

# The `l3calc` package\*

## Infix notation arithmetic in L<sup>A</sup>T<sub>E</sub>X3

The L<sup>A</sup>T<sub>E</sub>X3 Project†

2010/02/07

This is pretty much a straight adaption of the `calc` package and as such has same syntax for the *⟨calc expression⟩*. However, there are some noticeable differences.

- The `calc` expression is expanded fully, which means there are no problems with unfinished conditionals. However, the contents of `\widthof` etc. is not expanded at all. This includes uses in traditional L<sup>A</sup>T<sub>E</sub>X as in the `array` package, which tries to do an `\edef` several times. The code used in `l3calc` provides self-protection for these cases.
- Muskip registers are supported although they can only be used in `\ratio` if already evaluating a muskip expression. For the other three register types, you can use `points`.
- All results are rounded, not truncated. More precisely, the primitive T<sub>E</sub>X operations `\divide` and `\multiply` are not used. The only instance where one will observe an effect is when dividing integers.

This version of `l3calc` is a now a complete replacement for the original `calc` package providing the same functionality and will prevent the original `calc` package from loading.

---

\*This file has version number 1776, last revised 2010/02/07.

†Frank Mittelbach, Denys Duchier, Chris Rowley, Rainer Schöpf, Johannes Braams, Michael Downes, David Carlisle, Alan Jeffrey, Morten Høgholm, Thomas Lotze, Javier Bezos, Will Robertson, Joseph Wright

## 1 User functions

<pre>\maxof \minof \widthof \heightof \depthof \totalheightof \ratio \real \setlength \gsetlength \addtolength \gaddtolength \setcounter \addtocounter \stepcounter</pre>
---

See documentation for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> package `calc`.

<pre>\calc_maxof:nn \calc_minof:nn \calc_widthof:n \calc_heightof:n \calc_depthof:n \calc_totalheightof:n \calc_ratio:nn \calc_real:n \calc_setcounter:nn \calc_addtocounter:nn \calc_stepcounter:n</pre>
---

Equivalent commands as the above in the `expl3` namespace.

<pre>\calc_int_set:Nn \calc_int_gset:Nn \calc_int_add:Nn \calc_int_gadd:Nn \calc_int_sub:Nn \calc_int_gsub:Nn</pre>
---

`\calc_int_set:Nn <int> {<calc expression>}`

Evaluates *<calc expression>* and either adds or subtracts it from *<int>* or sets *<int>* to it. These operations can also be global.

<code>\calc_dim_set:Nn</code>
<code>\calc_dim_gset:Nn</code>
<code>\calc_dim_add:Nn</code>
<code>\calc_dim_gadd:Nn</code>
<code>\calc_dim_sub:Nn</code>
<code>\calc_dim_gsub:Nn</code>

`\calc_dim_set:Nn <dim> {<calc expression>}`

Evaluates  $\langle calc\ expression \rangle$  and either adds or subtracts it from  $\langle dim \rangle$  or sets  $\langle dim \rangle$  to it. These operations can also be global.

<code>\calc_skip_set:Nn</code>
<code>\calc_skip_gset:Nn</code>
<code>\calc_skip_add:Nn</code>
<code>\calc_skip_gadd:Nn</code>
<code>\calc_skip_sub:Nn</code>
<code>\calc_skip_gsub:Nn</code>

`\calc_skip_set:Nn <skip> {<calc expression>}`

Evaluates  $\langle calc\ expression \rangle$  and either adds or subtracts it from  $\langle skip \rangle$  or sets  $\langle skip \rangle$  to it. These operations can also be global.

<code>\calc_muskip_set:Nn</code>
<code>\calc_muskip_gset:Nn</code>
<code>\calc_muskip_add:Nn</code>
<code>\calc_muskip_gadd:Nn</code>
<code>\calc_muskip_sub:Nn</code>
<code>\calc_muskip_gsub:Nn</code>

`\calc_muskip_set:Nn <muskip> {<calc expression>}`

Evaluates  $\langle calc\ expression \rangle$  and either adds or subtracts it from  $\langle muskip \rangle$  or sets  $\langle muskip \rangle$  to it. These operations can also be global.

<code>\calc_calculate_box_size:nnn</code>
---

`\calc_calculate_box_size:nnn {<dim-set>} {<item1> <item2> ... <itemn>} {<contents>}`

Sets  $\langle contents \rangle$  in a temporary box `\l_tmpa_box`. Then  $\langle dim-set \rangle$  is put in front of a loop that inserts  $+ \langle item_i \rangle$  in front of `\l_tmpa_box` and this is evaluated. For instance, if we wanted to determine the total height of the text `xyz` and store it in `\l_tmpa_dim`, we would call it as.

```
\calc_calculate_box_size:nnn
  {\dim_set:Nn\l_tmpa_dim}{\box_ht:N\box_dp:N}{xyz}
```

Similarly, if we wanted the difference between height and depth, we could call it as

```
\calc_calculate_box_size:nnn
  {\dim_set:Nn\l_tmpa_dim}{\box_ht:N{-\box_dp:N}}{xyz}
```

## 2 l3calc implementation

### 2.1 Variables

```
\l_calc_expression_tl  
\g_calc_A_register  
\l_calc_B_register  
\l_calc_current_type_int  
\g_calc_A_int  
\l_calc_B_int  
\l_calc_C_int  
\g_calc_A_dim  
\l_calc_B_dim  
\l_calc_C_dim  
\g_calc_A_skip  
\l_calc_B_skip  
\l_calc_C_skip  
\g_calc_A_muskip  
\l_calc_B_muskip  
\l_calc_C_muskip
```

Internal registers.

