

# The atbegshi package

Heiko Oberdiek  
<oberdiek@uni-freiburg.de>

2007/04/27 v1.4

## 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  $\LaTeX$  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, traditional way . . . . .	7
2.6.3	<code>\lastkern</code> method . . . . .	7
<b>3</b>	<b>Implementation</b>	<b>8</b>
3.1	Reload check and package identification . . . . .	8
3.2	Catcodes . . . . .	9
3.3	Preparations . . . . .	9
3.4	Patches . . . . .	13
3.4.1	Package <code>crop</code> . . . . .	13
3.4.2	Package <code>everyshi</code> . . . . .	15
3.4.3	Class <code>memoir</code> . . . . .	16
<b>4</b>	<b>Test</b>	<b>17</b>
<b>5</b>	<b>Installation</b>	<b>20</b>
5.1	Download . . . . .	20
5.2	Bundle installation . . . . .	20
5.3	Package installation . . . . .	20
5.4	Refresh file name databases . . . . .	21
5.5	Some details for the interested . . . . .	21

<b>6 History</b>	<b>21</b>
[2007/04/17 v1.0]	21
[2007/04/18 v1.1]	21
[2007/04/19 v1.2]	21
[2007/04/26 v1.3]	22
[2007/04/27 v1.4]	22
<b>7 Index</b>	<b>22</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 `ℒATEX` calls `\shipout` in `\@outputpage` (part of its output routine), the meaning of `\protect` is `\noexpand`. `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 `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 T<sub>E</sub>X would put \@test after the first of such an assignment or after the box specification if the box lacks of assignments. Therefore T<sub>E</sub>X 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 $\varepsilon$ -T<sub>E</sub>X

With the  $\varepsilon$ -T<sub>E</sub>X'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 $\varepsilon$ -T<sub>E</sub>X

Life becomes complicate without  $\varepsilon$ -T<sub>E</sub>X. 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. T<sub>E</sub>X only tells us a few properties:

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

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

Nevertheless we have two alternatives for an initial value:

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

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

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

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

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

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

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

### 2.3.3 `\lastkern` method

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

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

```

\aftergroup\endgroup
\aftergroup\@output
\else
\egroup
\endgroup
\@output
\fi

```

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

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

## 2.4 Output

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

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

```

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

```

## 2.5 Separate box register

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

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

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

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

## 2.6 Summary

### 2.6.1 With $\varepsilon$ -TeX

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

```

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

\shipout :=
\edef\saved@grouplevel{\number\currentgrouplevel}

```

```

\afterassignment\@test
\setbox\mybox=

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

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

```

## 2.6.2 Without $\varepsilon$ -T<sub>E</sub>X, traditional way

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

```

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

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

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

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

```

## 2.6.3 \lastkern method

And for T<sub>E</sub>X without  $\varepsilon$ -T<sub>E</sub>X using the \lastkern method:

```

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

```

```

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

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

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

```

### 3 Implementation

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

```
1 \langle *package \rangle
```

#### 3.1 Reload check and package identification

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

```

2 \begingroup
3 \catcode44 12 % ,
4 \catcode45 12 % -
5 \catcode46 12 % .
6 \catcode58 12 % :
7 \catcode64 11 % @
8 \expandafter\let\expandafter\x\csname ver@atbegshi.sty\endcsname
9 \ifcase 0%
10 \ifx\x\relax % plain
11 \else
12 \ifx\x\empty % LaTeX
13 \else
14 1%
15 \fi
16 \fi
17 \else
18 \expandafter\ifx\csname PackageInfo\endcsname\relax
19 \def\x#1#2{%
20 \immediate\write-1{Package #1 Info: #2.}%
21 }%
22 \else
23 \def\x#1#2{\PackageInfo{#1}{#2, stopped}}%
24 \fi
25 \x{atbegshi}{The package is already loaded}%

```



```

26     \endgroup
27     \expandafter\endinput
28 \fi
29 \endgroup
Package identification:
30 \begingroup
31   \catcode44 12 % ,
32   \catcode45 12 % -
33   \catcode46 12 % .
34   \catcode58 12 % :
35   \catcode64 11 % @
36   \expandafter\ifx\csname ProvidesPackage\endcsname\relax
37     \def\x#1#2#3[#4]{\endgroup
38       \immediate\write-1{Package: #3 #4}%
39       \xdef#1{#4}%
40     }%
41   \else
42     \def\x#1#2[#3]{\endgroup
43       #2[#{#3}]%
44       \ifx#1\relax
45         \xdef#1{#3}%
46       \fi
47     }%
48   \fi
49 \expandafter\x\csname ver@atbegshi.sty\endcsname
50 \ProvidesPackage{atbegshi}%
51 [2007/04/27 v1.4 At begin shipout hook (H0)]

