

# 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	16
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]	26
[2007/04/19 v1.2]	26
[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 $\varepsilon$ -T<sub>E</sub>X, traditional way

And for T<sub>E</sub>X without  $\varepsilon$ -T<sub>E</sub>X:

```

\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 T<sub>E</sub>X without  $\varepsilon$ -T<sub>E</sub>X 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 \expandafter\let\expandafter\x\csname ver@atbegshi.sty\endcsname
64 \ifcase 0%
65 \ifx\x\relax % plain
66 \else
67 \ifx\x\empty % LaTeX
68 \else
69 1%
70 \fi
71 \fi
72 \else
73 \catcode35 6 % #
74 \catcode123 1 % {
75 \catcode125 2 % }
76 \expandafter\ifx\csname PackageInfo\endcsname\relax
77 \def\x#1#2{%
78 \immediate\write-1{Package #1 Info: #2.}%
79 }%
80 \else
81 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
82 \fi
83 \x{atbegshi}{The package is already loaded}%
84 \endgroup
85 \expandafter\endinput
86 \fi
87 \endgroup

```

Package identification:

```

88 \begingroup
89 \catcode35 6 % #
90 \catcode40 12 % (
91 \catcode41 12 % )
92 \catcode44 12 % ,
93 \catcode45 12 % -

```

```

94 \catcode46 12 % .
95 \catcode47 12 % /
96 \catcode58 12 % :
97 \catcode64 11 % @
98 \catcode123 1 % {
99 \catcode125 2 % }
100 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
101 \def\x#1#2#3[#4]{\endgroup
102 \immediate\write-1{Package: #3 #4}%
103 \xdef#1{#4}%
104 }%
105 \else
106 \def\x#1#2[#3]{\endgroup
107 #2[#3]}%
108 \ifx#1\relax
109 \xdef#1{#3}%
110 \fi
111 }%
112 \fi
113 \expandafter\x\csname ver@atbegshi.sty\endcsname
114 ProvidesPackage{atbegshi}%
115 [2008/07/31 v1.9 At begin shipout hook (H0)]

```

### 3.2 Catcodes

```

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

```

### 3.3 Preparations

```

149 \begingroup\expandafter\expandafter\expandafter\endgroup
150 \expandafter\ifx\csname RequirePackage\endcsname\relax

```

```

151 \input infwarerr.sty\relax
152 \else
153 \RequirePackage{infwarerr}[2007/09/09]%
154 \fi

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

179 \newif\ifAtBegShi@Discarded

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

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

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

\AtBegShi@Test
202 \def\AtBegShi@Test{%

```

```

203 X \ifdim\lastkern=\z@
204 E \ifnum\AtBegShi@GroupLevel<\currentgrouplevel
205     \expandafter\aftergroup
206     \fi
207     \AtBegShi@Output
208 }

```

\AtBegShi@Output

```

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

235 \catcode'\X=11 %
236 \catcode'\E=11 %

```

\AtBegShi@First

```

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

```

```

260 \endgroup
261 }

\AtBegShi@Hook
262 \gdef\AtBegShi@Hook{}

\AtBegShi@HookNext
263 \gdef\AtBegShi@HookNext{}

\AtBegShi@HookFirst
264 \gdef\AtBegShi@HookFirst{}

\AtBeginShipout
265 \AtBegShi@CheckDefinable\AtBeginShipout
266 \def\AtBeginShipout{%
267   \AtBegShi@AddHook\AtBegShi@Hook
268 }

\AtBeginShipoutNext
269 \AtBegShi@CheckDefinable\AtBeginShipoutNext
270 \def\AtBeginShipoutNext{%
271   \AtBegShi@AddHook\AtBegShi@HookNext
272 }

\AtBeginShipoutFirst
273 \AtBegShi@CheckDefinable\AtBeginShipoutFirst
274 \def\AtBeginShipoutFirst{%
275   \AtBegShi@AddTo\AtBegShi@HookFirst
276 }

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

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

\AtBegShi@AddHook
294 \long\def\AtBegShi@AddHook#1#2{%
295   \AtBegShi@AddTo#1{\AtBegShi@Item{#2}}%
296 }

\AtBegShi@Item
297 \long\def\AtBegShi@Item#1{%
298   \ifAtBegShi@Discarded
299   \else

