TAP – a set of TEX macros
for typesetting tables using POSTSCRIPT

Version 0.70 of Friday, May 3rd, 1996

Bogusław Jackowski, Piotr Pianowski and Piotr Strzelczyk
BOP s.c., ul. Piastowska 70, 80-363 Gdańsk, Poland
internet: jacko@ipipan.gda.pl

The starting point of this package was a neat package by Michael J. Ferguson. With his kind permission we rebuilt the package from scratch. Some notational "bells and whistles" have been added, and—which is perhaps more important—the package has been enhanced by PostScript-oriented features (painting cells with an arbitrary colour, putting diagonal crosses in cells, etc.).

TAP macros are designed to be convenient and efficient in, say, 80–90% of typical cases; on the other hand, special cases always require special treatment.

1. Overview. The general form of the table is as follows:

`\begintable`
⟨*table preamble*⟩
⟨*table row(s)*⟩
`\endtable`

Table preamble will be described momentarily.

Table row can be either a *horizontal rule*, or a *normal row*, or an *anchor row* (see the section on using POSTSCRIPT).

Horizontal rules can be obtained using two macros, `\-` and `\=`. A normal (0.4pt) rule is denoted by `\-`, a thicker one (1.2pt) is denoted by `\=`. Both can appear either as a full table rule or as a cell rule (see examples below).

Normal table rows have the following form:

`\B`⟨*suffix(es)*⟩ ⟨*cell contents*⟩ ⟨*cell separator*⟩ ⟨*cell contents*⟩ ⟨*cell separator*⟩ ... `\E`⟨*suffix*⟩

Cell contents seems obvious, *cell separator* can be one of the three characters: *vertical bar* `|`, *exclamation mark* `!`, and *double quotes* `"`. The first one denotes a normal vertical rule (0.4pt), the second one—a thicker rule (1.2pt), the last one—no rule. The spaces preceding and following *cell separators* are ignored.

The simplest suffixes for the `\B` and `\E` commands (*begin* and *end* of a row) are just cell separators, with exactly the same meaning. The command `\B`, however, may have an optional second suffix. Namely, one of the five characters can be used: `0` (zero), `-`, `+`, `_`, `^`, or `:`. They convey information about an extra space to be added above and/or below the current line:

- suffix `0` says that no extra space is needed, that is, that the natural height of the line is required;
- suffix `-` says that the cells of a given row are to be "squashed" as if they had zero height;
- suffix `+` says that a normal strut should precede the current row (the strut is defined as a `\vrule` of zero width, height h plus 0.25ex, and depth d plus 0.25ex, where h and d are the height and depth of `\hbox{(pl}` under the assumption that the current font on entering the table is used);
- suffix `_` says that a special strut (the normal strut lengthened downwards by 0.5ex) should precede the current row—to be used before (above) a horizontal rule;
- suffix `^` says that another special strut (the normal strut lengthened upwards by 0.5ex) should precede the current row—to be used after (below) a horizontal rule;
- suffix `:` says that yet another special strut (the normal strut lengthened both upwards and downwards by 0.5ex) should precede the current row (it is to be used between horizontal rules).

At the first glance, the classification of suffixes may look somewhat arbitrary. After thinking a while, however, you should admit that all the listed cases occur in almost each table, therefore it makes sense to provide standard means for handling them. This notation is supposed to be memorable as the character `-` can be remembered as "flat," character `^` as "up," character `_` as "down," and character `:` as "both up and down."

If you don't like this approach, you may ignore the optional suffixes and add the lines containing only the necessary rules above and below a given line.

If you like it, however, there is one more macro exploiting the same notation, namely, `\D` ("distender"). This macro can be followed by the same list of special characters: `0`, `-`, `+`, `_`, `^`, or `:`. As a result, a rule of the null width and the desired height and depth is typeset. Since `\D-` causes no squashing of any objects, `\D-` and `\D0` yield, in fact, the same results.

Table preamble has the following form:

```
\begintableformat
⟨cell descriptor⟩ ⟨cell separator⟩ ⟨cell descriptor⟩ ⟨cell separator⟩ ...
\endtableformat
```

In this case, *cell separator* is a *double quote* character `"`, *cell descriptor* can be `\left`, `\center` or `\right` command. The last *cell descriptor* can be preceded by an *ampersand* `&`, if the last cell descriptor is to be used for all subsequent cells.

Finally, there is an operator for joining a couple of cells together into one cell. It is represented by an *at* character `@`. For example, writing `@6` means that the contents should be spread over six cells and the template of the last cell is to be used for the placement.

