

# The atbegshi package

Heiko Oberdiek  
<oberdiek@uni-freiburg.de>

2007/04/19 v1.2

## 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>
<b>2</b>	<b>Method of <code>\shipout</code> overloading</b>	<b>3</b>
2.1	<code>\shipout</code> . . . . .	3
2.2	<code>\afterassignment</code> . . . . .	3
2.3	Test for direct or indirect boxes . . . . .	4
2.3.1	With $\varepsilon$ -TeX . . . . .	4
2.3.2	Without $\varepsilon$ -TeX . . . . .	4
2.3.3	<code>\lastkern</code> method . . . . .	5
2.4	Output . . . . .	6
2.5	Separate box register . . . . .	6
2.6	Summary . . . . .	6
2.6.1	With $\varepsilon$ -TeX . . . . .	6
2.6.2	Without $\varepsilon$ -TeX . . . . .	7
2.6.3	<code>\lastkern</code> method . . . . .	7
<b>3</b>	<b>Implementation</b>	<b>8</b>
3.1	Catcodes . . . . .	8
3.2	Reload check and package identification . . . . .	9
3.3	Preparations . . . . .	9
3.4	Patches . . . . .	13
3.4.1	Package <code>crop</code> . . . . .	14
3.4.2	Package <code>everyshi</code> . . . . .	15
3.4.3	Class <code>memoir</code> . . . . .	16
<b>4</b>	<b>Test</b>	<b>18</b>
<b>5</b>	<b>Installation</b>	<b>20</b>
5.1	Download . . . . .	20
5.2	Bundle installation . . . . .	21
5.3	Package installation . . . . .	21
5.4	Refresh file name databases . . . . .	21
5.5	Some details for the interested . . . . .	21
<b>6</b>	<b>History</b>	<b>22</b>
	[2007/04/17 v1.0] . . . . .	22
	[2007/04/18 v1.1] . . . . .	22
	[2007/04/19 v1.2] . . . . .	22

# 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  $\text{\LaTeX}$ '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`.

## 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  $\text{\TeX}$  would put `\@test` after the first of such an assignment or after the box specification if the box lacks of assignments. Therefore  $\text{\TeX}$  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 $\varepsilon\text{-TeX}$

With the  $\varepsilon\text{-TeX}$ '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 $\varepsilon\text{-TeX}$

Life becomes complicate without  $\varepsilon\text{-TeX}$ . 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.  $\text{\TeX}$  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 lose 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

```

```

        \@output
    \else
        \aftergroup\@output
    \fi

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

```

### 2.6.2 Without $\varepsilon$ -TeX

And for TeX without  $\varepsilon$ -TeX:

```

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

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

\@test ->
\ifvoid\mybox
    \aftergroup\@output
\else
    \@output
\fi

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

```

### 2.6.3 \lastkern method

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

```

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

\shipout ->
\begingroup

```

```

\setbox\mybox=\hbox\bgroup
\kern1pt
\afterassignment\@test
\setbox\mybox=

\@test ->
\ifdim\lastkern=0pt
\aftergroup\egroup
\aftergroup\endgroup
\aftergroup\@output
\else
\egroup
\endgroup
\@output
\fi

\@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

```

### 3 Implementation

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

```
1 (*package)
```

#### 3.1 Catcodes

```

2 \expandafter\edef\csname AtBegShi@End\endcsname{%
3   \catcode40 \the\catcode40\relax % (
4   \catcode41 \the\catcode41\relax % )
5   \catcode44 \the\catcode44\relax % ,
6   \catcode45 \the\catcode45\relax % -
7   \catcode46 \the\catcode46\relax % .
8   \catcode58 \the\catcode58\relax % :
9   \catcode61 \the\catcode61\relax % =
10  \catcode64 \the\catcode64\relax % @
11  \catcode94 \the\catcode94\relax % ^
12  \catcode96 \the\catcode96\relax % '
13  \noexpand\endinput
14 }
15 \catcode40 12 % (
16 \catcode41 12 % )
17 \catcode44 12 % ,
18 \catcode45 12 % -
19 \catcode46 12 % .
20 \catcode58 12 % :
21 \catcode61 12 % =
22 \catcode64 11 % @
23 \catcode94 7 % ^
24 \catcode96 12 % '

```

### 3.2 Reload check and package identification

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

```
25 \begingroup
26   \expandafter\let\expandafter\x\csname ver@atbegshi.sty\endcsname
27   \ifcase 0%
28     \ifx\x\relax % plain
29     \else
30       \ifx\x\empty % LaTeX
31       \else
32         1%
33       \fi
34     \fi
35   \else
36     \expandafter\ifx\csname PackageInfo\endcsname\relax
37     \def\x#1#2{%
38       \immediate\write-1{Package #1 Info: #2.}%
39     }%
40   \else
41     \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
42   \fi
43   \x{atbegshi}{The package is already loaded}%
44 \endgroup
45 \expandafter\endinput
46 \fi
47 \endgroup
```

Package identification:

