The atbegshi package

Heiko Oberdiek
<oberdiek@uni-freiburg.de>

2007/04/27 v1.4

Abstract

This package is a modern reimplementation of package everyshi without the burden of compatibility. It makes use of ε-TEX's if available. Both LaTeX and plain-TEX are supported.

Contents

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 catched 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 inpredictable if TEX would put `\@test` after the first of such an assignment or after the box specification if the box lacks of assignments. Therefore TEX puts `\@test` right at the beginning of the box specification, e.g:

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

2.3 Test for direct or indirect boxes

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

2.3.1 With ε-TEX

With the ε-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 ε-TEX

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

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

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

Unhappily all these qualities even combined are not sufficient for constructing an initial box value, because `\shipout` can be called with a box that is accidently 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 ε-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 ε-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 ε-TEX

Putting the pieces together we get for ε-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 ε-TEX, traditional way

And for TEX without ε-TEX:

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

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

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

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

2.6.3 `\lastkern` method

And for TEX without ε-TEX using the `\lastkern` method:

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

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

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

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

3 Implementation

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

```
1 ⟨*package⟩
```

3.1 Reload check and package identification

Reload check, especially if the package is not used with LaTeX.

```
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}%
```

```
    \endgroup
    \expandafter\endinput
  \fi
\endgroup
```

Package identification:

```
\begingroup
  \catcode44 12 % ,
  \catcode45 12 % -
  \catcode46 12 % .
  \catcode58 12 % :
  \catcode64 11 % @
  \expandafter\ifx\csname ProvidesPackage\endcsname\relax
    \def\x#1#2#3[#4]{\endgroup
      \immediate\write-1{Package: #3 #4}%
      \xdef#1{#4}%
    }%
  \else
    \def\x#1#2[#3]{\endgroup
      #2[{#3}]%
      \ifx#1\relax
        \xdef#1{#3}%
      \fi
    }%
  \fi
\expandafter\x\csname ver@atbegshi.sty\endcsname
\ProvidesPackage{atbegshi}%
  [2007/04/27 v1.4 At begin shipout hook (HO)]
```

3.2 Catcodes

```
\expandafter\edef\csname AtBegShi@End\endcsname{%
  \catcode40 \the\catcode40\relax % (
  \catcode41 \the\catcode41\relax % )
  \catcode44 \the\catcode44\relax % ,
  \catcode45 \the\catcode45\relax % -
  \catcode46 \the\catcode46\relax % .
  \catcode58 \the\catcode58\relax % :
  \catcode61 \the\catcode61\relax % =
  \catcode64 \the\catcode64\relax % @
  \catcode94 \the\catcode94\relax % ^
  \catcode96 \the\catcode96\relax % `
  \noexpand\endinput
}
\catcode40 12 % (
\catcode41 12 % )
\catcode44 12 % ,
\catcode45 12 % -
\catcode46 12 % .
\catcode58 12 % :
\catcode61 12 % =
\catcode64 11 % @
\catcode94 7  % ^
\catcode96 12 % `
```

3.3 Preparations

`\AtBegShi@Warning`

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname RequirePackage\endcsname\relax
  \input infwarerr.sty\relax
\else
  \RequirePackage{infwarerr}%
```

```
\fi
```

\AtBegShi@CheckDefinable

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname @ifdefinable\endcsname\relax
  \def\AtBegShi@CheckDefinable#1{%
    \ifcase\ifx#1\relax
             \@ne
           \else
             \ifx#1\@undefined
               \@ne
             \else
               \z@
             \fi
           \fi
      \errmessage{%
        Package atbegshi: \string#1\space
        is already defined%
      }%
      \endgroup
    \fi
  }%
\else
  \def\AtBegShi@CheckDefinable#1{%
    \@ifdefinable{#1}{}%
  }%
\fi
\newif\ifAtBegShi@Discarded
```

