

# The atbegshi package

Heiko Oberdiek  
<oberdiek@uni-freiburg.de>

2008/07/31 v1.9

## Abstract

This package is a modern reimplementaion of package `everyshi` without the burden of compatibility. It makes use of  $\varepsilon$ -TeX's if available. Both L<sup>A</sup>T<sub>E</sub>X and plain-T<sub>E</sub>X are supported.

## Contents

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	Examples	3
1.1.1	Example: circle in background	3
1.1.2	Example: adding TrimBox for dvipdfmx	4
<b>2</b>	<b>Method of <code>\shipout</code> overloading</b>	<b>5</b>
2.1	<code>\shipout</code>	5
2.2	<code>\afterassignment</code>	5
2.3	Test for direct or indirect boxes	6
2.3.1	With $\varepsilon$ -TeX	6
2.3.2	Without $\varepsilon$ -TeX	6
2.3.3	<code>\lastkern</code> method	7
2.4	Output	8
2.5	Separate box register	8
2.6	Summary	8
2.6.1	With $\varepsilon$ -TeX	8
2.6.2	Without $\varepsilon$ -TeX, traditional way	9
2.6.3	<code>\lastkern</code> method	9
<b>3</b>	<b>Implementation</b>	<b>10</b>
3.1	Reload check and package identification	10
3.2	Catcodes	11
3.3	Preparations	11
3.4	Positioning	15
3.5	Patches	17
3.5.1	Package <code>crop</code>	17
3.5.2	Package <code>everyshi</code>	18
3.5.3	Class <code>memoir</code>	19
<b>4</b>	<b>Test</b>	<b>21</b>
4.1	Catcode checks for loading	21
<b>5</b>	<b>Installation</b>	<b>25</b>
5.1	Download	25
5.2	Bundle installation	25
5.3	Package installation	25
5.4	Refresh file name databases	26
5.5	Some details for the interested	26

<b>6 History</b>	<b>26</b>
[2007/04/17 v1.0]	26
[2007/04/18 v1.1]	27
[2007/04/19 v1.2]	27
[2007/04/26 v1.3]	27
[2007/04/27 v1.4]	27
[2007/06/06 v1.5]	27
[2007/09/09 v1.6]	27
[2008/07/18 v1.7]	27
[2008/07/19 v1.8]	27
[2008/07/31 v1.9]	27
<b>7 Index</b>	<b>27</b>

## 1 Documentation

Package `atbegshi` redefines `\shipout` to insert hooks for user code that is executed before the page is shipped out. The code may modify or even discard the output page. Three hooks are implemented:

1. A hook that is executed for every page, see `\AtBeginShipout`
2. A hook that is executed for the next page only, see `\AtBeginShipoutNext`
3. A hook that is only executed for the first page, see `\AtBeginShipoutFirst`

The hooks are executed in this order. The following three macros provide the user interface for adding code to these hooks:

`\AtBeginShipout {⟨code⟩}`

Execute the `⟨code⟩` for every page. The page contents is held in box register `\AtBeginShipoutBox` and may be modified. Use `\AtBeginShipoutDiscard` if you want to discard the page.

*Note:* Package `everyshi` uses box register 255. With package `atbegshi` you must use `\AtBeginShipoutBox` instead.

If  $\text{\LaTeX}$  calls `\shipout` in `\@outputpage` (part of its output routine), the meaning of `\protect` is `\noexpand`.  $\text{\LaTeX}$  sets `\protect` to the appropriate `\@typeset@protect` in the box that is shipped out. This is too late for the hooks, they are called earlier in the redefined `\shipout`. Therefore package `atbegshi` sets `\protect` to `\@typeset@protect` before it calls the hooks. (In `\EveryShipout` of package `everyshi` the user is responsible for the correct setting of `\protect`.)

`\AtBeginShipoutNext {⟨code⟩}`

This reimplements package `everyshi`'s `\AtNextShipout`. The `⟨code⟩` is executed at shipout time of the next page only. It is just a convenience macro, it can be easily replaced by something like:

```

\newcommand{\MyShipoutHook}{}%
\AtBeginShipout{\MyShipoutHook}
\gdef\MyShipoutHook{%
  ... do something with next page ...
  \gdef\MyShipoutHook{}%
}
```

(This can be necessary, if hook order does matter).

`\AtBeginShipoutFirst {<code>}`

This reimplements L<sup>A</sup>T<sub>E</sub>X's `\AtBeginDvi`. This hook is usually used for `\special` commands that include PostScript header files. The `<code>` is directly executed in a `\vbox` that is put at the beginning of the output page. Dealing with the output box `\AtBeginShipoutBox` is not necessary and not permitted here.

`\AtBeginShipoutDiscard`

This macro notifies package `atbegshi` that the output page is discarded. The remaining hook code and the remaining hooks are not executed and the page is thrown away. Also `\deadcycles` is cleared to zero like an ordinary `\shipout` would do.

`\AtBeginShipoutInit`

Usually the redefinition of `\shipout` is delayed by `\AtBeginDocument` (if this macro exists). This can be too late, if other packages also redefines `\shipout` and the order does matter. `\AtBeginShipoutInit` forces the immediate redefinition of `\shipout`.

`\AtBeginShipoutUpperLeft {<background material>}`

This is a macro that puts material in the background of box `\AtBeginShipoutBox`. The *<background material>* is set in an `\hbox`, the reference point is the upper left corner of the output page. In case of pdf<sub>T</sub>E<sub>X</sub> in PDF mode, the settings of `\pdfhorigin` and `\pdfvorigin` are respected.

The macro `\AtBeginShipoutUpperLeft` is intended to be used in one of the hook setting macros, such as `\AtBeginShipout`, `\AtBeginShipoutFirst`, or `\AtBeginShipoutNext`.

For L<sup>A</sup>T<sub>E</sub>X users the *<background material>* is set inside a `picture` environment:

```
\begin{picture}(0,0)
  \setlength{\unitlength}{1pt}%
  <background material>
\end{picture}
```

`\AtBeginShipoutUpperLeftForeground {<foreground material>}`

See `\AtBeginShipoutUpperLeft`. The difference is that the material is put in the foreground.

## 1.1 Examples

### 1.1.1 Example: circle in background

In this example we put a circle in the background in the middle of the paper.

```
1 (*example1)
2 \documentclass[a4paper]{article}
3 \usepackage{color}
4 \usepackage{atbegshi}
```

Package `picture` makes life a little easier, because we can now also use length specifications in `picture`'s commands.

```
5 \usepackage{picture}
```

Now we draw the circle in the middle of the paper. `\put` moves downwards, because the origin is at the top of the page, not at its bottom.

```

6 \AtBeginShipout{%
7   \AtBeginShipoutUpperLeft{%
8     \put(0.5\paperwidth,-0.5\paperheight){\circle{10}}%
9   }%
10 }
11 \begin{document}
12 \section{Hello World}
13 \newpage
14 \AtBeginShipoutNext{%
15   \AtBeginShipoutUpperLeft{%
16     \color{red}%
17     \put(0,-0.5\paperheight){\line(1,0){\paperwidth}}%
18     \put(0.5\paperwidth, 0){\line(0,-1){\paperheight}}%
19   }%
20 }
21 Only on this page we add a red cross.
22 \newpage
23 This page has the circle only.
24 \par
25 \vspace{\fill}
26 The next page will be discarded.
27 \newpage
28 \AtBeginShipoutNext{%
29   \AtBeginShipoutDiscard
30 }
31 This page is discarded.
32 \newpage
33 The last page.
34 \end{document}
35 \end{example1}

```

### 1.1.2 Example: adding TrimBox for dvipdfmx

Now an example from “real life” follows. Someone from the mailing list for dvipdfmx wants to put a TrimBox on every page. If we use `\AtBeginShipout`, we have to put the `\special` inside the box `\AtBeginShipoutBox` that gets shipped out.

```

36 \begin{example2}
37 \documentclass{minimal}
38 \usepackage{atbegshi}
39 \usepackage[
40   dvipdfm,
41   paperwidth=630bp,
42   paperheight=810bp
43 ]{geometry}
44 \AtBeginShipout{%
45   \setbox\AtBeginShipoutBox=\hbox{%
46     \special{pdf: put @thispage <</TrimBox[9 9 621 801]>>}}%
47   \box\AtBeginShipoutBox
48 }%
49 }
50 \begin{document}
51   First page
52   \newpage
53   Second page
54 \end{document}
55 \end{example2}

```

Remember, in `\AtBeginShipoutBoxFirst` the `\setbox` wrapper code is implicitly given and the `\special` is used directly.