```

### 3.2 Catcodes

```

52 \expandafter\edef\csname AtBegShi@End\endcsname{%
53   \catcode40 \the\catcode40\relax % (
54   \catcode41 \the\catcode41\relax % )
55   \catcode44 \the\catcode44\relax % ,
56   \catcode45 \the\catcode45\relax % -
57   \catcode46 \the\catcode46\relax % .
58   \catcode58 \the\catcode58\relax % :
59   \catcode61 \the\catcode61\relax % =
60   \catcode64 \the\catcode64\relax % @
61   \catcode94 \the\catcode94\relax % ^
62   \catcode96 \the\catcode96\relax % '
63   \noexpand\endinput
64 }
65 \catcode40 12 % (
66 \catcode41 12 % )
67 \catcode44 12 % ,
68 \catcode45 12 % -
69 \catcode46 12 % .
70 \catcode58 12 % :
71 \catcode61 12 % =
72 \catcode64 11 % @
73 \catcode94 7 % ^
74 \catcode96 12 % '

```

### 3.3 Preparations

\AtBegShi@Warning

```

75 \begingroup\expandafter\expandafter\expandafter\endgroup
76 \expandafter\ifx\csname RequirePackage\endcsname\relax
77   \input infwarerr.sty\relax
78 \else
79   \RequirePackage{infwarerr}%

```

```

80 \fi

\AtBegShi@CheckDefinable

81 \begingroup\expandafter\expandafter\expandafter\endgroup
82 \expandafter\ifx\csname @ifdefinable\endcsname\relax
83   \def\AtBegShi@CheckDefinable#1{%
84     \ifcase\ifx#1\relax
85       \@ne
86     \else
87       \ifx#1\@undefined
88         \@ne
89       \else
90         \z@
91       \fi
92     \fi
93     \errmessage{%
94       Package atbegshi: \string#1\space
95       is already defined%
96     }%
97   \endgroup
98 \fi
99 }%
100 \else
101   \def\AtBegShi@CheckDefinable#1{%
102     \@ifdefinable{#1}{}%
103   }%
104 \fi

105 \newif\ifAtBegShi@Discarded

\AtBeginShipoutDiscard

106 \AtBegShi@CheckDefinable\AtBeginShipoutDiscard
107 \def\AtBeginShipoutDiscard{%
108   \deadcycles=\z@
109   \global\AtBegShi@Discardedtrue
110 }

111 \begingroup\expandafter\expandafter\expandafter\endgroup
112 \expandafter\ifx\csname currentgrouplevel\endcsname\relax
113   \catcode'X=9 % ignore
114   \catcode'E=14 % comment
115 \else
116   \catcode'X=14 % comment
117   \catcode'E=9 % ignore
118 \fi

\AtBegShi@Shipout

119 \def\AtBegShi@Shipout{%
120 X \begingroup
121 X \setbox\AtBeginShipoutBox=\hbox\bgroup
122 X \kern\p@
123 E \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
124 \afterassignment\AtBegShi@Test
125 X \global
126 \setbox\AtBeginShipoutBox=%
127 }

\AtBegShi@Test

128 \def\AtBegShi@Test{%
129 X \ifdim\lastkern=\z@
130 E \ifnum\AtBegShi@GroupLevel<\currentgrouplevel
131   \expandafter\aftergroup

```

```

132 \fi
133 \AtBegShi@Output
134 }

\AtBegShi@Output
135 \def\AtBegShi@Output{%
136 X \egroup
137 X \endgroup
138 \ifvoid\AtBeginShipoutBox
139 \PackageWarning{atbegshi}{Ignoring void shipout box}%
140 \else
141 \let\AtBegShi@OrgProtect\protect
142 \csname set@typeset@protect\endcsname
143 \global\AtBegShi@Discardedfalse
144 \AtBegShi@Hook
145 \AtBegShi@HookNext
146 \gdef\AtBegShi@HookNext{%
147 \ifAtBegShi@Discarded
148 \PackageInfoNoLine{atbegshi}{Shipout page discarded}%
149 \global\AtBegShi@Discardedfalse
150 \begingroup
151 \setbox\AtBeginShipoutBox\box\AtBeginShipoutBox
152 \endgroup
153 \let\protect\AtBegShi@OrgProtect
154 \else
155 \AtBegShi@First
156 \let\protect\AtBegShi@OrgProtect
157 \AtBegShi@OrgShipout\box\AtBeginShipoutBox
158 \fi
159 \fi
160 }

161 \catcode'\X=11 %
162 \catcode'\E=11 %

\AtBegShi@First
163 \def\AtBegShi@First{%
164 \begingroup
165 \def\@empty{%
166 \ifx\AtBegShi@HookFirst\@empty
167 \else
168 \setbox\z@=\vbox{%
169 \begingroup
170 \AtBegShi@HookFirst
171 \endgroup
172 }%
173 \wd\z@=\z@
174 \ht\z@=\z@
175 \dp\z@=\z@
176 \global\setbox\AtBeginShipoutBox=\vbox{%
177 \baselineskip\z@skip
178 \lineskip\z@skip
179 \lineskiplimit\z@
180 \copy\z@
181 \copy\AtBeginShipoutBox
182 }%
183 \fi
184 \global\let\AtBegShi@First\@empty
185 \global\let\AtBeginShipoutFirst\AtBegShi@FirstDisabled
186 \endgroup
187 }

\AtBegShi@Hook

