Tabularray: Typeset Tabulars and Arrays with LaTeX3

Jianrui Lyu (tolvjr@163.com)

Version 2021K (2021-06-05)
https://github.com/lvjr/tabularray

Contents

Chapter 1

From Old to New

1.1 Vertical Space

After loading `tabularrray` package in the preamble, we can use `tblr` environments to typeset tabulars and arrays. The name `tblr` is short for `tabularray` or `top-bottom-left-right`. The following is our first example:

```
\begin{tabular}{lccr}
\hline
 Alpha   & Beta  & Gamma  & Delta \\
\hline
 Epsilon & Zeta  & Eta    & Theta \\
\hline
 Iota    & Kappa & Lambda & Mu    \\
\hline
\end{tabular}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

```
\begin{tblr}{lccr}
\hline
 Alpha   & Beta  & Gamma  & Delta \\
\hline
 Epsilon & Zeta  & Eta    & Theta \\
\hline
 Iota    & Kappa & Lambda & Mu    \\
\hline
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

You may notice that there is extra space above and below the table rows with `tblr` envirenment. This space makes the table look better. If you don't like it, you could use `\SetTblrDefault` command:

```
\SetTblrDefault{rowsep=0pt}
\begin{tblr}{lccr}
\hline
 Alpha   & Beta  & Gamma  & Delta \\
\hline
 Epsilon & Zeta  & Eta    & Theta \\
\hline
 Iota    & Kappa & Lambda & Mu    \\
\hline
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

But in many cases, this rowsep is useful:

```
$\begin{array}{rrr}
\hline
 \dfrac{2}{3} &  \dfrac{2}{3} &  \dfrac{1}{3} \\
 \dfrac{2}{3} & -\dfrac{1}{3} & -\dfrac{2}{3} \\
 \dfrac{1}{3} & -\dfrac{2}{3} &  \dfrac{2}{3} \\
\hline
\end{array}$
```

$$\begin{array}{rrr}
\hline
 \dfrac{2}{3} & \dfrac{2}{3} & \dfrac{1}{3} \\
 \dfrac{2}{3} & -\dfrac{1}{3} & -\dfrac{2}{3} \\
 \dfrac{1}{3} & -\dfrac{2}{3} & \dfrac{2}{3} \\
\hline
\end{array}$$

```
$\begin{tblr}{rrr}
\hline
 \dfrac{2}{3} &  \dfrac{2}{3} &  \dfrac{1}{3} \\
 \dfrac{2}{3} & -\dfrac{1}{3} & -\dfrac{2}{3} \\
 \dfrac{1}{3} & -\dfrac{2}{3} &  \dfrac{2}{3} \\
\hline
\end{tblr}$
```

$$\begin{tblr}{rrr}
\hline
 \dfrac{2}{3} & \dfrac{2}{3} & \dfrac{1}{3} \\
 \dfrac{2}{3} & -\dfrac{1}{3} & -\dfrac{2}{3} \\
 \dfrac{1}{3} & -\dfrac{2}{3} & \dfrac{2}{3} \\
\hline
\end{tblr}$$

Note that you can use `tblr` in both text and math modes.

1.2 Multiline Cells

It's quite easy to write multiline cells without fixing the column width in `tblr` environments: just enclose the cell text with braces and use `\\` to break lines:

```
\begin{tblr}{|l|c|r|}
\hline
 Left & {Center \\ Cent \\ C} & {Right \\ R} \\
\hline
 {L \\ Left} & {C \\ Cent \\ Center} & R \\
\hline
\end{tblr}
```

Left	Center Cent C	Right R
L Left	C Cent Center	R

1.3 Cell Alignment

From time to time, you may want to specify the horizontal and vertical alignment of cells at the same time. `Tabularray` package provides a `Q` column for this (In fact, `Q` column is the only primitive column, other columns are defined as `Q` columns with some options):

```
\begin{tblr}{|Q[l,t]|Q[c,m]|Q[r,b]|}
\hline
 {Top Baseline \\ Left Left} & Middle Center & {Right Right \\ Bottom Baseline} \\
\hline
\end{tblr}
```

Top Baseline Left Left	Middle Center	Right Right Bottom Baseline

Note that you can use more meaningful `t` instead of `p` for top baseline alignment. For some users who are familiar with word processors, these `t` and `b` columns are counter-intuitive. In `tabularray` package, there are another two column types `h` and `f`, which will align cell text at row head and row foot, respectively:

```
\begin{tblr}{Q[h,4em]Q[t,4em]Q[m,4em]Q[b,4em]Q[f,4em]}
\hline
 {row\\head} & {top\\line} & {middle} & {line\\bottom} & {row\\foot} \\
\hline
 {row\\head} & {top\\line} & {11\\22\\mid\\44\\55} & {line\\bottom} & {row\\foot} \\
\hline
\end{tblr}
```

row head	top line	middle	line bottom	row foot
row head	top line	11 22 mid 44 55	line bottom	row foot

1.4 Multirow Cells

The above `h` and `f` columns are necessary for multirow cells. In `tabularray`, the `t` and `b` in the optional argument of `\multirow` commands will be treated as `h` and `f`, respectively:

```
\begin{tabular}{|l|l|l|l|}
\hline
 \multirow[t]{4}{1.5cm}{Multirow Cell One} & Alpha &
 \multirow[b]{4}{1.5cm}{Multirow Cell Two} & Alpha \\
 & Beta  & & Beta \\
 & Gamma & & Gamma \\
 & Delta & & Delta \\
\hline
\end{tabular}
```

Multirow Cell One	Alpha Beta Gamma Delta	Multirow Cell Two	Alpha Beta Gamma Delta

```
\begin{tblr}{|l|l|l|l|}
\hline
 \multirow[t]{4}{1.5cm}{Multirow Cell One} & Alpha &
 \multirow[b]{4}{1.5cm}{Multirow Cell Two} & Alpha \\
 & Beta  & & Beta \\
 & Gamma & & Gamma \\
 & Delta & & Delta \\
\hline
\end{tblr}
```

Multirow Cell One	Alpha Beta Gamma Delta	Multirow Cell Two	Alpha Beta Gamma Delta

Note that you don't need to load `multirow` package first, since `tabularrray` doesn't depend on it. Furthermore, `tabularray` will always typeset descent multirow cells. First, it will set correct vertical `c` alignment, even though some rows have large height:

```
\begin{tabular}{|l|m{4em}|}
\hline
 \multirow[c]{4}{1.5cm}{Multirow} & Alpha  \\
 & Beta  \\
 & Gamma \\
 & Delta Delta Delta \\
\hline
\end{tabular}
```

Multirow	Alpha Beta Gamma Delta Delta Delta

```
\begin{tblr}{|l|m{4em}|}
\hline
 \multirow[c]{4}{1.5cm}{Multirow} & Alpha  \\
 & Beta  \\
 & Gamma \\
 & Delta Delta Delta \\
\hline
\end{tblr}
```

Multirow	Alpha Beta Gamma Delta Delta Delta

Second, it will enlarge row heights if the multirow cells have large height, therefore it always avoids vertical overflow:

```
\begin{tabular}{|l|m{4em}|}
\hline
 \multirow[c]{2}{1cm}{Line \\ Line \\ Line \\ Line} & Alpha \\
\cline{2-2}
 & Beta \\
\hline
\end{tabular}
```

Line Line	Alpha Beta
Line Line	

```
\begin{tblr}{|l|m{4em}|}
\hline
 \multirow[c]{2}{1cm}{Line \\ Line \\ Line \\ Line} & Alpha \\
\cline{2}
 & Beta \\
\hline
\end{tblr}
```

Line Line Line Line	Alpha Beta

1.5 Multi Rows and Columns

It was a hard job to typeset cells with multiple rows and multiple columns. For example:

```
\begin{tabular}{|c|c|c|c|c|}
\hline
 \multirow{2}{*}{2 Rows}
     & \multicolumn{2}{c|}{2 Columns}
                 & \multicolumn{2}{c|}{\multirow{2}{*}{2 Rows 2 Columns}} \\
\cline{2-3}
     & 2-2 & 2-3 & \multicolumn{2}{c|}{} \\
\hline
 3-1 & 3-2 & 3-3 & 3-4 & 3-5 \\
\hline
\end{tabular}
```

2 Rows	2 Columns		2 Rows 2 Columns	
	2-2	2-3		
3-1	3-2	3-3	3-4	3-5

With `tabularray` package, you can set spanned cells with `\SetCell` command: within the optional argument of `\SetCell` command, option `r` is for rowspan number, and `c` for colspan number; within the

mandatory argument of it, horizontal and vertical alignment options are accepted. Therefore it's much simpler to typeset spanned cells:

```
\begin{tblr}{|c|c|c|c|c|}
\hline
 \SetCell[r=2]{c} 2 Rows
     & \SetCell[c=2]{c} 2 Columns
           &     & \SetCell[r=2,c=2]{c} 2 Rows 2 Columns & \\
\hline
     & 2-2 & 2-3 &     &     \\
\hline
 3-1 & 3-2 & 3-3 & 3-4 & 3-5 \\
\hline
\end{tblr}
```

2 Rows	2 Columns		2 Rows 2 Columns	
	2-2	2-3		
3-1	3-2	3-3	3-4	3-5

Using `\multicolumn` command, the omitted cells must be removed. On the contrary, using `\multirow` command, the omitted cells must not be removed. `\SetCell` command behaves the same as `\multirow` command in this aspect.

With `tblr` environment, any `\hline` segments inside a spanned cell will be ignored, therefore we're free to use `\hline` in the above example. Also, any omitted cell will definitely be ignored when typesetting, no matter it's empty or not. With this feature, we could put row and column numbers into the omitted cells, which will help us to locate cells when the tables are rather complex:

```
\begin{tblr}{|ll|c|rr|}
\hline
 \SetCell[r=3,c=2]{h} r=3 c=2 & 1-2 & \SetCell[r=2,c=3]{r} r=2 c=3 & 1-4 & 1-5 \\
 2-1 & 2-2 & 2-3 & 2-4 & 2-5 \\
\hline
 3-1 & 3-2 & MIDDLE & \SetCell[r=3,c=2]{f} r=3 c=2 & 3-5 \\
\hline
 \SetCell[r=2,c=3]{l} r=2 c=3 & 4-2 & 4-3 & 4-4 & 4-5 \\
 5-1 & 5-2 & 5-3 & 5-4 & 5-5 \\
\hline
\end{tblr}
```

r=3 c=2			r=2 c=3
	MIDDLE		
r=2 c=3			r=3 c=2

1.6 Column Types

`Tabularray` package supports all normal column types, as well as the extendable `X` column type, which first occured in `tabularx` package and was largely improved by `tabu` package:

```
\begin{tblr}{|X[2,l]|X[3,l]|X[1,r]|X[r]|}
\hline
 Alpha & Beta & Gamma & Delta \\
\hline
\end{tblr}
```

Alpha	Beta	Gamma	Delta

Also, `X` columns with negative coefficients are possible:

```
\begin{tblr}{|X[2,l]|X[3,l]|X[-1,r]|X[r]|}
\hline
 Alpha & Beta & Gamma & Delta \\
\hline
\end{tblr}
```

Alpha	Beta	Gamma	Delta

We need the width to typeset a table with `X` columns. If unset, the default is `\linewidth`. To change the width, we have to first put all column specifications into `colspec={...}`:

```
\begin{tblr}{width=0.8\linewidth,colspec={|X[2,l]|X[3,l]|X[-1,r]|X[r]|}}
\hline
 Alpha & Beta & Gamma & Delta \\
\hline
\end{tblr}
```

Alpha	Beta	Gamma	Delta

You can define new column types with `\NewColumnType` command. For example, in `tabularray` package, `b` and `X` columns are defined as special `Q` columns:

```
\NewColumnType{b}[1]{Q[b,wd=#1]}
\NewColumnType{X}[1][]{Q[co=1,#1]}
```

1.7 Row Types

Now that we have column types and `colspec` option, you may ask for row types and `rowspec` option. Yes, they are here:

