

The dashrule package*

Scott Pakin
scott+dash@pakin.org

March 28, 2013

Abstract

The dashrule package makes it easy to draw a huge variety of dashed rules (i.e., lines) in L^AT_EX. dashrule provides a command, `\hdashrule`, which is a cross between L^AT_EX's `\rule` and PostScript's `setdash` command. `\hdashrule` draws horizontally dashed rules using the same syntax as `\rule` but with an additional, `setdash`-like parameter that specifies the pattern of dash segments and the space between those segments. Because dashrule's rules are constructed internally using `\rule` (as opposed to, e.g., PostScript `\specials`) they are fully compatible with every L^AT_EX back-end processor.

1 Usage

`\hdashrule` L^AT_EX's `\rule` command draws a rectangular blob of ink with a given width, height, and distance above the baseline. The dashrule package introduces an analogous command, `\hdashrule`, which draws the same blob of ink, but horizontally dashed. `\hdashrule` takes five parameters, two of which are optional:

| |
|---|
| <code>\hdashrule</code> [<i>raise</i>] [<i>leader</i>] { <i>width</i> } { <i>height</i> } { <i>dash</i> } |
|---|

The *raise*, *width*, and *height* parameters have the same meaning as in L^AT_EX's `\rule` macro: the distance to raise the rule above the baseline and the width and height of the rule.

Because `\hdashrule` is implemented in terms of T_EX's primitive leader commands (`\leaders`, `\cleaders`, and `\xleaders`), the dash pattern must be repeated an integral number of times. *leader* specifies what to do with the extra whitespace (always less than the width of the dash pattern) that this requirement introduces. The default, which corresponds to T_EX's `\leaders` command, adds space to both ends of the rule so the dash patterns from multiple `\hdashrules` line up. If *leader* is `c`, which corresponds to T_EX's `\cleaders` command, an equal amount of whitespace is added to both ends of the rule. If *leader* is `x`, which corresponds to

*This document corresponds to dashrule v1.3, dated 2013/03/28.

TeX's `\xleaders` command, the whitespace is divided up, and the same amount of whitespace separates each repetition of the dash pattern.

The $\langle dash \rangle$ argument specifies the dash pattern and is analogous to the *array* argument to PostScript's `setdash` function. That is, it is a list of space-separated $\langle dimen \rangle$ s that alternate “on” and “off” distances. For instance, “`2pt 1pt`” means a 2 pt. rule, followed by a 1 pt. gap, followed by a 2 pt. rule, followed by a 1 pt. gap, and so forth. An odd number of $\langle dimen \rangle$ s is no different; “`2pt`” alternates 2 pt. rules and 2 pt. gaps, and “`1pt 2pt 3pt`” repeats “1 pt. rule, 2 pt. gap, 3 pt. rule, 1 pt. gap, 2 pt. rule, 3 pt. gap.” As a special case, an empty $\langle dash \rangle$ argument draws a solid rule.

2 Examples

The following are some typical ways to use `\hdashrule`. Each example changes from the previous in only one parameter. For clarity, underlines are used to indicate modified text, and the rule is bracketed by an upper- and lowercase “X”.

| | |
|---|---------------|
| <code>\rule{2cm}{1pt}</code> | X_____x |
| <code>\hdashrule{2cm}{1pt}{}</code> | X_____x |
| <code>\hdashrule{2cm}{1pt}{1pt}</code> | X.....x |
| <code>\hdashrule{4cm}{1pt}{1pt}</code> | X.....x |
| <code>\hdashrule[0.5ex]{4cm}{1pt}{1pt}</code> | X.....x |
| <code>\hdashrule[0.5ex]{4cm}{1pt}{3mm}</code> | X - - - - - x |
| <code>\hdashrule[0.5ex]{4cm}{1mm}{3mm}</code> | X ■ ■ ■ ■ ■ x |
| <code>\hdashrule[0.5ex]{4cm}{1mm}{3mm 3pt}</code> | X ■■■■■■■■■ x |
| <code>\hdashrule[0.5ex]{4cm}{1mm}{% 3mm 3pt 1mm 2pt}</code> | X ■■■■■■■■■ x |

These next examples show the effect of using different leader types. Each leader is used with both a 4 cm wide rule and a 3 cm wide rule.

| | |
|--|---------------|
| <code>\hdashrule[0.5ex]{4cm}{1mm}{8mm 2pt}</code> | X ■■■■■■■■■ x |
| <code>\hdashrule[0.5ex]{3cm}{1mm}{8mm 2pt}</code> | X ■■■■■ x |
| <code>\hdashrule[0.5ex][c]{4cm}{1mm}{8mm 2pt}</code> | X ■■■■■■■■■ x |
| <code>\hdashrule[0.5ex][c]{3cm}{1mm}{8mm 2pt}</code> | X ■■■■■ x |
| <code>\hdashrule[0.5ex][x]{4cm}{1mm}{8mm 2pt}</code> | X ■■■■■■■■■ x |
| <code>\hdashrule[0.5ex][x]{3cm}{1mm}{8mm 2pt}</code> | X ■■■■■ x |