Examples. The following data

```
\begintable
\begintableformat \right " &\center \endtableformat
\=
\B!^     | Polish     |         |         \E!
\B!- No. |            | Polish  | English \E! % flat row
\B!_     | diminutive |         |         \E!
\-
\B!^  1. | Ala        | Alicja  | Alice   \E!
\B!+  2. | Ania       | Anna    | Ann     \E!
\B":                @4 $\cdots$           \E"
\B!_ 10. | Zuzia      | Zuzanna | Susan   \E!
\=
\endtable
```

yield the following table:

No.	Polish diminutive	Polish	English
1.	Ala	Alicja	Alice
2.	Ania	Anna	Ann
		$\cdots$	
10.	Zuzia	Zuzanna	Susan

Assume now that the cell containing the text "No." is to be replaced by an empty space, that is, we wish to have a table with an incision at the upper left corner:

	Polish diminutive	Polish	English
1.	Ala	Alicja	Alice
2.	Ania	Anna	Ann
		$\cdots$	
10.	Zuzia	Zuzanna	Susan

Thinking in terms of "horizontal slices" is crucial here: such an approach is natural for the ASCII notation, and the ASCII notation is, in turn, intrinsic for typesetting using TeX. You will admit that the solution is still simple and legible:

```
\begintable
\begintableformat \right " &\center \endtableformat
\B"      !               @3 \=            \E! % incised row (slice)
```

```
\B"^     ! Polish     |         |         \E! % incised row (slice)
\B"-     !            | Polish  | English \E! % incised flat row (slice)
\B"_     ! diminutive |         |         \E! % incised row (slice)
\B! \=   ! \-         | \-      | \-      \E!
\B!^  1. | Ala        | Alicja  | Alice   \E!
\B!+  2. | Ania       | Anna    | Ann     \E!
\B":               @4 $\cdots$            \E"
\B!_ 10. | Zuzia      | Zuzanna | Susan   \E!
\=
\endtable
```

2. Useful macros. A couple of macros is provided for handling a bit more complex cases.

We found it useful to have the following abbreviations: `\>` standing for `\rlap`, `\<` standing for `\llap`, `\P` standing for `\phantom`, `\V` standing for `\vphantom`, `\K` standing for vertical spacing (kerning between rows).

A tilde `~` returns the space of the width of the digit zero of a current font which is useful for aligning the columns of numbers.

A one-parameter macro `\F` sets the current font for the rest of the table, for example, `\F\it` switches the font to italic for the subsequent rows (note that the font is not set globally). It should be used between rows (after `\E` and before `\B`).

Macro `\X` takes a series of lines (separated by double backslashes `\\`, surrounded obligatory by curly braces) as its parameter and returns a box with lines. The lines are aligned depending on a template for a current column. In order to override the inherited alignment template inside `\X` you can use one-parametr macros `\C`, `\L`, or `\R` for centring, aligning to left, or aligning to right, respectively. There is also macro `\x` working similarly, but the resulting box is flat (its height and depth are equal to zero) and centred with respect to a baseline. Usually, `\X` and `\x` expand to `\halign`. If you don't want to break all lines manually `\halign` is inconvenient. In such a case you should supply a desired width of the resulting box between the command name and the opening curly brace. For example, `\X 30mm {` ⟨*contents*⟩ `}` will cause that the contents will be set as a paragraph of width 30mm (inheriting centring, left alignment or right alignment from the template for a given column).

A pair of macros `\Y` and `\y` is very similar to the pair `\X` and `\x` except that `\vtop` is used for the construction of the resulting box instead of `\vbox`. This means that `\x` and `\y` are, in fact, equivalent.

Finally, you can use macros `\center`, `\left` and `\right` also in the table text, not only in the preamble. Their purpose is to override locally the template set in the preamble of a table. The first command they expand to is `\omit`, therefore no TEX command may appear between cell separators and these macros.

3. Configurability. The basic parameters of TAP macros can be adjusted to meet particular needs.

The width of the rules is controlled by two dimen registers: `\trth` (default 0.4pt, for normal rules) and `\TRTH` (default 1.2pt, for thicker rules). The macro `\defaultstrut` determines the height of rules inserted by the macro `\B` (see section "Overview") and can also be redefined if needed.

Another quantity that you may wish to adjust is a cell margin. It is governed by a dimen register `\cellmarg`, set at the beginning of a table to a default value 0.5em (in the units of a current font).

In more complex cases it may turn out that the actual parametrization does not provide sufficient flexibility and introducing new macros or modifying existing ones may turn out unavoidable. Fortunately, TEX programs are available mostly as sources. If programs are exceedingly large (LATEX may serve as a notable example), modifying them is nearly impossible. We hope, however, that here the task is not so difficult as it may appear at the first glance. If you are bold enough, you will certainly be able to make the alterations you need.

Table settings can be altered by assigning an appropriate value to either `\everytable` or `\thistable`, both being token registers; for example, `\everytable{\def\<{$<$}\def\>{$>$}` will override the previously mentioned meaning of `\<` and `\>` in all subsequent tables, `\thistable{\def\<{$<$}\def\>{$>$}`—only in the first encountered table.

