

# The `latex-lab-math` code<sup>\*</sup>

Frank Mittelbach, Joseph Wright, L<sup>A</sup>T<sub>E</sub>X Project

v0.6w 2025-10-02

## 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

<b>1</b>	<b>Introduction</b>	<b>2</b>
<b>2</b>	<b>Math capture</b>	<b>3</b>
<b>3</b>	<b>Avoiding math capture</b>	<b>3</b>
3.1	Options to suppressing math capture and tagging . . . . .	4
3.1.1	Using the trigger token <code>\m@th</code> <i>inside</i> the math . . . . .	4
3.1.2	Using <code>\m@th</code> <i>before</i> the opening <code>\$</code> . . . . .	4
3.1.3	Disabling math tagging with <code>\MathCollectFalse</code> . . . . .	5
<b>4</b>	<b>Math capture interfaces</b>	<b>5</b>
4.1	Code level interfaces . . . . .	5
4.2	Document level interfaces . . . . .	5
<b>5</b>	<b>Math tagging</b>	<b>6</b>
5.1	Code requirements . . . . .	6
5.2	Inline math . . . . .	6
5.3	Display math . . . . .	7
5.4	Associated Files . . . . .	7
5.5	Automatic mathml creation with <code>luamml</code> . . . . .	8
5.6	Summary of math options . . . . .	9
<b>6</b>	<b>Debugging</b>	<b>11</b>

---

\*

<b>7</b>	<b>Known current bugs, etc.</b>	<b>11</b>
7.1	Capture/grabbing problems . . . . .	11
7.2	Fake math . . . . .	11
7.2.1	Open problems . . . . .	12
7.3	Processor . . . . .	12
7.4	Other problems . . . . .	12
7.5	Other Todos . . . . .	13
<b>8</b>	<b>The Implementation</b>	<b>13</b>
8.1	File declaration . . . . .	13
8.2	Setup . . . . .	13
8.3	Debugging . . . . .	14
8.4	Data structures . . . . .	15
8.5	Tagging tools . . . . .	15
8.6	Code related to AF . . . . .	16
8.7	Mathstyle detection . . . . .	26
8.8	Tagging options . . . . .	27
8.8.1	Meta keys . . . . .	28
8.9	Sockets . . . . .	29
8.9.1	Main inline math sockets . . . . .	29
8.9.2	Main display math sockets . . . . .	29
8.9.3	Sockets plugs for tags (labels) . . . . .	30
8.9.4	Internal sockets . . . . .	31
8.10	Interface commands . . . . .	35
8.11	Content grabbing . . . . .	36
8.12	Token-by-token inline grabbing . . . . .	38
8.13	Marking math environments . . . . .	40
8.14	Regaining control after a display has finished with <b>\$\$</b> . . . . .	43
8.15	Document commands . . . . .	46
8.16	<b>\everymath</b> and <b>\everydisplay</b> . . . . .	48
8.17	Modifying kernel environments . . . . .	49
8.18	Modifying kernel commands . . . . .	49
8.19	Disable math grabbing in the begindocument hook . . . . .	50
	<b>Index</b>	<b>51</b>

## 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,

```
\[ a \in A \text{ for all } a<5\$ \]
```

would only result in a single capture of the tokens “`a\in A\text{for all }a<5\$`”.

## 2 Math capture

In the current setup

- `$`, `\(...\)` 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.

## 3 Avoiding math capture

In most cases when an environment or command switches into math, the semantic meaning of the content *is* math and grabbing and then tagging that as a Formula is adequate.

But there are exceptions, most prominently with the math shift token `$`.

- `$` can be used to center a box with `\vcenter`, for example in the tabular code. The opening `$` is then often in a different command than the closing `$` and the content of the box should be tagged normally, including all math it contains. This means one wants to avoid that the opening `$` triggers the math grabbing and creates a Formula structure, and after the `$` one wants to switch back to normal text and math capture/tagging.
- `$` is used to place superscripts and subscripts, see the definition of `\textsuperscript` which uses `\ensuremath` internally. Inside the superscript in special cases you may want more tagging (including nested math tagging) but typically it is simple text.
- `$` is used to access a char in a math font.

For example the `\meta` command uses internally the math commands `\langle` and `\rangle` around its argument:

```
\langle\textit{argument}\rangle
```

The symbols are then clearly not math. (Beside disabling math tagging one could also use text commands like `\textlangle` and `\textrangle`: `\textlangle argument \textrangle`). Depending on the font that could even look better, but it requires that the font supports the chars, which is not the case for various OpenType fonts.) Additional tagging inside the math should be not needed. If the symbols need an actualtext then a `Span` can be added around the whole material.

### 3.1 Options to suppressing math capture and tagging

We are there providing a number of options to suppress the collection/grabbing of the math and with it the tagging. These commands can be used to control locally the tagging of math. The first two are new commands. `\m@th` is a standard L<sup>A</sup>T<sub>E</sub>X command used in a number of places to set `\mathsurround` to zero; with active tagging it is also used to identify math that should not be tagged, because it has traditionally be used in exactly the places where math mode is used for its layout characteristics and not for representing a formula.<sup>1</sup> Details are described below. (`\SuspendTagging` is documented in source2e.pdf.)

To set `\mathsurround` without disabling math tagging, use `\mathsurround\z@` directly.

#### 3.1.1 Using the trigger token `\m@th` *inside* the math

If the grabbed math contains the token `\m@th` the tagging/processing of the math is suppressed. As the math is nevertheless grabbed, this requires that the end math shift token is not hidden in some other command. `\m@th` sets also `\mathsurround` to zero, which is often wanted in untagged math anyway.

Text tagging is *not* suppressed inside the math, e.g., `\mbox{\emph{text}}` will still create an Em-structure. In contrast nested math is not tagged.

To suppress the text tagging `\SuspendTagging{}` can be used:

- no math tagging, but `\emph` is tagged:  

$$\text{\m@th\langle\mbox{\emph{if and only if}}\ $x=y$}\rangle$$$
- no internal tagging:  

$$\text{\m@th\SuspendTagging{}\langle\mbox{\emph{if and only if}}\ $x=y$}\rangle$$$

The method is quite suitable for symbols and also for sub- and superscripts (as long as they don't contain nested math that should be tagged).

#### 3.1.2 Using `\m@th` *before* the opening `$`

`\m@th` (as defined in the latex-lab-math code) sets also the internal boolean which controls the grabbing to true and so when it is used *before* some math it disables math grabbing and tagging for all following math in the current group (including nested math). Text tagging inside the math is still active, it can be disabled as above with `\SuspendTagging{}`.

When `\m@th` is used like this it is possible to reenale the math tagging of nested math, by adding `\MathCollectTrue` after the opening `$`.

- no math tagging, but `\emph` is tagged:  

$$\text{\m@th$\langle\mbox{\emph{if and only if}}\ $x=y$}\rangle$}$$
- both nested math and `\emph` is tagged:  

$$\text{\m@th$\MathCollectTrue\langle\mbox{\emph{if and only if}}\ $x=y$}\rangle$}$$

The method is suitable for symbols and sub- and superscripts too (it is actually how superscripts avoid math tagging currently). It can also be used in cases where the end math shift token is hidden in some other command. But it is not so useful for larger boxes as it sets `\mathsurround` to zero. Grouping should be used to avoid side-effects on following math.

---

<sup>1</sup>This way even code that is not adjusted up for tagging is handled correctly if it uses `\m@th`.

### 3.1.3 Disabling math tagging with `\MathCollectFalse`

Without a side-effect on `\mathsurround` the math grabbing can be suppressed and reenabled by setting the boolean that controls the collecting. So in the following example the math in the `\mbox` is properly tagged, but the angled brackets are simple text.

```
%stop math grabbing
\MathCollectFalse
$
%(optional) restart math grabbing for nested math
\MathCollectTrue
\langle\mbox{\emph{if and only if}} $x=y$\rangle
$
%(optional) restart math grabbing for following math
% if there is no grouping
\MathCollectTrue
\quad $x=1$
```

The method is suitable if a box should be centered with `\vcenter` (and is used in `array.sty` for the tabular code).

## 4 Math capture interfaces

### 4.1 Code level interfaces

---

<code>\math_register_env:n</code>	<code>\math_register_env:n {&lt;env&gt;}</code>
<code>\math_register_env:nn</code>	<code>\math_register_env:nn {&lt;env&gt;} {&lt;options&gt;}</code>

---

Registers the `<env>` 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 `<options>` may also be given:

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

---

<code>\math_processor:n</code>	<code>\math_processor:n {&lt;tokens&gt;}</code>
--------------------------------	---

---

Declares that the captured math content should be passed to the `<tokens>`, 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 `\ifmeasuring@` is true.

### 4.2 Document level interfaces

---

<code>\RegisterMathEnvironment</code>	<code>\RegisterMathEnvironment [&lt;options&gt;] {&lt;env&gt;}</code>
---------------------------------------	---

---

Registers the `<env>` 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 `<options>` may also be given:

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

---

<code>\MathCollectTrue</code>	<code>\MathCollectTrue</code>
<code>\MathCollectFalse</code>	<code>\MathCollectFalse</code>

---

This activates/deactivates the math collection of the math shift token. See above for cases when this can be usefull.

## 5 Math tagging

### 5.1 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 elements of the equation with mathml structure elements (currently only luatex with luamml)

### 5.2 Inline math

The embedding code is added through the tagging sockets

- `math/inline/begin`
- `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 tagging sockets

- `math/inline/formula/begin` This socket takes the math as second argument and its code should output it for typesetting. The `default` plug of the socket calls these three internal sockets for the tagging support:
  - `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).
  - `math/struct/begin` This calls `\tag_struct_begin:n`. It should also write the associated files if needed.

- `math/inline/formula/end` This socket ends the formula structure(s). The default plug calls this internal socket:

– `tagssupport/math/struct/end`

### 5.3 Display math

*to be written*

### 5.4 Associated Files

The current code allows the attachment of two types of associated file to the Formula structure: the L<sup>A</sup>T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X 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 5.5) 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 Lb1 substructure is ignored if an MathML is used (see [https://github.com/foxitsoftware/PDF\\_UA-2](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<sup>A</sup>T<sub>E</sub>X 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"2
```

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/mathsourcetrue` (anywhere in the document) and `viewer/pane/mathml=false` (in the preamble, before the external file is read).

## 5.5 Automatic mathml creation with luamml

If `lualatex` and the package `unicode-math` is used, the package `luamml` is loaded and this package will then 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.

---

<sup>2</sup>But it is probably not needed and only blows up the PDF.



## 5.6 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 replaces 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 (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 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  $\text{\LaTeX}$ -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/load}`. 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<sup>A</sup>T<sub>E</sub>X 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<sup>A</sup>T<sub>E</sub>X 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 future 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.

## 6 Debugging

---

`\DebugMathOn`  
`\DebugMathOff`  
`\math_debug_on:`  
`\math_debug_off:`

---

These commands enable/disable debugging messages.

## 7 Known current bugs, etc.

### 7.1 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. Therefore in a tabular a slower token-by-token grabbing is used.

### 7.2 Fake math

For the current state see 3 above. The text here is mainly kept for history.

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 `\@iiiparbox` where the code from `$_vcenter` to `\m@th$` is grabbed and put back. It does not work for example for `tabular` where the dollars and the `\m@th` token are spread around over three commands. `tabular` needs therefore manual intervention. A look in the list of usages (in `usage-of-m@th.md`) justifies this approach. All usages are either not math at all, or related to small elements that probably shouldn't be grabbed and processed on their own.
- `\m@th` is redefined so that it sets the boolean `\l_@@_collected_bool` to true. If `\m@th` is used inside math that has been grabbed this doesn't change much as the boolean is set by the grabbing anyway. For usages outside math the benefit is not so clear: The setting avoids that in `\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 reenact math processing inside some (fake) math one has to do it after `\m@th` calls.

### 7.2.1 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.

## 7.3 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.

## 7.4 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.

## 7.5 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.

## 8 The Implementation

1 `<*kernel>`

### 8.1 File declaration

```

2 \ProvidesFile{latex-lab-math.ltx}
3   [\ltxlabmathdate\space
4     v\ltxlabmathversion\space
5     Grab all the math(s) and tag it (experiments)]

   Temp loading ...
6 \AddToHook{begindocument/before}{\RequirePackage{latex-lab-testphase-block}}

7 <@@=math>

8 \ExplSyntaxOn
```

### 8.2 Setup

Loading `amsmath` is an absolute requirement: this avoids needing to have conditional definitions and deals with how to define `\[/\]` neatly. The package is loaded at begin document to allow user to load it with options.