```
48 \begingroup
49   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
50   \def\x#1#2#3[#4]{\endgroup
51     \immediate\write-1{Package: #3 #4}%
52     \xdef#1{#4}%
53   }%
54   \else
55     \def\x#1#2[#3]{\endgroup
56       #2[#{#3}]%
57       \ifx#1\relax
58         \xdef#1{#3}%
59       \fi
60     }%
61   \fi
62 \expandafter\x\csname ver@atbegshi.sty\endcsname
63 \ProvidesPackage{atbegshi}%
64 [2007/04/19 v1.2 At begin shipout hook (HO)]
```

### 3.3 Preparations

\AtBegShi@Warning

```
65 \begingroup\expandafter\expandafter\expandafter\endgroup
66 \expandafter\ifx\csname PackageWarning\endcsname\relax
67   \def\AtBegShi@Warning#1{%
68     \begingroup
69     \newlinechar=10 %
70     \def@spaces{\space\space\space\space}%
71     \def\MessageBreak{%
72       ^^J(atbegshi)\@spaces\@spaces\@spaces\@spaces
73     }%
74     \immediate\write16{%
75       ^^JPackage atbegshi Warning: #1 %
76       on input line \the\inputlineno.^^J%
77     }%
```

```

78     \endgroup
79 }%
80 \else
81   \def\AtBegShi@Warning{%
82     \PackageWarning{atbegshi}%
83   }%
84 \fi

\AtBegShi@Info

85 \begingroup\expandafter\expandafter\expandafter\endgroup
86 \expandafter\ifx\csname PackageInfo\endcsname\relax
87   \def\AtBegShi@Info#1{%
88     \begingroup
89       \newlinechar=10 %
90       \def\@spaces{\space\space\space\space}%
91       \def\MessageBreak{%
92         ^^J(atbegshi) \@spaces\@spaces\@spaces
93       }%
94       \immediate\write-1{%
95         ^^JPackage atbegshi Info: #1.^^J%
96       }%
97     \endgroup
98   }%
99 \else
100   \def\AtBegShi@Info#1{%
101     \begingroup
102       \def\on@line{}%
103       \PackageInfo{atbegshi}{#1}%
104     \endgroup
105   }%
106 \fi

```

\AtBegShi@CheckDefinable

```

107 \begingroup\expandafter\expandafter\expandafter\endgroup
108 \expandafter\ifx\csname @ifdefinable\endcsname\relax
109   \def\AtBegShi@CheckDefinable#1{%
110     \ifcase\ifx#1\relax
111       \@ne
112     \else
113       \ifx#1\@undefined
114         \@ne
115       \else
116         \z@
117       \fi
118     \fi
119     \errmessage{%
120       Package atbegshi: \string#1\space
121       is already defined%
122     }%
123   \endgroup
124 \fi
125 }%
126 \else
127   \def\AtBegShi@CheckDefinable#1{%
128     \@ifdefinable{#1}{}%
129   }%
130 \fi

```

131 \newif\ifAtBegShi@Discarded

\AtBeginShipoutDiscard

132 \AtBegShi@CheckDefinable\AtBeginShipoutDiscard

```

133 \def\AtBeginShipoutDiscard{%
134   \deadcycles=\z@
135   \global\AtBegShi@Discardedtrue
136 }

137 \begingroup\expandafter\expandafter\expandafter\endgroup
138 \expandafter\ifx\csname currentgrouplevel\endcsname\relax
139   \catcode'X=9 % ignore
140   \catcode'E=14 % comment
141 \else
142   \catcode'X=14 % comment
143   \catcode'E=9 % ignore
144 \fi

```

\AtBegShi@Shipout

```

145 \def\AtBegShi@Shipout{%
146 X \begingroup
147 X \setbox\AtBeginShipoutBox=\hbox\bgroup
148 X \kern\p@
149 E \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
150 \afterassignment\AtBegShi@Test
151 X \global
152 \setbox\AtBeginShipoutBox=%
153 }

```

\AtBegShi@Test

```

154 \def\AtBegShi@Test{%
155 X \ifdim\lastkern=\z@
156 X \aftergroup\egroup
157 X \aftergroup\endgroup
158 X \expandafter\aftergroup
159 X \else
160 X \egroup
161 X \endgroup
162 E \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
163 E \else
164 E \expandafter\aftergroup
165 \fi
166 \AtBegShi@Output
167 }

168 \catcode'\X=11 %
169 \catcode'\E=11 %

```

\AtBegShi@Output

```

170 \def\AtBegShi@Output{%
171 \ifvoid\AtBeginShipoutBox
172 \AtBegShi@Warning{Ignoring void shipout box}%
173 \else
174 \let\AtBegShi@OrgProtect\protect
175 \csname set@typeset@protect\endcsname
176 \global\AtBegShi@Discardedfalse
177 \AtBegShi@Hook
178 \AtBegShi@HookNext
179 \gdef\AtBegShi@HookNext{}%
180 \ifAtBegShi@Discarded
181 \AtBegShi@Info{Shipout page discarded}%
182 \global\AtBegShi@Discardedfalse
183 \begingroup
184 \setbox\AtBeginShipoutBox\box\AtBeginShipoutBox
185 \endgroup
186 \let\protect\AtBegShi@OrgProtect

```

```

187     \else
188         \AtBegShi@First
189         \let\protect\AtBegShi@OrgProtect
190         \AtBegShi@OrgShipout\box\AtBeginShipoutBox
191     \fi
192 \fi
193 }

