
673           \typeout {Ignoring~mathml~with~Hash~\l__math_content_hash_tl}
674         }
675       }
676     {
677       \bool_if:NT \l__tag_math_mathml_AF_bool
678         {
679           \typeout {WARNING:~mathml~missing~for~hash~\l__math_content_hash_tl}
680         }
681       }
682     \socket_use:n {tagssupport/math/mathml/write/prepare}
683     \socket_use:n {tagssupport/math/mathml/write} % write hash if request
684     \bool_if:NTF\l__tag_math_texsource_AF_bool
685       { \tl_set_eq:NN \l__math_content_AF_source_tmpa_tl \l__math_content_AF_source_tl }
686       { \tl_clear:N \l__math_content_AF_source_tmpa_tl }
687     \tag_struct_begin:n
688       {
689         tag=Formula,
690         attribute-class=\l__math_attribute_class_tl, %
691         AFref           =
692           \bool_if:NT\l__tag_math_mathml_AF_bool
693             {
694               \cs_if_exist_use:c {g__math_mathml_ \l__math_content_hash_tl _tl}
695             },
696         texsource      = \l__math_content_AF_source_tmpa_tl, % should be after mathml AF!
697         title-o        = \g__math_grabbed_env_tl, %
698         alt             = \l__math_content_alt_tl
699       }
700     }

```

not really needed but looks more symmetric:

```

701 \socket_new_plug:nnn
702   {tagssupport/math/struct/end}
703   {mathml-AF}
704   {
705     \tag_struct_end:
706   }

```

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

```

port/math/substruct/end (socket) 707 \socket_new:nn {tagssupport/math/substruct/begin}{0}
708 \socket_new:nn {tagssupport/math/substruct/end}{0}

```

default (*plug*)

```
709 \socket_new_plug:nnn
710 {tagsupport/math/substruct/begin}
711 {default}
712 { \grabaformulapartandstart }
713 \socket_new_plug:nnn
714 {tagsupport/math/substruct/end}
715 {default}
716 {
717   \tagmccend
718   \if@subformulas
719     \tagstructend
720   \fi
721 }
722 \socket_assign_plug:nn {tagsupport/math/substruct/begin}{default}
723 \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:

```
724 \socket_new_plug:nnn
725 {tagsupport/math/substruct/begin}
726 {single}
727 {
728   \typeout{====>subpart~splitting~deactivated}
729   \typeout{====>grabbed~math=\meaning\g__math_grabbed_math_tl}
730   \tag_mc_begin:n{ }
731 }
732 \socket_new_plug:nnn
733 {tagsupport/math/substruct/end}
734 {single}
735 { \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.

```
736 \socket_new:nn {tagsupport/math/end}{0}
```

_tag_math_disable: Similar to the table code we collect the plugs that should be assigned to do nothing if we don't want tagging

```
737 \cs_new_protected:Npn \_tag_math_disable:
738 {
739   \socket_assign_plug:nn {tagsupport/math/inline/begin}{noop}
740   \socket_assign_plug:nn {tagsupport/math/inline/end}{noop}
741   \socket_assign_plug:nn {tagsupport/math/inline/formula/begin}{identity}
742   \socket_assign_plug:nn {tagsupport/math/inline/formula/end}{noop}
743   \socket_assign_plug:nn {tagsupport/math/display/begin}{noop}
744   \socket_assign_plug:nn {tagsupport/math/display/end}{noop}
745   \socket_assign_plug:nn {tagsupport/math/display/formula/begin}{identity}
746   \socket_assign_plug:nn {tagsupport/math/display/formula/end}{noop}
747 }
```

(End of definition for _tag_math_disable:.)

_tag_math_enable: Similar to the table code we collect the default plugs that should be assigned if we want tagging

```

748 \cs_new_protected:Npn \__tag_math_enable:
749 {
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}
758 }

```

(End of definition for `__tag_math_enable:`.)

At begin document we can activate:

```

759 \AtBeginDocument{\tag_if_active:T{\__tag_math_enable: }}

```

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`

```

760 \cs_new_protected:Npn \__math_process:nn #1#2
761 {
762   \legacy_if:nF {measuring@ }
763   {
764     \tl_if_in:nnTF {#2} { \m@th }
765     { \bool_set_true:N\l__math_fakemath_bool }
766     { \tl_trim_spaces_apply:nN {#2} \__math_process_auxi:nn {#1} }
767   }
768 }
769 \cs_generate_variant:Nn \__math_process:nn { V }
770 \cs_new_protected:Npn \__math_process_auxi:nn #1#2
771 {
772   \tl_gset:Nn \g__math_grabbed_env_tl {#2}
773   \tl_gset:Nn \g__math_grabbed_math_tl {#1}
774   \__math_process_auxii:nn {#2} {#1}
775 }
776 \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

```

777 \cs_new_protected:Npn \math_processor:n #1
778 { \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.

```

779 \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.

```
780 \cs_new_protected:Npn \__math_grab_dollar:n #1
781 {
```

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

```
782   \__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.

```
783   \bool_lazy_any:nTF
784     {
785     {\legacy_if_p:n { measuring@ }}
786     { \l__math_fakemath_bool }
787     { \tl_if_blank_p:n {#1} }
788   }
789   {
790     #1 $ % $
791   }
792   {
793     \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.

```
794     \socket_use:nn {tagsupport/math/inline/formula/begin}{#1}
795     $ % $
796     \tag_socket_use:n {math/inline/formula/end}
797     \tag_socket_use:n {math/inline/end} % restart P-MC
798   }
799 }
```

(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.

```
800 \cs_new_protected:Npn \__math_grab_dollar_delim:w #1 $ % $
801 { \__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.

```
802 \cs_new_protected:Npn \__math_grab_dollardollar:w % $$
803 #1 $$
804 {
805   \tl_if_blank:nF {#1}
806   {
807     \__math_process:nn { equation* } {#1}
808     \tag_socket_use:n {math/display/begin}
809     \socket_use:nn{tagsupport/math/display/formula/begin}{#1}
810   }
811   $$
812 }
```

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

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

TODO why is that needed? where is para-tagging disabled?

```
818   \tagpdfpara0n
```

The `\postdisplaypenalty` was temporarily set to 10000 inside the display and the `\belowdisplayskip` and the `\belowdisplayshortskip` 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.

```
819   \l__math_tmpa_skip \lastskip
820   \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”.

```
821   \nobreak
822   \skip_vertical:n { -\l__math_tmpa_skip } % remove the negative belowdisplayskip
```

Then we finally add the real stuff:

```
823   \penalty \postdisplaypenalty
824   \skip_vertical:n { -\l__math_tmpa_skip } % insert the correct skip
825   \@doendpe           % this has no \end{...} to take care of it
826 }
827
```

(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.

```
828 \cs_new_protected:Npn \__math_grab_inline:w % \langle
829 #1 \rangle
830 {
831   \tl_if_blank:nF {#1}
832   {
833     $ #1 $
834   }
835   \bool_set_false:N \l__math_collected_bool
836 }
```

(End of definition for __math_grab_inline:w.)