## 2.2 Internal functions

```
\calc_assign_generic:NNNNnn
\calc_pre_scan:N
\calc_open:w
\calc_init_B:
\calc_numeric:
\calc_close:
\calc_post_scan:N
\calc_multiply:N
\calc_divide:N
\calc_generic_add_or_subtract:N
\calc_add:
\calc_subtract:
\calc_add_A_to_B:
\calc_subtract_A_from_B:
\calc_generic_multiply_or_divide:N
\calc_multiply_B_by_A:
\calc_divide_B_by_A:
\calc_multiply:
\calc_divide:
\calc_textsize:Nn
\calc_ratio_multiply:nn
\calc_ratio_divide:nn
\calc_real_evaluate:nn
\calc_real_multiply:n
\calc_real_divide:n
\calc_maxmin_operation:Nnn
\calc_maxmin_generic:Nnn
\calc_maxmin_div_or_mul:NNnn
\calc_maxmin_multiply:
\calc_maxmin_multiply:
\calc_error:N
\calc_chk_document_counter:nn
```

Awaiting better documentation :)

## 2.3 Module code

Since this is basically a re-worked version of the `calc` package, I haven't bothered with too many comments except for in the places where this package differs. This may (and should) change at some point.

We start by ensuring that the required packages are loaded.

```
1 \<*package>
2 \ProvidesExplPackage
```

```

3   {\filename}{\filedate}{\fileversion}{\filedescription}
4   \package_check_loaded_expl:
5   </package>
6   <*initex | package>

```

\l\_calc\_expression\_tl Here we define some registers and pointers we will need.

```

\g_calc_A_register
\l_calc_B_register
\l_calc_current_type_int
7 \tl_new:N \l_calc_expression_tl
8 \cs_new_nopar:Npn \g_calc_A_register{}
9 \cs_new_nopar:Npn \l_calc_B_register{}
10 \int_new:N \l_calc_current_type_int

```

\g\_calc\_A\_int For each type of register we will need three registers to do our manipulations.

```

\l_calc_B_int
\l_calc_C_int
\g_calc_A_dim
\l_calc_B_dim
\l_calc_C_dim
\g_calc_A_skip
\l_calc_B_skip
\l_calc_C_skip
\g_calc_A_muskip
\l_calc_B_muskip
\l_calc_C_muskip
11 \int_new:N \g_calc_A_int
12 \int_new:N \l_calc_B_int
13 \int_new:N \l_calc_C_int
14 \dim_new:N \g_calc_A_dim
15 \dim_new:N \l_calc_B_dim
16 \dim_new:N \l_calc_C_dim
17 \skip_new:N \g_calc_A_skip
18 \skip_new:N \l_calc_B_skip
19 \skip_new:N \l_calc_C_skip
20 \muskip_new:N \g_calc_A_muskip
21 \muskip_new:N \l_calc_B_muskip
22 \muskip_new:N \l_calc_C_muskip

```

\calc\_assign\_generic:NNNNnn The generic function. #1 is a number denoting which type we are doing. (0=int, 1=dim, 2=skip, 3=muskip), #2 = temp register A, #3 = temp register B, #4 is a function acting on #5 which is the register to be set. #6 is the calc expression. We do a little extra work so that \real and \ratio can still be used by the user.

```

23 \cs_new:Npn \calc_assign_generic:NNNNnn#1#2#3#4#5#6{
24   \cs_set_eq:NN\g_calc_A_register#2
25   \cs_set_eq:NN\l_calc_B_register#3
26   \int_set:Nn \l_calc_current_type_int {#1}
27   \group_begin:
28     \cs_set_eq:NN \real \calc_real:n
29     \cs_set_eq:NN \ratio\calc_ratio:nn
30     \tl_set:Nx\l_calc_expression_tl{#6}
31     \exp_after:wN
32   \group_end:
33   \exp_after:wN\calc_open:w\exp_after:wN(\l_calc_expression_tl !
34   \pref_global:D\g_calc_A_register\l_calc_B_register
35   \group_end:
36   #4{#5}\l_calc_B_register
37 }