```

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

### 8.3 Debugging

---

```

\g__math_debug_bool
10 \bool_new:N \g__math_debug_bool

__math_debug:n
__math_debug_typeout:n
11 \cs_new_eq:NN \__math_debug:n \use_none:n
12 \cs_new_eq:NN \__math_debug_typeout:n \use_none:n

(End of definition for \__math_debug:n and \__math_debug_typeout:n.)

\math_debug_on:
\math_debug_off:
__math_debug_gset:13 \cs_new_protected:Npn \math_debug_on:
14 {
15   \bool_gset_true:N \g__math_debug_bool
16   \__math_debug_gset:
17 }

18 \cs_new_protected:Npn \math_debug_off:
19 {
20   \bool_gset_false:N \g__math_debug_bool
21   \__math_debug_gset:
22 }

23 \cs_new_protected:Npn \__math_debug_gset:
24 {
25   \cs_gset_protected:Npe \__math_debug:n ##1
26   { \bool_if:NT \g__math_debug_bool {##1} }
27   \cs_gset_protected:Npe \__math_debug_typeout:n ##1
28   { \bool_if:NT \g__math_debug_bool { \typeout{[Math]~ ==>~ ##1} } }
29 }

(End of definition for \math_debug_on:, \math_debug_off:, and \__math_debug_gset:. These functions
are documented on page 11.)

\DebugMathOn If we are debugging blocks we also want to know about template instances, so we turn
\DebugMathOff the debugging for templates as well (for now).

30 \cs_new_protected:Npn \DebugMathOn { \math_debug_on: }
31 \cs_new_protected:Npn \DebugMathOff { \math_debug_off: }

32 \DebugMathOff

(End of definition for \DebugMathOn and \DebugMathOff. These functions are documented on page 11.)

```

## 8.4 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.

33 `\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.

34 `\bool_new:N \l__math_fakemath_bool`

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

Or do we need an

extra state 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.

35 `\tl_new:N \g__math_grabbed_env_tl`

36 `\tl_new:N \g__math_grabbed_math_tl`

---

`\l__math_tmpa_tl` Temporary variables

`\l__math_tmpa_skip`

`\l__math_tmpa_str` 37 `\tl_new:N \l__math_tmpa_tl`

38 `\skip_new:N \l__math_tmpa_skip`

39 `\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 stored as AF.

`\l__math_content_actual_tl`

`\l__math_content_AF_tl`

40 `\tl_new:N \l__math_content_alt_tl`

41 `\tl_new:N \l__math_content_actual_tl`

42 `\tl_new:N \l__math_content_AF_source_tl`

43 `\tl_new:N \l__math_content_AF_source_tmpa_tl`

44 `\tl_new:N \l__math_content_AF_mathml_tl`

## 8.5 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.

```

45 \cs_new_protected:Npn \__tag_tool_close_P:
46 {
47   \tag_if_active:T
48   {
49     \tag_mc_end: %end P-chunk, should perhaps be \tag_mc_end_push: ...
50     \__tag_gincr_para_end_int:
51     \__tag_check_para_end_show:nn{red}{} %debug: show para
52     \tag_struct_end:
53   }
54 }

```

(End of definition for \\_\_tag\_tool\_close\_P:.)

We add also an attribute.

```

55 \tl_new:N\l__math_attribute_class_tl
56 \tagpdfsetup
57   {role/new-attribute = {inline}      {/0 /Layout /Placement/Inline},
58   role/new-attribute = {display}     {/0 /Layout /Placement/Block},
59   }

```

## 8.6 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.

```

60 \bool_new:N\l__tag_math_texsource_AF_bool
61 \bool_new:N\l__tag_math_texsource_pane_bool
62 \bool_new:N\l__tag_math_mathml_AF_bool
63 \bool_new:N\g__tag_math_mathml_AF_bool
64 \bool_new:N\l__tag_math_mathml_pane_bool
65 \bool_new:N\l__tag_math_alt_bool
66 \tl_new:N\g__tag_math_luamml_tl
67 \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).

```

68 \int_new:N\g__math_mathml_total_int
69 \int_new:N\g__math_mathml_int
70 \int_new:N\g__math_math_total_int
71 \int_new:N\g__math_mathml_AF_found_int
72 \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.

```

73 \clist_new:N\l__tag_math_mathml_files_clist
74 \clist_put_right:Ne\l__tag_math_mathml_files_clist
75   {\c_sys_jobname_str-mathml,\c_sys_jobname_str-luamml-mathml}

```

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

```

\__math_AF_mml:nnnn

```

```

76 \cs_new_protected:Npn \__math_AF_mml:nnnn #1 #2 #3 #4
77   {%#1 number, #2 tex source for debugging, #3 hash, #4 mathml
78    {
79      \int_gincr:N \g__math_mathml_total_int

```

mathml with the same hash should be included only once:

```

80     \tl_if_exist:cF { g__math_mathml_#3_tl }
81     {
82       \int_gincr:N \g__math_mathml_int

```

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

```

83       \pdfdict_put:nne {l_pdffile/Filespec} {Desc}{(mathml-#1)}
84       \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

```

85       \bool_if:NT \l__tag_math_mathml_pane_bool
86       {\pdfmanagement_add:nne {Catalog/Names}{EmbeddedFiles}{\l__math_tmpa_tl}}
87       \tl_new:c{g__math_mathml_#3_tl}

```

```

88     \tl_gset_eq:cN{g__math_mathml_#3_tl}\l__math_tmpa_tl
89   }
90 }

```

(End of definition for \\_\_math\_AF\_mml:nnnn.)

The html reader.

```

91 \cs_new_protected:Npn \__math_AF_html_reader:w#1</h2>#2<p>#3</p>#4<p>#5</p>#6<math{
92   \begin{group}
93     \char_set_catcode_other:N\{
94     \char_set_catcode_other:N\}
95     \char_set_catcode_other:N\#
96     \char_set_catcode_other:N\%
97     \char_set_catcode_other:N\^
98     \__math_AF_html_reader_verb:w{#1}{#3}{#5}<math
99   }

100 \cs_new_protected:Npn \__math_AF_html_reader_verb:w#1#2#3#4~</div>{
101   \endgroup
102   \__math_AF_mml:nnnn{#1}{#2}{#3}{#4}
103 }

```

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
106   <!DOCTYPE~html>
107   \iow_newline:
108   <html~ xmlns="http://www.w3.org/1999/xhtml">
109   \iow_newline:
110   }
111 \tl_new:N \l__math_mathml_write_before_tl
112 \tl_const:Nn \c__math_mathml_write_after_tl
113   {
114     \iow_newline:
115     </div>
116     \iow_newline:
117   }
118 \tl_const:Nn \c__math_mathml_write_final_tl
119   {
120     </html>
121   }

```

(End of definition for \c\_\_math\_mathml\_write\_init\_tl and others.)

**mathml/write/prepare (tag 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.

```

122 \NewTaggingSocket{math/mathml/write/prepare}{0}

```

**On (plug)**

```

123 \NewTaggingSocketPlug{math/mathml/write/prepare}{0n}
124 {
125   \str_set:NV\l__math_tmpa_str\l__math_content_AF_source_tl
126   \str_replace_all:Nnn\l__math_tmpa_str{&}{&#;}
127   \str_replace_all:Nnn\l__math_tmpa_str{<}{&lt;}
128   \tl_set:Nn \l__math_mathml_write_before_tl
129     {
130       <div>
131       \iow_newline:
132       <h2>\c_backslash_str mml\c_space_tl \int_use:N \g__math_math_total_int </h2>
133       \iow_newline:
134       <p>\l__math_tmpa_str</p>
135       \iow_newline:
136       <p>\l__math_content_hash_tl </p>
137       \iow_newline:
138     }
139 }

```

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

```

140 \cs_new_protected:Npn\__math_provide_luamml_commands:
141 {
142   \providecommand\luamml_flag_structelem:{}
143   \cs_if_free:NT \luamml_structelem:
144   {
145     \cs_set_eq:NN\luamml_structelem:\luamml_flag_structelem:
146   }
147   \providecommand\luamml_flag_process:{}
148   \cs_if_free:NT \luamml_process:
149   {
150     \cs_set_eq:NN\luamml_process:\luamml_flag_process:
151   }
152   \providecommand\luamml_flag_ignore:{}
153   \cs_if_free:NT \luamml_ignore:
154   {
155     \cs_set_eq:NN\luamml_ignore:\luamml_flag_ignore:
156   }
157 }

158 \sys_if_engine luatex:TF
159 {
160   \AddToHook{begindocument/before}
161   {
162     \str_case:on \g__math_luamml_load_tl
163     {
164       { 1 } {
165         \RequirePackage { luamml }
166         \AddToHook{begindocument/end}
167         {
168           \__math_luamml_activate_write:

```

```

169         }
170     }
171     {-1 } {
172         \AddToHook{begindocument/end}
173         {
174             \msg_note:nnnn { tag }
175             { luamml-status }{ disabled }{ not~create }
176         }
177     }
178     { 0 }
179     {
180         \@ifpackageloaded { unicode-math }
181         {
182             \RequirePackage { luamml }
183             \AddToHook{begindocument/end}
184             {
185                 \__math_luamml_activate_write:
186             }
187         }
188         { \msg_warning:nn { tag }{ unicode-math-missing } }
189     }
190 }
191 \__math_provide_luamml_commands:
192 }
193 }
194 {
195     \AddToHook{begindocument/before}
196     {
197         \__math_provide_luamml_commands:
198     }
199 }
200 \msg_new:nnn { tag }{ luamml-status }
201 {
202     luamml~has~been~#1~and~will~#2~an~MathML~file.
203 }
204
205 \msg_new:nnn { tag }{ unicode-math-missing }
206 {
207     The~package~unicode-math~is~missing\\
208     luamml~will~not~create~an~MathML~file.\\
209     To~avoid~this~warning~load~unicode-math~\\
210     or~disable~luamml~with~\\
211     \tl_to_str:n{\tagpdfsetup{math/mathml/luamml/load=false}}\\
212     or~force~luamml~with~\\
213     \tl_to_str:n{\tagpdfsetup{math/mathml/luamml/load=true}}
214 }
215 \cs_new_protected:Npn \__math_luamml_activate_write:
216 {
217     \bool_if:NT \g__math_luamml_write_bool
218     {

```

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

```

219     \bool_gset_true:N\g__tag_math_mathml_AF_bool
220     \AssignTaggingSocketPlug{math/struct/begin}{mathml-AF}

```

```

221 \AssignTaggingSocketPlug{math/struct/end}{mathml-AF}
222 \int_set:Nn \l__luamml_pretty_int { 7 }
223 \RegisterFamilyMapping\symsymbols{oms}
224 \RegisterFamilyMapping\symletters{oml}
225 \AssignTaggingSocketPlug{math/mathml/write/prepare}{On}
226 \iow_new:N \g__math_luamml_iow
227 \iow_open:Nn \g__math_luamml_iow {\c_sys_jobname_str-luamml-mathml.html}
228 \iow_now:Ne \g__math_luamml_iow { \c__math_mathml_write_init_tl }
229 \cs_new:Npn \__math_luamml_output_hook:n ##1
230 {
231     \tl_if_empty:NF \l__math_mathml_write_before_tl
232     {

```

We check here if the current group level is equal to the one stored for the outer math. We only write output if that is the case.