\AtBegShi@First
194 \def\AtBegShi@First{%
195     \begingroup
196     \def\@empty{%
197         \ifx\AtBegShi@HookFirst\@empty
198             \else
199                 \setbox\z@=\vbox{%
200                     \begingroup
201                     \AtBegShi@HookFirst
202                     \endgroup
203                 }%
204                 \wd\z@=\z@
205                 \ht\z@=\z@
206                 \dp\z@=\z@
207                 \global\setbox\AtBeginShipoutBox=\vbox{%
208                     \baselineskip\z@skip
209                     \lineskip\z@skip
210                     \lineskiplimit\z@
211                     \copy\z@
212                     \copy\AtBeginShipoutBox
213                 }%
214             \fi
215             \global\let\AtBegShi@First\@empty
216             \global\let\AtBeginShipoutFirst\AtBegShi@FirstDisabled
217         \endgroup
218     }

\AtBegShi@Hook
219 \gdef\AtBegShi@Hook{}

\AtBegShi@HookNext
220 \gdef\AtBegShi@HookNext{}

\AtBegShi@HookFirst
221 \gdef\AtBegShi@HookFirst{}

\AtBeginShipout
222 \AtBegShi@CheckDefinable\AtBeginShipout
223 \def\AtBeginShipout{%
224     \AtBegShi@AddHook\AtBegShi@Hook
225 }

\AtBeginShipoutNext
226 \AtBegShi@CheckDefinable\AtBeginShipoutNext
227 \def\AtBeginShipoutNext{%
228     \AtBegShi@AddHook\AtBegShi@HookNext
229 }

\AtBeginShipoutFirst
230 \AtBegShi@CheckDefinable\AtBeginShipoutFirst
231 \def\AtBeginShipoutFirst{%
232     \AtBegShi@AddTo\AtBegShi@HookFirst
233 }

```

```

\AtBegShi@FirstDisabled
234 \long\def\AtBegShi@FirstDisabled#1{%
235   \AtBegShi@Warning{%
236     First page is already shipped out, ignoring\MessageBreak
237     \string\AtBeginShipoutFirst
238   }%
239 }

\AtBegShi@AddTo
240 \begingroup\expandafter\expandafter\expandafter\endgroup
241 \expandafter\ifx\csname g@addto@macro\endcsname\relax
242   \long\def\AtBegShi@AddTo#1#2{%
243     \begingroup
244       \toks\z@\expandafter{#1#2}%
245       \xdef#1{\the\toks\z@}%
246     \endgroup
247   }%
248 \else
249   \let\AtBegShi@AddTo\g@addto@macro
250 \fi

\AtBegShi@AddHook
251 \long\def\AtBegShi@AddHook#1#2{%
252   \AtBegShi@AddTo#1{\AtBegShi@Item{#2}}%
253 }

\AtBegShi@Item
254 \long\def\AtBegShi@Item#1{%
255   \ifAtBegShi@Discarded
256   \else
257     #1%
258     \ifvoid\AtBeginShipoutBox
259       \AtBegShi@Warning{%
260         Shipout box was voided by hook,\MessageBreak
261         ignoring shipout box%
262       }%
263       \AtBeginShipoutDiscard
264     \fi
265   \fi
266 }

\AtBeginShipoutInit
267 \AtBegShi@CheckDefinable\AtBeginShipoutInit
268 \def\AtBeginShipoutInit{%
269   \csname newbox\endcsname\AtBeginShipoutBox
270   \AtBegShi@CheckDefinable\AtBegShi@OrgShipout
271   \global\let\AtBegShi@OrgShipout\shipout
272   \global\let\shipout\AtBegShi@Shipout
273   \gdef\AtBeginShipoutInit{}%
274 }

275 \begingroup\expandafter\expandafter\expandafter\endgroup
276 \expandafter\ifx\csname AtBeginDocument\endcsname\relax
277   \AtBeginShipoutInit
278 \else
279   \AtBeginDocument{\AtBeginShipoutInit}%
280 \fi

```

### 3.4 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.

```

281 \def\AtBegShi@AbortIfUndefined#1{%
282   \begingroup\expandafter\expandafter\expandafter\endgroup
283   \expandafter\ifx\csname#1\endcsname\relax
284     \expandafter\AtBegShi@End
285   \fi
286 }
287 \AtBegShi@AbortIfUndefined{currentgrouplevel}
288 \AtBegShi@AbortIfUndefined{AtBeginDocument}
289 \AtBegShi@AbortIfUndefined{@ifpackageloaded}
290 \AtBegShi@AbortIfUndefined{@ifclassloaded}

```

### 3.4.1 Package crop

Fix of method and box.

