

# The atbegshi package

Heiko Oberdiek  
<oberdiek@uni-freiburg.de>

2007/09/09 v1.6

## 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-TeX are supported.

## Contents

<b>1</b>	<b>Documentation</b>	<b>2</b>
1.1	Example . . . . .	3
<b>2</b>	<b>Method of <code>\shipout</code> overloading</b>	<b>4</b>
2.1	<code>\shipout</code> . . . . .	4
2.2	<code>\afterassignment</code> . . . . .	4
2.3	Test for direct or indirect boxes . . . . .	5
2.3.1	With $\varepsilon$ -TeX . . . . .	5
2.3.2	Without $\varepsilon$ -TeX . . . . .	5
2.3.3	<code>\lastkern</code> method . . . . .	6
2.4	Output . . . . .	7
2.5	Separate box register . . . . .	7
2.6	Summary . . . . .	7
2.6.1	With $\varepsilon$ -TeX . . . . .	7
2.6.2	Without $\varepsilon$ -TeX, traditional way . . . . .	8
2.6.3	<code>\lastkern</code> method . . . . .	9
<b>3</b>	<b>Implementation</b>	<b>9</b>
3.1	Reload check and package identification . . . . .	9
3.2	Catcodes . . . . .	10
3.3	Preparations . . . . .	11
3.4	Positioning . . . . .	14
3.5	Patches . . . . .	15
3.5.1	Package <code>crop</code> . . . . .	16
3.5.2	Package <code>everyshi</code> . . . . .	17
3.5.3	Class <code>memoir</code> . . . . .	18
<b>4</b>	<b>Test</b>	<b>20</b>
4.1	Catcode checks for loading . . . . .	20
<b>5</b>	<b>Installation</b>	<b>24</b>
5.1	Download . . . . .	24
5.2	Bundle installation . . . . .	24
5.3	Package installation . . . . .	24
5.4	Refresh file name databases . . . . .	25
5.5	Some details for the interested . . . . .	25

<b>6 History</b>	<b>25</b>
[2007/04/17 v1.0]	25
[2007/04/18 v1.1]	25
[2007/04/19 v1.2]	25
[2007/04/26 v1.3]	26
[2007/04/27 v1.4]	26
[2007/06/06 v1.5]	26
[2007/09/09 v1.6]	26
<b>7 Index</b>	<b>26</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.

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}
```

## 1.1 Example

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

```
1 <*example>
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 \example

```

## 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{}` (`= \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 ⇒ \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}
⇒ \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 chosen 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 0pt. 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 0pt.

```
\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
\begin{group}
\setbox\mybox=\box\mybox % ensure \mybox is void
\end{group}
\let\original@shipout\shipout

\shipout :=
% trick to get a void box \mybox
\begin{group}
\setbox\mybox=\box\mybox
\end{group}
\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  $\mathrm{T}_{\mathrm{E}}\mathrm{X}$  without  $\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$  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  $\varepsilon\text{-}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ 's \currentgrouplevel, if it is available. Otherwise the \lastkern method is used.

```
36 <*package>
```

### 3.1 Reload check and package identification

Reload check, especially if the package is not used with  $\mathrm{L}^{\mathrm{A}}\mathrm{T}_{\mathrm{E}}\mathrm{X}$ .

```
37 \begingroup
38 \catcode44 12 % ,
39 \catcode45 12 % -
40 \catcode46 12 % .
41 \catcode58 12 % :
42 \catcode64 11 % @
43 \expandafter\let\expandafter\x\csname ver@atbegshi.sty\endcsname
44 \ifcase 0%
45   \ifx\x\relax % plain
46   \else
47     \ifx\x\empty % LaTeX
48     \else
49       1%
50     \fi
51   \fi
52 \else
53   \catcode35 6 % #
54   \catcode123 1 % {
```

```

55 \catcode125 2 % }
56 \expandafter\ifx\csname PackageInfo\endcsname\relax
57 \def\x#1#2{%
58 \immediate\write-1{Package #1 Info: #2.}%
59 }%
60 \else
61 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
62 \fi
63 \x{atbegshi}{The package is already loaded}%
64 \endgroup
65 \expandafter\endinput
66 \fi
67 \endgroup

```

Package identification:

```

68 \begingroup
69 \catcode35 6 % #
70 \catcode40 12 % (
71 \catcode41 12 % )
72 \catcode44 12 % ,
73 \catcode45 12 % -
74 \catcode46 12 % .
75 \catcode47 12 % /
76 \catcode58 12 % :
77 \catcode64 11 % @
78 \catcode123 1 % {
79 \catcode125 2 % }
80 \expandafter\ifx\csname ProvidesPackage\endcsname\relax
81 \def\x#1#2#3[#4]{\endgroup
82 \immediate\write-1{Package: #3 #4}%
83 \xdef#1{#4}%
84 }%
85 \else
86 \def\x#1#2[#3]{\endgroup
87 #2[#{#3}]%
88 \ifx#1\relax
89 \xdef#1{#3}%
90 \fi
91 }%
92 \fi
93 \expandafter\x\csname ver@atbegshi.sty\endcsname
94 \ProvidesPackage{atbegshi}%
95 [2007/09/09 v1.6 At begin shipout hook (H0)]

```

## 3.2 Catcodes

```

96 \begingroup
97 \catcode123 1 % {
98 \catcode125 2 % }
99 \def\x{\endgroup
100 \expandafter\edef\csname AtBegShi@AtEnd\endcsname{%
101 \catcode35 \the\catcode35\relax
102 \catcode64 \the\catcode64\relax
103 \catcode123 \the\catcode123\relax
104 \catcode125 \the\catcode125\relax
105 }%
106 }%
107 \x
108 \catcode35 6 % #
109 \catcode64 11 % @
110 \catcode123 1 % {
111 \catcode125 2 % }
112 \def\TMP@EnsureCode#1#2{%