```
\begin{tblr}{colspec={Q[l]Q[c]Q[r]},rowspec={|Q[t]|Q[m]|Q[b]|}}
 {Alpha \\ Alpha} & Beta              & Gamma \\
 Delta            & Epsilon           & {Zeta \\ Zeta}  \\
 Eta              & {Theta \\ Theta}  & Iota  \\
\end{tblr}
```

Alpha Alpha	Beta	Gamma
Delta	Epsilon	Zeta Zeta
Eta	Theta Theta	Iota

Same as column types, `Q` is the only primitive row type, and other row types are defined as `Q` types with different options. It's better to specify horizontal alignment in `colspec`, and vertical alignment in `rowspec`, respectively.

Inside `rowspec`, `|` is the hline type. Therefore we need not to write `\hline` commnad, which makes table code cleaner.

1.8 Hlines and Vlines

Hlines and vlines have been improved too. You can specify the widths and styles of them:

```
\begin{tblr}{|l|[dotted]|[2pt]c|r|[solid]|[dashed]|}
\hline
One   &  Two  & Three \\
\hline\hline[dotted]\hline
Four  & Five  &   Six \\
\hline[dashed]\hline[1pt]
Seven & Eight &  Nine \\
\hline
\end{tblr}
```

One	Two	Three
Four	Five	Six
Seven	Eight	Nine

1.9 Colorful Tables

To add colors to your tables, you need to load `xcolor` package first. `Tabularray` package will also load `ninecolors` package for proper color contrast. First you can specify background option for `Q` rows/columns inside `rowspec`/`colspec`:

```
\begin{tblr}{colspec={lcr},rowspec={|Q[cyan7]|Q[azure7]|Q[blue7]|}}
 Alpha   & Beta  & Gamma  \\
 Epsilon & Zeta  & Eta    \\
 Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

```
\begin{tblr}{colspec={Q[l,brown7]Q[c,yellow7]Q[r,olive7]},rowspec={|Q|Q|Q|}}
 Alpha   & Beta  & Gamma  \\
 Epsilon & Zeta  & Eta    \\
 Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

Also you can use `\SetRow` or `\SetColumn` command to specify row or column colors:

```
\begin{tblr}{colspec={lcr},rowspec={|Q|Q|Q|}}
 \SetRow{cyan7}  Alpha   & Beta  & Gamma  \\
 \SetRow{azure7} Epsilon & Zeta  & Eta    \\
 \SetRow{blue7}  Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

```
\begin{tblr}{colspec={lcr},rowspec={|Q|Q|Q|}}
 \SetColumn{brown7}
 Alpha           & \SetColumn{yellow7}
                   Beta            & \SetColumn{olive7}
                                     Gamma  \\
 Epsilon         & Zeta            & Eta    \\
 Iota            & Kappa           & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

Hlines and vlines can also have colors:

```
\begin{tblr}{colspec={lcr},rowspec={|[2pt,green7]Q|[teal7]Q|[green7]Q|[3pt,teal7]}}
 Alpha   & Beta  & Gamma  \\
 Epsilon & Zeta  & Eta    \\
 Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

```
\begin{tblr}{colspec={|[2pt,violet5]l|[2pt,magenta5]c|[2pt,purple5]r|[2pt,red5]}}
 Alpha   & Beta  & Gamma  \\
 Epsilon & Zeta  & Eta    \\
 Iota    & Kappa & Lambda \\
\end{tblr}
```

Alpha	Beta	Gamma
Epsilon	Zeta	Eta
Iota	Kappa	Lambda

1.10 New Table Commands

All commands which change the specifications of tables must be defined with `\NewTableCommand`. The following example demonstrates how to define similar rules as in `booktabs` package:

```
\NewTableCommand\toprule{\hline[0.08em]}
\NewTableCommand\midrule{\hline[0.05em]}
\NewTableCommand\bottomrule{\hline[0.08em]}
\begin{tblr}{llll}
\toprule
 Alpha   & Beta  & Gamma   & Delta \\
\midrule
 Epsilon & Zeta  & Eta     & Theta \\
 Iota    & Kappa & Lambda  & Mu    \\
 Nu      & Xi    & Omicron & Pi    \\
\bottomrule
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi

Chapter 2

New Interfaces

With `tabularray` package, you can separate style and content totally in tables.

2.1 Hlines and Vlines

Options `hlines` and `vlines` are for setting all hlines and vlines, respectively. With empty value, all hlines/vlines will be solid.

```
\begin{tblr}{hlines,vlines}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
 Phi     & Chi   & Psi     & Omega   \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi
Rho	Sigma	Tau	Upsilon
Phi	Chi	Psi	Omega

With values inside one pair of braces, all hlines/vlines will be styled.

```
\begin{tblr}{
 hlines = {1pt,solid},
 vlines = {red3,dashed},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
 Phi     & Chi   & Psi     & Omega   \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi
Rho	Sigma	Tau	Upsilon
Phi	Chi	Psi	Omega

Another pair of braces before will select segments in all hlines/vlines.

```
\begin{tblr}{
 vlines = {1,3,5}{dashed},
 vlines = {2,4,6}{solid},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
 Phi     & Chi   & Psi     & Omega   \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi
Rho	Sigma	Tau	Upsilon
Phi	Chi	Psi	Omega

The above example can be simplified with `odd` and `even` values. (More child selectors can be defined with `\NewChildSelector` command. Advanced users could read the source code for this.)

```
\begin{tblr}{
 vlines = {odd}{dashed},
 vlines = {even}{solid},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
 Phi     & Chi   & Psi     & Omega   \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi
Rho	Sigma	Tau	Upsilon
Phi	Chi	Psi	Omega

Another pair of braces before will draw more hlines/vlines (in which `-` stands for all line segments).

```
\begin{tblr}{
 hlines = {1}{-}{dashed},
 hlines = {2}{-}{solid},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
 Phi     & Chi   & Psi     & Omega   \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi
Rho	Sigma	Tau	Upsilon
Phi	Chi	Psi	Omega

Options `hline{i}` and `vline{j}` are for setting some hlines and vlines, respectively.

```
\begin{tblr}{
 hline{1,7} = {1pt,solid},
 hline{3-5} = {blue3,dashed},
 vline{1,5} = {3-4}{dotted},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
 Phi     & Chi   & Psi     & Omega   \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi
Rho	Sigma	Tau	Upsilon
Phi	Chi	Psi	Omega

2.2 Cells and Spancells

Option `cells` is for setting all cells.

```
\begin{tblr}{hlines={white},cells={c,blue7}}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
 Phi     & Chi   & Psi     & Omega   \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi
Rho	Sigma	Tau	Upsilon
Phi	Chi	Psi	Omega

Option `cell{i}{j}` is for setting some cells.

```
\begin{tblr}{
 hlines = {white},
 vlines = {white},
 cell{1,6}{odd} = {teal7},
 cell{1,6}{even} = {green7},
 cell{2,4}{1,4} = {red7},
 cell{3,5}{1,4} = {purple7},
 cell{2}{2} = {r=4,c=2}{c,azure7},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
 Phi     & Chi   & Psi     & Omega   \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta		Theta
Iota			Mu
Nu			Pi
Rho			Upsilon
Phi	Chi	Psi	Omega

2.3 Rows and Columns

Options `rows` and `columns` are for setting all rows and columns, respectively.

```
\begin{tblr}{
 hlines,
 vlines,
 rows = {7mm},
 columns = {15mm,c},
}
 Alpha   & Beta  & Gamma   & Delta \\
 Epsilon & Zeta  & Eta     & Theta \\
 Iota    & Kappa & Lambda  & Mu    \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

Options `row{i}` and `column{j}` are for setting some rows and columns, respectively.

```
\begin{tblr}{
 hlines = {1pt,white},
 row{odd} = {blue7},
 row{even} = {azure7},
 column{1} = {purple7,c},
}
 Alpha   & Beta  & Gamma   & Delta   \\
 Epsilon & Zeta  & Eta     & Theta   \\
 Iota    & Kappa & Lambda  & Mu      \\
 Nu      & Xi    & Omicron & Pi      \\
 Rho     & Sigma & Tau     & Upsilon \\
 Phi     & Chi   & Psi     & Omega   \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu
Nu	Xi	Omicron	Pi
Rho	Sigma	Tau	Upsilon
Phi	Chi	Psi	Omega

We can specify foreground colors, background colors and fonts with `bg`, `fg` and `font` keys, respectively, for cells/rows/columns. In most cases, `bg` key can be omitted, which you can see in the previous examples.

```
\begin{tblr}{
 colspec  = {lcr},
 row{odd} = {bg=azure8},
 row{1}   = {bg=azure3, fg=white, font=\sffamily},
}
 Alpha & Beta    & Gamma \\
 Delta & Epsilon & Zeta  \\
 Eta   & Theta   & Iota  \\
 Kappa & Lambda  & Mu    \\
 Nu Xi Omikron & Pi Rho Sigma & Tau Upsilon Phi \\
\end{tblr}
```

Alpha	Beta	Gamma
Delta	Epsilon	Zeta
Eta	Theta	Iota
Kappa	Lambda	Mu
Nu Xi Omikron	Pi Rho Sigma	Tau Upsilon Phi

2.4 Space in Tables

Options `rowsep` and `colsep` are for setting padding for rows and columns, respectively.

```
\SetTblrDefault{rowsep=2pt,colsep=2pt}
\begin{tblr}{hlines,vlines}
 Alpha   & Beta  & Gamma  & Delta \\
 Epsilon & Zeta  & Eta    & Theta \\
 Iota    & Kappa & Lambda & Mu    \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

Also `abovesep`, `belowsep`, `leftsep`, `rightsep` options are available:

```
\begin{tblr}{
 hlines,
 vlines,
 rows = {abovesep=1pt,belowsep=5pt},
 columns = {leftsep=1pt,rightsep=5pt},
}
 Alpha   & Beta  & Gamma  & Delta \\
 Epsilon & Zeta  & Eta    & Theta \\
 Iota    & Kappa & Lambda & Mu    \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

And `\\[dimen]` can be replaced by `belowsep+` option:

```
\begin{tblr}{
 hlines, row{2} = {belowsep+=5pt},
}
 Alpha   & Beta  & Gamma  & Delta \\
 Epsilon & Zeta  & Eta    & Theta \\
 Iota    & Kappa & Lambda & Mu    \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

Also `\doublerulesep` parameter can be replaced by `rulesep` option:

```
\begin{tblr}{
 colspec={||llll||},rowspec={|QQQ|},rulesep=4pt,
}
 Alpha   & Beta  & Gamma  & Delta \\
 Epsilon & Zeta  & Eta    & Theta \\
 Iota    & Kappa & Lambda & Mu    \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

Also `\arraystretch` parameter can be replaced by `stretch` option:

```
\begin{tblr}{hlines,stretch=1.5}
 Alpha   & Beta  & Gamma  & Delta \\
 Epsilon & Zeta  & Eta    & Theta \\
 Iota    & Kappa & Lambda & Mu    \\
\end{tblr}
```

Alpha	Beta	Gamma	Delta
Epsilon	Zeta	Eta	Theta
Iota	Kappa	Lambda	Mu

2.5 Counters in Tables

Counters `rownum`, `colnum`, `rowcount`, `colcount` can be used in cell text:

```
\begin{tblr}{hlines}
 Cell[\arabic{rownum}][\arabic{colnum}] & Cell[\arabic{rownum}][\arabic{colnum}] &
 Cell[\arabic{rownum}][\arabic{colnum}] & Cell[\arabic{rownum}][\arabic{colnum}] \\
 Cell[\arabic{rownum}][\arabic{colnum}] & Cell[\arabic{rownum}][\arabic{colnum}] &
 Cell[\arabic{rownum}][\arabic{colnum}] & Cell[\arabic{rownum}][\arabic{colnum}] \\
 Row=\arabic{rowcount}, Col=\arabic{colcount} &
 Row=\arabic{rowcount}, Col=\arabic{colcount} &
 Row=\arabic{rowcount}, Col=\arabic{colcount} &
 Row=\arabic{rowcount}, Col=\arabic{colcount} \\
 Cell[\arabic{rownum}][\arabic{colnum}] & Cell[\arabic{rownum}][\arabic{colnum}] &
 Cell[\arabic{rownum}][\arabic{colnum}] & Cell[\arabic{rownum}][\arabic{colnum}] \\
 Cell[\arabic{rownum}][\arabic{colnum}] & Cell[\arabic{rownum}][\arabic{colnum}] &
 Cell[\arabic{rownum}][\arabic{colnum}] & Cell[\arabic{rownum}][\arabic{colnum}] \\
\end{tblr}
```

Cell[1][1]	Cell[1][2]	Cell[1][3]	Cell[1][4]
Cell[2][1]	Cell[2][2]	Cell[2][3]	Cell[2][4]
Row=5, Col=4	Row=5, Col=4	Row=5, Col=4	Row=5, Col=4
Cell[4][1]	Cell[4][2]	Cell[4][3]	Cell[4][4]
Cell[5][1]	Cell[5][2]	Cell[5][3]	Cell[5][4]

2.6 Experimental Interfaces

Everything described in this section is in **experimental** status. Don't use them in important documents, unless you have time to update them for the newer versions of `tabularray` package in the future.

By default `tabularray` package will compute column widths from span widths. By setting option `hspan=minimal`, it will compute span widths from column widths. The following two examples show this difference:

