

The `arydshln` package*

Hiroshi Nakashima
(Kyoto University)

2018/09/26

Abstract

This file gives L^AT_EX's `array` and `tabular` environments the capability to draw horizontal/vertical dash-lines.

Contents

1	Introduction	2
2	Usage	2
2.1	Loading Package	2
2.2	Basic Usage	3
2.3	Style Parameters	3
2.4	Fine Tuning	4
2.5	Finer Tuning	4
2.6	Performance Tuning	5
2.7	Compatibility with Other Packages	6
3	Known Problems	8

*This file has version number v1.74, last revised 2018/09/26.

1 Introduction

In January 1993, Weimin Zhang kindly posted a style `hvdashln` written by the author, which draws horizontal/vertical dash-lines in L^AT_EX's `array` and `tabular` environments, to the news group `comp.text.tex`. The style, unfortunately, has a known problem that vertical lines are broken when an array contains tall rows.

In March of the year, Monty Hayes complained of this problem encouraging the author to make a new version `arydshln` to solve the problem. The new style also has new features, such as allowing `:` to specify a vertical dash-line in preamble, and `\cdashline` being a counterpart of `\cline`.

In March 1999, Sebastian Rahtz kindly invited the style, which had been improved following the bug report from Takahiro Kubota, to be included in T_EX CTAN and also in the online catalogue compiled by Graham Williams. This invitation gave the style new users including Peter Ehrbar who wished to use it with `array` style in Standard L^AT_EX Tools Bundle and had trouble because these styles were incompatible with each other. Therefore, the style became compatible with `array` and got additional new features.

In February 2000, Zsuzsanna Nagy reported that `arydshln` is not compatible with `colortab` style to let the author work on the compatibility issue again.

In February 2001, Craig Leech reported another compatibility problem with `longtable`. Although the author promised that the problem would be attacked some day, the issue had left long time¹ until three other complaints were made. Then the author attacked the problem hoping it is the last compatibility issue².

In May 2004, Klaus Dalinghaus found another incompatibility with `colortbl`. Although he was satisfied by a quick hack for cell painting, the author attacked a harder problem for line coloring to solve the problem³.

2 Usage

2.1 Loading Package

The package is usable to both L^AT_EX 2_ε and L^AT_EX-2.09 users with their standard package loading declaration. If you use L^AT_EX 2_ε, simply do the following.

```
\usepackage{arydshln}
```

If you still love L^AT_EX-2.09, the following is what you have to do.

```
\documentstyle[...arydshln,...]{<style>}
```

Only one caution given to users of `array` (v2.3m or later) and `longtable` (v4.10 or later) packages, included in Standard L^AT_EX Tools Bundle, and `colortab` and `colortbl` package is that `arydshln` has to be loaded *after* `array`, `longtable`, `colortab` and/or `colortbl` done. That is, the following is correct but reversing the order of `\usepackage` will cause some mysterious error.

¹Two years and a half! Sorry Craig.

²But his hope was dashed as described below.

³Without dreaming it is the last compatibility issue.

```

\usepackage{array}      % and/or
\usepackage{longtable} % and/or
\usepackage{colortab}  % or
\usepackage{colortbl}
\usepackage{arydshln}

```

2.2 Basic Usage

`array` You can simply use `array` or `tabular(*)` environments with standard preamble, such as `{r|c|l}`, and standard commands `\`, `\hline`, `\cline` and `\multicolumn`.

`tabular` : Drawing a vertical dash-line is quite simple. Use ‘:’ in the preamble as the separator of columns separated by the dash-line, just like using ‘|’ to draw a vertical solid-line. The *preamble* means not only that of the environment, but also the first argument of `\multicolumn`.

`\hdashline` It is also simple to draw a horizontal dash-line. Use `\hdashline` and `\cdashline` as the counterparts of `\hline` and `\cline`.

For example;

```

\begin{tabular}{|l::c:r|}\hline
A&B&C\\\hdashline
AAA&BBB&CCC\\\cdashline{1-2}
\multicolumn{2}{|l:}{AB}&C\\\hdashline\hdashline
\end{tabular}

```

will produce the following result.

Note that the intersections of leftmost/rightmost vertical lines and horizontal dash-lines are little bit different from those produced by ordinary `array/tabular`. That is, with very careful examination you will find that vertical lines of ordinary ones are *broken* with small white specks at intersections, while in the example above they have no specks. In addition, the four corners of outermost rectangular also have specks in ordinary ones, while those in the example above have perfect contacts of L-shape⁴.