```

```

300     #1%
301     \ifvoid\AtBeginShipoutBox
302         \@PackageWarning{atbegshi}{%
303             Shipout box was voided by hook,\MessageBreak
304             ignoring shipout box%
305         }%
306     \AtBeginShipoutDiscard
307 \fi
308 \fi
309 }

```

\AtBeginShipoutInit

```

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

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

```

### 3.4 Positioning

```

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

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

337 \begingroup
338 \ifcase
339     \expandafter\ifx\csname picture\endcsname\relax
340         1%
341     \else
342         \expandafter\ifx\csname endpicture\endcsname\relax
343             1%
344         \else
345             0%
346         \fi
347     \fi
348 \endgroup
349 \def\AtBegShi@BeginPicture{%
350     \begingroup
351     \picture(0,0)\relax
352     \begingroup\expandafter\expandafter\expandafter\endgroup
353     \expandafter\ifx\csname unitlength\endcsname\relax
354     \else
355         \unitlength=1pt\relax

```

```

356     \fi
357     \ignorespaces
358 }%
359 \def\AtBegShi@EndPicture{%
360     \endpicture
361     \endgroup
362 }%
363 \else
364     \endgroup
365     \def\AtBegShi@BeginPicture{%
366         \setbox0=\hbox\bgroup
367         \begingroup
368         \ignorespaces
369     }%
370     \def\AtBegShi@EndPicture{%
371         \endgroup
372         \egroup
373         \ht0=0pt\relax
374         \dp0=0pt\relax
375         \copy0 %
376     }%
377 \fi

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

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

```

### 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.



```

413 \def\AtBegShi@AbortIfUndefined#1{%
414   \begingroup\expandafter\expandafter\expandafter\endgroup
415   \expandafter\ifx\csname#1\endcsname\relax
416     \AtBegShi@AtEnd
417   \expandafter\endinput
418   \fi
419 }
420 \AtBegShi@AbortIfUndefined{currentgrouplevel}
421 \AtBegShi@AbortIfUndefined{AtBeginDocument}
422 \AtBegShi@AbortIfUndefined{@ifpackageloaded}
423 \AtBegShi@AbortIfUndefined{@ifclassloaded}

```

### 3.5.1 Package crop

Fix of method and box.

```

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

```

```

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

```

### 3.5.2 Package everyshi

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

```

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

```

```

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

```

### 3.5.3 Class memoir

Fix of method and box.

```

581 \def\AtBegShi@PatchMemoir{%
582 \begingroup
583 \def\AtBegShi@Memoir@shipout{%
584 \afterassignment\mem@shipi
585 \setbox\@cclv=%
586 }%