```
\SetTblrDefault{hlines,vlines}
\begin{tblr}{cell{2}{1}={c=2}{l},cell{3}{1}={c=3}{l},cell{4}{2}={c=2}{l}}
 111 111 & 222 222 & 333 333 \\
 12 Multi Columns Multi Columns 12 & & 333 \\
 13 Multi Columns Multi Columns Multi Columns 13 & & \\
 111 & 23 Multi Columns Multi Columns 23 & \\
\end{tblr}
```

111 111	222 222	333 333
12 Multi Columns Multi Columns 12		333
13 Multi Columns Multi Columns Multi Columns 13		
111	23 Multi Columns Multi Columns 23	

```
\SetTblrDefault{hlines,vlines,hspan=minimal}
\begin{tblr}{cell{2}{1}={c=2}{l},cell{3}{1}={c=3}{l},cell{4}{2}={c=2}{l}}
 111 111 & 222 222 & 333 333 \\
 12 Multi Columns Multi Columns 12 & & 333 \\
 13 Multi Columns Multi Columns Multi Columns 13 & & \\
 111 & 23 Multi Columns Multi Columns 23 & \\
\end{tblr}
```

111 111	222 222	333 333
12 Multi Columns Multi Columns 12		333
13 Multi Columns Multi Columns Multi Columns 13		
111	23 Multi Columns Multi Columns 23	

To trace internal data behind `tblr` environment, you can use `\SetTblrTracing` command. For example, `\SetTblrTracing{all}` will turn on all tracings, and `\SetTblrTracing{none}` will turn off all tracings. `\SetTblrTracing{+row,+column}` will only tracing row and column data. All tracing messages will be written to the log files.

Chapter 3

Source Code

```
%%% % -*- coding: utf-8 -*-
%%% ----------------------------------------------------------------------------
%%% Tabularray: Typeset tabulars and arrays with LaTeX3
%%% Author    : Jianrui Lyu <tolvjr@163.com>
%%% Repository: https://github.com/lvjr/tabularray
%%% License   : The LaTeX Project Public License 1.3
%%% ----------------------------------------------------------------------------

%%% ------------------------------------------------------------
%%  \section{Scratch Variables and Function Variants}
%%% ------------------------------------------------------------

\NeedsTeXFormat{LaTeX2e}
\RequirePackage{expl3}
\ProvidesExplPackage{tabularray}{2021-06-05}{2021K}
  {Typeset tabulars and arrays with LaTeX3}

\RequirePackage{xparse}
\AtBeginDocument{\@ifpackageloaded{xcolor}{\RequirePackage{ninecolors}}{}}

\ExplSyntaxOn

%% Backport \tl_if_eq:NnTF for old texlive 2020
\cs_if_exist:NF \tl_if_eq:NnTF
  {
    \tl_new:N \l__tblr_backport_b_tl
    \prg_new_protected_conditional:Npnn \tl_if_eq:Nn #1 #2 { T, F, TF }
      {
        \group_begin:
          \tl_set:Nn \l__tblr_backport_b_tl {#2}
          \exp_after:wN
        \group_end:
        \if_meaning:w #1 \l__tblr_backport_b_tl
          \prg_return_true:
        \else:
          \prg_return_false:
        \fi:
      }
    \prg_generate_conditional_variant:Nnn \tl_if_eq:Nn { c } { TF, T, F }
  }

%% Compatible with texlive 2020
```

```
\cs_if_exist:NF \seq_map_indexed_function:NN
  {
    \cs_set_eq:NN \seq_map_indexed_function:NN \seq_indexed_map_function:NN
  }

\cs_generate_variant:Nn \msg_error:nnnn { nnVn }
\cs_generate_variant:Nn \prop_item:Nn { Ne, NV }
\cs_generate_variant:Nn \prop_put:Nnn { Nxn, Nxx, NxV }
\cs_generate_variant:Nn \regex_replace_all:NnN { NVN }
\cs_generate_variant:Nn \seq_map_indexed_inline:Nn { cn }
\cs_generate_variant:Nn \tl_const:Nn { ce }
\cs_generate_variant:Nn \tl_log:n { x }
\cs_generate_variant:Nn \tl_gput_right:Nn { Nf }
\prg_generate_conditional_variant:Nnn \clist_if_in:Nn { Nx } { TF }
\prg_generate_conditional_variant:Nnn \prop_if_in:Nn { c } { T }
\prg_generate_conditional_variant:Nnn \str_if_eq:nn { xn } { TF }
\prg_generate_conditional_variant:Nnn \tl_if_eq:nn { en } { T, TF }
\prg_generate_conditional_variant:Nnn \tl_if_head_eq_meaning:nN { VN } { T, TF }

\tl_new:N  \l__tblr_a_tl
\tl_new:N  \l__tblr_b_tl
\tl_new:N  \l__tblr_c_tl
\tl_new:N  \l__tblr_d_tl
\tl_new:N  \l__tblr_e_tl
\tl_new:N  \l__tblr_f_tl
\tl_new:N  \l__tblr_h_tl
\tl_new:N  \l__tblr_i_tl  % for row index
\tl_new:N  \l__tblr_j_tl  % for column index
\tl_new:N  \l__tblr_k_tl
\tl_new:N  \l__tblr_n_tl
\tl_new:N  \l__tblr_o_tl
\tl_new:N  \l__tblr_r_tl
\tl_new:N  \l__tblr_s_tl
\tl_new:N  \l__tblr_t_tl
\tl_new:N  \l__tblr_u_tl
\tl_new:N  \l__tblr_v_tl
\tl_new:N  \l__tblr_w_tl
\tl_new:N  \l__tblr_x_tl
\tl_new:N  \l__tblr_y_tl
\int_new:N \l__tblr_a_int
\int_new:N \l__tblr_c_int % for column number
\int_new:N \l__tblr_r_int % for row number
\dim_new:N \l__tblr_d_dim % for depth
\dim_new:N \l__tblr_h_dim % for height
\dim_new:N \l__tblr_o_dim
\dim_new:N \l__tblr_p_dim
\dim_new:N \l__tblr_q_dim
\dim_new:N \l__tblr_r_dim
\dim_new:N \l__tblr_s_dim
\dim_new:N \l__tblr_t_dim
\dim_new:N \l__tblr_v_dim
\dim_new:N \l__tblr_w_dim % for width
\box_new:N \l__tblr_a_box
\box_new:N \l__tblr_b_box
\box_new:N \l__tblr_c_box % for cell box
\box_new:N \l__tblr_d_box
```

```
%% Some counters for row and column numbering
\newcounter{rownum}
\newcounter{colnum}
\newcounter{rowcount}
\newcounter{colcount}

%%% ------------------------------------------------------------
%%  \section{Data Structures Based on Property Lists}
%%% ------------------------------------------------------------

\int_new:N \g_tblr_level_int % store table nesting level

\cs_new_protected:Npn \__tblr_clear_prop_lists:
  {
    \prop_gclear_new:c { g__tblr_text_ \int_use:N \g_tblr_level_int _prop }
    \prop_gclear_new:c { g__tblr_command_ \int_use:N \g_tblr_level_int _prop }
    \prop_gclear_new:c { g__tblr_table_ \int_use:N \g_tblr_level_int _prop }
    \prop_gclear_new:c { g__tblr_row_ \int_use:N \g_tblr_level_int _prop }
    \prop_gclear_new:c { g__tblr_column_ \int_use:N \g_tblr_level_int _prop }
    \prop_gclear_new:c { g__tblr_cell_ \int_use:N \g_tblr_level_int _prop }
    \prop_gclear_new:c { g__tblr_hline_ \int_use:N \g_tblr_level_int _prop }
    \prop_gclear_new:c { g__tblr_vline_ \int_use:N \g_tblr_level_int _prop }
  }

\cs_new_protected:Npn \__tblr_prop_gput:nnn #1 #2 #3
  {
    \prop_gput:cnn
      { g__tblr_#1_ \int_use:N \g_tblr_level_int _prop } { #2 } { #3 }
  }
\cs_generate_variant:Nn \__tblr_prop_gput:nnn { nnx, nnV, nxn, nxx, nxV }

\cs_new:Npn \__tblr_prop_item:nn #1 #2
  {
    \prop_item:cn { g__tblr_#1_ \int_use:N \g_tblr_level_int _prop } { #2 }
  }
\cs_generate_variant:Nn \__tblr_prop_item:nn { ne }

\cs_new_protected:Npn \__tblr_prop_if_in:nnT #1
  {
    \prop_if_in:cnT { g__tblr_#1_ \int_use:N \g_tblr_level_int _prop }
  }
\cs_new_protected:Npn \__tblr_prop_if_in:nnF #1
  {
    \prop_if_in:cnF { g__tblr_#1_ \int_use:N \g_tblr_level_int _prop }
  }
\cs_new_protected:Npn \__tblr_prop_if_in:nnTF #1
  {
    \prop_if_in:cnTF { g__tblr_#1_ \int_use:N \g_tblr_level_int _prop }
  }
\prg_generate_conditional_variant:Nnn \__tblr_prop_if_in:nn { nx } { T, F, TF }

\cs_new_protected:Npn \__tblr_prop_log:n #1
  {
    \prop_log:c { g__tblr_#1_ \int_use:N \g_tblr_level_int _prop }
  }
```

```
\cs_new_protected:Npn \__tblr_prop_map_inline:nn #1 #2
  {
    \prop_map_inline:cn { g__tblr_#1_ \int_use:N \g_tblr_level_int _prop } {#2}
  }


\cs_new_protected:Npn \__tblr_prop_gput_if_larger:nnn #1 #2 #3
  {
    \__tblr_gput_if_larger:cnn
      { g__tblr_#1_ \int_use:N \g_tblr_level_int _prop } { #2 } { #3 }
  }
\cs_generate_variant:Nn \__tblr_prop_gput_if_larger:nnn { nnx, nnV, nxn, nxx, nxV }

\cs_new_protected:Npn \__tblr_prop_gadd_dimen_value:nnn #1 #2 #3
  {
    \__tblr_gadd_dimen_value:cnn
      { g__tblr_#1_ \int_use:N \g_tblr_level_int _prop } { #2 } { #3 }
  }
\cs_generate_variant:Nn \__tblr_prop_gadd_dimen_value:nnn { nnx, nnV, nxn, nxx }

%% Put the dimension to the prop list only if it's larger than the old one

\tl_new:N \l__tblr_put_if_larger_tl

\cs_new_protected:Npn \__tblr_put_if_larger:Nnn #1 #2 #3
  {
    \tl_set:Nx \l__tblr_put_if_larger_tl { \prop_item:Nn #1 { #2 } }
    \bool_lazy_or:nnT
      { \tl_if_empty_p:N \l__tblr_put_if_larger_tl }
      { \dim_compare_p:nNn { #3 } > { \l__tblr_put_if_larger_tl } }
      { \prop_put:Nnn #1 { #2 } { #3 }  }
  }
\cs_generate_variant:Nn \__tblr_put_if_larger:Nnn { Nnx, Nxn, Nxx }

\cs_new_protected:Npn \__tblr_gput_if_larger:Nnn #1 #2 #3
  {
    \tl_set:Nx \l__tblr_put_if_larger_tl { \prop_item:Nn #1 { #2 } }
    \bool_lazy_or:nnT
      { \tl_if_empty_p:N \l__tblr_put_if_larger_tl }
      { \dim_compare_p:nNn { #3 } > { \l__tblr_put_if_larger_tl } }
      { \prop_gput:Nnn #1 { #2 } { #3 }  }
  }
\cs_generate_variant:Nn \__tblr_gput_if_larger:Nnn { Nnx, Nxn, Nxx, cnn }

%% Add the dimension to some key value of the prop list
%% #1: the prop list, #2: the key, #3: the dimen to add