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

```
837 \cs_new_protected:Npn \__math_grab_eqn:w % \[
838 #1 \]
839 {
840   % \typeout{collected? = \bool_if:NTF \l__math_collected_bool {true}{false}}
841   \begin { equation* } #1 \end { equation* }
842 }
```

(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.

```
843 \tl_new:N \l__math_grabbed_tl
```

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

```
844 \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:`

```
845 \cs_new_protected:Npn \__math_grab_dollar_loop:
846 {
847   \group_begin:
848   \tl_clear:N \l__math_grabbed_tl
849   \__math_grab_loop:
850 }
851 \cs_new_protected:Npn \__math_grab_loop:
852 {
853   \peek_remove_spaces:n
854   {
855     \peek_meaning:NTF \c_group_begin_token
856     { \__math_grab_loop_group:n }
857     { \__math_grab_loop_token:N }
858   }
859 }
```

(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`

```
860 \cs_new_protected:Npn \__math_grab_loop_group:n #1
861 { \__math_grab_loop_store:n { #1 } }
862 \cs_new_protected:Npn \__math_grab_loop_store:n #1
863 {
864   \tl_put_right:Nn \l__math_grabbed_tl {#1}
865   \__math_grab_loop:
866 }
```

(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:`

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

```
868 {
```

`__math_grab_loop_\begin:`

`__math_grab_loop_\end:`

`__math_grab_loop_\ignorespaces:`

`__math_grab_loop_\unskip:`

`__math_grab_loop_\textonly@unskip:`

```

869 \cs_if_exist_use:cF
870 { __math_grab_loop_ \token_to_str:N #1 : }
871 { \__math_grab_loop_store:n {#1} }
872 }
873 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N $ : }
874 { \__math_grab_loop_end: }
875 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N \ : }
876 {
877 \int_compare:nNnTF \l__math_grab_env_int = 0
878 { \__math_grab_loop_newline: }
879 { \__math_grab_loop_store:n { \ } }
880 }

```

In contrast to `colcell`, 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.

```

881 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N \begin : }
882 {
883 \int_incr:N \l__math_grab_env_int
884 \__math_grab_loop_store:n { \begin }
885 }
886 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N \end : }
887 {
888 \int_compare:nNnTF \l__math_grab_env_int = 0
889 {
890 \__math_grab_loop_newline:
891 \end
892 }
893 {
894 \int_decr:N \l__math_grab_env_int
895 \__math_grab_loop_store:n { \end }
896 }
897 }
898 \tl_map_inline:nn { \ignorespaces \unskip \textonly@unskip }
899 {
900 \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N #1 : }
901 { \__math_grab_loop: }
902 }

```

(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`.

```

903 \cs_new_protected:Npn \__math_grab_loop_newline:
904 {
905 \if_false: { \fi:
906 \cs_set_protected:Npn \cr
907 {
908 \__math_grab_loop:

```

```

909     \tex_cr:D
910   }
911   \if_false: } \fi:
912   \\\
913 }

```

(End of definition for `_math_grab_loop_newline:.`)

`_math_grab_loop_end:` Clean up and pass on.

```

914 \cs_new_protected:Npn \_math_grab_loop_end:
915 {
916   \exp_args:NNV \group_end:
917   \_math_grab_dollar:n \l\_math_grabbed_tl
918 }

```

(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”

```

919 \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 `lcmd`.

```

920 \keys_define:nn { \_math }
921 {
922   arg-spec .code:n =
923     {
924       \ExpandArgs { c } \DeclareDocumentCommand
925       { \_math_env \l\_math_env_name_tl _aux: }
926       {#1}
927       { \_math_env_forward:w }
928     }
929 }

```

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

```

930 \cs_new_protected:Npn \math_register_env:nn #1#2
931 {
932   \tl_set:Nn \l\_math_env_name_tl {#1}
933   \keys_set:nn { \_math } {#2}
934   \cs_gset_eq:cc { \_math_env_ #1 _begin: } {#1}
935   \cs_gset_eq:cc { \_math_env_ #1 _end: } { end #1 }
936   %
937   \ExpandArgs { nne } \RenewDocumentEnvironment {#1} { b }
938   {
939     \exp_not:N \bool_if:NTF \exp_not:N \l\_math_collected_bool
940     {

```

```

941 %           \typeout{===>B1}
942         }
943       {
944 %           \typeout{===>B2}
945       \cs_if_exist:cTF { __math_env #1 _aux: }
946         {
947           \exp_not:c { __math_env #1 _aux: }
948             ##1 \exp_not:N \__math_env_end: {#1}
949         }
950         { \exp_not:N \__math_process:nn {#1} {##1} }
951       \exp_not:n { \@kernel@math@registered@begin }
952       \bool_set_true:N \exp_not:N \l__math_collected_bool
953     }
954 %       \exp_not:N \tracingall
955     \exp_not:c { __math_env_ #1 _begin: }
956     ##1
957     \exp_not:c { __math_env_ #1 _end: }
958 %       \exp_not:N \tracingnone
959   }
960   {
961   }
962 }
963
964 \cs_new_protected:Npn \math_register_halign_env:nn #1#2
965 {
966   \tl_set:Nn \l__math_env_name_tl {#1}
967   \keys_set:nn { __math } {#2}
968   \cs_gset_eq:cc { __math_env_ #1 _begin: } {#1}
969   \cs_gset_eq:cc { __math_env_ #1 _end: } { end #1 }
970 %
971   \ExpandArgs { nnee } \RenewDocumentEnvironment {#1} { b }
972   {
973     \exp_not:N \bool_if:NTF \exp_not:N \l__math_collected_bool
974     {
975 %       \typeout{===>B1}
976     }
977     {
978 %       \typeout{===>B2}
979     \cs_if_exist:cTF { __math_env #1 _aux: }
980       {
981         \exp_not:c { __math_env #1 _aux: }
982           ##1 \exp_not:N \__math_env_end: {#1}
983       }
984       { \exp_not:N \__math_process:nn {#1} {##1} }
985     \exp_not:n { \@kernel@math@registered@begin }
986     \bool_set_true:N \exp_not:N \l__math_collected_bool
987   }
988 %     \exp_not:N \tracingall
989     \exp_not:c { __math_env_ #1 _begin: }
990     ##1
991 %     \exp_not:N \tracingnone
992   }
993   {
994     \exp_not:c { __math_env_ #1 _end: }