Currently in L<sup>A</sup>T<sub>E</sub>X, the `\math@level` is increased for every nested math mode (via `\frozen@everymath`). However, to make the code below work correctly we undo that in the case of “fake math”, i.e., in math mode that is only entered to make use of super or subscript positioning of text or for `\vcentering` text, i.e., if it is not really a “math formula”. We therefore provide a special declaration, `\UseMathForPositioningText`, to indicate that the directly following `$` represents such “fake math”.

```

233 \int_compare:nNnT
234 { \math@level } = { 1 }
235 {
236     \iow_now:Ne \g__math_luamml_iow
237     {
238         \l__math_mathml_write_before_tl
239         ##1
240         \c__math_mathml_write_after_tl
241     }
242 }
243 }
244 }
245 \__luamml_register_output_hook:N \__math_luamml_output_hook:n

```

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

```

246 \AddToHook{enddocument/afterlastpage}
247 {
248     \iow_now:Ne \g__math_luamml_iow
249     { \c__math_mathml_write_final_tl }
250     \iow_close:N \g__math_luamml_iow
251 }
252 \msg_note:nnnn { tag }
253 { luamml-status }{ enabled }{ create }
254 }
255 }

```

#### `\UseMathForPositioningText`

The `\UseMathForPositioningText` command indicates that a directly following `$` is not a real math formula, but that the math mode is only entered to position ordinary text with the help of built in T<sub>E</sub>X algorithms normally used for math, e.g., `\vcenter`, super and subscripts.

Signalling that special usage is necessary in the case tagged PDF is produced, because then such math mode should not generate a “formula” structure.

The command requires that it is immediately followed by  $\$$ ; anything else will result in a low-level  $\text{\TeX}$  error. As it is not a user but a developer command, this seems acceptable for the sake of fast runtime execution.

The way it is implemented it can be used in legacy code in front of any  $\$$  that switches to math mode for non-math purposes. If tagging is active it will ensure that this particular math mode content is not treated as math with respect to tagging (though nested math still is). If tagging is inactive the command is simply ignored.

```
256 \cs_set_protected:Npn \UseMathForPositioningText $ {
```

Before the math mode is started we ensure that its content is not collected by the grabber.

```
257   \bool_if:NTF \l__math_collected_bool
258     { $ }
259     {
260       \bool_set_true:N \l__math_collected_bool
261       $
```

Once inside math mode we reenables grabbing, so that any nested math (within text) is grabbed.

```
262       \bool_set_false:N \l__math_collected_bool
```

In addition we decrement the math level (which was automatically incremented when entering math mode) so that this math mode is transparent in `\__math_luamml_activate_write:`.

```
263       \int_decr:N \@math@level
264     }
265 }
```

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

`\@textsuperscript` Updates to the kernel macros.  
`\@textsubscript`

```
266 \def\@textsuperscript#1{%
267   {\m@th\@unreal@math{\mbox{\fontsize\sf@size\sf@size#1}}}}
```

For the math subscript we have to use `\sb` because this file is processed with `\ExplSyntaxOn`. Alternatively, we could have used `\c_math_subscript_token`, but that looks odd as long as the rest is in 2e style):

```
268 \def\@textsubscript#1{%
269   {\m@th\@unreal@math{\sb{\mbox{\fontsize\sf@size\sf@size#1}}}}}
```

*(End of definition for \@textsuperscript and \@textsubscript. These functions are documented on page ??.)*

`\@unreal@math`  
`\@ensured@unreal@math`

```

270 \protected\def\@unreal@math{%
271   \ifmmode
272     \expandafter\@firstofone
273   \else
274     \expandafter\@ensured@unreal@math
275   \fi}

```

If we are not in math mode we start one. Because of `\m@th` outside this will not be collected, but inside we want to collect again in case the argument contains nested math.

```

276 \long\def\@ensured@unreal@math#1{
277   $
278   \bool_set_false:N \l__math_collected_bool

```

We also decrement the math level so that any inner math is subject to MathML handling if appropriate.

```

279   \int_decr:N \@math@level
280   #1 $
281 }

```

*(End of definition for \@unreal@math and \@ensured@unreal@math. These functions are documented on page ??.)*

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.

```

282 \tl_new:N \g__math_luamml_load_tl
283 \tl_gset:Nn \g__math_luamml_load_tl {}

```

---

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

```

284 \bool_new:N \g__math_luamml_write_bool
285 \bool_gset_true:N \g__math_luamml_write_bool

```

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

```

286 \cs_new:Npn\__math_luamml_structelem:{}
287 \cs_new:Npn\__math_luamml_ignore:{}

```

*(End of definition for \\_\_math\_luamml\_ignore: and \\_\_math\_luamml\_structelem:.)*

```

288 \msg_new:nnn { tag }{ PDF-2.0-recommended }
289 {
290   The~key~#1~will~not~work~properly~with~PDF~#2.\\
291   Switching~to~PDF~2.0~is~recommended.
292 }
293 \keys_define:nn { __tag / setup }
294 {

```

At first a key to suppress the loading altogether

```

295     math/mathml/luamml/load .choice: ,
296     math/mathml/luamml/load/true .code:n = {\tl_gset:Nn \g__math_luamml_load_tl{1}},
297     math/mathml/luamml/load/false .code:n = {\tl_gset:Nn \g__math_luamml_load_tl{-
1}},
298     math/mathml/luamml/load .default:n = true,
299     math/mathml/luamml/load .usage:n=preamble,

```

A key to activate math structure elements.

```

300     math/mathml/structelem .choice:,
301     math/mathml/structelem/true .code:n =
302     {
303         \pdf_version_compare:NnT < {2.0}
304         {
305             \msg_warning:nnne { tag }{ PDF-2.0-recommended }
306             { math/mathml/structelem }{ \pdf_version: }
307         }
308         \cs_set:Npn\__math_luamml_structelem:{\luamml_structelem:}
309         \cs_set:Npn\__math_luamml_ignore:{\luamml_ignore:}
310     },
311     math/mathml/structelem/false .code:n =
312     {
313         \cs_set_eq:NN\__math_luamml_structelem:\prg_do_nothing:
314         \cs_set_eq:NN\__math_luamml_ignore:\prg_do_nothing:
315     },
316     math/mathml/structelem .default:n = true,

```

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

```

317     math/mathml/ignore .code:n = {\luamml_ignore:},

318     math/mathml/luamml/write .choice:,
319     math/mathml/luamml/write/true .code:n =
320     {
321         \tl_gset:Nn \g__math_luamml_load_tl{1}
322         \bool_gset_true:N \g__math_luamml_write_bool
323     },
324     math/mathml/luamml/write/false .code:n =
325     {
326         \bool_gset_false:N \g__math_luamml_write_bool
327     },
328     math/mathml/luamml/write .default:n = true,
329     math/mathml/luamml/write .usage:n=preamble,

```

alias keys for compatibility

```

330     math/mathml/luamml .bool_gset:N = \g__math_luamml_write_bool,
331     math/mathml/luamml .usage:n=preamble
332 }

```

`math/mathml/write` (*tag 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

```

333 \NewTaggingSocket{math/mathml/write}{0}

```



On (*plug*)

```
334 \NewTaggingSocketPlug{math/mathml/write}{On}
335 {
336   \iow_now:Ne \g__math_writedummy_iow
337   {
338     \l__math_mathml_write_before_tl
339     <math~ xmlns="http://www.w3.org/1998/Math/MathML"></math>
340     \c__math_mathml_write_after_tl
341   }
342 }
```

And now a key to activate the socket.

```
343 \keys_define:nn { __tag / setup }
344 {
345   math/mathml/write-dummy .code:n =
346   {
347     \bool_gset_true:N \g__tag_math_mathml_AF_bool
348     \tl_if_exist:NF\g__math_writedummy_iow
349     {
350       \iow_new:N \g__math_writedummy_iow
351       \iow_open:Nn \g__math_writedummy_iow
352       {
353         \c_sys_jobname_str-mathml-dummy.html
354       }
355       \iow_now:Ne \g__math_writedummy_iow
356       {
357         \c__math_mathml_write_init_tl
358       }
359       \AssignTaggingSocketPlug{math/mathml/write/prepare}{On}
360       \AssignTaggingSocketPlug{math/mathml/write}{On}
361       \AddToHook{enddocument/afterlastpage}
362       {
363         \iow_now:Ne \g__math_writedummy_iow
364         { \c__math_mathml_write_final_tl }
365         \iow_close:N \g__math_writedummy_iow
366       }
367     }
368   },
369   math/mathml/write-dummy .usage:n=preamble
370 }
```

\\_\_math\_AF\_process\_mathml\_files:

```
371 \box_new:N\l__math_tmpa_box
372 \cs_new_protected:Npn \__math_AF_process_mathml_files:
373 {
374   \hbox_set:Nn \l__math_tmpa_box
375   {
376     \pdfdict_put:nnn { l_pdffile/Filespec }{AFRelationship} { /Supplement }
377     \pdfdict_put:nne
378     { l_pdffile }{Subtype}
379     { \pdf_name_from_unicode_e:n{application/mathml+xml} } }
```

```

380 \char_set_catcode_other:N \#
381 \cs_set_eq:NN\mml \__math_AF_html_reader:w
382 \clist_map_inline:Nn \l__tag_math_mathml_files_clist
383 {
384   \file_if_exist:nTF {##1.html}
385   {
386     \typeout{Info:~reading~mathml~file~##1}
387     \file_input:n {##1.html}
388     \bool_gset_true:N\g__tag_math_mathml_AF_bool
389   }
390   {
391     \typeout{Info:~mathml~file~##1~does~not~exist}%info message
392   }
393 }
394 }
395 \box_clear:N \l__math_tmpa_box
396 \bool_if:NT\g__tag_math_mathml_AF_bool
397 {
398   \typeout{Info:~Activating~mathml~support}
399   \AssignTaggingSocketPlug{math/struct/begin}{mathml-AF}
400   \AssignTaggingSocketPlug{math/struct/end}{mathml-AF}
401   \AddToHook{enddocument/info}
402   {
403     \iow_term:n{MathML~statistic}
404     \iow_term:n{=====}
405     \iow_term:e{==>~\int_use:N\g__math_mathml_total_int\c_space_tl
406     MathML~fragments~read}
407     \iow_term:e{==>~\int_use:N\g__math_mathml_int\c_space_tl
408     different~MathML~fragments}
409     \iow_term:e{==>~\int_use:N\g__math_math_total_int\c_space_tl
410     math~fragments~found}
411     \iow_term:e{==>~\int_use:N\g__math_mathml_AF_found_int\c_space_tl
412     fitting~MathML~AF~found}
413     \iow_term:e{==>~\int_use:N\g__math_mathml_AF_attached_int\c_space_tl
414     MathML~AF~attached}
415   }
416 }
417 }
418 \AddToHook{begindocument}{\__math_AF_process_mathml_files:}

```

(End of definition for \\_\_math\_AF\_process\_mathml\_files:.)

## 8.7 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\int_new:N \l__math_mathstyle_int
\int_new:N \g__math_mathchoice_int
\property_new:nnnn{mathstyle}{now}{-1}{\int_use:N \l__math_mathstyle_int }

```

(End of definition for `\l__math_mathstyle_int`, `\g__math_mathchoice_int`, and `mathstyle`.)

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`

```

422 \cs_new_protected:Npn \__math_tag_if_mathstyle:nn #1 #2
423   {%#1 refers to label
424   {%#2 is a number for the mathstyle (typically 0,2,4,6)
425   {
426     \int_set:Nn \l__math_mathstyle_int {#2}
427     \property_record:nn {#1} { mathstyle }
428     \int_compare:nNnTF { \property_ref:nn {#1}{ mathstyle} } = { #2 }
429     { \tag_resume:n{\mathchoice} } { \tag_suspend:n{\mathchoice} }
430   }
431 \cs_generate_variant:Nn \__math_tag_if_mathstyle:nn {en}

```

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

## 8.8 Tagging options

```

432 \keys_define:nn { __tag / setup }
433 {
434   math/mathml/sources .clist_set:N = \l__tag_math_mathml_files_clist,
435   math/alt/use .bool_set:N = \l__tag_math_alt_bool,
436   viewer/pane/mathml .bool_set:N = \l__tag_math_mathml_pane_bool,
437   viewer/pane/mathml .initial:n = true,
438   viewer/pane/mathsource .bool_set:N = \l__tag_math_texsource_pane_bool,
439   math/mathml/AF .bool_set:N = \l__tag_math_mathml_AF_bool,
440   math/mathml/AF .initial:n = true,
441   math/tex/AF .bool_set:N = \l__tag_math_texsource_AF_bool,
442   math/tex/AF .initial:n = true
443 }

```

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

```

444 \AddToHook{begindocument/end}
445 {
446   \str_if_eq:eeT
447   {1}
448   {
449     \exp_last_unbraced:Ne\use_i:nn
450     {\GetDocumentProperties{document/pdfstandard-UA}}
451     \c_empty_tl\c_empty_tl
452   }
453   {
454     \bool_if:NF \l__tag_math_alt_bool
455     {
456       \typeout{PDF/UA-1-detected.~Enabling~alt~text~on~Formula}
457     }
458     \bool_set_true:N\l__tag_math_alt_bool
459   }

```

```
460 }
```

### 8.8.1 Meta keys

The `math/setup` key accepts a list with the values `mathml-SE`, `mathml-AF` and `tex-AF`. It is a fast way to set the main option. It at first disables them all, to get a clean state.