```

```

113 \edef\AtBegShi@AtEnd{%
114   \AtBegShi@AtEnd
115   \catcode#1 \the\catcode#1\relax
116 }%
117 \catcode#1 #2\relax
118 }
119 \TMP@EnsureCode{40}{12}% (
120 \TMP@EnsureCode{41}{12}% )
121 \TMP@EnsureCode{44}{12}% ,
122 \TMP@EnsureCode{45}{12}% -
123 \TMP@EnsureCode{47}{12}% /
124 \TMP@EnsureCode{46}{12}% .
125 \TMP@EnsureCode{58}{12}% :
126 \TMP@EnsureCode{61}{12}% =
127 \TMP@EnsureCode{94}{7}% ^ (superscript)
128 \TMP@EnsureCode{96}{12}% ‘

```

### 3.3 Preparations

```

129 \begingroup\expandafter\expandafter\expandafter\endgroup
130 \expandafter\ifx\csname RequirePackage\endcsname\relax
131   \input infwarerr.sty\relax
132 \else
133   \RequirePackage{infwarerr}[2007/09/09]%
134 \fi

```

\AtBegShi@CheckDefinable

```

135 \begingroup\expandafter\expandafter\expandafter\endgroup
136 \expandafter\ifx\csname @ifdefinable\endcsname\relax
137   \def\AtBegShi@CheckDefinable#1{%
138     \ifcase\ifx#1\relax
139       \@ne
140     \else
141       \ifx#1\@undefined
142         \@ne
143       \else
144         \z@
145       \fi
146     \fi
147     \errmessage{%
148       Package atbegshi: \string#1\space
149       is already defined%
150     }%
151   \endgroup
152 \fi
153 }%
154 \else
155   \def\AtBegShi@CheckDefinable#1{%
156     \@ifdefinable{#1}{}%
157   }%
158 \fi

```

```

159 \newif\ifAtBegShi@Discarded

```

\AtBeginShipoutDiscard

```

160 \AtBegShi@CheckDefinable\AtBeginShipoutDiscard
161 \def\AtBeginShipoutDiscard{%
162   \deadcycles=\z@
163   \global\AtBegShi@Discardedtrue
164 }

165 \begingroup\expandafter\expandafter\expandafter\endgroup
166 \expandafter\ifx\csname currentgrouplevel\endcsname\relax
167   \catcode‘X=9 % ignore

```

```

168 \catcode'E=14 % comment
169 \else
170 \catcode'X=14 % comment
171 \catcode'E=9 % ignore
172 \fi

\AtBegShi@Shipout

173 \def\AtBegShi@Shipout{%
174 X \begingroup
175 X \setbox\AtBeginShipoutBox=\hbox\bgroup
176 X \kern\p@
177 E \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
178 \afterassignment\AtBegShi@Test
179 X \global
180 \setbox\AtBeginShipoutBox=%
181 }

\AtBegShi@Test

182 \def\AtBegShi@Test{%
183 X \ifdim\lastkern=\z@
184 E \ifnum\AtBegShi@GroupLevel<\currentgrouplevel
185 \expandafter\aftergroup
186 \fi
187 \AtBegShi@Output
188 }

\AtBegShi@Output

189 \def\AtBegShi@Output{%
190 X \egroup
191 X \endgroup
192 \ifvoid\AtBeginShipoutBox
193 \@PackageWarning{atbegshi}{Ignoring void shipout box}%
194 \else
195 \let\AtBegShi@OrgProtect\protect
196 \csname set@typeset@protect\endcsname
197 \global\AtBegShi@Discardedfalse
198 \AtBegShi@Hook
199 \AtBegShi@HookNext
200 \gdef\AtBegShi@HookNext{%
201 \ifAtBegShi@Discarded
202 \@PackageInfoNoLine{atbegshi}{Shipout page discarded}%
203 \global\AtBegShi@Discardedfalse
204 \begingroup
205 \setbox\AtBeginShipoutBox\box\AtBeginShipoutBox
206 \endgroup
207 \let\protect\AtBegShi@OrgProtect
208 \else
209 \AtBegShi@First
210 \let\protect\AtBegShi@OrgProtect
211 \AtBegShi@OrgShipout\box\AtBeginShipoutBox
212 \fi
213 \fi
214 }

215 \catcode'\X=11 %
216 \catcode'\E=11 %

\AtBegShi@First

217 \def\AtBegShi@First{%
218 \begingroup
219 \def\@empty{%
220 \ifx\AtBegShi@HookFirst\@empty

```

```

221     \else
222         \setbox\z@=\vbox{%
223             \begingroup
224                 \AtBegShi@HookFirst
225             \endgroup
226         }%
227         \wd\z@=\z@
228         \ht\z@=\z@
229         \dp\z@=\z@
230         \global\setbox\AtBeginShipoutBox=\vbox{%
231             \baselineskip\z@skip
232             \lineskip\z@skip
233             \lineskiplimit\z@
234             \copy\z@
235             \copy\AtBeginShipoutBox
236         }%
237     \fi
238     \global\let\AtBegShi@First\@empty
239     \global\let\AtBeginShipoutFirst\AtBegShi@FirstDisabled
240 \endgroup
241 }

\AtBegShi@Hook
242 \gdef\AtBegShi@Hook{}

\AtBegShi@HookNext
243 \gdef\AtBegShi@HookNext{}

\AtBegShi@HookFirst
244 \gdef\AtBegShi@HookFirst{}

\AtBeginShipout
245 \AtBegShi@CheckDefinable\AtBeginShipout
246 \def\AtBeginShipout{%
247     \AtBegShi@AddHook\AtBegShi@Hook
248 }

\AtBeginShipoutNext
249 \AtBegShi@CheckDefinable\AtBeginShipoutNext
250 \def\AtBeginShipoutNext{%
251     \AtBegShi@AddHook\AtBegShi@HookNext
252 }

\AtBeginShipoutFirst
253 \AtBegShi@CheckDefinable\AtBeginShipoutFirst
254 \def\AtBeginShipoutFirst{%
255     \AtBegShi@AddTo\AtBegShi@HookFirst
256 }

\AtBegShi@FirstDisabled
257 \long\def\AtBegShi@FirstDisabled#1{%
258     \@PackageWarning{atbegshi}{%
259         First page is already shipped out, ignoring\MessageBreak
260         \string\AtBeginShipoutFirst
261     }%
262 }

\AtBegShi@AddTo
263 \begingroup\expandafter\expandafter\expandafter\endgroup
264 \expandafter\ifx\csname g@addto@macro\endcsname\relax
265     \long\def\AtBegShi@AddTo#1#2{%