## 2 Method of `\shipout` overloading

### 2.1 `\shipout`

The TeX primitive command `\shipout` takes a box specification and puts the box as a new page in the output file. There are two kinds of box specifications:

**Direct boxes:** They are given by `\hbox`, `\vbox`, or `\vtop`,  
e.g. `\shipout\hbox{Hello World}`.

**Indirect boxes:** `\box` or `\copy` references a box register by number. The box register contains the contents of the box.

*Note:* `\box` also clears the box register globally.

Then we have to differentiate between void and empty boxes:

**Void:** Initially or after `\box` there is no box in the box register. In this cases the box register is not empty, but *void*.

**Empty:** A box with empty contents, such as `\hbox{}` ( $= \text{\null}$ ) or `\vbox{}` is an *empty hbox* or *empty vbox*. If a box register holds such a box, the box still exists, therefore the box register is *not void*.

### 2.2 `\afterassignment`

We want to overload `\shipout` to do something with the box. It is quite impossible to do this reliable by catching the box using macro arguments. The variety of box specifications is too large, Examples:

```
\shipout\null
\shipout\vbox{...}
\shipout\vtop\bgroup ... \egroup
\shipout\box255
```

Even worse, the braces don't need to be balanced:

```
\shipout\hbox\bgroup}
\shipout\vbox{\egroup
```

Happily TeX provides a reliable way via `\afterassignment`. It takes a macro name and executes it just after the assignment.

Now we can redefine `\shipout`. The box specification that follows `\shipout` is caught by `\setbox`. This is an assignment to a box register. `\afterassignment` notifies TeX, that we want to call `\@test` right after the assignment:

```
\shipout :=
  \afterassignment\@test
  \setbox\mybox=
```

We have seen different box specifications. Indirect boxes are easy to understand:

```
\shipout\box0  $\Rightarrow$  \setbox\mybox=\box0 \@test
```

However direct boxes can have arbitrary contents with lots of other assignments. It would be quite unpredictable if TeX would put `\@test` after the first of such an assignment or after the box specification if the box lacks of assignments. Therefore TeX puts `\@test` right at the beginning of the box specification, e.g:

```
\shipout\hbox{Hello World}
 $\Rightarrow$  \setbox\mybox=\hbox{\@test Hello World}
```

## 2.3 Test for direct or indirect boxes

Now we want to execute `\@test`, but where are we? We can be after the completed box assignment, if `\shipout` was called with an indirect box. Or we are right at the beginning of a direct box.

### 2.3.1 With $\varepsilon$ -TeX

With the  $\varepsilon$ -TeX's extensions the answer is very easy: Being inside the direct box means that we are inside a new group. The new primitive command `\currentgrouplevel` tells how deeply the groups are currently nested. Macro `\@test` just compares the previously stored group level with the current one:

```
\shipout :=
  \edef\saved@grouplevel{\number\currentgrouplevel}
  \afterassignment\@test
  \setbox\mybox=

\@test :=
  \ifnum\saved@grouplevel=\currentgrouplevel
    % case: indirect box, the assignment is completed
    \@output
  \else
    % case: direct box, we are inside the box
    \aftergroup\@outbox
  \fi
```

### 2.3.2 Without $\varepsilon$ -TeX

Life becomes complicate without  $\varepsilon$ -TeX. We cannot ask the group level. However, if we are inside a direct box, the box register `\mybox` is not yet changed by `\setbox`. Thus we need a special initial value and compare it in `\@test` with the current value of the box.

What can be used as initial value? Arbitrary box contents cannot be compared. TeX only tells us a few properties:

- Box type: `\ifhbox`, `\ifvbox`
- Dimensions: `\wd`, `\ht`, `\dp`
- Voidness: `\ifvoid`

Unhappily all these qualities even combined are not sufficient for constructing an initial box value, because `\shipout` can be called with a box that is accidentally just the same as the choosen initial value.

Nevertheless we have two alternatives for an initial value:

- A box of some type with some funny settings that are unlikely to occur in real life, e.g a height of `4911sp-\maxdimen`.
- A void box.

A collision between this initial value and an indirect `\shipout` box with just the same value is possible. Then `\@test` will make a wrong decision that it is executed inside a direct box and delays `\@output` by `\aftergroup`. Thus `\@output` is not called at the place we want. In contrary, the result is an uncertainty about the place:

- `\shipout` is used in a group that perhaps closes some pages later. A bad place for `\@output`.
- Without a surrounding group `\aftergroup` effectively kills its argument.

In the first case of a box with special dimensions we can even loose the page. However in the case of the void box, this effect is even desired, because the original `\shipout` does not output void boxes. All we have to do is to ensure that our box `\mybox` is always void except for the phase when the overloaded `\shipout` is executed. And secondly we must keep this semantics of `\shipout` for the void case in our macros, namely `\@output`.

```
\shipout :=
% trick to get a void box \mybox
\begingroup
\setbox\mybox=\box\mybox
\endgroup
\afterassignment\@test
\setbox\mybox=

\@test :=
\ifvoid\mybox
\aftergroup\@output
\else
\@output
\fi
```

The nasty case is `\shipout\box\voidb@x` where the indirect box is void and that must not generate an output page. If a surrounding group is missing the output is ignored because of `\aftergroup`. Otherwise output is called some time later when the surrounding group closes. But `\mybox` is void outside the execution phase of the redefined `\shipout`. Also `\@output` checks for a void box and cancels the page output. The disadvantage remains that the hook in `\@output` is called for a page that will not be output.

### 2.3.3 `\lastkern` method

At the beginning of a new box, there is no `\kern`, the contents of the box is still empty and `\lastkern` returns 0 pt. This can be used to distinguish between direct and indirect boxes: We execute `\setbox` in a box with a preceding non-zero kern. After an indirect box, `\lastkern` sees this kern, otherwise it returns 0 pt.

```
\shipout :=
\begingroup
\setbox\mybox=\hbox\bgroup
\kern1pt
\afterassignment\shipout@test
\global\setbox\mybox=
\@test :=
\ifdim\lastkern=0pt
% direct box
\aftergroup\egroup
\aftergroup\endgroup
\aftergroup\@output
\else
\egroup
\endgroup
\@output
\fi
```

We have two `\setbox` commands. The first creates a controlled context box where we can safely insert a `\kern`. We get rid of this temporarily used context box by putting the local `\setbox` in a group.

After the group we want to have our shipout box in `\mybox`. Therefore we use a global assignment here.

## 2.4 Output

With or without  $\varepsilon$ -TeX we ensure the original behaviour of `\shipout` that void boxes do not generate output pages.

Now we can place the hook `\@hook` for the user code that wants to manipulate the output box.

```
\@output :=
\ifvoid\mybox
% cancel output of void box
\else
\@hook
\ifvoid\mybox
% user code in \@hook could has voided the box
\else
\original@shipout\box\mybox
\fi
\fi
```

## 2.5 Separate box register

So far we have said nothing about the box number of `\mybox`. The following case that outputs the same page twice shows that we are not free in the use of the box register:

```
\shipout\copy<num> \shipout\box<num>
```

We manipulate the box by the hook and without  $\varepsilon$ -TeX the box must even be voided. However, the use case above requires that the box contents does not change at all. Therefore we must reserve a separate box register to avoid collisions with user box registers.

*Note:* Box register number 255 is special for the output routine, because TeX complains if this box is not voided by the output routine. However, this requirement does not apply to `\shipout` at all. In fact `\shipout` does not change any box register. This is usually done by a call of `\box`, but the output routine can do it later *after* invoking of `\shipout`.

## 2.6 Summary

### 2.6.1 With $\varepsilon$ -TeX

Putting the pieces together we get for  $\varepsilon$ -TeX:

```
\newbox\mybox
\let\original@shipout\shipout

\shipout :=
\edef\saved@grouplevel{\number\currentgrouplevel}
\afterassignment\@test
\setbox\mybox=

\@test :=
\ifnum\saved@grouplevel<\currentgrouplevel
\expandafter\aftergroup
\fi
\@output

\@output :=
\ifvoid\mybox
% cancel output of void box
\else
\@hook
\ifvoid\mybox
```



```

        % user code in \@hook could have voided the box
    \else
        \original@shipout\box\mybox
    \fi
\fi

```

## 2.6.2 Without $\varepsilon$ -TeX, traditional way

And for TeX without  $\varepsilon$ -TeX:

```

\newbox\mybox
\begingroup
    \setbox\mybox=\box\mybox % ensure \mybox is void
\endgroup
\let\original@shipout\shipout

\shipout :=
    % trick to get a void box \mybox
    \begingroup
        \setbox\mybox=\box\mybox
    \endgroup
    \afterassignment\@test
    \setbox\mybox=

\@test :=
    \ifvoid\mybox
        \expandafter\aftergroup
    \fi
    \@output

\@output :=
    \ifvoid\mybox
        % cancel output of void box
    \else
        \@hook
        \ifvoid\mybox
            % user code in \@hook could have voided the box
        \else
            \original@shipout\box\mybox
        \fi
    \fi