```

```

188 \gdef\AtBegShi@Hook{}

\AtBegShi@HookNext
189 \gdef\AtBegShi@HookNext{}

\AtBegShi@HookFirst
190 \gdef\AtBegShi@HookFirst{}

\AtBeginShipout
191 \AtBegShi@CheckDefinable\AtBeginShipout
192 \def\AtBeginShipout{%
193   \AtBegShi@AddHook\AtBegShi@Hook
194 }

\AtBeginShipoutNext
195 \AtBegShi@CheckDefinable\AtBeginShipoutNext
196 \def\AtBeginShipoutNext{%
197   \AtBegShi@AddHook\AtBegShi@HookNext
198 }

\AtBeginShipoutFirst
199 \AtBegShi@CheckDefinable\AtBeginShipoutFirst
200 \def\AtBeginShipoutFirst{%
201   \AtBegShi@AddTo\AtBegShi@HookFirst
202 }

\AtBegShi@FirstDisabled
203 \long\def\AtBegShi@FirstDisabled#1{%
204   \@PackageWarning{atbegshi}{%
205     First page is already shipped out, ignoring\MessageBreak
206     \string\AtBeginShipoutFirst
207   }%
208 }

\AtBegShi@AddTo
209 \begingroup\expandafter\expandafter\expandafter\endgroup
210 \expandafter\ifx\csname g@addto@macro\endcsname\relax
211   \long\def\AtBegShi@AddTo#1#2{%
212     \begingroup
213       \toks\z@\expandafter{#1#2}%
214       \xdef#1{\the\toks\z@}%
215     \endgroup
216   }%
217 \else
218   \let\AtBegShi@AddTo\g@addto@macro
219 \fi

\AtBegShi@AddHook
220 \long\def\AtBegShi@AddHook#1#2{%
221   \AtBegShi@AddTo#1{\AtBegShi@Item{#2}}%
222 }

\AtBegShi@Item
223 \long\def\AtBegShi@Item#1{%
224   \ifAtBegShi@Discarded
225   \else
226     #1%
227   \ifvoid\AtBeginShipoutBox
228     \@PackageWarning{atbegshi}{%
229       Shipout box was voided by hook,\MessageBreak

```

```

230     ignoring shipout box%
231 }%
232 \AtBeginShipoutDiscard
233 \fi
234 \fi
235 }

\AtBeginShipoutInit

236 \AtBegShi@CheckDefinable\AtBeginShipoutInit
237 \def\AtBeginShipoutInit{%
238   \csname newbox\endcsname\AtBeginShipoutBox
239   \AtBegShi@CheckDefinable\AtBegShi@OrgShipout
240   \global\let\AtBegShi@OrgShipout\shipout
241   \global\let\shipout\AtBegShi@Shipout
242   \gdef\AtBeginShipoutInit{}%
243 }

244 \begingroup\expandafter\expandafter\expandafter\endgroup
245 \expandafter\ifx\csname AtBeginDocument\endcsname\relax
246   \AtBeginShipoutInit
247 \else
248   \AtBeginDocument{\AtBeginShipoutInit}%
249 \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.

```

250 \def\AtBegShi@AbortIfUndefined#1{%
251   \begingroup\expandafter\expandafter\expandafter\endgroup
252   \expandafter\ifx\csname#1\endcsname\relax
253     \expandafter\AtBegShi@End
254   \fi
255 }
256 \AtBegShi@AbortIfUndefined{currentgrouplevel}
257 \AtBegShi@AbortIfUndefined{AtBeginDocument}
258 \AtBegShi@AbortIfUndefined{@ifpackageloaded}
259 \AtBegShi@AbortIfUndefined{@ifclassloaded}

```

#### 3.4.1 Package crop

Fix of method and box.

```

260 \def\AtBegShi@PatchCrop{%
261   \begingroup
262     \def\AtBegShi@Crop@shipout{%
263       \afterassignment\CROP@ship
264       \setbox\@cclv=%
265     }%
266     \def\AtBegShi@Crop@ship{%
267       \ifvoid\@cclv
268         \expandafter\aftergroup
269       \fi
270       \CROP@@ship
271     }%
272     \def\AtBegShi@Crop@shiplist{%
273       \lineskip\z@
274       \lineskiplimit\z@
275       \baselineskip\z@
276       \CROP@kernel
277       \box\@cclv
278     }%
279     \def\AtBegShi@Crop@@ship{%

```