```

291 \def\AtBegShi@PatchCrop{%
292   \begingroup
293     \def\AtBegShi@Crop@shipout{%
294       \afterassignment\CROP@ship
295       \setbox\@cclv=%
296     }%
297     \def\AtBegShi@Crop@ship{%
298       \ifvoid\@cclv
299         \expandafter\aftergroup
300       \fi
301       \CROP@@ship
302     }%
303     \def\AtBegShi@Crop@shiplist{%
304       \lineskip\z@
305       \lineskiplimit\z@
306       \baselineskip\z@
307       \CROP@kernel
308       \box\@cclv
309     }%
310     \def\AtBegShi@Crop@@ship{%
311       \CROP@shipout\vbox{%
312         \CROP@shiplist
313       }%
314     }%
315     \ifx\AtBegShi@Crop@ship\CROP@ship
316       \ifx\AtBegShi@Crop@shiplist\CROP@shiplist
317         \ifx\AtBegShi@Crop@@ship\CROP@@ship
318           \let\AtBegShi@found\relax
319           \ifx\shipout\AtBegShi@Crop@shipout
320             \def\AtBegShi@found{\shipout}%
321           \else\ifx\AtBegShi@OrgShipout\AtBegShi@Crop@shipout
322             \def\AtBegShi@found{\AtBegShi@OrgShipout}%
323           \else\ifx\@EveryShipout@Org@Shipout\AtBegShi@Crop@shipout
324             \def\AtBegShi@found{\@EveryShipout@Org@Shipout}%
325           \else\ifx\GPTorg@shipout\AtBegShi@Crop@shipout
326             \def\AtBegShi@found{\GPTorg@shipout}%
327           \else\ifx\THBorg@shipout\AtBegShi@Crop@shipout
328             \def\AtBegShi@found{\THBorg@shipout}%
329           \else\ifx\mem@oldshipout\AtBegShi@Crop@shipout
330             \def\AtBegShi@found{\mem@oldshipout}%
331           \fi\fi\fi\fi\fi
332           \ifx\AtBegShi@found\relax
333             \else
334               \expandafter\endgroup
335               \expandafter\def\AtBegShi@found{%
336                 \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
337                 \afterassignment\CROP@ship
338                 \setbox\AtBeginShipoutBox=%

```

```

339         }%
340         \def\CROP@ship{%
341             \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
342             \else
343                 \expandafter\aftergroup
344             \fi
345             \CROP@@ship
346         }%
347         \def\CROP@shiplist{%
348             \lineskip\z@
349             \lineskiplimit\z@
350             \baselineskip\z@
351             \CROP@kernel
352             \box\AtBeginShipoutBox
353         }%
354         \def\CROP@@ship{%
355             \ifvoid\AtBeginShipoutBox
356             \else
357                 \setbox\AtBeginShipoutBox=\vbox{%
358                     \CROP@shiplist
359                 }%
360                 \expandafter\CROP@shipout
361                 \expandafter\box
362                 \expandafter\AtBeginShipoutBox
363             \fi
364         }%
365         \AtBegShi@Info{Package 'crop' patched}%
366         \begingroup
367         \fi
368     \fi
369 \fi
370 \fi
371 \endgroup
372 \let\AtBegShi@PatchCrop\relax
373 }
374 \ifpackageloaded{crop}{%
375     \AtBegShi@PatchCrop
376 }{%
377     \AtBeginDocument{\AtBegShi@PatchCrop}%
378 }

```

### 3.4.2 Package everyshi

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

```

379 \def\AtBegShi@PatchEveryshi{%
380     \begingroup
381     \long\def\AtBegShi@Everyshi@shipout{%
382         \afterassignment\@EveryShipout@Test
383         \global\setbox\@cclv= %
384     }%
385     \long\def\AtBegShi@Everyshi@Test{%
386         \ifvoid\@cclv\relax
387             \aftergroup\@EveryShipout@Output
388         \else
389             \@EveryShipout@Output
390         \fi
391     }%
392     \ifx\AtBegShi@Everyshi@Test\@EveryShipout@Test
393         \let\AtBegShi@found\relax
394         \ifx\shipout\AtBegShi@Everyshi@shipout
395             \def\AtBegShi@found{\shipout}%
396         \else\ifx\AtBegShi@OrgShipout\AtBegShi@Everyshi@shipout

```

```

397     \def\AtBegShi@found{\AtBegShi@OrgShipout}%
398 \else\ifx\CROP@shipout\AtBegShi@Everyshi@shipout
399     \def\AtBegShi@found{\CROP@shipout}%
400 \else\ifx\GPTorg@shipout\AtBegShi@Everyshi@shipout
401     \def\AtBegShi@found{\GPTorg@shipout}%
402 \else\ifx\THBorg@shipout\AtBegShi@Everyshi@shipout
403     \def\AtBegShi@found{\THBorg@shipout}%
404 \else\ifx\mem@oldshipout\AtBegShi@Everyshi@shipout
405     \def\AtBegShi@found{\mem@oldshipout}%
406 \else
407     \expandafter\ifx\csname @EveryShipout@Org@Shipout\endcsname
408         \relax
409         \ifx\@EveryShipout@Shipout\AtBegShi@Everyshi@shipout
410             \def\AtBegShi@found{\@EveryShipout@Shipout}%
411         \fi
412     \fi
413 \fi\fi\fi\fi\fi\fi
414 \ifx\AtBegShi@found\relax
415 \else
416     \expandafter\endgroup
417     \expandafter\def\AtBegShi@found{%
418         \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
419         \afterassignment\@EveryShipout@Test
420         \setbox\AtBeginShipoutBox=%
421     }%
422     \def\@EveryShipout@Test{%
423         \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
424         \else
425             \expandafter\aftergroup
426             \fi
427         \AtBegShi@Everyshi@Output
428     }%
429     \def\AtBegShi@Everyshi@Output{%
430         \ifvoid\AtBeginShipoutBox
431         \else
432             \global\setbox\@cclv\box\AtBeginShipoutBox
433             \expandafter\@EveryShipout@Output
434         \fi
435     }%
436     \AtBegShi@Info{Package 'everyshi' patched}%
437     \begingroup
438     \fi
439 \fi
440 \endgroup
441 \let\AtBegShi@PatchEveryshi\relax
442 }
443 \@ifpackageloaded{everyshi}{%
444     \AtBegShi@PatchEveryshi
445 }{%
446     \AtBeginDocument{\AtBegShi@PatchEveryshi}%
447 }

```

### 3.4.3 Class memoir

Fix of method and box.