```

## 2.6.3 \lastkern method

And for TeX without  $\varepsilon$ -TeX using the \lastkern method:

```

\newbox\mybox
\let\original@shipout\shipout

\shipout :=
    \begingroup
        \setbox\mybox=\hbox\bgroup
        \kern1pt
        \afterassignment\@test
        \setbox\mybox=

\@test :=
    \ifdim\lastkern=0pt
        \expandafter\aftergroup
    \fi
    \@output

\@output :=

```

```

\egroup
\endgroup
\ifvoid\mybox
% cancel output of void box
\else
\@hook
\ifvoid\mybox
% user code in \@hook could have voided the box
\else
\original@shipout\box\mybox
\fi
\fi

```

### 3 Implementation

Package atbegshi uses  $\varepsilon$ -TeX's `\currentgrouplevel`, if it is available. Otherwise the `\lastkern` method is used.

```
56 <*package>
```

#### 3.1 Reload check and package identification

Reload check, especially if the package is not used with L<sup>A</sup>T<sub>E</sub>X.

```

57 \begingroup
58 \catcode44 12 % ,
59 \catcode45 12 % -
60 \catcode46 12 % .
61 \catcode58 12 % :
62 \catcode64 11 % @
63 \catcode123 1 % {
64 \catcode125 2 % }
65 \expandafter\let\expandafter\x\csname ver@atbegshi.sty\endcsname
66 \ifx\x\relax % plain-TeX, first loading
67 \else
68 \def\empty{}%
69 \ifx\x\empty % LaTeX, first loading,
70 % variable is initialized, but \ProvidesPackage not yet seen
71 \else
72 \catcode35 6 % #
73 \expandafter\ifx\csname PackageInfo\endcsname\relax
74 \def\x#1#2{%
75 \immediate\write-1{Package #1 Info: #2.}%
76 }%
77 \else
78 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
79 \fi
80 \x{atbegshi}{The package is already loaded}%
81 \aftergroup\endinput
82 \fi
83 \fi
84 \endgroup

```

Package identification:

```

85 \begingroup
86 \catcode35 6 % #
87 \catcode40 12 % (
88 \catcode41 12 % )
89 \catcode44 12 % ,
90 \catcode45 12 % -
91 \catcode46 12 % .
92 \catcode47 12 % /
93 \catcode58 12 % :

```

```

94 \catcode64 11 % @
95 \catcode91 12 % [
96 \catcode93 12 % ]
97 \catcode123 1 % {
98 \catcode125 2 % }
99 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
100 \def\x#1#2#3[#4]{\endgroup
101 \immediate\write-1{Package: #3 #4}%
102 \xdef#1{#4}%
103 }%
104 \else
105 \def\x#1#2[#3]{\endgroup
106 #2[{#3}]%
107 \ifx#1\@undefined
108 \xdef#1{#3}%
109 \fi
110 \ifx#1\relax
111 \xdef#1{#3}%
112 \fi
113 }%
114 \fi
115 \expandafter\x\csname ver@atbegshi.sty\endcsname
116 \ProvidesPackage{atbegshi}%
117 [2008/07/31 v1.9 At begin shipout hook (H0)]

```

### 3.2 Catcodes

```

118 \begingroup
119 \catcode123 1 % {
120 \catcode125 2 % }
121 \def\x{\endgroup
122 \expandafter\edef\csname AtBegShi@AtEnd\endcsname{%
123 \catcode35 \the\catcode35\relax
124 \catcode64 \the\catcode64\relax
125 \catcode123 \the\catcode123\relax
126 \catcode125 \the\catcode125\relax
127 }%
128 }%
129 \x
130 \catcode35 6 % #
131 \catcode64 11 % @
132 \catcode123 1 % {
133 \catcode125 2 % }
134 \def\TMP@EnsureCode#1#2{%
135 \edef\AtBegShi@AtEnd{%
136 \AtBegShi@AtEnd
137 \catcode#1 \the\catcode#1\relax
138 }%
139 \catcode#1 #2\relax
140 }
141 \TMP@EnsureCode{40}{12}% (
142 \TMP@EnsureCode{41}{12}% )
143 \TMP@EnsureCode{44}{12}% ,
144 \TMP@EnsureCode{45}{12}% -
145 \TMP@EnsureCode{47}{12}% /
146 \TMP@EnsureCode{46}{12}% .
147 \TMP@EnsureCode{58}{12}% :
148 \TMP@EnsureCode{61}{12}% =
149 \TMP@EnsureCode{94}{7}% ^ (superscript)
150 \TMP@EnsureCode{96}{12}% '

```

### 3.3 Preparations

```

151 \begingroup\expandafter\expandafter\expandafter\endgroup
152 \expandafter\ifx\csname RequirePackage\endcsname\relax
153   \input infwarerr.sty\relax
154 \else
155   \RequirePackage{infwarerr}[2007/09/09]%
156 \fi

```

\AtBegShi@CheckDefinable

```

157 \begingroup\expandafter\expandafter\expandafter\endgroup
158 \expandafter\ifx\csname @ifdefinable\endcsname\relax
159   \def\AtBegShi@CheckDefinable#1{%
160     \ifcase\ifx#1\relax
161       \@ne
162     \else
163       \ifx#1\@undefined
164         \@ne
165       \else
166         \z@
167       \fi
168     \fi
169     \errmessage{%
170       Package atbegshi: \string#1\space
171       is already defined%
172     }%
173   \endgroup
174 \fi
175 }%
176 \else
177   \def\AtBegShi@CheckDefinable#1{%
178     \@ifdefinable{#1}{}%
179   }%
180 \fi

181 \newif\ifAtBegShi@Discarded

```

\AtBeginShipoutDiscard

```

182 \AtBegShi@CheckDefinable\AtBeginShipoutDiscard
183 \def\AtBeginShipoutDiscard{%
184   \deadcycles=\z@
185   \global\AtBegShi@Discardedtrue
186 }

187 \begingroup\expandafter\expandafter\expandafter\endgroup
188 \expandafter\ifx\csname currentgrouplevel\endcsname\relax
189   \catcode'X=9 % ignore
190   \catcode'E=14 % comment
191 \else
192   \catcode'X=14 % comment
193   \catcode'E=9 % ignore
194 \fi

```

\AtBegShi@Shipout

```

195 \def\AtBegShi@Shipout{%
196 X \begingroup
197 X \setbox\AtBeginShipoutBox=\hbox\bgroup
198 X \kern\p@
199 E \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
200 \afterassignment\AtBegShi@Test
201 X \global
202 \setbox\AtBeginShipoutBox=%
203 }

```

\AtBegShi@Test

```
204 \def\AtBegShi@Test{%
205 X \ifdim\lastkern=\z@
206 E \ifnum\AtBegShi@GroupLevel<\currentgrouplevel
207 \expandafter\aftergroup
208 \fi
209 \AtBegShi@Output
210 }
```

\AtBegShi@Output

```
211 \def\AtBegShi@Output{%
212 X \egroup
213 X \endgroup
214 \ifvoid\AtBeginShipoutBox
215 \@PackageWarning{atbegshi}{Ignoring void shipout box}%
216 \else
217 \let\AtBegShi@OrgProtect\protect
218 \csname set@typeset@protect\endcsname
219 \global\AtBegShi@Discardedfalse
220 \AtBegShi@Hook
221 \AtBegShi@HookNext
222 \gdef\AtBegShi@HookNext{}%
223 \ifAtBegShi@Discarded
224 \@PackageInfoNoLine{atbegshi}{Shipout page discarded}%
225 \global\AtBegShi@Discardedfalse
226 \begingroup
227 \setbox\AtBeginShipoutBox\box\AtBeginShipoutBox
228 \endgroup
229 \let\protect\AtBegShi@OrgProtect
230 \else
231 \AtBegShi@First
232 \let\protect\AtBegShi@OrgProtect
233 \AtBegShi@OrgShipout\box\AtBeginShipoutBox
234 \fi
235 \fi
236 }

237 \catcode'\X=11 %
238 \catcode'\E=11 %
```