`\firsthdashline` If you use `array`, the dashed version of `\firsthline` and `\lasthline` named `\firsthdashline` and `\lasthdashline` are available.

2.3 Style Parameters

`\dashlinedash` You have two style parameters to control the shape of dash-lines: `\dashlinedash` is for the length of each dash segment in a dash line; `\dashlinegap` controls the amount of each gap between dash segments. Both parameters have a common default value, 4 pt.

⁴The top-left/right corners had specks before v1.73, the fix in which made the topmost dash segment of a vertical dash-line a little bit shorter.

2.4 Fine Tuning

- Although you can control the shape of dash-lines in an `array`/`tabular` environment as described in §2.3, you might want to draw a dash-line of a shape different from others. To specify the shape of a vertical dash-line explicitly, you may use;

```
;\langle dash\rangle/\langle gap\rangle}
```

instead of ordinary ‘:’ and will have a dash-line with dash segments of $\langle dash \rangle$ long separated by spaces of $\langle gap \rangle$.

`\hdashline` As for horizontal dash-lines, explicit shape specifications may be given through optional arguments of `\hdashline` and `\cdashline` as follows.

```
\hdashline[\langle dash\rangle/\langle gap\rangle]
\cdashline{\langle col1\rangle-\langle col2\rangle}[\langle dash\rangle/\langle gap\rangle]
```

For example;

```
\begin{tabular}{|l:c;{2pt/2pt}r|}\hline
A&B&C\\hdashline[1pt/1pt]
AAA&BBB&CCC\\cdashline{1-2} [.4pt/1pt]
\multicolumn{2}{|l;{2pt/2pt}}{AB}&C\\hdashline\hdashline
\end{tabular}
```

will produce the following result.

A	B	C
AAA	BBB	CCC
AB		C

`\ADLnullwide` The vertical solid and dashed lines are drawn as if their width is zero, as standard `\ADLsomewide` \LaTeX 's `array` and `tabular` do, if you don't use `array` package. Otherwise, they have *real* width of `\arrayrulewidth` as the authors of `array` prefers. However, you may explicitly tell `arydshln` to follow your own preference by `\ADLnullwide` if you love \LaTeX standard, or `\ADLsomewide` if you second the preference of `array` authors.

2.5 Finer Tuning

To draw dash-lines, we use a powerful primitive of \TeX called `\xleaders`. It replicates a segment that consist of a dash and gap so that a dash-line has as many segments as possible and distributes *remainder* space to make the spaces between adjacent dash segments (almost) equal to each other. Therefore, you will have dash-lines with consistent steps of gaps and spaces the lines in Figure 1(1) are.

However, because of a bug (or buggy feature) of `\xleaders`, there *had been* a small possibility that a dash segment near the right/bottom end drops, until it was fixed in the version of 3.141592⁵. Though the fix ultimately made any effort to cope with the problem unnecessary, the package still gives you alternative *drawing modes* which you may specify by `\ADLdrawingmode{\langle m \rangle}` as follows.

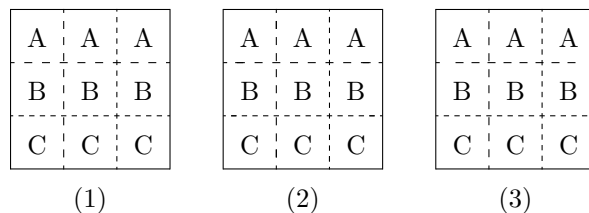


Figure 1: Drawing mode controlled by `\ADLdrawingmode`

`\ADLdrawingmode`

- $m = 1$
As shown in Figure 1(1), it gives most beautiful result by `\xleaders`⁶. This is default.
- $m = 2$
As shown in (2) of the figure, beautiful if dash-lines are not so sparse as right/lower lines, but dash segments near the both ends may be a little bit too long as left/upper lines, because in this mode the second first/last segments are drawn by a special mechanism.
- $m = 3$
As shown in (3) of the figure, beautiful if dash-lines are not so sparse as right/lower lines, but gaps near the both ends may be considerably too large as left/upper lines, because in this mode the lines are drawn by `\cleaders`.

It is strongly recommended to use default mode 1 unless you want to have some special effect.

2.6 Performance Tuning

Since drawing dash-lines is a hard job, you have to be patient with the fact that the performance of typesetting `array/tabular` with dash-lines is poorer than that of ordinary ones. In fact, according to author's small performance evaluation with a `tabular` having nine vertical and ten horizontal dash-lines, typesetting the `tabular` is approximately ten times as slow as its ordinary counterpart with solid lines.

