

The Changebar package *

Michael Fine
Distributed Systems Architecture

Johannes Braams
Kersengarde 33
2723 BP Zoetermeer
The Netherlands
texniek at braams.cistron.nl

Printed April 15, 2004

| | | | |
|---|----------|--|----------|
| Contents | | 5 The implementation | 6 |
| 1 Introduction | 1 | 5.1 Declarations And Initializations | 6 |
| 2 The user interface | 2 | 5.2 Option Processing | 8 |
| 2.1 The package options | 2 | 5.3 User Level Commands And Parameters | 11 |
| 2.1.1 Specifying the printer driver | 2 | 5.4 Macros for beginning and ending bars | 17 |
| 2.1.2 Specifying the bar position | 3 | 5.5 Macros for Making It Work Across Page Breaks | 21 |
| 2.1.3 Color | 3 | 5.6 Macros For Managing The Stacks of Bar points . | 23 |
| 2.1.4 Tracing | 3 | 5.7 Macros For Checking That The .aux File Is Stable | 24 |
| 2.2 Macros defined by the package | 3 | 5.8 Macros For Making It Work With Nested Floats/Footnotes | 26 |
| 2.3 Changebar parameters | 3 | | |
| 3 Deficiencies and bugs | 4 | | |
| 4 The basic algorithm | 5 | | |

Abstract

This package implements a way to indicate modifications in a L^AT_EX-document by putting bars in the margin. It realizes this by making use of the `\special` commands supported by ‘dvi drivers’. Currently six different drivers are supported. More can easily be added.

1 Introduction

Important note Just as with cross references and labels, you usually need to process the document twice (and sometimes three times) to ensure that the changebars

*This file has version number v3.4g, last revised 2004/02/20.

come out correctly. However, a warning will be given if another pass is required.

Features

- Changebars may be nested within each other. Each level of nesting can be given a different thickness bar.
- Changebars may be nested in other environments including floats and footnotes.
- Changebars are applied to all the material within the “barred” environment, including floating bodies regardless of where the floats float to. An exception to this is margin floats.
- Changebars may cross page boundaries.
- Changebars can appear on the *outside* of the columns of `twocolumn` text.
- The colour of the changebars can be changed. This has sofar been tested with the `dvips` and `vtex` drivers, but it may also work with other PostScript based drivers. It will *not* work for the `DVItoLN03` and `emTeX` drivers. For colored changebars to work, make sure that you specify the option `color`.

2 The user interface

This package has options to specify some details of its operation, and also defines several macros.

2.1 The package options

2.1.1 Specifying the printer driver

One set of package options¹ specify the driver that will be used to print the document can be indicated. The driver may be one of:

- `DVItoLN03`
- `DVItoPS`
- `DVIps`
- `emTeX`
- `TeXtures`
- `VTeX`

The drivers are represented in the normal typewriter method of typing these names, or by the same entirely in lower case. Since version 3.4d the driver can be specified in a configuration file, not surprisingly called `changebar.cfg`. If it contains the command `\ExecuteOption{textures}` the `textures` option will be used for all documents that are processed while the configuration file is in `TeX`'s search path.

¹For older documents the command `\driver` is available in the preamble of the document. It takes the options as defined for `LATεX 2ε` as argument.

2.1.2 Specifying the bar position

The position of the bars may either be on the inner edge of the page (the left column on a recto or single-sided page, the right column of a verso page) by use of the `innerbars` package option (the default), or on the outer edge of the page by use of the `outerbars` package option.

Another set of options gives the user the possibility of specifying that the bars should *always* come out on the left side of the text (`leftbars`) or on the right side of the text (`rightbars`).

Note that these options only work for *onecolumn* documents and will be ignored for a *twocolumn* document.

2.1.3 Color

For people who want their changebars to be colourful the option `color` is available. It defines the user command `\cbcolor` and loads the `color` package.

If a configuration file specifies the `color` option and you want to override it for a certain document you can use the `grey` option.

2.1.4 Tracing

The package also implements tracing for its own debugging. The package options `traceon` and `traceoff` control tracing. An additional option `tracestacks` is available for the die hard who wants to know what goes on in the internal stacks maintained by this package.

2.2 Macros defined by the package

`\cbstart` `\cbend` All material between the macros `\cbstart` and `\cbend` is barred. The nesting of multiple changebars is allowed. The macro `\cbstart` has an optional parameter that specifies the width of the bar. The syntax is `\cbstart[⟨dimension⟩]`. If no width is specified, the current value of the parameter `\changebarwidth` is used. Note that `\cbstart` and `\cbend` can be used anywhere but must be correctly nested with floats and footnotes. That is, one cannot have one end of the bar inside a floating insertion and the other outside, but that would be a meaningless thing to do anyhow.

`changebar` Apart from the macros `\cbstart` and `\cbend` a proper \LaTeX environment is defined. The advantage of using the environment whenever possible is that \LaTeX will do all the work of checking the correct nesting of different environments.

`\cbdelete` The macro `\cbdelete` puts a square bar in the margin to indicate that some text was removed from the document. The macro has an optional argument to specify the width of the bar. When no argument is specified the current value of the parameter `\deletebarwidth` will be used.

`\nochangebars` The macro `\nochangebars` disables the changebar commands.

`\cbcolor` This macro is defined when the `color` option is selected. Its syntax is the same as the `\color` command from the `color` package.

2.3 Changebar parameters

`\changebarwidth` The width of the changebars is controlled with the \LaTeX length parameter `\changebarwidth`. Its value can be changed with the `\setlength` command.

Changing the value of `\changebarwidth` affects all subsequent changebars subject to the scoping rules of `\setlength`.

`\deletebarwidth` The width of the deletebars is controlled with the \LaTeX length parameter `\deletebarwidth`. Its value can be changed with the `\setlength` command. Changing the value of `\deletebarwidth` affects all subsequent deletebars subject to the scoping rules of `\setlength`.

`\changebarsep` The separation between the text and the changebars is determined by the value of the \LaTeX length parameter `\changebarsep`.

`changebargrey` When one of the supported dvi to PostScript translators is used the ‘blackness’ of the bars can be controlled. The \LaTeX counter `changebargrey` is used for this purpose. Its value can be changed with a command like:

```
\setcounter{changebargrey}{85}
```

The value of the counter is a percentage, where the value 0 yields black bars, the value 100 yields white bars.

`outerbars` The changebars will be printed in the ‘inside’ margin of your document. This means they appear on the left side of the page. When `twoside` is in effect the bars will be printed on the right side of even pages. This behaviour can be changed by including the command `\outerbarstrue` in your document.

3 Deficiencies and bugs

- The macros blindly use special points `\cb@minpoint` through `\cb@maxpoint`. If this conflicts with another set of macros, the results will be unpredictable. (What is really needed is a `\newspecialpoint`, analogous to `\newcount` etc. — it’s not provided because the use of the points is rather rare.)
- There is a limit of $(\cb@maxpoint - \cb@minpoint + 1) / 4$ bars per page (four special points per bar). Using more than this number yields unpredictable results (but that could be called a feature for a page with so many bars). This limitation could be increased if desired.
- Internal macro names are all of the form `\cb@xxxx`. No checking for conflicts with other macros is done.
- This implementation does not work with the `multicolumn` package.
- The algorithms may fail if a floating insertion is split over multiple pages. In \LaTeX floats are not split but footnotes may be. The simplest fix to this is to prevent footnotes from being split but this may make \TeX very unhappy.
- The `\cbend` normally gets “attached” to the token after it rather than the one before it. This may lead to a longer bar than intended. For example, consider the sequence ‘word1 `\cbend` word2’. If there is a line break between ‘word1’ and ‘word2’ the bar will incorrectly be extended an extra line. This particular case can be fixed with the incantation ‘word1`\cbend{}` word2’.
- The colour support has only been tested with the `dvips` driver.

4 The basic algorithm

The changebars are implemented using the `\specials` of various `dvi` interpreting programs like `DVItO LN03` or `DVIps`. In essence, the start of a changebar defines two `\special` points in the margins at the current vertical position on the page. The end of a changebar defines another set of two points and then joins (using the “connect” `\special`) either the two points to the left or the two points to the right of the text, depending on the setting of `innerbars`, `outerbars`, `leftbars`, `rightbars` and/or `twoside`.

This works fine as long as the two points being connected lie on the same page. However, if they don't, the bar must be artificially terminated at the page break and restarted at the top of the next page. The only way to do this (that I can think of) is to modify the output routine so that it checks if any bar is in progress when it ships out a page and, if so, adds the necessary artificial end and begin.

The obvious way to indicate to the output routine that a bar is in progress is to set a flag when the bar is begun and to unset this flag when the bar is ended. This works most of the time but, because of the asynchronous behavior of the output routine, errors occur if the bar begins or ends near a page break. To illustrate, consider the following scenario.

```
blah blah blah           % page n
blah blah blah
\cbstart                 % this does its thing and set the flag
more blah
                          <----- pagebreak occurs here
more blah
\cbend                   % does its thing and unsets flag
blah blah
```

Since `TEX` processes ahead of the page break before invoking the output routine, it is possible that the `\cbend` is processed, and the flag unset, before the output routine is called. If this happens, special action is required to generate an artificial end and begin to be added to page n and $n + 1$ respectively, as it is not possible to use a flag to signal the output routine that a bar crosses a page break.

The method used by these macros is to create a stack of the beginning and end points of each bar in the document together with the page number corresponding to each point. Then, as a page is completed, a modified output routine checks the stack to determine if any bars begun on or before the current page are terminated on subsequent pages, and handles those bars appropriately. To build the stack, information about each changebar is written to the `.aux` file as bars are processed. This information is re-read when the document is next processed. Thus, to ensure that changebars are correct, the document must be processed twice. Luckily, this is generally required for `LATEX` anyway.

This approach is sufficiently general to allow nested bars, bars in floating insertions, and bars around floating insertions. Bars inside floats and footnotes are handled in the same way as bars in regular text. Bars that encompass floats or footnotes are handled by creating an additional bar that floats with the floating material. Modifications to the appropriate `LATEX` macros check for this condition and add the extra bar.