\AtBegShi@First

```
239 \def\AtBegShi@First{%
240 \begingroup
241 \def\@empty{}%
242 \ifx\AtBegShi@HookFirst\@empty
243 \else
244 \setbox\z@=\vbox{%
245 \begingroup
246 \AtBegShi@HookFirst
247 \endgroup
248 }%
249 \wd\z@=\z@
250 \ht\z@=\z@
251 \dp\z@=\z@
252 \global\setbox\AtBeginShipoutBox=\vbox{%
253 \baselineskip\z@skip
254 \lineskip\z@skip
255 \lineskiplimit\z@
256 \copy\z@
257 \copy\AtBeginShipoutBox
258 }%
259 \fi
```

```

260     \global\let\AtBegShi@First\@empty
261     \global\let\AtBeginShipoutFirst\AtBegShi@FirstDisabled
262 \endgroup
263 }

\AtBegShi@Hook
264 \gdef\AtBegShi@Hook{}

\AtBegShi@HookNext
265 \gdef\AtBegShi@HookNext{}

\AtBegShi@HookFirst
266 \gdef\AtBegShi@HookFirst{}

\AtBeginShipout
267 \AtBegShi@CheckDefinable\AtBeginShipout
268 \def\AtBeginShipout{%
269     \AtBegShi@AddHook\AtBegShi@Hook
270 }

\AtBeginShipoutNext
271 \AtBegShi@CheckDefinable\AtBeginShipoutNext
272 \def\AtBeginShipoutNext{%
273     \AtBegShi@AddHook\AtBegShi@HookNext
274 }

\AtBeginShipoutFirst
275 \AtBegShi@CheckDefinable\AtBeginShipoutFirst
276 \def\AtBeginShipoutFirst{%
277     \AtBegShi@AddTo\AtBegShi@HookFirst
278 }

\AtBegShi@FirstDisabled
279 \long\def\AtBegShi@FirstDisabled#1{%
280     \@PackageWarning{atbegshi}{%
281         First page is already shipped out, ignoring\MessageBreak
282         \string\AtBeginShipoutFirst
283     }%
284 }

\AtBegShi@AddTo
285 \begingroup\expandafter\expandafter\expandafter\endgroup
286 \expandafter\ifx\csname g@addto@macro\endcsname\relax
287     \long\def\AtBegShi@AddTo#1#2{%
288         \begingroup
289             \toks\z@\expandafter{#1#2}%
290             \xdef#1{\the\toks\z0}%
291         \endgroup
292     }%
293 \else
294     \let\AtBegShi@AddTo\g@addto@macro
295 \fi

\AtBegShi@AddHook
296 \long\def\AtBegShi@AddHook#1#2{%
297     \AtBegShi@AddTo#1{\AtBegShi@Item{#2}}%
298 }

```

\AtBegShi@Item

```

299 \long\def\AtBegShi@Item#1{%
300   \ifAtBegShi@Discarded
301   \else
302     #1%
303     \ifvoid\AtBeginShipoutBox
304       \@PackageWarning{atbegshi}{%
305         Shipout box was voided by hook,\MessageBreak
306         ignoring shipout box%
307       }%
308       \AtBeginShipoutDiscard
309     \fi
310   \fi
311 }

```

\AtBeginShipoutInit

```

312 \AtBegShi@CheckDefinable\AtBeginShipoutInit
313 \def\AtBeginShipoutInit{%
314   \csname newbox\endcsname\AtBeginShipoutBox
315   \AtBegShi@CheckDefinable\AtBegShi@OrgShipout
316   \global\let\AtBegShi@OrgShipout\shipout
317   \global\let\shipout\AtBegShi@Shipout
318   \gdef\AtBeginShipoutInit{}%
319 }

320 \begingroup\expandafter\expandafter\expandafter\endgroup
321 \expandafter\ifx\csname AtBeginDocument\endcsname\relax
322   \AtBeginShipoutInit
323 \else
324   \AtBeginDocument{\AtBeginShipoutInit}%
325 \fi

```

### 3.4 Positioning

```

326 \begingroup\expandafter\expandafter\expandafter\endgroup
327 \expandafter\ifx\csname RequirePackage\endcsname\relax
328   \input ifpdf.sty\relax
329 \else
330   \RequirePackage{ifpdf}\relax
331 \fi

332 \ifpdf
333   \def\AtBegShi@horigin{\pdfhorigin}%
334   \def\AtBegShi@vorigin{\pdfvorigin}%
335 \else
336   \def\AtBegShi@horigin{72.27pt}%
337   \def\AtBegShi@vorigin{72.27pt}%
338 \fi

339 \begingroup
340 \ifcase
341   \expandafter\ifx\csname picture\endcsname\relax
342     1%
343   \else
344     \expandafter\ifx\csname endpicture\endcsname\relax
345       1%
346     \else
347       0%
348     \fi
349   \fi
350 \endgroup
351 \def\AtBegShi@BeginPicture{%
352   \begingroup

```

```

353     \picture(0,0)\relax
354     \begingroup\expandafter\expandafter\expandafter\endgroup
355     \expandafter\ifx\csname unitlength\endcsname\relax
356     \else
357         \unitlength=1pt\relax
358     \fi
359     \ignorespaces
360 }%
361 \def\AtBegShi@EndPicture{%
362     \endpicture
363     \endgroup
364 }%
365 \else
366     \endgroup
367     \def\AtBegShi@BeginPicture{%
368         \setbox0=\hbox\bgroup
369         \begingroup
370         \ignorespaces
371     }%
372     \def\AtBegShi@EndPicture{%
373         \endgroup
374         \egroup
375         \ht0=0pt\relax
376         \dp0=0pt\relax
377         \copy0 %
378     }%
379 \fi

380 \def\AtBeginShipoutUpperLeft#1{%
381     \global\setbox\AtBeginShipoutBox=\hbox{%
382         \rlap{%
383             \kern-\AtBegShi@horigin\relax
384             \vbox to 0pt{%
385                 \kern-\AtBegShi@vorigin\relax
386                 \kern-\ht\AtBeginShipoutBox
387                 \AtBegShi@BeginPicture
388                 #1%
389                 \AtBegShi@EndPicture
390                 \vss
391             }%
392         }%
393         \box\AtBeginShipoutBox
394     }%
395 }

396 \def\AtBeginShipoutUpperLeftForeground#1{%
397     \global\setbox\AtBeginShipoutBox=\hbox to \wd\AtBeginShipoutBox{%
398         \rlap{%
399             \copy\AtBeginShipoutBox
400         }%
401         \rlap{%
402             \kern-\AtBegShi@horigin\relax
403             \vbox to 0pt{%
404                 \kern-\AtBegShi@vorigin\relax
405                 \kern-\ht\AtBeginShipoutBox
406                 \AtBegShi@BeginPicture
407                 #1%
408                 \AtBegShi@EndPicture
409                 \vss
410             }%
411         }%
412         \hss
413     }%
414 }

```



### 3.5 Patches

Patches for L<sup>A</sup>T<sub>E</sub>X packages that redefine `\shipout`. L<sup>A</sup>T<sub>E</sub>X is now supposed to use  $\varepsilon$ -T<sub>E</sub>X. Thus we do not patch, without L<sup>A</sup>T<sub>E</sub>X and  $\varepsilon$ -T<sub>E</sub>X.

```

415 \def\AtBegShi@AbortIfUndefined#1{%
416   \begingroup\expandafter\expandafter\expandafter\endgroup
417   \expandafter\ifx\curname#1\endcurname\relax
418   \AtBegShi@AtEnd
419   \expandafter\endinput
420   \fi
421 }
422 \AtBegShi@AbortIfUndefined{currentgrouplevel}
423 \AtBegShi@AbortIfUndefined{AtBeginDocument}
424 \AtBegShi@AbortIfUndefined{@ifpackageloaded}
425 \AtBegShi@AbortIfUndefined{@ifclassloaded}

```

#### 3.5.1 Package crop

Fix of method and box.

```

426 \def\AtBegShi@PatchCrop{%
427   \begingroup
428   \def\AtBegShi@Crop@shipout{%
429     \afterassignment\CROP@ship
430     \setbox\@cclv=%
431   }%
432   \def\AtBegShi@Crop@ship{%
433     \ifvoid\@cclv
434     \expandafter\aftergroup
435     \fi
436     \CROP@@ship
437   }%
438   \def\AtBegShi@Crop@shiplist{%
439     \lineskip\z@
440     \lineskiplimit\z@
441     \baselineskip\z@
442     \CROP@kernel
443     \box\@cclv
444   }%
445   \def\AtBegShi@Crop@@ship{%
446     \CROP@shipout\vbox{%
447       \CROP@shiplist
448     }%
449   }%
450   \ifx\AtBegShi@Crop@ship\CROP@ship
451     \ifx\AtBegShi@Crop@shiplist\CROP@shiplist
452     \ifx\AtBegShi@Crop@@ship\CROP@@ship
453       \let\AtBegShi@found\relax
454       \ifx\shipout\AtBegShi@Crop@shipout
455         \def\AtBegShi@found{\shipout}%
456       \else\ifx\AtBegShi@OrgShipout\AtBegShi@Crop@shipout
457         \def\AtBegShi@found{\AtBegShi@OrgShipout}%
458       \else\ifx\@EveryShipout@Org@Shipout\AtBegShi@Crop@shipout
459         \def\AtBegShi@found{\@EveryShipout@Org@Shipout}%
460       \else\ifx\GPTorg@shipout\AtBegShi@Crop@shipout
461         \def\AtBegShi@found{\GPTorg@shipout}%
462       \else\ifx\THBorg@shipout\AtBegShi@Crop@shipout
463         \def\AtBegShi@found{\THBorg@shipout}%
464       \else\ifx\mem@oldshipout\AtBegShi@Crop@shipout
465         \def\AtBegShi@found{\mem@oldshipout}%
466       \fi\fi\fi\fi\fi\fi
467       \ifx\AtBegShi@found\relax
468       \else