However, this is not a really bad news, unfortunately. The real one is that loading `arydshln` makes typesetting `array/tabular` slower even if they only have solid lines which the package treats as special ones of dash-lines. The evaluation result shows the degradation factor is about nine. Therefore, if your document has many `array/tabular` with solid lines, `LATEX` will run slowly even with quite few (or no) `array/tabular` with dash-lines,

`\ADLinactivate`

To cope with this problem, you may inactivate dash-line functions by the command `\ADLinactivate` that replaces dash-lines with solid lines drawn by a faster (i.e. ordinary) mechanism. Although the inactivation does not completely solve the performance problem, the degradation factor will become much smaller and acceptable, approximately 1.5 in

⁵By pointing out this problem, the author got a check of \$327.68 plus a significantly large amount of interest from DEK. Wow!!

⁶Until the fix of `\xleaders`, the second bottom/rightmost segments of right/lower lines were dropped.

the author's evaluation. For example, the draft version of your document will have the command in its preamble, which you will remove from your final version.

`\ADLactivate` Alternatively, you may do `\ADLinactivate` in the preamble, switch on by `\ADLactivate` before you really need dash-lines, and switch off again afterword. A wiser way could be surrounding `array/tabular` by `\begin{ADLactivate}` and `\end{ADLactivate}`.

`Array` If you feel it tiresome to type the long command/environment name for the activation,
`Tabular` you may use `Array` and `Tabular(*)` environment in which dash-line functions are always active. Note that, however, since these environment names are too natural to keep them from being used by authors of other packages or yourself, name conflict could occur. If `Array` and/or `Tabular` have already been defined when `arydshln` is loaded, you will get a warning to show you have to define new environments, say `dllarray` and `dltabular`, as follows.

```
\newenvironment{dllarray}{\ADLactivate\begin{array}}%
{\end{array}}
\newenvironment{dltabular}{\ADLactivate\begin{tabular}}%
{\end{tabular}}
\newenvironment{dltabular*}{\ADLactivate\begin{tabular*}}%
{\end{tabular*}}
```

`\ADLnoshorthanded` On the other hand, if they are defined after `arydshln` is loaded, their definitions are silently replaced or L^AT_EX complains of multiple definitions. The error in the latter case will be avoided by putting `\ADLnoshorthanded` just after `\usepackage{arydshln}`.

2.7 Compatibility with Other Packages

Users of `array` package may use all of newly introduced preamble characters, such as `'>`, `'<`, `'m`, `'b`, and all the commands such as `\extrarowheight`, `\firsthline` and `\lasthline`. The preamble characters given by `arydshln` may be included in the second argument of `\newcolumnstype`.

Also users of `colortab` package may use `\LCC/\ECC` construct to color columns. A horizontal solid/dash line may be colored by, e.g. `\NAC\hdashline\ENAC`. The pair of `\AC` and `\EAC` may be used to color everything between them *but*, unfortunately, vertical lines are not. There are no ways to color vertical lines in a table having dash lines. You may color vertical lines of an ordinary table inactivating dash line functions by `\ADLinactivate`.

Another (and more convenient) table coloring tool `colortbl` may be also used simply by loading it before `arydshln`. Not only the painting commands `\rowcolor`, `\columncolor` and `\cellcolor` work well, but both solid and dash lines are also colored by the command `\arrayrulecolor` of `colortbl`⁷. One caution is that `\arrayrulecolor` defines the color of the dash-part of dash lines and thus gap-part has no color (i.e. color of the paper on which the line drawn). Therefore, if you have a `tabular` like;

```
\begin{tabular}{|>\columncolor{red}l:>\columncolor{green}r|}
...
\end{tabular}
```

⁷The `colortbl` manual says `\arrayrulecolor` and `\doublerulesepcolor` may be in `>{...}` in a preamble but they cause an error with the original implementation. This bug is fixed in `arydshln` and they are now usable to specify the color of the vertical (dash) lines whose specifications occur after the commands.

you will find the vertical dash line is a sequence of black (or the color of `\arrayrulecolor`) and white segments. This problem is partly solved by declaring `\ADLnullwide`⁸ to conjunct the red and blue columns and to draw the dash line on their border.