```

A simpler version relying on \real and \ratio having our definition is

```

\cs_new:Npn \calc_assign_generic:NNNNnn#1#2#3#4#5#6{
  \cs_set_eq:NN\g_calc_A_register#2\cs_set_eq:NN\l_calc_B_register#3
  \int_set:Nn \l_calc_current_type_int {#1}
  \tl_set:Nx\l_calc_expression_tl{#6}
  \exp_after:wN\calc_open:w\exp_after:wN(\l_calc_expression_tl !
  \pref_global:D\g_calc_A_register\l_calc_B_register
  \group_end:
  #4{#5}\l_calc_B_register
}

```

\calc\_int\_set:Nn Here are the individual versions for the different register types. First integer registers.

```

\calc_int_gset:Nn
\calc_int_add:Nn 38 \cs_new_nopar:Npn\calc_int_set:Nn{
\calc_int_gadd:Nn 39 \calc_assign_generic:NNNNnn\c_zero\g_calc_A_int\l_calc_B_int\int_set:Nn
\calc_int_sub:Nn 40 }
\calc_int_gsub:Nn 41 \cs_new_nopar:Npn\calc_int_gset:Nn{
42 \calc_assign_generic:NNNNnn\c_zero\g_calc_A_int\l_calc_B_int\int_gset:Nn
43 }
44 \cs_new_nopar:Npn\calc_int_add:Nn{
45 \calc_assign_generic:NNNNnn\c_zero\g_calc_A_int\l_calc_B_int\int_add:Nn
46 }
47 \cs_new_nopar:Npn\calc_int_gadd:Nn{
48 \calc_assign_generic:NNNNnn\c_zero\g_calc_A_int\l_calc_B_int\int_gadd:Nn
49 }
50 \cs_new_nopar:Npn\calc_int_sub:Nn{
51 \calc_assign_generic:NNNNnn\c_zero\g_calc_A_int\l_calc_B_int\int_sub:Nn
52 }
53 \cs_new_nopar:Npn\calc_int_gsub:Nn{
54 \calc_assign_generic:NNNNnn\c_zero\g_calc_A_int\l_calc_B_int\int_gsub:Nn
55 }

```

\calc\_dim\_set:Nn Dimens.

```

\calc_dim_gset:Nn
\calc_dim_add:Nn 56 \cs_new_nopar:Npn\calc_dim_set:Nn{
\calc_dim_gadd:Nn 57 \calc_assign_generic:NNNNnn\c_one\g_calc_A_dim\l_calc_B_dim\dim_set:Nn
58 }
59 \cs_new_nopar:Npn\calc_dim_gset:Nn{
60 \calc_assign_generic:NNNNnn\c_one\g_calc_A_dim\l_calc_B_dim\dim_gset:Nn
61 }
62 \cs_new_nopar:Npn\calc_dim_add:Nn{
63 \calc_assign_generic:NNNNnn\c_one\g_calc_A_dim\l_calc_B_dim\dim_add:Nn
64 }
65 \cs_new_nopar:Npn\calc_dim_gadd:Nn{
66 \calc_assign_generic:NNNNnn\c_one\g_calc_A_dim\l_calc_B_dim\dim_gadd:Nn
67 }
68 \cs_new_nopar:Npn\calc_dim_sub:Nn{
69 \calc_assign_generic:NNNNnn\c_one\g_calc_A_int\l_calc_B_int\dim_sub:Nn
70 }
71 \cs_new_nopar:Npn\calc_dim_gsub:Nn{

```

```

72 \calc_assign_generic:NNNNnn\c_one\g_calc_A_int\l_calc_B_int\dim_gsub:Nn
73 }

```

\calc\_skip\_set:Nn Skips.

```

\calc_skip_gset:Nn
\calc_skip_add:Nn
\calc_skip_gadd:Nn
\calc_skip_sub:Nn
\calc_skip_gsub:Nn
74 \cs_new_nopar:Npn\calc_skip_set:Nn{
75   \calc_assign_generic:NNNNnn\c_two\g_calc_A_skip\l_calc_B_skip\skip_set:Nn
76 }
77 \cs_new_nopar:Npn\calc_skip_gset:Nn{
78   \calc_assign_generic:NNNNnn\c_two\g_calc_A_skip\l_calc_B_skip\skip_gset:Nn
79 }
80 \cs_new_nopar:Npn\calc_skip_add:Nn{
81   \calc_assign_generic:NNNNnn\c_two\g_calc_A_skip\l_calc_B_skip\skip_add:Nn
82 }
83 \cs_new_nopar:Npn\calc_skip_gadd:Nn{
84   \calc_assign_generic:NNNNnn\c_two\g_calc_A_skip\l_calc_B_skip\skip_gadd:Nn
85 }
86 \cs_new_nopar:Npn\calc_skip_sub:Nn{
87   \calc_assign_generic:NNNNnn\c_two\g_calc_A_skip\l_calc_B_skip\skip_sub:Nn
88 }
89 \cs_new_nopar:Npn\calc_skip_gsub:Nn{
90   \calc_assign_generic:NNNNnn\c_two\g_calc_A_skip\l_calc_B_skip\skip_gsub:Nn
91 }