```
461 \keys_define:nn {__tag / setup}
462 {
463   math/setup .code:n =
464   {
465     %deactivate loading of luamml
466     \tl_gset:Nn \g__math_luamml_load_tl{-1}
467     \keys_set:nn {__tag / setup}
468     {
469       %deactivate tex source AF
470       math/tex/AF = false,
471       %deactivate reading of mathml-AF
472       math/mathml/sources=,
473       math/mathml/AF=false,
474       %deactivate structelem
475       math/mathml/structelem=false,
476       %handle value
477     }
478     \clist_map_inline:nn { #1}
479     {
480       \keys_set:nn {__tag / setup}{math/__setup/##1}
481     }
482   },
483   math/__setup / mathml-SE .code:n =
484   {
485     \tl_gset:Nn \g__math_luamml_load_tl{1}
486     \keys_set:nn {__tag / setup}
487     {
488       math/mathml/structelem=true
489     }
490   },
491   math/__setup / mathml-AF .code:n =
492   {
493     \tl_gset:Nn \g__math_luamml_load_tl{1}
494     \clist_put_right:Nn \l__tag_math_mathml_files_clist
495     {\c_sys_jobname_str-mathml,\c_sys_jobname_str-luamml-mathml}
496     \keys_set:nn {__tag / setup}
497     {
498       math/mathml/AF=true
499     }
500   },
501   math/__setup / tex-AF .code:n =
502   {
503     \keys_set:nn {__tag / setup}
504     {
505       math/tex/AF =true
506     }
507   },
508 }
```

## 8.9 Sockets

### 8.9.1 Main inline math sockets

`math/inline/begin` (*tag socket*) These sockets are already declared in `ltagging` and only documented here. The first two sockets are meant to embed inline math into the surrounding (so to close/reopen, e.g., MC-chunks). 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!

```
509 %\NewTaggingSocket{math/inline/begin}{0}
510 %\NewTaggingSocket{math/inline/end}{0}
511 %\NewTaggingSocket{math/inline/formula/begin}{2} %
512 %\NewTaggingSocket{math/inline/formula/end}{0}
```

MC (*plug*)

```
513 \NewTaggingSocketPlug
514   {math/inline/begin}
515   {MC}
516   {\tag_mc_end_push:}
517 \NewTaggingSocketPlug
518   {math/inline/end}
519   {MC}
520   {\tag_mc_begin_pop:n{}}
```

We probably will want to test different tagging recipes.

default (*plug*)

```
521 \NewTaggingSocketPlug
522   {math/inline/formula/begin}
523   {default}

524   { \tagpdfparaOff
525     \_math_luaaml_structelem:
526     \tag_socket_use:n{math/content}
527     \tag_socket_use:n{math/struct/begin}
528     #2
529     \tag_socket_use:n{math/end}
530   }
531 \NewTaggingSocketPlug
532   {math/inline/formula/end}
533   {default}
534   {
535     \tag_socket_use:n{math/struct/end}
536   }
```

### 8.9.2 Main display math sockets

`math/display/begin` (*tag socket*) These sockets are already declared in `ltagging` and only documented here. 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!

```

537 %\NewTaggingSocket{math/display/begin}{0}
538 %\NewTaggingSocket{math/display/end}{0}
539 %\NewTaggingSocket{math/display/formula/begin}{2} %
540 %\NewTaggingSocket{math/display/formula/end}{0}

```

default (*plug*)

```

541 \NewTaggingSocketPlug
542 {math/display/begin}
543 {default}
544 { \_tag_tool_close_P: }
545 \NewTaggingSocketPlug
546 {math/display/end}
547 {default}
548 {
549 }

```

default (*plug*)

```

550 \NewTaggingSocketPlug
551 {math/display/formula/begin}
552 {default}
553 {
554   \tagpdfparaOff
555   \_math_luamml_structelem:
556   \tag_socket_use:n{math/content}
557   \tag_socket_use:n{math/struct/begin}
558   #2
559   \tag_socket_use:n{math/end}
560 }
561 \NewTaggingSocketPlug
562 {math/display/formula/end}
563 {default}
564 {
565   \tag_socket_use:n{math/struct/end}
566 }

```

### 8.9.3 Sockets plugs for tags (labels)

$\mathit{h}/\mathit{display}/\mathit{tag}/\mathit{begin}$  (*tag socket*) These sockets are already declared in `ltagging` and only documented here. These sockets  
 $\mathit{ath}/\mathit{display}/\mathit{tag}/\mathit{end}$  (*tag socket*) are used in `\maketag__math@` to tag labels as `Lbl`. `luamml` changes the plug to move the  
`Lbl` into the `math` structure with an intent.

```

567 %\NewTaggingSocket{math/display/tag/begin}{0}
568 %\NewTaggingSocket{math/display/tag/end}{0}

```

default (*plug*)

```

569 \NewTaggingSocketPlug
570 {math/display/tag/begin}
571 {default}
572 {
573   \tag_mc_end:

```

```

574     \tag_struct_begin:n {tag=Lbl}
575     \tag_mc_begin:n {}
576   }
577   \NewTaggingSocketPlug
578     {math/display/tag/end}
579     {default}
580   {
581     \tag_mc_end:
582     \tag_struct_end:
583     \tag_mc_begin:n{ }
584   }
585   \AssignTaggingSocketPlug{math/display/tag/begin}{default}
586   \AssignTaggingSocketPlug{math/display/tag/end}{default}

```

#### 8.9.4 Internal sockets

---

##### \l\_\_math\_content\_template\_tl

The default text used as alt or actual text.

```

587 \tl_new:N\l__math_content_template_tl
588 \tl_set:Nn \l__math_content_template_tl
589   {
590     LaTeX~ formula~ starts~
591     \exp_not:N\begin{\g__math_grabbed_env_tl}
592     \c_space_tl
593     \exp_not:V\g__math_grabbed_math_tl
594     \c_space_tl
595     \exp_not:N\end{\g__math_grabbed_env_tl}
596     \c_space_tl LaTeX~ formula~ ends~
597   }

```

---

##### \l\_\_math\_texsource\_template\_tl

The default text used as texsource

```

598 \tl_new:N\l__math_texsource_template_tl
599 \tl_const:Nn\c__math_inline_env_tl {math}
600 \tl_set:Nn \l__math_texsource_template_tl
601   {
602     \tl_if_eq:NNTF\g__math_grabbed_env_tl\c__math_inline_env_tl
603     {
604       $
605       \exp_not:V\g__math_grabbed_math_tl
606       $
607     }
608     {
609       \exp_not:N\begin{\g__math_grabbed_env_tl}
610       \exp_not:V\g__math_grabbed_math_tl
611       \exp_not:N\end{\g__math_grabbed_env_tl}
612     }
613   }

```

`math/content` (*tag 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`

```
614 \NewTaggingSocket{math/content}{0}
```

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

`actual+source` (*plug*)

```
615 \NewTaggingSocketPlug
616   {math/content}
617   {actual+source}
618   {
619     \tl_set:N\l__math_content_actual_tl
620     {
621       \l__math_content_template_tl
622     }
623     \tl_set:N\l__math_content_AF_source_tl
624     {
625       \l__math_texsource_template_tl
626     }
627     \tl_set:Nn \l__math_content_AF_mathml_tl {}
628     \tl_set:Nn \l__math_content_alt_tl {}
629   }
```

`alt+source` (*plug*)

```
630 \NewTaggingSocketPlug
631   {math/content}
632   {alt+source}
633   {
634     \tl_set:N\l__math_content_alt_tl
635     {
636       \l__math_content_template_tl
637     }
638     \tl_set:N\l__math_content_AF_source_tl
639     {
640       \l__math_texsource_template_tl
641     }
642     \tl_set:Nn \l__math_content_AF_mathml_tl {}
643     \tl_set:Nn \l__math_content_actual_tl {}
644   }
```

```
645 \AssignTaggingSocketPlug{math/content}{alt+source}
```

`math/struct/begin` (*tag socket*) For the main structure we use a socket too. This allows, e.g., to create a special one `math/struct/end` (*tag 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`

```
646 \NewTaggingSocket{math/struct/begin}{0}
647 \NewTaggingSocket{math/struct/end}{0}
```



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

```

648 \NewTaggingSocketPlug
649 {math/struct/begin}
650 {default}
651 {
652   \bool_if:NTF\l__tag_math_texsource_AF_bool
653   { \tl_set_eq:NN \l__math_content_AF_source_tmpa_tl \l__math_content_AF_source_tl }
654   { \tl_clear:N \l__math_content_AF_source_tmpa_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{} }
664   \tag_struct_begin:n
665   {
666     tag=Formula,
667     attribute-class=\l__math_attribute_class_tl,
668     texsource      = \l__math_content_AF_source_tmpa_tl,
669     title-o        = \g__math_grabbed_env_tl,
670     actualtext     = \l__math_content_actual_tl,
671     alt            = \l__math_content_alt_tl
672   }

673   \__math_debug_typeout:n{grabbed-math=\meaning\g__math_grabbed_math_tl}
674   \tag_mc_begin:n{}
675 }
676 \NewTaggingSocketPlug
677 {math/struct/end}
678 {default}
679 { \tag_mc_end: \tag_struct_end: }
680
681 \AssignTaggingSocketPlug{math/struct/begin}{default}
682 \AssignTaggingSocketPlug{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`

```

683 \cs_generate_variant:Nn \str_mdfive_hash:n {o}
684 \tl_new:N\l__math_content_hash_tl

```

we need to save the grabbed math:

```

685 \tl_new:N\l__math_grabbed_math_tl

```

the socket definition

```

686 \NewTaggingSocketPlug
687 {math/struct/begin}
688 {mathml-AF}
689 {
690   \int_gincr:N\g__math_math_total_int
691   \tl_set:N\l__math_content_hash_tl
692   {\str_mdfive_hash:o { \l__math_content_AF_source_tl }}
693   \tl_set_eq:NN\l__math_grabbed_math_tl\g__math_grabbed_math_tl
694   \tl_if_eq:NnTF\g__math_grabbed_env_tl {math}
695   {
696     \tl_set:Nn\l__math_attribute_class_tl{inline}
697   }
698   {
699     \tl_set:Nn\l__math_attribute_class_tl{display}
700   }
701   \bool_if:NF\l__tag_math_alt_bool
702   { \tl_set:Nn \l__math_content_alt_tl{} }

```

debugging option. TODO: hide in debug key.

```

703   \tl_if_exist:cTF { g__math_mathml_ \l__math_content_hash_tl _tl }
704   {
705     \int_gincr:N\g__math_mathml_AF_found_int
706     \bool_if:NTF \l__tag_math_mathml_AF_bool
707     {
708       \int_gincr:N\g__math_mathml_AF_attached_int
709
710       \__math_debug_typeout:n {Inserting~mathml~with~Hash~\l__math_content_hash_tl}
711       {
712         \__math_debug_typeout:n {Ignoring~mathml~with~Hash~\l__math_content_hash_tl}
713       }
714     }
715     {
716       \bool_if:NT \l__tag_math_mathml_AF_bool
717       {
718         \typeout {WARNING:~mathml~missing~for~hash~\l__math_content_hash_tl}
719       }
720     }
721     \tag_socket_use:n {math/mathml/write/prepare}
722     \tag_socket_use:n {math/mathml/write} % write hash if request
723     \bool_if:NTF\l__tag_math_texsource_AF_bool
724     { \tl_set_eq:NN \l__math_content_AF_source_tmpa_tl \l__math_content_AF_source_tl }
725     { \tl_clear:N \l__math_content_AF_source_tmpa_tl }
726     \tag_struct_begin:n
727     {
728       tag=Formula,
729       attribute-class=\l__math_attribute_class_tl, %
730       AFref =
731       \bool_if:NT\l__tag_math_mathml_AF_bool
732       {
733         \cs_if_exist_use:c {g__math_mathml_ \l__math_content_hash_tl _tl}
734       },