```

```

469         \expandafter\endgroup
470         \expandafter\def\AtBegShi@found{%
471             \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
472             \afterassignment\CROP@ship
473             \setbox\AtBeginShipoutBox=
474         }%
475         \def\CROP@ship{%
476             \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
477             \else
478                 \expandafter\aftergroup
479             \fi
480             \CROP@@ship
481         }%
482         \def\CROP@shiplist{%
483             \lineskip\z@
484             \lineskiplimit\z@
485             \baselineskip\z@
486             \CROP@kernel
487             \box\AtBeginShipoutBox
488         }%
489         \def\CROP@@ship{%
490             \ifvoid\AtBeginShipoutBox
491             \else
492                 \setbox\AtBeginShipoutBox=\vbox{%
493                     \CROP@shiplist
494                 }%
495                 \expandafter\CROP@shipout
496                 \expandafter\box
497                 \expandafter\AtBeginShipoutBox
498             \fi
499         }%
500         \@PackageInfoNoLine{atbegshi}{Package ‘crop’ patched}%
501         \begingroup
502     \fi
503 \fi
504 \fi
505 \fi
506 \endgroup
507 \let\AtBegShi@PatchCrop\relax
508 }
509 \ifpackageloaded{crop}{%
510     \AtBegShi@PatchCrop
511 }{%
512     \AtBeginDocument{\AtBegShi@PatchCrop}%
513 }

```

### 3.5.2 Package everyshi

Fix of method. Use of box 255 is not changed.

```

514 \def\AtBegShi@PatchEveryshi{%
515     \begingroup
516     \long\def\AtBegShi@Everyshi@shipout{%
517         \afterassignment\@EveryShipout@Test
518         \global\setbox\@cclv=
519     }%
520     \long\def\AtBegShi@Everyshi@Test{%
521         \ifvoid\@cclv\relax
522             \aftergroup\@EveryShipout@Output
523         \else
524             \@EveryShipout@Output
525         \fi
526     }%

```

```

527 \ifx\AtBegShi@Everyshi@Test\@EveryShipout@Test
528 \let\AtBegShi@found\relax
529 \ifx\shipout\AtBegShi@Everyshi@shipout
530 \def\AtBegShi@found{\shipout}%
531 \else\ifx\AtBegShi@OrgShipout\AtBegShi@Everyshi@shipout
532 \def\AtBegShi@found{\AtBegShi@OrgShipout}%
533 \else\ifx\CROP@shipout\AtBegShi@Everyshi@shipout
534 \def\AtBegShi@found{\CROP@shipout}%
535 \else\ifx\GPTorg@shipout\AtBegShi@Everyshi@shipout
536 \def\AtBegShi@found{\GPTorg@shipout}%
537 \else\ifx\THBorg@shipout\AtBegShi@Everyshi@shipout
538 \def\AtBegShi@found{\THBorg@shipout}%
539 \else\ifx\mem@oldshipout\AtBegShi@Everyshi@shipout
540 \def\AtBegShi@found{\mem@oldshipout}%
541 \else
542 \expandafter\ifx\csname @EveryShipout@Org@Shipout\endcsname
543 \relax
544 \ifx\@EveryShipout@Shipout\AtBegShi@Everyshi@shipout
545 \def\AtBegShi@found{\@EveryShipout@Shipout}%
546 \fi
547 \fi
548 \fi\fi\fi\fi\fi\fi
549 \ifx\AtBegShi@found\relax
550 \else
551 \expandafter\endgroup
552 \expandafter\def\AtBegShi@found{%
553 \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
554 \afterassignment\@EveryShipout@Test
555 \setbox\AtBeginShipoutBox=%
556 }%
557 \def\@EveryShipout@Test{%
558 \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
559 \else
560 \expandafter\aftergroup
561 \fi
562 \AtBegShi@Everyshi@Output
563 }%
564 \def\AtBegShi@Everyshi@Output{%
565 \ifvoid\AtBeginShipoutBox
566 \else
567 \global\setbox\@cclv\box\AtBeginShipoutBox
568 \expandafter\@EveryShipout@Output
569 \fi
570 }%
571 \@PackageInfoNoLine{atbegshi}{Package 'everyshi' patched}%
572 \begingroup
573 \fi
574 \fi
575 \endgroup
576 \let\AtBegShi@PatchEveryshi\relax
577 }
578 \ifpackageloaded{everyshi}{%
579 \AtBegShi@PatchEveryshi
580 }{%
581 \AtBeginDocument{\AtBegShi@PatchEveryshi}%
582 }

```

### 3.5.3 Class memoir

Fix of method and box.

```

583 \def\AtBegShi@PatchMemoir{%
584 \begingroup

```

```

585 \def\AtBegShi@Memoir@shipout{%
586 \afterassignment\mem@shipi
587 \setbox\@cclv=%
588 }%
589 \def\AtBegShi@Memoir@shipi{%
590 \ifvoid\@cclv
591 \expandafter\aftergroup
592 \fi
593 \mem@shipii
594 }%
595 \def\AtBegShi@Memoir@shipiiA{%
596 \mem@oldshipout\vbox{%
597 \trimmarks
598 \unvbox\@cclv
599 }%
600 }%
601 \def\AtBegShi@Memoir@shipiiB{%
602 \ifvoid\@cclv
603 \mem@oldshipout\box\@cclv
604 \else
605 \mem@oldshipout\vbox{%
606 \trimmarks
607 \unvbox\@cclv
608 }%
609 \fi
610 }%
611 \ifx\AtBegShi@Memoir@shipi\mem@shipi
612 \ifcase\ifx\AtBegShi@Memoir@shipiiA\mem@shipii
613 \z@
614 \else
615 \ifx\AtBegShi@Memoir@shipiiB\mem@shipii
616 \z@
617 \else
618 \@ne
619 \fi
620 \fi
621 \let\AtBegShi@found\relax
622 \ifx\shipout\AtBegShi@Memoir@shipout
623 \def\AtBegShi@found{\shipout}%
624 \else\ifx\AtBegShi@OrgShipout\AtBegShi@Memoir@shipout
625 \def\AtBegShi@found{\AtBegShi@OrgShipout}%
626 \else\ifx\CROP@shipout\AtBegShi@Memoir@shipout
627 \def\AtBegShi@found{\CROP@shipout}%
628 \else\ifx\GPTorg@shipout\AtBegShi@Memoir@shipout
629 \def\AtBegShi@found{\GPTorg@shipout}%
630 \else\ifx\THBorg@shipout\AtBegShi@Memoir@shipout
631 \def\AtBegShi@found{\THBorg@shipout}%
632 \else\ifx\@EveryShipout@Org@Shipout\AtBegShi@Memoir@shipout
633 \def\AtBegShi@found{\@EveryShipout@Org@Shipout}%
634 \fi\fi\fi\fi\fi\fi
635 \ifx\AtBegShi@found\relax
636 \else
637 \expandafter\endgroup
638 \expandafter\def\AtBegShi@found{%
639 \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
640 \afterassignment\mem@shipi
641 \setbox\AtBeginShipoutBox=%
642 }%
643 \def\mem@shipi{%
644 \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
645 \else
646 \expandafter\aftergroup

```

```

647         \fi
648         \mem@shipii
649     }%
650     \def\mem@shipii{%
651         \ifvoid\AtBeginShipoutBox
652         \else
653             \setbox\AtBeginShipoutBox=\vbox{%
654                 \trimmarks
655                 \ifvbox\AtBeginShipoutBox
656                     \unvbox\AtBeginShipoutBox
657                 \else
658                     \box\AtBeginShipoutBox
659                 \fi
660             }%
661             \expandafter\mem@oldshipout
662             \expandafter\box
663             \expandafter\AtBeginShipoutBox
664         \fi
665     }%
666     \@PackageInfoNoLine{atbegshi}{Class 'memoir' patched}%
667     \begingroup
668     \fi
669     \fi
670     \fi
671 \endgroup
672 \let\AtBegShi@PatchMemoir\relax
673 }
674 \@ifclassloaded{memoir}{%
675     \AtBegShi@PatchMemoir
676 }{%
677     \AtBeginDocument{\AtBegShi@PatchMemoir}%
678 }
679 \AtBegShi@AtEnd
680 </package>