\AtBeginShipoutDiscard

```
\AtBegShi@CheckDefinable\AtBeginShipoutDiscard
\def\AtBeginShipoutDiscard{%
  \deadcycles=\z@
  \global\AtBegShi@Discardedtrue
}
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname currentgrouplevel\endcsname\relax
  \catcode`X=9  % ignore
  \catcode`E=14 % comment
\else
  \catcode`X=14 % comment
  \catcode`E=9  % ignore
\fi
```

\AtBegShi@Shipout

```
\def\AtBegShi@Shipout{%
X \begingroup
X \setbox\AtBeginShipoutBox=\hbox\bgroup
X \kern\p@
E \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
  \afterassignment\AtBegShi@Test
X \global
  \setbox\AtBeginShipoutBox=%
}
```

\AtBegShi@Test

```
\def\AtBegShi@Test{%
X \ifdim\lastkern=\z@
E \ifnum\AtBegShi@GroupLevel<\currentgrouplevel
    \expandafter\aftergroup
```

```
  \fi
  \AtBegShi@Output
}
```

\AtBegShi@Output

```
\def\AtBegShi@Output{%
X \egroup
X \endgroup
  \ifvoid\AtBeginShipoutBox
    \@PackageWarning{atbegshi}{Ignoring void shipout box}%
  \else
    \let\AtBegShi@OrgProtect\protect
    \csname set@typeset@protect\endcsname
    \global\AtBegShi@Discardedfalse
    \AtBegShi@Hook
    \AtBegShi@HookNext
    \gdef\AtBegShi@HookNext{}%
    \ifAtBegShi@Discarded
      \@PackageInfoNoLine{atbegshi}{Shipout page discarded}%
      \global\AtBegShi@Discardedfalse
      \begingroup
        \setbox\AtBeginShipoutBox\box\AtBeginShipoutBox
      \endgroup
      \let\protect\AtBegShi@OrgProtect
    \else
      \AtBegShi@First
      \let\protect\AtBegShi@OrgProtect
      \AtBegShi@OrgShipout\box\AtBeginShipoutBox
    \fi
  \fi
}
\catcode`\X=11 %
\catcode`\E=11 %
```

\AtBegShi@First

```
\def\AtBegShi@First{%
  \begingroup
    \def\@empty{}%
    \ifx\AtBegShi@HookFirst\@empty
    \else
      \setbox\z@=\vbox{%
        \begingroup
          \AtBegShi@HookFirst
        \endgroup
      }%
      \wd\z@=\z@
      \ht\z@=\z@
      \dp\z@=\z@
      \global\setbox\AtBeginShipoutBox=\vbox{%
        \baselineskip\z@skip
        \lineskip\z@skip
        \lineskiplimit\z@
        \copy\z@
        \copy\AtBeginShipoutBox
      }%
    \fi
    \global\let\AtBegShi@First\@empty
    \global\let\AtBeginShipoutFirst\AtBegShi@FirstDisabled
  \endgroup
}
```

\AtBegShi@Hook

```
\gdef\AtBegShi@Hook{}
```

\AtBegShi@HookNext

```
\gdef\AtBegShi@HookNext{}
```

\AtBegShi@HookFirst

```
\gdef\AtBegShi@HookFirst{}
```

\AtBeginShipout

```
\AtBegShi@CheckDefinable\AtBeginShipout
\def\AtBeginShipout{%
  \AtBegShi@AddHook\AtBegShi@Hook
}
```

\AtBeginShipoutNext

```
\AtBegShi@CheckDefinable\AtBeginShipoutNext
\def\AtBeginShipoutNext{%
  \AtBegShi@AddHook\AtBegShi@HookNext
}
```

\AtBeginShipoutFirst

```
\AtBegShi@CheckDefinable\AtBeginShipoutFirst
\def\AtBeginShipoutFirst{%
  \AtBegShi@AddTo\AtBegShi@HookFirst
}
```

\AtBegShi@FirstDisabled

```
\long\def\AtBegShi@FirstDisabled#1{%
  \@PackageWarning{atbegshi}{%
    First page is already shipped out, ignoring\MessageBreak
    \string\AtBeginShipoutFirst
  }%
}
```