Notice how the dashes in the first pair of `\hdashrules` line up; the rules in the second pair each have an equal amount of whitespace on either side of the rule; and the rules in the third pair have extra spaces within the dash pattern itself instead of around it. The `x` qualifier is rarely useful for dashed rules because it alters the pattern itself. However, `x` does enable rules with long dashes to better fill a comparatively small width, as in the following example:

```
\hdashrule[0.5ex][x]{3in}{2pt}{2cm 0pt}
X  _____ x
```

The gaps in the above are clearly wider than `0pt`, but they *are* evenly spaced.

3 Differences from `setdash`

`\hdashrule` is different from PostScript's `setdash` command in the following ways:

- `setdash` takes on/off values in terms of PostScript points (TeX “big points” or “bp”), while `\hdashrule` requires explicit units.
- There is no equivalent of `setdash`'s *offset* parameter to specify a starting offset into the pattern. If you're desperate you can fake *offset* with a leading `\rule` and `\hspace`.

4 Implementation

We load the `ifmtarg` package to help check if the final argument to `\hdashrule` is empty.

```
1 \RequirePackage{ifmtarg}
```

`\hdr@do@rule` This macro is exactly like L^AT_EX's `\rule` except that the optional argument is required, and it has the side effect of pointing `\hdr@do@something` to `\hdr@do@skip`.

```
2 \def\hdr@do@rule[#1]#2#3{%
3   \rule[#1]{#2}{#3}%
4   \let\hdr@do@something=\hdr@do@skip
5 }
```

`\hdr@do@skip` This macro takes the same arguments as `\hdr@do@rule`, but instead of drawing a rule, it inserts an equivalent amount of horizontal whitespace. Additionally, it points `\hdr@do@something` to `\hdr@do@rule` as a side effect.

```
6 \def\hdr@do@skip[#1]#2#3{%
7   \hspace*{#2}%
8   \let\hdr@do@something=\hdr@do@rule
9 }
```

`\c@hdr@segments` Dash patterns containing an odd number of segments are treated differently from dash patterns containing an even number of segments. We therefore define a macro, `\hdr@tally@segments`, which counts the number of space-separated segments in a dash pattern and stores the tally in the `hdr@segments` counter. Note that `hdr@segments` should be initialized to 0 before invoking `\hdr@tally@segments`.

```

10 \newcounter{hdr@segments}
11 \def\hdr@tally@segments#1 {%
12   \ifx#1!%
13   \else
14     \addtocounter{hdr@segments}{1}%
15     \expandafter\hdr@tally@segments
16   \fi
17 }
```

`\hdashrule` This is the only macro in `dashrule`'s external interface. (`\hdashrule@ii` does most of the work for `\hdashrule`, though.) All `\hdashrule` itself does is invoke `\hdashrule@i` with its first optional argument or `0.0pt` if none was provided. `\hdashrule@i`, in turn, invokes `\hdashrule@ii` with the two optional arguments, supplying `\empty` as the default value of the second optional argument.

```

18 \DeclareRobustCommand{\hdashrule}{\mbox{}}\@testopt{\hdashrule@i}{0pt}}
```

`\hdashrule@i` Supply `\empty` as the default second argument and call `\hdashrule@ii`.

```

19 \def\hdashrule@i[#1]{\@testopt{\hdashrule@ii[#1]}\empty}
```

`\hdashrule@ii` Now we can do the real work for `\hdashrule`. `\hdashrule@ii` takes the following parameters:

| | | | | |
|------------------------------|-------------------------------|------------------------------|-------------------------------|-----------------------------|
| #1 | #2 | #3 | #4 | #5 |
| <code>[<raise>]</code> | <code>[<leader>]</code> | <code>{<width>}</code> | <code>{<height>}</code> | <code>{<dash>}</code> |

The `<raise>`, `<width>`, and `<height>` parameters have the same meaning as in L^AT_EX's `\rule` macro. `<leader>` specifies the T_EX leader function to use to fill `<width>` amount of space. It should be `c` for `\cleaders`, `x` for `\xleaders`, or nothing for ordinary `\leaders`. The `<dash>` argument specifies the dash pattern and is analogous to the `array` argument to PostScript's `setdash` function. That is, it is a list of space-separated `<dimen>`s that alternate "on" and "off" distances.

```

20 \def\hdashrule@ii[#1][#2]#3#4#5{%
```

If the final argument, `<dash>`, is empty, we fall back to using an ordinary `\rule` command. This is not terribly useful in practice but does make `\hdashrule` behave more like PostScript's `setdash`.

```

21   \ifmtarg{#5}{%
22     \rule[#1]{#3}{#4}%
23   }{%
```

Here begins the common case, in which the `<dash>` argument is nonempty.