```

```

735 texsource = \l__math_content_AF_source_tmpa_tl, % should be after mathml AF!
736 title-o   = \g__math_grabbed_env_tl, %
737 alt       = \l__math_content_alt_tl
738 }
739 \__math_debug_typeout:n {grabbed~math=\meaning\g__math_grabbed_math_tl}
740 \tag_mc_begin:n{
741 }

```

not really needed but looks more symmetric:

```

742 \NewTaggingSocketPlug
743 {math/struct/end}
744 {mathml-AF}
745 {
746   \tag_mc_end:
747   \tag_struct_end:
748 }

```

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

```

749 \NewTaggingSocket{math/end}{0}

```

```

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

```

## 8.10 Interface commands

```

\__math_process:nn A no-op place-holder; the internal wrapper means that it does not need to be concerned
\__math_process:Vn with internals.
\__math_process_auxi:nn
\__math_process_auxii:nn \newif\ifmeasuring@
758 \cs_new_protected:Npn \__math_process:nn #1#2
759 {
760   \legacy_if:nF { measuring@ }
761   {
762     \tl_if_in:nnTF {#2} { \m@th }
763     { \bool_set_true:N\l__math_fakemath_bool }
764     { \tl_trim_spaces_apply:nN {#2} \__math_process_auxi:nn {#1} }
765   }
766 }
767 }
768 \cs_generate_variant:Nn \__math_process:nn { V }
769 \cs_new_protected:Npn \__math_process_auxi:nn #1#2
770 {
771   \tl_gset:Nn \g__math_grabbed_env_tl {#2}
772   \tl_gset:Nn \g__math_grabbed_math_tl {#1}

```

```

773 \_math_process_auxii:nn {#2} {#1}
774 }
775 \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

```

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

(End of definition for \math_processor:n. This function is documented on page 5.)

```

## 8.11 Content grabbing

**\MathCollectTrue**  
**\MathCollectFalse**

```

778 \cs_set_protected:Npn\MathCollectTrue{\bool_set_false:N \l_math_collected_bool}
779 \cs_set_protected:Npn\MathCollectFalse{\bool_set_true:N \l_math_collected_bool}

(End of definition for \MathCollectTrue and \MathCollectFalse. These functions are documented on
page 6.)

```

**\\_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.

```

780 \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.

```

781 \cs_new_protected:Npn \_math_grab:n #1
782 {

```

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

```

783 \_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.

```

784 \bool_lazy_any:nTF
785 {
786   {\legacy_if_p:n { measuring@ }}
787   { \l_math_fakemath_bool }
788   { \tl_if_blank_p:n {#1} }
789 }
790 {
791   \_math_luaamml_ignore:
792   #1 $ % $
793 }
794 {
795   \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.

```

796         \tag_socket_use:nnn {math/inline/formula/begin}{#1}
797         $ % $
798         \tag_socket_use:n {math/inline/formula/end}
799         \tag_socket_use:n {math/inline/end} % restart P-MC
800     }
801 }

```

*(End of definition for `\_math_grab_dollar:w` and `\_math_grab: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.

```

802 \cs_new_protected:Npn \_math_grab_dollar_delim:w #1 $ % $
803 { \_math_grab:n {#1} }

```

*(End of definition for `\_math_grab_dollar_delim:w`.)*

`\_math_grab_dollardollar:w` And for the classical TeX display structure.

```

804 \cs_new_protected:Npn \_math_grab_dollardollar:w #1 $$ {
805   \tl_if_blank:nF {#1}
806   {
807     \_math_process:nn { equation* } {#1}
808     \tag_socket_use:n {math/display/begin}
809     \tag_socket_use:nn{math/display/formula/begin}{#1}
810   }

```

Prepare to finish the display math formula and let TeX do its job; then regain control after the formula has been contributed to the page, in order to add the tagging followed by the correct penalty and skip.

```

811   \_math_prepare_display_end:
812   $$
813 }

```

*(End of definition for `\_math_grab_dollardollar:w`.)*

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

```

814 \cs_new_protected:Npn \_math_grab_inline_delim:w % \ (
815   #1 \)
816   {
817     \tl_if_blank:nF {#1}
818     {
819       $ #1 $
820     }
821     \bool_set_false:N \l_math_collected_bool
822   }

```

*(End of definition for `\_math_grab_inline_delim:w`.)*

`\_math_grab_inline:w` See `\@@_grab_dollar:w`.

```
823 \cs_new_protected:Npn \_math_grab_inline:w { \_math_grab_inline_delim:w }
      (End of definition for \_math_grab_inline:w.)
```

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

```
824 \cs_new_protected:Npn \_math_grab_eqn:w % \[
825   #1 \]
826   {
827   % \typeout{collected? = \bool_if:NTF \l_math_collected_bool {true}{false}}
828   \begin { equation* } #1 \end { equation* }
829   }
      (End of definition for \_math_grab_eqn:w.)
```

## 8.12 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.

```
830 \tl_new:N \l_math_grabbed_tl
```

---

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

```
831 \int_new:N \l_math_grab_env_int
```

`\_math_grab_loop_aux:` 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.

```
832 \cs_new_protected:Npn \_math_grab_loop_aux:
833   {
834   \group_begin:
835   \tl_clear:N \l_math_grabbed_tl
836   \_math_grab_loop:
837   }
838 \cs_new_protected:Npn \_math_grab_loop:
839   {
840   \peek_remove_spaces:n
841   {
842   \peek_meaning:NTF \c_group_begin_token
843     { \_math_grab_loop_group:n }
844     { \_math_grab_loop_token:N }
845   }
846   }
```

(End of definition for `\__math_grab_loop_aux:` and `\__math_grab_loop:.`)

```

\__math_grab_loop_group:n Handling of grabbed groups is pretty easy.
\__math_grab_loop_store:n
847 \cs_new_protected:Npn \__math_grab_loop_group:n #1
848   { \__math_grab_loop_store:n { {#1} } }
849 \cs_new_protected:Npn \__math_grab_loop_store:n #1
850   {
851     \tl_put_right:Nn \l__math_grabbed_tl {#1}
852     \__math_grab_loop:
853   }

```

(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_\\:
\__math_grab_loop_\begin 854 \cs_new_protected:Npn \__math_grab_loop_token:N #1
\__math_grab_loop_\end 855   {
\__math_grab_loop_\ignorespaces: 856     \cs_if_exist_use:cF
\__math_grab_loop_\unskip: 857       { \__math_grab_loop_ \token_to_str:N #1 : }
\__math_grab_loop_\textonly@unskip: 858       { \__math_grab_loop_store:n {#1} }
859     }
860 \cs_new_protected:cpn { \__math_grab_loop_ \token_to_str:N $ : }
861   { \__math_grab_loop_end: }
862 \cs_new_protected:cpn { \__math_grab_loop_ \token_to_str:N \ ) : }
863   { \__math_grab_loop_end: }
864 \cs_new_protected:cpn { \__math_grab_loop_ \token_to_str:N \\ : }
865   {
866     \int_compare:nNnTF \l__math_grab_env_int = 0
867       { \__math_grab_loop_newline: }
868       { \__math_grab_loop_store:n { \\ } }
869   }

```

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

```

870 \cs_new_protected:cpn { \__math_grab_loop_ \token_to_str:N \begin : }
871   {
872     \int_incr:N \l__math_grab_env_int
873     \__math_grab_loop_store:n { \begin }
874   }
875 \cs_new_protected:cpn { \__math_grab_loop_ \token_to_str:N \end : }
876   {
877     \int_compare:nNnTF \l__math_grab_env_int = 0
878       {
879         \__math_grab_loop_newline:
880         \end
881       }
882       {
883         \int_decr:N \l__math_grab_env_int

```

```

884         \_math_grab_loop_store:n { \end }
885     }
886 }
887 \tl_map_inline:nn { \ignorespaces \unskip \textonly@unskip }
888 {
889     \cs_new_protected:cpn { __math_grab_loop_ \token_to_str:N #1 : }
890     { \_math_grab_loop: }
891 }

```

(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  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  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`.

```

892 \cs_new_protected:Npn \_math_grab_loop_newline:
893 {
894     \if_false: { \fi:
895     \cs_set_protected:Npn \cr
896     {
897         \_math_grab_loop:
898         \tex_cr:D
899     }
900     \if_false: } \fi:
901     \\\
902 }

```

(End of definition for `\_math_grab_loop_newline:.`)

`\_math_grab_loop_end:` Clean up and pass on.

```

903 \cs_new_protected:Npn \_math_grab_loop_end:
904 {
905     \exp_args:NNV \group_end:
906     \_math_grab:n \l_math_grabbed_tl
907 }

```

(End of definition for `\_math_grab_loop_end:.`)

### 8.13 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”

```

908 \tl_new:N \l_math_env_name_tl

```

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



```

909 \keys_define:nn { __math }
910 {
911   arg-spec .code:n =
912   {
913     \ExpandArgs { c } \DeclareDocumentCommand
914     { __math_env \l__math_env_name_tl _aux: }
915     {#1}
916     { \__math_env_forward:w }
917   }
918 }

\math_register_env:nn Set up to capture environment content and make available.
\math_register_env:n
\RegisterMathEnvironment 919 \cs_new_protected:Npn \math_register_env:nn #1#2
920 {
921   \tl_set:Nn \l__math_env_name_tl {#1}
922   \keys_set:nn { __math } {#2}
923   \cs_gset_eq:cc { __math_env_ #1 _begin: } {#1}
924   \cs_gset_eq:cc { __math_env_ #1 _end: } { end #1 }
925   %
926   \ExpandArgs { nne } \RenewDocumentEnvironment {#1} { b }
927   {
928     \exp_not:N \bool_if:NTF \exp_not:N \l__math_collected_bool
929     {
930       % \typeout{===>B1}
931     }
932     {
933       % \typeout{===>B2}
934       \cs_if_exist:CTF { __math_env #1 _aux: }
935       {
936         \exp_not:c { __math_env #1 _aux: }
937         ##1 \exp_not:N \__math_env_end: {#1}
938       }
939       { \exp_not:N \__math_process:nn {#1} {##1} }
940       \exp_not:n { \@kernel@math@registered@begin }
941       \bool_set_true:N \exp_not:N \l__math_collected_bool
942     }
943     % \exp_not:N \tracingall
944     \exp_not:c { __math_env_ #1 _begin: }
945     ##1
946     \exp_not:c { __math_env_ #1 _end: }
947     % \exp_not:N \tracingnone
948   }
949   {
950   }
951 }

952 \cs_new_protected:Npn \math_register_halign_env:nn #1#2
953 {
954   \tl_set:Nn \l__math_env_name_tl {#1}
955   \keys_set:nn { __math } {#2}
956   \cs_gset_eq:cc { __math_env_ #1 _begin: } {#1}
957   \cs_gset_eq:cc { __math_env_ #1 _end: } { end #1 }

```