```

\calc\_muskip\_set:Nn Muskips.

```

\calc_muskip_gset:Nn
\calc_muskip_add:Nn
\calc_muskip_gadd:Nn
\calc_muskip_sub:Nn
\calc_muskip_gsub:Nn
92 \cs_new_nopar:Npn\calc_muskip_set:Nn{
93   \calc_assign_generic:NNNNnn\c_three\g_calc_A_muskip\l_calc_B_muskip
94   \muskip_set:Nn
95 }
96 \cs_new_nopar:Npn\calc_muskip_gset:Nn{
97   \calc_assign_generic:NNNNnn\c_three\g_calc_A_muskip\l_calc_B_muskip
98   \muskip_gset:Nn
99 }
100 \cs_new_nopar:Npn\calc_muskip_add:Nn{
101   \calc_assign_generic:NNNNnn\c_three\g_calc_A_muskip\l_calc_B_muskip
102   \muskip_add:Nn
103 }
104 \cs_new_nopar:Npn\calc_muskip_gadd:Nn{
105   \calc_assign_generic:NNNNnn\c_three\g_calc_A_muskip\l_calc_B_muskip
106   \muskip_gadd:Nn
107 }
108 \cs_new_nopar:Npn\calc_muskip_sub:Nn{
109   \calc_assign_generic:NNNNnn\c_three\g_calc_A_muskip\l_calc_B_muskip
110   \muskip_sub:Nn
111 }
112 \cs_new_nopar:Npn\calc_muskip_gsub:Nn{
113   \calc_assign_generic:NNNNnn\c_three\g_calc_A_muskip\l_calc_B_muskip
114   \muskip_gsub:Nn
115 }

```



`\calc_pre_scan:N` In case we found one of the special operations, this should just be executed.

```

116 \cs_new_nopar:Npn \calc_pre_scan:N #1{
117   \if_meaning:w(#1
118     \exp_after:wN\calc_open:w
119   \else:
120     \if_meaning:w \calc_textsize:Nn #1
121   \else:
122     \if_meaning:w \calc_maxmin_operation:Nnn #1
123   \else:

```

`\calc_numeric:` uses a primitive assignment so doesn't care about these dangling `\fi:s`.

```

124     \calc_numeric:
125     \fi:
126   \fi:
127 \fi:
128 #1}

```

`\calc_open:w`

```

129 \cs_new_nopar:Npn \calc_open:w({
130   \group_begin:\group_execute_after:N\calc_init_B:
131   \group_begin:\group_execute_after:N\calc_init_B:
132   \calc_pre_scan:N
133 }

```

`\calc_init_B:`

`\calc_numeric:`

`\calc_close:`

```

134 \cs_new_nopar:Npn\calc_init_B:{\l_calc_B_register\g_calc_A_register}
135 \cs_new_nopar:Npn\calc_numeric:{
136   \tex_afterassignment:D\calc_post_scan:N
137   \pref_global:D\g_calc_A_register
138 }
139 \cs_new_nopar:Npn\calc_close:{
140   \group_end:\pref_global:D\g_calc_A_register\l_calc_B_register
141   \group_end:\pref_global:D\g_calc_A_register\l_calc_B_register
142   \calc_post_scan:N}

```

`\calc_post_scan:N` Look at what token we have and decide where to go.

```

143 \cs_new_nopar:Npn\calc_post_scan:N#1{
144   \if_meaning:w#1!\cs_set_eq:NN\calc_next:w\group_end: \else:
145     \if_meaning:w#1+\cs_set_eq:NN\calc_next:w\calc_add: \else:
146       \if_meaning:w#1-\cs_set_eq:NN\calc_next:w\calc_subtract: \else:
147         \if_meaning:w#1*\cs_set_eq:NN\calc_next:w\calc_multiply:N \else:
148           \if_meaning:w#1/\cs_set_eq:NN\calc_next:w\calc_divide:N \else:
149             \if_meaning:w#1)\cs_set_eq:NN\calc_next:w\calc_close: \else:
150               \if_meaning:w#1\scan_stop:\cs_set_eq:NN\calc_next:w\calc_post_scan:N
151             \else:

```

If we get here, there is an error but let's also disable `\calc_next:w` since it is otherwise undefined. No need to give extra errors just for that.

```

152             \cs_set_eq:NN \calc_next:w \prg_do_nothing:
153             \calc_error:N#1
154         \fi:
155     \fi:
156 \fi:
157 \fi:
158 \fi:
159 \fi:
160 \fi:
161 \calc_next:w}