```

```

995     }
996 }

```

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

```

997 \cs_new_protected:Npn \math_register_odd_env:nn #1#2
998 {
999   \tl_set:Nn \l__math_env_name_tl {#1}
1000   \keys_set:nn { __math } {#2}
1001   \cs_gset_eq:cc { __math_env_ #1 _begin: } {#1}
1002   \cs_gset_eq:cc { __math_env_ #1 _end: } { end #1 }
1003 %
1004   \ExpandArgs { nnee } \RenewDocumentEnvironment {#1} { b }
1005   {
1006     \exp_not:N \bool_if:NTF \exp_not:N \l__math_collected_bool
1007     {
1008       %
1009       \typeout{===>B1}
1010     }
1011     {
1012       %
1013       \typeout{===>B2}
1014       \cs_if_exist:cTF { __math_env_ #1 _aux: }
1015       {
1016         \exp_not:c { __math_env_ #1 _aux: }
1017         ##1 \exp_not:N \__math_env_end: {#1}
1018       }
1019       { \exp_not:N \__math_process:nn {#1} {##1} }
1020       \exp_not:n { \@kernel@math@registered@begin }
1021       \bool_set_true:N \exp_not:N \l__math_collected_bool
1022     }
1023   }
1024   \exp_not:N \tracingall
1025   \exp_not:c { __math_env_ #1 _begin: }
1026   ##1
1027 }
1028 {
1029   \exp_not:c { __math_env_ #1 _end: }
1030 % needed if we don't have $$...$$
1031   \exp_not:n { \typeout{---> @kernel@math@registered@end } }
1032   \exp_not:n { \@kernel@math@registered@end }
1033 }
1034 }
1035 %
1036 % FMi: compare with block change!
1037 %
1038 % \DeclareRobustCommand*\begin[1]{%
1039 % \UseHook{env/#1/before}%
1040 % \ifundefined{#1}%
1041 %   {\def\reserved@a{\@latex@error{Environment #1 undefined}\@eha}}%
1042 %   {\def\reserved@a{\def\@currenvir{#1}%
1043 %     \edef\@currenvline{\on@line}%
1044 %     \@execute@begin@hook{#1}%
1045 %     \csname #1\endcsname}}%
1046 % \ignorefalse
1047 % \begin@group
1048 % \end@pefalse % tmp!!! is it ok to drop this here?

```

```

1047 % \reserved@a}
1048
1049
1050 \cs_new:Npn \@kernel@math@registered@begin {
1051 % \ShowTagging{struct-stack}
1052 %\typeout{==>A1}\ShowTagging{struct-stack,mc-current}
1053 \mode_if_vertical:TF
1054 {
1055 % \legacy_if:nTF { @endpe }
1056 % { \legacy_if_set_false:n { @endpe } }
1057 % { \__block_list_beginpar_vmode: }
1058 %
1059 % \typeout{==>~ at:~ \g__tag_struct_tag_tl}
1060 %
1061 \tag_if_active:T
1062 {
1063 \exp_args:Noo\str_if_eq:nnF \g__tag_struct_tag_tl { \l__tag_para_main_tag_tl }
1064 {
1065 % \typeout{==>A2}
1066 \__block_beginpar_vmode:
1067 } % needs correction!
1068 }
1069 }
1070 {
1071 % \typeout{==>A3}
1072 \__tag_tool_close_P:
1073 }
1074 \socket_use:nn{tagsupport/math/display/formula/begin}{ }
1075 \tagpdfparaOff
1076 % \typeout{==>MC1}\ShowTagging{mc-current}
1077 }
1078
1079 \cs_new:Npn \@kernel@math@registered@end {
1080 % \typeout{==>MC2}\ShowTagging{mc-current}
1081 \para_raw_end:
1082 \tagpdfparaOn
1083 \socket_use:nn{tagsupport/math/display/formula/end}
1084 % \typeout{==>MC3}\ShowTagging{mc-current}
1085 \@endpetrue
1086 }
1087
1088 \cs_new_protected:Npn \math_register_env:n #1
1089 { \math_register_env:nn {#1} { } }
1090
1091 \NewDocumentCommand \RegisterMathEnvironment { 0{ } m }
1092 { \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`

```

1093 \cs_new_protected:Npn \__math_env_forward:w #1 \__math_env_end: #2
1094 { \__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.

```

1095 \tl_gput_right:Nn \@kernel@before@begindocument
1096 {
1097   \math_register_env:n { equation }
1098   \math_register_env:n { equation* }
1099   % at the moment register_env can only do display math
1100   % \math_register_env:n { math }
1101   \RenewDocumentEnvironment{math} {b}{\${#1$}}{}
1102   % and this one doesn't work either
1103   % \math_register_env:n { displaymath }
1104   \RenewDocumentEnvironment{displaymath} {b}{\[#1\]}{}
1105 }

```

(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.

```

1106 \cs_gset_protected:Npn \( % \)
1107 {
1108   \bool_if:NTF \l_math_collected_bool
1109   {
1110     \mode_if_math:TF
1111     { \badmath }
1112     { $ }
1113   }
1114   {
1115     \__math_grab_inline:w
1116   }
1117 } % \(
1118 \cs_gset_protected:Npn \)
1119 {
1120   \mode_if_math:TF
1121   { $ }
1122   { \badmath }
1123 }

```

(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.

```

1124 \tl_gput_right:Nn \@kernel@before@begindocument

```

```

1125 {
1126   \cs_gset_protected:Npn \[ % \]
1127   {
1128     \__math_grab_eqn:w
1129 %     \bool_if:NTF \l__math_collected_bool
1130 %     { \begin { equation* } }
1131 %     { \__math_grab_eqn:w }
1132   } % \[
1133   \cs_gset_protected:Npn \]
1134   {
1135     \@badmath
1136 %     \bool_if:NTF \l__math_collected_bool
1137 %     { \end{ equation* } }
1138 %     { \@badmath }
1139   }
1140 }

```

(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.

```

1141 %\cs_gset_protected:Npn \ensuremath #1
1142 % {
1143 %   \mode_if_math:TF
1144 %   {#1}
1145 %   {
1146 %     \bool_if:NTF \l__math_collected_bool
1147 %     { \@ensuredmath {#1} }
1148 %     {
1149 %       \bool_set_true:N \l__math_collected_bool
1150 %       \__math_process:nn { math } {#1}
1151 %       \@ensuredmath {#1}
1152 %       \bool_set_false:N \l__math_collected_bool
1153 %     }
1154 %   }
1155 % }

```

(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.

```

1156
1157 \exp_args:No \tex_everymath:D
1158 {
1159   \tex_the:D \tex_everymath:D
1160   \bool_if:NF \l__math_collected_bool
1161   {
1162     \bool_set_true:N \l__math_collected_bool
1163     \__math_grab_dollar:w
1164   }
1165 }
1166
1167 \exp_args:No \tex_everydisplay:D

```

```

1168 {
1169   \tex_the:D \tex_everydisplay:D
1170   \iftrue % this may have to be a settable flag!
1171 %       \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)