```

```

587 \def\AtBegShi@Memoir@shipi{%
588 \ifvoid\@cclv
589 \expandafter\aftergroup
590 \fi
591 \mem@shipii
592 }%
593 \def\AtBegShi@Memoir@shipiiA{%
594 \mem@oldshipout\vbox{%
595 \trimmarks
596 \unvbox\@cclv
597 }%
598 }%
599 \def\AtBegShi@Memoir@shipiiB{%
600 \ifvoid\@cclv
601 \mem@oldshipout\box\@cclv
602 \else
603 \mem@oldshipout\vbox{%
604 \trimmarks
605 \unvbox\@cclv
606 }%
607 \fi
608 }%
609 \ifx\AtBegShi@Memoir@shipi\mem@shipi
610 \ifcase\ifx\AtBegShi@Memoir@shipiiA\mem@shipii
611 \z@
612 \else
613 \ifx\AtBegShi@Memoir@shipiiB\mem@shipii
614 \z@
615 \else
616 \@ne
617 \fi
618 \fi
619 \let\AtBegShi@found\relax
620 \ifx\shipout\AtBegShi@Memoir@shipout
621 \def\AtBegShi@found{\shipout}%
622 \else\ifx\AtBegShi@OrgShipout\AtBegShi@Memoir@shipout
623 \def\AtBegShi@found{\AtBegShi@OrgShipout}%
624 \else\ifx\CROP@shipout\AtBegShi@Memoir@shipout
625 \def\AtBegShi@found{\CROP@shipout}%
626 \else\ifx\GPTorg@shipout\AtBegShi@Memoir@shipout
627 \def\AtBegShi@found{\GPTorg@shipout}%
628 \else\ifx\THBorg@shipout\AtBegShi@Memoir@shipout
629 \def\AtBegShi@found{\THBorg@shipout}%
630 \else\ifx\@EveryShipout@Org@Shipout\AtBegShi@Memoir@shipout
631 \def\AtBegShi@found{\@EveryShipout@Org@Shipout}%
632 \fi\fi\fi\fi\fi\fi
633 \ifx\AtBegShi@found\relax
634 \else
635 \expandafter\endgroup
636 \expandafter\def\AtBegShi@found{%
637 \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
638 \afterassignment\mem@shipi
639 \setbox\AtBeginShipoutBox=%
640 }%
641 \def\mem@shipi{%
642 \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
643 \else
644 \expandafter\aftergroup
645 \fi
646 \mem@shipii
647 }%
648 \def\mem@shipii{%

```

```

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

```

## 4 Test

### 4.1 Catcode checks for loading

```

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

```

```

705     \let\next\iterate
706   \else
707     \let\next\relax
708   \fi
709   \next
710 }%
711 \let\repeat=\fi
712 }%
713 \def\RestoreCatcodes{}
714 \count@=0 %
715 \loop
716   \edef\RestoreCatcodes{%
717     \RestoreCatcodes
718     \catcode\the\count@=\the\catcode\count@\relax
719   }%
720 \ifnum\count@<255 %
721   \advance\count@ 1 %
722 \repeat
723
724 \def\RangeCatcodeInvalid#1#2{%
725   \count@=#1\relax
726   \loop
727     \catcode\count@=15 %
728   \ifnum\count@<#2\relax
729     \advance\count@ 1 %
730   \repeat
731 }
732 \expandafter\ifx\csname LoadCommand\endcsname\relax
733   \def\LoadCommand{\input atbegshi.sty\relax}%
734 \fi
735 \def\Test{%
736   \RangeCatcodeInvalid{0}{47}%
737   \RangeCatcodeInvalid{58}{64}%
738   \RangeCatcodeInvalid{91}{96}%
739   \RangeCatcodeInvalid{123}{255}%
740   \catcode'\@=12 %
741   \catcode'\=0 %
742   \catcode'\{=1 %
743   \catcode'\}=2 %
744   \catcode'\#=6 %
745   \catcode'\[=12 %
746   \catcode'\]=12 %
747   \catcode'\%=14 %
748   \catcode'\ =10 %
749   \catcode13=5 %
750   \LoadCommand
751   \RestoreCatcodes
752 }
753 \Test
754 \csname @@end\endcsname
755 \end
756 </test1>
757 <*test2>
758 \input atbegshi.sty\relax
759 \def\msg#\immediate\write16{
760 \msg{File: atbegshi-test2.tex 2008/07/31 v1.9 Test file for plain-TeX}
761 \def\testmsg#1#2{%
762   \msg{}%
763   \msg{*** Test with box (#1), expected page output [#2]}% hash-ok
764 }
765
766 \newbox\voidbox