4. Active characters. The "magic" characters `|`, `!`, `"`, and `@` are active inside a table (they are defined as characters of category 13). You should be aware that using active characters is a mixed blessing: on one hand, notation becomes more legible, on the other hand, using active characters foretells troubles, because they cannot be used in parameters to most TEX macros. For example, the construction `\centerline {\begintable ... \endtable}` will not work since `\centerline` swallows its parameter before `\begintable` changes the category codes. Incidentally, the construction `\line {\hss\begintable ... \endtable\hss}` does work. Can you see why?

In order to facilitate dealing with cases where a table—for some reasons—is to be passed in a parameter, the macro `\deftable` was introduced. It allows for storing the contents of a table as a macro of a given name. For example, after saying `\deftable\foo \begintable` ⟨*contents*⟩ `\endtable` you may safely say `\centerline {\begintable \foo \endtable}` (`\foo` is defined globally).

5. Forcing the width of a table. If a dimen register `\desiredwidth` is assigned a positive value (default is 0mm) a special technique is used in order to obtain a table of the desired width by "padding" cells appropriately with empty space. A self-suggesting approach, namely, using a `\tabskip` glue, seems inadequate here because of the techniques used for employing PostScript (see below).

6. Using PostScript. Frequently, some cells or rows, or even the whole table, is to be printed against a non-white background. Sometimes a diagonal line is to be drawn across a cell. PostScript is an excellent tool that can be used for this purpose. The technique implemented in the TAP package bases on `\special` commands we call "anchors." Obviously, using `\special` commands is always driver-oriented. The current version of the TAP package is meant for further processing using T. Rokicki's DVIPS, although switching to an arbitrary PostScript driver should be a rather simple task.

TEX should be informed that anchors are employed. To do this you should use a pair of macros `\beginanchtable ... \endanchtable` instead of `\begintable ... \endtable`.

A pair of macros, `\uranchor` (cast an anchor at the *upper right* corner of a cell) and `\llanchor` (cast an anchor at the *lower left corner* of a cell) can be put between rows for determining a rectangle to be filled or crossed. The filling can be accomplished using macros `\rectfill` and `\trianfill`, diagonal lines can be drawn using macro `\diagstroke`. Macros `\uranchor` and `\llanchor` are to be put below and above a rule, respectively. Macros `\rectfill`, `\trianfill` and `\diagstroke` can be put between any two rows of a table, their order, however, is significant.

Here you have the exact syntax of the anchor commands:

`\uranchor` ⟨*number of a column*⟩ ⟨*anchor identifier*⟩
`\llanchor` ⟨*number of a column*⟩ ⟨*anchor identifier*⟩
`\rectfill {`⟨*cmyk coordinates of a colour*⟩`}` ⟨*anchor identifier*⟩
`\trianfill` ⟨*triangle descriptor*⟩ `{`⟨*cmyk coordinates of a colour*⟩`}` ⟨*anchor identifier*⟩
`\diagstroke` ⟨*pen width*⟩ ⟨*diagonal descriptor*⟩ `{`⟨*cmyk coordinates of a colour*⟩`}` ⟨*anchor identifier*⟩

Remarks.

(a) *The number of a column* is simply a TEX count (integer number). If a whole row is meant rather than a particular cell you may use an artificial column index ≤ 0.
(b) *Anchor identifier* is an arbitrary sequence of characters (typically a single letter).
(c) *Cmyk coordinates* are four numbers from the range [0. .1] separated by spaces. Curly braces are obligatory, even if you pass a macro expanding to cmyk coordinates as a parameter.
(d) *Triangle descriptor* can be one of the following two-letter sequences: `SW`, `SE`, `NW`, `NE` (standing for south-west, south-east, north-west, and north-east, respectively). It determines the position of the right angle of a triangle. The case of letters is meaningless, curly braces should not be used.
(e) *Pen width* is just a TEX dimen that is swallowed using assignment, therefore curly braces should not be used.
(f) *Diagonal descriptor* can be one of the following sequences: `BOTH`, `DOWN`, `UP`. The meaning, hopefully, is self-explaining. The case of letters is meaningless, curly braces should not be used.

Examples.