```

1172     \skip_set:Nn \belowdisplayskip      {-\belowdisplayskip}
1173     \skip_set:Nn \belowdisplayshortskip {-\belowdisplayshortskip}
1174     \int_set:Nn \postdisplaypenalty {10000}
1175     \group_insert_after:N \__math_tag_dollardollar_display_end:
1176   \fi
1177   \bool_if:NF \l__math_collected_bool
1178   {
1179     \bool_set_true:N \l__math_collected_bool
1180     \__math_grab_dollardollar:w
1181   }
1182 }

```

5.15 Modifying kernel environments

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

```

1183 \math_register_env:n { eqnarray }
1184 \math_register_env:n { eqnarray* }

```

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

```

1185 \RequirePackage{array}
1186 \tl_if_in:NnT\@tabular{$}
1187 {
1188   \def\@tabular{%
1189     \leavevmode
1190     \UseTaggingSocket{tbl/hmode/begin}%
1191     \hbox \bgroup
1192     \bool_set_true:N \l__math_collected_bool
1193     $
1194     \bool_set_false:N \l__math_collected_bool
1195     \col@sep\tabcolsep \let\d@llarbegin\begin\group
1196                                     \let\d@llarend\endgroup

```

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

```

1197   \cs_set_protected:Npn \__math_grab_dollar:w { \__math_grab_dollar_loop: }
1198   \@tabarray}
1199 }

```

`__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!

```

1200 \cs_new_eq:NN \__math_m@th: \m@th
1201 \cs_gset_protected:Npn \m@th
1202 {
1203   \bool_set_true:N \l__math_collected_bool

```

```

1204     \_math_m@th:
1205   }

```

(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.

```

1206 \tl_gput_right:Nn\@kernel@before@begindocument
1207   {
1208     \bool_set_true:N\l__math_collected_bool
1209   }
1210 \tl_gput_right:Nn\@kernel@after@begindocument
1211   {
1212     \bool_set_false:N\l__math_collected_bool
1213   }

```

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@

```

```

1214
1215
1216
1217 \tl_gput_right:Nn \@kernel@before@begindocument {
1218   %
1219   \renewenvironment{gather*}{%
1220     \start@gather\st@rredtrue
1221   }
1222   {%
1223     % this redirection doesn't work if we alter "gather"!
1224     % \endgather
1225     % so replace it with its real meaning
1226     \math@cr \black@\totwidth@ \egroup
1227     $$\ignorespacesafterend
1228   }
1229   \def\common@align@ending {
1230     \math@cr \black@\totwidth@
1231     \egroup
1232     \ifingather@
1233       \restorealignstate@
1234       \egroup
1235       \nonumber
1236       \ifnum0='{ \fi\iffalse}\fi
1237     \else
1238       $$$
1239     \fi
1240     \ignorespacesafterend

```

```

1241 }
1242 \renewenvironment{alignat}{%
1243   \start@align@z@\st@rredfalse
1244 }{%
1245   \common@align@ending
1246 }
1247 \renewenvironment{alignat*}{%
1248   \start@align@z@\st@rredtrue
1249 }{%
1250   \common@align@ending
1251 }
1252 \renewenvironment{xalignat}{%
1253   \start@align@ne\st@rredfalse
1254 }{%
1255   \common@align@ending
1256 }
1257 \renewenvironment{xalignat*}{%
1258   \start@align@ne\st@rredtrue
1259 }{%
1260   \common@align@ending
1261 }
1262 \renewenvironment{xxalignat}{%
1263   \start@align@tw@\st@rredtrue
1264 }{%
1265   \common@align@ending
1266 }
1267 \renewenvironment{align}{%
1268   \start@align@ne\st@rredfalse\m@ne
1269 }{%
1270   \common@align@ending
1271 }
1272 \renewenvironment{align*}{%
1273   \start@align@ne\st@rredtrue\m@ne
1274 }{%
1275   \common@align@ending
1276 }
1277 \renewenvironment{flalign}{%
1278   \start@align@tw@\st@rredfalse\m@ne
1279 }{%
1280   \common@align@ending
1281 }
1282 \renewenvironment{flalign*}{%
1283   \start@align@tw@\st@rredtrue\m@ne
1284 }{%
1285   \common@align@ending
1286 }
1287 %
1288 \renewenvironment{multline*}{\start@multline\st@rredtrue}
1289 {%
1290   \iftagsleft@ \exp\lendmultline@ \else \exp\rendmultline@ \fi
1291   \ignorespacesafterend
1292 }

```

Also for false?

```

1293 \def\measuring@true{\let\ifmeasuring@\iftrue\tag_suspend:n{\measuring}}
1294 %
1295 \math_register_halign_env:nn {align}{}
1296 \math_register_halign_env:nn {align*}{}
1297 \math_register_halign_env:nn {alignat}{}
1298 \math_register_halign_env:nn {alignat*}{}
1299 \math_register_halign_env:nn {flalign}{}
1300 \math_register_halign_env:nn {flalign*}{}
1301 \math_register_halign_env:nn {gather}{}
1302 \math_register_halign_env:nn {gather*}{}
1303 \math_register_halign_env:nn {multiline}{}
1304 \math_register_halign_env:nn {multiline*}{}
1305 \math_register_halign_env:nn {xalignat}{}
1306 \math_register_halign_env:nn {xalignat*}{}
1307 \math_register_halign_env:nn {xxalignat}{}
1308 %
1309 \@namedef{maketag @ @ @} #1{%
1310 % \typeout{--->maketag @ @ @}
1311 \ifmeasuring@
1312 \hbox{\m@th\normalfont#1}%
1313 \else
1314 \tagmccend \tagstructbegin{tag=Lbl}%
1315 \tagmccbegin{tag=Lbl}%
1316 \hbox{\m@th\normalfont#1}%
1317 \tagmccend \tagstructend \tagmccbegin{}%
1318 \fi
1319 }
1320 \@namedef{math@cr @ @ @ gather}{%
1321 \ifst@rred\nonumber\fi
1322 &\relax
1323 \make@display@tag
1324 %
1325 \maybestartnewformulatag
1326 %
1327 \ifst@rred\else\global\@eqnswtrue\fi
1328 \global\advance\row@\@ne
1329 \cr
1330 }
1331 \@namedef{math@cr @ @ @ align}{%
1332 \ifst@rred\nonumber\fi
1333 \if@eqnsw \global\tag@true \fi
1334 \global\advance\row@\@ne
1335 \add@amps\maxfields@
1336 \omit
1337 \kern-\alignsep@
1338 \iftag@
1339 \setboxz@h{\@lign\strut@{\make@display@tag}}%
1340 \place@tag
1341 \fi
1342 %
1343 \maybestartnewformulatag
1344 %
1345 \ifst@rred\else\global\@eqnswtrue\fi

```

```

1346 \global\lineht@z@
1347 \cr
1348 }
1349 \def\restore@math@cr{\@namedef{math@cr @ @ @}{
1350 %
1351 \maybestartnewformulatag
1352 %
1353 \cr}}
1354 \restore@math@cr
1355 }