```

448 \def\AtBegShi@PatchMemoir{%
449     \begingroup
450     \def\AtBegShi@Memoir@shipout{%
451         \afterassignment\mem@shipi
452         \setbox\@cclv=%
453     }%
454     \def\AtBegShi@Memoir@shipi{%

```

```

455     \ifvoid\@cclv
456     \expandafter\aftergroup
457     \fi
458     \mem@shipii
459 }%
460 \def\AtBegShi@Memoir@shipiiA{%
461     \mem@oldshipout\ vbox{%
462         \trimmarks
463         \unvbox\@cclv
464     }%
465 }%
466 \def\AtBegShi@Memoir@shipiiB{%
467     \ifvoid\@cclv
468     \mem@oldshipout\ box\@cclv
469     \else
470     \mem@oldshipout\ vbox{%
471         \trimmarks
472         \unvbox\@cclv
473     }%
474     \fi
475 }%
476 \ifx\AtBegShi@Memoir@shipi\mem@shipi
477     \ifcase\ifx\AtBegShi@Memoir@shipiiA\mem@shipii
478         \z@
479     \else
480         \ifx\AtBegShi@Memoir@shipiiB\mem@shipii
481             \z@
482         \else
483             \@ne
484         \fi
485     \fi
486     \let\AtBegShi@found\relax
487     \ifx\shipout\AtBegShi@Memoir@shipout
488         \def\AtBegShi@found{\shipout}%
489     \else\ifx\AtBegShi@OrgShipout\AtBegShi@Memoir@shipout
490         \def\AtBegShi@found{\AtBegShi@OrgShipout}%
491     \else\ifx\CROP@shipout\AtBegShi@Memoir@shipout
492         \def\AtBegShi@found{\CROP@shipout}%
493     \else\ifx\GPTorg@shipout\AtBegShi@Memoir@shipout
494         \def\AtBegShi@found{\GPTorg@shipout}%
495     \else\ifx\THBorg@shipout\AtBegShi@Memoir@shipout
496         \def\AtBegShi@found{\THBorg@shipout}%
497     \else\ifx\@EveryShipout@Org@Shipout\AtBegShi@Memoir@shipout
498         \def\AtBegShi@found{\@EveryShipout@Org@Shipout}%
499     \fi\fi\fi\fi\fi\fi
500     \ifx\AtBegShi@found\relax
501     \else
502     \expandafter\endgroup
503     \expandafter\def\AtBegShi@found{%
504         \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
505         \afterassignment\mem@shipi
506         \setbox\AtBeginShipoutBox=%
507     }%
508     \def\mem@shipi{%
509         \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
510         \else
511             \expandafter\aftergroup
512         \fi
513         \mem@shipii
514     }%
515     \def\mem@shipii{%
516         \ifvoid\AtBeginShipoutBox

```

```

517         \else
518         \setbox\AtBeginShipoutBox=\vbox{%
519         \trimmarks
520         \ifvbox\AtBeginShipoutBox
521         \unvbox\AtBeginShipoutBox
522         \else
523         \box\AtBeginShipoutBox
524         \fi
525         }%
526         \expandafter\mem@oldshipout
527         \expandafter\box
528         \expandafter\AtBeginShipoutBox
529         \fi
530     }%
531     \AtBegShi@Info{Class 'memoir' patched}%
532     \begingroup
533     \fi
534     \fi
535     \fi
536 \endgroup
537 \let\AtBegShi@PatchMemoir\relax
538 }
539 \ifclassloaded{memoir}{%
540     \AtBegShi@PatchMemoir
541 }{%
542     \AtBeginDocument{\AtBegShi@PatchMemoir}%
543 }
544 \AtBegShi@End
545 \end{package}

```

## 4 Test

```

546 (*test1)
547 \input atbegshi.sty\relax
548 \def\msg#{\immediate\write16}
549 \msg{File: atbegshi-test1.tex 2007/04/19 v1.2 Test file for plain-TeX}
550 \def\testmsg#1#2{%
551     \msg{}%
552     \msg{*** Test with box (#1), expected page output [#2]}%
553 }
554
555 \newbox\voidbox
556 \def\void{\box\voidbox}
557 \begingroup
558     \setbox\voidbox=\void
559 \endgroup
560
561 \count0=0\relax
562 \AtBeginShipout{%
563     \global\advance\count0 by 1\relax
564     \msg{* Inside \string\AtBeginShipout: [\the\count0]}%
565 }
566
567 \AtBeginShipoutFirst{%
568     \msg{* Inside \string\AtBeginShipoutFirst}%
569     Hello World%
570 }
571
572 \testmsg{\string\null}{1}
573 \shipout\null
574

```