```

## 4 Test

### 4.1 Catcode checks for loading

```

681 <{*test1}>
682 \catcode'\{=1 %
683 \catcode'\}=2 %
684 \catcode'\#=6 %
685 \catcode'\@=11 %
686 \expandafter\ifx\csname count@\endcsname\relax
687     \countdef\count@=255 %
688 \fi
689 \expandafter\ifx\csname @gobble\endcsname\relax
690     \long\def\@gobble#1{%
691 \fi
692 \expandafter\ifx\csname @firstofone\endcsname\relax
693     \long\def\@firstofone#1{#1}%
694 \fi
695 \expandafter\ifx\csname loop\endcsname\relax
696     \expandafter\@firstofone
697 \else
698     \expandafter\@gobble
699 \fi
700 {%
701     \def\loop#1\repeat{%
702         \def\body{#1}%

```

```

703     \iterate
704 }%
705 \def\iterate{%
706     \body
707     \let\next\iterate
708     \else
709     \let\next\relax
710     \fi
711     \next
712 }%
713 \let\repeat=\fi
714 }%
715 \def\RestoreCatcodes{}
716 \count@=0 %
717 \loop
718     \edef\RestoreCatcodes{%
719         \RestoreCatcodes
720         \catcode\the\count@=\the\catcode\count@\relax
721     }%
722 \ifnum\count@<255 %
723     \advance\count@ 1 %
724 \repeat
725
726 \def\RangeCatcodeInvalid#1#2{%
727     \count@=#1\relax
728     \loop
729         \catcode\count@=15 %
730     \ifnum\count@<#2\relax
731         \advance\count@ 1 %
732     \repeat
733 }
734 \expandafter\ifx\csname LoadCommand\endcsname\relax
735     \def\LoadCommand{\input atbegshi.sty\relax}%
736 \fi
737 \def\Test{%
738     \RangeCatcodeInvalid{0}{47}%
739     \RangeCatcodeInvalid{58}{64}%
740     \RangeCatcodeInvalid{91}{96}%
741     \RangeCatcodeInvalid{123}{255}%
742     \catcode'\@=12 %
743     \catcode'\=0 %
744     \catcode'\{=1 %
745     \catcode'\}=2 %
746     \catcode'\#=6 %
747     \catcode'\[=12 %
748     \catcode'\]=12 %
749     \catcode'\%=14 %
750     \catcode'\ =10 %
751     \catcode13=5 %
752     \LoadCommand
753     \RestoreCatcodes
754 }
755 \Test
756 \csname @@end\endcsname
757 \end
758 </test1>
759 <*test2>
760 \input atbegshi.sty\relax
761 \def\msg#{\immediate\write16}
762 \msg{File: atbegshi-test2.tex 2008/07/31 v1.9 Test file for plain-TeX}
763 \def\testmsg#1#2{%
764     \msg{}%

```

```

765 \msg{*** Test with box (#1), expected page output [#2]}% hash-ok
766 }
767
768 \newbox\voidbox
769 \def\void{\box\voidbox}
770 \begingroup
771 \setbox\voidbox=\void
772 \endgroup
773
774 \count0=0\relax
775 \AtBeginShipout{%
776 \global\advance\count0 by 1\relax
777 \msg{* Inside \string\AtBeginShipout: [\the\count0]}%
778 }
779
780 \AtBeginShipoutFirst{%
781 \msg{* Inside \string\AtBeginShipoutFirst}%
782 Hello World%
783 }
784
785 \testmsg{\string\null}{1}
786 \shipout\null
787
788 \AtBeginShipoutFirst{%
789 This is too late%
790 }
791
792 \testmsg{void}{1}
793 \shipout\void
794
795 \testmsg{\string\copy255 (not void)}{2}
796 \setbox255\hbox{\vrule height 10bp width 10bp}
797 \shipout\copy255 %
798
799 \testmsg{\string\copy255 (again)}{3}
800 \shipout\copy255 %
801
802 \testmsg{\string\box255}{4}
803 \shipout\box255 %
804
805 \testmsg{\string\box255 (again)}{5}
806 \shipout\box255 %
807
808 \testmsg{\string\hbox}{6}
809 \shipout\hbox{\vrule height 5bp width 20bp}
810
811 \testmsg{\string\vbox}{7}
812 \shipout\vbox{\hrule height 20bp width 5bp}
813
814 \testmsg{\string\null, voided by hook}{8}
815 \def\VoidBox{%
816 \begingroup
817 \setbox\AtBeginShipoutBox=\box\AtBeginShipoutBox
818 \endgroup
819 }
820 \AtBeginShipout{\VoidBox}
821 \shipout\null
822 \def\VoidBox{}
823
824 \msg{*** \string\begingroup}
825 \begingroup
826 \testmsg{void}{9}%

```

```

827 \shipout\void
828 \msg{*** \string\endgroup}
829 \endgroup
830
831 \msg{*** \string\begingroup}
832 \begingroup
833 \testmsg{void}{}%
834 \shipout\void
835 \testmsg{\string\null}{8}%
836 \shipout\null
837 \msg{*** \string\endgroup}
838 \endgroup
839
840 \testmsg{output routine}{9}
841 Hello World
842 \vfill
843 \eject
844
845 \testmsg{\string\null\space(discarded)}{}
846 \AtBeginShipout{%
847 \msg{* Inside \string\AtBeginShipout: DISCARD}%
848 \AtBeginShipoutDiscard
849 }
850 \shipout\null
851
852 \end
853 \test2)

854 (*test3)
855 \NeedsTeXFormat{LaTeX2e}
856 \ProvidesFile{atbegshi-test3.tex}[2008/07/31 v1.9 Test file for LaTeX]
857 \RequirePackage{color}
858 \pagecolor{yellow}
859 \documentclass[a5paper,showtrims]{memoir}
860 \usepackage{atbegshi}
861 \AtBeginShipout{%
862 \setbox\AtBeginShipoutBox=\vbox{%
863 \vbox to Opt{%
864 \kern-1.5in %
865 \hbox to Opt{%
866 \kern-1.5in %
867 \color{blue}%
868 \rule{1in}{1in}%
869 \hss
870 }%
871 \vss
872 }%
873 \hrule
874 \hbox{\vrule\box\AtBeginShipoutBox\vrule}%
875 \hrule
876 }%
877 }
878 \usepackage{eso-pic}
879 \makeatletter
880 \@EveryShipout@Init
881 \let\@EveryShipout@Init\relax
882 \makeatother
883 \AddToShipoutPicture{%
884 \hspace{.52\paperwidth}%
885 \colorbox{cyan}{%
886 \rule{0mm}{\paperheight}%
887 \hspace{.48\paperwidth}%
888 }%

```



889 }

Newer versions of class `memoir` emulate package `crop` and prevents its loading. This is undone in next line for this test file.

```
890 \expandafter\let\csname ver@crop.sty\endcsname\relax
891 \usepackage[color=red,cross,a4,center]{crop}
892 \begin{document}
893 \shipout\null
894 \shipout\box\csname voidb@x\endcsname
895 \section{Hello World}
896 \end{document}
897 \end{test3}
```

## 5 Installation

### 5.1 Download

**Package.** This package is available on CTAN<sup>1</sup>:

[CTAN:macros/latex/contrib/oberdiek/atbegshi.dtx](#) The source file.

[CTAN:macros/latex/contrib/oberdiek/atbegshi.pdf](#) Documentation.

**Bundle.** All the packages of the bundle ‘oberdiek’ are also available in a TDS compliant ZIP archive. There the packages are already unpacked and the documentation files are generated. The files and directories obey the TDS standard.

[CTAN:install/macros/latex/contrib/oberdiek.tds.zip](#)

*TDS* refers to the standard “A Directory Structure for  $\text{\TeX}$  Files” ([CTAN:tds/tds.pdf](#)). Directories with `texmf` in their name are usually organized this way.

### 5.2 Bundle installation

**Unpacking.** Unpack the `oberdiek.tds.zip` in the TDS tree (also known as `texmf` tree) of your choice. Example (linux):

```
unzip oberdiek.tds.zip -d ~/texmf
```

**Script installation.** Check the directory `TDS:scripts/oberdiek/` for scripts that need further installation steps. Package `attachfile2` comes with the Perl script `pdfatfi.pl` that should be installed in such a way that it can be called as `pdfatfi`. Example (linux):

```
chmod +x scripts/oberdiek/pdfatfi.pl
cp scripts/oberdiek/pdfatfi.pl /usr/local/bin/
```

### 5.3 Package installation

**Unpacking.** The `.dtx` file is a self-extracting `docstrip` archive. The files are extracted by running the `.dtx` through plain- $\text{\TeX}$ :

```
tex atbegshi.dtx
```

**TDS.** Now the different files must be moved into the different directories in your installation TDS tree (also known as `texmf` tree):

---

<sup>1</sup><http://ftp.ctan.org/tex-archive/>

atbegshi.sty	→ tex/generic/oberdiek/atbegshi.sty
atbegshi.pdf	→ doc/latex/oberdiek/atbegshi.pdf
atbegshi-example1.tex	→ doc/latex/oberdiek/atbegshi-example1.tex
atbegshi-example2.tex	→ doc/latex/oberdiek/atbegshi-example2.tex
test/atbegshi-test1.tex	→ doc/latex/oberdiek/test/atbegshi-test1.tex
test/atbegshi-test2.tex	→ doc/latex/oberdiek/test/atbegshi-test2.tex
test/atbegshi-test3.tex	→ doc/latex/oberdiek/test/atbegshi-test3.tex
atbegshi.dtx	→ source/latex/oberdiek/atbegshi.dtx

If you have a `docstrip.cfg` that configures and enables `docstrip`'s TDS installing feature, then some files can already be in the right place, see the documentation of `docstrip`.