```

(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.

```

1356 \cs_new:Npn \_math_split_at_nl:NN #1#2 {
1357 \tl_set:Nf \l_math_tmpa_tl {
1358 \exp_after:wN \_math_split_at_nl_first:w #1 \ \ \q_nil \ \ \s_stop }
1359 \exp_after:wN \_math_split_at_nl_aux:nnNN \l_math_tmpa_tl #1 #2
1360 }

```

and the auxiliary commands

```

1361 \cs_new:Npn \_math_split_at_nl_first:w #1 \ \ #2 \ \ #3 \s_stop
1362 {
1363 \quark_if_nil:nTF {#2}
1364 { {#1} { } }
1365 {
1366 \_math_split_chk_if_begin:ww #1 \begin \q_nil \s_mark
1367 #2 \ \ #3 \s_stop
1368 }
1369 }

```

```

1370
1371 \cs_new_protected:Npn \_math_split_at_nl_aux:nnNN #1 #2 #3 #4
1372 {
1373 \tl_gset:Nn #4 {#1}
1374 \tl_gset:Nn #3 {#2}
1375 }

```

```

1376
1377 \cs_new:Npn \_math_split_chk_if_begin:ww
1378 #1 \begin #2 #3 \s_mark #4 \ \ \q_nil \ \ \s_stop
1379 {
1380 \quark_if_nil:nTF {#2}
1381 { {#1} {#4} }
1382 {
1383 \exp_after:wN \_math_split_collect_one_end:w
1384 \_math_split_cleanup_begin_q_nil:w #1 \begin{#2} #3 \ \ #4 \s_stop
1385 { } { 1 }
1386 }
1387 }

```

```

1388
1389 \cs_new:Npn \_math_split_cleanup_begin_q_nil:w #1 \begin \q_nil {#1}
1390
1391 \cs_new:Npn \_math_split_collect_one_end:w #1 \end #2 #3 \s_stop #4 #5
1392 {

```

```

1393     \exp_args:Nf \_math_split_check_count_begins:nmmm
1394     { \_math_split_count_begins:n { #4 #1 } } {#5}
1395     { #4 #1 \end{#2} } {#3}
1396   }
1397 \cs_new:Npn \_math_split_count_begins:n #1
1398   { \int_eval:n { 0 \_math_split_count_begins:w #1 \begin \q_nil } }
1399
1400 \cs_new:Npn \_math_split_count_begins:w #1 \begin #2
1401   { \quark_if_nil:nF {#2} { +1 \_math_split_count_begins:w } }
1402
1403 \cs_new:Npn \_math_split_check_count_begins:nmmm #1 #2 #3 #4
1404   {
1405     \int_compare:nNnTF {#1} = {#2}
1406     {
1407       \exp_last_unbraced:Nf \_math_split_final_cleanup:nn
1408       { \_math_split:n { \_math_split_guard:n {#3} #4 } }
1409     }
1410     {
1411       \exp_args:No \use_ii_i:nn
1412       { \exp_after:wN { \int_value:w \int_eval:n { #2 + 1 } } }
1413       { \_math_split_collect_one_end:w #4 \s_stop {#3} }
1414     }
1415   }
1416 \cs_new:Npn \_math_split_final_cleanup:nn #1 #2
1417   {
1418     \exp:w \_math_split_final_cleanup:w #1
1419     \_math_split_guard:n \q_nil \s_mark { }
1420     {#2}
1421   }
1422 \cs_new:Npn \_math_split_final_cleanup:w #1 \_math_split_guard:n #2 #3 \s_mark #4
1423   {
1424     \quark_if_nil:nTF {#2}
1425     { \exp_end: { #4 #1 } }
1426     { \_math_split_final_cleanup:w #3 \s_mark { #4 #1 #2 } }
1427   }
1428
1429 \cs_new:Npn \_math_split:n #1 {
1430   \_math_split_at_nl_first:w #1 \\ \q_nil \\ \s_stop }
1431
1432 % this looks unused.
1433 %\NewDocumentCommand \splitnl { mm +m }
1434 % {
1435 %   \tl_set:Nf \l__math_tmpa_tl { \split:n {#3} }
1436 %   \show \l__math_tmpa_tl
1437 %   \exp_after:wN \_splitnl_aux:nnNN \l__math_tmpa_tl #1 #2
1438 % }

```

(End of definition for _math_split_at_nl:NN.)

\maybestartnewformulatag

```

1439
1440 \newif\if@subformulas
1441 \tl_new:N \result
1442

```

```

1443 \cs_new_protected:Npn\grabaformulapartandstart {
1444   \__math_split_at_nl:NN \g__math_grabbed_math_tl \result
1445   \typeout{====>first-result=\meaning\result}
1446   \typeout{====>first-tmpmathcontent=\meaning\g__math_grabbed_math_tl}
1447   \tl_if_empty:NTF \g__math_grabbed_math_tl
1448     {
1449       \typeout{====>formula~ has~ no~ subparts}
1450       \global\@subformulasfalse
1451     }
1452     {
1453       \typeout{====>formula~ has~ subparts}
1454       \global\@subformulastrue
1455       \edef\resulttitle{\g__math_grabbed_env_tl\space (part)}
1456       \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

```

1457   %       alt=\result,
1458   %       alt = subformula,
1459   %       title-o=\resulttitle
1460   %     }
1461   %   }
1462   % \tagmcbegin{}
1463 }
1464
1465 \cs_new_protected:Npn\grabaformulapartandmayberestart {
1466   \__math_split_at_nl:NN \g__math_grabbed_math_tl \result
1467   \typeout{====>result=\meaning\result}
1468   \typeout{====>tmpmathcontent=\meaning\g__math_grabbed_math_tl}
1469   % \tl_if_empty:NTF \g__math_grabbed_math_tl
1470   % {
1471   %   \typeout{====>tmpmathcontent=empty}
1472   % }
1473   % {
1474   %   \typeout{====>tmpmathcontent=not-empty}
1475   %   \edef\resulttitle{\g__math_grabbed_env_tl\space (part)}
1476   %   \tagstructbegin{tag=Formula,
1477   %     alt=\result,
1478   %     title-o=\resulttitle
1479   %   }
1480   % }
1481   % \tagmcbegin{}
1482 }

```

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

```

1483 \def\maybestartnewformulatag {
1484   \if@subformulas
1485   \ifmeasuring@\else
1486   %
1487   \tl_if_empty:NF \g__math_grabbed_math_tl
1488     {
1489       \tagmcbegin{}
1490       \tagstructend
1491       \grabaformulapartandmayberestart

```