```

575 \AtBeginShipoutFirst{%
576   This is too late%
577 }
578
579 \testmsg{void}{}
580 \shipout\void
581
582 \testmsg{\string\copy255 (not void)}{2}
583 \setbox255\hbox{\vrule height 10bp width 10bp}
584 \shipout\copy255 %
585
586 \testmsg{\string\copy255 (again)}{3}
587 \shipout\copy255 %
588
589 \testmsg{\string\box255}{4}
590 \shipout\box255 %
591
592 \testmsg{\string\box255 (again)}{}
593 \shipout\box255 %
594
595 \testmsg{\string\hbox}{5}
596 \shipout\hbox{\vrule height 5bp width 20bp}
597
598 \testmsg{\string\vbox}{6}
599 \shipout\vbox{\hrule height 20bp width 5bp}
600
601 \testmsg{\string\null, voided by hook}{}
602 \def\VoidBox{%
603   \begingroup
604   \setbox\AtBeginShipoutBox=\box\AtBeginShipoutBox
605   \endgroup
606 }
607 \AtBeginShipout{\VoidBox}
608 \shipout\null
609 \def\VoidBox{}
610
611 \msg{*** \string\begingroup}
612 \begingroup
613   \testmsg{void}{}%
614   \shipout\void
615 \msg{*** \string\endgroup}
616 \endgroup
617
618 \msg{*** \string\begingroup}
619 \begingroup
620   \testmsg{void}{}%
621   \shipout\void
622   \testmsg{\string\null}{8}%
623   \shipout\null
624 \msg{*** \string\endgroup}
625 \endgroup
626
627 \testmsg{output routine}{9}
628 Hello World
629 \vfill
630 \eject
631
632 \testmsg{\string\null\space(discarded)}{}
633 \AtBeginShipout{%
634   \msg{* Inside \string\AtBeginShipout: DISCARD}%
635   \AtBeginShipoutDiscard
636 }

```

```

637 \shipout\null
638
639 \end
640 </test1>

641 <*test2>
642 \NeedsTeXFormat{LaTeX2e}
643 \ProvidesFile{atbegshi-test2.tex}[2007/04/19 v1.2 Test file for LaTeX]
644 \RequirePackage{color}
645 \pagecolor{yellow}
646 \documentclass[a5paper,showtrims]{memoir}
647 \usepackage{atbegshi}
648 \AtBeginShipout{%
649   \setbox\AtBeginShipoutBox=\vbox{%
650     \vbox to Opt{%
651       \kern-1.5in %
652       \hbox to Opt{%
653         \kern-1.5in %
654         \color{blue}%
655         \rule{1in}{1in}%
656         \hss
657       }%
658       \vss
659     }%
660     \hrule
661     \hbox{\vrule\box\AtBeginShipoutBox\vrule}%
662     \hrule
663   }%
664 }
665 \usepackage{eso-pic}
666 \makeatletter
667 \@EveryShipout@Init
668 \let\@EveryShipout@Init\relax
669 \makeatother
670 \AddToShipoutPicture{%
671   \hspace{.52\paperwidth}%
672   \colorbox{cyan}{%
673     \rule{0mm}{\paperheight}%
674     \hspace{.48\paperwidth}%
675   }%
676 }
677 \usepackage[color=red,cross,a4,center]{crop}
678 \begin{document}
679 \shipout\null
680 \shipout\box\csname voidb@x\endcsname
681 \section{Hello World}
682 \end{document}
683 </test2>

```

## 5 Installation

### 5.1 Download

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

[CTAN:macros/latex/contrib/oberdiek/atbegshi.dtx](http://ftp.ctan.org/tex-archive/macros/latex/contrib/oberdiek/atbegshi.dtx) The source file.

[CTAN:macros/latex/contrib/oberdiek/atbegshi.pdf](http://ftp.ctan.org/tex-archive/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.

<sup>1</sup>[ftp://ftp.ctan.org/tex-archive/](http://ftp.ctan.org/tex-archive/)

CTAN:macros/latex/contrib/oberdiek/oberdiek-tds.zip

## 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-TeX:

```
tex atbegshi.dtx
```

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

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

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

## 5.4 Refresh file name databases

If your TeX distribution (TeX, MikTeX, ...) relies on file name databases, you must refresh these. For example, TeX users run texhash or mktexlsr.

## 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 L<sup>A</sup>T<sub>E</sub>X.** The .dtx chooses its action depending on the format:

**plain-TeX:** Run docstrip and extract the files.

**L<sup>A</sup>T<sub>E</sub>X:** Generate the documentation.

If you insist on using L<sup>A</sup>T<sub>E</sub>X for docstrip (really, docstrip does not need L<sup>A</sup>T<sub>E</sub>X), 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 pdfL<sup>A</sup>T<sub>E</sub>X:

```
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  $\epsilon$ -T<sub>E</sub>X 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.