\AtBegShi@AddTo

```
\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname g@addto@macro\endcsname\relax
  \long\def\AtBegShi@AddTo#1#2{%
    \begingroup
      \toks\z@\expandafter{#1#2}%
      \xdef#1{\the\toks\z@}%
    \endgroup
  }%
\else
  \let\AtBegShi@AddTo\g@addto@macro
\fi
```

\AtBegShi@AddHook

```
\long\def\AtBegShi@AddHook#1#2{%
  \AtBegShi@AddTo#1{\AtBegShi@Item{#2}}%
}
```

\AtBegShi@Item

```
\long\def\AtBegShi@Item#1{%
  \ifAtBegShi@Discarded
  \else
    #1%
    \ifvoid\AtBeginShipoutBox
      \@PackageWarning{atbegshi}{%
        Shipout box was voided by hook,\MessageBreak
```

```
        ignoring shipout box%
      }%
      \AtBeginShipoutDiscard
    \fi
  \fi
}
```

\AtBeginShipoutInit

```
\AtBegShi@CheckDefinable\AtBeginShipoutInit
\def\AtBeginShipoutInit{%
  \csname newbox\endcsname\AtBeginShipoutBox
  \AtBegShi@CheckDefinable\AtBegShi@OrgShipout
  \global\let\AtBegShi@OrgShipout\shipout
  \global\let\shipout\AtBegShi@Shipout
  \gdef\AtBeginShipoutInit{}%
}

\begingroup\expandafter\expandafter\expandafter\endgroup
\expandafter\ifx\csname AtBeginDocument\endcsname\relax
  \AtBeginShipoutInit
\else
  \AtBeginDocument{\AtBeginShipoutInit}%
\fi
```

3.4 Patches

Patches for LaTeX packages that redefine `\shipout`. LaTeX is now supposed to use ε-TeX. Thus we do not patch, without LaTeX and ε-TeX.

```
\def\AtBegShi@AbortIfUndefined#1{%
  \begingroup\expandafter\expandafter\expandafter\endgroup
  \expandafter\ifx\csname#1\endcsname\relax
    \expandafter\AtBegShi@End
  \fi
}
\AtBegShi@AbortIfUndefined{currentgrouplevel}
\AtBegShi@AbortIfUndefined{AtBeginDocument}
\AtBegShi@AbortIfUndefined{@ifpackageloaded}
\AtBegShi@AbortIfUndefined{@ifclassloaded}
```

3.4.1 Package **crop**

Fix of method and box.

```
\def\AtBegShi@PatchCrop{%
  \begingroup
    \def\AtBegShi@Crop@shipout{%
      \afterassignment\CROP@ship
      \setbox\@cclv=%
    }%
    \def\AtBegShi@Crop@ship{%
      \ifvoid\@cclv
        \expandafter\aftergroup
      \fi
      \CROP@@ship
    }%
    \def\AtBegShi@Crop@shiplist{%
      \lineskip\z@
      \lineskiplimit\z@
      \baselineskip\z@
      \CROP@kernel
      \box\@cclv
    }%
    \def\AtBegShi@Crop@@ship{%
```

```
    \CROP@shipout\vbox{%
      \CROP@shiplist
    }%
  }%
  \ifx\AtBegShi@Crop@ship\CROP@ship
    \ifx\AtBegShi@Crop@shiplist\CROP@shiplist
      \ifx\AtBegShi@Crop@@ship\CROP@@ship
        \let\AtBegShi@found\relax
        \ifx\shipout\AtBegShi@Crop@shipout
          \def\AtBegShi@found{\shipout}%
        \else\ifx\AtBegShi@OrgShipout\AtBegShi@Crop@shipout
          \def\AtBegShi@found{\AtBegShi@OrgShipout}%
        \else\ifx\@EveryShipout@Org@Shipout\AtBegShi@Crop@shipout
          \def\AtBegShi@found{\@EveryShipout@Org@Shipout}%
        \else\ifx\GPTorg@shipout\AtBegShi@Crop@shipout
          \def\AtBegShi@found{\GPTorg@shipout}%
        \else\ifx\THBorg@shipout\AtBegShi@Crop@shipout
          \def\AtBegShi@found{\THBorg@shipout}%
        \else\ifx\mem@oldshipout\AtBegShi@Crop@shipout
          \def\AtBegShi@found{\mem@oldshipout}%
        \fi\fi\fi\fi\fi\fi
        \ifx\AtBegShi@found\relax
        \else
          \expandafter\endgroup
          \expandafter\def\AtBegShi@found{%
            \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
            \afterassignment\CROP@ship
            \setbox\AtBeginShipoutBox=%
          }%
          \def\CROP@ship{%
            \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
            \else
              \expandafter\aftergroup
            \fi
            \CROP@@ship
          }%
          \def\CROP@shiplist{%
            \lineskip\z@
            \lineskiplimit\z@
            \baselineskip\z@
            \CROP@kernel
            \box\AtBeginShipoutBox
          }%
          \def\CROP@@ship{%
            \ifvoid\AtBeginShipoutBox
            \else
              \setbox\AtBeginShipoutBox=\vbox{%
                \CROP@shiplist
              }%
              \expandafter\CROP@shipout
              \expandafter\box
              \expandafter\AtBeginShipoutBox
            \fi
          }%
          \@PackageInfoNoLine{atbegshi}{Package `crop' patched}%
          \begingroup
        \fi
      \fi
    \fi
  \fi
\endgroup
\let\AtBegShi@PatchCrop\relax
```

```
}
\@ifpackageloaded{crop}{%
  \AtBegShi@PatchCrop
}{%
  \AtBeginDocument{\AtBegShi@PatchCrop}%
}
```