```

```

266 \begingroup
267 \toks\z@\expandafter{#1#2}%
268 \xdef#1{\the\toks\z@}%
269 \endgroup
270 }%
271 \else
272 \let\AtBegShi@AddTo\g@addto@macro
273 \fi

```

\AtBegShi@AddHook

```

274 \long\def\AtBegShi@AddHook#1#2{%
275 \AtBegShi@AddTo#1{\AtBegShi@Item{#2}}%
276 }

```

\AtBegShi@Item

```

277 \long\def\AtBegShi@Item#1{%
278 \ifAtBegShi@Discarded
279 \else
280 #1%
281 \ifvoid\AtBeginShipoutBox
282 \@PackageWarning{atbegshi}{%
283 Shipout box was voided by hook,\MessageBreak
284 ignoring shipout box%
285 }%
286 \AtBeginShipoutDiscard
287 \fi
288 \fi
289 }

```

\AtBeginShipoutInit

```

290 \AtBegShi@CheckDefinable\AtBeginShipoutInit
291 \def\AtBeginShipoutInit{%
292 \csname newbox\endcsname\AtBeginShipoutBox
293 \AtBegShi@CheckDefinable\AtBegShi@OrgShipout
294 \global\let\AtBegShi@OrgShipout\shipout
295 \global\let\shipout\AtBegShi@Shipout
296 \gdef\AtBeginShipoutInit{}%
297 }

298 \begingroup\expandafter\expandafter\expandafter\endgroup
299 \expandafter\ifx\csname AtBeginDocument\endcsname\relax
300 \AtBeginShipoutInit
301 \else
302 \AtBeginDocument{\AtBeginShipoutInit}%
303 \fi

```

### 3.4 Positioning

```

304 \begingroup\expandafter\expandafter\expandafter\endgroup
305 \expandafter\ifx\csname RequirePackage\endcsname\relax
306 \input ifpdf.sty\relax
307 \else
308 \RequirePackage{ifpdf}\relax
309 \fi

310 \ifpdf
311 \def\AtBegShi@horigin{\pdfhorigin}%
312 \def\AtBegShi@vorigin{\pdfvorigin}%
313 \else
314 \def\AtBegShi@horigin{72.27pt}%
315 \def\AtBegShi@vorigin{72.27pt}%
316 \fi

```

```

317 \begingroup
318 \ifcase
319   \expandafter\ifx\csname picture\endcsname\relax
320     1%
321   \else
322     \expandafter\ifx\csname endpicture\endcsname\relax
323       1%
324     \else
325       0%
326     \fi
327   \fi
328 \endgroup
329 \def\AtBegShi@BeginPicture{%
330   \begingroup
331   \picture(0,0)\relax
332   \begingroup\expandafter\expandafter\expandafter\endgroup
333   \expandafter\ifx\csname unitlength\endcsname\relax
334   \else
335     \unitlength=1pt\relax
336   \fi
337   \ignorespaces
338 }%
339 \def\AtBegShi@EndPicture{%
340   \endpicture
341   \endgroup
342 }%
343 \else
344   \endgroup
345   \def\AtBegShi@BeginPicture{%
346     \setbox0=\hbox\bgroup
347     \begingroup
348     \ignorespaces
349   }%
350   \def\AtBegShi@EndPicture{%
351     \endgroup
352     \egroup
353     \ht0=0pt\relax
354     \dp0=0pt\relax
355     \copy0 %
356   }%
357 \fi
358 \def\AtBeginShipoutUpperLeft#1{%
359   \global\setbox\AtBeginShipoutBox=\hbox{%
360     \rlap{%
361       \kern-\AtBegShi@horigin\relax
362       \vbox to 0pt{%
363         \kern-\AtBegShi@vorigin\relax
364         \kern-\ht\AtBeginShipoutBox
365         \AtBegShi@BeginPicture
366         #1%
367         \AtBegShi@EndPicture
368         \vss
369       }%
370     }%
371   \box\AtBeginShipoutBox
372 }%
373 }

```

### 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.

```

374 \def\AtBegShi@AbortIfUndefined#1{%

```

```

375 \begingroup\expandafter\expandafter\expandafter\endgroup
376 \expandafter\ifx\csname#1\endcsname\relax
377   \AtBegShi@AtEnd
378   \expandafter\endinput
379 \fi
380 }
381 \AtBegShi@AbortIfUndefined{currentgrouplevel}
382 \AtBegShi@AbortIfUndefined{AtBeginDocument}
383 \AtBegShi@AbortIfUndefined{@ifpackageloaded}
384 \AtBegShi@AbortIfUndefined{@ifclassloaded}