## 5.4 Refresh file name databases

If your  $\text{\TeX}$  distribution (te $\text{\TeX}$ , mi $\text{\TeX}$ , ...) relies on file name databases, you must refresh these. For example, te $\text{\TeX}$  users run `texhash` or `mktextlsr`.

## 5.5 Some details for the interested

**Attached source.** The PDF documentation on CTAN also includes the `.dtx` source file. It can be extracted by AcrobatReader 6 or higher. Another option is `pdftk`, e.g. unpack the file into the current directory:

```
pdftk atbegshi.pdf unpack_files output .
```

**Unpacking with  $\text{\LaTeX}$ .** The `.dtx` chooses its action depending on the format:

**plain- $\text{\TeX}$ :** Run `docstrip` and extract the files.

**$\text{\LaTeX}$ :** Generate the documentation.

If you insist on using  $\text{\LaTeX}$  for `docstrip` (really, `docstrip` does not need  $\text{\LaTeX}$ ), then inform the autodetect routine about your intention:

```
latex \let\install=y\input{atbegshi.dtx}
```

Do not forget to quote the argument according to the demands of your shell.

**Generating the documentation.** You can use both the `.dtx` or the `.drv` to generate the documentation. The process can be configured by the configuration file `ltxdoc.cfg`. For instance, put this line into this file, if you want to have A4 as paper format:

```
\PassOptionsToClass{a4paper}{article}
```

An example follows how to generate the documentation with pdf $\text{\LaTeX}$ :

```
pdflatex atbegshi.dtx
makeindex -s gind.ist atbegshi.idx
pdflatex atbegshi.dtx
makeindex -s gind.ist atbegshi.idx
pdflatex atbegshi.dtx
```

# 6 History

[2007/04/17 v1.0]

- First version.

## [2007/04/18 v1.1]

- New method based on `\lastkern` is used if  $\varepsilon$ -TeX is missing.
- `\AtBeginShipoutDiscard` also resets `\deadcycles`.

## [2007/04/19 v1.2]

- `\AtBeginShipoutEarly` removed for simplification reasons.
- Forgotten definition of `\AtBegShi@Info` added.
- Patches for packages `crop` and `everyshi` and class `memoir` added.

## [2007/04/26 v1.3]

- Use of package `infwarerr`.
- Catcode section after generic header.

## [2007/04/27 v1.4]

- Small optimizations.

## [2007/06/06 v1.5]

- `\AtBeginShipoutUpperLeft` added.
- Example added.
- Fix in second test file for newer version of `memoir`.

## [2007/09/09 v1.6]

- Catcode section rewritten.

## [2008/07/18 v1.7]

- Documentation of `\AtBeginShipoutUpperLeft` fixed and extended.

## [2008/07/19 v1.8]

- `\AtBeginShipoutUpperLeftForeground` added.

## [2008/07/31 v1.9]

- Second example (`TrimBox` for `dvipdfmx`) added.
- No changes in package code.

## 7 Index

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

Symbols		<code>\@EveryShipout@Org@Shipout</code> . . . . .	
<code>\#</code> . . . . .	684, 746		458, 459, 632, 633
<code>\%</code> . . . . .	749	<code>\@EveryShipout@Output</code> .	522, 524, 568
<code>\@</code> . . . . .	685, 742	<code>\@EveryShipout@Shipout</code> . . . .	544, 545
<code>\@EveryShipout@Init</code> . . . . .	880, 881	<code>\@EveryShipout@Test</code>	517, 527, 554, 557

\@PackageInfoNoLine	224, 500, 571, 666	\AtBegShi@Everyshi@Output	562, 564
\@PackageWarning	215, 280, 304	\AtBegShi@Everyshi@shipout	516, 529, 531, 533, 535, 537, 539, 544
\@cclv	430, 433, 443, 518, 521, 567, 587, 590, 598, 602, 603, 607	\AtBegShi@Everyshi@Test	520, 527
\@empty	241, 242, 260	\AtBegShi@First	231, 239
\@firstofone	693, 696	\AtBegShi@FirstDisabled	261, 279
\@gobble	690, 698	\AtBegShi@found	453, 455, 457, 459, 461, 463, 465, 467, 470, 528, 530, 532, 534, 536, 538, 540, 545, 549, 552, 621, 623, 625, 627, 629, 631, 633, 635, 638
\@ifclassloaded	674	\AtBegShi@GroupLevel	199, 206, 471, 476, 553, 558, 639, 644
\@ifdefinable	178	\AtBegShi@Hook	220, 264, 269
\@ifpackageloaded	509, 578	\AtBegShi@HookFirst	242, 246, 266, 277
\@one	161, 164, 618	\AtBegShi@HookNext	221, 222, 265, 273
\@undefined	107, 163	\AtBegShi@horigin	333, 336, 383, 402
\[	747	\AtBegShi@Item	297, 299
\	743	\AtBegShi@Memoir@shipi	589, 611
\{	682, 744	\AtBegShi@Memoir@shipiiA	595, 612
\}	683, 745	\AtBegShi@Memoir@shipiiB	601, 615
\]	748	\AtBegShi@Memoir@shipout	585, 622, 624, 626, 628, 630, 632
\_	750	\AtBegShi@OrgProtect	217, 229, 232
<b>A</b>			
\AddToShipoutPicture	883	\AtBegShi@OrgShipout	233, 315, 316, 456, 457, 531, 532, 624, 625
\advance	723, 731, 776	\AtBegShi@Output	209, 211
\afterassignment	200, 429, 472, 517, 554, 586, 640	\AtBegShi@PatchCrop	426, 507, 510, 512
\aftergroup	81, 207, 434, 478, 522, 560, 591, 646	\AtBegShi@PatchEveryshi	514, 576, 579, 581
\AtBeginDocument	324, 512, 581, 677	\AtBegShi@PatchMemoir	583, 672, 675, 677
\AtBeginShipout	2, 6, 44, 267, 775, 777, 820, 846, 847, 861	\AtBegShi@Shipout	195, 317
\AtBeginShipoutBox	45, 47, 197, 202, 214, 227, 233, 252, 257, 303, 314, 381, 386, 393, 397, 399, 405, 473, 487, 490, 492, 497, 555, 565, 567, 641, 651, 653, 655, 656, 658, 663, 817, 862, 874	\AtBegShi@Test	200, 204
\AtBeginShipoutDiscard	3, 29, 182, 308, 848	\AtBegShi@vorigin	334, 337, 385, 404
\AtBeginShipoutFirst	3, 261, 275, 282, 780, 781, 788	<b>B</b>	
\AtBeginShipoutInit	3, 312, 322, 324	\baselineskip	253, 441, 485
\AtBeginShipoutNext	2, 14, 28, 271	\begin	11, 50, 892
\AtBeginShipoutUpperLeft	3, 7, 15, 380	\body	702, 706
\AtBeginShipoutUpperLeftForeground	3, 396	\box	47, 227, 233, 393, 443, 487, 496, 567, 603, 658, 662, 769, 802, 803, 805, 806, 817, 874, 894
\AtBegShi@AbortIfUndefined	415, 422, 423, 424, 425	<b>C</b>	
\AtBegShi@AddHook	269, 273, 296	\catcode	58, 59, 60, 61, 62, 63, 64, 72, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 119, 120, 123, 124, 125, 126, 130, 131, 132, 133, 137, 139, 189, 190, 192, 193, 237, 238, 682, 683, 684, 685, 720, 729, 742, 743, 744, 745, 746, 747, 748, 749, 750, 751
\AtBegShi@AddTo	277, 285, 297	\circle	8
\AtBegShi@AtEnd	135, 136, 418, 679	\color	16, 867
\AtBegShi@BeginPicture	351, 367, 387, 406	\colorbox	885
\AtBegShi@CheckDefinable	157, 182, 267, 271, 275, 312, 315	\copy	256, 257, 377, 399, 795, 797, 799, 800
\AtBegShi@Crop@ship	445, 452	\count	774, 776, 777
\AtBegShi@Crop@ship	432, 450	\count@	687, 716, 720, 722, 723, 727, 729, 730, 731
\AtBegShi@Crop@shiplist	438, 451	\countdef	687
\AtBegShi@Crop@shipout	428, 454, 456, 458, 460, 462, 464	\CROP@@ship	436, 452, 480, 489
\AtBegShi@Discardedfalse	219, 225		
\AtBegShi@Discardedtrue	185		
\AtBegShi@EndPicture	361, 372, 389, 408		