```

958 %
959 \ExpandArgs { nnee } \RenewDocumentEnvironment {#1} { b }
960 {
961   \exp_not:N \bool_if:NTF \exp_not:N \l__math_collected_bool
962   {
963     \typeout{==>B1}
964   }
965   {
966     \typeout{==>B2}
967     \cs_if_exist:CTF { __math_env #1 _aux: }
968     {
969       \exp_not:c { __math_env #1 _aux: }
970       ##1 \exp_not:N \__math_env_end: {#1}
971     }
972     { \exp_not:N \__math_process:nn {#1} {##1} }
973     \exp_not:n { \@kernel@math@registered@begin }
974     \bool_set_true:N \exp_not:N \l__math_collected_bool
975   }
976   \exp_not:N \tracingall
977   \exp_not:c { __math_env_ #1 _begin: }
978   ##1
979   \exp_not:N \tracingnone
980 }
981 {
982   \exp_not:c { __math_env_ #1 _end: }
983 }
984 }
985
986
987
988 \cs_new:Npn \@kernel@math@registered@begin {
989 % \ShowTagging{struct-stack}
990 %\typeout{==>A1}\ShowTagging{struct-stack,mc-current}
991 \mode_if_vertical:TF
992 {
993   \legacy_if:nTF { @endpe }
994   { \legacy_if_set_false:n { @endpe } }
995   { \__block_list_beginpar_vmode: }
996 %
997 % \typeout{==>~ at:~ \g__tag_struct_tag_tl}
998 %
999 \tag_if_active:T
1000 {
1001   \exp_args:Noo\str_if_eq:nnF \g__tag_struct_tag_tl { \l__tag_para_main_tag_tl } %
1002   {
1003     \typeout{==>A2}
1004     \__block_beginpar_vmode:
1005     } % needs correction!
1006   }
1007 }
1008 {
1009 % \typeout{==>A3}
1010 \__tag_tool_close_P:
1011 }

```

```

1012 \tag_socket_use:nn{math/display/formula/begin}{-}{-}
1013 % \typeout{==>MC1}\ShowTagging{mc-current}
1014 }
1015
1016 \cs_new_protected:Npn \math_register_env:n #1
1017 { \math_register_env:nn {#1} { } }
1018
1019 \NewDocumentCommand \RegisterMathEnvironment { 0{} m }
1020 { \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 5.)*

`\__math_env_forward:w`

```

1021 \cs_new_protected:Npn \__math_env_forward:w #1 \__math_env_end: #2
1022 { \__math_process:nn {#2} {#1} }

```

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

## 8.14 Regaining control after a display has finished with `$$`

The tagging structures have to be added after the formula content but before  $\TeX$  is allowed to break a page. But  $\TeX$  will automatically add `\postdisplaypenalty` followed by `\belowdisplayskip` or `\belowdisplayshortskip` without giving us any chance to take control before these are added.

We therefore implement the following approach:

- Just before we end the formula we save away the current value of `\postdisplaypenalty` in case it was altered within the formula. Then we set it to 10000 so that what is inserted by  $\TeX$  is not allowing for a page break at this point
- At this point We also normally flip the sign of `\belowdisplayskip` and `\belowdisplayshortskip` so that they become negative and record that we have done this, by setting the token list `\g__math_skip_sign_tl` to `-`.
- However, we only do that if both are non-negative. If either of them is negative we assume that this was deliberately done by the user to adjust the spacing around this particular display. Again, we record in `\g__math_skip_sign_tl` that we haven't done the sign flip by setting the variable to empty.
- Making these two skips negative is essential, because the formula will be added using the special `\postdisplaypenalty` value that doesn't allow a page break even though the real value (that we use later on) will probably do that. Thus the below skip added by  $\TeX$  will add to the size of the display and not vanish into the bottom margin and if it is positive  $\TeX$  might decide to move the whole formula onto the next page even though it might well fit.
- Once  $\TeX$  has finished processing the closing `$$` it has added something like

```

\penalty 100000          % our special value for \postdisplaypenalty
\glue -10.0 plus -3pt    % the \belowdisplayskip (with sign flipped)

```

after the formula. This means that at this that point we can retrieve this skip by using `\lastskip` and we just have to flip the sign again to know how to correct it (no need to know whether the normal or the short skip was added by `TEX`) and the right penalty is available to us too as we have saved it away in `\g__math_postdisplaypenalty_int`.

`\_math_prepare_display_end:`

Prepare to finish the formula and give control to `TEX`. This is normally used directly in front of `$$` (`\eqno` or `\leqno`).

```
1023 \cs_new_protected:Npn \_math_prepare_display_end: {
1024   \bool_lazy_or:nnTF
1025     { \dim_compare_p:nNn \belowdisplayskip < {0pt} }
1026     { \dim_compare_p:nNn \belowdisplayshortskip < {0pt} }
```

No sign flipping if one or both of the skips are already negative.

```
1027   { \tl_gclear:N \g__math_skip_sign_tl }
```

Otherwise flip the sign of both.

```
1028   {
1029     \tl_gset:Nn \g__math_skip_sign_tl {-}
1030     \skip_set:Nn \belowdisplayskip {-\belowdisplayskip}
1031     \skip_set:Nn \belowdisplayshortskip {-\belowdisplayshortskip}
1032   }
```

For the skip it is enough to flip the sign, because we get the value back through `\lastskip` but for the `\postdisplaypenalty` change we have to save the current value somewhere, because it may have been changed within the display formula. This has to be done globally, because it is used after the formula has ended.

```
1033   \int_gset_eq:NN \g__math_postdisplaypenalty_int \postdisplaypenalty
1034   \int_set_eq:NN \postdisplaypenalty \@M
1035 }
```

*(End of definition for `\_math_prepare_display_end:.`)*

`\_math_tag_dollardollar_display_end:`

The code to be executed after the formula has ended is injected using `\group_insert_after:N` inside of `\tex_everydisplay:D` (i.e., when the formula starts but inside the group started by the display). As `\group_insert_after:N` expects a single token we have to store that code in a command.

```
1036 \cs_new_protected:Npn \_math_tag_dollardollar_display_end:
1037   {
1038     % \typeout{== tag dollarldollar display end}
1039     % \ShowTagging{struct-stack}
1040     \para_raw_end:
```

The `\postdisplaypenalty` was temporarily set to 10000 inside the display and both the `\belowdisplayskip` and the `\belowdisplayshortskip` were negated (with some exceptions, see above), so whatever was inserted it should have been a negative or possibly zero skip. Whatever was added, we pick up the value, so that we can correct the spacing after the tagging code has been inserted.

```

1041 \l__math_tmpa_skip \lastskip
1042 \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 T<sub>E</sub>X has added the penalty and the “below skip”.

```

1043 \nobreak
1044 \skip_vertical:n { -\l__math_tmpa_skip }

```

Then we finally add the real stuff: the true `\postdisplaypenalty` (saved in `\g__math_postdisplaypenalty_int` earlier) and the negated value of the skip value we saved in `\l__math_tmpa_skip`. It may look strange that we have two identical negated skips next to each other, but if you think about it, that is correct: the first cancels the “below skip” that T<sub>E</sub>X has added and the second puts the same amount of skip after the penalty (which is where it should be).

```

1045 \penalty \g__math_postdisplaypenalty_int
1046 \skip_vertical:n

```

Actually we do use the skip without flipping the sign when our records show that it wasn’t flipped earlier i.e., when the negative value was due to the user specifying it inside the formula.

```

1047 { \g__math_skip_sign_tl \l__math_tmpa_skip }

```

As we are now in vertical mode the situation is different from the way T<sub>E</sub>X would handle things after a display: T<sub>E</sub>X would internally switch to horizontal mode without adding a `\parskip`. But this is not possible to do on the macro level. Therefore we have to neutralize the upcoming `\parskip` since we can’t prevent it from being added.

Actually, we could combine this correction with the previous `\skip_vertical:n`, which would produce marginally smaller PDFs; but that will make `\showoutput` tracing somewhat less easy to understand, so I haven’t done that (yet).

```

1048 % \typeout{----->~ add~ negative~ parskip~ (to~ cancel~ the~
1049 % one~ that~ TeX~ will~ add)}
1050 \skip_vertical:n { -\tex_parskip:D }

```

We also set the `@domathendpetrue` flag to signal that paragraph continuation should happen after such a display.

```

1051 \@domathendpetrue
1052 \@doendpe % this has no \end{...} to take care of it
1053 }

```

*(End of definition for \\_\_math\_tag\_dollardollar\_display\_end:.)*

`\g__math_postdisplaypenalty_int` This is the internal variable in which we save the `\postdisplaypenalty`. It is global because we use it immediately after the formula has ended.

```

1054 \int_new:N \g__math_postdisplaypenalty_int

```

(End of definition for `\g__math_postdisplaypenalty_int`.)

`\g__math_skip_sign_tl` We record whether we have flipped the signs of `\belowdisplayskip`, etc., so that we know what to do after the formula has ended. If we have it flipped the signs then variable holds `-`, whilst otherwise it is empty. This means we can flip the signs again by simply prefixing the value with this token list variable.

```
1055 \tl_new:N \g__math_skip_sign_tl
```

(End of definition for `\g__math_skip_sign_tl`.)

`\dollar@end` When we do tagging we augment the kernel definition for `\dollar@end` in order to regain control at the very end of the formula (after the user may have altered `\postdisplaypenalty` or `\belowdisplayskip`).

```
1056 \def \dollar@end { \__math_prepare_display_end: $$ }
```

(End of definition for `\dollar@end`.)

`\eqno` However, in case of a formula using the primitives `\eqno` or `\leqno` the `\dollar@end` comes too late as it is executed in the context of the equation number; and the values for `\postdisplaypenalty`, etc. set at this point are not the ones used for the formula. We therefore have to execute `\__math_prepare_display_end:` just in front of the primitive.

```
1057 \protected\def\eqno{
```

```
1058   \__math_prepare_display_end:
```

Inside the equation we set it temporarily to do nothing, because otherwise it would be executed again when `\dollar@end` is reached (not that this would matter, but it would just take unnecessary extra time).

```
1059   \@kernel@eqno
```

```
1060   \cs_set_eq:NN \__math_prepare_display_end: \prg_do_nothing:
```

```
1061   \aftergroup\ignorespaces
```

```
1062 }
```

Same game for `\leqno`.

```
1063 \protected\def\leqno{
```

```
1064   \__math_prepare_display_end:
```

```
1065   \@kernel@leqno
```

```
1066   \cs_set_eq:NN \__math_prepare_display_end: \prg_do_nothing:
```

```
1067   \aftergroup\ignorespaces
```

```
1068 }
```

(End of definition for `\eqno` and `\leqno`.)

## 8.15 Document commands

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

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

`\equation*`

`\_math\_equation\_star\_begin:`

`\endequation`

`\_math\_equation\_end:`

`\endequation*`

```
\_math\_equation\_star\_end:
1069 \tl_gput_right:Nn \@kernel@before@begindocument
1070 {
1071   \math_register_env:n { equation }
1072   \math_register_env:n { equation* }
1073   % at the moment register_env can only do display math
1074   %   \math_register_env:n { math }
1075   \RenewDocumentEnvironment{math} {b}{\${1$}}{}
1076   % and this one doesn't work either
1077   %   \math_register_env:n { displaymath }
1078   \RenewDocumentEnvironment{displaymath} {b}{\[1\]}{}
1079 }
```

(End of definition for `\equation` and others.)

`\(` If math mode has not been collected, we need to do that; otherwise, worry about whether  
`\)` 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.

```
1080 \cs_gset_protected:Npn \( % \)
1081 {
1082   \bool_if:NTF \l_math_collected_bool
1083   {
1084     \mode_if_math:TF
1085     { \@badmath }
1086     { $ }
1087   }
1088   {
1089     \_math_grab_inline:w
1090   }
1091 } % \)
1092 \cs_gset_protected:Npn \)
1093 {
1094   \mode_if_math:TF
1095   { $ }
1096   { \@badmath }
1097 }
```

(End of definition for `\(` and `\)`.)

`\[` Again, we need to watch for when `amsmath` is loaded after this code. The flag usage here  
`\]` 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.