```

### 3.5.1 Package crop

Fix of method and box.

```

385 \def\AtBegShi@PatchCrop{%
386   \begingroup
387   \def\AtBegShi@Crop@shipout{%
388     \afterassignment\CROP@ship
389     \setbox\@cclv=%
390   }%
391   \def\AtBegShi@Crop@ship{%
392     \ifvoid\@cclv
393       \expandafter\aftergroup
394     \fi
395     \CROP@@ship
396   }%
397   \def\AtBegShi@Crop@shiplist{%
398     \lineskip\z@
399     \lineskiplimit\z@
400     \baselineskip\z@
401     \CROP@kernel
402     \box\@cclv
403   }%
404   \def\AtBegShi@Crop@@ship{%
405     \CROP@shipout\vbox{%
406       \CROP@shiplist
407     }%
408   }%
409   \ifx\AtBegShi@Crop@ship\CROP@ship
410     \ifx\AtBegShi@Crop@shiplist\CROP@shiplist
411       \ifx\AtBegShi@Crop@@ship\CROP@@ship
412         \let\AtBegShi@found\relax
413         \ifx\shipout\AtBegShi@Crop@shipout
414           \def\AtBegShi@found{\shipout}%
415         \else\ifx\AtBegShi@OrgShipout\AtBegShi@Crop@shipout
416           \def\AtBegShi@found{\AtBegShi@OrgShipout}%
417         \else\ifx\@EveryShipout@Org@Shipout\AtBegShi@Crop@shipout
418           \def\AtBegShi@found{\@EveryShipout@Org@Shipout}%
419         \else\ifx\GPTorg@shipout\AtBegShi@Crop@shipout
420           \def\AtBegShi@found{\GPTorg@shipout}%
421         \else\ifx\THBorg@shipout\AtBegShi@Crop@shipout
422           \def\AtBegShi@found{\THBorg@shipout}%
423         \else\ifx\mem@oldshipout\AtBegShi@Crop@shipout
424           \def\AtBegShi@found{\mem@oldshipout}%
425         \fi\fi\fi\fi\fi\fi
426         \ifx\AtBegShi@found\relax
427           \else
428             \expandafter\endgroup
429             \expandafter\def\AtBegShi@found{%
430               \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
431               \afterassignment\CROP@ship
432               \setbox\AtBeginShipoutBox=%

```



```

433     }%
434     \def\CROP@ship{%
435         \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
436         \else
437             \expandafter\aftergroup
438             \fi
439         \CROP@@ship
440     }%
441     \def\CROP@shiplist{%
442         \lineskip\z@
443         \lineskiplimit\z@
444         \baselineskip\z@
445         \CROP@kernel
446         \box\AtBeginShipoutBox
447     }%
448     \def\CROP@@ship{%
449         \ifvoid\AtBeginShipoutBox
450         \else
451             \setbox\AtBeginShipoutBox=\vbox{%
452                 \CROP@shiplist
453             }%
454             \expandafter\CROP@shipout
455             \expandafter\box
456             \expandafter\AtBeginShipoutBox
457         \fi
458     }%
459     \@PackageInfoNoLine{atbegshi}{Package 'crop' patched}%
460     \begingroup
461     \fi
462     \fi
463     \fi
464     \fi
465     \endgroup
466     \let\AtBegShi@PatchCrop\relax
467 }
468 \@ifpackageloaded{crop}{%
469     \AtBegShi@PatchCrop
470 }{%
471     \AtBeginDocument{\AtBegShi@PatchCrop}%
472 }

```

### 3.5.2 Package everyshi

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

```

473 \def\AtBegShi@PatchEveryshi{%
474     \begingroup
475     \long\def\AtBegShi@Everyshi@shipout{%
476         \afterassignment\@EveryShipout@Test
477         \global\setbox\@cclv= %
478     }%
479     \long\def\AtBegShi@Everyshi@Test{%
480         \ifvoid\@cclv\relax
481             \aftergroup\@EveryShipout@Output
482         \else
483             \@EveryShipout@Output
484         \fi
485     }%
486     \ifx\AtBegShi@Everyshi@Test\@EveryShipout@Test
487         \let\AtBegShi@found\relax
488         \ifx\shipout\AtBegShi@Everyshi@shipout
489             \def\AtBegShi@found{\shipout}%
490         \else\ifx\AtBegShi@OrgShipout\AtBegShi@Everyshi@shipout

```

```

491     \def\AtBegShi@found{\AtBegShi@OrgShipout}%
492 \else\ifx\CROP@shipout\AtBegShi@Everyshi@shipout
493     \def\AtBegShi@found{\CROP@shipout}%
494 \else\ifx\GPTorg@shipout\AtBegShi@Everyshi@shipout
495     \def\AtBegShi@found{\GPTorg@shipout}%
496 \else\ifx\THBorg@shipout\AtBegShi@Everyshi@shipout
497     \def\AtBegShi@found{\THBorg@shipout}%
498 \else\ifx\mem@oldshipout\AtBegShi@Everyshi@shipout
499     \def\AtBegShi@found{\mem@oldshipout}%
500 \else
501     \expandafter\ifx\csname @EveryShipout@Org@Shipout\endcsname
502         \relax
503         \ifx\@EveryShipout@Shipout\AtBegShi@Everyshi@shipout
504             \def\AtBegShi@found{\@EveryShipout@Shipout}%
505         \fi
506     \fi
507 \fi\fi\fi\fi\fi\fi
508 \ifx\AtBegShi@found\relax
509 \else
510     \expandafter\endgroup
511     \expandafter\def\AtBegShi@found{%
512         \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
513         \afterassignment\@EveryShipout@Test
514         \setbox\AtBeginShipoutBox=%
515     }%
516     \def\@EveryShipout@Test{%
517         \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
518         \else
519             \expandafter\aftergroup
520             \fi
521         \AtBegShi@Everyshi@Output
522     }%
523     \def\AtBegShi@Everyshi@Output{%
524         \ifvoid\AtBeginShipoutBox
525         \else
526             \global\setbox\@cclv\box\AtBeginShipoutBox
527             \expandafter\@EveryShipout@Output
528         \fi
529     }%
530     \@PackageInfoNoLine{atbegshi}{Package ‘everyshi’ patched}%
531     \begingroup
532     \fi
533 \fi
534 \endgroup
535 \let\AtBegShi@PatchEveryshi\relax
536 }
537 \ifpackageloaded{everyshi}{%
538     \AtBegShi@PatchEveryshi
539 }{%
540     \AtBeginDocument{\AtBegShi@PatchEveryshi}%
541 }

```

### 3.5.3 Class memoir

Fix of method and box.

```

542 \def\AtBegShi@PatchMemoir{%
543     \begingroup
544     \def\AtBegShi@Memoir@shipout{%
545         \afterassignment\mem@shipi
546         \setbox\@cclv=%
547     }%
548     \def\AtBegShi@Memoir@shipi{%