## 7 Index

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

Symbols	
<code>\@EveryShipout@Init</code> . . . . .	667, 668
<code>\@EveryShipout@Org@Shipout</code> . . . . .	323, 324, 497, 498
<code>\@EveryShipout@Output</code> . . . . .	387, 389, 433
<code>\@EveryShipout@Shipout</code> . . . . .	409, 410
<code>\@EveryShipout@Test</code> . . . . .	382, 392, 419, 422
<code>\@cclv</code> . . . . .	295, 298, 308, 383, 386, 432, 452, 455, 463, 467, 468, 472
<code>\@empty</code> . . . . .	196, 197, 215
<code>\@ifclassloaded</code> . . . . .	539
<code>\@ifdefinable</code> . . . . .	128
<code>\@ifpackageloaded</code> . . . . .	374, 443
<code>\@one</code> . . . . .	111, 114, 483
<code>\@spaces</code> . . . . .	70, 72, 90, 92
<code>\@undefined</code> . . . . .	113
<b>A</b>	
<code>\AddToShipoutPicture</code> . . . . .	670
<code>\advance</code> . . . . .	563
<code>\afterassignment</code> . . . . .	150, 294, 337, 382, 419, 451, 505
<code>\aftergroup</code> . . . . .	156, 157, 158, 164, 299, 343, 387, 425, 456, 511
<code>\AtBeginDocument</code> . . . . .	279, 377, 446, 542
<code>\AtBeginShipout</code> . . . . .	2, 222, 562, 564, 607, 633, 634, 648
<code>\AtBeginShipoutBox</code> . . . . .	147, 152, 171, 184, 190, 207, 212, 258, 269, 338, 352, 355, 357, 362, 420, 430, 432, 506, 516, 518, 520, 521, 523, 528, 604, 649, 661
<code>\AtBeginShipoutDiscard</code> . . . . .	3, 132, 263, 635
<code>\AtBeginShipoutFirst</code> . . . . .	2, 216, 230, 237, 567, 568, 575
<code>\AtBeginShipoutInit</code> . . . . .	3, 267, 277, 279
<code>\AtBeginShipoutNext</code> . . . . .	2, 226
<code>\AtBegShi@AbortIfUndefined</code> . . . . .	281, 287, 288, 289, 290
<code>\AtBegShi@AddHook</code> . . . . .	224, 228, 251

\AtBegShi@AddTo	232, 240, 252	\CROP@kernel	307, 351
\AtBegShi@CheckDefinable	107, 132, 222, 226, 230, 267, 270	\CROP@ship	294, 315, 337, 340
\AtBegShi@Crop@@ship	310, 317	\CROP@shiplist	312, 316, 347, 358
\AtBegShi@Crop@ship	297, 315	\CROP@shipout	311, 360, 398, 399, 491, 492
\AtBegShi@Crop@shiplist	303, 316	\csname	2, 26, 36, 49, 62, 66, 86, 108, 138, 175, 241, 269, 276, 283, 407, 680
\AtBegShi@Crop@shipout	293, 319, 321, 323, 325, 327, 329	\currentgrouplevel	149, 162, 336, 341, 418, 423, 504, 509
\AtBegShi@Discardedfalse	176, 182	<b>D</b>	
\AtBegShi@Discardedtrue	135	\deadcycles	134
\AtBegShi@End	284, 544	\documentclass	646
\AtBegShi@Everyshi@Output	427, 429	\dp	206
\AtBegShi@Everyshi@shipout	381, 394, 396, 398, 400, 402, 404, 409	<b>E</b>	
\AtBegShi@Everyshi@Test	385, 392	\E	169
\AtBegShi@First	188, 194	\eject	630
\AtBegShi@FirstDisabled	216, 234	\empty	30
\AtBegShi@found	318, 320, 322, 324, 326, 328, 330, 332, 335, 393, 395, 397, 399, 401, 403, 405, 410, 414, 417, 486, 488, 490, 492, 494, 496, 498, 500, 503	\end	639, 682
\AtBegShi@GroupLevel	149, 162, 336, 341, 418, 423, 504, 509	\endcsname	2, 26, 36, 49, 62, 66, 86, 108, 138, 175, 241, 269, 276, 283, 407, 680
\AtBegShi@Hook	177, 219, 224	\endinginput	13, 45
\AtBegShi@HookFirst	197, 201, 221, 232	\errmessage	119
\AtBegShi@HookNext	178, 179, 220, 228	<b>G</b>	
\AtBegShi@Info	85, 181, 365, 436, 531	\g@addto@macro	249
\AtBegShi@Item	252, 254	\gdef	179, 219, 220, 221, 273
\AtBegShi@Memoir@shipi	454, 476	\GPTorg@shipout	325, 326, 400, 401, 493, 494
\AtBegShi@Memoir@shipiiA	460, 477	<b>H</b>	
\AtBegShi@Memoir@shipiiB	466, 480	\hbox	147, 583, 595, 596, 652, 661
\AtBegShi@Memoir@shipout	450, 487, 489, 491, 493, 495, 497	\hrule	599, 660, 662
\AtBegShi@OrgProtect	174, 186, 189	\hspace	671, 674
\AtBegShi@OrgShipout	190, 270, 271, 321, 322, 396, 397, 489, 490	\hss	656
\AtBegShi@Output	166, 170	\ht	205
\AtBegShi@PatchCrop	291, 372, 375, 377	<b>I</b>	
\AtBegShi@PatchEveryshi	379, 441, 444, 446	\ifAtBegShi@Discarded	131, 180, 255
\AtBegShi@PatchMemoir	448, 537, 540, 542	\ifcase	27, 110, 477
\AtBegShi@Shipout	145, 272	\ifdim	155
\AtBegShi@Test	150, 154	\ifnum	162, 341, 423, 509
\AtBegShi@Warning	65, 172, 235, 259	\ifvbox	520
<b>B</b>		\ifvoid	171, 258, 298, 355, 386, 430, 455, 467, 516
\baselineskip	208, 306, 350	\ifx	28, 30, 36, 49, 57, 66, 86, 108, 110, 113, 138, 197, 241, 276, 283, 315, 316, 317, 319, 321, 323, 325, 327, 329, 332, 392, 394, 396, 398, 400, 402, 404, 407, 409, 414, 476, 477, 480, 487, 489, 491, 493, 495, 497, 500
\begin	678	\immediate	38, 51, 74, 94, 548
\box	184, 190, 308, 352, 361, 432, 468, 523, 527, 556, 589, 590, 592, 593, 604, 661, 680	\input	547
<b>C</b>		\inputlineno	76
\catcode	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 139, 140, 142, 143, 168, 169	<b>K</b>	
\color	654	\kern	148, 651, 653
\colorbox	672	<b>L</b>	
\copy	211, 212, 582, 584, 586, 587	\lastkern	155
\count	561, 563, 564		
\CROP@@ship	301, 317, 345, 354		