```
1098 \tl_gput_right:Nn \@kernel@before@begindocument
1099 {
1100   \cs_gset_protected:Npn \[ % \]
```

```

1101     {
1102         \__math_grab_eqn:w
1103     %     \bool_if:NTF \l__math_collected_bool
1104     %     { \begin { equation* } }
1105     %     { \__math_grab_eqn:w }
1106     } % \[
1107     \cs_gset_protected:Npn \]
1108     {
1109         \@badmath
1110     %     \bool_if:NTF \l__math_collected_bool
1111     %     { \end{ equation* } }
1112     %     { \@badmath }
1113     }
1114 }

```

(End of definition for `\[` and `\]`.)

why does ensuremath  
need handling at all?

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

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

```

1115 %\cs_gset_protected:Npn \ensuremath #1
1116 % {
1117 %     \mode_if_math:TF
1118 %     {#1}
1119 %     {
1120 %         \bool_if:NTF \l__math_collected_bool
1121 %         { \@ensuredmath {#1} }
1122 %         {
1123 %             \bool_set_true:N \l__math_collected_bool
1124 %             \__math_process:nn { math } {#1}
1125 %             \@ensuredmath {#1}
1126 %             \bool_set_false:N \l__math_collected_bool
1127 %         }
1128 %     }
1129 % }

```

(End of definition for `\ensuremath`.)

## 8.16 `\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.

```

1130 \exp_args:No \tex_everymath:D
1131 {
1132     \tex_the:D \tex_everymath:D
1133     \bool_if:NF \l__math_collected_bool
1134     {
1135         \bool_set_true:N \l__math_collected_bool
1136         \__math_grab_dollar:w
1137     }
1138 }
1139
1140 \exp_args:No \tex_everydisplay:D
1141 {
1142     \tex_the:D \tex_everydisplay:D
1143 %     \typeout{==>~ in~ everydisplay}

```



We need to attach tagging after the display math has been processed by  $\TeX$ , and we also need to correct the penalty and spacing that will get added there. This is done by the following code through which we regain control after the display math group ends.

```

1144 \group_insert_after:N \__math_tag_dollardollar_display_end:
1145 \bool_if:NF \l__math_collected_bool
1146 {
1147   \bool_set_true:N \l__math_collected_bool
1148   \__math_grab_dollardollar:w
1149 }
1150 }
```

## 8.17 Modifying kernel environments

We need to cover this even though it is, of course, not encouraged. Registration is delayed until `\begindocument` to avoid error with redefinition in, e.g., `fleqn.clo`.

```

1151 \tl_gput_right:Nn \@kernel@before@begindocument
1152 {
1153   \math_register_env:n { eqnarray }
1154   \math_register_env:n { eqnarray* }
1155 }
```

Tabulars currently contain a `$` that shouldn't trigger math tagging. Also we do need to change the grabbing method to the slow loop-method.

```

1156 \RequirePackage{array}
1157 \tl_if_exist:NT \@kernel@tabular@init
1158 {
1159   \tl_put_right:Nn \@kernel@tabular@init
1160   {
1161     \cs_set_protected:Npn \__math_grab_dollar:w { \__math_grab_loop_aux: }
1162     \cs_set_protected:Npn \__math_grab_inline:w { $ }
1163   }
1164 }
```

`\__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!

```

1165 \cs_new_eq:NN \__math_m@th: \m@th
1166 \cs_gset_protected:Npn \m@th
1167 {
1168   \bool_set_true:N \l__math_collected_bool
1169   \__math_m@th:
1170 }
```

(End of definition for `\__math_m@th:` and `\m@th.`)

## 8.18 Modifying kernel commands

`\mathpalette` The `\mathpalette` command is a problem if `lualatex` is used and math structure elements are created as the math is then processed more than once. We therefore redefine it to use `\mathstyle`.

```

1171 \sys_if_engine luatex:T
1172 {
1173   \def\mathpalette#1#2{%
1174     \ifcase\mathstyle
1175       #1\displaystyle{#2}\or
1176       #1\displaystyle{#2}\or
1177       #1\textstyle{#2}\or
1178       #1\textstyle{#2}\or
1179       #1\scriptstyle{#2}\or
1180       #1\scriptstyle{#2}\or
1181       #1\scriptscriptstyle{#2}\or
1182       #1\scriptscriptstyle{#2}
1183     \fi}
1184   }

```

(End of definition for `\mathpalette`.)

`\mathsm@sh` Similar to the phantom commands in latex-lab-text, `\mathsm@sh` must be redefined to include luamml sockets.

```

1185 \def\mathsm@sh#1#2{%
1186   \setbox\z@\hbox{$\mathsm@sh#1{#2}
1187     \UseTaggingSocket{math/luamml/save/nNn}{\mathsmash} #1 {mpadded}}
1188   $}%
1189   \UseTaggingSocket{math/luamml/finsm@sh}{\finsm@sh}}

```

(End of definition for `\mathsm@sh`.)

## 8.19 Disable math grabbing in the begindocument hook

For example amsart uses math to measure text there.

```

1190 \tl_gput_right:Nn\@kernel@before@begindocument
1191 {
1192   \bool_set_true:N\l__math_collected_bool
1193 }
1194 \tl_gput_right:Nn\@kernel@after@begindocument
1195 {
1196   \bool_set_false:N\l__math_collected_bool
1197 }
1198 \ExplSyntaxOff
1199 <@@=
1200 </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	
\#	95, 380
\%	96
\(	814
\(	<u>1080</u>
\)	815, 862
\)	<u>1080</u>
@@ commands:	
\l_@@_collected_bool	3, <u>12</u>
\l_@@_content_template_tl	9
\l_@@_fakemath_bool	12
\[	824, <u>1078</u>
\[	13, 38, 46, 47, <u>1098</u>
\]	207, 208,
	209, 210, 211, 212, 290, 864, 868, 901
\{	93
\}	94
\]	825, <u>1078</u>
\]	13, 38, 46, 47, <u>1098</u>
\^	97
A	
actual+source (plug)	<u>615</u>
\AddToHook	6, 9, 160, 166,
	172, 183, 195, 246, 361, 401, 418, 444
\aftergroup	1061, <u>1067</u>
alt+source (plug)	<u>630</u>
\AssignTaggingSocketPlug	
	220, 221, 225, 359, 360,
	399, 400, 585, 586, 645, 681, 682,
	750, 751, 752, 753, 754, 755, 756, 757
B	
\begin ...	39, 591, 609, 828, 870, 873, 1104
\begingroup	92
\belowdisplayshortskip	43, 44, 1026, 1031
\belowdisplayskip	43, 44, 46, 1025, 1030
block internal commands:	
\__block_beginpar_vmode:	1004
\__block_list_beginpar_vmode:	995
bool commands:	
\bool_gset_false:N	20, 326
\bool_gset_true:N	
	15, 219, 285, 322, 347, 388
\bool_if:NTF	26,
	28, 85, 217, 257, 396, 454, 652, 662,
	701, 706, 716, 723, 731, 827, 928,
	961, 1082, 1103, 1110, 1120, 1133, 1145
\bool_lazy_any:nTF	784
\bool_lazy_or:nnTF	1024
\bool_new:N	10,
	33, 34, 60, 61, 62, 63, 64, 65, 284
\bool_set_false:N	
	262, 278, 778, 821, 1126, 1196
\bool_set_true:N	260, 458, 764, 779,
	941, 974, 1123, 1135, 1147, 1168, 1192
bool internal commands:	
\l__math_collected_bool	15,
	33, 257, 260, 262, 278, 778, 779,
	821, 827, 928, 941, 961, 974, 1082,
	1103, 1110, 1120, 1123, 1126, 1133,
	1135, 1145, 1147, 1168, 1192, 1196
\g__math_debug_bool	
	14, 10, 15, 20, 26, 28
\l__math_fakemath_bool	15, 34, 764, 787
\g__math_luamml_write_bool	
	23, 217, 284, 285, 322, 326, 330
box commands:	
\box_clear:N	395
\box_new:N	371
box internal commands:	
\l__math_tmpa_box	371, 374, 395
\boxed	12
C	
char commands:	
\char_set_catcode_other:N	
	93, 94, 95, 96, 97, 380
clist commands:	
\clist_map_inline:Nn	382
\clist_map_inline:nn	478
\clist_new:N	73
\clist_put_right:Nn	74, 494
\cr	895
cs commands:	
\cs_generate_variant:Nn	431, 683, 768
\cs_gset_eq:NN	923, 924, 956, 957
\cs_gset_protected:Npe	25, 27
\cs_gset_protected:Npn	
	1080, 1092, 1100, 1107, 1115, 1166
\cs_if_exist:NTF	934, 967
\cs_if_exist_use:N	733
\cs_if_exist_use:NTF	856
\cs_if_free:NTF	143, 148, 153
\cs_new:Npn	229, 286, 287, 988
\cs_new_eq:NN	11, 12, 1165



<code>\g_math_mathml_total_int</code> . . . . .		maketag internal commands:	
. . . . .	17, 68, 79, 405	<code>\maketag_math@</code> . . . . .	30
<code>\l_math_mathstyle_int</code> . . . . .	419, 426	math commands:	
<code>\g_math_mml_int</code> . . . . .	17	<code>\math_debug_off:</code> . . . . .	11, 13, 18, 31
<code>\g_math_mml_total_int</code> . . . . .	17	<code>\math_debug_on:</code> . . . . .	11, 13, 13, 30
<code>\g_math_postdisplaypenalty_int</code>		<code>\math_processor:n</code> . . . . .	5, 776, 776
. . . . .	44, 45, 1033, 1045, 1054	<code>\math_register_env:n</code> . . . . .	5, 919, 1016,
<code>\intertext</code> . . . . .	13	. . . . .	1071, 1072, 1074, 1077, 1153, 1154
iow commands:		<code>\math_register_env:nn</code> . . . . .	
<code>\iow_close:N</code> . . . . .	250, 365	. . . . .	5, 919, 919, 1017, 1020
<code>\iow_new:N</code> . . . . .	226, 350	<code>\math_register_halign_env:nn</code> . . .	952
<code>\iow_newline:</code> . . . . .		math internal commands:	
. . . . .	107, 109, 114, 116, 131, 133, 135, 137	<code>__math_AF_html_reader:w</code> . . .	91, 381
<code>\iow_now:Nn</code> . . . . .	228, 236, 248, 336, 355, 363	<code>__math_AF_html_reader_verb:w</code> . . .	98, 100
<code>\iow_open:Nn</code> . . . . .	227, 351	<code>__math_AF_mml:nnnn</code> . . . . .	76, 76, 102
<code>\iow_term:n</code> . . . . .		<code>__math_AF_process_mathml_files:</code>	
. . . . .	403, 404, 405, 407, 409, 411, 413	. . . . .	371, 372, 418
iow internal commands:		<code>__math_debug:n</code> . . . . .	11, 11, 25
<code>\g_math_luamml_iow</code> . . . . .		<code>__math_debug_gset:</code> . . . . .	13, 16, 21, 23
. . . . .	226, 227, 228, 236, 248, 250	<code>__math_debug_typeout:n</code> . . . . .	
<code>\g_math_writedummy_iow</code> . . . . .		. . . . .	11, 12, 27, 673, 709, 712, 739
. . . . .	336, 348, 350, 351, 355, 363, 365	<code>__math_env_end:</code> . . . . .	937, 970, 1021
<b>K</b>			
keys commands:		<code>__math_env_forward:w</code> . . . . .	916, 1021, 1021
<code>\keys_define:nn</code> . . . . .	293, 343, 432, 461, 909	<code>__math_equation_begin:</code> . . . . .	1069
<code>\keys_set:nn</code> . . . . .		<code>__math_equation_end:</code> . . . . .	1069
. . . . .	467, 480, 486, 496, 503, 922, 955	<code>__math_equation_star_begin:</code> . . .	1069
<b>L</b>		<code>__math_equation_star_end:</code> . . .	1069
<code>\lastskip</code> . . . . .	44, 1041	<code>__math_grab:n</code> . . . . .	780, 781, 803, 906
<code>\LaTeXe</code> . . . . .	12	<code>__math_grab_dollar:w</code> . . . . .	
<code>\leavevmode</code> . . . . .	9	. . . . .	780, 780, 1136, 1161
legacy commands:		<code>__math_grab_dollar_delim:w</code> . . . . .	
<code>\legacy_if:nTF</code> . . . . .	761, 993	. . . . .	780, 802, 802
<code>\legacy_if_p:n</code> . . . . .	786	<code>__math_grab_dollardollar:w</code> . . . . .	
<code>\legacy_if_set_false:n</code> . . . . .	994	. . . . .	804, 804, 1148
<code>\leqno</code> . . . . .	44, 46, 1057	<code>__math_grab_eqn:w</code> . . . . .	824, 824, 1102, 1105
<code>\long</code> . . . . .	276	<code>__math_grab_inline:w</code> . . . . .	
<code>\ltxlabmathdate</code> . . . . .	3	. . . . .	823, 823, 1089, 1162
<code>\ltxlabmathversion</code> . . . . .	4	<code>__math_grab_inline_delim:w</code> . . . . .	
luamml commands:		. . . . .	814, 814, 823
<code>\luamml_flag_ignore:</code> . . . . .	152, 155	<code>__math_grab_loop:</code> . . . . .	
<code>\luamml_flag_process:</code> . . . . .	147, 150	. . . . .	832, 836, 838, 852, 890, 897
<code>\luamml_flag_structelem:</code> . . .	142, 145	<code>__math_grab_loop_\$:</code> . . . . .	854
<code>\luamml_ignore:</code> . . . . .	153, 155, 309, 317	<code>__math_grab_loop_\backslash:</code> . . . . .	854
<code>\luamml_process:</code> . . . . .	148, 150	<code>__math_grab_loop_\begin:</code> . . . . .	854
<code>\luamml_structelem:</code> . . . . .	143, 145, 308	<code>__math_grab_loop_\end:</code> . . . . .	854
luamml internal commands:		<code>__math_grab_loop_\ignorespaces:</code> . . .	854
<code>\l_luamml_pretty_int</code> . . . . .	222	<code>__math_grab_loop_\textonly@unskip:</code>	
<code>\_luamml_register_output_hook:N</code> . . .	245	. . . . .	854
<b>M</b>		<code>__math_grab_loop_\unskip:</code> . . . . .	854
<code>\m@th</code> . . . . .	4	<code>__math_grab_loop_aux:</code> . . . . .	832, 832, 1161
		<code>__math_grab_loop_end:</code> . . . . .	
		. . . . .	861, 863, 903, 903
		<code>__math_grab_loop_group:n</code> . . . . .	
		. . . . .	843, 847, 847