```

```

549     \ifvoid\@cclv
550     \expandafter\aftergroup
551     \fi
552     \mem@shipii
553 }%
554 \def\AtBegShi@Memoir@shipiiA{%
555     \mem@oldshipout\vbox{%
556         \trimmarks
557         \unvbox\@cclv
558     }%
559 }%
560 \def\AtBegShi@Memoir@shipiiB{%
561     \ifvoid\@cclv
562     \mem@oldshipout\box\@cclv
563     \else
564     \mem@oldshipout\vbox{%
565         \trimmarks
566         \unvbox\@cclv
567     }%
568     \fi
569 }%
570 \ifx\AtBegShi@Memoir@shipi\mem@shipi
571     \ifcase\ifx\AtBegShi@Memoir@shipiiA\mem@shipii
572         \z@
573     \else
574         \ifx\AtBegShi@Memoir@shipiiB\mem@shipii
575             \z@
576         \else
577             \@ne
578         \fi
579     \fi
580     \let\AtBegShi@found\relax
581     \ifx\shipout\AtBegShi@Memoir@shipout
582         \def\AtBegShi@found{\shipout}%
583     \else\ifx\AtBegShi@OrgShipout\AtBegShi@Memoir@shipout
584         \def\AtBegShi@found{\AtBegShi@OrgShipout}%
585     \else\ifx\CROP@shipout\AtBegShi@Memoir@shipout
586         \def\AtBegShi@found{\CROP@shipout}%
587     \else\ifx\GPTorg@shipout\AtBegShi@Memoir@shipout
588         \def\AtBegShi@found{\GPTorg@shipout}%
589     \else\ifx\THBorg@shipout\AtBegShi@Memoir@shipout
590         \def\AtBegShi@found{\THBorg@shipout}%
591     \else\ifx\@EveryShipout@Org@Shipout\AtBegShi@Memoir@shipout
592         \def\AtBegShi@found{\@EveryShipout@Org@Shipout}%
593     \fi\fi\fi\fi\fi\fi
594     \ifx\AtBegShi@found\relax
595     \else
596     \expandafter\endgroup
597     \expandafter\def\AtBegShi@found{%
598         \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
599         \afterassignment\mem@shipi
600         \setbox\AtBeginShipoutBox=%
601     }%
602     \def\mem@shipi{%
603         \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
604         \else
605             \expandafter\aftergroup
606         \fi
607         \mem@shipii
608     }%
609     \def\mem@shipii{%
610         \ifvoid\AtBeginShipoutBox

```

```

611         \else
612         \setbox\AtBeginShipoutBox=\vbox{%
613         \trimmarks
614         \ifvbox\AtBeginShipoutBox
615         \unvbox\AtBeginShipoutBox
616         \else
617         \box\AtBeginShipoutBox
618         \fi
619         }%
620         \expandafter\mem@oldshipout
621         \expandafter\box
622         \expandafter\AtBeginShipoutBox
623         \fi
624     }%
625     \@PackageInfoNoLine{atbegshi}{Class 'memoir' patched}%
626     \begingroup
627     \fi
628     \fi
629     \fi
630 \endgroup
631 \let\AtBegShi@PatchMemoir\relax
632 }
633 \ifclassloaded{memoir}{%
634     \AtBegShi@PatchMemoir
635 }{%
636     \AtBeginDocument{\AtBegShi@PatchMemoir}%
637 }
638 \AtBegShi@AtEnd
639 </package>

```

## 4 Test

### 4.1 Catcode checks for loading

```

640 <*test1>
641 \catcode'\{=1 %
642 \catcode'\}=2 %
643 \catcode'\#=6 %
644 \catcode'\@=11 %
645 \expandafter\ifx\csname count@\endcsname\relax
646     \countdef\count@=255 %
647 \fi
648 \expandafter\ifx\csname @gobble\endcsname\relax
649     \long\def\@gobble#1{%
650 \fi
651 \expandafter\ifx\csname @firstofone\endcsname\relax
652     \long\def\@firstofone#1{#1}%
653 \fi
654 \expandafter\ifx\csname loop\endcsname\relax
655     \expandafter\@firstofone
656 \else
657     \expandafter\@gobble
658 \fi
659 {%
660     \def\loop#1\repeat{%
661         \def\body{#1}%
662         \iterate
663     }%
664     \def\iterate{%
665         \body
666         \let\next\iterate

```

```

667     \else
668         \let\next\relax
669     \fi
670     \next
671 }%
672 \let\repeat=\fi
673 }%
674 \def\RestoreCatcodes{}
675 \count@=0 %
676 \loop
677   \edef\RestoreCatcodes{%
678     \RestoreCatcodes
679     \catcode\the\count@=\the\catcode\count@\relax
680   }%
681 \ifnum\count@<255 %
682   \advance\count@ 1 %
683 \repeat
684
685 \def\RangeCatcodeInvalid#1#2{%
686   \count@=#1\relax
687   \loop
688     \catcode\count@=15 %
689     \ifnum\count@<#2\relax
690       \advance\count@ 1 %
691     \repeat
692 }
693 \expandafter\ifx\csname LoadCommand\endcsname\relax
694   \def\LoadCommand{\input atbegshi.sty\relax}%
695 \fi
696 \def\Test{%
697   \RangeCatcodeInvalid{0}{47}%
698   \RangeCatcodeInvalid{58}{64}%
699   \RangeCatcodeInvalid{91}{96}%
700   \RangeCatcodeInvalid{123}{255}%
701   \catcode'\@=12 %
702   \catcode'\=0 %
703   \catcode'\{=1 %
704   \catcode'\}=2 %
705   \catcode'\#=6 %
706   \catcode'\[=12 %
707   \catcode'\]=12 %
708   \catcode'\%=14 %
709   \catcode'\ =10 %
710   \catcode13=5 %
711   \LoadCommand
712   \RestoreCatcodes
713 }
714 \Test
715 \csname @@end\endcsname
716 \end
717 </test1>
718 <*test2>
719 \input atbegshi.sty\relax
720 \def\msg#{\immediate\write16}
721 \msg{File: atbegshi-test2.tex 2007/09/09 v1.6 Test file for plain-TeX}
722 \def\testmsg#1#2{%
723   \msg{}%
724   \msg{*** Test with box (#1), expected page output [#2]}% hash-ok
725 }
726
727 \newbox\voidbox
728 \def\void{\box\voidbox}