```

```

767 \def\void{\box\voidbox}
768 \begingroup
769   \setbox\voidbox=\void
770 \endgroup
771
772 \count0=0\relax
773 \AtBeginShipout{%
774   \global\advance\count0 by 1\relax
775   \msg{* Inside \string\AtBeginShipout: [\the\count0]]}%
776 }
777
778 \AtBeginShipoutFirst{%
779   \msg{* Inside \string\AtBeginShipoutFirst}%
780   Hello World%
781 }
782
783 \testmsg{\string\null}{1}
784 \shipout\null
785
786 \AtBeginShipoutFirst{%
787   This is too late%
788 }
789
790 \testmsg{void}{}
791 \shipout\void
792
793 \testmsg{\string\copy255 (not void)}{2}
794 \setbox255\hbox{\vrule height 10bp width 10bp}
795 \shipout\copy255 %
796
797 \testmsg{\string\copy255 (again)}{3}
798 \shipout\copy255 %
799
800 \testmsg{\string\box255}{4}
801 \shipout\box255 %
802
803 \testmsg{\string\box255 (again)}{}
804 \shipout\box255 %
805
806 \testmsg{\string\hbox}{5}
807 \shipout\hbox{\vrule height 5bp width 20bp}
808
809 \testmsg{\string\vbox}{6}
810 \shipout\vbox{\hrule height 20bp width 5bp}
811
812 \testmsg{\string\null, voided by hook}{}
813 \def\VoidBox{%
814   \begingroup
815     \setbox\AtBeginShipoutBox=\box\AtBeginShipoutBox
816   \endgroup
817 }
818 \AtBeginShipout{\VoidBox}
819 \shipout\null
820 \def\VoidBox{}
821
822 \msg{*** \string\begingroup}
823 \begingroup
824   \testmsg{void}{}%
825   \shipout\void
826 \msg{*** \string\endgroup}
827 \endgroup
828

```

```

829 \msg{*** \string\beginpgroup}
830 \beginpgroup
831   \testmsg{void}{}%
832   \shipout\void
833   \testmsg{\string\null}{8}%
834   \shipout\null
835 \msg{*** \string\endpgroup}
836 \endpgroup
837
838 \testmsg{output routine}{9}
839 Hello World
840 \vfill
841 \eject
842
843 \testmsg{\string\null\space(discarded)}{-}
844 \AtBeginShipout{%
845   \msg{* Inside \string\AtBeginShipout: DISCARD}%
846   \AtBeginShipoutDiscard
847 }
848 \shipout\null
849
850 \end
851 /test2)
852 (*test3)
853 \NeedsTeXFormat{LaTeX2e}
854 \ProvidesFile{atbegshi-test3.tex}[2008/07/31 v1.9 Test file for LaTeX]
855 \RequirePackage{color}
856 \pagecolor{yellow}
857 \documentclass[a5paper,showtrims]{memoir}
858 \usepackage{atbegshi}
859 \AtBeginShipout{%
860   \setbox\AtBeginShipoutBox=\vbox{%
861     \vbox to 0pt{%
862       \kern-1.5in %
863       \hbox to 0pt{%
864         \kern-1.5in %
865         \color{blue}%
866         \rule{1in}{1in}%
867         \hss
868       }%
869       \vss
870     }%
871     \hrule
872     \hbox{\vrule\box\AtBeginShipoutBox\vrule}%
873     \hrule
874   }%
875 }
876 \usepackage{eso-pic}
877 \makeatletter
878 \@EveryShipout@Init
879 \let\@EveryShipout@Init\relax
880 \makeatother
881 \AddToShipoutPicture{%
882   \hspace{.52\paperwidth}%
883   \colorbox{cyan}{%
884     \rule{0mm}{\paperheight}%
885     \hspace{.48\paperwidth}%
886   }%
887 }

```

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

```

888 \expandafter\let\csname ver@crop.sty\endcsname\relax

```



```

889 \usepackage[color=red,cross,a4,center]{crop}
890 \begin{document}
891 \shipout\null
892 \shipout\box\csname voidb@x\endcsname
893 \section{Hello World}
894 \end{document}
895 \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 T<sub>E</sub>X 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-T<sub>E</sub>X:

```
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):

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

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.