\postdisplaypenalty .. 43–46, 1033, 1034  
 prg commands:  
   \prg\_do\_nothing: . 313, 314, 1060, 1066  
 property commands:  
   \property\_new:nnnn ..... 421  
   \property\_record:nn ..... 427  
   \property\_ref:nn ..... 428  
 \protected ..... 270, 1057, 1063  
 \providecommand ..... 142, 147, 152  
 \ProvidesFile ..... 2

## R

\RegisterFamilyMapping ..... 223, 224  
 \RegisterMathEnvironment .... 5, 13, 919  
 \RenewDocumentEnvironment .....  
   ..... 926, 959, 1075, 1078  
 \RequirePackage ..... 6, 9, 165, 182, 1156

## S

\sb ..... 22, 269  
 \scriptscriptstyle ..... 1181, 1182  
 \scriptstyle ..... 1179, 1180  
 \setbox ..... 1186  
 \showoutput ..... 45  
 \ShowTagging ..... 989, 990, 1013, 1039  
 skip commands:  
   \skip\_new:N ..... 38  
   \skip\_set:Nn ..... 1030, 1031  
   \skip\_vertical:n 45, 1044, 1046, 1050  
 skip internal commands:  
   \l\_math\_tmpa\_skip .....  
   ..... 15, 45, 38, 1041, 1044, 1047  
 \space ..... 3, 4  
 str commands:  
   \c\_backslash\_str ..... 132  
   \str\_case:nn ..... 162  
   \str\_if\_eq:nnTF ..... 446, 1001  
   \str\_mdfive\_hash:n ..... 683, 692  
   \str\_new:N ..... 39  
   \str\_replace\_all:Nnn ..... 126, 127  
   \str\_set:Nn ..... 125  
 str internal commands:  
   \l\_math\_tmpa\_str .....  
   ..... 15, 39, 125, 126, 127, 134  
 \SuspendTagging ..... 4  
 \symletters ..... 224  
 \symsymbols ..... 223  
 sys commands:  
   \sys\_if\_engine luatex:TF .. 158, 1171  
   \c\_sys\_jobname\_str . 75, 227, 353, 495

## T

tag commands:  
   \tag\_if\_active:TF ..... 47, 999

\tag\_mc\_begin:n ... 575, 583, 674, 740  
 \tag\_mc\_begin\_pop:n ..... 520  
 \tag\_mc\_end: ... 49, 573, 581, 679, 746  
 \tag\_mc\_end\_push: ..... 49, 516  
 \tag\_resume:n ..... 429  
 \tag\_socket\_use:n .....  
   . 526, 527, 529, 535, 556, 557, 559,  
   565, 721, 722, 795, 798, 799, 808, 1042  
 \tag\_socket\_use:nn ..... 809, 1012  
 \tag\_socket\_use:nnn ..... 796  
 \tag\_struct\_begin:n .. 6, 574, 664, 726  
 \tag\_struct\_end: ... 52, 582, 679, 747  
 \tag\_suspend:n ..... 429

tag internal commands:

\\_\_tag\_check\_para\_end\_show:nn ... 51  
 \\_\_tag\_gincr\_para\_end\_int: ..... 50  
 \l\_\_tag\_math\_alt\_bool .....  
   ..... 16, 65, 435, 454, 458, 662, 701  
 \g\_\_tag\_math\_luaaml\_tl ... 16, 66, 67  
 \g\_\_tag\_math\_mathml\_AF\_bool ....  
   ..... 16, 63, 219, 347, 388, 396  
 \l\_\_tag\_math\_mathml\_AF\_bool ....  
   ..... 16, 62, 439, 706, 716, 731  
 \l\_\_tag\_math\_mathml\_files\_clist  
   ..... 17, 73, 74, 382, 434, 494  
 \l\_\_tag\_math\_mathml\_pane\_bool ..  
   ..... 16, 64, 85, 436  
 \l\_\_tag\_math\_texsource\_AF\_bool .  
   ..... 16, 60, 441, 652, 723  
 \l\_\_tag\_math\_texsource\_pane\_bool  
   ..... 16, 61, 438  
 \l\_\_tag\_para\_main\_tag\_tl ..... 1001  
 \g\_\_tag\_struct\_tag\_tl ..... 997, 1001  
 \\_\_tag\_tool\_close\_P: 45, 45, 544, 1010

Tagging sockets:

math/content ..... 614  
 math/display/begin ..... 537  
 math/display/end ..... 537  
 math/display/formula/begin .... 537  
 math/display/formula/end ..... 537  
 math/display/tag/begin ..... 567  
 math/display/tag/end ..... 567  
 math/end ..... 749  
 math/inline/begin ..... 509  
 math/inline/end ..... 509  
 math/inline/formula/begin .... 509  
 math/inline/formula/end ..... 509  
 math/mathml/write ..... 333  
 math/mathml/write/prepare ..... 122  
 math/struct/begin ..... 646  
 math/struct/end ..... 646  
 \tagpdfparaOff ..... 524, 554  
 \tagpdfsetup ..... 56, 211, 213

T<sub>E</sub>X and L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> commands:

<code>\@M</code>	1034
<code>\@badmath</code>	1085, 1096, 1109, 1112
<code>\@doendpe</code>	1052
<code>\@domathendpetrue</code>	1051
<code>\@ensured@unreal@math</code>	270
<code>\@ensuredmath</code>	1121, 1125
<code>\@firstofone</code>	272
<code>\@ifpackageloaded</code>	180
<code>\@iiiiparbox</code>	12
<code>\@kernel@after@begindocument</code>	1194
<code>\@kernel@before@begindocument</code>	1069, 1098, 1151, 1190
<code>\@kernel@eqno</code>	1059
<code>\@kernel@leqno</code>	1065
<code>\@kernel@math@registered@begin</code>	940, 973, 988
<code>\@kernel@tabular@init</code>	1157, 1159
<code>\@math@level</code>	234, 263, 279
<code>\@textsubscript</code>	266
<code>\@textsuperscript</code>	266
<code>\@unreal@math</code>	267, 269, 270
<code>\cr</code>	38, 40
<code>\dollar@end</code>	46, 1056
<code>\finsm@sh</code>	1189
<code>\frozen@everymath</code>	21
<code>\halign</code>	40
<code>\ifmeasuring@</code>	5, 12, 758
<code>\m@th</code>	4, 11, 12, 15, 23, 37, 49, 267, 269, 763, 1165, 1186
<code>\math@level</code>	21
<code>\mathsm@sh</code>	50, 1185
<code>\sf@size</code>	267, 269
<code>\textonly@unskip</code>	887
<code>\z@</code>	1186
tex commands:	
<code>\tex_cr:D</code>	898
<code>\tex_everydisplay:D</code>	44, 1140, 1142
<code>\tex_everymath:D</code>	1130, 1132
<code>\tex_parskip:D</code>	1050
<code>\tex_the:D</code>	1132, 1142
<code>\text</code>	26
<code>\textstyle</code>	1177, 1178
tl commands:	
<code>\c_empty_tl</code>	451
<code>\c_space_tl</code>	132, 405, 407, 409, 411, 413, 592, 594, 596
<code>\tl_clear:N</code>	654, 725, 835
<code>\tl_const:Nn</code>	104, 112, 118, 599
<code>\tl_gclear:N</code>	1027
<code>\tl_gput_right:Nn</code>	1069, 1098, 1151, 1190, 1194
<code>\tl_gset:Nn</code>	67, 283, 296, 297, 321, 466, 485, 493, 771, 772, 1029

<code>\tl_gset_eq:NN</code>	88
<code>\tl_if_blank:nTF</code>	805, 817
<code>\tl_if_blank_p:n</code>	788
<code>\tl_if_empty:N</code>	231
<code>\tl_if_eq:NNTF</code>	602
<code>\tl_if_eq:NnTF</code>	655, 694
<code>\tl_if_exist:N</code>	80, 348, 703, 1157
<code>\tl_if_in:nTF</code>	763
<code>\tl_map_inline:nn</code>	887
<code>\tl_new:N</code>	35, 36, 37, 40, 41, 42, 43, 44, 55, 66, 87, 111, 282, 587, 598, 684, 685, 830, 908, 1055
<code>\tl_put_right:Nn</code>	851, 1159
<code>\tl_set:Nn</code>	128, 588, 600, 619, 623, 627, 628, 634, 638, 642, 643, 657, 660, 663, 691, 696, 699, 702, 921, 954
<code>\tl_set_eq:NN</code>	653, 693, 724
<code>\tl_to_str:n</code>	211, 213
<code>\tl_trim_spaces_apply:nN</code>	765
tl internal commands:	
<code>\l__math_attribute_class_tl</code>	55, 657, 660, 667, 696, 699, 729
<code>\l__math_content_actual_tl</code>	15, 41, 619, 643, 670
<code>\l__math_content_AF_mathml_tl</code>	44, 627, 642
<code>\l__math_content_AF_source_tl</code>	42, 125, 623, 638, 653, 692, 724
<code>\l__math_content_AF_source_tmpa_tl</code>	43, 653, 654, 668, 724, 725, 735
<code>\l__math_content_AF_tl</code>	15
<code>\l__math_content_alt_tl</code>	15, 40, 628, 634, 663, 671, 702, 737
<code>\l__math_content_hash_tl</code>	136, 684, 691, 703, 709, 712, 718, 733
<code>\l__math_content_template_tl</code>	31, 587, 588, 621, 636
<code>\l__math_env_name_tl</code>	40, 908, 914, 921, 954
<code>\g__math_grabbed_env_tl</code>	15, 32, 35, 591, 595, 602, 609, 611, 655, 669, 694, 736, 771
<code>\g__math_grabbed_math_tl</code>	15, 32, 36, 593, 605, 610, 673, 693, 739, 772
<code>\l__math_grabbed_math_tl</code>	685, 693
<code>\l__math_grabbed_tl</code>	38, 830, 835, 851, 906
<code>\c__math_inline_env_tl</code>	599, 602
<code>\g__math_luaamml_load_tl</code>	23, 162, 282, 283, 296, 297, 321, 466, 485, 493
<code>\c__math_mathml_write_after_tl</code>	104, 240, 340
<code>\l__math_mathml_write_before_tl</code>	104, 128, 231, 238, 338