```

1492     }
1493   \fi
1494 \fi
1495 }

    The breqn packages changes catcodes and that isn't yet covered by our mechanism.
1496 %\AddToHook{package/breqn/after}{
1497 % \typeout{==>~ in~ hook}
1498 % \math_register_halign_env:nn {dmath}{}
1499 % \math_register_halign_env:nn {dgroup*}{}
1500 %}

1501 \ExplSyntaxOff
1502 <@@=)
1503 </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	<code>\AtBeginDocument</code>	759
<code>\#</code>	72, 408		
<code>\%</code>	73		
<code>\(</code>	828		
<code>\(</code>	<u>1106</u>		
<code>\)</code>	829		
<code>\)</code>	<u>1106</u>		
<code>@@</code> commands:			
<code>\l_@@_collected_bool</code>	3, 9		
<code>\l_@@_content_template_tl</code>	7		
<code>\l_@@_fakemath_bool</code>	10		
<code>\[</code>	837, <u>1104</u>		
<code>\[</code>	11, 32, 39, <u>1124</u>		
<code>\</code>	272, 273, 274, 275, 276, 277, 325, 875, 879, 912, 1358, 1361, 1367, 1378, 1384, 1430		
<code>\{</code>	70		
<code>\}</code>	71		
<code>\]</code>	838, <u>1104</u>		
<code>\]</code>	11, 32, 39, <u>1124</u>		
	A		
<code>actual+source (plug)</code>	<u>578</u>		
<code>\AddToHook</code>	7, 9, 138, 226, 233, 239, 251, 309, 342, 389, 430, 447, 473, 1496		
<code>\advance</code>	1328, 1334		
<code>\aftergroup</code>	32		
<code>alt+source (plug)</code>	<u>593</u>		
<code>\AssignSocketPlug</code>	285, 286, 287, 288, 292, 387, 388, 426, 427, 428, 429		
	B		
<code>\begin</code>	34, 554, 572, 841, 881, 884, 1036, 1130, 1366, 1378, 1384, 1389, 1398, 1400		
<code>\begingroup</code>	69, 162, 177, 193, 1045, 1195		
<code>\belowdisplay</code>	41		
<code>\belowdisplayshortskip</code>	32, 1173		
<code>\belowdisplayskip</code>	32, 41, 1172		
<code>\bgroup</code>	1191		
block internal commands:			
<code>_block_beginpar_vmode:</code>	1066		
<code>_block_list_beginpar_vmode:</code>	1057		
bool commands:			
<code>\bool_gset_false:N</code>	353		
<code>\bool_gset_true:N</code>	284, 322, 349, 375, 416		
<code>\bool_if:NTF</code>	62, 282, 423, 483, 615, 625, 662, 667, 677, 684, 692, 840, 939, 973, 1006, 1108, 1129, 1136, 1146, 1160, 1177		
<code>\bool_lazy_any:nTF</code>	783		
<code>\bool_new:N</code>	10, 11, 37, 38, 39, 40, 41, 42, 321		
<code>\bool_set_false:N</code>	835, 1152, 1194, 1212		
<code>\bool_set_true:N</code>	487, 765, 952, 986, 1019, 1149, 1162, 1179, 1192, 1203, 1208		

file commands:		<code>\g__math_mathchoice_int</code>	22, 448
<code>\file_if_exist:nTF</code>	412	<code>\g__math_mathml_AF_attached_int</code>	13, 49, 442, 669
<code>\file_input:n</code>	415	<code>\g__math_mathml_AF_found_int</code>	13, 48, 440, 666
G			
<code>\gdef</code>	150	<code>\g__math_mathml_int</code>	13, 46, 59, 436
<code>\GetDocumentProperties</code>	479	<code>\g__math_mathml_total_int</code>	13, 45, 56, 434
<code>\global</code>	1327, 1328, 1333, 1334, 1345, 1346, 1450, 1454	<code>\l__math_mathstyle_int</code>	448, 455
<code>\grabaformulapartandmayberestart</code>	1465, 1491	<code>\g__math_mml_int</code>	13
<code>\grabaformulapartandstart</code>	712, 1443	<code>\g__math_mml_total_int</code>	13
group commands:		<code>\intertext</code>	10
<code>\group_begin:</code>	847	iow commands:	
<code>\c_group_begin_token</code>	855	<code>\iow_close:N</code>	313, 393
<code>\group_end:</code>	916	<code>\iow_new:N</code>	293, 378
<code>\group_insert_after:N</code>	1175	<code>\iow_newline:</code>	83, 85, 90, 92, 107, 109, 111, 113
H			
<code>\hbox</code>	1191, 1312, 1316	<code>\iow_now:Nn</code>	295, 300, 311, 363, 383, 391
hbox commands:		<code>\iow_open:Nn</code>	294, 379
<code>\hbox_set:Nn</code>	402	<code>\iow_term:n</code>	432, 433, 434, 436, 438, 440, 442
<code>\hfil</code>	159, 173, 188, 190	iow internal commands:	
<code>\hskip</code>	160, 175, 191	<code>\g__math_luamml_iow</code>	293, 294, 295, 300, 311, 313
I			
if commands:		<code>\g__math_writedummy_iow</code>	363, 376, 378, 379, 383, 391, 393
<code>\if_false:</code>	905, 911	K	
<code>\ifcase</code>	158	<code>\kern</code>	1337
<code>\iffalse</code>	1236	keys commands:	
<code>\ifnum</code>	1236	<code>\keys_define:nn</code>	328, 371, 461, 920
<code>\iftrue</code>	1170, 1293	<code>\keys_set:nn</code>	933, 967, 1000
<code>\ignorespaces</code>	898	L	
<code>\ignorespacesafterend</code>	1227, 1240, 1291	<code>\lastskip</code>	819
int commands:		<code>\LaTeXe</code>	9, 10
<code>\int_compare:nNnTF</code>	457, 877, 888, 1405	<code>\leavevmode</code>	7, 1189
<code>\int_decr:N</code>	894	legacy commands:	
<code>\int_eval:n</code>	1398, 1412	<code>\legacy_if:nTF</code>	762, 1055
<code>\int_gincr:N</code>	56, 59, 651, 666, 669	<code>\legacy_if_p:n</code>	785
<code>\int_incr:N</code>	883	<code>\legacy_if_set_false:n</code>	1056
<code>\int_new:N</code>	45, 46, 47, 48, 49, 448, 449, 844	<code>\let</code>	1195, 1196, 1293
<code>\int_set:Nn</code>	289, 455, 1174	<code>\l__tlabmathdate</code>	4
<code>\int_use:N</code>	108, 434, 436, 438, 440, 442, 450	<code>\l__tlabmathversion</code>	5
<code>\int_value:w</code>	1412	luamml commands:	
int internal commands:		<code>\luamml_flag_ignore:</code>	128, 131
<code>\g__math_AF_attached_int</code>	13	<code>\luamml_flag_process:</code>	123, 126
<code>\g__math_AF_found_int</code>	13	<code>\luamml_flag_save:nn</code>	168, 183, 199
<code>\g__math_AF_total_int</code>	13	<code>\luamml_flag_structelem:</code>	118, 121
<code>\l__math_grab_env_int</code>	33, 844, 877, 883, 888, 894	<code>\luamml_ignore:</code>	129, 131, 344
<code>\g__math_math_total_int</code>	13, 47, 108, 438, 651	<code>\luamml_process:</code>	124, 126
		<code>\luamml_structelem:</code>	119, 121, 342
		luamml internal commands:	
		<code>__luamml_array_finalize_array:</code>	148