<code>\lineskip</code> . . . . .	209, 304, 348	<code>\shipout</code> . . . . .	271, 272, 319, 320, 394, 395, 487, 488, 573, 580, 584, 587, 590, 593, 596, 599, 608, 614, 621, 623, 637, 679, 680
<code>\lineskiplimit</code> . . . . .	210, 305, 349	<code>\space</code> . . . . .	70, 90, 120, 632
M			
<code>\makeatletter</code> . . . . .	666		
<code>\makeatother</code> . . . . .	669		
<code>\mem@oldshipout</code> . . . . .	329, 330, 404, 405, 461, 468, 470, 526		
<code>\mem@shipi</code> . . . . .	451, 476, 505, 508		
<code>\mem@shipii</code> . . . . .	458, 477, 480, 513, 515		
<code>\MessageBreak</code> . . . . .	71, 91, 236, 260		
<code>\msg</code> . . . . .	548, 549, 551, 552, 564, 568, 611, 615, 618, 624, 634		
N			
<code>\NeedsTeXFormat</code> . . . . .	642		
<code>\newbox</code> . . . . .	555		
<code>\newif</code> . . . . .	131		
<code>\newlinechar</code> . . . . .	69, 89		
<code>\null</code> . . . . .	572, 573, 601, 608, 622, 623, 632, 637, 679		
<code>\number</code> . . . . .	149, 336, 418, 504		
O			
<code>\on@line</code> . . . . .	102		
P			
<code>\p@</code> . . . . .	148		
<code>\PackageInfo</code> . . . . .	41, 103		
<code>\PackageWarning</code> . . . . .	82		
<code>\pagecolor</code> . . . . .	645		
<code>\paperheight</code> . . . . .	673		
<code>\paperwidth</code> . . . . .	671, 674		
<code>\protect</code> . . . . .	174, 186, 189		
<code>\ProvidesFile</code> . . . . .	643		
<code>\ProvidesPackage</code> . . . . .	63		
R			
<code>\RequirePackage</code> . . . . .	644		
<code>\rule</code> . . . . .	655, 673		
S			
<code>\section</code> . . . . .	681		
<code>\setbox</code> . . . . .	147, 152, 184, 199, 207, 295, 338, 357, 383, 420, 432, 452, 506, 518, 558, 583, 604, 649		
T			
<code>\testmsg</code> . . . . .	550, 572, 579, 582, 586, 589, 592, 595, 598, 601, 613, 620, 622, 627, 632		
<code>\THBorg@shipout</code> . . . . .	327, 328, 402, 403, 495, 496		
<code>\the</code> . . . . .	3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 76, 245, 564		
<code>\toks</code> . . . . .	244, 245		
<code>\trimmarks</code> . . . . .	462, 471, 519		
U			
<code>\unvbox</code> . . . . .	463, 472, 521		
<code>\usepackage</code> . . . . .	647, 665, 677		
V			
<code>\vbox</code> . . . . .	199, 207, 311, 357, 461, 470, 518, 598, 599, 649, 650		
<code>\vfill</code> . . . . .	629		
<code>\void</code> . . . . .	556, 558, 580, 614, 621		
<code>\VoidBox</code> . . . . .	602, 607, 609		
<code>\voidbox</code> . . . . .	555, 556, 558		
<code>\vrule</code> . . . . .	583, 596, 661		
<code>\vss</code> . . . . .	658		
W			
<code>\wd</code> . . . . .	204		
<code>\write</code> . . . . .	38, 51, 74, 94, 548		
X			
<code>\X</code> . . . . .	168		
<code>\x</code> . . . . .	26, 28, 30, 37, 41, 43, 50, 55, 62		
Z			
<code>\z@</code> . . . . .	116, 134, 155, 199, 204, 205, 206, 210, 211, 244, 245, 304, 305, 306, 348, 349, 350, 478, 481		
<code>\z@skip</code> . . . . .	208, 209		