5 The implementation

5.1 Declarations And Initializations

- `\cb@maxpoint` The original version of `changebar.sty` only supported the DVItoLN03 specials. The LN03 printer has a maximum number of points that can be defined on a page. Also for some PostScript printers the number of points that can be defined can be limited by the amount of memory used. Therefore, the consecutive numbering of points has to be reset when the maximum is reached. This maximum can be adapted to the printers needs.
- ```
1 (*package)
2 \def\cb@maxpoint{80}
```
- `\cb@minpoint` When resetting the point number we need to know what to reset it to, this is minimum number is stored in `\cb@minpoint`. **This number has to be *odd*** because the algorithm that decides whether a bar has to be continued on the next page depends on this.
- ```
3 \def\cb@minpoint{1}
```
- `\cb@nil` Sometimes a void value for a point has to be returned by one of the macros. For this purpose `\cb@nil` is used.
- ```
4 \def\cb@nil{0}
```
- `\cb@nextpoint` The number of the next special point is stored in the count register `\cb@nextpoint` and initially equal to `\cb@minpoint`.
- ```
5 \newcount\cb@nextpoint
6 \cb@nextpoint=\cb@minpoint
```
- `\cb@topleft` `\cb@topright` `\cb@botleft` `\cb@botright` These four counters are used to identify the four special points that specify a changebar. The point defined by `\cb@topleft` is the one used to identify the changebar; the values of the other points are derived from it.
- ```
7 \newcount\cb@topleft
8 \newcount\cb@topright
9 \newcount\cb@botleft
10 \newcount\cb@botright
```
- `\cb@cnta` `\cb@cntb` Sometimes we need temporarily store a value. For this purpose two count registers and a dimension register are allocated.
- ```
11 \newcount\cb@cnta
12 \newcount\cb@cntb
13 \newdimen\cb@dima
```
- `\cb@curbarwd` The dimension register `\cb@curbarwd` is used to store the width of the current bar.
- ```
14 \newdimen\cb@curbarwd
```
- `\cb@page` `\cb@pagecount` The macros need to keep track of the number of pages output so far. To this end the counter `\cb@pagecount` is used. When a pagenumber is read from the history stack, it is stored in the counter `\cb@page`. The counter `\cb@pagecount` is initially 0; it gets incremented during the call to `\@makebox` (see section 5.5).
- ```
15 \newcount\cb@page
16 \newcount\cb@pagecount
17 \cb@pagecount=0
```

`outerbars` A switch is provided to control where the changebars will be printed.

```
18 \newif\ifouterbars
```

`@cb@trace` A switch to enable tracing of the actions of this package

```
19 \newif\if@cb@trace
```

`\cb@positions` This macro calculates the (horizontal) positions of the changebars.

`\cb@odd@left` Because the margins can differ for even and odd pages and because changebars
`\cb@odd@right` are sometimes on different sides of the paper we need four dimensions to store the
`\cb@even@left` result.
`\cb@even@right`

```
20 \newdimen\cb@odd@left
21 \newdimen\cb@odd@right
22 \newdimen\cb@even@left
23 \newdimen\cb@even@right
```

Since the changebars are drawn with the POSTSCRIPT command `lineto` and not as T_EX-like rules the reference points lie on the center of the changebar, therefore the calculation has to add or subtract half of the width of the bar to keep `\changebarsep` whitespace between the bar and the body text.

First the position for odd pages is calculated. I

```
24 \def\cb@positions{%
25   \global\cb@odd@left=\hoffset
26   \global\cb@even@left\cb@odd@left
27   \global\advance\cb@odd@left by \oddsidemargin
28   \global\cb@odd@right\cb@odd@left
29   \global\advance\cb@odd@right by \textwidth
30   \global\advance\cb@odd@right by \changebarsep
31   \global\advance\cb@odd@right by 0.5\changebarwidth
32   \global\advance\cb@odd@left by -\changebarsep
33   \global\advance\cb@odd@left by -0.5\changebarwidth
```

On even sided pages we need to use `\evensidemargin` in the calculations when `twoside` is in effect.

```
34 \if@twoside
35   \global\advance\cb@even@left by \evensidemargin
36   \global\cb@even@right\cb@even@left
37   \global\advance\cb@even@left by -\changebarsep
38   \global\advance\cb@even@left by -0.5\changebarwidth
39   \global\advance\cb@even@right by \textwidth
40   \global\advance\cb@even@right by \changebarsep
41   \global\advance\cb@even@right by 0.5\changebarwidth
42 \else
```

Otherwise just copy the result for odd pages.

```
43   \global\let\cb@even@left\cb@odd@left
44   \global\let\cb@even@right\cb@odd@right
45 \fi
46 }
```

`\cb@removedim` In PostScript code, length specifications are without dimensions. Therefore we need a way to remove the letters ‘pt’ from the result of the operation

`\the\langle dimen \rangle`. This can be done by defining a command that has a delimited argument like:

```
\def\cb@removedim#1pt{#1}
```

We encounter one problem though, the category code of the letters ‘pt’ is 12 when produced as the output from `\the\langle dimen \rangle`. Thus the characters that delimit the argument of `\cb@removedim` also have to have category code 12. To keep the changes local the macro `\cb@removedim` is defined in a group.

```
47 {\catcode'\p=12\catcode'\t=12 \gdef\cb@removedim#1pt{#1}}
```

5.2 Option Processing

The user should select the specials that should be used by specifying the driver name as an option to the `\usepackage` call. Possible choices are:

- DVItolN03
- DVItOPS
- DVIPs
- emT_EX
- Textures
- V_TE_X

The intent is that the driver names should be case-insensitive, but the following code doesn’t achieve this: it only permits the forms given above and their lower-case equivalents.

```
48 \DeclareOption{DVItolN03}{\global\chardef\cb@driver@setup=0\relax}
49 \DeclareOption{dvitoln03}{\global\chardef\cb@driver@setup=0\relax}
50 \DeclareOption{DVItOPS}{\global\chardef\cb@driver@setup=1\relax}
51 \DeclareOption{dvitops}{\global\chardef\cb@driver@setup=1\relax}
52 \DeclareOption{DVIPs}{\global\chardef\cb@driver@setup=2\relax}
53 \DeclareOption{dvips}{\global\chardef\cb@driver@setup=2\relax}
54 \DeclareOption{emTeX}{\global\chardef\cb@driver@setup=3\relax}
55 \DeclareOption{emtex}{\global\chardef\cb@driver@setup=3\relax}
56 \DeclareOption{textures}{\global\chardef\cb@driver@setup=4\relax}
57 \DeclareOption{Textures}{\global\chardef\cb@driver@setup=4\relax}
58 \DeclareOption{VTeX}{\global\chardef\cb@driver@setup=5\relax}
59 \DeclareOption{vtex}{\global\chardef\cb@driver@setup=5\relax}
```

The new features of L^AT_EX_{2 ϵ} make it possible to implement the `outerbars` option.

```
60 \DeclareOption{outerbars}{\outerbarstrue}
61 \DeclareOption{innerbars}{\outerbarsfalse}
```

It is also possible to specify that the change bars should *always* be printed on either the left or the right side of the text. For this we have the options `leftbars` and `rightbars`. Specifying *either* of these options will overrule a possible `twoside` option at the document level.

```
62 \DeclareOption{leftbars}{%
```

```

63 \def\cb@positions{%
64   \global\cb@odd@left=\hoffset
65   \global\cb@even@left\cb@odd@left
66   \global\advance\cb@odd@left by \oddsidemargin
67   \global\advance\cb@odd@left by -\changebarsep
68   \global\advance\cb@odd@left by -0.5\changebarwidth
69   \if@twoside
70     \global\advance\cb@even@left by \evensidemargin
71     \global\advance\cb@even@left by -\changebarsep
72     \global\advance\cb@even@left by -0.5\changebarwidth
73   \else
74     \global\let\cb@even@left\cb@odd@left
75   \fi
76   \global\let\cb@odd@right\cb@odd@left
77   \global\let\cb@even@right\cb@even@left
78 }
79 \DeclareOption{rightbars}{%
80   \def\cb@positions{%
81     \global\cb@odd@right=\hoffset
82     \global\cb@even@right\cb@odd@right
83     \global\advance\cb@odd@right by \oddsidemargin
84     \global\advance\cb@odd@right by \textwidth
85     \global\advance\cb@odd@right by \changebarsep
86     \global\advance\cb@odd@right by 0.5\changebarwidth
87     \if@twoside
88       \global\advance\cb@even@right by \evensidemargin
89       \global\advance\cb@even@right by \textwidth
90       \global\advance\cb@even@right by \changebarsep
91       \global\advance\cb@even@right by 0.5\changebarwidth
92     \else
93       \global\let\cb@even@right\cb@odd@right
94     \fi
95     \global\let\cb@odd@left\cb@odd@right
96     \global\let\cb@even@left\cb@even@right
97   }}

```

A set of options to control tracing.

```

98 \DeclareOption{traceton}{\cb@tracetrue}
99 \DeclareOption{traceoff}{\cb@tracefalse}
100 \DeclareOption{tracestacks}{%
101   \let\cb@trace@stack\cb@show@stack
102   \def\cb@trace@push#1{\cb@trace{%
103     Pushed point \the\cb@topleft\space on \noexpand#1: #1}}%
104   \def\cb@trace@pop#1{\cb@trace{%
105     Popped point \the\cb@topleft\space from \noexpand#1: #1}}%
106   }

```

Two options are introduced for colour support. The first one, `grey`, is activated by default.

```

107 \DeclareOption{grey}{%
108   \def\cb@ps@color{\thechangebargrey\space 100 div setgray}}
109 \DeclareOption{color}{%
110   \def\cb@ps@color{\expandafter\color\to@ps\cb@current@color\@@}}

```

Signal an error if an unknown option was specified.

```

111 \DeclareOption*{\OptionNotUsed\PackageError
112     {changebar}%
113     {Unrecognised option ‘\CurrentOption’\MessageBreak
114     known options are dviton03, dvitops, dvips,\MessageBreak
115     emtex, textures and vtex,
116     grey, color,\MessageBreak
117     outerbars, innerbars, leftbars and rightbars}}

```

The default is to have grey change bars on the left side of the text on odd pages. When \TeX is used the option `dvips` is not the right one, so in that case we have `vtex` as the default driver.

```

118 \ifx\VTexversion\undefined
119   \ExecuteOptions{innerbars,traceoff,dvips,grey}
120 \else
121   \ExecuteOptions{innerbars,traceoff,vtex,grey}
122 \fi

```

A local configuration file may be used to define a site wide default for the driver, by calling `\ExecuteOptions` with the appropriate option. This will override the default specified above.

```

123 \InputIfFileExists{changebar.cfg}{}{}

```

`\cb@@show@stack` When the stack tracing facility is turned on this command is executed. It needs to be defined *before* we call `\ProcessOptions`. This command shows the contents of the stack with currently ‘open’ bars, the stack with pending ends and the history stack. It does *not* show the temporary stack.

```

124 \def\cb@@show@stack#1{%
125   \cb@trace{%
126     stack status at #1:\MessageBreak
127     current stack: \cb@currentstack\MessageBreak
128     \@spaces end stack: \cb@endstack\MessageBreak
129     \space\space begin stack: \cb@beginstack\MessageBreak
130     history stack: \cb@historystack
131   }}

```

The default is to *not* trace the stacks. This is achieved by `\letting \cb@trace@stack` to `\@gobble`.

```

132 \let\cb@trace@stack\@gobble

```

`\cb@trace@push` When stack tracing is turned on, these macros are used to display the push and `\cb@trace@pop` pop operations that go on. They are defined when the package option `tracestacks` is selected.