```

```

729 \begingroup
730   \setbox\voidbox=\void
731 \endgroup
732
733 \count0=0\relax
734 \AtBeginShipout{%
735   \global\advance\count0 by 1\relax
736   \msg{* Inside \string\AtBeginShipout: [\the\count0]}%
737 }
738
739 \AtBeginShipoutFirst{%
740   \msg{* Inside \string\AtBeginShipoutFirst}%
741   Hello World%
742 }
743
744 \testmsg{\string\null}{1}
745 \shipout\null
746
747 \AtBeginShipoutFirst{%
748   This is too late%
749 }
750
751 \testmsg{void}{}
752 \shipout\void
753
754 \testmsg{\string\copy255 (not void)}{2}
755 \setbox255\hbox{\vrule height 10bp width 10bp}
756 \shipout\copy255 %
757
758 \testmsg{\string\copy255 (again)}{3}
759 \shipout\copy255 %
760
761 \testmsg{\string\box255}{4}
762 \shipout\box255 %
763
764 \testmsg{\string\box255 (again)}{}
765 \shipout\box255 %
766
767 \testmsg{\string\hbox}{5}
768 \shipout\hbox{\vrule height 5bp width 20bp}
769
770 \testmsg{\string\vbox}{6}
771 \shipout\vbox{\hrule height 20bp width 5bp}
772
773 \testmsg{\string\null, voided by hook}{}
774 \def\VoidBox{%
775   \begingroup
776     \setbox\AtBeginShipoutBox=\box\AtBeginShipoutBox
777   \endgroup
778 }
779 \AtBeginShipout{\VoidBox}
780 \shipout\null
781 \def\VoidBox{}
782
783 \msg{*** \string\begingroup}
784 \begingroup
785   \testmsg{void}{}%
786   \shipout\void
787 \msg{*** \string\endgroup}
788 \endgroup
789
790 \msg{*** \string\begingroup}

```

```

791 \begingroup
792   \testmsg{void}{}%
793   \shipout\void
794   \testmsg{\string\null}{8}%
795   \shipout\null
796 \msg{*** \string\endgroup}
797 \endgroup
798
799 \testmsg{output routine}{9}
800 Hello World
801 \vfill
802 \eject
803
804 \testmsg{\string\null\space(discarded)}{}
805 \AtBeginShipout{%
806   \msg{* Inside \string\AtBeginShipout: DISCARD}%
807   \AtBeginShipoutDiscard
808 }
809 \shipout\null
810
811 \end
812 \test2)

813 (*test3)
814 \NeedsTeXFormat{LaTeX2e}
815 \ProvidesFile{atbegshi-test3.tex}[2007/09/09 v1.6 Test file for LaTeX]
816 \RequirePackage{color}
817 \pagecolor{yellow}
818 \documentclass[a5paper,showtrims]{memoir}
819 \usepackage{atbegshi}
820 \AtBeginShipout{%
821   \setbox\AtBeginShipoutBox=\vbox{%
822     \vbox to 0pt{%
823       \kern-1.5in %
824       \hbox to 0pt{%
825         \kern-1.5in %
826         \color{blue}%
827         \rule{1in}{1in}%
828         \hss
829       }%
830       \vss
831     }%
832     \hrule
833     \hbox{\vrule\box\AtBeginShipoutBox\vrule}%
834     \hrule
835   }%
836 }
837 \usepackage{eso-pic}
838 \makeatletter
839 \@EveryShipout@Init
840 \let\@EveryShipout@Init\relax
841 \makeatother
842 \AddToShipoutPicture{%
843   \hspace{.52\paperwidth}%
844   \colorbox{cyan}{%
845     \rule{0mm}{\paperheight}%
846     \hspace{.48\paperwidth}%
847   }%
848 }

```

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

```

849 \expandafter\let\csname ver@crop.sty\endcsname\relax
850 \usepackage[color=red,cross,a4,center]{crop}