The following example shows how anchors can be used. The application is admittedly artificial, nevertheless, it demonstrates possibilities. The following TEX code:

```
\beginanchtable
\begintableformat \right " &\center \endtableformat
```

```
\rectfill {0 0 0 .1} a          % fill a rectangle marked by the anchors suffixed ‘a’
\trianfill SE {0 0 0 .1} b      % fill a triangle marked by the anchors suffixed ‘b’
\diagstroke .4pt UP {0 0 0 1} b % draw a stroke marked by the anchors suffixed ‘b’
\=
\uranchor0a \uranchor1b
\B!^     | Polish     |         |         \E!
\B!-     |            | Polish  | English \E!
\B!_     | diminutive |         |         \E!
\llanchor2a \llanchor1b
\-
\B!^  1. | Ala        | Alicja  | Alice   \E!
\B!+  2. | Ania       | Anna    | Ann     \E!
\B":               @4 $\cdots$          \E"
\B!_ 10. | Zuzia      | Zuzanna | Susan   \E!
\=
\endanchtable
```

produces a table that is not so easy to obtain without PostScript:

	Polish diminutive	Polish	English
1.	Ala	Alicja	Alice
2.	Ania	Anna	Ann
		...	
10.	Zuzia	Zuzanna	Susan

Finally, an excerpt from a real-life example demonstrating most of the described features.

group/ subgroup	name of the group	number of tested persons	citizens of Poland			foreigners	
			after leaving the country	not leaving the country	missing data	from Africa and Asia	from Cuba
I.	**disease X**	**378**	**262**	**62**	**0**	**51**	**3**
1.	variant A	195	131	38	0	23	3
2.	variant B	131	95	24	0	12	0
3.	variant C	52	36	0	0	16	0
II.	**control group**	**3990**	**2808**	**276**	**143**	**577**	**186**

As usually, real-life examples demonstrate not only the advantages of a method, but also uncover its drawbacks. In this particular example, the main difficulty was a convenient alignment to the right both bold and roman numbers and centring the result in a column. If the only typeface used were either roman or bold, macro tilde ~ might have been used (see section “Useful macros”). Aligning digits of varying width can be considered a weird demand and the TAP package is not supposed to be prepared for such cases.

We decided to define macros putting a number into a box of a desired width. It sufficed to introduce two such macros, \3 and \4, for formatting three and four digit numbers, respectively (note that they are not defined in the TAP package):

```
\newcount\anycount
\def\3{\afterassignment\threecont\anycount}
\def\threecont{\setbox0\hbox{\rm000}\hbox to\wd0{\hss\the\anycount}}
\def\4{\afterassignment\fourcont\anycount}
\def\fourcont{\setbox0\hbox{\rm0000}\hbox to\wd0{\hss\the\anycount}}
```

The TEX code for the table reads:

```
\thistable{\desiredwidth\hsize}% set the width of this table to \hsize
\beginanchtable
\begintableformat
  \right " \left " &\center
\endtableformat
%
\rectfill{0 0 0 .1}a
\rectfill{0 0 0 .1}b
%
\=
\uranchor0a
\B!^       !           !        !   @3                       !  @2              \E!
\B!-       !           !        !   @3 citizens of Poland    !  @2 foreigners   \E!
\B!_       !           !        !   @3                       !  @2              \E!
\B!- \center{\X{group/\\subgroup}}
     !  \center{\X{name\\of the group}}
     !  \X{number\\of tested\\persons}
                                !                   @5  \-                      \E!
\B!^       !           !        !         |        |        !        |          \E!
\B!-       !           !        !\X{after leaving\\the country}
                                |\X{not leaving\\the country}
                                |\X{missing\\data}
                                !\X{from Africa\\and Asia}
                                |\X{from\\Cuba} \E!
\B!_       !           !        !         |        |        !        |          \E!
\llanchor0a
\=
\B!  @8 \vrule width 0mm height 0.75mm \E!
\=
\uranchor1b
\F\bf
\B!:  I.    ! disease X ! \4  378 ! \4  262 | \3  62 | \3   0 ! \3  51 | \3   3 \E!
\F\rm
\-
\B!^  1.    ! variant A ! \4  195 ! \4  131 | \3  38 | \3   0 ! \3  23 | \3   3 \E!
\B!+  2.    ! variant B ! \4  131 ! \4   95 | \3  24 | \3   0 ! \3  12 | \3   0 \E!
\B!_  3.    ! variant C ! \4   52 ! \4   36 | \3   0 | \3   0 ! \3  16 | \3   0 \E!
\=
\F\bf
\B!^        ! control   !         !         |        |        !        |        \E!
\K-.5pt % tiny correction of vertical spacing
\B!- II.    !           ! \4 3990 ! \4 2808 | \3 276 | \3 143 ! \3 577 | \3 186 \E!
\K-.5pt % tiny correction of vertical spacing
\B!_        ! group     !         !         |        |        !        |        \E!
\llanchor1b
\=
\endanchtable
```

7. Final remarks. As always, the ultimate documentation are the source files, in this case `TAP.TeX` and `TAPdoc.TeX`. A careful inspection of the files may provide some hints, reveal possibilities not mentioned in this documentation, explain some points that were explained vaguely. The code is not as simple as one would expect, however, it should be remembered that there is a trade-off between notation and implementation. Here we sacrificed the simplicity of implementation in favour of the ease of use and the legibility of user's TEX code.