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

## 5.4 Refresh file name databases

If your  $\text{\TeX}$  distribution (te $\text{\TeX}$ , mik $\text{\TeX}$ , ...) 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  $\epsilon$ - $\text{\TeX}$  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>\#</code> .....	682, 744	<code>\@ifpackageloaded</code> ..... 507, 576
<code>\%</code> .....	747	<code>\@one</code> ..... 159, 162, 616
<code>\@</code> .....	683, 740	<code>\@undefined</code> ..... 161
<code>\@EveryShipout@Init</code> .....	878, 879	<code>\[</code> ..... 745
<code>\@EveryShipout@Org@Shipout</code> .....		<code>\]</code> ..... 741
.....	456, 457, 630, 631	<code>\{</code> ..... 680, 742
<code>\@EveryShipout@Output</code> .	520, 522, 566	<code>\}</code> ..... 681, 743
<code>\@EveryShipout@Shipout</code> ....	542, 543	<code>\]</code> ..... 746
<code>\@EveryShipout@Test</code> .....	515, 525, 552, 555	
<code>\@PackageInfoNoLine</code> .....	222, 498, 569, 664	<code>\_</code> ..... 748
<code>\@PackageWarning</code> .....	213, 278, 302	
<code>\@cclv</code> ....	428, 431, 441, 516, 519,	
.....	565, 585, 588, 596, 600, 601, 605	<b>A</b>
<code>\@empty</code> .....	239, 240, 258	<code>\AddToShipoutPicture</code> ..... 881
<code>\@firstofone</code> .....	691, 694	<code>\advance</code> ..... 721, 729, 774
<code>\@gobble</code> .....	688, 696	<code>\afterassignment</code> .....
<code>\@ifclassloaded</code> .....	672	..... 198, 427, 470, 515, 552, 584, 638
<code>\@ifdefinable</code> .....	176	<code>\aftergroup</code> .....
		..... 205, 432, 476, 520, 558, 589, 644