```

```

851 \begin{document}
852 \shipout\null
853 \shipout\box\csname voidb@x\endcsname
854 \section{Hello World}
855 \end{document}
856 \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-example.tex</code>	→ <code>doc/latex/oberdiek/atbegshi-example.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.

## 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		
\#	643, 705	\afterassignment . . . . . 178, 388, 431, 476, 513, 545, 599
\%	708	\aftergroup . . . . . 185, 393, 437, 481, 519, 550, 605
\@	644, 701	\AtBeginDocument . . . . . 302, 471, 540, 636
\@EveryShipout@Init	839, 840	\AtBeginShipout . . . . . 2, 6, 245, 734, 736, 779, 805, 806, 820
\@EveryShipout@Org@Shipout	417, 418, 591, 592	\AtBeginShipoutBox . . . . . 175, 180, 192, 205, 211, 230, 235, 281, 292, 359, 364, 371, 432, 446, 449, 451, 456, 514, 524, 526, 600, 610, 612, 614, 615, 617, 622, 776, 821, 833
\@EveryShipout@Output	481, 483, 527	\AtBeginShipoutDiscard . . . . . 3, 29, 160, 286, 807
\@EveryShipout@Shipout	503, 504	\AtBeginShipoutFirst . . . . . 3, 239, 253, 260, 739, 740, 747
\@EveryShipout@Test	476, 486, 513, 516	\AtBeginShipoutInit . . . . . 3, 290, 300, 302
\@PackageInfoNoLine	202, 459, 530, 625	\AtBeginShipoutNext . . . . . 2, 14, 28, 249
\@PackageWarning	193, 258, 282	\AtBeginShipoutUpperLeft 3, 7, 15, 358
\@ccclv	389, 392, 402, 477, 480, 526, 546, 549, 557, 561, 562, 566	\AtBegShi@AbortIfUndefined . . . . . 374, 381, 382, 383, 384
\@empty	219, 220, 238	\AtBegShi@AddHook . . . . . 247, 251, 274
\@firstofone	652, 655	\AtBegShi@AddTo . . . . . 255, 263, 275
\@gobble	649, 657	\AtBegShi@AtEnd . . . . . 113, 114, 377, 638
\@ifclassloaded	633	\AtBegShi@BeginPicture 329, 345, 365
\@ifdefinable	156	\AtBegShi@CheckDefinable . . . . . 135, 160, 245, 249, 253, 290, 293
\@ifpackageloaded	468, 537	\AtBegShi@Crop@ship . . . . . 404, 411
\@ne	139, 142, 577	\AtBegShi@Crop@ship . . . . . 391, 409
\@undefined	141	\AtBegShi@Crop@shiplist . . . . . 397, 410
\[	706	\AtBegShi@Crop@shipout . . . . . 387, 413, 415, 417, 419, 421, 423
\]	702	
\{	641, 703	
\}	642, 704	
\]	707	
\_	709	
A		
\AddToShipoutPicture	842	
\advance	682, 690, 735	

<code>\AtBegShi@Discardedfalse</code> . . .	197, 203	<code>\countdef</code> . . . . .	646
<code>\AtBegShi@Discardedtrue</code> . . . . .	163	<code>\CROP@@ship</code> . . . . .	395, 411, 439, 448
<code>\AtBegShi@endPicture</code> . .	339, 350, 367	<code>\CROP@kernel</code> . . . . .	401, 445
<code>\AtBegShi@Everyshi@Output</code> . .	521, 523	<code>\CROP@ship</code> . . . . .	388, 409, 431, 434
<code>\AtBegShi@Everyshi@shipout</code> .	475,	<code>\CROP@shiplist</code> . . . .	406, 410, 441, 452
	488, 490, 492, 494, 496, 498, 503	<code>\CROP@shipout</code> . . . . .	
<code>\AtBegShi@Everyshi@Test</code> . . .	479, 486		405, 454, 492, 493, 585, 586
<code>\AtBegShi@First</code> . . . . .	209, 217	<code>\csname</code> 43, 56, 80, 93, 100, 130, 136,	
<code>\AtBegShi@FirstDisabled</code> . . .	239, 257		166, 196, 264, 292, 299, 305,
<code>\AtBegShi@found</code> . . .	412, 414, 416,		319, 322, 333, 376, 501, 645,
	418, 420, 422, 424, 426, 429,		648, 651, 654, 693, 715, 849, 853
	487, 489, 491, 493, 495, 497,	<code>\currentgrouplevel</code> . . . . .	177,
	499, 504, 508, 511, 580, 582,		184, 430, 435, 512, 517, 598, 603
	584, 586, 588, 590, 592, 594, 597		
<code>\AtBegShi@GroupLevel</code> . . . . .	177,		
	184, 430, 435, 512, 517, 598, 603	<b>D</b>	
<code>\AtBegShi@Hook</code> . . . . .	198, 242, 247	<code>\deadcycles</code> . . . . .	162
<code>\AtBegShi@HookFirst</code> 220, 224, 244, 255		<code>\documentclass</code> . . . . .	2, 818
<code>\AtBegShi@HookNext</code> .	199, 200, 243, 251	<code>\dp</code> . . . . .	229, 354
<code>\AtBegShi@horigin</code> . . . .	311, 314, 361		
<code>\AtBegShi@Item</code> . . . . .	275, 277	<b>E</b>	
<code>\AtBegShi@Memoir@shipi</code> . . . .	548, 570	<code>\E</code> . . . . .	216
<code>\AtBegShi@Memoir@shipiiA</code> . . .	554, 571	<code>\eject</code> . . . . .	802
<code>\AtBegShi@Memoir@shipiiB</code> . . .	560, 574	<code>\empty</code> . . . . .	47
<code>\AtBegShi@Memoir@shipout</code> . . . . .		<code>\end</code> . . . . .	34, 716, 811, 855
	544, 581, 583, 585, 587, 589, 591	<code>\endcsname</code> . . . . .	
<code>\AtBegShi@OrgProtect</code> . .	195, 207, 210		43, 56, 80, 93, 100, 130, 136,
<code>\AtBegShi@OrgShipout</code> . . .	211, 293,		166, 196, 264, 292, 299, 305,
	294, 415, 416, 490, 491, 583, 584		319, 322, 333, 376, 501, 645,
<code>\AtBegShi@Output</code> . . . . .	187, 189		648, 651, 654, 693, 715, 849, 853
<code>\AtBegShi@PatchCrop</code> 385, 466, 469, 471		<code>\endinput</code> . . . . .	65, 378
<code>\AtBegShi@PatchEveryshi</code> . . . . .		<code>\endpicture</code> . . . . .	340
	473, 535, 538, 540	<code>\errmessage</code> . . . . .	147
<code>\AtBegShi@PatchMemoir</code> . . . . .			
	542, 631, 634, 636	<b>F</b>	
<code>\AtBegShi@Shipout</code> . . . . .	173, 295	<code>\fill</code> . . . . .	25