```

`\calc_multiply:N`    The switches for multiplication and division.  
`\calc_divide:N`

```

162 \cs_new_nopar:Npn \calc_multiply:N #1{
163     \if_meaning:w \calc_maxmin_operation:Nnn #1
164     \cs_set_eq:NN \calc_next:w \calc_maxmin_multiply:
165 \else:
166     \if_meaning:w \calc_ratio_multiply:nn #1
167     \cs_set_eq:NN \calc_next:w \calc_ratio_multiply:nn
168 \else:
169     \if_meaning:w \calc_real_evaluate:nn #1
170     \cs_set_eq:NN \calc_next:w \calc_real_multiply:n
171 \else:
172     \cs_set_nopar:Npn \calc_next:w{\calc_multiply: #1}
173 \fi:
174 \fi:
175 \fi:
176 \calc_next:w
177 }
178 \cs_new_nopar:Npn \calc_divide:N #1{
179     \if_meaning:w \calc_maxmin_operation:Nnn #1
180     \cs_set_eq:NN \calc_next:w \calc_maxmin_divide:
181 \else:
182     \if_meaning:w \calc_ratio_multiply:nn #1
183     \cs_set_eq:NN \calc_next:w \calc_ratio_divide:nn
184 \else:
185     \if_meaning:w \calc_real_evaluate:nn #1
186     \cs_set_eq:NN \calc_next:w \calc_real_divide:n
187 \else:
188     \cs_set_nopar:Npn \calc_next:w{\calc_divide: #1}
189 \fi:
190 \fi:
191 \fi:
192 \calc_next:w
193 }

```

`\calc_generic_add_or_subtract:N`    Here is how we add and subtract.

```

        \calc_add:
        \calc_subtract:
        \calc_add_A_to_B:
\calc_subtract_A_from_B:

```

```

194 \cs_new_nopar:Npn\calc_generic_add_or_subtract:N#1{
195   \group_end:
196   \pref_global:D\g_calc_A_register\l_calc_B_register\group_end:
197   \group_begin:\group_execute_after:N#1\group_begin:
198   \group_execute_after:N\calc_init_B:
199   \calc_pre_scan:N}
200 \cs_new_nopar:Npn\calc_add:{\calc_generic_add_or_subtract:N\calc_add_A_to_B:}
201 \cs_new_nopar:Npn\calc_subtract:{
202   \calc_generic_add_or_subtract:N\calc_subtract_A_from_B:}

```

Don't use `\tex_advance:D` since it allows overflows.

```

203 \cs_new_nopar:Npn\calc_add_A_to_B:{
204   \l_calc_B_register
205   \if_case:w\l_calc_current_type_int
206   \etex_numexpr:D\or:
207   \etex_dimexpr:D\or:
208   \etex_glueexpr:D\or:
209   \etex_muexpr:D\fi:
210   \l_calc_B_register + \g_calc_A_register\scan_stop:
211 }
212 \cs_new_nopar:Npn\calc_subtract_A_from_B:{
213   \l_calc_B_register
214   \if_case:w\l_calc_current_type_int
215   \etex_numexpr:D\or:
216   \etex_dimexpr:D\or:
217   \etex_glueexpr:D\or:
218   \etex_muexpr:D\fi:
219   \l_calc_B_register - \g_calc_A_register\scan_stop:
220 }

```

`\calc_generic_multiply_or_divide:N` And here is how we multiply and divide. Note that we do not use the primitive  $\TeX$  operations but the expandable operations provided by  $\varepsilon\text{-}\TeX$ . This means that all results are rounded not truncated!

```

\calc_multiply_B_by_A:
\calc_divide_B_by_A:
\calc_multiply:
\calc_divide:
221 \cs_new_nopar:Npn\calc_generic_multiply_or_divide:N#1{
222   \group_end:
223   \group_begin:
224   \cs_set_eq:NN\g_calc_A_register\g_calc_A_int
225   \cs_set_eq:NN\l_calc_B_register\l_calc_B_int
226   \int_zero:N \l_calc_current_type_int
227   \group_execute_after:N#1\calc_pre_scan:N
228 }
229 \cs_new_nopar:Npn\calc_multiply_B_by_A:{
230   \l_calc_B_register
231   \if_case:w\l_calc_current_type_int
232   \etex_numexpr:D\or:
233   \etex_dimexpr:D\or:
234   \etex_glueexpr:D\or:
235   \etex_muexpr:D\fi:

```

```

236 \l_calc_B_register*\g_calc_A_int\scan_stop:
237 }
238 \cs_new_nopar:Npn\calc_divide_B_by_A:{
239 \l_calc_B_register
240 \if_case:w\l_calc_current_type_int
241 \etex_numexpr:D\or:
242 \etex_dimexpr:D\or:
243 \etex_glueexpr:D\or:
244 \etex_muexpr:D\fi:
245 \l_calc_B_register/\g_calc_A_int\scan_stop:
246 }
247 \cs_new_nopar:Npn\calc_multiply:{
248 \calc_generic_multiply_or_divide:N\calc_multiply_B_by_A:}
249 \cs_new_nopar:Npn\calc_divide:{
250 \calc_generic_multiply_or_divide:N\calc_divide_B_by_A:}

```

`\calc_calculate_box_size:nnn` Put something in a box and measure it. #1 is a list of `\box_ht:N` etc., #2 should be `\dim_set:Nn<dim register>` or `\dim_gset:Nn<dim register>` and #3 is the contents.

```

251 \cs_new:Npn \calc_calculate_box_size:nnn #1#2#3{
252 \hbox_set:Nn \l_tmpa_box {{#3}}
253 #2{\c_zero_dim \tl_map_function:nN{#1}\calc_calculate_box_size_aux:n}
254 }