The default is to *not* trace the stacks.

```

133 \let\cb@trace@push\@gobble
134 \let\cb@trace@pop\@gobble
135 \ProcessOptions\relax

```

`\cb@trace` A macro that formats the tracing messages.

```

136 \newcommand{\cb@trace}[1]{%
137   \if@cb@trace
138     \GenericWarning
139     {(changebar)\@spaces\@spaces}%
140     {Package changebar: #1\@gobble}%
141   \fi
142 }

```

5.3 User Level Commands And Parameters

`\driver` The user can select the specials that should be used by calling the command `\driver{<drivername>}`. Possible choices are:

- DVIToLN03
- DVIToPS
- DVIPs
- emTeX
- TeXtures
- VTeX

This command can only be used in the preamble of the document.

The argument should be case-insensitive, so it is turned into a string containing all uppercase characters. To keep some definitions local, everything is done within a group.

```

143 \if@compatibility
144   \def\driver#1{%
145     \bgroup\edef\next{\def\noexpand\tempa{#1}}%
146     \uppercase\expandafter{\next}%
147     \def\LN{DVITOLN03}%
148     \def\DVIToPS{DVITOPS}%
149     \def\DVIPS{DVIPS}%
150     \def\emTeX{EMTEX}%
151     \def\Textures{TEXTURES}%
152     \def\VTex{VTEX}%

```

The choice has to be communicated to the macro `\cb@setup@specials` that will be called from within `\document`. For this purpose the control sequence `\cb@driver@setup` is used. It receives a numeric value using `\chardef`.

```

153   \global\chardef\cb@driver@setup=0\relax
154   \ifx\tempa\LN      \global\chardef\cb@driver@setup=0\fi
155   \ifx\tempa\DVIToPS \global\chardef\cb@driver@setup=1\fi
156   \ifx\tempa\DVIPS   \global\chardef\cb@driver@setup=2\fi
157   \ifx\tempa\emTeX   \global\chardef\cb@driver@setup=3\fi
158   \ifx\tempa\Textures \global\chardef\cb@driver@setup=4\fi
159   \ifx\tempa\VTex    \global\chardef\cb@driver@setup=5\fi
160   \egroup}

```

We add `\driver` to `\@preamblecmds`, which is a list of commands to be used only in the preamble of a document.

```

161   {\def\do{\noexpand\do\noexpand}
162   \xdef\@preamblecmds{\@preamblecmds \do\driver}
163   }
164 \fi

```

`\cb@setup@specials` The macro `\cb@setup@specials` defines macros containing the driver specific `\special` macros. It will be called from within the `\begin{document}` command.

`\cb@trace@defpoint` When tracing is on, write information about the point being defined to the log file.

```
165 \def\cb@trace@defpoint#1#2{%
166   \cb@trace{%
167     defining point \the#1 at position \the#2
168     \MessageBreak
169     cb@pagecount: \the\cb@pagecount; page \thepage}}
```

`\cb@trace@connect` When tracing is on, write information about the points being connected to the log file.

```
170 \def\cb@trace@connect#1#2#3{%
171   \cb@trace{%
172     connecting points \the#1 and \the#2; barwidth: \the#3
173     \MessageBreak
174     cb@pagecount: \the\cb@pagecount; page \thepage}}
```

`\cb@defpoint` The macro `\cb@defpoint` is used to define one of the two points of a bar. It has two arguments, the number of the point and the distance from the left side of the paper. Its syntax is: `\cb@defpoint{<number>}{<length>}`.

`\cb@resetpoints` The macro `\cb@resetpoints` can be used to instruct the printer driver that it should send a corresponding instruction to the printer. This is really only used for the LN03 printer.

`\cb@connect` The macro `\cb@connect` is used to instruct the printer driver to connect two points with a bar. The syntax is `\cb@connect{<number>}{<number>}{<length>}`. The two `<number>`s indicate the two points to be connected; the `<length>` is the width of the bar.

```
175 \def\cb@setup@specials{%
```

The control sequence `\cb@driver@setup` expands to a number which indicates the driver that will be used. The original `changebar.sty` was written with only the `\special` syntax of the program `DVItoLN03` (actually one of its predecessors, `ln03dvi`). Therefore this syntax is defined first.

```
176 \ifcase\cb@driver@setup
177   \def\cb@defpoint##1##2{%
178     \special{ln03:defpoint \the##1(\the##2,)}%
179     \cb@trace@defpoint##1##2}
180   \def\cb@connect##1##2##3{%
181     \special{ln03:connect \the##1\space\space \the##2\space \the##3}%
182     \cb@trace@connect##1##2##3}
183   \def\cb@resetpoints{%
184     \special{ln03:resetpoints \cb@minpoint \space\cb@maxpoint}}
```

The first extension to the `changebar` option was for the `\special` syntax of the program `DVItoPS` by James Clark.

```
185 \or
186   \def\cb@defpoint##1##2{%
187     \special{dvitops: inline
188       \expandafter\cb@removedim\the##2\space 6.5536 mul\space
189       /CBarX\the##1\space exch def currentpoint exch pop
190       /CBarY\the##1\space exch def}}%
191     \cb@trace@defpoint##1##2}
```

```

192 \def\cb@connect##1##2##3{%
193   \special{dvitops: inline
194     gsave \cb@ps@color\space
195       \expandafter\cb@removedim\the##3\space 6.5536 mul\space
196       CBarX\the##1\space\space CBarY\the##1\space\space moveto
197       CBarX\the##2\space\space CBarY\the##2\space\space lineto
198       stroke grestore}%
199   \cb@trace@connect##1##2##3}
200 \let\cb@resetpoints\relax

```

The program DVIPs by Thomas Rokicki is also supported. The PostScript code is nearly the same as for DVIToPS, but the coordinate space has a different dimension. Also this code has been made resolution independent, whereas the code for DVIToPS might still be resolution dependent.

So far all the positions have been calculated in pt units. DVIPs uses pixels internally, so we have to convert pts into pixels which of course is done by dividing by 72.27 (pts per inch) and multiplying by Resolution giving the resolution of the POSTSCRIPT device in use as a POSTSCRIPT variable.

```

201 \or
202 \def\cb@defpoint##1##2{%
203   \special{ps:
204     \expandafter\cb@removedim\the##2\space
205     Resolution\space mul\space 72.27\space div\space
206     /CBarX\the##1\space exch def currentpoint exch pop
207     /CBarY\the##1\space exch def}%
208   \cb@trace@defpoint##1##2}
209 \def\cb@connect##1##2##3{%
210   \special{ps:
211     gsave \cb@ps@color\space
212       \expandafter\cb@removedim\the##3\space
213       Resolution\space mul\space 72.27\space div\space
214       setlinewidth
215       CBarX\the##1\space\space CBarY\the##1\space\space moveto
216       CBarX\the##2\space\space CBarY\the##2\space\space lineto
217       stroke grestore}%
218   \cb@trace@connect##1##2##3}
219 \let\cb@resetpoints\relax

```

The latest addition is for the drivers written by Eberhard Mattes. The `\special` syntax used here is supported since version 1.5 of his driver programs.

```

220 \or
221 \def\cb@defpoint##1##2{%
222   \special{em:point \the##1,\the##2}%
223   \cb@trace@defpoint##1##2}
224 \def\cb@connect##1##2##3{%
225   \special{em:line \the##1,\the##2,\the##3}%
226   \cb@trace@connect##1##2##3}
227 \let\cb@resetpoints\relax

```

The following definitions are validated with `TeXtures` version 1.7.7, but will very likely also work with later releases of `TeXtures`.

The `\cbdelete` command seemed to create degenerate lines (i.e., lines of 0 length). PostScript will not render such lines unless the `linecap` is set to 1, (semi-circular ends) in which case a filled circle is shown for such lines.

```

228 \or
229 \def\cb@defpoint##1##2{%
230   \special{rawpostscript
231     \expandafter\cb@removedim\the##2\space
232     /CBarX\the##1\space exch def currentpoint exch pop
233     /CBarY\the##1\space exch def}%
234   \if@cb@trace\cb@trace@defpoint##1##2\fi}
235 \def\cb@connect##1##2##3{%
236   \special{rawpostscript
237     gsave 1 setlinecap \cb@ps@color\space
238     \expandafter\cb@removedim\the##3\space
239     setlinewidth
240     CBarX\the##1\space\space CBarY\the##1\space\space moveto
241     CBarX\the##2\space\space CBarY\the##2\space\space lineto
242     stroke grestore}%
243   \if@cb@trace\cb@trace@connect##1##2##3\fi}
244 \let\cb@resetpoints\relax

```

The following definitions were kindly provided by Michael Vulis.

```

245 \or
246 \def\cb@defpoint##1##2{%
247   \special{pS:
248     \expandafter\cb@removedim\the##2\space
249     Resolution\space mul\space 72.27\space div\space
250     /CBarX\the##1\space exch def currentpoint exch pop
251     /CBarY\the##1\space exch def}%
252   \cb@trace@defpoint##1##2}
253 \def\cb@connect##1##2##3{%
254   \special{pS:
255     gsave \cb@ps@color\space
256     \expandafter\cb@removedim\the##3\space
257     Resolution\space mul\space 72.27\space div\space
258     setlinewidth
259     CBarX\the##1\space\space CBarY\the##1\space\space moveto
260     CBarX\the##2\space\space CBarY\the##2\space\space lineto
261     stroke grestore}%
262   \cb@trace@connect##1##2##3}
263 \let\cb@resetpoints\relax