<code>\AtBegShi@Test</code> . . . . .	178, 182		
<code>\AtBegShi@vorigin</code> . . . .	312, 315, 363	<b>G</b>	
		<code>\g@addto@macro</code> . . . . .	272
<b>B</b>		<code>\gdef</code> . . . . .	200, 242, 243, 244, 296
<code>\baselineskip</code> . . . . .	231, 400, 444	<code>\GPTorg@shipout</code> . . . . .	
<code>\begin</code> . . . . .	11, 851		419, 420, 494, 495, 587, 588
<code>\body</code> . . . . .	661, 665		
<code>\box</code> . . . . .	205, 211, 371, 402, 446,	<b>H</b>	
	455, 526, 562, 617, 621, 728,	<code>\hbox</code> . . . . .	175,
	761, 762, 764, 765, 776, 833, 853		346, 359, 755, 767, 768, 824, 833
<b>C</b>		<code>\hrule</code> . . . . .	771, 832, 834
<code>\catcode</code> . . . . .	38, 39, 40, 41, 42,	<code>\hspace</code> . . . . .	843, 846
	53, 54, 55, 69, 70, 71, 72, 73, 74,	<code>\hss</code> . . . . .	828
	75, 76, 77, 78, 79, 97, 98, 101,	<code>\ht</code> . . . . .	228, 353, 364
	102, 103, 104, 108, 109, 110,		
	111, 115, 117, 167, 168, 170,	<b>I</b>	
	171, 215, 216, 641, 642, 643,	<code>\ifAtBegShi@Discarded</code> .	159, 201, 278
	644, 679, 688, 701, 702, 703,	<code>\ifcase</code> . . . . .	44, 138, 318, 571
	704, 705, 706, 707, 708, 709, 710	<code>\ifdim</code> . . . . .	183
<code>\circle</code> . . . . .	8	<code>\ifnum</code> . . . .	184, 435, 517, 603, 681, 689
<code>\color</code> . . . . .	16, 826	<code>\ifpdf</code> . . . . .	310
<code>\colorbox</code> . . . . .	844	<code>\ifvbox</code> . . . . .	614
<code>\copy</code> .	234, 235, 355, 754, 756, 758, 759	<code>\ifvoid</code> . . . . .	192, 281,
<code>\count</code> . . . . .	733, 735, 736		392, 449, 480, 524, 549, 561, 610
<code>\count@</code> . . . . .	646, 675,	<code>\ifx</code> . . . . .	45, 47, 56,
	679, 681, 682, 686, 688, 689, 690		80, 88, 130, 136, 138, 141, 166,
			220, 264, 299, 305, 319, 322,
			333, 376, 409, 410, 411, 413,

415, 417, 419, 421, 423, 426, 486, 488, 490, 492, 494, 496, 498, 501, 503, 508, 570, 571, 574, 581, 583, 585, 587, 589, 591, 594, 645, 648, 651, 654, 693	\repeat . . . . . 660, 672, 683, 691
\ignorespaces . . . . . 337, 348	\RequirePackage . . . . . 133, 308, 816
\immediate . . . . . 58, 82, 720	\RestoreCatcodes . . 674, 677, 678, 712
\input . . . . . 131, 306, 694, 719	\rlap . . . . . 360
\iterate . . . . . 662, 664, 666	\rule . . . . . 827, 845
<b>K</b>	<b>S</b>
\kern . . . . . 176, 361, 363, 364, 823, 825	\section . . . . . 12, 854
<b>L</b>	\setbox . . . . . 175, 180, 205, 222, 230, 346, 359, 389, 432, 451, 477, 514, 526, 546, 600, 612, 730, 755, 776, 821
\lastkern . . . . . 183	\shipout . . . . . 294, 295, 413, 414, 488, 489, 581, 582, 745, 752, 756, 759, 762, 765, 768, 771, 780, 786, 793, 795, 809, 852, 853
\line . . . . . 17, 18	\space . . . . . 148, 804
\lineskip . . . . . 232, 398, 442	<b>T</b>
\lineskiplimit . . . . . 233, 399, 443	\Test . . . . . 696, 714
\LoadCommand . . . . . 694, 711	\testmsg . . . . . 722, 744, 751, 754, 758, 761, 764, 767, 770, 773, 785, 792, 794, 799, 804
\loop . . . . . 660, 676, 687	\THBorg@shipout . . . . . . . . . . 421, 422, 496, 497, 589, 590
<b>M</b>	\the 101, 102, 103, 104, 115, 268, 679, 736
\makeatletter . . . . . 838	\TMP@EnsureCode 112, 119, 120, 121, 122, 123, 124, 125, 126, 127, 128
\makeatother . . . . . 841	\toks . . . . . 267, 268
\mem@oldshipout . . . . . 423, 424, 498, 499, 555, 562, 564, 620	\trimmarks . . . . . 556, 565, 613
\mem@shipi . . . . . 545, 570, 599, 602	<b>U</b>
\mem@shipii . . . 552, 571, 574, 607, 609	\unitlength . . . . . 335
\MessageBreak . . . . . 259, 283	\unvbox . . . . . 557, 566, 615
\msg . . . . . 720, 721, 723, 724, 736, 740, 783, 787, 790, 796, 806	\usepackage . . . . . 3, 4, 5, 819, 837, 850
<b>N</b>	<b>V</b>
\NeedsTeXFormat . . . . . 814	\vbox . . . . . 222, 230, 362, 405, 451, 555, 564, 612, 770, 771, 821, 822
\newbox . . . . . 727	\vfill . . . . . 801
\newif . . . . . 159	\void . . . . . 728, 730, 752, 786, 793
\newpage . . . . . 13, 22, 27, 32	\VoidBox . . . . . 774, 779, 781
\next . . . . . 666, 668, 670	\voidbox . . . . . 727, 728, 730
\null . . . . . 744, 745, 773, 780, 794, 795, 804, 809, 852	\vrule . . . . . 755, 768, 833
\number . . . . . 177, 430, 512, 598	\vspace . . . . . 25
<b>P</b>	\vss . . . . . 368, 830
\p@ . . . . . 176	<b>W</b>
\PackageInfo . . . . . 61	\wd . . . . . 227
\pagecolor . . . . . 817	\write . . . . . 58, 82, 720
\paperheight . . . . . 8, 17, 18, 845	<b>X</b>
\paperwidth . . . . . 8, 17, 18, 843, 846	\X . . . . . 215
\par . . . . . 24	\x 43, 45, 47, 57, 61, 63, 81, 86, 93, 99, 107
\pdfhorigin . . . . . 311	<b>Z</b>
\pdfvorigin . . . . . 312	\z@ . . . 144, 162, 183, 222, 227, 228, 229, 233, 234, 267, 268, 398, 399, 400, 442, 443, 444, 572, 575
\picture . . . . . 331	\z@skip . . . . . 231, 232
\protect . . . . . 195, 207, 210	
\ProvidesFile . . . . . 815	
\ProvidesPackage . . . . . 94	
\put . . . . . 8, 17, 18	
<b>R</b>	
\RangeCatcodeInvalid . . . . . . . . . . 685, 697, 698, 699, 700	