<code>\CROP@kernel</code> . . . . .	442, 486	539, 542, 544, 549, 611, 612,
<code>\CROP@ship</code> . . . . .	429, 450, 472, 475	615, 622, 624, 626, 628, 630,
<code>\CROP@shiplist</code> . . . .	447, 451, 482, 493	632, 635, 686, 689, 692, 695, 734
<code>\CROP@shipout</code> . . . . .	446, 495, 533, 534, 626, 627	<code>\ignorespaces</code> . . . . .
<code>\csname</code> 65, 73, 99, 115, 122, 152, 158,		<code>\immediate</code> . . . . .
188, 218, 286, 314, 321, 327,		<code>\input</code> . . . . .
341, 344, 355, 417, 542, 686,		<code>\iterate</code> . . . . .
689, 692, 695, 734, 756, 890, 894		
<code>\currentgrouplevel</code> . . . . .	199,	<b>K</b>
206, 471, 476, 553, 558, 639, 644		<code>\kern</code> . . . . .
		198, 383,
		385, 386, 402, 404, 405, 864, 866
<b>D</b>		
<code>\deadcycles</code> . . . . .	184	<b>L</b>
<code>\documentclass</code> . . . . .	2, 37, 859	<code>\lastkern</code> . . . . .
<code>\dp</code> . . . . .	251, 376	<code>\line</code> . . . . .
		<code>\lineskip</code> . . . . .
<b>E</b>		<code>\lineskiplimit</code> . . . . .
<code>\E</code> . . . . .	238	<code>\LoadCommand</code> . . . . .
<code>\eject</code> . . . . .	843	<code>\loop</code> . . . . .
<code>\empty</code> . . . . .	68, 69	
<code>\end</code> . . . . .	34, 54, 757, 852, 896	<b>M</b>
<code>\endcsname</code> . . . . .		<code>\makeatletter</code> . . . . .
65, 73, 99, 115, 122, 152, 158,		<code>\makeatother</code> . . . . .
188, 218, 286, 314, 321, 327,		<code>\mem@oldshipout</code> . . . . .
341, 344, 355, 417, 542, 686,		464,
689, 692, 695, 734, 756, 890, 894		465, 539, 540, 596, 603, 605, 661
<code>\endinput</code> . . . . .	81, 419	<code>\mem@shipi</code> . . . . .
<code>\endpicture</code> . . . . .	362	586, 611, 640, 643
<code>\errmessage</code> . . . . .	169	<code>\mem@shipii</code> . . . .
		593, 612, 615, 648, 650
		<code>\MessageBreak</code> . . . . .
		281, 305
		<code>\msg</code> . . . . .
		761, 762, 764, 765,
		777, 781, 824, 828, 831, 837, 847
<b>F</b>		
<code>\fill</code> . . . . .	25	<b>N</b>
		<code>\NeedsTeXFormat</code> . . . . .
<b>G</b>		855
<code>\g@addto@macro</code> . . . . .	294	<code>\newbox</code> . . . . .
<code>\gdef</code> . . . . .	222, 264, 265, 266, 318	768
<code>\GPTorg@shipout</code> . . . . .		<code>\newif</code> . . . . .
460, 461, 535, 536, 628, 629		181
		<code>\newpage</code> . . . . .
<b>H</b>		13, 22, 27, 32, 52
<code>\hbox</code> . . . . .	45, 197, 368,	<code>\next</code> . . . . .
381, 397, 796, 808, 809, 865, 874		707, 709, 711
<code>\hrule</code> . . . . .	812, 873, 875	<code>\null</code> . . . . .
<code>\hspace</code> . . . . .	884, 887	785, 786,
<code>\hss</code> . . . . .	412, 869	814, 821, 835, 836, 845, 850, 893
<code>\ht</code> . . . . .	250, 375, 386, 405	<code>\number</code> . . . . .
		199, 471, 553, 639
<b>I</b>		
<code>\ifAtBegShi@Discarded</code> .	181, 223, 300	<b>P</b>
<code>\ifcase</code> . . . . .	160, 340, 612	<code>\p@</code> . . . . .
<code>\ifdim</code> . . . . .	205	<code>\PackageInfo</code> . . . . .
<code>\ifnum</code> . . . .	206, 476, 558, 644, 722, 730	<code>\pagecolor</code> . . . . .
<code>\ifpdf</code> . . . . .	332	858
<code>\ifvbox</code> . . . . .	655	<code>\paperheight</code> . . . . .
<code>\ifvoid</code> . . . . .	214, 303,	8, 17, 18, 886
433, 490, 521, 565, 590, 602, 651		<code>\paperwidth</code> . . . . .
<code>\ifx</code> . . . . .	66, 69, 73, 99, 107,	8, 17, 18, 884, 887
110, 152, 158, 160, 163, 188,		<code>\par</code> . . . . .
242, 286, 321, 327, 341, 344,		24
355, 417, 450, 451, 452, 454,		<code>\pdfhorigin</code> . . . . .
456, 458, 460, 462, 464, 467,		333
527, 529, 531, 533, 535, 537,		<code>\pdfvorigin</code> . . . . .
		334
		<code>\picture</code> . . . . .
		353
		<code>\protect</code> . . . . .
		217, 229, 232
		<code>\ProvidesFile</code> . . . . .
		856
		<code>\ProvidesPackage</code> . . . . .
		70, 116
		<code>\put</code> . . . . .
		8, 17, 18
		<b>R</b>
		<code>\RangeCatcodeInvalid</code> . . . . .
		726, 738, 739, 740, 741
		<code>\repeat</code> . . . . .
		701, 713, 724, 732
		<code>\RequirePackage</code> . . . . .
		155, 330, 857
		<code>\RestoreCatcodes</code> . .
		715, 718, 719, 753
		<code>\rlap</code> . . . . .
		382, 398, 401

<code>\rule</code> .....	868, 886	<code>\unvbox</code> .....	598, 607, 656
		<code>\usepackage</code>	3, 4, 5, 38, 39, 860, 878, 891
<b>S</b>			
<code>\section</code> .....	12, 895	<b>V</b>	
<code>\setbox</code> .....	45, 197, 202,	<code>\vbox</code> ..	244, 252, 384, 403, 446, 492,
	227, 244, 252, 368, 381, 397,		596, 605, 653, 811, 812, 862, 863
	430, 473, 492, 518, 555, 567,	<code>\vfill</code> .....	842
	587, 641, 653, 771, 796, 817, 862	<code>\void</code> .....	769, 771, 793, 827, 834
<code>\shipout</code> .....	316, 317, 454, 455,	<code>\VoidBox</code> .....	815, 820, 822
	529, 530, 622, 623, 786, 793,	<code>\voidbox</code> .....	768, 769, 771
	797, 800, 803, 806, 809, 812,	<code>\vrule</code> .....	796, 809, 874
	821, 827, 834, 836, 850, 893, 894	<code>\vspace</code> .....	25
<code>\space</code> .....	170, 845	<code>\vss</code> .....	390, 409, 871
<code>\special</code> .....	46		
<b>T</b>			
<code>\Test</code> .....	737, 755	<b>W</b>	
<code>\testmsg</code> .....	763, 785,	<code>\wd</code> .....	249, 397
	792, 795, 799, 802, 805, 808,	<code>\write</code> .....	75, 101, 761
	811, 814, 826, 833, 835, 840, 845		
<code>\THBorg@shipout</code> .....		<b>X</b>	
	462, 463, 537, 538, 630, 631	<code>\X</code> .....	237
<code>\the</code>	123, 124, 125, 126, 137, 290, 720, 777	<code>\x</code> .....	65, 66, 69,
<code>\TMP@EnsureCode</code>	134, 141, 142, 143,		74, 78, 80, 100, 105, 115, 121, 129
	144, 145, 146, 147, 148, 149, 150		
<code>\toks</code> .....	289, 290	<b>Z</b>	
<code>\trimmarks</code> .....	597, 606, 654	<code>\z@</code> ...	166, 184, 205, 244, 249, 250,
			251, 255, 256, 289, 290, 439,
			440, 441, 483, 484, 485, 613, 616
<b>U</b>			
<code>\unitlength</code> .....	357	<code>\z@skip</code> .....	253, 254