```

When code for other drivers should be added it can be inserted here. When someone makes a mistake and somehow selects an unknown driver a warning is issued and the macros are defined to be no-ops.

```

264 \or
265 \else
266   \PackageWarning{Changebar}{changebars not supported in unknown setup}
267   \def\cb@defpoint##1##2{\cb@trace@defpoint##1##2}
268   \def\cb@connect##1##2##3{\cb@trace@connect##1##2##3}
269   \let\cb@resetpoints\relax
270 \fi

```

The last thing to do is to forget about \cb@setup@specials.

```

271 \global\let\cb@setup@specials\relax}

```

`\cbstart` The macro `\cbstart` starts a new changebar. It has an (optional) argument that will be used to determine the width of the bar. The default width is `\changebarwidth`.

```

272 \newcommand*{\cbstart}{\@ifnextchar [%]
273     {\cb@start}%
274     {\cb@start[\changebarwidth]}}
```

`\cbend` The macro `\cbend` (surprisingly) ends a changebar. The macros `\cbstart` and `\cbend` can be used when the use of a proper \LaTeX environment is not possible.

```

275 \newcommand*{\cbend}{\cb@end}
```

`\cbdelete` The macro `\cbdelete` inserts a ‘deletebar’ in the margin. It too has an optional argument to determine the width of the bar. The default width (and length) of it are stored in `\deletebarwidth`.

```

276 \newcommand*{\cbdelete}{\@ifnextchar [%]
277     {\cb@delete}%
278     {\cb@delete[\deletebarwidth]}}
```

`\cb@delete` Deletebars are implemented as a special ‘change bar’. The bar is started and immediately ended. It is as long as it is wide.

```

279 \def\cb@delete[#1]{\vbox to \z@{\vss\cb@start[#1]\vskip #1\cb@end}}
```

`\changebar` and `\endchangebar` The macros `\changebar` and `\endchangebar` have the same function as `\cbstart` and `\cbend` but they can be used as a \LaTeX environment to enforce correct nesting. They can *not* be used in the `tabular` and `tabbing` environments.

```

280 \newenvironment{changebar}%
281     {\@ifnextchar [{\cb@start}%
282     {\cb@start[\changebarwidth]}}%
283     {\cb@end}
```

`\nochangebars` To disable changebars altogether without having to remove them from the document the macro `\nochangebars` is provided. It makes no-ops of three internal macros.

```

284 \newcommand*{\nochangebars}{%
285     \def\cb@start[##1]{}%
286     \def\cb@delete[##1]{}%
287     \let\cb@end\relax}
```

`\changebarwidth` The default width of the changebars is stored in the dimension register `\changebarwidth`.

```

288 \newlength{\changebarwidth}
289 \setlength{\changebarwidth}{2pt}
```

`\deletebarwidth` The default width of the deletebars is stored in the dimension register `\deletebarwidth`.

```

290 \newlength{\deletebarwidth}
291 \setlength{\deletebarwidth}{4pt}
```

`\changebarsep` The default separation between all bars and the text is stored in the dimen register `\changebarsep`.

```

292 \newlength{\changebarsep}
293 \setlength{\changebarsep}{30pt}
```

`changebargrey` When the document is printed using one of the PostScript drivers the bars do not need to be black; with PostScript it is possible to have grey, and colored, bars. The percentage of greyness of the bar is stored in the count register `\changebargrey`. It can have values between 0 (meaning white) and 100 (meaning black).

```
294 \newcounter{changebargrey}
295 \setcounter{changebargrey}{65}
```

When the option `color` was selected we need to load the `color` package. When we're run by pdf^LA_TE_X we need to pass that information on to the `color` package.

```
296 \@ifpackagewith{changebar}{color}{%
297   \ifx\pdfoutput\undefined
298     \RequirePackage[dvipsnames]{color}%
299   \else
300     \RequirePackage[pdftex,dvipsnames]{color}%
301   \fi
```

Then we need to define the command `\cbcolor` which is a slightly modified copy of the command `\color` from the `color` package.

`\cbcolor` `\cbcolor{declared-colour}` switches the colour of the changebars to *declared-colour*, which must previously have been defined using `\definecolor`. This colour will stay in effect until the end of the current T_EX group.

`\cbcolor[model]{colour-specification}` is similar to the above, but uses a colour not declared by `\definecolor`. The allowed *model*'s vary depending on the driver. The syntax of the *colour-specification* argument depends on the model.

```
302 \DeclareRobustCommand\cbcolor{%
303   \@ifnextchar[\@undeclaredcbcolor\@declaredcbcolor}
```

`\@undeclaredcbcolor` Call the driver-dependent command `\color@<model>` to define `\cb@current@color`.

```
304 \def\@undeclaredcbcolor[#1]#2{%
305   \ifundefined{color@#1}%
306     {\color@error{model ' #1' }}%
307     {\csname color@#1\endcsname\cb@current@color{#2}}%
308   \ignorespaces}
```

`\@declaredcbcolor`

```
309 \def\@declaredcbcolor#1{%
310   \@ifundefined{\string\color @#1}%
311     {\color@error{' #1' }}%
312     {\expandafter\let\expandafter\cb@current@color
313       \csname\string\color @#1\endcsname}%
314   \ignorespaces}%
315 }
```

When the `color` option wasn't specified the usage of the `\cbcolor` command results in a warning message.

```
316 \def\cbcolor{\@ifnextchar[%]
317   \@@cbcolor\@cbcolor}%
318 \def\@@cbcolor[#1]#2{\cb@colwarn\def\@@cbcolor{##1}##2{}}%
319 \def\@cbcolor#1{\cb@colwarn\def\@cbcolor##1{}}%
320 \def\cb@colwarn{\PackageWarning{Changebar}%
321   {You didn't specify the option 'color';\MessageBreak
322   your command \string\cbcolor\space will be ignored}}%
323 }
```

5.4 Macros for beginning and ending bars

`\cb@start` This macro starts a change bar. It assigns a new value to the current point and advances the counter for the next point to be assigned. It pushes this info onto `\cb@currentstack` and then sets the point by calling `\cb@setBeginPoints` with the point number. Finally, it writes the `.aux` file.

```
324 \def\cb@start[#1]{%
325   \cb@topleft=\cb@nextpoint

   Store the width of the current bar in \cb@curbarwd.

326   \cb@curbarwd#1\relax
327   \cb@push\cb@currentstack
```

Now find out on which page the start of this bar finally ends up; due to the asynchronous nature of the output routine it might be a different page. The macro `\cb@checkpage` finds the page number on the history stack.

```
328   \cb@checkpage\@ne

   Temporarily assign the page number to \cb@pagecount as that register is used by
   \cb@setBeginPoints. Note that it's value is offset by one from the page counter.

329   \cb@cmta\cb@pagecount
330   \cb@pagecount\cb@page\advance\cb@pagecount\m@ne
331   \ifvmode
332     \cb@setBeginPoints
333   \else
334     \vbox to \z@{%
```

When we are in horizontal mode we jump up a line to set the starting point of the changebar.

```
335     \vskip -\ht\strutbox
336     \cb@setBeginPoints
337     \vskip \ht\strutbox}%
338   \fi
```

Restore `\cb@pagecount`.

```
339   \cb@pagecount\cb@cmta
340   \cb@advancePoint}
```

`\cb@advancePoint` The macro `\cb@advancePoint` advances the count register `\cb@nextpoint`. When the maximum number is reached, the numbering is reset.

```
341 \def\cb@advancePoint{%
342   \global\advance\cb@nextpoint by 4\relax
343   \ifnum\cb@nextpoint>\cb@maxpoint
344     \global\cb@nextpoint=\cb@minpoint\relax
345   \fi}
```

`\cb@end` This macro ends a changebar. It pops the current point and nesting level off `\cb@currentstack` and sets the end point by calling `\cb@setEndPoints` with the parameter corresponding to the *beginning* point number. It writes the `.aux` file and joins the points.

```
346 \def\cb@end{%
347   \cb@trace@stack{end of bar on page \the\c@page}%
348   \cb@pop\cb@currentstack
349   \ifnum\cb@topleft=\cb@nil
350     \PackageWarning{Changebar}%
```

```

351     {Badly nested changebars; Expect erroneous results}%
352   \else

   Call \cb@checkpage to find the page this point finally ends up on.
353   \cb@checkpage\tw@

   Again, we need to temporarily overwrite \cb@pagecount.
354   \cb@cmta\cb@pagecount
355   \cb@pagecount\cb@page\advance\cb@pagecount\m@ne
356   \cb@setEndPoints
357   \cb@pagecount\cb@cmta
358   \fi
359   \ignorespaces}

```

`\cb@checkpage` The macro `\cb@checkpage` checks the history stack in order to find out on which page a set of points finally ends up.

We expect the identification of the points in `\cb@topleft` and `\cb@page`. The resulting page will be stored in `\cb@page`.

```

360 \def\cb@checkpage#1{%

```

First store the identifiers in temporary registers.

```

361   \cb@cmta\cb@topleft\relax
362   \cb@cntb\cb@page\relax
363   \cb@dima\cb@curbarwd\relax

```

Then pop the history stack.

```

364   \cb@pop\cb@historystack

```

If it was empty there is nothing to check and we're done.

```

365   \ifnum\cb@topleft=\cb@nil
366   \else

```

Now keep popping the stack until `\cb@topleft` is no longer less than the value of `\cb@cmta`. The values popped from the stack are pushed on a temporary stack to be pushed back later. This could perhaps be implemented more efficiently if the stacks had a different design.

```

367     \@whilenum\cb@topleft<\cb@cmta\do{%
368       \cb@push\cb@tempstack
369       \cb@pop\cb@historystack

```

When the user adds changebars to his document we might run out of the history stack before we find a match. This would send TeX into an endless loop if it wasn't detected and handled.

```

370     \ifnum\cb@topleft=\cb@nil
371     \cb@trace{Ran out of history stack, new changebar?}%

```

In this case we give `\cb@topleft` an 'impossible value' to remember this special situation.

```

372     \cb@topleft\cb@maxpoint\advance\cb@topleft\@ne
373     \fi
374   }%

```

If we are looking for the start point of a bar we may have found it now, for the end point we need to pop one more value. If `\cb@topleft` has become larger than `\cb@maxpoint` we haven't found what we're looking for and we've run out of the stack.

```

375 \ifnum\cb@topleft>\cb@maxpoint\else
376 \ifodd#1\else
377 \cb@push\cb@tempstack
378 \cb@pop\cb@historystack
379 \fi

```

Now that we've found it overwrite `\cb@cntb` with the `\cb@page` from the stack.

```

380 \cb@cntb\cb@page
381 \fi

```

Now we restore the history stack to it's original state.

```

382 \@whilenum\cb@topleft>\cb@nil\do{%
383 \cb@push\cb@historystack
384 \cb@pop\cb@tempstack}%
385 \fi

```

Finally return the correct values.

```

386 \cb@topleft\cb@cmta\relax
387 \cb@page\cb@cntb\relax
388 \cb@curbarwd\cb@dimas\relax
389 }

```

`\cb@setBeginPoints` The macro `\cb@setBeginPoints` assigns a position to the top left and top right points. It determines whether the point is on an even or an odd page and uses the right dimension to position the point. Keep in mind that the value of `\cb@pagecount` is one less than the value of `\cb@page` unless the latter has been reset by the user.