\cs_new_protected:Npn \__tblr_add_dimen_value:Nnn #1 #2 #3
  {
    \prop_put:Nnx #1 { #2 } { \dim_eval:n { \prop_item:Nn #1 { #2 } + #3 } }
  }
\cs_generate_variant:Nn \__tblr_add_dimen_value:Nnn { cnn }

\cs_new_protected:Npn \__tblr_gadd_dimen_value:Nnn #1 #2 #3
  {
    \prop_gput:Nnx #1 { #2 } { \dim_eval:n { \prop_item:Nn #1 { #2 } + #3 } }
```

```
  }
\cs_generate_variant:Nn \__tblr_gadd_dimen_value:Nnn { cnn }

%%% ------------------------------------------------------------
%%  \section{Data Structures Based on Token Lists}
%%% ------------------------------------------------------------

\cs_new_protected:Npn \__tblr_clear_text_lists:
  {
    \__tblr_clear_one_text_lists:n { text }
    \__tblr_clear_one_text_lists:n { hline }
    \__tblr_clear_one_text_lists:n { vline }
  }

\cs_new_protected:Npn \__tblr_clear_one_text_lists:n #1
  {
    \clist_if_exist:cTF { g__tblr_#1_ \int_use:N \g_tblr_level_int _clist }
      {
        \clist_map_inline:cn { g__tblr_#1_ \int_use:N \g_tblr_level_int _clist }
          {
            \tl_gclear:c { g__tblr_text_ \int_use:N \g_tblr_level_int _#1_##1_tl }
          }
      }
      { \clist_new:c { g__tblr_#1_ \int_use:N \g_tblr_level_int _clist } }
  }

\cs_new_protected:Npn \__tblr_text_gput:nnn #1 #2 #3
  {
    \tl_gset:cn
      { g__tblr_text_ \int_use:N \g_tblr_level_int _#1_#2_tl } {#3}
    \clist_gput_right:cx { g__tblr_#1_ \int_use:N \g_tblr_level_int _clist } {#2}
  }
\cs_generate_variant:Nn \__tblr_text_gput:nnn { nne, nnV, nen, nee, neV }

\cs_new:Npn \__tblr_text_item:nn #1 #2
  {
    \tl_if_exist:cT { g__tblr_text_ \int_use:N \g_tblr_level_int _#1_#2_tl }
      {
        \exp_args:Nv \exp_not:n
          { g__tblr_text_ \int_use:N \g_tblr_level_int _#1_#2_tl }
      }
  }
\cs_generate_variant:Nn \__tblr_text_item:nn { ne }

\cs_new_protected:Npn \__tblr_text_gput_if_larger:nnn #1 #2 #3
  {
    \tl_set:Nx \l__tblr_put_if_larger_tl { \__tblr_text_item:nn {#1} {#2} }
    \bool_lazy_or:nnT
      { \tl_if_empty_p:N \l__tblr_put_if_larger_tl }
      { \dim_compare_p:nNn {#3} > { \l__tblr_put_if_larger_tl } }
      { \__tblr_text_gput:nnn {#1} {#2} {#3} }
  }
\cs_generate_variant:Nn \__tblr_text_gput_if_larger:nnn { nne, nnV, nen, nee, neV }

\cs_new_protected:Npn \__tblr_text_log:n #1
  {
```

```
    \clist_gremove_duplicates:c
      { g__tblr_#1_ \int_use:N \g_tblr_level_int _clist }
    \tl_log:n { ----------~----------~----------~----------~---------- }
    \clist_map_inline:cn { g__tblr_#1_ \int_use:N \g_tblr_level_int _clist }
      {
        \tl_log:x
          {
            \space { #1 ##1 } ~\space=>~\space { \__tblr_text_item:nn {#1} {##1} }
          }
      }
  }

%%% ------------------------------------------------------------
%%  \section{Data Structures Based on Integer Arrays}
%%% ------------------------------------------------------------

\msg_new:nnn { tabularray } { intarray-beyond-bound }
  { Position ~ #2 ~ is ~ beyond ~ the ~ bound ~ of ~ intarray ~ #1.}

\cs_new_protected:Npn \__tblr_intarray_gset:Nnn #1 #2 #3
  {
    \bool_lazy_or:nnTF
      { \int_compare_p:nNn {#2} < {0} }
      { \int_compare_p:nNn {#2} > {\intarray_count:N #1} }
      {
        \bool_if:NT \g__tblr_tracing_intarray_bool
          { \msg_warning:nnnn { tabularray } { intarray-beyond-bound } {#1} {#2} }
      }
      { \intarray_gset:Nnn #1 {#2} {#3} }
  }
\cs_generate_variant:Nn \__tblr_intarray_gset:Nnn { cnn }

%% #1: data name; #2: key name; #3: value type
\cs_new_protected:Npn \__tblr_data_new_key:nnn #1 #2 #3
  {
    \int_gincr:c { g__tblr_data_#1_key_count_int }
    \tl_const:ce
      {
        g__tblr_data_#1_key_name_
          \int_use:c { g__tblr_data_#1_key_count_int } _tl
      }
      { #2 }
    \tl_const:ce { g__tblr_data_#1_key_number_#2_tl }
      { \int_use:c { g__tblr_data_#1_key_count_int } }
    \tl_const:cn { g__tblr_data_#1_key_type_#2_tl } {#3}
  }

\int_new:N \g__tblr_data_row_key_count_int
\__tblr_data_new_key:nnn { row } { height }      { dim }
\__tblr_data_new_key:nnn { row } { coefficient } { dec }
\__tblr_data_new_key:nnn { row } { abovesep }    { dim }
\__tblr_data_new_key:nnn { row } { belowsep }    { dim }
\__tblr_data_new_key:nnn { row } { @row-height } { dim }
\__tblr_data_new_key:nnn { row } { @row-head }   { dim }
\__tblr_data_new_key:nnn { row } { @row-foot }   { dim }
\__tblr_data_new_key:nnn { row } { @row-upper }  { dim }
\__tblr_data_new_key:nnn { row } { @row-lower }  { dim }
```

```
\int_new:N \g__tblr_data_column_key_count_int
\__tblr_data_new_key:nnn { column } { width }       { dim }
\__tblr_data_new_key:nnn { column } { coefficient } { dec }
\__tblr_data_new_key:nnn { column } { leftsep }     { dim }
\__tblr_data_new_key:nnn { column } { rightsep }    { dim }
\__tblr_data_new_key:nnn { column } { @col-width }  { dim }

\int_new:N \g__tblr_data_cell_key_count_int
\__tblr_data_new_key:nnn { cell } { width }        { dim }
\__tblr_data_new_key:nnn { cell } { rowspan }      { int }
\__tblr_data_new_key:nnn { cell } { colspan }      { int }
\__tblr_data_new_key:nnn { cell } { halign }       { str }
\__tblr_data_new_key:nnn { cell } { valign }       { str }
\__tblr_data_new_key:nnn { cell } { background }   { str }
\__tblr_data_new_key:nnn { cell } { omit }         { int }
\__tblr_data_new_key:nnn { cell } { @cell-width }  { dim }
\__tblr_data_new_key:nnn { cell } { @cell-height } { dim }
\__tblr_data_new_key:nnn { cell } { @cell-depth }  { dim }

\clist_const:Nn \g__tblr_data_clist { row, column, cell }
\tl_const:Nn \g__tblr_data_row_count_tl { \c@rowcount }
\tl_const:Nn \g__tblr_data_column_count_tl { \c@colcount }
\tl_const:Nn \g__tblr_data_cell_count_tl { \c@rowcount * \c@colcount }
\tl_const:Nn \g__tblr_data_row_index_number_tl {1}
\tl_const:Nn \g__tblr_data_column_index_number_tl {1}
\tl_const:Nn \g__tblr_data_cell_index_number_tl {2}
\int_new:N \g__tblr_array_int

\cs_new_protected:Npn \__tblr_initial_table_data:
  {
    \clist_map_function:NN \g__tblr_data_clist \__tblr_initial_one_data:n
  }

\cs_new_protected:Npn \__tblr_initial_one_data:n #1
  {
    \int_gincr:N \g__tblr_array_int
    \intarray_new:cn { g__tblr_#1_ \int_use:N \g__tblr_array_int _intarray }
      {
        \int_use:c { g__tblr_data_#1_key_count_int }
          * \tl_use:c { g__tblr_data_#1_count_tl }
      }
    \cs_set_eq:cc { g__tblr_#1_ \int_use:N \g_tblr_level_int _intarray }
      { g__tblr_#1_ \int_use:N \g__tblr_array_int _intarray }
    %\intarray_log:c { g__tblr_#1_ \int_use:N \g_tblr_level_int _intarray }
  }

%% #1: data name; #2: data index; #3: key name
\cs_new:Npn \__tblr_data_key_to_int:nnn #1 #2 #3
  {
    ( #2 - 1 ) * \int_use:c { g__tblr_data_#1_key_count_int }
      + \tl_use:c { g__tblr_data_#1_key_number_#3_tl }
  }

%% #1: data name; #2: data index 1; #3: data index 2; #4: key name
\cs_new:Npn \__tblr_data_key_to_int:nnnn #1 #2 #3 #4
  {
```

```
    ( #2 - 1 ) * \c@colcount * \int_use:c { g__tblr_data_#1_key_count_int }
      + ( #3 - 1 ) * \int_use:c { g__tblr_data_#1_key_count_int }
      + \tl_use:c { g__tblr_data_#1_key_number_#4_tl }
  }

\int_new:N \l__tblr_key_count_int
\int_new:N \l__tblr_key_quotient_int
\int_new:N \l__tblr_key_quotient_two_int
\int_new:N \l__tblr_key_remainder_int

%% #1: data name; #2: array position;
%% #3: returning tl with index; #4: returning tl with key name
\cs_new:Npn \__tblr_data_int_to_key:nnNN #1 #2 #3 #4
  {
    \int_set_eq:Nc \l__tblr_key_count_int { g__tblr_data_#1_key_count_int }
    \int_set:Nn \l__tblr_key_quotient_int
      {
        \int_div_truncate:nn
          { #2 + \l__tblr_key_count_int - 1 } { \l__tblr_key_count_int }
      }
    \int_set:Nn \l__tblr_key_remainder_int
      {
        #2 + \l__tblr_key_count_int
          - \l__tblr_key_quotient_int * \l__tblr_key_count_int
      }
    \int_compare:nNnT { \l__tblr_key_remainder_int } = { 0 }
      { \int_set_eq:NN \l__tblr_key_remainder_int \l__tblr_key_count_int }
    \tl_set:Nx #3 { \int_use:N \l__tblr_key_quotient_int }
    \tl_set_eq:Nc #4
      { g__tblr_data_#1_key_name_ \int_use:N \l__tblr_key_remainder_int _tl }
  }

%% #1: data name; #2: array position;
%% #3: returning tl with index 1; #4: returning tl with index 2;
%% #5: returning tl with key name
\cs_new:Npn \__tblr_data_int_to_key:nnNNN #1 #2 #3 #4 #5
  {
    \int_set_eq:Nc \l__tblr_key_count_int { g__tblr_data_#1_key_count_int }
    \int_set:Nn \l__tblr_key_quotient_int
      {
        \int_div_truncate:nn
          { #2 + \l__tblr_key_count_int - 1 } { \l__tblr_key_count_int }
      }
    \int_set:Nn \l__tblr_key_remainder_int
      {
        #2 + \l__tblr_key_count_int
          - \l__tblr_key_quotient_int * \l__tblr_key_count_int
      }
    \int_compare:nNnT { \l__tblr_key_remainder_int } = { 0 }
      { \int_set_eq:NN \l__tblr_key_remainder_int \l__tblr_key_count_int }
    \tl_set_eq:Nc #5
      { g__tblr_data_#1_key_name_ \int_use:N \l__tblr_key_remainder_int _tl }
    \int_set:Nn \l__tblr_key_quotient_two_int
      {
        \int_div_truncate:nn
          { \l__tblr_key_quotient_int + \c@colcount - 1 } { \c@colcount }
      }
```

```
    \int_set:Nn \l__tblr_key_remainder_int
      {
        \l__tblr_key_quotient_int + \c@colcount
          - \l__tblr_key_quotient_two_int * \c@colcount
      }
    \int_compare:nNnT { \l__tblr_key_remainder_int } = { 0 }
      { \int_set_eq:NN \l__tblr_key_remainder_int \c@colcount }
    \tl_set:Nx #4 { \int_use:N \l__tblr_key_remainder_int }
    \tl_set:Nx #3 { \int_use:N \l__tblr_key_quotient_two_int }
  }