3.4.2 Package everyshi

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

```
\def\AtBegShi@PatchEveryshi{%
  \begingroup
    \long\def\AtBegShi@Everyshi@shipout{%
      \afterassignment\@EveryShipout@Test
      \global\setbox\@cclv= %
    }%
    \long\def\AtBegShi@Everyshi@Test{%
      \ifvoid\@cclv\relax
        \aftergroup\@EveryShipout@Output
      \else
        \@EveryShipout@Output
      \fi
    }%
    \ifx\AtBegShi@Everyshi@Test\@EveryShipout@Test
      \let\AtBegShi@found\relax
      \ifx\shipout\AtBegShi@Everyshi@shipout
        \def\AtBegShi@found{\shipout}%
      \else\ifx\AtBegShi@OrgShipout\AtBegShi@Everyshi@shipout
        \def\AtBegShi@found{\AtBegShi@OrgShipout}%
      \else\ifx\CROP@shipout\AtBegShi@Everyshi@shipout
        \def\AtBegShi@found{\CROP@shipout}%
      \else\ifx\GPTorg@shipout\AtBegShi@Everyshi@shipout
        \def\AtBegShi@found{\GPTorg@shipout}%
      \else\ifx\THBorg@shipout\AtBegShi@Everyshi@shipout
        \def\AtBegShi@found{\THBorg@shipout}%
      \else\ifx\mem@oldshipout\AtBegShi@Everyshi@shipout
        \def\AtBegShi@found{\mem@oldshipout}%
      \else
        \expandafter\ifx\csname @EveryShipout@Org@Shipout\endcsname
                        \relax
           \ifx\@EveryShipout@Shipout\AtBegShi@Everyshi@shipout
             \def\AtBegShi@found{\@EveryShipout@Shipout}%
           \fi
        \fi
      \fi\fi\fi\fi\fi\fi
      \ifx\AtBegShi@found\relax
      \else
        \expandafter\endgroup
        \expandafter\def\AtBegShi@found{%
          \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
          \afterassignment\@EveryShipout@Test
          \setbox\AtBeginShipoutBox=%
        }%
        \def\@EveryShipout@Test{%
          \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
          \else
            \expandafter\aftergroup
          \fi
          \AtBegShi@Everyshi@Output
        }%
        \def\AtBegShi@Everyshi@Output{%
          \ifvoid\AtBeginShipoutBox
```

```
          \else
            \global\setbox\@cclv\box\AtBeginShipoutBox
            \expandafter\@EveryShipout@Output
          \fi
        }%
        \@PackageInfoNoLine{atbegshi}{Package 'everyshi' patched}%
        \begingroup
      \fi
    \fi
  \endgroup
  \let\AtBegShi@PatchEveryshi\relax
}
\@ifpackageloaded{everyshi}{%
  \AtBegShi@PatchEveryshi
}{%
  \AtBeginDocument{\AtBegShi@PatchEveryshi}%
}
```