```

280     \CROP@shipout\vbox{%
281     \CROP@shiplist
282     }%
283 }%
284 \ifx\AtBegShi@Crop@ship\CROP@ship
285 \ifx\AtBegShi@Crop@shiplist\CROP@shiplist
286 \ifx\AtBegShi@Crop@@@ship\CROP@@@ship
287 \let\AtBegShi@found\relax
288 \ifx\shipout\AtBegShi@Crop@shipout
289 \def\AtBegShi@found{\shipout}%
290 \else\ifx\AtBegShi@OrgShipout\AtBegShi@Crop@shipout
291 \def\AtBegShi@found{\AtBegShi@OrgShipout}%
292 \else\ifx\@EveryShipout@Org@Shipout\AtBegShi@Crop@shipout
293 \def\AtBegShi@found{\@EveryShipout@Org@Shipout}%
294 \else\ifx\GPTorg@shipout\AtBegShi@Crop@shipout
295 \def\AtBegShi@found{\GPTorg@shipout}%
296 \else\ifx\THBorg@shipout\AtBegShi@Crop@shipout
297 \def\AtBegShi@found{\THBorg@shipout}%
298 \else\ifx\mem@oldshipout\AtBegShi@Crop@shipout
299 \def\AtBegShi@found{\mem@oldshipout}%
300 \fi\fi\fi\fi\fi
301 \ifx\AtBegShi@found\relax
302 \else
303 \expandafter\endgroup
304 \expandafter\def\AtBegShi@found{%
305 \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
306 \afterassignment\CROP@ship
307 \setbox\AtBeginShipoutBox=%
308 }%
309 \def\CROP@ship{%
310 \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
311 \else
312 \expandafter\aftergroup
313 \fi
314 \CROP@@@ship
315 }%
316 \def\CROP@shiplist{%
317 \lineskip\z@
318 \lineskiplimit\z@
319 \baselineskip\z@
320 \CROP@kernel
321 \box\AtBeginShipoutBox
322 }%
323 \def\CROP@@@ship{%
324 \ifvoid\AtBeginShipoutBox
325 \else
326 \setbox\AtBeginShipoutBox=\vbox{%
327 \CROP@shiplist
328 }%
329 \expandafter\CROP@shipout
330 \expandafter\box
331 \expandafter\AtBeginShipoutBox
332 \fi
333 }%
334 \@PackageInfoNoLine{atbegshi}{Package 'crop' patched}%
335 \begingroup
336 \fi
337 \fi
338 \fi
339 \fi
340 \endgroup
341 \let\AtBegShi@PatchCrop\relax

```

```

342 }
343 \@ifpackageloaded{crop}{%
344   \AtBegShi@PatchCrop
345 }{%
346   \AtBeginDocument{\AtBegShi@PatchCrop}%
347 }

```

### 3.4.2 Package everyshi

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

```

348 \def\AtBegShi@PatchEveryshi{%
349   \begingroup
350     \long\def\AtBegShi@Everyshi@shipout{%
351       \afterassignment\@EveryShipout@Test
352       \global\setbox\@cclv= %
353     }%
354     \long\def\AtBegShi@Everyshi@Test{%
355       \ifvoid\@cclv\relax
356         \aftergroup\@EveryShipout@Output
357       \else
358         \@EveryShipout@Output
359       \fi
360     }%
361     \ifx\AtBegShi@Everyshi@Test\@EveryShipout@Test
362       \let\AtBegShi@found\relax
363       \ifx\shipout\AtBegShi@Everyshi@shipout
364         \def\AtBegShi@found{\shipout}%
365       \else\ifx\AtBegShi@OrgShipout\AtBegShi@Everyshi@shipout
366         \def\AtBegShi@found{\AtBegShi@OrgShipout}%
367       \else\ifx\CROP@shipout\AtBegShi@Everyshi@shipout
368         \def\AtBegShi@found{\CROP@shipout}%
369       \else\ifx\GPTorg@shipout\AtBegShi@Everyshi@shipout
370         \def\AtBegShi@found{\GPTorg@shipout}%
371       \else\ifx\THBorg@shipout\AtBegShi@Everyshi@shipout
372         \def\AtBegShi@found{\THBorg@shipout}%
373       \else\ifx\mem@oldshipout\AtBegShi@Everyshi@shipout
374         \def\AtBegShi@found{\mem@oldshipout}%
375       \else
376         \expandafter\ifx\csname @EveryShipout@Org@Shipout\endcsname
377           \relax
378           \ifx\@EveryShipout@Shipout\AtBegShi@Everyshi@shipout
379             \def\AtBegShi@found{\@EveryShipout@Shipout}%
380           \fi
381         \fi
382       \fi\fi\fi\fi\fi\fi
383       \ifx\AtBegShi@found\relax
384       \else
385         \expandafter\endgroup
386         \expandafter\def\AtBegShi@found{%
387           \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
388           \afterassignment\@EveryShipout@Test
389           \setbox\AtBeginShipoutBox=%
390         }%
391       \def\@EveryShipout@Test{%
392         \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
393         \else
394           \expandafter\aftergroup
395         \fi
396       \AtBegShi@Everyshi@Output
397     }%
398     \def\AtBegShi@Everyshi@Output{%
399       \ifvoid\AtBeginShipoutBox

```

```

400         \else
401         \global\setbox\@cclv\box\AtBeginShipoutBox
402         \expandafter\@EveryShipout@Output
403         \fi
404     }%
405     \@PackageInfoNoLine{atbegshi}{Package 'everyshi' patched}%
406     \begingroup
407     \fi
408     \fi
409 \endgroup
410 \let\AtBegShi@PatchEveryshi\relax
411 }
412 \@ifpackageloaded{everyshi}{%
413     \AtBegShi@PatchEveryshi
414 }{%
415     \AtBeginDocument{\AtBegShi@PatchEveryshi}%
416 }

```

### 3.4.3 Class memoir

Fix of method and box.