\tl_new:N \g__tblr_data_int_from_value_tl

%% #1: data name; #2: key name; #3: value
%% The result will be stored in \g__tblr_data_int_from_value_tl
\cs_new_protected:Npn \__tblr_data_int_from_value:nnn #1 #2 #3
  {
    \cs:w
      __tblr_data_int_from_ \tl_use:c { g__tblr_data_#1_key_type_#2_tl } :n
    \cs_end:
    {#3}
  }

%% #1: data name; #2: key name; #3: int
\cs_new:Npn \__tblr_data_int_to_value:nnn #1 #2 #3
  {
    \cs:w
      __tblr_data_int_to_ \tl_use:c { g__tblr_data_#1_key_type_#2_tl } :n
    \cs_end:
    {#3}
  }
\cs_generate_variant:Nn \__tblr_data_int_to_value:nnn { nne, nVe }

\cs_new_protected:Npn \__tblr_data_int_from_int:n #1
  {
    \tl_gset:Nn \g__tblr_data_int_from_value_tl {#1}
  }

\cs_new:Npn \__tblr_data_int_to_int:n #1
  {
    #1
  }

\cs_new_protected:Npn \__tblr_data_int_from_dim:n #1
  {
    \tl_gset:Nx \g__tblr_data_int_from_value_tl { \dim_to_decimal_in_sp:n {#1} }
  }

%% Return a dimension in pt so that it's easier to understand in tracing messages
\cs_new:Npn \__tblr_data_int_to_dim:n #1
  {
    %#1 sp
    %\dim_eval:n { #1 sp }
    \dim_to_decimal:n { #1 sp } pt
  }
```

```
\cs_new_protected:Npn \__tblr_data_int_from_dec:n #1
  {
    \tl_gset:Nx \g__tblr_data_int_from_value_tl
      { \dim_to_decimal_in_sp:n {#1 pt} }
  }

\cs_new:Npn \__tblr_data_int_to_dec:n #1
  {
    \dim_to_decimal:n {#1 sp}
  }

\int_new:N \g__tblr_data_str_value_count_int
\tl_set:cn { g__tblr_data_0_to_str_tl } { }

\cs_new_protected:Npn \__tblr_data_int_from_str:n #1
  {
    \tl_if_exist:cTF { g__tblr_data_#1_to_int_tl }
      {
        \tl_gset_eq:Nc \g__tblr_data_int_from_value_tl
          { g__tblr_data_#1_to_int_tl }
      }
      {
        \int_gincr:N \g__tblr_data_str_value_count_int
        \tl_gset:cx { g__tblr_data_#1_to_int_tl }
          { \int_use:N \g__tblr_data_str_value_count_int }
        \tl_gset:cx
          { g__tblr_data_ \int_use:N \g__tblr_data_str_value_count_int _to_str_tl }
          { #1 }
        \tl_gset:Nx \g__tblr_data_int_from_value_tl
          { \int_use:N \g__tblr_data_str_value_count_int }
      }
  }

\cs_new:Npn \__tblr_data_int_to_str:n #1
  {
    \tl_use:c { g__tblr_data_#1_to_str_tl }
  }

%% #1: data name; #2: data index; #3: key; #4: value
\cs_new_protected:Npn \__tblr_data_gput:nnnn #1 #2 #3 #4
  {
    \__tblr_data_int_from_value:nnn {#1} {#3} {#4}
    \__tblr_intarray_gset:cnn
      { g__tblr_#1_ \int_use:N \g_tblr_level_int _intarray }
      { \__tblr_data_key_to_int:nnn {#1} {#2} {#3} }
      { \g__tblr_data_int_from_value_tl }
  }
\cs_generate_variant:Nn \__tblr_data_gput:nnnn
  { nnne, nnnV, nenn, nene, nenV, nVnn }

%% #1: data name; #2: data index 1; #3: data index 2; #4: key; #5: value
\cs_new_protected:Npn \__tblr_data_gput:nnnnn #1 #2 #3 #4 #5
  {
    \__tblr_data_int_from_value:nnn {#1} {#4} {#5}
    \__tblr_intarray_gset:cnn
      { g__tblr_#1_ \int_use:N \g_tblr_level_int _intarray }
```

```
      { \__tblr_data_key_to_int:nnnn {#1} {#2} {#3} {#4} }
      { \g__tblr_data_int_from_value_tl }
  }
\cs_generate_variant:Nn \__tblr_data_gput:nnnnn
  { nnnne, nnnnV, neenn, neene, neenV, neeen, nVVnn }

%% #1: data name; #2: data index; #3: key
\cs_new:Npn \__tblr_data_item:nnn #1 #2 #3
  {
    \__tblr_data_int_to_value:nne {#1} {#3}
      {
        \intarray_item:cn { g__tblr_#1_ \int_use:N \g_tblr_level_int _intarray }
          { \__tblr_data_key_to_int:nnn {#1} {#2} {#3} }
      }
  }
\cs_generate_variant:Nn \__tblr_data_item:nnn { nen }

%% #1: data name; #2: data index 1; #3: data index 2; #4: key
\cs_new:Npn \__tblr_data_item:nnnn #1 #2 #3 #4
  {
    \__tblr_data_int_to_value:nne {#1} {#4}
      {
        \intarray_item:cn { g__tblr_#1_ \int_use:N \g_tblr_level_int _intarray }
          { \__tblr_data_key_to_int:nnnn {#1} {#2} {#3} {#4} }
      }
  }
\cs_generate_variant:Nn \__tblr_data_item:nnnn { neen }

\tl_new:N \l__tblr_data_key_tl
\tl_new:N \l__tblr_data_index_tl
\tl_new:N \l__tblr_data_index_two_tl

\cs_new_protected:Npn \__tblr_data_log:n #1
  {
    \use:c { __tblr_data_log_ \use:c { g__tblr_data_#1_index_number_tl } :n } {#1}
    \__tblr_prop_log:n {#1}
  }

\cs_new_protected:cpn { __tblr_data_log_1:n } #1
  {
    %\intarray_log:c { g__tblr_#1_ \int_use:N \g_tblr_level_int _intarray }
    \tl_set:Nx \l_tmpa_tl { g__tblr_#1_ \int_use:N \g_tblr_level_int _intarray }
    \tl_log:n { ----------~----------~----------~----------~---------- }
    \int_step_inline:nn
      { \intarray_count:c { \l_tmpa_tl } }
      {
        \__tblr_data_int_to_key:nnNN {#1} {##1}
          \l__tblr_data_index_tl \l__tblr_data_key_tl
        \tl_log:x
          {
            \space
            { #1 [\l__tblr_data_index_tl] / \l__tblr_data_key_tl }
            ~\space => ~\space
            {
              \__tblr_data_int_to_value:nVe {#1} \l__tblr_data_key_tl
                { \intarray_item:cn { \l_tmpa_tl } {##1} }
            }
```

```
          }
      }
  }

\cs_new_protected:cpn { __tblr_data_log_2:n } #1
  {
    %\intarray_log:c { g__tblr_#1_ \int_use:N \g_tblr_level_int _intarray }
    \tl_set:Nx \l_tmpa_tl { g__tblr_#1_ \int_use:N \g_tblr_level_int _intarray }
    \tl_log:n { ----------~----------~----------~----------~---------- }
    \int_step_inline:nn
      { \intarray_count:c { \l_tmpa_tl } }
      {
        \__tblr_data_int_to_key:nnNNN {#1} {##1}
          \l__tblr_data_index_tl \l__tblr_data_index_two_tl \l__tblr_data_key_tl
        \tl_log:x
          {
            \space
            {
              #1 [\l__tblr_data_index_tl][\l__tblr_data_index_two_tl]
                 / \l__tblr_data_key_tl
            }
            ~\space => ~\space
            {
              \__tblr_data_int_to_value:nVe {#1} \l__tblr_data_key_tl
                { \intarray_item:cn { \l_tmpa_tl } {##1} }
            }
          }
      }
  }

%% #1: data name; #2: row index; #3: key; #4: value
\cs_new_protected:Npn \__tblr_data_gput_if_larger:nnnn #1 #2 #3 #4
  {
    \__tblr_data_int_from_value:nnn {#1} {#3} {#4}
    \__tblr_array_gput_if_larger:cnn
      { g__tblr_#1_ \int_use:N \g_tblr_level_int _intarray }
      { \__tblr_data_key_to_int:nnn {#1} {#2} {#3} }
      { \g__tblr_data_int_from_value_tl }
  }
\cs_generate_variant:Nn \__tblr_data_gput_if_larger:nnnn { nnne, nnnV, nene, nenV }

\cs_new_protected:Npn \__tblr_array_gput_if_larger:Nnn #1 #2 #3
  {
    \int_compare:nNnT {#3} > { \intarray_item:Nn #1 {#2} }
      { \__tblr_intarray_gset:Nnn #1 {#2} {#3} }
  }
\cs_generate_variant:Nn \__tblr_array_gput_if_larger:Nnn { cnn }

%% #1: data name; #2: data index; #3: key; #4: value
\cs_new_protected:Npn \__tblr_data_gadd_dimen_value:nnnn #1 #2 #3 #4
  {
    \__tblr_data_int_from_value:nnn {#1} {#3} {#4}
    \__tblr_array_gadd_value:cnn
      { g__tblr_#1_ \int_use:N \g_tblr_level_int _intarray }
      { \__tblr_data_key_to_int:nnn {#1} {#2} {#3} }
      { \g__tblr_data_int_from_value_tl }
  }
```

```
\cs_generate_variant:Nn \__tblr_data_gadd_dimen_value:nnnn { nnne, nnnV, nene }

\cs_new_protected:Npn \__tblr_array_gadd_value:Nnn #1 #2 #3
  {
    \__tblr_intarray_gset:Nnn #1 {#2} { \intarray_item:Nn #1 {#2} + #3 }
  }
\cs_generate_variant:Nn \__tblr_array_gadd_value:Nnn { cnn }

\bool_new:N \g__tblr_use_intarray_bool
\bool_set_true:N \g__tblr_use_intarray_bool

\AtBeginDocument
  {
    \bool_if:NF \g__tblr_use_intarray_bool
      {
        \cs_set_protected:Npn \__tblr_data_gput:nnnn #1 #2 #3 #4
          {
            \__tblr_prop_gput:nnn {#1} { [#2] / #3 } {#4}
          }
        \cs_set_protected:Npn \__tblr_data_gput:nnnnn #1 #2 #3 #4 #5
          {
            \__tblr_prop_gput:nnn {#1} { [#2][#3] / #4 } {#5}
          }
        \cs_set:Npn \__tblr_data_item:nnn #1 #2 #3
          {
            \__tblr_prop_item:nn {#1} { [#2] / #3 }
          }
        \cs_set:Npn \__tblr_data_item:nnnn #1 #2 #3 #4
          {
            \__tblr_prop_item:nn {#1} { [#2][#3] / #4 }
          }
        \cs_set_protected:Npn \__tblr_data_log:n #1
          {
            \__tblr_prop_log:n {#1}
          }
        \cs_set_protected:Npn \__tblr_data_gput_if_larger:nnnn #1 #2 #3 #4
          {
            \__tblr_prop_gput_if_larger:nnn {#1} { [#2] / #3 } {#4}
          }
        \cs_set_protected:Npn \__tblr_data_gput_if_larger:nnnnn #1 #2 #3 #4 #5
          {
            \__tblr_prop_gput_if_larger:nnn {#1} { [#2][#3] / #4 } {#5}
          }
        \cs_set_protected:Npn \__tblr_data_gadd_dimen_value:nnnn #1 #2 #3 #4
          {
            \__tblr_prop_gadd_dimen_value:nnn {#1} { [#2] / #3 } {#4}
          }
        \cs_set_protected:Npn \__tblr_data_gadd_dimen_value:nnnnn #1 #2 #3 #4 #5
          {
            \__tblr_prop_gadd_dimen_value:nnn {#1} { [#2][#3] / #4 } {#5}
          }
      }
  }

%%% ------------------------------------------------------------
%%  \section{Child Selectors}
%%% ------------------------------------------------------------
```