3.4.3 Class memoir

Fix of method and box.

```
\def\AtBegShi@PatchMemoir{%
  \begingroup
    \def\AtBegShi@Memoir@shipout{%
      \afterassignment\mem@shipi
      \setbox\@cclv=%
    }%
    \def\AtBegShi@Memoir@shipi{%
      \ifvoid\@cclv
        \expandafter\aftergroup
      \fi
      \mem@shipii
    }%
    \def\AtBegShi@Memoir@shipiiA{%
      \mem@oldshipout\vbox{%
        \trimmarks
        \unvbox\@cclv
      }%
    }%
    \def\AtBegShi@Memoir@shipiiB{%
      \ifvoid\@cclv
        \mem@oldshipout\box\@cclv
      \else
        \mem@oldshipout\vbox{%
          \trimmarks
          \unvbox\@cclv
        }%
      \fi
    }%
    \ifx\AtBegShi@Memoir@shipi\mem@shipi
      \ifcase\ifx\AtBegShi@Memoir@shipiiA\mem@shipii
                \z@
              \else
                \ifx\AtBegShi@Memoir@shipiiB\mem@shipii
                  \z@
                \else
                  \@ne
                \fi
              \fi
        \let\AtBegShi@found\relax
        \ifx\shipout\AtBegShi@Memoir@shipout
          \def\AtBegShi@found{\shipout}%
```

```
        \else\ifx\AtBegShi@OrgShipout\AtBegShi@Memoir@shipout
          \def\AtBegShi@found{\AtBegShi@OrgShipout}%
        \else\ifx\CROP@shipout\AtBegShi@Memoir@shipout
          \def\AtBegShi@found{\CROP@shipout}%
        \else\ifx\GPTorg@shipout\AtBegShi@Memoir@shipout
          \def\AtBegShi@found{\GPTorg@shipout}%
        \else\ifx\THBorg@shipout\AtBegShi@Memoir@shipout
          \def\AtBegShi@found{\THBorg@shipout}%
        \else\ifx\@EveryShipout@Org@Shipout\AtBegShi@Memoir@shipout
          \def\AtBegShi@found{\@EveryShipout@Org@Shipout}%
        \fi\fi\fi\fi\fi\fi
        \ifx\AtBegShi@found\relax
        \else
          \expandafter\endgroup
          \expandafter\def\AtBegShi@found{%
            \edef\AtBegShi@GroupLevel{\number\currentgrouplevel}%
            \afterassignment\mem@shipi
            \setbox\AtBeginShipoutBox=%
          }%
          \def\mem@shipi{%
            \ifnum\AtBegShi@GroupLevel=\currentgrouplevel
            \else
              \expandafter\aftergroup
            \fi
            \mem@shipii
          }%
          \def\mem@shipii{%
            \ifvoid\AtBeginShipoutBox
            \else
              \setbox\AtBeginShipoutBox=\vbox{%
                \trimmarks
                \ifvbox\AtBeginShipoutBox
                  \unvbox\AtBeginShipoutBox
                \else
                  \box\AtBeginShipoutBox
                \fi
              }%
              \expandafter\mem@oldshipout
              \expandafter\box
              \expandafter\AtBeginShipoutBox
            \fi
          }%
          \@PackageInfoNoLine{atbegshi}{Class `memoir' patched}%
          \begingroup
        \fi
      \fi
    \fi
  \endgroup
  \let\AtBegShi@PatchMemoir\relax
}
\@ifclassloaded{memoir}{%
  \AtBegShi@PatchMemoir
}{%
  \AtBeginDocument{\AtBegShi@PatchMemoir}%
}
```

```
\AtBegShi@End
```