The top left point is used to write an entry on the `.aux` file to create the history stack on the next run.

```

390 \def\cb@setBeginPoints{%
391 \cb@topright=\cb@topleft\advance\cb@topright by\@ne
392 \ifodd\cb@pagecount
393 \cb@defpoint\cb@topleft\cb@even@left
394 \cb@defpoint\cb@topright\cb@even@right
395 \else
396 \cb@defpoint\cb@topleft\cb@odd@left
397 \cb@defpoint\cb@topright\cb@odd@right
398 \fi
399 \cb@writeAux\cb@topleft
400 }

```

`\cb@setEndPoints` The macro `\cb@setEndPoints` assigns positions to the bottom points for a change bar. It then instructs the driver to connect two points with a bar. The macro assumes that the width of the bar is stored in `\cb@curbarwd`.

The bottom right point is used to write to the `.aux` file to signal the end of the current bar on the history stack.

```

401 \def\cb@setEndPoints{%
402 \cb@topright=\cb@topleft\advance\cb@topright by\@ne
403 \cb@botleft=\cb@topleft\advance\cb@botleft by\tw@
404 \cb@botright=\cb@topleft\advance\cb@botright by\thr@@
405 \ifodd\cb@pagecount
406 \cb@defpoint\cb@botleft\cb@even@left
407 \cb@defpoint\cb@botright\cb@even@right
408 \else

```

```

409 \if@twoside
410   \cb@defpoint\cb@botleft\cb@odd@left
411   \cb@defpoint\cb@botright\cb@odd@right
412 \else
413   \cb@defpoint\cb@botleft\cb@even@left
414   \cb@defpoint\cb@botright\cb@even@right
415 \fi
416 \fi
417 \cb@writeAux\cb@botright
418 \edef\cb@leftbar{%
419   \noexpand\cb@connect{\cb@topleft}{\cb@botleft}{\cb@curbarwd}}%
420 \edef\cb@rightbar{%
421   \noexpand\cb@connect{\cb@topright}{\cb@botright}{\cb@curbarwd}}%
422 \if@twocolumn
423   \if@firstcolumn\cb@leftbar\else\cb@rightbar\fi
424 \else
425   \ifouterbars
426     \ifodd\cb@pagecount
427       \cb@leftbar
428     \else
429       \if@twoside\cb@rightbar\else\cb@leftbar\fi
430     \fi
431   \else
432     \ifodd\cb@pagecount
433       \cb@rightbar
434     \else
435       \if@twoside\cb@leftbar\else\cb@rightbar\fi
436     \fi
437   \fi
438 \fi
439 }%

```

`\cb@writeAux` The macro `\cb@writeAux` writes information about a changebar point to the auxiliary file. The number of the point, the pagenumber and the width of the bar are written out as arguments to `\cb@barpoint`. This latter macro will be expanded when the auxiliary file is read in. The macro assumes that the width of bar is stored in `\cb@curbarwd`.

The code is only executed when auxiliary files are enabled, as there's no sense in trying to write to an unopened file.

```

440 \def\cb@writeAux#1{%
441   \if@filesw
442     \begingroup
443     \edef\point{\the#1}%
444     \edef\level{\the\cb@curbarwd}%
445     \let\the=\z@
446     \edef\cb@temp{\write\@auxout
447       {\string\cb@barpoint{\point}{\the\cb@pagecount}{\level}}}%
448     \cb@temp
449   \endgroup
450 \fi}

```

5.5 Macros for Making It Work Across Page Breaks

`\@makecol` These internal L^AT_EX macros are modified in order to end the changebars spanning the current page break (if any) and restart them on the next page. The modifications are needed to reset the special points for this page and add begin bars to top of box255. The bars carried over from the previous page, and hence to be restarted on this page, have been saved on the stack `\cb@beginstack`. This stack is used to define new starting points for the change bars, which are added to the top of box `\@cc1v`. Then the stack `\cb@endstack` is built and processed by `\cb@processActive`. Finally the original `\@makecol` (saved as `\cb@makecol`) is executed.

```

451 \let\ltx@makecol\@makecol
452 \def\cb@makecol{%
453   \cb@trace@stack{before makecol, page \the\c@page}%
454   \let\cb@writeAux\@gobble
455   \setbox\@cc1v \vbox{%
456     \cb@resetpoints
457     \cb@startSpanBars
458     \unvbox\@cc1v
459     \boxmaxdepth\maxdepth}%
460   \global\advance\cb@pagecount by\@ne
461   \cb@buildstack\cb@processActive
462   \ltx@makecol
463   \cb@trace@stack{after makecol, page \the\c@page}%
464 }
465 \let\@makecol\cb@makecol

```

When L^AT_EX makes a page with only floats it doesn't use `\@makecol`; instead it calls `\@vtryfc`, so we have to modify this macro as well.

```

466 \let\ltx@vtryfc\@vtryfc
467 \def\cb@vtryfc{%
468   \let\cb@writeAux\@gobble
469   \setbox\@outputbox \vbox{%
470     \cb@resetpoints
471     \cb@startSpanBars
472     \unvbox\@cc1v
473     \boxmaxdepth\maxdepth}%
474   \global\advance\cb@pagecount by \@ne
475   \cb@buildstack\cb@processActive
476   \ltx@vtryfc}
477 \let\@vtryfc\cb@vtryfc

```

`\cb@processActive` This macro processes each element on span stack. Each element represents a bar that crosses the page break. There could be more than one if bars are nested. It works as follows:

```

pop top element of span stack
if point null (i.e., stack empty) then done
else
  do an end bar on box255
  save start for new bar at top of next page in \cb@startSaves
  push active point back onto history stack (need to reprocess
  on next page).

```

```

478 \def\cb@processActive{%
479   \cb@pop\cb@endstack
480   \ifnum\cb@topleft=\cb@nil
481   \else
482     \setbox\@cclv\vbox{%
483       \unvbox\@cclv
484       \boxmaxdepth\maxdepth
485       \advance\cb@pagecount by -1\relax
486       \cb@setEndPoints}%
487     \cb@push\cb@historystack
488     \cb@push\cb@beginstack
489     \expandafter\cb@processActive
490   \fi}

```

`\cb@startSpanBars` This macro defines new points for each bar that was pushed on the `\cb@beginstack`. Afterwards `\cb@beginstack` is empty.

```

491 \def\cb@startSpanBars{%
492   \cb@pop\cb@beginstack
493   \ifnum\cb@topleft=\cb@nil
494   \else
495     \cb@setBeginPoints
496     \cb@trace@stack{after StartSpanBars, page \the\c@page}%
497     \expandafter\cb@startSpanBars
498   \fi
499 }

```

`\cb@buildstack` The macro `\cb@buildstack` initializes the stack with open bars and starts populating it.

```

500 \def\cb@buildstack{%
501   \cb@initstack\cb@endstack
502   \cb@pushNextActive}

```

`\cb@pushNextActive` This macro pops the top element off the history stack (`\cb@historystack`). If the top left point is on a future page, it is pushed back onto the history stack and processing stops. If the point on the current or a previous page and it has an odd number, the point is pushed on the stack with end points `\cb@endstack`; if the point has an even number, it is popped off the stack with end points since the bar to which it belongs has terminated on the current page.

```

503 \def\cb@pushNextActive{%
504   \cb@pop\cb@historystack
505   \ifnum\cb@topleft=\cb@nil
506   \else
507     \ifnum\cb@page>\cb@pagecount
508       \cb@push\cb@historystack
509     \else
510       \ifodd\cb@topleft
511         \cb@push\cb@endstack
512       \else
513         \cb@pop\cb@endstack
514       \fi
515     \expandafter\expandafter\expandafter\cb@pushNextActive
516   \fi
517 \fi}

```

5.6 Macros For Managing The Stacks of Bar points

The macros make use of four stacks corresponding to `\special` defpoints. Each stack takes the form `<element> ... <element>`

Each element is of the form `xxxnyyyppzzz` where `xxx` is the number of the special point, `yyy` is the page on which this point is set, and `zzz` is the dimension used when connecting this point.

The stack `\cb@historystack` is built from the log information and initially lists all the points. As pages are processed, points are popped off the stack and discarded.

The stack `\cb@endstack` and `\cb@beginstack` are two temporary stacks used by the output routine and contain the stack with definitions for of all bars crossing the current pagebreak (there may be more than one with nested bars). They are built by popping elements off the history stack.

The stack `\cb@currentstack` contains all the current bars. A `\cb@start` pushes an element onto this stack. A `\cb@end` pops the top element off the stack and uses the info to terminate the bar.