<code>\AtBeginDocument</code> . . .	322, 510, 579, 675	<code>\AtBegShi@PatchCrop</code>	424, 505, 508, 510
<code>\AtBeginShipout</code> . . . . .	2, 6, 44, 265, 773, 775, 818, 844, 845, 859	<code>\AtBegShi@PatchEveryshi</code> . . . . .	512, 574, 577, 579
<code>\AtBeginShipoutBox</code> 45, 47, 195, 200, 212, 225, 231, 250, 255, 301, 312, 379, 384, 391, 395, 397, 403, 471, 485, 488, 490, 495, 553, 563, 565, 639, 649, 651, 653, 654, 656, 661, 815, 860, 872		<code>\AtBegShi@PatchMemoir</code> . . . . .	581, 670, 673, 675
<code>\AtBeginShipoutDiscard</code> . . . . .	3, 29, 180, 306, 846	<code>\AtBegShi@Shipout</code> . . . . .	193, 315
<code>\AtBeginShipoutFirst</code> . . . . .	3, 259, 273, 280, 778, 779, 786	<code>\AtBegShi@Test</code> . . . . .	198, 202
<code>\AtBeginShipoutInit</code> . . .	3, 310, 320, 322	<code>\AtBegShi@vorigin</code> . . .	332, 335, 383, 402
<code>\AtBeginShipoutNext</code> . . .	2, 14, 28, 269		
<code>\AtBeginShipoutUpperLeft</code> 3, 7, 15, 378			
<code>\AtBeginShipoutUpperLeftForeground</code> . . . . .	3, 394		
<code>\AtBegShi@AbortIfUndefined</code> . . . . .	413, 420, 421, 422, 423		
<code>\AtBegShi@AddHook</code> . . . .	267, 271, 294		
<code>\AtBegShi@AddTo</code> . . . . .	275, 283, 295		
<code>\AtBegShi@AtEnd</code> . . . .	133, 134, 416, 677		
<code>\AtBegShi@BeginPicture</code> . . . . .	349, 365, 385, 404		
<code>\AtBegShi@CheckDefinable</code> . . . . .	155, 180, 265, 269, 273, 310, 313		
<code>\AtBegShi@Crop@ship</code> . . . . .	443, 450		
<code>\AtBegShi@Crop@ship</code> . . . . .	430, 448		
<code>\AtBegShi@Crop@shiplist</code> . . .	436, 449		
<code>\AtBegShi@Crop@shipout</code> . . . . .	426, 452, 454, 456, 458, 460, 462		
<code>\AtBegShi@Discardedfalse</code> . . .	217, 223		
<code>\AtBegShi@Discardedtrue</code> . . . . .	183		
<code>\AtBegShi@EndPicture</code> 359, 370, 387, 406			
<code>\AtBegShi@Everyshi@Output</code> . .	560, 562		
<code>\AtBegShi@Everyshi@shipout</code> .	514, 527, 529, 531, 533, 535, 537, 542		
<code>\AtBegShi@Everyshi@Test</code> . . .	518, 525		
<code>\AtBegShi@First</code> . . . . .	229, 237		
<code>\AtBegShi@FirstDisabled</code> . . .	259, 277		
<code>\AtBegShi@found</code> . . . .	451, 453, 455, 457, 459, 461, 463, 465, 468, 526, 528, 530, 532, 534, 536, 538, 543, 547, 550, 619, 621, 623, 625, 627, 629, 631, 633, 636		
<code>\AtBegShi@GroupLevel</code> . . . . .	197, 204, 469, 474, 551, 556, 637, 642		
<code>\AtBegShi@Hook</code> . . . . .	218, 262, 267		
<code>\AtBegShi@HookFirst</code> 240, 244, 264, 275			