```

417 \def\AtBegShi@PatchMemoir{%
418     \begingroup
419     \def\AtBegShi@Memoir@shipout{%
420         \afterassignment\mem@shipi
421         \setbox\@cclv=%
422     }%
423     \def\AtBegShi@Memoir@shipi{%
424         \ifvoid\@cclv
425         \expandafter\aftergroup
426         \fi
427         \mem@shipii
428     }%
429     \def\AtBegShi@Memoir@shipiiA{%
430         \mem@oldshipout\vbox{%
431             \trimmarks
432             \unvbox\@cclv
433         }%
434     }%
435     \def\AtBegShi@Memoir@shipiiB{%
436         \ifvoid\@cclv
437         \mem@oldshipout\box\@cclv
438     \else
439         \mem@oldshipout\vbox{%
440             \trimmarks
441             \unvbox\@cclv
442         }%
443     \fi
444 }%
445 \ifx\AtBegShi@Memoir@shipi\mem@shipi
446     \ifcase\ifx\AtBegShi@Memoir@shipiiA\mem@shipii
447         \z@
448     \else
449         \ifx\AtBegShi@Memoir@shipiiB\mem@shipii
450             \z@
451         \else
452             \@ne
453         \fi
454     \fi
455     \let\AtBegShi@found\relax
456     \ifx\shipout\AtBegShi@Memoir@shipout
457         \def\AtBegShi@found{\shipout}%

```



```

458 \else\ifx\AtBegShi@OrgShipout\AtBegShi@Mmemoir@shipout
459 \def\AtBegShi@found{\AtBegShi@OrgShipout}%
460 \else\ifx\CROP@shipout\AtBegShi@Mmemoir@shipout
461 \def\AtBegShi@found{\CROP@shipout}%
462 \else\ifx\GPTorg@shipout\AtBegShi@Mmemoir@shipout
463 \def\AtBegShi@found{\GPTorg@shipout}%
464 \else\ifx\THBorg@shipout\AtBegShi@Mmemoir@shipout
465 \def\AtBegShi@found{\THBorg@shipout}%
466 \else\ifx\@EveryShipout@Org@Shipout\AtBegShi@Mmemoir@shipout
467 \def\AtBegShi@found{\@EveryShipout@Org@Shipout}%
468 \fi\fi\fi\fi\fi\fi
469 \ifx\AtBegShi@found\relax
470 \else
471 \expandafter\endgroup
472 \expandafter\def\AtBegShi@found{%
473 \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
474 \afterassignment\mem@shipi
475 \setbox\AtBeginShipoutBox=%
476 }%
477 \def\mem@shipi{%
478 \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
479 \else
480 \expandafter\aftergroup
481 \fi
482 \mem@shipii
483 }%
484 \def\mem@shipii{%
485 \ifvoid\AtBeginShipoutBox
486 \else
487 \setbox\AtBeginShipoutBox=\vbox{%
488 \trimmarks
489 \ifvbox\AtBeginShipoutBox
490 \unvbox\AtBeginShipoutBox
491 \else
492 \box\AtBeginShipoutBox
493 \fi
494 }%
495 \expandafter\mem@oldshipout
496 \expandafter\box
497 \expandafter\AtBeginShipoutBox
498 \fi
499 }%
500 \@PackageInfoNoLine{atbegshi}{Class 'memoir' patched}%
501 \begingroup
502 \fi
503 \fi
504 \fi
505 \endgroup
506 \let\AtBegShi@PatchMemoir\relax
507 }
508 \ifclassloaded{memoir}{%
509 \AtBegShi@PatchMemoir
510 }{%
511 \AtBeginDocument{\AtBegShi@PatchMemoir}%
512 }
513 \AtBegShi@End
514 \end{package}

```

## 4 Test