For performance and memory reasons, the history stack, which can be very long, is special cased and a file is used to store this stack rather than an internal macro. The “external” interface to this stack is identical to what is described above. However, when the history stack is popped, a line from the file is first read and appended to the macro `\cb@historystack`.

```

\cb@initstack A macro to (globally) initialize a stack.
518 \def\cb@initstack#1{\xdef#1{}}

\cb@historystack We need to initialise a stack to store the entries read from the external history
\cb@write file.
\cb@read 519 \cb@initstack\cb@historystack

We also need to allocate a read and a write stream for the history file.
520 \newwrite\cb@write
521 \newread\cb@read

And we open the history file for writing (which is done when the .aux file is read
in).
522 \immediate\openout\cb@write=\jobname.cb\relax

\cb@endstack Allocate two stacks for the bars that span the current page break.
\cb@beginstack 523 \cb@initstack\cb@endstack
524 \cb@initstack\cb@beginstack

\cb@tempstack Allocate a stack for temporary storage
525 \cb@initstack\cb@tempstack

\cb@currentstack And we allocate an extra stack that is needed to implement nesting without having
to rely on TeX's grouping mechanism.
526 \cb@initstack\cb@currentstack

\cb@pop This macro pops the top element off the named stack and puts the point value into
\cb@topleft, the page value into \cb@page and the bar width into \cb@curbarwd.
```

If the stack is empty, it returns a void value (`\cb@nil`) in `\cb@topleft` and sets `\cb@page=0`.

```

527 \def\cb@pop#1{%
528   \ifx #1\cb@historystack
529     \ifeof\cb@read
530     \else
531       {\endlinechar=-1\read\cb@read to\@temp
532        \xdef\cb@historystack{\cb@historystack\@temp}%
533       }%
534     \fi
535   \fi
536   \ifx#1\@empty
537     \global\cb@topleft\cb@nil
538     \global\cb@page\z\relax
539   \else
540     \expandafter\cb@carcdr#1e#1%
541   \fi
542   \cb@trace@pop{#1}}

```

`\cb@carcdr` This macro is used to ‘decode’ a stack entry.

```

543 \def\cb@carcdr#1n#2p#3l#4e#5{%
544   \global\cb@topleft#1\relax
545   \global\cb@page#2\relax
546   \global\cb@curbarwd#3\relax
547   \xdef#5{#4}}

```

`\cb@push` The macro `\cb@push` Pushes `\cb@topleft`, `\cb@page` and `\cb@curbarwd` onto the top of the named stack.

```

548 \def\cb@push#1{%
549   \xdef#1{\the\cb@topleft n\the\cb@page p\the\cb@curbarwd l#1}%
550   \cb@trace@push{#1}}
551

```

`\cb@barpoint` The macro `\cb@barpoint` populates the history file. It writes one line to `.cb` file which is equivalent to one *element* described above.

```

552 \def\cb@barpoint#1#2#3{\immediate\write\cb@write{#1n#2p#3l}}

```

5.7 Macros For Checking That The .aux File Is Stable

`\AtBeginDocument` While reading the `.aux` file, L^AT_EX has created the history stack in a separate file. We need to close that file and open it for reading. Also the ‘initialisation’ of the `\special` commands has to take place. While we are modifying the macro we also include the computation of the possible positions of the changebars

For these actions we need to add to the L^AT_EX begin-document hook.

```

553 \AtBeginDocument{%
554   \cb@setup@specials

```

When the document doesn’t start with page 1 but with another page set with `\setcounter{page}{...}` the code needs to know about it. Therefore we need to initialize `\cb@pagecount`.

```

555   \global\cb@pagecount\c@page
556   \global\advance\cb@pagecount\m@ne

```

Compute the left and right positions of the changebars.

```
557 \cb@positions
558 \cb@trace{%
559   Odd left : \the\cb@odd@left\space
560   Odd right : \the\cb@odd@right\MessageBreak
561   Even left: \the\cb@even@left\space
562   Even right: \the\cb@even@right
563 }%
564 \immediate\closeout\cb@write
565 \immediate\openin\cb@read=\jobname.cb}
```

`\AtEndDocument` We need to issue a `\clearpage` to flush rest of document. (Note that I believe there is contention in this area: are there in fact situations in which the end-document hooks need to be called *before* the final `\clearpage`? — the documentation of L^AT_EX itself implies that there are.) Then closes the `.cb` file and reopens it for checking. Initialize history stack (to be read from file). Let `\cb@barpoint=\cb@checkHistory` for checking.

```
566 \AtEndDocument{%
567   \clearpage
568   \cb@initstack\cb@historystack
569   \immediate\closein\cb@read
570   \immediate\openin\cb@read=\jobname.cb%
571   \let\cb@barpoint\cb@checkHistory}
```

`\cb@checkHistory` Pops the top of the history stack (`\jobname.cb`) and checks to see if the point and page numbers are the same as the arguments #1 and #2 respectively. Prints a warning message if different.

```
572 \def\cb@checkHistory#1#2#3{%
573   \cb@pop\cb@historystack
574   \ifnum #1=\cb@topleft\relax
575     \ifnum #2=\cb@page\relax
```

Both page and point numbers are equal; do nothing,

```
576   \else
but generate a warning when page numbers don't match, or
```

```
577     \cb@error
578     \fi
579   \else
```

when point numbers don't match.

```
580     \cb@error
581     \fi}
```

`\cb@error` When a mismatch between the changebar information in the auxiliary file and the history stack is detected a warning is issued; further checking is disabled.

```
582 \def\cb@error{%
583   \PackageWarning{Changebar}%
584   {Changebar info has changed.\MessageBreak
585   Rerun to get the bars right}
586   \gdef\cb@checkHistory##1##2##3{%
587   \let\cb@barpoint\cb@checkHistory}
```

5.8 Macros For Making It Work With Nested Floats/Footnotes

`\end@float` This is a replacement for the \LaTeX -macro of the same name. All it does is check to see if changebars are active and, if so, it puts changebars around the box containing the float. Then it calls the original \LaTeX `\end@float`.

```
588 \let\ltx@end@float\end@float
589 \def\cb@end@float{%
590   \cb@trace@stack{end float on page \the\c@page}%
591   \cb@pop\cb@currentstack
592   \ifnum\cb@topleft=\cb@nil
593   \else
594     \cb@push\cb@currentstack
595     \global\cb@curbarwd=\cb@curbarwd
596     \@endfloatbox
597     \global\setbox\@currbox
598       \color@vbox
599       \normalcolor
600       \vbox\bgroup\cb@start[\cb@curbarwd]\unvbox\@currbox\cb@end
601   \fi
602   \ltx@end@float}
603 \let\end@float\cb@end@float
```

This only works if this new version of `\end@float` is really used. With \LaTeX 2.09 the documentstyles used to contain:

```
\let\endfigure\end@float
```

In that case this binding has to be repeated after the redefinition of `\end@float`. However, the \LaTeX 2 ϵ class files use `\newenvironment` to define the figure and table environments. In that case there is no need to rebind `\endfigure`.

`\float@end` When the `float` package is being used we need to take care of its changes to the float mechanism. It defines its own macros (`\float@end` and `\float@dblend`) which need to be modified for changebars to work.

First we'll save the original as `\flt@float@end`.

```
604 \let\flt@float@end\float@end
    Then we redefine it to insert the changebarcode.
605 \def\float@end{%
606   \cb@trace@stack{end float on page \the\c@page}%
607   \cb@pop\cb@currentstack
608   \ifnum\cb@topleft=\cb@nil
609   \else
610     \cb@push\cb@currentstack
611     \global\cb@curbarwd\cb@curbarwd
612     \@endfloatbox
613     \global\setbox\@currbox
614       \color@vbox
615       \normalcolor
616       \vbox\bgroup\cb@start[\cb@curbarwd]\unvbox\@currbox\cb@end
617   \fi
618   \let\end@float\ltx@end@float
619   \flt@float@end
620 }
```

`\end@dblfloat` This is a replacement for the \LaTeX -macro of the same name. All it does is check to see if changebars are active and, if so, it puts changebars around the box containing the float. In this case the \LaTeX macro had to be rewritten.

```

621 \let\ltx@end@dblfloat\end@dblfloat
622 \def\cb@end@dblfloat{%
623   \if@twocolumn
624     \cb@trace@stack{end dblfloat on page \the\c@page}%
625     \cb@pop\cb@currentstack
626     \ifnum\cb@topleft=\cb@nil
627       \else
628         \cb@push\cb@currentstack
629         \global\cb@curbarwd=\cb@curbarwd
630         \@endfloatbox
631         \global\setbox\@currbox
632           \color@vbox
633           \normalcolor
634           \vbox\bgroup\cb@start[\cb@curbarwd]\unvbox\@currbox\cb@end
635         \fi
636         \@endfloatbox
637         \ifnum\@floatpenalty <\z@
638           \@largefloatcheck
639           \@cons\@dbldeferlist\@currbox
640         \fi
641         \ifnum \@floatpenalty =-\@Mii \@Esphack\fi
642       \else
643         \end@float
644       \fi}
645 \let\end@dblfloat\cb@end@dblfloat

```

`\float@dblend` Something similar needs to be done for the case where the float package is being used...

```

646 \let\flt@float@dblend\float@dblend
647 \def\float@dblend{%
648   \cb@trace@stack{end dbl float on page \the\c@page}%
649   \cb@pop\cb@currentstack
650   \ifnum\cb@topleft=\cb@nil
651     \else
652       \cb@push\cb@currentstack
653       \global\cb@curbarwd=\cb@curbarwd
654       \@endfloatbox
655       \global\setbox\@currbox
656         \color@vbox
657         \normalcolor
658         \vbox\bgroup\cb@start[\cb@curbarwd]\unvbox\@currbox\cb@end
659     \fi
660   \let\end@dblfloat\ltx@end@dblfloat
661   \flt@float@dblend
662 }

```

`\@footnotetext` This is a replacement for the \LaTeX macro of the same name. It simply checks to see if changebars are active, and if so, wraps the macro argument (i.e., the footnote) in changebars.

```

663 \let\ltx@footnotetext\@footnotetext

```

```

664 \long\def\cb@footnotetext#1{%
665   \cb@trace@stack{end footnote on page \the\c@page}%
666   \cb@pop\cb@currentstack
667   \ifnum\cb@topleft=\cb@nil
668     \ltx@footnotetext{#1}%
669   \else
670     \cb@push\cb@currentstack
671     \edef\cb@temp{\the\cb@curbarwd}%
672     \ltx@footnotetext{\cb@start[\cb@temp]#1\cb@end}%
673   \fi}
674 \let\@footnotetext\cb@footnotetext

```

`\@mpfootnotetext` Replacement for the L^AT_EX macro of the same name. Same thing as `\@footnotetext`.

```

675 \let\ltx@mpfootnotetext\@mpfootnotetext
676 \long\def\cb@mpfootnotetext#1{%
677   \cb@pop\cb@currentstack
678   \ifnum\cb@topleft=\cb@nil
679     \ltx@mpfootnotetext{#1}%
680   \else
681     \cb@push\cb@currentstack
682     \edef\cb@temp{\the\cb@curbarwd}%
683     \ltx@mpfootnotetext{\cb@start[\cb@temp]#1\cb@end}%
684   \fi}
685 \let\@mpfootnotetext\cb@mpfootnotetext
686 \</package>