<code>\AtBegShi@HookNext</code> . . .	219, 220, 263, 271		
<code>\AtBegShi@horigin</code> . . . .	331, 334, 381, 400		
<code>\AtBegShi@Item</code> . . . . .	295, 297		
<code>\AtBegShi@Memoir@shipi</code> . . . .	587, 609		
<code>\AtBegShi@Memoir@shipiiA</code> . . .	593, 610		
<code>\AtBegShi@Memoir@shipiiB</code> . . .	599, 613		
<code>\AtBegShi@Memoir@shipout</code> . . . . .	583, 620, 622, 624, 626, 628, 630		
<code>\AtBegShi@OrgProtect</code> . . .	215, 227, 230		
<code>\AtBegShi@OrgShipout</code> . . .	231, 313, 314, 454, 455, 529, 530, 622, 623		
<code>\AtBegShi@Output</code> . . . . .	207, 209		
		<b>B</b>	
		<code>\baselineskip</code> . . . . .	251, 439, 483
		<code>\begin</code> . . . . .	11, 50, 890
		<code>\body</code> . . . . .	700, 704
		<code>\box</code> . . . .	47, 225, 231, 391, 441, 485, 494, 565, 601, 656, 660, 767, 800, 801, 803, 804, 815, 872, 892
		<b>C</b>	
		<code>\catcode</code> . . . . .	58, 59, 60, 61, 62, 73, 74, 75, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99, 117, 118, 121, 122, 123, 124, 128, 129, 130, 131, 135, 137, 187, 188, 190, 191, 235, 236, 680, 681, 682, 683, 718, 727, 740, 741, 742, 743, 744, 745, 746, 747, 748, 749
		<code>\circle</code> . . . . .	8
		<code>\color</code> . . . . .	16, 865
		<code>\colorbox</code> . . . . .	883
		<code>\copy</code> . . . . .	254, 255, 375, 397, 793, 795, 797, 798
		<code>\count</code> . . . . .	772, 774, 775
		<code>\count@</code> . . . . .	685, 714, 718, 720, 721, 725, 727, 728, 729
		<code>\countdef</code> . . . . .	685
		<code>\CROP@ship</code> . . . . .	434, 450, 478, 487
		<code>\CROP@kernel</code> . . . . .	440, 484
		<code>\CROP@ship</code> . . . . .	427, 448, 470, 473
		<code>\CROP@shiplist</code> . . . .	445, 449, 480, 491
		<code>\CROP@shipout</code> . . . . .	444, 493, 531, 532, 624, 625
		<code>\csname</code> 63, 76, 100, 113, 120, 150, 156, 186, 216, 284, 312, 319, 325, 339, 342, 353, 415, 540, 684, 687, 690, 693, 732, 754, 888, 892	
		<code>\currentgrouplevel</code> . . . . .	197, 204, 469, 474, 551, 556, 637, 642
		<b>D</b>	
		<code>\deadcycles</code> . . . . .	182
		<code>\documentclass</code> . . . . .	2, 37, 857
		<code>\dp</code> . . . . .	249, 374
		<b>E</b>	
		<code>\E</code> . . . . .	236
		<code>\eject</code> . . . . .	841
		<code>\empty</code> . . . . .	67
		<code>\end</code> . . . . .	34, 54, 755, 850, 894
		<code>\endcsname</code> . . . . .	63, 76, 100, 113, 120, 150, 156, 186, 216, 284, 312, 319, 325, 339, 342, 353, 415, 540, 684, 687, 690, 693, 732, 754, 888, 892