```

Helper for calculating the final dimension.

```

255 \cs_set_nopar:Npn \calc_calculate_box_size_aux:n#1{ + #1\l_tmpa_box}

```

`\calc_textsize:Nn` Now we can define `\calc_textsize:Nn`.

```

256 \cs_set_protected:Npn \calc_textsize:Nn#1#2{
257 \group_begin:
258 \cs_set_eq:NN\calc_widthof_aux:n\box_wd:N
259 \cs_set_eq:NN\calc_heightof_aux:n\box_ht:N
260 \cs_set_eq:NN\calc_depthof_aux:n\box_dp:N
261 \cs_set_nopar:Npn\calc_totalheightof_aux:n{\box_ht:N\box_dp:N}
262 \exp_args:No\calc_calculate_box_size:nnn{#1}
263 {\dim_gset:Nn\g_calc_A_register}

```

Restore the four user commands here since there might be a recursive call.

```

264 {
265 \cs_set_eq:NN \calc_depthof_aux:n \calc_depthof_auxi:n
266 \cs_set_eq:NN \calc_widthof_aux:n \calc_widthof_auxi:n
267 \cs_set_eq:NN \calc_heightof_aux:n \calc_heightof_auxi:n
268 \cs_set_eq:NN \calc_totalheightof_aux:n \calc_totalheightof_auxi:n
269 #2
270 }
271 \group_end:
272 \calc_post_scan:N
273 }

```

`\calc_ratio_multiply:nn` Evaluate a ratio. If we were already evaluation a  $\langle muskip \rangle$  register, the ratio is probably  
`\calc_ratio_divide:nn` also done with this type and we'll have to convert them to regular points.

```

274 \cs_set_protected:Npn\calc_ratio_multiply:nn#1#2{
275   \group_end:\group_begin:
276   \if_num:w\l_calc_current_type_int < \c_three
277     \calc_dim_set:Nn\l_calc_B_int{#1}
278     \calc_dim_set:Nn\l_calc_C_int{#2}
279   \else:
280     \calc_dim_muskip:Nn{\l_calc_B_int\etex_mutogluue:D}{#1}
281     \calc_dim_muskip:Nn{\l_calc_C_int\etex_mutogluue:D}{#2}
282   \fi:

```

Then store the ratio as a fraction, which we just pass on.

```

283   \cs_gset_nopar:Npx\calc_calculated_ratio:{
284     \int_use:N\l_calc_B_int/\int_use:N\l_calc_C_int
285   }
286   \group_end:

```

Here we set the new value of `\l_calc_B_register` and remember to evaluate it as the correct type. Note that the intermediate calculation is a scaled operation (meaning the intermediate value is 64-bit) so we don't get into trouble when first multiplying by a large number and then dividing.

```

287   \l_calc_B_register
288   \if_case:w\l_calc_current_type_int
289     \etex_numexpr:D\or:
290     \etex_dimexpr:D\or:
291     \etex_glueexpr:D\or:
292     \etex_muexpr:D\fi:
293   \l_calc_B_register*\calc_calculated_ratio:\scan_stop:
294   \group_begin:
295   \calc_post_scan:N}

```

Division is just flipping the arguments around.

```

296 \cs_new:Npn \calc_ratio_divide:nn#1#2{\calc_ratio_multiply:nn{#2}{#1}}

```

`\calc_real_evaluate:nn` Although we could define the `\real` function as a subcase of `\ratio`, this is horribly  
`\calc_real_multiply:n` inefficient since we just want to convert the decimal to a fraction.  
`\calc_real_divide:n`

```

297 \cs_new_protected_nopar:Npn\calc_real_evaluate:nn #1#2{
298   \group_end:
299   \l_calc_B_register
300   \if_case:w\l_calc_current_type_int
301     \etex_numexpr:D\or:
302     \etex_dimexpr:D\or:
303     \etex_glueexpr:D\or:
304     \etex_muexpr:D\fi:

```

```

305 \l_calc_B_register *
306 \tex_number:D \dim_eval:n{#1pt}/
307 \tex_number:D\dim_eval:n{#2pt}
308 \scan_stop:
309 \group_begin:
310 \calc_post_scan:N}
311 \cs_new_nopar:Npn \calc_real_multiply:n #1{\calc_real_evaluate:nn{#1}{1}}
312 \cs_new_nopar:Npn \calc_real_divide:n {\calc_real_evaluate:nn{1}}

```

\calc\_maxmin\_operation:Nnn The max and min functions.

```

\calc_maxmin_generic:Nnn
\calc_maxmin_div_or_mul:NNnn
\calc_maxmin_multiply:
\calc_maxmin_multiply:
313 \cs_set_protected:Npn\calc_maxmin_operation:Nnn#1#2#3{
314 \group_begin:
315 \calc_maxmin_generic:Nnn#1{#2}{#3}
316 \group_end:
317 \calc_post_scan:N
318 }