```

Index

Numbers in *italics> indicate the page where the macro is described, the underlined numbers indicate the number of the line of code where the macro is defined, all other numbers indicate where a macro is used.*

| | | |
|---|--|--|
| Symbols | <code>\@endfloatbox</code> . 596, | <code>\@undeclaredcbcolor</code> |
| <code>\@@</code> 110 | 612, 630, 636, 654 | 303, <u>304</u> |
| <code>\@cbcolor</code> . . . 317, 318 | <code>\@floatpenalty</code> 637, 641 | <code>\@vtryfc</code> <u>451</u> |
| <code>\@Esphack</code> 641 | <code>\@footnotetext</code> <u>663</u> | <code>\@whilenum</code> . . . 367, 382 |
| <code>\@Mii</code> 641 | <code>\@gobble</code> 132– | |
| <code>\@auxout</code> 446 | 134, 140, 454, 468 | A |
| <code>\cb@trace</code> <u>7</u> | <code>\@ifnextchar</code> . . 272, | <code>\advance</code> . 27, 29–33, |
| <code>\@cb@tracefalse</code> . . . 99 | 276, 281, 303, 316 | 35, 37–41, 66– |
| <code>\@cb@tracetrue</code> 98 | <code>\@ifpackagewith</code> . . . 296 | 68, 70–72, 83– |
| <code>\@cbcolor</code> 317, 319 | <code>\@ifundefined</code> . 305, 310 | 86, 88–91, 330, |
| <code>\@cclv</code> 455, | <code>\@largefloatcheck</code> . 638 | 342, 355, 372, |
| 458, 472, 482, 483 | <code>\@makecol</code> <u>451</u> | 391, 402–404, |
| <code>\@cons</code> 639 | <code>\@mpfootnotetext</code> . . <u>675</u> | 460, 474, 485, 556 |
| <code>\@currbox</code> . 597, 600, | <code>\@ne</code> 328, 372, | <code>\AtBeginDocument</code> . . <u>553</u> |
| 613, 616, 631, | 391, 402, 460, 474 | <code>\AtEndDocument</code> <u>566</u> |
| 634, 639, 655, 658 | <code>\@outputbox</code> 469 | B |
| <code>\@dbldeferlist</code> 639 | <code>\@preamblecmds</code> 162 | <code>\begingroup</code> 442 |
| <code>\@declaredcbcolor</code> . | <code>\@spaces</code> 128, 139 | <code>\bgroup</code> 145, |
| 303, <u>309</u> | <code>\@temp</code> 531, 532 | 600, 616, 634, 658 |
| <code>\@empty</code> 536 | | |

| | | |
|---|---|--|
| <code>\boxmaxdepth</code> | <code>\cb@delete</code> | 64–68, 74, 76, 95, 396, 410, 559 |
| 459, 473, 484 | . 277, 278, <u>279</u> , 286 | |
| C | | |
| <code>\c@lor@error</code> | <code>\cb@dima</code> | <code>\cb@odd@right</code> |
| <code>\c@lor@to@ps</code> | <code>\cb@driver@setup</code> | <u>20</u> , 28–31, 44, 76, 81–86, 93, 95, 397, 411, 560 |
| <code>\c@page</code> 347, 453, 463, 496, 555, 590, 606, 624, 648, 665 | <code>\cb@end</code> | <code>\cb@page</code> |
| <code>\catcode</code> | . 279, 283, 287, 346, 600, 616, 634, 658, 672, 683 | . <u>15</u> , 330, 355, 362, 380, 387, 507, 538, 545, 549, 575 |
| <code>\cb@@show@stack</code> 101, <u>124</u> | <code>\cb@end@dblfloat</code> | <code>\cb@pagecount</code> |
| <code>\cb@advancePoint</code> | 622, 645 | . 169, 174, 329, 330, 339, 354, 355, 357, 392, 405, 426, 432, 447, 460, 474, 485, 507, 555, 556 |
| <code>\cb@barpoint</code> | <code>\cb@end@float</code> | <code>\cb@pop</code> |
| . 447, <u>552</u> , 571, 587 | . 589, 603 | . 348, 364, 369, 378, 384, 479, 492, 504, 513, <u>527</u> , 573, 591, 607, 625, 649, 666, 677 |
| <code>\cb@beginstack</code> | <code>\cb@endstack</code> 128, 479, <u>500</u> , 511, 513, <u>523</u> | <code>\cb@positions</code> |
| . 129, 488, 492, <u>523</u> | <code>\cb@error</code> . 577, 580, <u>582</u> | <u>20</u> , 63, 80, 557 |
| <code>\cb@botleft</code> | <code>\cb@even@left</code> | <code>\cb@processActive</code> |
| . 406, 410, 413, 419 | <u>20</u> , 26, 35– 38, 43, 65, 70– 72, 74, 77, 96, 393, 406, 413, 561 | 461, 475, <u>478</u> |
| <code>\cb@botright</code> | <code>\cb@even@right</code> | <code>\cb@ps@color</code> 108, 110, 194, 211, 237, 255 |
| <u>7</u> , 404, 407, 411, 414, 417, 421 | <u>20</u> , 36, 39– 41, 44, 77, 82, 88–91, 93, 96, 394, 407, 414, 562 | <code>\cb@push</code> |
| <code>\cb@buildstack</code> | <code>\cb@footnotetext</code> | . 327, 368, 377, 383, 487, 488, 508, 511, <u>548</u> , 594, 610, 628, 652, 670, 681 |
| 461, 475, <u>500</u> | 664, 674 | <code>\cb@pushNextActive</code> |
| <code>\cb@carcdr</code> | <code>\cb@historystack</code> | 502, <u>503</u> |
| <code>\cb@checkHistory</code> | . 130, 364, 369, 378, 383, 487, 504, 508, <u>519</u> , 528, 532, 568, 573 | <code>\cb@read</code> |
| . 571, <u>572</u> , 586, 587 | <code>\cb@initstack</code> | . 519, 529, 531, 565, 569, 570 |
| <code>\cb@checkpage</code> | 501, <u>518</u> , 519, 523–526, 568 | <code>\cb@removedim</code> <u>47</u> , 188, 195, 204, 212, 231, 238, 248, 256 |
| 328, 353, <u>360</u> | <code>\cb@leftbar</code> | <code>\cb@resetpoints</code> |
| <code>\cb@canta</code> | . 418, 423, 427, 429, 435 | <u>175</u> , 456, 470 |
| . 329, 339, 354, 357, 361, 367, 386 | <code>\cb@makecol</code> | <code>\cb@rightbar</code> |
| <code>\cb@cntb</code> <u>11</u> , 362, 380, 387 | . 452, 465 | . 420, 423, 429, 433, 435 |
| <code>\cb@colwarn</code> | <code>\cb@maxpoint</code> | <code>\cb@setBeginPoints</code> |
| . 318–320 | <u>1</u> , 184, 343, 372, 375 | . 332, 336, <u>390</u> , 495 |
| <code>\cb@connect</code> <u>175</u> , 419, 421 | <code>\cb@minpoint</code> | <code>\cb@setEndPoints</code> |
| <code>\cb@curbarwd</code> | <u>3</u> , 6, 184, 344 | 356, <u>401</u> , 486 |
| <u>14</u> , 326, 363, 388, 419, 421, 444, 546, 549, 595, 600, 611, 616, 629, 634, 653, 658, 671, 682 | <code>\cb@mpfootnotetext</code> | <code>\cb@setup@specials</code> |
| <code>\cb@current@color</code> | 676, 685 | <u>165</u> , 554 |
| 110, 307, 312 | <code>\cb@nextpoint</code> | <code>\cb@start</code> |
| <code>\cb@currentstack</code> | <u>5</u> , 325, 342–344 | . 273, 274, 279, 281, 282, 285, |
| . 127, 327, 348, <u>526</u> , 591, 594, 607, 610, 625, 628, 649, 652, 666, 670, 677, 681 | <code>\cb@nil</code> | |
| <code>\cb@defpoint</code> | <u>4</u> , 349, 365, 370, 382, 480, 493, 505, 537, 592, 608, 626, 650, 667, 678 | |
| . 393, 394, 396, 397, 406, 407, 410, 411, 413, 414 | <code>\cb@odd@left</code> <u>20</u> , 25– 28, 32, 33, 43, | |