```
\clist_new:N \g_tblr_used_child_selectors_clist

\tl_new:N \l__tblr_childs_arg_spec_tl

\msg_new:nnn { tabularray } { used-child-selector }
  { Child ~ selector ~ name ~ "#1" ~ has ~ been ~ used! }

\NewDocumentCommand \NewChildSelector { m O{0} o m }
  {
    \__tblr_new_child_selector_aux:xnnn { \tl_trim_spaces:n {#1} } {#2} {#3} {#4}
  }

\cs_new_protected:Npn \__tblr_new_child_selector_aux:nnnn #1 #2 #3 #4
  {
    \clist_if_in:NnTF \g_tblr_used_child_selectors_clist { #1 }
      {
        \msg_error:nnn { tabularray } { used-child-selector } { #1 }
        \clist_log:N \g_tblr_used_child_selectors_clist
      }
      {
        \__tblr_make_xparse_arg_spec:nnN { #2 } { #3 } \l__tblr_childs_arg_spec_tl
        \exp_args:NcV \NewDocumentCommand
          { __tblr_child_selector_ #1 :w } \l__tblr_childs_arg_spec_tl { #4 }
        \clist_gput_right:Nn \g_tblr_used_child_selectors_clist { #1 }
      }
  }
\cs_generate_variant:Nn \__tblr_new_child_selector_aux:nnnn { xnnn }

%% #1: argument number, #2: optional argument default, #3: result tl
\cs_new_protected:Npn \__tblr_make_xparse_arg_spec:nnN #1 #2 #3
  {
    \tl_clear:N #3
    \int_compare:nNnT { #1 } > { 0 }
      {
        \IfValueTF { #2 }
          { \tl_set:Nn #3 { O{#2} } }
          { \tl_set:Nn #3 { m } }
        \tl_put_right:Nx #3 { \prg_replicate:nn { #1 - 1 } { m } }
      }
  }

\clist_new:N \l_tblr_childs_clist
\tl_new:N \l_tblr_childs_total_tl

\NewChildSelector { odd }
  {
    \int_step_inline:nnnn {1} {2} { \l_tblr_childs_total_tl }
      { \clist_put_right:Nn \l_tblr_childs_clist {##1} }
  }

\NewChildSelector { even }
  {
    \int_step_inline:nnnn {2} {2} { \l_tblr_childs_total_tl }
      { \clist_put_right:Nn \l_tblr_childs_clist {##1} }
  }
```

```
\regex_const:Nn \c__tblr_split_selector_name_regex { ^ ( [A-Za-z] {2,} ) ( . * ) }
\seq_new:N \l__tblr_childs_split_seq
\seq_new:N \l__tblr_childs_regex_seq
\tl_new:N \l__tblr_childs_selector_tl

%% #1, child specifications; #2, total number.
%% The result will be put into \l_tblr_childs_clist
\cs_new_protected:Npn \__tblr_get_childs:nn #1 #2
  {
    \clist_clear:N \l_tblr_childs_clist
    \tl_set:Nx \l_tblr_childs_total_tl {#2}
    \regex_extract_once:NnNTF \c__tblr_split_selector_name_regex {#1}
      \l__tblr_childs_regex_seq
      {
        \tl_set:No \l__tblr_childs_selector_tl
          {
            \cs:w
            __tblr_child_selector_ \seq_item:Nn \l__tblr_childs_regex_seq {2} :w
            \cs_end:
          }
        \exp_args:Nx \l__tblr_childs_selector_tl
          { \seq_item:Nn \l__tblr_childs_regex_seq{3} }
      }
      {
        \tl_if_eq:nnTF {#1} {-}
          { \__tblr_get_childs_normal:nn {1-#2} {#2} }
          { \__tblr_get_childs_normal:nn {#1} {#2} }
      }
    %\clist_log:N \l_tblr_childs_clist
  }
\cs_generate_variant:Nn \__tblr_get_childs:nn { nx }

\cs_new_protected:Npn \__tblr_get_childs_normal:nn #1 #2
  {
    \seq_set_split:Nnn \l__tblr_childs_split_seq {,} {#1}
    \seq_map_inline:Nn \l__tblr_childs_split_seq
      {
        \tl_if_in:nnTF {##1} {-}
          { \__tblr_get_childs_normal_aux:w ##1 \scan_stop }
          { \__tblr_get_childs_normal_aux:w ##1 - ##1 \scan_stop }
      }
  }

\cs_new_protected_nopar:Npn \__tblr_get_childs_normal_aux:w #1 - #2 \scan_stop
  {
    \int_step_inline:nnn {#1} {#2}
      { \clist_put_right:Nn \l_tblr_childs_clist {##1} }
  }

%%% ------------------------------------------------------------
%%  \section{New Table Commands}
%%% ------------------------------------------------------------

%% We need some commands to modify table/row/column/cell specifications.
%% These commands must be defined with \NewTableCommand command,
%% so that we could extract them, execute them once, then disable them.
```

```
\clist_new:N \g__tblr_table_commands_clist

\msg_new:nnn { tabularray } { defined-table-command }
  { Table ~ commnad ~ #1 has ~ been ~ defined! }

\NewDocumentCommand \NewTableCommand { m O{0} o m }
  {
    \clist_if_in:NnTF \g__tblr_table_commands_clist { #1 }
      {
        \msg_error:nnn { tabularray } { defined-table-command } { #1 }
        \clist_log:N \g__tblr_table_commands_clist
      }
      {
        \__tblr_make_xparse_arg_spec:nnN { #2 } { #3 } \l__tblr_a_tl
        \exp_args:NcV \NewDocumentCommand
          { __tblr_table_command_ \cs_to_str:N #1 :w } \l__tblr_a_tl { #4 }
        \exp_args:NcV \NewDocumentCommand
          { __tblr_table_command_ \cs_to_str:N #1 _gobble :w } \l__tblr_a_tl { }
        \IfValueTF { #3 }
          {
            \tl_gset:cn { g__tblr_table_cmd_ \cs_to_str:N #1 _arg_numb_tl } {-#2}
          }
          {
            \tl_gset:cn { g__tblr_table_cmd_ \cs_to_str:N #1 _arg_numb_tl } {#2}
          }
        \clist_gput_right:Nn \g__tblr_table_commands_clist { #1 }
      }
  }

\cs_new_protected:Npn \__tblr_enable_table_commands:
  {
    \clist_map_inline:Nn \g__tblr_table_commands_clist
      { \cs_set_eq:Nc ##1 { __tblr_table_command_ \cs_to_str:N ##1 :w } }
  }

\cs_new_protected:Npn \__tblr_disable_table_commands:
  {
    \clist_map_inline:Nn \g__tblr_table_commands_clist
      { \cs_set_eq:Nc ##1 { __tblr_table_command_ \cs_to_str:N ##1 _gobble:w } }
  }

\cs_new_protected:Npn \__tblr_execute_table_commands:
  {
    \__tblr_prop_map_inline:nn { command }
      {
        \__tblr_set_row_col_from_key_name:w ##1
        ##2
      }
    \LogTblrTracing { cell }
  }

\cs_new_protected:Npn \__tblr_set_row_col_from_key_name:w [#1][#2]
  {
    \int_set:Nn \c@rownum {#1}
    \int_set:Nn \c@colnum {#2}
  }
```

```
%%% ------------------------------------------------------------
%%  \section{New Dash Styles}
%%% ------------------------------------------------------------

%% \NewDashStyle commands

\dim_zero_new:N \rulewidth
\dim_set:Nn \rulewidth {0.4pt}

\prop_gset_from_keyval:Nn \g__tblr_defined_hdash_styles_prop
  { solid = \hrule height \rulewidth }
\prop_gset_from_keyval:Nn \g__tblr_defined_vdash_styles_prop
  { solid = \vrule width \rulewidth }

\NewDocumentCommand \NewDashStyle { m m }
  {
    \seq_set_split:Nnn \l_tmpa_seq { ~ } {#2}
    \tl_set:Nx \l__tblr_a_tl { \seq_item:Nn \l_tmpa_seq {1} }
    \tl_set:Nx \l__tblr_b_tl { \seq_item:Nn \l_tmpa_seq {2} }
    \tl_set:Nx \l__tblr_c_tl { \seq_item:Nn \l_tmpa_seq {3} }
    \tl_set:Nx \l__tblr_d_tl { \seq_item:Nn \l_tmpa_seq {4} }
    \tl_if_eq:NnT \l__tblr_a_tl { on }
      {
        \tl_if_eq:NnT \l__tblr_c_tl { off }
          {
            \__tblr_dash_style_make_boxes:nxx {#1}
              { \dim_eval:n {\l__tblr_b_tl} } { \dim_eval:n {\l__tblr_d_tl} }
          }
      }
  }

\cs_new_protected:Npn \__tblr_dash_style_make_boxes:nnn #1 #2 #3
  {
    \dim_set:Nn \l_tmpa_dim { #2 + #3 }
    \tl_set:Nn \l__tblr_h_tl { \hbox_to_wd:nn }
    \tl_put_right:Nx \l__tblr_h_tl { { \dim_use:N \l_tmpa_dim } }
    \tl_put_right:Nn \l__tblr_h_tl
      {
        { \hss \vbox:n { \hbox_to_wd:nn {#2} {} \hrule height \rulewidth } \hss }
      }
    \prop_gput:NnV \g__tblr_defined_hdash_styles_prop {#1} \l__tblr_h_tl
    %\prop_log:N \g__tblr_defined_hdash_styles_prop
    \tl_set:Nn \l__tblr_v_tl { \vbox_to_ht:nn }
    \tl_put_right:Nx \l__tblr_v_tl { { \dim_use:N \l_tmpa_dim } }
    \tl_put_right:Nn \l__tblr_v_tl
      {
        { \vss \hbox:n { \vbox_to_ht:nn {#2} {} \vrule width \rulewidth } \vss }
      }
    \prop_gput:NnV \g__tblr_defined_vdash_styles_prop {#1} \l__tblr_v_tl
    %\prop_log:N \g__tblr_defined_vdash_styles_prop
  }
\cs_generate_variant:Nn \__tblr_dash_style_make_boxes:nnn { nxx }

\cs_new_protected:Npn \__tblr_get_hline_dash_style:N #1
  {
    \tl_set:Nx \l_tmpa_tl
```

```
      { \prop_item:NV \g__tblr_defined_hdash_styles_prop #1 }
    \tl_if_empty:NF \l_tmpa_tl { \tl_set_eq:NN #1 \l_tmpa_tl }
  }

\cs_new_protected:Npn \__tblr_get_vline_dash_style:N #1
  {
    \tl_set:Nx \l_tmpa_tl
      { \prop_item:NV \g__tblr_defined_vdash_styles_prop #1 }
    \tl_if_empty:NF \l_tmpa_tl { \tl_set_eq:NN #1 \l_tmpa_tl }
  }

\NewDashStyle {dashed} {on ~ 2pt ~ off ~ 2pt}
\NewDashStyle {dotted} {on ~ 0.4pt ~ off ~ 1pt}

%%% ------------------------------------------------------------
%%  \section{Set Hlines and Vlines}
%%% ------------------------------------------------------------

\tl_set:Nn \@tblr@dash { dash }
\tl_set:Nn \@tblr@text { text }

\regex_const:Nn \c__tblr_is_color_key_regex { ^[A-Za-z] }

%% \SetHlines command for setting every hline in the table
\NewTableCommand \SetHlines [3] [+]
  {
    \tblr_set_every_hline:nnn {#1} {#2} {#3}
  }

%% We put all code inside a group to avoid affecting other table commands
\cs_new_protected:Npn \tblr_set_every_hline:nnn #1 #2 #3
  {
    \group_begin:
    \int_step_inline:nn { \int_eval:n { \c@rowcount + 1 } }
      {
        \int_set:Nn \c@rownum {##1}
        \tblr_set_hline:nnn {#1} {#2} {#3}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_every_hline in different ways
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_every_hline_aux:n #1
  {
    \tl_if_head_is_group:nTF {#1}
      {
        \int_compare:nNnTF { \tl_count:n {#1} } = {3}
          { \tblr_set_every_hline:nnn #1 }
          { \tblr_set_every_hline:nnn {1} #1 }
      }
      { \tblr_set_every_hline:nnn {1} {-} {#1} }
  }