<code>__luamml_array_finalize_col:w</code> ..	<code>__math_grab_loop\end:</code>	867
..... 170, 185, 201	<code>__math_grab_loop\ignorespaces:</code>	867
<code>__luamml_array_init_col:</code>	<code>__math_grab_loop\textonly@unskip:</code>	867
..... 166, 181, 197	867
<code>__luamml_array_save_array:</code> ...	<code>__math_grab_loop\unskip:</code> ...	867
<code>\l__luamml_pretty_int</code>	<code>__math_grab_loop_end:</code> .	874, 914, 914
<code>__luamml_register_output_hook:N</code>	<code>__math_grab_loop_group:n</code>	856, 860, 860
	856, 860, 860
	<code>__math_grab_loop_newline:</code>	878, 890, 903, 903
	878, 890, 903, 903
	<code>__math_grab_loop_store:n</code>	860, 861, 862, 871, 879, 884, 895
	860, 861, 862, 871, 879, 884, 895
	<code>__math_grab_loop_token:N</code>	857, 867, 867
	857, 867, 867
	<code>__math_luamml_activate_write:</code> ..	235, 253, 280
	235, 253, 280
	<code>__math_luamml_output_hook:n</code>	296, 308
	<code>__math_m@th:</code>	1200, 1200, 1204
	<code>__math_process:nn</code> .	760, 760, 769,
	782, 807, 950, 984, 1017, 1094, 1150	
	<code>__math_process_auxi:nn</code>	760, 766, 770
	<code>__math_process_auxii:nn</code>	760, 774, 776, 778
	760, 774, 776, 778
	<code>__math_provide_luamml_commands:</code>	116, 259, 263
	116, 259, 263
	<code>__math_split:n</code>	1408, 1429
	<code>__math_split_at_nl:NN</code>	1356, 1356, 1444, 1466
	1356, 1356, 1444, 1466
	<code>__math_split_at_nl_aux:nnNN</code> ...	1359, 1371
	1359, 1371
	<code>__math_split_at_nl_first:w</code>	1358, 1361, 1430
	1358, 1361, 1430
	<code>__math_split_check_count_-</code>	1393, 1403
	<code>begins:nnnn</code>	1393, 1403
	<code>__math_split_chk_if_begin:ww</code> ...	1366, 1377
	1366, 1377
	<code>__math_split_cleanup_begin_q_-</code>	1384, 1389
	<code>nil:w</code>	1384, 1389
	<code>__math_split_collect_one_end:w</code> .	1383, 1391, 1413
	1383, 1391, 1413
	<code>__math_split_count_begins:n</code> ...	1394, 1397
	1394, 1397
	<code>__math_split_count_begins:w</code> ...	1398, 1400, 1401
	1398, 1400, 1401
	<code>__math_split_final_cleanup:nn</code> ..	1407, 1416
	1407, 1416
	<code>__math_split_final_cleanup:w</code> ...	1418, 1422, 1426
	1418, 1422, 1426
	<code>__math_split_guard:n</code>	1408, 1419, 1422
	<code>__math_tag_dollardollar_-</code>	813, 1175
	<code>display_end:</code>	813, 1175
	<code>__math_tag_if_mathstyle:nn</code>	451, 451, 460
	451, 451, 460
	<code>\mathchoice</code>	22, 458
	22, 458

<code>\socket_new_plug:nnn</code>	99, 361, 494, 498, 502, 513, 524, 528, 533, 543, 578, 593, 611, 637, 647, 701, 709, 713, 724, 732	tag commands:	
<code>\socket_use:n</code>	517, 518, 547, 548, 682, 683, 1083	<code>\tag_if_active:TF</code>	24, 759, 1061
<code>\socket_use:nn</code>	794, 809, 1074	<code>\tag_mc_begin:n</code>	730
Sockets:		<code>\tag_mc_end:pop:n</code>	501
<code>tagsupport/math/content</code>	577	<code>\tag_mc_end:push:</code>	26, 497
<code>tagsupport/math/display/begin</code> ..	520	<code>\tag_resume:n</code>	458
<code>tagsupport/math/display/end</code> ...	520	<code>\tag_socket_use:n</code>	506, 507, 509, 511, 537, 538, 539, 541, 793, 796, 797, 808, 820
<code>tagsupport/math/display/formula/begin</code>	520	<code>\tag_struct_begin:n</code>	4, 627, 687
<code>tagsupport/math/display/formula/end</code>	520	<code>\tag_struct_end:</code>	29, 640, 705
<code>tagsupport/math/end</code>	736	<code>\tag_suspend:n</code>	458, 1293
<code>tagsupport/math/inline/begin</code> ..	490	tag internal commands:	
<code>tagsupport/math/inline/end</code> ...	490	<code>__tag_check_para_end_show:nn</code> ...	28
<code>tagsupport/math/inline/formula/begin</code>	490	<code>__tag_gincr_para_end_int:</code>	27
<code>tagsupport/math/inline/formula/end</code>	490	<code>\l__tag_math_alt_bool</code>	13, 42, 464, 483, 487, 625, 662
<code>tagsupport/math/mathml/write</code> ..	360	<code>__tag_math_disable:</code>	737, 737
<code>tagsupport/math/mathml/write/prepare</code>	98	<code>__tag_math_enable:</code> ...	748, 748, 759
<code>tagsupport/math/struct/begin</code> ..	609	<code>\g__tag_math_luamml_tl</code> ...	13, 43, 44
<code>tagsupport/math/struct/end</code>	609	<code>\g__tag_math_mathml_AF_bool</code>	13, 40, 284, 375, 416, 423
<code>tagsupport/math/substruct/begin</code> ..	707	<code>\l__tag_math_mathml_AF_bool</code>	13, 39, 468, 667, 677, 692
<code>tagsupport/math/substruct/end</code> ..	707	<code>\l__tag_math_mathml_files_clist</code> ..	13, 50, 51, 410, 463
<code>\space</code>	4, 5, 1455, 1475	<code>\l__tag_math_mathml_pane_bool</code> ...	13, 41, 62, 465
split commands:		<code>\l__tag_math_texsource_AF_bool</code> ..	13, 37, 470, 615, 684
<code>\split:n</code>	1435	<code>\l__tag_math_texsource_pane_bool</code>	13, 38, 467
<code>\splitnl</code>	1433	<code>\l__tag_para_main_tag_tl</code>	1063
splitnl internal commands:		<code>\g__tag_struct_tag_tl</code> ...	1059, 1063
<code>__splitnl_aux:nnNN</code>	1437	<code>__tag_tool_close_P:</code>	22, 22, 527, 1072
str commands:		<code>\tagmcbegin</code>	1315, 1317, 1462, 1481
<code>\c_backslash_str</code>	108	<code>\tagmcentd</code>	717, 1314, 1317, 1489
<code>\str_case:nn</code>	228	<code>\tagpdfparaOff</code>	505, 1075
<code>\str_if_eq:nnTF</code>	475, 1063	<code>\tagpdfparaOn</code>	818, 1082
<code>\str_mdfive_hash:n</code>	644, 653	<code>\tagpdfsetup</code>	33, 276, 278
<code>\str_new:N</code>	16	<code>\tagstructbegin</code>	1314, 1456, 1476
<code>\str_replace_all:Nnn</code>	102, 103	<code>\tagstructend</code>	719, 1317, 1490
<code>\str_set:Nn</code>	101	<code>tagsupport/math/content (socket)</code> ...	577
str internal commands:		<code>tagsupport/math/display/begin (socket)</code>	520
<code>\l__math_tmpa_str</code>	11, 16, 101, 102, 103, 110	<code>tagsupport/math/display/end (socket)</code> ..	520
<code>\symletters</code>	291	<code>tagsupport/math/display/formula/begin</code> (socket)	520
<code>\symsymbols</code>	290	<code>tagsupport/math/display/formula/end</code> (socket)	520
sys commands:		<code>tagsupport/math/end (socket)</code>	736
<code>\sys_if_engine luatex:TF</code>	134	<code>tagsupport/math/inline/begin (socket)</code> ..	490
<code>\c_sys_jobname_str</code>	52, 294, 381	<code>tagsupport/math/inline/end (socket)</code> .	490
T			
<code>\tabcolsep</code>	1195		