<code>\endinput</code> .....	85, 417	<code>\MessageBreak</code> .....	279, 303
<code>\endpicture</code> .....	360	<code>\msg</code> .....	759, 760, 762, 763, 775, 779, 822, 826, 829, 835, 845
<code>\errmessage</code> .....	167		
<b>F</b>		<b>N</b>	
<code>\fill</code> .....	25	<code>\NeedsTeXFormat</code> .....	853
<b>G</b>		<code>\newbox</code> .....	766
<code>\g@addto@macro</code> .....	292	<code>\newif</code> .....	179
<code>\gdef</code> .....	220, 262, 263, 264, 316	<code>\newpage</code> .....	13, 22, 27, 32, 52
<code>\GPTorg@shipout</code> .....		<code>\next</code> .....	705, 707, 709
.....	458, 459, 533, 534, 626, 627	<code>\null</code> .....	783, 784, 812, 819, 833, 834, 843, 848, 891
<b>H</b>		<code>\number</code> .....	197, 469, 551, 637
<code>\hbox</code> .....	45, 195, 366, 379, 395, 794, 806, 807, 863, 872	<b>P</b>	
<code>\hrule</code> .....	810, 871, 873	<code>\p@</code> .....	196
<code>\hspace</code> .....	882, 885	<code>\PackageInfo</code> .....	81
<code>\hss</code> .....	410, 867	<code>\pagecolor</code> .....	856
<code>\ht</code> .....	248, 373, 384, 403	<code>\paperheight</code> .....	8, 17, 18, 884
<b>I</b>		<code>\paperwidth</code> .....	8, 17, 18, 882, 885
<code>\ifAtBegShi@Discarded</code> ..	179, 221, 298	<code>\par</code> .....	24
<code>\ifcase</code> .....	64, 158, 338, 610	<code>\pdfhorigin</code> .....	331
<code>\ifdim</code> .....	203	<code>\pdfvorigin</code> .....	332
<code>\ifnum</code> ....	204, 474, 556, 642, 720, 728	<code>\picture</code> .....	351
<code>\ifpdf</code> .....	330	<code>\protect</code> .....	215, 227, 230
<code>\ifvbox</code> .....	653	<code>\ProvidesFile</code> .....	854
<code>\ifvoid</code> .....	212, 301, 431, 488, 519, 563, 588, 600, 649	<code>\ProvidesPackage</code> .....	114
<code>\ifx</code> .....	65, 67, 76, 100, 108, 150, 156, 158, 161, 186, 240, 284, 319, 325, 339, 342, 353, 415, 448, 449, 450, 452, 454, 456, 458, 460, 462, 465, 525, 527, 529, 531, 533, 535, 537, 540, 542, 547, 609, 610, 613, 620, 622, 624, 626, 628, 630, 633, 684, 687, 690, 693, 732	<code>\put</code> .....	8, 17, 18
<code>\ignorespaces</code> .....	357, 368	<b>R</b>	
<code>\immediate</code> .....	78, 102, 759	<code>\RangeCatcodeInvalid</code> .....	
<code>\input</code> .....	151, 326, 733, 758	.....	724, 736, 737, 738, 739
<code>\iterate</code> .....	701, 703, 705	<code>\repeat</code> .....	699, 711, 722, 730
<b>K</b>		<code>\RequirePackage</code> .....	153, 328, 855
<code>\kern</code> .....	196, 381, 383, 384, 400, 402, 403, 862, 864	<code>\RestoreCatcodes</code> ..	713, 716, 717, 751
<b>L</b>		<code>\rlap</code> .....	380, 396, 399
<code>\lastkern</code> .....	203	<code>\rule</code> .....	866, 884
<code>\line</code> .....	17, 18	<b>S</b>	
<code>\lineskip</code> .....	252, 437, 481	<code>\section</code> .....	12, 893
<code>\lineskiplimit</code> .....	253, 438, 482	<code>\setbox</code> .....	45, 195, 200, 225, 242, 250, 366, 379, 395, 428, 471, 490, 516, 553, 565, 585, 639, 651, 769, 794, 815, 860
<code>\LoadCommand</code> .....	733, 750	<code>\shipout</code> .....	314, 315, 452, 453, 527, 528, 620, 621, 784, 791, 795, 798, 801, 804, 807, 810, 819, 825, 832, 834, 848, 891, 892
<code>\loop</code> .....	699, 715, 726	<code>\space</code> .....	168, 843
<b>M</b>		<code>\special</code> .....	46
<code>\makeatletter</code> .....	877	<b>T</b>	
<code>\makeatother</code> .....	880	<code>\Test</code> .....	735, 753
<code>\mem@oldshipout</code> .....	462, 463, 537, 538, 594, 601, 603, 659	<code>\testmsg</code> .....	761, 783, 790, 793, 797, 800, 803, 806, 809, 812, 824, 831, 833, 838, 843
<code>\mem@shipi</code> .....	584, 609, 638, 641	<code>\THBorg@shipout</code> .....	
<code>\mem@shipii</code> ...	591, 610, 613, 646, 648	.....	460, 461, 535, 536, 628, 629
		<code>\the</code>	121, 122, 123, 124, 135, 288, 718, 775
		<code>\TMP@EnsureCode</code>	132, 139, 140, 141, 142, 143, 144, 145, 146, 147, 148
		<code>\toks</code> .....	287, 288
		<code>\trimmarks</code> .....	595, 604, 652

U		W	
<code>\unitlength</code>	355	<code>\wd</code>	247, 395
<code>\unvbox</code>	596, 605, 654	<code>\write</code>	78, 102, 759
<code>\usepackage</code>	3, 4, 5, 38, 39, 858, 876, 889		
V		X	
<code>\vbox</code>	242, 250, 382, 401, 444, 490, 594, 603, 651, 809, 810, 860, 861	<code>\X</code>	235
<code>\vfill</code>	840	<code>\x</code>	63, 65, 67, 77, 81, 83, 101, 106, 113, 119, 127
<code>\void</code>	767, 769, 791, 825, 832		
<code>\VoidBox</code>	813, 818, 820	Z	
<code>\voidbox</code>	766, 767, 769	<code>\z@</code>	164, 182, 203, 242, 247, 248, 249, 253, 254, 287, 288, 437, 438, 439, 481, 482, 483, 611, 614
<code>\vrule</code>	794, 807, 872	<code>\z@skip</code>	251, 252
<code>\vspace</code>	25		
<code>\vss</code>	388, 407, 869		