```

#1 is either > or < and was expanded into this initially.

```

319 \cs_new_protected:Npn \calc_maxmin_generic:Nnn#1#2#3{
320 \group_begin:
321 \if_case:w\l_calc_current_type_int
322 \calc_int_set:Nn\l_calc_C_int{#2}%
323 \calc_int_set:Nn\l_calc_B_int{#3}%
324 \pref_global:D\g_calc_A_register
325 \if_num:w\l_calc_C_int#1\l_calc_B_int
326 \l_calc_C_int\else:\l_calc_B_int\fi:
327 \or:
328 \calc_dim_set:Nn\l_calc_C_dim{#2}%
329 \calc_dim_set:Nn\l_calc_B_dim{#3}%
330 \pref_global:D\g_calc_A_register
331 \if_dim:w\l_calc_C_dim#1\l_calc_B_dim
332 \l_calc_C_dim\else:\l_calc_B_dim\fi:
333 \or:
334 \calc_skip_set:Nn\l_calc_C_skip{#2}%
335 \calc_skip_set:Nn\l_calc_B_skip{#3}%
336 \pref_global:D\g_calc_A_register
337 \if_dim:w\l_calc_C_skip#1\l_calc_B_skip
338 \l_calc_C_skip\else:\l_calc_B_skip\fi:
339 \else:
340 \calc_muskip_set:Nn\l_calc_C_muskip{#2}%
341 \calc_muskip_set:Nn\l_calc_B_muskip{#3}%
342 \pref_global:D\g_calc_A_register
343 \if_dim:w\l_calc_C_muskip#1\l_calc_B_muskip
344 \l_calc_C_muskip\else:\l_calc_B_muskip\fi:
345 \fi:
346 \group_end:
347 }
348 \cs_new:Npn\calc_maxmin_div_or_mul:NNnn#1#2#3#4{

```

```

349 \group_end:
350 \group_begin:
351 \int_zero:N\l_calc_current_type_int
352 \group_execute_after:N#1
353 \calc_maxmin_generic:Nnn#2{#3}{#4}
354 \group_end:
355 \group_begin:
356 \calc_post_scan:N
357 }
358 \cs_new_nopar:Npn\calc_maxmin_multiply:{
359   \calc_maxmin_div_or_mul:NNnn\calc_multiply_B_by_A:}
360 \cs_new_nopar:Npn\calc_maxmin_divide: {
361   \calc_maxmin_div_or_mul:NNnn\calc_divide_B_by_A:}

```

`\calc_error:N` The error message.

```

362 \cs_new_nopar:Npn\calc_error:N#1{
363   \PackageError{calc}
364   {'\token_to_str:N#1'~ invalid~ at~ this~ point}
365   {I~ expected~ to~ see~ one~ of:~ +~ -- ~*~ /~ )}
366 }

```

## 2.4 Higher level commands

The various operations allowed.

`\calc_maxof:nn` Max and min operations

`\calc_minof:nn`

`\maxof`

`\minof`

```

367 \cs_new:Npn \calc_maxof:nn#1#2{
368   \calc_maxmin_operation:Nnn > \exp_not:n{{#1}{#2}}
369 }
370 \cs_new:Npn \calc_minof:nn#1#2{
371   \calc_maxmin_operation:Nnn < \exp_not:n{{#1}{#2}}
372 }
373 \cs_set_eq:NN \maxof \calc_maxof:nn
374 \cs_set_eq:NN \minof \calc_minof:nn

```

`\calc_widthof:n` Text dimension commands.

`\calc_widthof_aux:n`

`\calc_widthof_auxi:n`

`\calc_heightof:n`

`\calc_heightof_aux:n`

`\calc_heightof_auxi:n`

`\calc_depthof:n`

`\calc_depthof_aux:n`

`\calc_depthof_auxi:n`

`\calc_totalheightof:n`

`\calc_totalheightof_aux:n`

`\calc_totalheightof_auxi:n`

```

375 \cs_new:Npn \calc_widthof:n#1{
376   \calc_textsize:Nn \exp_not:N\calc_widthof_aux:n\exp_not:n{{#1}}
377 }
378 \cs_new:Npn \calc_heightof:n#1{
379   \calc_textsize:Nn \exp_not:N\calc_heightof_aux:n\exp_not:n{{#1}}
380 }
381 \cs_new:Npn \calc_depthof:n#1{
382   \calc_textsize:Nn \exp_not:N\calc_depthof_aux:n\exp_not:n{{#1}}
383 }