tagsupport/math/inline/formula/begin (socket)	490	<code>\@tabular</code>	1186, 1188
tagsupport/math/inline/formula/end (socket)	490	<code>\@tempcnta</code>	155
tagsupport/math/mathml/write (socket)	360	<code>\@xp</code>	1290
tagsupport/math/mathml/write/prepare (socket)	98	<code>\add@amps</code>	1335
tagsupport/math/struct/begin (socket)	609	<code>\align@</code>	1214
tagsupport/math/struct/end (socket) .	609	<code>\alignsep@</code>	1337
tagsupport/math/substruct/begin (socket)	707	<code>\ar@align@cell</code>	208
tagsupport/math/substruct/end (socket)	707	<code>\ar@mcellbox</code>	205
tbl commands:		<code>\black@</code>	1226, 1230
<code>\tbl_crcl:n</code>	142	<code>\col@sep</code>	1195
<code>\tbl_restore_outer_cell_data:</code> ..	146	<code>\common@align@ending</code>	1229, 1245, 1250, 1255, 1260, 1265, 1270, 1275, 1280, 1285
TeX and L ^A T _E X commands:		<code>\count@</code>	155
<code>\@addtopreamble</code>	157	<code>\cr</code>	33, 34
<code>\@arrayright</code>	149	<code>\d@llarbegin</code>	161, 162, 176, 177, 192, 193, 1195
<code>\@badmath</code>	1111, 1122, 1135, 1138	<code>\d@llarend</code>	164, 169, 179, 184, 195, 200, 1196
<code>\@chnum</code>	158	<code>\do@row@strut</code>	172, 187, 203, 209, 214, 219
<code>\@classx</code>	154	<code>\gather@</code>	1214
<code>\@classz</code>	152	<code>\halign</code>	34
<code>\@currenvir</code>	1040	<code>\if@eqnsw</code>	1333
<code>\@currenvline</code>	1041	<code>\if@subformulas</code>	718, 1440, 1484
<code>\@doendpe</code>	825	<code>\ifingather@</code>	1232
<code>\@eha</code>	1039	<code>\ifmeasuring@</code> .	3, 10, 1293, 1311, 1485
<code>\@endpbox</code>	207, 213, 218	<code>\ifst@rred</code>	1321, 1327, 1332, 1345
<code>\@endpefalse</code>	1046	<code>\iftag@</code>	1338
<code>\@endpetrue</code>	1085	<code>\iftagsleft@</code>	1290
<code>\@ensuredmath</code>	1147, 1151	<code>\insert@column</code>	163, 167, 178, 182, 194, 198
<code>\@eqnswtrue</code>	1327, 1345	<code>\insert@pcolumn</code>	206, 212, 217
<code>\@execute@begin@hook</code>	1042	<code>\lendmultline@</code>	1290
<code>\@ifpackageloaded</code>	247	<code>\lineht@</code>	1346
<code>\@ifundefined</code>	1038	<code>\m@ne</code>	1268, 1273, 1278, 1283
<code>\@ignorefalse</code>	1044	<code>\m@th</code> <i>9–11, 31, 41, 764, 1200,</i>	1312, 1316
<code>\@iiiparbox</code>	9	<code>\make@display@tag</code>	1323, 1339
<code>\@kernel@after@begindocument</code> .	1210	<code>\math@cr</code>	1226, 1230
<code>\@kernel@before@begindocument</code> ...	1095, 1124, 1206, 1217	<code>\maxfields@</code>	1335
<code>\@kernel@math@registered@begin</code> ..	951, 985, 1018, 1050	<code>\measuring@true</code>	1293
<code>\@kernel@math@registered@end</code> ...	1029, 1079	<code>\multline@</code>	1214
<code>\@latex@error</code>	1039	<code>\on@line</code>	1041
<code>\@lign</code>	1339	<code>\place@tag</code>	1340
<code>\@namedef</code>	1309, 1320, 1331, 1349	<code>\prepnext@tok</code>	156, 222
<code>\@ne</code> .	1253, 1258, 1268, 1273, 1328, 1334	<code>\rendmultline@</code>	1290
<code>\@nextchar</code>	205, 211, 216	<code>\reserved@a</code>	1039, 1040, 1047
<code>\@preamble</code>	150	<code>\restore@math@cr</code>	1349, 1354
<code>\@startpbox</code>	205, 211, 216	<code>\restorealignstate@</code>	1233
<code>\@subformulasfalse</code>	1450	<code>\row@</code>	1328, 1334
<code>\@subformulastrue</code>	1454	<code>\setboxz@h</code>	1339
<code>\@tabarray</code>	1198	<code>\split@tag</code>	42
		<code>\st@rredfalse</code> .	1243, 1253, 1268, 1278

use commands:		V	
\use_i:nn	478	\vbox	205, 216
\use_ii_i:nn	1411	\vcenter	9
\UseHook	1037	\vtop	211
\UseTaggingSocket	145, 1190		