`\ADLnullwidehline`
`\ADLsomewidehline`

Unfortunately, however, `\ADLnullwide` does not affect the real width of horizontal (dash) lines and thus you will still see white gaps in `\hdashline` and `\cdashline`. A solution is to put `\ADLnullwidehline` before you start a `array/tabular`⁹. With this command, a horizontal (dash) line is drawn adjusting its bottom edge to that of the row above. The command `\ADLsomewidehline` turns the switch to default and the top edge of a horizontal (dash) line will be adjusted to the bottom edge of the row above.

`\dashgapcolor`
`\nodashgapcolor`

Another method to avoid white gaps is to give a color to gaps by `\dashgapcolor` with arguments same as `\color`. For example;

```
\arrayrulecolor{green}\dashgapcolor[rgb]{1,1,0}
```

makes colorful dash lines with green dashes and yellow gaps. The command can be placed outside of `array/tabular` for dash lines in the environment, in the argument of preamble character `>` for vertical dash lines following them, or at the beginning of a row for horizontal dash lines following the command. The command `\nodashgapcolor` (no arguments) nullifies the effect of `\dashgapcolor`. Note that `\nodashgapcolor` is different from `\dashgapcolor{white}` because the former makes gaps *transparent* while the later whiten them.

`longtable`
`Longtable`

Usage of `longtable` with `arydshln` is quite simple. Just loading `arydshln` after `longtable` is enough to make the `longtable` environment able to draw dash-lines. A shorthand activation of dash-line functions is also available by `Longtable` environment. One caution to `longtable` users is that the temporary results before the *convergence* of the column widths may be different from those without `arydshln`. For example, the following is the first pass result of the example shown in Table 3 of the `longtable` manual.

1	2	3
wide multicolumn spanning 1-3		
multicolumn 1-2		3
wide 1	2	3

Since `LTchunksize` is one in the example, columns of each row has their own widths and thus has vertical lines drawn at the edges of the columns. On the other hand, you will have the following as the first pass result with `arydshln`.

1	2	3
wide multicolumn spanning 1-3		
multicolumn 1-2		3
wide 1	2	3

As you see, the vertical lines are drawn at the column edges of the last row¹⁰ because `arydshln` draws them when it see the last row. Anyway, you may ignore temporary results and will have a compatible result when the column widths are converged like the following.

⁸Since `colortbl` automatically loads `array`, the default is `\ADLsomewide`

⁹This command also makes `\cline` and `\cdashline` visible even if the row below is painted.

¹⁰More precisely, drawn according to the column widths established by all the chunks preceding page output.

1	2	3
wide multicolumn spanning 1-3		
multicolumn 1-2		3
wide 1	2	3

3 Known Problems

There are following known problems.

1. The new preamble specifiers ‘:’ and ‘;{*dash*}/<gap>’ cannot be followed or preceded by ‘@{*text*}’, or you will have an ugly result. More specifically, a specifier to draw a dash-line at the left edge of a column cannot be preceded by ‘@{*text*}’, while that to draw at the right edge cannot be followed by ‘@{*text*}’.
2. If you use `array` package, the restriction of ‘@’ shown above is also applied to ‘!’.
3. In order to make it sure that a dash-line always *touches* its both end, i.e. a dash-line always begins and ends with a dash segment, the amount of a gap will slightly vary depending on the dash-line length.
4. If a dash-line is too short, you will have an ugly result without overfull message. More specifically, in mode 1 or 3, a line will look to protrude beyond its column/row borders if it is shorter than a half of `\dashlinedash`. In mode 2, the minimum length to avoid the protrusion is $1.5 \times \text{\dashlinedash} + \text{\dashlinegap}$.
5. As described in §2.6, the processing speed for `array` and `tabular` environment will become slower even if dash-lines are not included.
6. As described in §2.7, `\AC` and `\EAC` pair of `colortab` such as `\AC&\EAC` cannot color the vertical line at `&`. Use `\ADLinactivate` if you want to have an ordinary table with colored vertical lines. Note that you may color vertical lines with `colortbl` package.
7. There should be a number of packages whose own `array`/`tabular` implementations are not compatible with `arydshln`, though the author has made efforts at the compatibility. One of them is `plext` package for Japanese typesetting but it has a style file named `plextarydshln.sty` to solve the compatibility issue. So if you use the functionality of `arydshln` with `plext`, do `\usepackage{plextarydshln}` instead of `\usepackage{arydshln}`.