%% Add \SetHline, \hline and \cline commands
```

```
\tl_new:N \l__tblr_hline_count_tl % the count of all hlines
\tl_new:N \l__tblr_hline_num_tl   % the index of the hline
\tl_new:N \l__tblr_hline_cols_tl  % the columns of the hline
\tl_new:N \l__tblr_hline_dash_tl  % dash style
\tl_new:N \l__tblr_hline_fg_tl    % dash foreground
\tl_new:N \l__tblr_hline_wd_tl    % dash width

\NewTableCommand \cline [2] [] { \SetHline [=] {#2} {#1} }

\NewTableCommand \hline [1] [] { \SetHline [+] {-} {#1} }

%% #1: the index of the hline (may be + or =)
%% #2: which columns of the hline, separate by commas
%% #3: key=value pairs
\NewTableCommand \SetHline [3] [+]
  {
    \tblr_set_hline:nnn {#1} {#2} {#3}
  }

%% We need to check "text" key first
%% If it does exist and has empty value, then do nothing
\cs_new_protected:Npn \tblr_set_hline:nnn #1 #2 #3
  {
    \group_begin:
    \keys_set_groups:nnn { tblr-hline } { text } {#3}
    \tl_if_eq:NnF \l__tblr_hline_dash_tl { \exp_not:N \@tblr@text }
      {
        \__tblr_set_hline_num:n {#1}
        \tl_clear:N \l__tblr_hline_dash_tl
        \keys_set:nn { tblr-hline } { dash = solid, #3 }
        \__tblr_set_hline_cmd:n {#2}
      }
    \group_end:
  }

\cs_new_protected:Npn \tblr_set_hline:nnnn #1 #2 #3 #4
  {
    \group_begin:
    \__tblr_get_childs:nx {#1} { \int_eval:n { \c@rowcount + 1 } }
    \clist_map_inline:Nn \l_tblr_childs_clist
      {
        \int_set:Nn \c@rownum {##1}
        \tblr_set_hline:nnn {#2} {#3} {#4}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_hline in different ways
%% Note that #1 always includes an outer pair of braces
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_hline_aux:nn #1 #2
  {
    \tl_if_head_is_group:nTF {#2}
      {
        \int_compare:nNnTF { \tl_count:n {#2} } = {3}
          { \tblr_set_hline:nnnn #1 #2 }
```

```
          { \tblr_set_hline:nnnn #1 {1} #2 }
      }
      { \tblr_set_hline:nnnn #1 {1} {-} {#2} }
  }
\cs_generate_variant:Nn \__tblr_set_hline_aux:nn { Vn }

%% #1: the index of hline to set (may be + or =)
\cs_new_protected:Npn \__tblr_set_hline_num:n #1
  {
    \tl_clear:N \l__tblr_hline_num_tl
    \tl_set:Nx \l__tblr_hline_count_tl
      { \__tblr_text_item:ne { hline } { [\int_use:N \c@rownum] / @hline-count } }
    %% \l__tblr_hline_count_tl may be empty when rowspec has extra |'s
    \int_compare:nNnTF { \l__tblr_hline_count_tl + 0 } = {0}
      {
        \tl_set:Nx \l__tblr_hline_num_tl { 1 }
        \__tblr_text_gput:nen { hline }
          { [\int_use:N \c@rownum] / @hline-count } { 1 }
      }
      {
        \tl_if_eq:nnTF {#1} {+}
          { \__tblr_set_hline_num_incr: }
          {
            \tl_if_eq:nnTF {#1} {=}
              { \tl_set_eq:NN \l__tblr_hline_num_tl \l__tblr_hline_count_tl }
              {
                \int_compare:nNnTF {#1} > { \l__tblr_hline_count_tl }
                  { \__tblr_set_hline_num_incr: }
                  { \tl_set:Nn \l__tblr_hline_num_tl {#1} }
              }
          }
      }
  }

\cs_new_protected:Npn \__tblr_set_hline_num_incr:
  {
    \tl_set:Nx \l__tblr_hline_count_tl
      { \int_eval:n { \l__tblr_hline_count_tl + 1 } }
    \__tblr_text_gput:nee { hline }
      { [\int_use:N \c@rownum] / @hline-count } { \l__tblr_hline_count_tl }
    \tl_set_eq:NN \l__tblr_hline_num_tl \l__tblr_hline_count_tl
  }

\keys_define:nn { tblr-hline }
  {
    dash .code:n = \tl_set:Nn \l__tblr_hline_dash_tl { \exp_not:N \@tblr@dash #1 },
    text .code:n = \tl_set:Nn \l__tblr_hline_dash_tl { \exp_not:N \@tblr@text #1 },
    text .groups:n = { text },
    wd .code:n = \tl_set:Nn \l__tblr_hline_wd_tl { \dim_eval:n {#1} },
    fg .code:n = \tl_set:Nn \l__tblr_hline_fg_tl {#1},
    baseline .code:n = \__tblr_hline_set_baseline:n {#1},
    unknown .code:n = \__tblr_hline_unknown_key:V \l_keys_key_str,
  }

\cs_new_protected:Npn \__tblr_hline_unknown_key:n #1
  {
    \prop_if_in:NnTF \g__tblr_defined_hdash_styles_prop {#1}
```

```
      { \tl_set:Nn \l__tblr_hline_dash_tl { \exp_not:N \@tblr@dash #1 } }
      {
        \regex_match:NnTF \c__tblr_is_color_key_regex {#1}
          { \tl_set:Nn \l__tblr_hline_fg_tl {#1} }
          {
            \tl_set_rescan:Nnn \l__tblr_v_tl {} {#1}
            \tl_set:Nn \l__tblr_hline_wd_tl { \dim_eval:n {\l__tblr_v_tl} }
          }
      }
  }
\cs_generate_variant:Nn \__tblr_hline_unknown_key:n { V }

\cs_new_protected_nopar:Npn \__tblr_set_hline_cmd:n #1
  {
    \__tblr_get_childs:nx {#1} { \int_use:N \c@colcount }
    \clist_map_inline:Nn \l_tblr_childs_clist
      {
        \__tblr_text_gput:nee { hline }
          { [\int_use:N \c@rownum][##1](\l__tblr_hline_num_tl) / @dash }
          { \l__tblr_hline_dash_tl }
        \tl_if_empty:NF \l__tblr_hline_wd_tl
          {
            \__tblr_text_gput:nee { hline }
              { [\int_use:N \c@rownum][##1](\l__tblr_hline_num_tl) / wd }
              { \l__tblr_hline_wd_tl }
          }
        \tl_if_empty:NF \l__tblr_hline_fg_tl
          {
            \__tblr_text_gput:nee { hline }
              { [\int_use:N \c@rownum][##1](\l__tblr_hline_num_tl) / fg }
              { \l__tblr_hline_fg_tl }
          }
      }
  }

\NewTableCommand \firsthline [1] [] { \SetHline [+] {-} { #1, baseline=below } }
\NewTableCommand \lasthline [1] [] { \SetHline [+] {-} { #1, baseline=above } }

\cs_new_protected:Npn \__tblr_hline_set_baseline:n #1
  {
    \tl_if_eq:nnTF {#1} {above}
      {
        \__tblr_prop_gput:nnx { table }
          { baseline } { \int_eval:n { \c@rownum - 1 } }
      }
      {
        \tl_if_eq:nnT {#1} {below}
          {
            \__tblr_prop_gput:nnx { table } { baseline } { \int_use:N \c@rownum }
          }
      }
  }

%% \SetVlines command for setting every vline in the table
\NewTableCommand \SetVlines [3] [+]
  {
    \tblr_set_every_vline:nnn {#1} {#2} {#3}
```

```
  }

%% We put all code inside a group to avoid affecting other table commands
\cs_new_protected:Npn \tblr_set_every_vline:nnn #1 #2 #3
  {
    \group_begin:
    \int_step_inline:nn { \int_eval:n { \c@colcount + 1 } }
      {
        \int_set:Nn \c@colnum {##1}
        \tblr_set_vline:nnn {#1} {#2} {#3}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_every_vline in different ways
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_every_vline_aux:n #1
  {
    \tl_if_head_is_group:nTF {#1}
      {
        \int_compare:nNnTF { \tl_count:n {#1} } = {3}
          { \tblr_set_every_vline:nnn #1 }
          { \tblr_set_every_vline:nnn {1} #1 }
      }
      { \tblr_set_every_vline:nnn {1} {-} {#1} }
  }

%% Add \SetVline, \vline and \rline commands

\tl_new:N \l__tblr_vline_count_tl % the count of all vlines
\tl_new:N \l__tblr_vline_num_tl   % the index of the vline
\tl_new:N \l__tblr_vline_rows_tl  % the rows of the vline
\tl_new:N \l__tblr_vline_dash_tl  % dash style
\tl_new:N \l__tblr_vline_fg_tl    % dash foreground
\tl_new:N \l__tblr_vline_wd_tl    % dash width

\NewTableCommand \rline [2] [] { \SetVline [=] {#2} {#1} }

\NewTableCommand \vline [1] [] { \SetVline [+] {-} {#1} }

%% #1: the index of the vline (may be + or =)
%% #2: which rows of the vline, separate by commas
%% #3: key=value pairs
\NewTableCommand \SetVline [3] [+]
  {
    \tblr_set_vline:nnn {#1} {#2} {#3}
  }

%% We need to check "text" key first
%% If it does exist and has empty value, then do nothing
\cs_new_protected:Npn \tblr_set_vline:nnn #1 #2 #3
  {
    \group_begin:
    \keys_set_groups:nnn { tblr-vline } { text } {#3}
    \tl_if_eq:NnF \l__tblr_vline_dash_tl { \exp_not:N \@tblr@text }
      {
```

```
        \__tblr_set_vline_num:n {#1}
        \tl_clear:N \l__tblr_vline_dash_tl
        \keys_set:nn { tblr-vline } { dash = solid, #3 }
        \__tblr_set_vline_cmd:n {#2}
      }
    \group_end:
  }

\cs_new_protected:Npn \tblr_set_vline:nnnn #1 #2 #3 #4
  {
    \group_begin:
    \__tblr_get_childs:nx {#1} { \int_eval:n { \c@colcount + 1} }
    \clist_map_inline:Nn \l_tblr_childs_clist
      {
        \int_set:Nn \c@colnum {##1}
        \tblr_set_vline:nnn {#2} {#3} {#4}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_vline in different ways
%% Note that #1 always includes an outer pair of braces
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_vline_aux:nn #1 #2
  {
    \tl_if_head_is_group:nTF {#2}
      {
        \int_compare:nNnTF { \tl_count:n {#2} } = {3}
          { \tblr_set_vline:nnnn #1 #2 }
          { \tblr_set_vline:nnnn #1 {1} #2 }
      }
      { \tblr_set_vline:nnnn #1 {1} {-} {#2} }
  }
\cs_generate_variant:Nn \__tblr_set_vline_aux:nn { Vn }

%% #1: the index of vline to set (may be + or =)
\cs_new_protected:Npn \__tblr_set_vline_num:n #1
  {
    \tl_clear:N \l__tblr_vline_num_tl
    \tl_set:Nx \l__tblr_vline_count_tl
      { \__tblr_text_item:ne { vline } { [\int_use:N \c@colnum] / @vline-count } }
    %% \l__tblr_vline_count_tl may be empty when colspec has extra |'s
    \int_compare:nNnTF { \l__tblr_vline_count_tl + 0 } = {0}
      {
        \tl_set:Nx \l__tblr_vline_num_tl { 1 }
        \__tblr_text_gput:nen { vline }
          { [\int_use:N \c@colnum] / @vline-count } { 1 }
      }
      {
        \tl_if_eq:nnTF {#1} {+}
          { \__tblr_set_vline_num_incr: }
          {
            \tl_if_eq:nnTF {#1} {=}
              { \tl_set_eq:NN \l__tblr_vline_num_tl \l__tblr_vline_count_tl }
              {
                \int_compare:nNnTF {#1} > { \l__tblr_vline_count_tl }
                  { \__tblr_set_vline_num_incr: }
```

```
                { \tl_set:Nn \l__tblr_vline_num_tl {#1} }
            }
        }
    }
  }

\cs_new_protected:Npn \__tblr_set_vline_num_incr:
  {
    \tl_set:Nx \l__tblr_vline_count_tl
      { \int_eval:n { \l__tblr_vline_count_tl + 1 } }
    \__tblr_text_gput:nee { vline }
      { [\int_use:N \c@colnum] / @vline-count } { \l__tblr_vline_count_tl }
    \tl_set_eq:NN \l__tblr_vline_num_tl \l__tblr_vline_count_tl
  }