⟨/package⟩

4 Test

⟨*test1⟩

```
\input atbegshi.sty\relax
\def\msg#{\immediate\write16}
\msg{File: atbegshi-test1.tex 2007/04/27 v1.4 Test file for plain-TeX}
\def\testmsg#1#2{%
  \msg{}%
  \msg{*** Test with box (#1), expected page output [#2]}%
}

\newbox\voidbox
\def\void{\box\voidbox}
\begingroup
  \setbox\voidbox=\void
\endgroup

\count0=0\relax
\AtBeginShipout{%
  \global\advance\count0 by 1\relax
  \msg{* Inside \string\AtBeginShipout: [\the\count0]}%
}

\AtBeginShipoutFirst{%
  \msg{* Inside \string\AtBeginShipoutFirst}%
  Hello World%
}

\testmsg{\string\null}{1}
\shipout\null

\AtBeginShipoutFirst{%
  This is too late%
}

\testmsg{void}{}
\shipout\void

\testmsg{\string\copy255 (not void)}{2}
\setbox255\hbox{\vrule height 10bp width 10bp}
\shipout\copy255 %

\testmsg{\string\copy255 (again)}{3}
\shipout\copy255 %

\testmsg{\string\box255}{4}
\shipout\box255 %

\testmsg{\string\box255 (again)}{}
\shipout\box255 %

\testmsg{\string\hbox}{5}
\shipout\hbox{\vrule height 5bp width 20bp}

\testmsg{\string\vbox}{6}
\shipout\vbox{\hrule height 20bp width 5bp}

\testmsg{\string\null, voided by hook}{}
\def\VoidBox{%
  \begingroup
    \setbox\AtBeginShipoutBox=\box\AtBeginShipoutBox
  \endgroup
}
\AtBeginShipout{\VoidBox}
\shipout\null
```

```
\def\VoidBox{}

\msg{*** \string\begingroup}
\begingroup
  \testmsg{void}{}%
  \shipout\void
\msg{*** \string\endgroup}
\endgroup

\msg{*** \string\begingroup}
\begingroup
  \testmsg{void}{}%
  \shipout\void
  \testmsg{\string\null}{8}%
  \shipout\null
\msg{*** \string\endgroup}
\endgroup

\testmsg{output routine}{9}
Hello World
\vfill
\eject

\testmsg{\string\null\space(discarded)}{}
\AtBeginShipout{%
  \msg{* Inside \string\AtBeginShipout: DISCARD}%
  \AtBeginShipoutDiscard
}
\shipout\null

\end
⟨/test1⟩
⟨*test2⟩
\NeedsTeXFormat{LaTeX2e}
\ProvidesFile{atbegshi-test2.tex}[2007/04/27 v1.4 Test file for LaTeX]
\RequirePackage{color}
\pagecolor{yellow}
\documentclass[a5paper,showtrims]{memoir}
\usepackage{atbegshi}
\AtBeginShipout{%
  \setbox\AtBeginShipoutBox=\vbox{%
    \vbox to 0pt{%
      \kern-1.5in %
      \hbox to 0pt{%
        \kern-1.5in %
        \color{blue}%
        \rule{1in}{1in}%
        \hss
      }%
      \vss
    }%
    \hrule
    \hbox{\vrule\box\AtBeginShipoutBox\vrule}%
    \hrule
  }%
}
\usepackage{eso-pic}
\makeatletter
\@EveryShipout@Init
\let\@EveryShipout@Init\relax
\makeatother
\AddToShipoutPicture{%
```

```
  \hspace{.52\paperwidth}%
  \colorbox{cyan}{%
    \rule{0mm}{\paperheight}%
    \hspace{.48\paperwidth}%
  }%
}
\usepackage[color=red,cross,a4,center]{crop}
\begin{document}
\shipout\null
\shipout\box\csname voidb@x\endcsname
\section{Hello World}
\end{document}
```
⟨/test2⟩

5 Installation

5.1 Download

Package. This package is available on CTAN[1]:

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
```

[1] ftp://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 pdfLATEX:

```
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 ε-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.

U

V

W

X

Z