```

515 \test1

```

```

516 \input atbegshi.sty\relax
517 \def\msg#\{\immediate\write16}
518 \msg{File: atbegshi-test1.tex 2007/04/27 v1.4 Test file for plain-TeX}
519 \def\testmsg#1#2{%
520   \msg{}%
521   \msg{*** Test with box (#1), expected page output [#2]}}%
522 }
523
524 \newbox\voidbox
525 \def\void{\box\voidbox}
526 \begingroup
527   \setbox\voidbox=\void
528 \endgroup
529
530 \count0=0\relax
531 \AtBeginShipout{%
532   \global\advance\count0 by 1\relax
533   \msg{* Inside \string\AtBeginShipout: [\the\count0]}}%
534 }
535
536 \AtBeginShipoutFirst{%
537   \msg{* Inside \string\AtBeginShipoutFirst}}%
538   Hello World%
539 }
540
541 \testmsg{\string\null}{1}
542 \shipout\null
543
544 \AtBeginShipoutFirst{%
545   This is too late%
546 }
547
548 \testmsg{void}{1}
549 \shipout\void
550
551 \testmsg{\string\copy255 (not void)}{2}
552 \setbox255\hbox{\vrule height 10bp width 10bp}
553 \shipout\copy255 %
554
555 \testmsg{\string\copy255 (again)}{3}
556 \shipout\copy255 %
557
558 \testmsg{\string\box255}{4}
559 \shipout\box255 %
560
561 \testmsg{\string\box255 (again)}{5}
562 \shipout\box255 %
563
564 \testmsg{\string\hbox}{5}
565 \shipout\hbox{\vrule height 5bp width 20bp}
566
567 \testmsg{\string\vbox}{6}
568 \shipout\vbox{\hrule height 20bp width 5bp}
569
570 \testmsg{\string\null, voided by hook}{1}
571 \def\VoidBox{%
572   \begingroup
573     \setbox\AtBeginShipoutBox=\box\AtBeginShipoutBox
574   \endgroup
575 }
576 \AtBeginShipout{\VoidBox}
577 \shipout\null

```

```

578 \def\VoidBox{}
579
580 \msg{*** \string\beginpgroup}
581 \beginpgroup
582   \testmsg{void}{}%
583   \shipout\void
584 \msg{*** \string\endpgroup}
585 \endpgroup
586
587 \msg{*** \string\beginpgroup}
588 \beginpgroup
589   \testmsg{void}{}%
590   \shipout\void
591   \testmsg{\string\null}{8}%
592   \shipout\null
593 \msg{*** \string\endpgroup}
594 \endpgroup
595
596 \testmsg{output routine}{9}
597 Hello World
598 \vfill
599 \eject
600
601 \testmsg{\string\null\space(discarded)}{}
602 \AtBeginShipout{%
603   \msg{* Inside \string\AtBeginShipout: DISCARD}%
604   \AtBeginShipoutDiscard
605 }
606 \shipout\null
607
608 \end
609 </test1>
610 <*test2>
611 \NeedsTeXFormat{LaTeX2e}
612 \ProvidesFile{atbegshi-test2.tex}[2007/04/27 v1.4 Test file for LaTeX]
613 \RequirePackage{color}
614 \pagecolor{yellow}
615 \documentclass[a5paper,showtrims]{memoir}
616 \usepackage{atbegshi}
617 \AtBeginShipout{%
618   \setbox\AtBeginShipoutBox=\vbox{%
619     \vbox to 0pt{%
620       \kern-1.5in %
621       \hbox to 0pt{%
622         \kern-1.5in %
623         \color{blue}%
624         \rule{1in}{1in}%
625         \hss
626       }%
627       \vss
628     }%
629     \hrule
630     \hbox{\vrule\box\AtBeginShipoutBox\vrule}%
631     \hrule
632   }%
633 }
634 \usepackage{eso-pic}
635 \makeatletter
636 \@EveryShipout@Init
637 \let\@EveryShipout@Init\relax
638 \makeatother
639 \AddToShipoutPicture{%

```

```

640 \hspace{.52\paperwidth}%
641 \colorbox{cyan}{%
642   \rule{0mm}{\paperheight}%
643   \hspace{.48\paperwidth}%
644 }%
645 }
646 \usepackage[color=red,cross,a4,center]{crop}
647 \begin{document}
648 \shipout\null
649 \shipout\box\csname voidb@x\endcsname
650 \section{Hello World}
651 \end{document}
652 </test2>

```

## 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: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

---

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

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 (`teTeX`, `mikTeX`, ...) relies on file name databases, you must refresh these. For example, `teTeX` 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  $\LaTeX$ .** The `.dtx` chooses its action depending on the format:

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

**$\LaTeX$ :** Generate the documentation.

If you insist on using  $\LaTeX$  for `docstrip` (really, `docstrip` does not need  $\LaTeX$ ), 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 $\LaTeX$` :

```
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  $\epsilon$ - $\TeX$  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.

