

# The Changebar package \*

Michael Fine  
Distributed Systems Architecture

Johannes Braams  
Kersengarde 33  
2723 BP Zoetermeer  
The Netherlands  
JLBraams@cistron.nl

Printed October 22, 2001

## Contents

		2.2	Macros defined by the package . . . . .	2
1	Introduction	1	2.3	Changebar parameters . . . . . 3
2	The user interface	2	3	Deficiencies and bugs . . . . . 3
2.1	The package options . . . . .	2	4	The basic algorithm . . . . . 4

## Abstract

This package implements a way to indicate modifications in a  $\text{\LaTeX}$ -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 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 only been tested with the `dvips` driver, but it may also work with other PostScript based drivers. It will *not* work for the `DVItoLN03` and `emTeX` drivers..

---

\*This file has version number v3.4d, last revised 2001/09/04.

## 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

One set of package options<sup>1</sup> specify the driver that will be used to print the document can be indicated. The driver may be one of:

- DVItolN03
- DVItolPS
- DVIPs
- emTeX
- VTeX

The drivers are represented in the normal typewriter method of typing these names, or by the same entirely in lower case.

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`).

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

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 this package maintains.

### 2.2 Macros defined by the package

<code>\cbstart</code> <code>\cbend</code>	All material between the macros <code>\cbstart</code> and <code>\cbend</code> is barred. The nesting of multiple changebars is allowed. The macro <code>\cbstart</code> has an optional parameter that specifies the width of the bar. The syntax is <code>\cbstart[<i>&lt;dimension&gt;</i>]</code> . If no width is specified, the current value of the parameter <code>\changebarwidth</code> is used. Note that <code>\cbstart</code> and <code>\cbend</code> 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.
<code>changebar</code>	Apart from the macros <code>\cbstart</code> and <code>\cbend</code> a proper $\text{\LaTeX}$ environment is defined. The advantage of using the environment whenever possible is that $\text{\LaTeX}$ will do all the work of checking the correct nesting of different environments.
<code>\cbdelete</code>	The macro <code>\cbdelete</code> 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 <code>\deletebarwidth</code> will be used.
<code>\nochangebars</code>	The macro <code>\nochangebars</code> disables the changebar commands.
<code>\cbcolor</code>	This macro is defined when the <code>color</code> option is selected. It's syntax is the same as the <code>\color</code> command from the <code>color</code> package.

---

<sup>1</sup>For older documents the command `\driver` is available in the preamble of the document. It takes the options as defined for  $\text{\LaTeX}$  2<sub>ε</sub> as argument.

## 2.3 Changebar parameters

<code>\changebarwidth</code>	The width of the changebars is controlled with the $\text{\LaTeX}$ length parameter <code>\changebarwidth</code> . Its value can be changed with the <code>\setlength</code> command. Changing the value of <code>\changebarwidth</code> affects all subsequent changebars subject to the scoping rules of <code>\setlength</code> .
<code>\deletebarwidth</code>	The width of the deletebars is controlled with the $\text{\LaTeX}$ length parameter <code>\deletebarwidth</code> . Its value can be changed with the <code>\setlength</code> command. Changing the value of <code>\changebarwidth</code> affects all subsequent deletebars subject to the scoping rules of <code>\setlength</code> .
<code>\changebarsep</code>	The separation between the text and the changebars is determined by the value of the $\text{\LaTeX}$ length parameter <code>\changebarsep</code> .
<code>changebargrey</code>	When one of the supported dvi to PostScript translators is used the ‘blackness’ of the bars can be controlled. The $\text{\LaTeX}$ counter <code>changebargrey</code> 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.

<code>outerbars</code>	The changebars will be printed in the ‘inside’ margin of your document. This means they appear on the left side of the page. When <code>twoside</code> is in effect the bars will be printed on the right side of even pages. This behaviour can be changed by including the command <code>\outerbarstrue</code> 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  $(\text{\cb@maxpoint} - \text{\cb@minpoint} + 1)/4$  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  $\text{\LaTeX}$  floats are not split but footnotes may be. The simplest fix to this is to prevent footnotes from being split but this may make  $\text{\TeX}$  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 DVItolN03 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  $\text{\TeX}$  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  $\text{\LaTeX}$  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  $\text{\LaTeX}$  macros check for this condition and add the extra bar.