```

```

384 \cs_new:Npn \calc_totalheightof:n#1{
385   \calc_textsize:Nn \exp_not:N\calc_totalheightof_aux:n \exp_not:n{{#1}}
386 }
387 \cs_new:Npn \calc_widthof_aux:n #1{
388   \exp_not:N\calc_widthof_aux:n\exp_not:n{{#1}}
389 }
390 \cs_new_eq:NN \calc_widthof_auxi:n \calc_widthof_aux:n
391 \cs_new:Npn \calc_depthof_aux:n #1{
392   \exp_not:N\calc_depthof_aux:n\exp_not:n{{#1}}
393 }
394 \cs_new_eq:NN \calc_depthof_auxi:n \calc_depthof_aux:n
395 \cs_new:Npn \calc_heightof_aux:n #1{
396   \exp_not:N\calc_heightof_aux:n\exp_not:n{{#1}}
397 }
398 \cs_new_eq:NN \calc_heightof_auxi:n \calc_heightof_aux:n
399 \cs_new:Npn \calc_totalheightof_aux:n #1{
400   \exp_not:N\calc_totalheightof_aux:n\exp_not:n{{#1}}
401 }
402 \cs_new_eq:NN \calc_totalheightof_auxi:n \calc_totalheightof_aux:n

```

\calc\_ratio:nn Ratio and real.

\calc\_real:n

```

403 \cs_new:Npn \calc_ratio:nn#1#2{
404   \calc_ratio_multiply:nn\exp_not:n{{#1}{#2}}}
405 \cs_new_nopar:Npn \calc_real:n {\calc_real_evaluate:nn}

```

We can implement real and ratio without actually using these names. We'll see.

\widthof User commands.

\heightof

\depthof

\totalheightof

\ratio

\real

```

406 \cs_set_eq:NN \depthof\calc_depthof:n
407 \cs_set_eq:NN \widthof\calc_widthof:n
408 \cs_set_eq:NN \heightof\calc_heightof:n
409 \cs_set_eq:NN \totalheightof\calc_totalheightof:n
410 %%\cs_set_eq:NN \ratio\calc_ratio:nn
411 %%\cs_set_eq:NN \real\calc_real:n

```

\setlength

\gsetlength

\addtolength

\gaddtolength

```

412 \cs_set_protected_nopar:Npn \setlength{\calc_skip_set:Nn}
413 \cs_set_protected_nopar:Npn \gsetlength{\calc_skip_gset:Nn}
414 \cs_set_protected_nopar:Npn \addtolength{\calc_skip_add:Nn}
415 \cs_set_protected_nopar:Npn \gaddtolength{\calc_skip_gadd:Nn}

```

\calc\_setcounter:nn

\calc\_addtocounter:nn

\calc\_stepcounter:n

\setcounter

\addtocounter

\stepcounter

\calc\_chk\_document\_counter:nn

Document commands for L<sup>A</sup>T<sub>E</sub>X 2<sub>ε</sub> counters. Also add support for amstext. Note that when l3breqn is used, \mathchoice will no longer need this switch as the argument is only executed once.



```

416 </initex | package>
417 <*package>
418 \newif\iffirstchoice@ \firstchoice@true
419 </package>
420 <*initex | package>
421 \cs_set_protected_nopar:Npn \calc_setcounter:nn#1#2{
422   \calc_chk_document_counter:nn{#1}{
423     \exp_args:Nc\calc_int_gset:Nn {c@#1}{#2}
424   }
425 }
426 \cs_set_protected_nopar:Npn \calc_addtocounter:nn#1#2{
427 </initex | package>
428 <*package>
429 \iffirstchoice@
430 </package>
431 <*initex | package>
432   \calc_chk_document_counter:nn{#1}{
433     \exp_args:Nc\calc_int_gadd:Nn {c@#1}{#2}
434   }
435 </initex | package>
436 <*package>
437 \fi:
438 </package>
439 <*initex | package>
440 }
441 \cs_set_protected_nopar:Npn \calc_stepcounter:n#1{
442 </initex | package>
443 <*package>
444 \iffirstchoice@
445 </package>
446 <*initex | package>
447   \calc_chk_document_counter:nn{#1}{
448     \int_gincr:c {c@#1}
449     \group_begin:
450       \cs_set_eq:NN \@elt\@stpelt \use:c{cl@#1}
451     \group_end:
452   }
453 </initex | package>
454 <*package>
455 \fi:
456 </package>
457 <*initex | package>
458 }
459 \cs_new_nopar:Npn \calc_chk_document_counter:nn#1{
460   \cs_if_free:cTF{c@#1}{\@nocounterr {#1}}
461 }
462 \cs_set_eq:NN \setcounter \calc_setcounter:nn
463 \cs_set_eq:NN \addtocounter \calc_addtocounter:nn
464 \cs_set_eq:NN \stepcounter \calc_stepcounter:n
465 </initex | package>

```

```

466 \*package>
467 \AtBeginDocument{
468   \cs_set_eq:NN \setcounter \calc_setcounter:nn
469   \cs_set_eq:NN \addtocounter \calc_addtocounter:nn
470   \cs_set_eq:NN \stepcounter \calc_stepcounter:n
471 }

```

Prevent the usual calc from loading.

```

472 \cs_set_nopar:cpn{ver@calc.sty}{2005/08/06}
473 \*package>

474 \*showmemory>
475 \showMemUsage
476 \*showmemory>

```