| | | | |
|---------------------------------|---|------------------------------------|--|
| | 324, 600, 616, 634, 658, 672, 683 | | |
| <code>\cb@startSpanBars</code> | 457, 471, <u>491</u> | <code>\changebarwidth</code> | 3, 31, 33, 38, 41, 68, 72, 86, 91, 274, 282, <u>288</u> |
| <code>\cb@temp</code> | .. 446, 448, 671, 672, 682, 683 | <code>\chardef</code> | 48–59, 153–159 |
| <code>\cb@tempstack</code> | 368, 377, 384, <u>525</u> | <code>\clearpage</code> | 567 |
| <code>\cb@topleft</code> | <u>7</u> , 103, 105, 325, 349, 361, 365, 367, 370, 372, 375, 382, 386, 391, 393, 396, 399, 402– 404, 419, 480, 493, 505, 510, 537, 544, 549, 574, 592, 608, 626, 650, 667, 678 | <code>\closein</code> | 569 |
| <code>\cb@topright</code> | . <u>7</u> , 391, 394, 397, 402, 421 | <code>\closeout</code> | 564 |
| <code>\cb@trace</code> | 102, 104, 125, <u>136</u> , 166, 171, 371, 558 | <code>\color</code> | 310, 313 |
| <code>\cb@trace@connect</code> | <u>170</u> , 182, 199, 218, 226, 243, 262, 268 | <code>\color@vbox</code> | 598, 614, 632, 656 |
| <code>\cb@trace@defpoint</code> | <u>165</u> , 179, 191, 208, 223, 234, 252, 267 | <code>\csname</code> | 307, 313 |
| <code>\cb@trace@pop</code> | 104, <u>133</u> , 542 | <code>\CurrentOption</code> | 113 |
| <code>\cb@trace@push</code> | 102, <u>133</u> , 550 | D | |
| <code>\cb@trace@stack</code> | 101, 132, 347, 453, 463, 496, 590, 606, 624, 648, 665 | <code>\DeclareOption</code> | 48–62, 79, 98– 100, 107, 109, 111 |
| <code>\cb@vtryfc</code> | ... 467, 477 | <code>\DeclareRobustCommand</code> | 302 |
| <code>\cb@write</code> | . <u>519</u> , 552, 564 | <code>\def</code> | 2–4, 24, 63, 80, 102, 104, 108, 110, 124, 144, 145, 147–152, 161, 165, 170, 175, 177, 180, 183, 186, 192, 202, 209, 221, 224, 229, 235, 246, 253, 267, 268, 279, 285, 286, 304, 309, 316, 318–320, 324, 341, 346, 360, 390, 401, 440, 452, 467, 478, 491, 500, 503, 518, 527, 543, 548, 552, 572, 582, 589, 605, 622, 647, 664, 676 |
| <code>\cb@writeAux</code> | .. 399, 417, <u>440</u> , 454, 468 | <code>\deletebarwidth</code> | 4, 278, <u>290</u> |
| <code>\cbcolor</code> | 3, <u>302</u> , 316, 322 | <code>\do</code> | .. 161, 162, 367, 382 |
| <code>\cbdelete</code> | 3, <u>276</u> | <code>\driver</code> | <u>143</u> |
| <code>\cbend</code> | 3, <u>275</u> | <code>\DVIPS</code> | 149, 156 |
| <code>\cbstart</code> | 3, <u>272</u> | <code>\DVIToPS</code> | 148, 155 |
| <code>\changebar</code> | <u>280</u> | E | |
| changebar (environ- ment) | 3 | <code>\edef</code> | 145, |
| changebargrey | 4, <u>16</u> | <code>\changebarsep</code> | 418, 420, 443, 444, 446, 671, <u>682</u> |
| | | <code>\egroup</code> | 160 |
| | | <code>\else</code> | 42, 73, 92, 120, 265, 299, 333, 352, 366, 375, 376, 395, 408, 412, 423, 424, 428, 429, 431, 434, 435, 481, 494, 506, 509, 512, 530, 539, 576, 579, 593, 609, 627, 642, 651, 669, 680 |
| | | <code>\emTeX</code> | 150, 157 |
| | | <code>\end@dblfloat</code> | . 621, 660 |
| | | <code>\end@float</code> | <u>588</u> , 618, 643 |
| | | <code>\endchangebar</code> | <u>280</u> |
| | | <code>\endcsname</code> | ... 307, 313 |
| | | <code>\endgroup</code> | 449 |
| | | <code>\endlinechar</code> | 531 |
| | | environments:changebar | |
| | | changebar | 3 |
| | | <code>\evensidemargin</code> | 35, 70, 88 |
| | | <code>\ExecuteOptions</code> | 119, 121 |
| | | <code>\expandafter</code> | 110, 146, 188, 195, 204, 212, 231, 238, 248, 256, 312, 489, 497, 515, 540 |
| | | F | |
| | | <code>\fi</code> | . 45, 75, 94, 122, 141, 154–159, 164, 234, 243, 270, 301, 338, 345, 358, 373, 379, 381, 385, 398, 415, 416, 423, 429, 430, 435–438, 450, 490, 498, 514, 516, 517, 534, 535, 541, 578, 581, 601, 617, 635, 640, 641, 644, 659, 673, 684 |
| | | <code>\float@dblend</code> | <u>646</u> |
| | | <code>\float@end</code> | <u>604</u> |
| | | <code>\flt@float@dblend</code> | 646, 661 |

| | | | |
|---------------------------------|--|----------------------------------|--|
| <code>\flt@float@end</code> | 604, 619 | | |
| G | | L | |
| <code>\gdef</code> | 47, 586 | <code>\let</code> | 43, 44, 74, 76, 77, 93, 95, 96, 101, 132–134, 200, 219, 227, 244, 263, 269, 271, 287, 312, 445, 451, 454, 465, 466, 468, 477, 571, 587, 588, 603, 604, 618, 621, 645, 646, 660, 663, 674, 675, 685 |
| <code>\GenericWarning</code> | 138 | <code>\level</code> | 444, 447 |
| <code>\global</code> | 25–33, 35–41, 43, 44, 48–59, 64–68, 70–72, 74, 76, 77, 81–86, 88–91, 93, 95, 96, 153–159, 271, 342, 344, 460, 474, 537, 538, 544–546, 555, 556, 595, 597, 611, 613, 629, 631, 653, 655 | <code>\LN</code> | 147, 154 |
| H | | <code>\long</code> | 664, 676 |
| <code>\hoffset</code> | 25, 64, 81 | <code>\ltx@end@dblfloat</code> | 621, 660 |
| <code>\ht</code> | 335, 337 | <code>\ltx@end@float</code> | 588, 602, 618 |
| I | | <code>\ltx@footnotetext</code> | 663, 668, 672 |
| <code>\if@cb@trace</code> | 19, 137, 234, 243 | <code>\ltx@makecol</code> | 451, 462 |
| <code>\if@compatibility</code> | 143 | <code>\ltx@mpfootnotetext</code> | 675, 679, 683 |
| <code>\if@filesw</code> | 441 | <code>\ltx@vtryfc</code> | 466, 476 |
| <code>\if@firstcolumn</code> | 423 | M | |
| <code>\if@twocolumn</code> | 422, 623 | <code>\m@ne</code> | 330, 355, 556 |
| <code>\if@twoside</code> | 34, 69, 87, 409, 429, 435 | <code>\maxdepth</code> | 459, 473, 484 |
| <code>\ifcase</code> | 176 | <code>\MessageBreak</code> | 113, 114, 116, 126–129, 168, 173, 321, 560, 584 |
| <code>\ifeof</code> | 529 | N | |
| <code>\ifnum</code> | 343, 349, 365, 370, 375, 480, 493, 505, 507, 574, 575, 592, 608, 626, 637, 641, 650, 667, 678 | <code>\newcommand</code> | 136, 272, 275, 276, 284 |
| <code>\ifodd</code> | 376, 392, 405, 426, 432, 510 | <code>\newcount</code> | 5, 7–12, 15, 16 |
| <code>\ifouterbars</code> | 18, 425 | <code>\newcounter</code> | 294 |
| <code>\ifvmode</code> | 331 | <code>\newdimen</code> | 13, 14, 20–23 |
| <code>\ifx</code> | 118, 154–159, 297, 528, 536 | <code>\newenvironment</code> | 280 |
| <code>\ignorespaces</code> | 308, 314, 359 | <code>\newif</code> | 18, 19 |
| <code>\immediate</code> | 522, 552, 564, 565, 569, 570 | <code>\newlength</code> | 288, 290, 292 |
| <code>\InputIfFileExists</code> | 123 | <code>\newread</code> | 521 |
| J | | <code>\newwrite</code> | 520 |
| <code>\jobname</code> | 522, 565, 570 | <code>\next</code> | 145, 146 |
| | | <code>\nochangebars</code> | 3, 284 |
| | | <code>\noexpand</code> | 103, 105, 145, 161, 419, 421 |
| | | <code>\normalcolor</code> | 599, 615, 633, 657 |
| | | O | |
| | | <code>\oddsidemargin</code> | 27, 66, 83 |
| | | <code>\openin</code> | 565, 570 |
| | | <code>\openout</code> | 522 |
| | | <code>\OptionNotUsed</code> | 111 |
| | | <code>\or</code> | 185, 201, 220, 228, 245, 264 |
| | | <code>outerbars</code> | 4, 7 |
| | | <code>\outerbarsfalse</code> | 61 |
| | | <code>\outerbarstrue</code> | 60 |
| | | P | |
| | | <code>\p</code> | 47 |
| | | <code>\PackageError</code> | 111 |
| | | <code>\PackageWarning</code> | 266, 320, 350, 583 |
| | | <code>\pdfoutput</code> | 297 |
| | | <code>\point</code> | 443, 447 |
| | | <code>\ProcessOptions</code> | 135 |
| | | R | |
| | | <code>\read</code> | 531 |
| | | <code>\relax</code> | 48–59, 135, 153, 200, 219, 227, 244, 263, 269, 271, 287, 326, 342, 344, 361–363, 386–388, 485, 522, 538, 544–546, 574, 575 |
| | | <code>\RequirePackage</code> | 298, 300 |
| | | S | |
| | | <code>\setbox</code> | 455, 469, 482, 597, 613, 631, 655 |
| | | <code>\setcounter</code> | 295 |
| | | <code>\setlength</code> | 289, 291, 293 |
| | | <code>\space</code> | 103, 105, 108, 129, 181, 184, 188–190, 194–197, 204–207, 211–213, 215, 216, 231–233, 237, 238, 240, 241, 248–251, 255–257, 259, 260, 322, 559, 561 |
| | | <code>\special</code> | 178, 181, 184, 187, 193, 203, 210, 222, 225, 230, 236, 247, 254 |
| | | <code>\string</code> | 310, 313, 322, 447 |
| | | <code>\strutbox</code> | 335, 337 |

| | | | |
|-------------------------------|--------------------|--|--|
| T | | | |
| <code>\t</code> | 47 | 453, 463, 496, | 455, 469, 482, |
| <code>\tempa</code> | 145, 154–159 | 549, 559–562, | 600, 616, 634, 658 |
| <code>\Textures</code> | 151, 158 | 590, 606, 624, | <code>\vskip</code> 279, 335, 337 |
| <code>\textwidth</code> | 29, 39, 84, 89 | 648, 665, 671, 682 | <code>\vss</code> |
| <code>\the</code> | 103, 105, | <code>\thechangebargrey</code> . | 279 |
| | 167, 169, 172, | <code>\thepage</code> | <code>\VTeX</code> |
| | 174, 178, 181, | <code>\thr@@</code> | 152, 159 |
| | 188–190, 195– | <code>\tw@</code> | <code>\VTeXversion</code> |
| | 197, 204, 206, | | 118 |
| | 207, 212, 215, | W | |
| | 216, 222, 225, | <code>\write</code> | |
| | 231–233, 238, | 446, 552 | |
| | 240, 241, 248, | X | |
| | 250, 251, 256, | <code>\xdef</code> | |
| | 259, 260, 347, | 162, | |
| | 443–445, 447, | 518, 532, 547, 549 | |
| | | Z | |
| U | | <code>\z@</code> 279, 334, 445, 538, 637 | |
| <code>\undefined</code> ... | 118, 297 | | |
| <code>\unvbox</code> | 458, 472, 483, | | |
| | 600, 616, 634, 658 | | |
| <code>\uppercase</code> | 146 | | |
| V | | | |
| <code>\vbox</code> | 279, 334, | | |