`\hdr@do@something` The `\hdr@do@something` alias alternates between `\hdr@do@rule` and `\hdr@do@skip`, starting with `\hdr@do@rule`.

```
24 \let\hdr@do@something=\hdr@do@rule
```

`\hdr@parse@dash` For every space-separated $\langle dimen \rangle$ in $\langle dash \rangle$, we invoke `\hdr@do@something` to draw a rule or a space, as appropriate. We define `\hdr@parse@dash` within `\hdashrule@ii` so we don't have to pass in `\hdashrule@ii`'s `#1` and `#4` on every invocation.

```
25 \def\hdr@parse@dash##1 {%
26   \ifx##1!%
27   \else
28     \hdr@do@something[#1]{##1}{#4}%
29     \expandafter\hdr@parse@dash
30   \fi
31 }%
```

We now count the number of segments in the dash pattern so we can determine if we have an even or odd number of them.

```
32 \setcounter{hdr@segments}{0}%
33 \hdr@tally@segments#5 !
```

Finally, we invoke `\leaders`, `\cleaders`, or `\xleaders` to draw the dashed line, repeating the pattern until $\langle width \rangle$ space is filled. The trick here is that odd-lengthed pattern descriptions must be repeated to yield the complete pattern. For instance, the pattern “1pt” is actually short for “1 pt. rule, 1 pt. space,” and “2pt 4pt 6pt” is an abridged version of “2 pt. rule, 4 pt. space, 6 pt. rule, 2 pt. space, 4 pt. rule, 6 pt. space.” Although it is valid to repeat even-lengthed patterns as well—an earlier draft of `\hdashrule@ii` did just that—this produces inferior results because T_EX's various leader commands do not split boxes. The longer the pattern, the less likely it will fit snugly into the given width.

```
34 \ifodd\c@hdr@segments
35   \csname#2leaders\endcsname
36   \hbox{\hdr@parse@dash#5 #5 ! }%
37   \hskip#3%
38 \else
39   \csname#2leaders\endcsname
40   \hbox{\hdr@parse@dash#5 ! }%
41   \hskip#3%
42 \fi
43 \mbox{}%
44 }%
45 }
```

5 Future Work

`dashrule v1.3` supports only horizontally dashed rules. Future versions (if any) may support vertically dashed rules as well. For the time being, the `graphicx` package's

`\rotatebox` can be used to define a `\vdashrule` in terms of a rotated `\hdashrule`.

The next logical step after adding a `\vdashrule` is to support dashed rectangles, which would be composed of `\hdashrules` and `\vdashrules`. Other possible enhancements would be a way of drawing dotted lines, presumably composed from the limited set of circle characters available in L^AT_EX's fonts.

Change History

| | | |
|---|------|---|
| v1.0 | | with an empty box to enable |
| General: Initial version | 1 | <code>\hdashrule</code> to work within a |
| v1.1 | | <code>tabular</code> cell. Thanks to Piazza |
| <code>\hdashrule</code> : Preceded the invo- | | Luca for the bug report |
| cation of <code>\@testopt</code> with an | v1.3 | |
| empty box so that <code>\hdashrule</code> | | <code>\hdr@parse@dash</code> : Prevented spu- |
| can now begin a paragraph . . . | 4 | rious spaces from being intro- |
| v1.2 | | duced into the output. Thanks |
| <code>\hdr@parse@dash</code> : Ended the rule | | to Halil Şen for the bug report . . |
| | | 5 |

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 | | H | | 4, 8, <u>24</u> , 28 |
| <code>\@ifmtarg</code> | 21 | <code>\hdashrule</code> | <u>18</u> | <code>\hdr@parse@dash</code> . . . | <u>25</u> |
| <code>\@testopt</code> | 18, 19 | <code>\hdashrule@i</code> | 18, <u>19</u> | <code>\hdr@tally@segments</code> | |
| | | <code>\hdashrule@ii</code> . . . | 19, <u>20</u> | | <u>10</u> , 33 |
| | | <code>\hdr@do@rule</code> . . . | <u>2</u> , 8, 24 | | |
| | C | <code>\hdr@do@skip</code> | 4, <u>6</u> | R | |
| <code>\c@hdr@segments</code> . | <u>10</u> , 34 | <code>\hdr@do@something</code> .. | | <code>\RequirePackage</code> | 1 |
| | | | | <code>\rule</code> | 3, 22 |