Acknowledgments

The author thanks to Monty Hayes who gave the author the opportunity to make this style, and Weimin Zhang and Takahiro Kubota who pointed out bugs in early versions. He also thanks to the following people; Sebastian Rahtz and Graham Williams who kindly invited the style to T_EX CTAN and online catalogue compiled by Graham; Peter Ehrbar who showed the style was incompatible with `array` and kindly accepted the offer to be an alpha-user of v1.4 alone; Zsuzsanna Nagy who reported another incompatibility problem with

`colortab`; Ralf Heydenreich who reported the bug causing that glues in a column have no effect; Yaxin Liu who reported the incompatibility bug of `array` and `\ADLinactivate`; Craig Leech who reported the incompatibility problem with `longtable`, which was also reported by Uwe Jehmlich, Torge Thielemann and Florian Weig, and had waited for two years and a half (!) for the solution; Klaus Dalinghaus who reported yet another incompatibility with `colortbl`; Morten Høgholm who reported the bug of m-type columns of `array` which had not manifested in five (!!) years since the author released the first `array`-compatible version; Maïeul Rouquette who reported another bug of m-type columns of `longtable` with `array` which had peacefully hidden in the package for eleven years and a half (!!!) since the author made the bug fix shown above carelessly, yet another bug related to `longtable`, and most surprisingly a problem on intersections of horizontal and vertical (dash-)lines which has hidden for 23 years (!!!!) since the very first version of the package; and Hironobu Yamashita who pointed out bugs hidden for 19 years (!!!!) by which `delarray` did not work.

The base implementation of `array` and `tabular` environments, part of which the author gives new definitions referring original ones, are written by Leslie Lamport as a part of \LaTeX-2.09 and $\text{\LaTeX 2}_{\epsilon}$ (1997/12/01) to which Johannes Braams and other authors also contributed. The author also refers `array` package (v2.4c) written by Frank Mittelbach and David Carlisle; `colortab` package (v0.9) written by Timothy van Zandt; and `longtable` (v4.10) and `colortbl` (v0.1j) packages written by David Carlisle; to make the style compatible with those packages.

Index

Italicized number refers to the page where the specification of corresponding entry are described.

Symbols		<code>longtable</code>	7
<code>:</code>	3	<code>Tabular</code>	6
<code>;</code>	4	<code>tabular</code>	3
		<code>\extrarowheight</code>	6
A		F	
<code>\AC</code>	6	<code>\firsthdashline</code>	3
<code>\ADLactivate</code>	6	<code>\firsthline</code>	6
<code>\ADLdrawingmode</code>	5	H	
<code>\ADLinactivate</code>	5	<code>\hdashline</code>	3, 4
<code>\ADLnoshorthanded</code>	6	<code>\hline</code>	3
<code>\ADLnullwide</code>	4	L	
<code>\ADLnullwidehline</code>	7	<code>\lasthdashline</code>	3
<code>\ADLsomewide</code>	4	<code>\lasthline</code>	6
<code>\ADLsomewidehline</code>	7	<code>\LCC</code>	6
<code>Array</code> (environment)	6	<code>Longtable</code> (environment)	7
<code>array</code> (environment)	3	<code>longtable</code> (environment)	7
<code>array</code> (package)	2, 6	<code>longtable</code> (package)	2, 7
<code>\arrayrulecolor</code>	6	<code>LTchunksize</code> (counter)	7
<code>\arrayrulewidth</code>	4	M	
C		<code>\multicolumn</code>	3
<code>\cdashline</code>	3, 4	N	
<code>\cellcolor</code>	6	<code>\NAC</code>	6
<code>\cline</code>	3	<code>\newcolumnntype</code>	6
<code>\color</code>	7	<code>\nodashgapcolor</code>	7
<code>colortab</code> (package)	2, 6	P	
<code>colortbl</code> (package)	2, 6	packages:	
<code>\columncolor</code>	6	<code>array</code>	2, 6
counters:		<code>colortab</code>	2, 6
<code>LTchunksize</code>	7	<code>colortbl</code>	2, 6
D		<code>longtable</code>	2, 7
<code>\dashgapcolor</code>	7	R	
<code>\dashlinedash</code>	3	<code>\rowcolor</code>	6
<code>\dashlinegap</code>	3	T	
<code>\documentstyle</code>	2	<code>Tabular</code> (environment)	6
<code>\doublerulesepcolor</code>	6	<code>tabular</code> (environment)	3
E		U	
<code>\EAC</code>	6	<code>\usepackage</code>	2
<code>\ENAC</code>	6		
environments:			
<code>Array</code>	6		
<code>array</code>	3		
<code>Longtable</code>	7		