## 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> . . . . .	636, 637
<code>\@EveryShipout@Org@Shipout</code> . . . . .	292, 293, 466, 467
<code>\@EveryShipout@Output</code> . . . . .	356, 358, 402
<code>\@EveryShipout@Shipout</code> . . . . .	378, 379
<code>\@EveryShipout@Test</code> . . . . .	351, 361, 388, 391
<code>\@PackageInfoNoLine</code> . . . . .	148, 334, 405, 500
<code>\@PackageWarning</code> . . . . .	139, 204, 228
<code>\@ccclv</code> . . . . .	264, 267, 277, 352, 355, 401, 421, 424, 432, 436, 437, 441
<code>\@empty</code> . . . . .	165, 166, 184
<code>\@ifclassloaded</code> . . . . .	508
<code>\@ifdefinable</code> . . . . .	102
<code>\@ifpackageloaded</code> . . . . .	343, 412
<code>\@one</code> . . . . .	85, 88, 452
<code>\@undefined</code> . . . . .	87
A	
<code>\AddToShipoutPicture</code> . . . . .	639
<code>\advance</code> . . . . .	532
<code>\afterassignment</code> . . . . .	124, 263, 306, 351, 388, 420, 474
<code>\aftergroup</code> . . . . .	131, 268, 312, 356, 394, 425, 480
<code>\AtBeginDocument</code> . . . . .	248, 346, 415, 511
<code>\AtBeginShipout</code> . . . . .	2, 191, 531, 533, 576, 602, 603, 617
<code>\AtBeginShipoutBox</code> . . . . .	121, 126, 138, 151, 157, 176, 181, 227, 238, 307, 321, 324, 326, 331, 389, 399, 401, 475, 485, 487, 489, 490, 492, 497, 573, 618, 630
<code>\AtBeginShipoutDiscard</code> . . . . .	3, 106, 232, 604
<code>\AtBeginShipoutFirst</code> . . . . .	2, 185, 199, 206, 536, 537, 544
<code>\AtBeginShipoutInit</code> . . . . .	3, 236, 246, 248
<code>\AtBeginShipoutNext</code> . . . . .	2, 195
<code>\AtBegShi@AbortIfUndefined</code> . . . . .	250, 256, 257, 258, 259
<code>\AtBegShi@AddHook</code> . . . . .	193, 197, 220
<code>\AtBegShi@AddTo</code> . . . . .	201, 209, 221
<code>\AtBegShi@CheckDefinable</code> . . . . .	81, 106, 191, 195, 199, 236, 239
<code>\AtBegShi@Crop@ship</code> . . . . .	279, 286
<code>\AtBegShi@Crop@ship</code> . . . . .	266, 284
<code>\AtBegShi@Crop@shiplist</code> . . . . .	272, 285
<code>\AtBegShi@Crop@shipout</code> . . . . .	262, 288, 290, 292, 294, 296, 298
<code>\AtBegShi@Discardedfalse</code> . . . . .	143, 149
<code>\AtBegShi@Discardedtrue</code> . . . . .	109
<code>\AtBegShi@End</code> . . . . .	253, 513
<code>\AtBegShi@Everyshi@Output</code> . . . . .	396, 398
<code>\AtBegShi@Everyshi@shipout</code> . . . . .	350, 363, 365, 367, 369, 371, 373, 378
<code>\AtBegShi@Everyshi@Test</code> . . . . .	354, 361
<code>\AtBegShi@First</code> . . . . .	155, 163
<code>\AtBegShi@FirstDisabled</code> . . . . .	185, 203
<code>\AtBegShi@found</code> . . . . .	287, 289, 291, 293, 295, 297, 299, 301, 304, 362, 364, 366, 368, 370, 372, 374, 379, 383, 386, 455, 457, 459, 461, 463, 465, 467, 469, 472
<code>\AtBegShi@GroupLevel</code> . . . . .	123, 130, 305, 310, 387, 392, 473, 478
<code>\AtBegShi@Hook</code> . . . . .	144, 188, 193
<code>\AtBegShi@HookFirst</code> . . . . .	166, 170, 190, 201
<code>\AtBegShi@HookNext</code> . . . . .	145, 146, 189, 197
<code>\AtBegShi@Item</code> . . . . .	221, 223
<code>\AtBegShi@Memoir@shipi</code> . . . . .	423, 445
<code>\AtBegShi@Memoir@shipiiA</code> . . . . .	429, 446
<code>\AtBegShi@Memoir@shipiiB</code> . . . . .	435, 449
<code>\AtBegShi@Memoir@shipout</code> . . . . .	419, 456, 458, 460, 462, 464, 466
<code>\AtBegShi@OrgProtect</code> . . . . .	141, 153, 156
<code>\AtBegShi@OrgShipout</code> . . . . .	157, 239, 240, 290, 291, 365, 366, 458, 459
<code>\AtBegShi@Output</code> . . . . .	133, 135
<code>\AtBegShi@PatchCrop</code> . . . . .	260, 341, 344, 346
<code>\AtBegShi@PatchEveryshi</code> . . . . .	348, 410, 413, 415
<code>\AtBegShi@PatchMemoir</code> . . . . .	417, 506, 509, 511
<code>\AtBegShi@Shipout</code> . . . . .	119, 241
<code>\AtBegShi@Test</code> . . . . .	124, 128
<code>\AtBegShi@Warning</code> . . . . .	75
B	
<code>\baselineskip</code> . . . . .	177, 275, 319
<code>\begin</code> . . . . .	647

<code>\box</code> . . . . .	151, 157, 277, 321, 330, 401, 437, 492, 496, 525, 558, 559, 561, 562, 573, 630, 649	252, 284, 285, 286, 288, 290, 292, 294, 296, 298, 301, 361, 363, 365, 367, 369, 371, 373, 376, 378, 383, 445, 446, 449, 456, 458, 460, 462, 464, 466, 469
<b>C</b>		
<code>\catcode</code> . . . . .	3, 4, 5, 6, 7, 31, 32, 33, 34, 35, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 113, 114, 116, 117, 161, 162	<code>\immediate</code> . . . . . 20, 38, 517 <code>\input</code> . . . . . 77, 516
<code>\color</code> . . . . .	623	<b>K</b>
<code>\colorbox</code> . . . . .	641	<code>\kern</code> . . . . . 122, 620, 622
<code>\copy</code> . . . . .	180, 181, 551, 553, 555, 556	<b>L</b>
<code>\count</code> . . . . .	530, 532, 533	<code>\lastkern</code> . . . . . 129
<code>\CROP@ship</code> . . . . .	270, 286, 314, 323	<code>\lineskip</code> . . . . . 178, 273, 317
<code>\CROP@kernel</code> . . . . .	276, 320	<code>\lineskiplimit</code> . . . . . 179, 274, 318
<code>\CROP@ship</code> . . . . .	263, 284, 306, 309	<b>M</b>
<code>\CROP@shiplist</code> . . . . .	281, 285, 316, 327	<code>\makeatletter</code> . . . . . 635
<code>\CROP@shipout</code> . . . . .	280, 329, 367, 368, 460, 461	<code>\makeatother</code> . . . . . 638
<code>\csname</code> 8, 18, 36, 49, 52, 76, 82, 112, 142, 210, 238, 245, 252, 376, 649		<code>\mem@oldshipout</code> . . . . . 298, 299, 373, 374, 430, 437, 439, 495
<code>\currentgrouplevel</code> . . . . .	123, 130, 305, 310, 387, 392, 473, 478	<code>\mem@shipi</code> . . . . . 420, 445, 474, 477
<b>D</b>		<code>\mem@shipii</code> . . . . . 427, 446, 449, 482, 484
<code>\deadcycles</code> . . . . .	108	<code>\MessageBreak</code> . . . . . 205, 229
<code>\documentclass</code> . . . . .	615	<code>\msg</code> . . . . . 517, 518, 520, 521, 533, 537, 580, 584, 587, 593, 603
<code>\dp</code> . . . . .	175	<b>N</b>
<b>E</b>		<code>\NeedsTeXFormat</code> . . . . . 611
<code>\E</code> . . . . .	162	<code>\newbox</code> . . . . . 524
<code>\eject</code> . . . . .	599	<code>\newif</code> . . . . . 105
<code>\empty</code> . . . . .	12	<code>\null</code> . . . . . 541, 542, 570, 577, 591, 592, 601, 606, 648
<code>\end</code> . . . . .	608, 651	<code>\number</code> . . . . . 123, 305, 387, 473
<code>\endcsname</code> . . . . .	8, 18, 36, 49, 52, 76, 82, 112, 142, 210, 238, 245, 252, 376, 649	<b>P</b>
<code>\endinginput</code> . . . . .	27, 63	<code>\p@</code> . . . . . 122
<code>\errmessage</code> . . . . .	93	<code>\PackageInfo</code> . . . . . 23
<b>G</b>		<code>\pagecolor</code> . . . . . 614
<code>\g@addto@macro</code> . . . . .	218	<code>\paperheight</code> . . . . . 642
<code>\gdef</code> . . . . .	146, 188, 189, 190, 242	<code>\paperwidth</code> . . . . . 640, 643
<code>\GPTorg@shipout</code> . . . . .	294, 295, 369, 370, 462, 463	<code>\protect</code> . . . . . 141, 153, 156
<b>H</b>		<code>\ProvidesFile</code> . . . . . 612
<code>\hbox</code> . . . . .	121, 552, 564, 565, 621, 630	<code>\ProvidesPackage</code> . . . . . 50
<code>\hrule</code> . . . . .	568, 629, 631	<b>R</b>
<code>\hspace</code> . . . . .	640, 643	<code>\RequirePackage</code> . . . . . 79, 613
<code>\hss</code> . . . . .	625	<code>\rule</code> . . . . . 624, 642
<code>\ht</code> . . . . .	174	<b>S</b>
<b>I</b>		<code>\section</code> . . . . . 650
<code>\ifAtBegShi@Discarded</code> . . . . .	105, 147, 224	<code>\setbox</code> . . . . . 121, 126, 151, 168, 176, 264, 307, 326, 352, 389, 401, 421, 475, 487, 527, 552, 573, 618
<code>\ifcase</code> . . . . .	9, 84, 446	<code>\shipout</code> . . . . . 240, 241, 288, 289, 363, 364, 456, 457, 542, 549, 553, 556, 559, 562, 565, 568, 577, 583, 590, 592, 606, 648, 649
<code>\ifdim</code> . . . . .	129	<code>\space</code> . . . . . 94, 601
<code>\ifnum</code> . . . . .	130, 310, 392, 478	<b>T</b>
<code>\ifvbox</code> . . . . .	489	<code>\testmsg</code> . . . . . 519, 541, 548, 551, 555, 558, 561, 564, 567, 570, 582, 589, 591, 596, 601
<code>\ifvoid</code> . . . . .	138, 227, 267, 324, 355, 399, 424, 436, 485	
<code>\ifx</code> . . . . .	10, 12, 18, 36, 44, 76, 82, 84, 87, 112, 166, 210, 245,	