\keys_define:nn { tblr-vline }
  {
    dash .code:n = \tl_set:Nn \l__tblr_vline_dash_tl { \exp_not:N \@tblr@dash #1 },
    text .code:n = \tl_set:Nn \l__tblr_vline_dash_tl { \exp_not:N \@tblr@text #1 },
    text .groups:n = { text },
    wd .code:n = \tl_set:Nn \l__tblr_vline_wd_tl { \dim_eval:n {#1} },
    fg .code:n = \tl_set:Nn \l__tblr_vline_fg_tl {#1},
    unknown .code:n = \__tblr_vline_unknown_key:V \l_keys_key_str,
  }

\cs_new_protected:Npn \__tblr_vline_unknown_key:n #1
  {
    \prop_if_in:NnTF \g__tblr_defined_vdash_styles_prop {#1}
      { \tl_set:Nn \l__tblr_vline_dash_tl { \exp_not:N \@tblr@dash #1 } }
      {
        \regex_match:NnTF \c__tblr_is_color_key_regex {#1}
          { \tl_set:Nn \l__tblr_vline_fg_tl {#1} }
          {
            \tl_set_rescan:Nnn \l__tblr_v_tl {} {#1}
            \tl_set:Nn \l__tblr_vline_wd_tl { \dim_eval:n {\l__tblr_v_tl} }
          }
      }
  }
\cs_generate_variant:Nn \__tblr_vline_unknown_key:n { V }

\cs_new_protected_nopar:Npn \__tblr_set_vline_cmd:n #1
  {
    \__tblr_get_childs:nx {#1} { \int_use:N \c@rowcount }
    \clist_map_inline:Nn \l_tblr_childs_clist
      {
        \__tblr_text_gput:nee { vline }
          { [##1][\int_use:N \c@colnum](\l__tblr_vline_num_tl) / @dash }
          { \l__tblr_vline_dash_tl }
        \tl_if_empty:NF \l__tblr_vline_wd_tl
          {
            \__tblr_text_gput:nee { vline }
              { [##1][\int_use:N \c@colnum](\l__tblr_vline_num_tl) / wd }
              { \l__tblr_vline_wd_tl }
          }
        \tl_if_empty:NF \l__tblr_vline_fg_tl
          {
            \__tblr_text_gput:nee { vline }
```

```
              { [##1][\int_use:N \c@colnum](\l__tblr_vline_num_tl) / fg }
              { \l__tblr_vline_fg_tl }
          }
      }
  }

%%% ------------------------------------------------------------
%%  \section{Set Cells}
%%% ------------------------------------------------------------

%% \SetCells command for setting every cell in the table
\NewTableCommand \SetCells [2] []
  {
    \tblr_set_every_cell:nn {#1} {#2}
  }

%% We put all code inside a group to avoid affecting other table commands
\cs_new_protected:Npn \tblr_set_every_cell:nn #1 #2
  {
    \group_begin:
    \int_step_inline:nn { \c@rowcount }
      {
        \int_set:Nn \c@rownum {##1}
        \int_step_inline:nn { \c@colcount }
          {
            \int_set:Nn \c@colnum {####1}
            \tblr_set_cell:nn {#1} {#2}
          }
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_every_cell in different ways
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_every_cell_aux:n #1
  {
    \tl_if_head_is_group:nTF {#1}
      { \tblr_set_every_cell:nn #1 }
      { \tblr_set_every_cell:nn {} {#1} }
  }

%% \SetCell command for multirow and/or multicolumn cells

\NewTableCommand \SetCell [2] []
  {
    \tblr_set_cell:nn { #1 } { #2 }
  }

\tl_new:N \l__tblr_row_span_num_tl
\tl_new:N \l__tblr_col_span_num_tl

\cs_new_protected:Npn \tblr_set_cell:nn #1 #2
  {
    \tl_set:Nn \l__tblr_row_span_num_tl { 1 }
    \tl_set:Nn \l__tblr_col_span_num_tl { 1 }
    \keys_set:nn { tblr-cell-span } { #1 }
```

```
    \keys_set:nn { tblr-cell-spec } { #2 }
    \__tblr_set_span_spec:VV \l__tblr_row_span_num_tl \l__tblr_col_span_num_tl
  }
\cs_generate_variant:Nn \tblr_set_cell:nn { nV }

\cs_new_protected:Npn \tblr_set_cell:nnnn #1 #2 #3 #4
  {
    \group_begin:
    \__tblr_get_childs:nx {#1} { \int_use:N \c@rowcount }
    \clist_set_eq:NN \l_tmpa_clist \l_tblr_childs_clist
    \__tblr_get_childs:nx {#2} { \int_use:N \c@colcount }
    \clist_set_eq:NN \l_tmpb_clist \l_tblr_childs_clist
    \clist_map_inline:Nn \l_tmpa_clist
      {
        \int_set:Nn \c@rownum {##1}
        \clist_map_inline:Nn \l_tmpb_clist
          {
            \int_set:Nn \c@colnum {####1}
            \tblr_set_cell:nn {#3} {#4}
          }
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_cell in different ways
%% Note that #1 is always of the type {<i>}{<j>}
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_cell_aux:nn #1 #2
  {
    \tl_if_head_is_group:nTF {#2}
      { \tblr_set_cell:nnnn #1 #2 }
      { \tblr_set_cell:nnnn #1 {} {#2} }
  }
\cs_generate_variant:Nn \__tblr_set_cell_aux:nn { Vn }

\keys_define:nn { tblr-cell-span }
  {
    r .tl_set:N = \l__tblr_row_span_num_tl,
    c .tl_set:N = \l__tblr_col_span_num_tl,
  }

\keys_define:nn { tblr-cell-spec }
  {
    l .code:n = \__tblr_data_gput:neenn { cell }
                  { \int_use:N \c@rownum } { \int_use:N \c@colnum } { halign } {l},
    c .code:n = \__tblr_data_gput:neenn { cell }
                  { \int_use:N \c@rownum } { \int_use:N \c@colnum } { halign } {c},
    r .code:n = \__tblr_data_gput:neenn { cell }
                  { \int_use:N \c@rownum } { \int_use:N \c@colnum } { halign } {r},
    t .code:n = \__tblr_data_gput:neenn { cell }
                  { \int_use:N \c@rownum } { \int_use:N \c@colnum } { valign} {t},
    p .code:n = \__tblr_data_gput:neenn { cell }
                  { \int_use:N \c@rownum } { \int_use:N \c@colnum } { valign} {t},
    m .code:n = \__tblr_data_gput:neenn { cell }
                  { \int_use:N \c@rownum } { \int_use:N \c@colnum } { valign} {m},
    b .code:n = \__tblr_data_gput:neenn { cell }
                  { \int_use:N \c@rownum } { \int_use:N \c@colnum } { valign} {b},
```

```
    h .code:n = \__tblr_data_gput:neenn { cell }
                  { \int_use:N \c@rownum } { \int_use:N \c@colnum } { valign} {h},
    f .code:n = \__tblr_data_gput:neenn { cell }
                  { \int_use:N \c@rownum } { \int_use:N \c@colnum } { valign} {f},
    wd .code:n = \__tblr_data_gput:neene { cell }
                  { \int_use:N \c@rownum } { \int_use:N \c@colnum } { width } {#1},
    bg .code:n = \__tblr_data_gput:neene { cell }
                  { \int_use:N \c@rownum } { \int_use:N \c@colnum }
                  { background } {#1},
    preto .code:n = \__tblr_cell_preto_text:n {#1},
    appto .code:n = \__tblr_cell_appto_text:n {#1},
    fg .code:n = \__tblr_cell_preto_text:n { \color{#1} },
    font .code:n = \__tblr_cell_preto_text:n { #1 \selectfont },
    unknown .code:n = \__tblr_cell_unknown_key:V \l_keys_key_str,
  }

\tl_new:N \l__tblr_cell_text_tl

\cs_new_protected:Npn \__tblr_cell_preto_text:n #1
  {
    \__tblr_cell_preto_text:een
      { \int_use:N \c@rownum } { \int_use:N \c@colnum } {#1}
  }

\cs_new_protected:Npn \__tblr_cell_preto_text:nnn #1 #2 #3
  {
    \tl_set:Nx \l__tblr_cell_text_tl { \__tblr_text_item:nn { text } { [#1][#2] } }
    \tl_put_left:Nn \l__tblr_cell_text_tl {#3}
    \__tblr_text_gput:nnV { text } { [#1][#2] } \l__tblr_cell_text_tl
  }
\cs_generate_variant:Nn \__tblr_cell_preto_text:nnn { nen, enn, een }

\cs_new_protected:Npn \__tblr_cell_appto_text:n #1
  {
    \__tblr_cell_appto_text:een
      { \int_use:N \c@rownum } { \int_use:N \c@colnum } {#1}
  }

\cs_new_protected:Npn \__tblr_cell_appto_text:nnn #1 #2 #3
  {
    \tl_set:Nx \l__tblr_cell_text_tl { \__tblr_text_item:ne { text } { [#1][#2] } }
    \tl_put_right:Nn \l__tblr_cell_text_tl {#3}
    \__tblr_text_gput:neV { text } { [#1][#2] } \l__tblr_cell_text_tl
  }
\cs_generate_variant:Nn \__tblr_cell_appto_text:nnn { nen, enn, een }

\cs_new_protected:Npn \__tblr_cell_unknown_key:n #1
  {
    \regex_match:NnTF \c__tblr_is_color_key_regex {#1}
      {
        \__tblr_data_gput:neene { cell }
          { \int_use:N \c@rownum } { \int_use:N \c@colnum } { background } {#1}
      }
      {
        \tl_set_rescan:Nnn \l__tblr_v_tl {} {#1}
        \__tblr_data_gput:neene { cell }
```

```
          { \int_use:N \c@rownum } { \int_use:N \c@colnum } { width }
          { \dim_eval:n { \l__tblr_v_tl } }
      }
  }
\cs_generate_variant:Nn \__tblr_cell_unknown_key:n { V }

\cs_new_protected:Npn \__tblr_set_span_spec:nn #1 #2
  {
    \int_compare:nNnT { #1 } > { 1 }
      {
        \__tblr_prop_gput:nnn {table} {rowspan} {true}
        \__tblr_data_gput:neenn { cell }
          { \int_use:N \c@rownum } { \int_use:N \c@colnum } { rowspan } {#1}
      }
    \int_compare:nNnT { #2 } > { 1 }
      {
        \__tblr_prop_gput:nnn {table} {colspan} {true}
        \__tblr_data_gput:neenn { cell }
          { \int_use:N \c@rownum } { \int_use:N \c@colnum } { colspan } {#2}
      }
    \int_step_variable:nnNn
      { \int_use:N \c@rownum } { \int_eval:n { \c@rownum + #1 - 1 } } \l__tblr_i_tl
      {
        \int_step_variable:nnNn
          { \int_use:N \c@colnum } { \int_eval:n { \c@colnum + #2 - 1 } }
          \l__tblr_j_tl
          {
            \bool_lazy_and:nnF
              { \int_compare_p:nNn { \l__tblr_i_tl } = { \c@rownum } }
              { \int_compare_p:nNn { \l__tblr_j_tl } = { \c@colnum } }
              {
                \__tblr_data_gput:neenn { cell }
                  { \l__tblr_i_tl } { \l__tblr_j_tl } { omit } {1}
              }
            \int_compare:nNnF { \l__tblr_i_tl } = { \c@rownum }
              {
                \__tblr_text_gput:nen { hline }
                  { [\l__tblr_i_tl][\l__tblr_j_tl] / omit } {true}
              }
            \int_compare:nNnF { \l__tblr_j_tl } = { \c@colnum }
              {
                \__tblr_text_gput:nee { vline }
                  { [\l__tblr_i_tl][\l__tblr_j_tl] / omit } {true}
              }
          }
      }
  }
\cs_generate_variant:Nn \__tblr_set_span_spec:nn { VV }

%% Legacy \multicolumn and \multirow commands
%% Both of them could be replaced with \SetCell command
%% Note that they don't have cell text as the last arguments

%% If \multicolumn is followed by \multirow,
%% We need to call \tblr_set_cell together
%% in order to omit all hlines inside the span cell.
\tl_new:N \g__tblr_multicolumn_num_tl
```

```
\tl_new:N \g__tblr_multicolumn_spec_tl

%% There maybe p{2em} inside #2 of \multicolumn command
\NewTableCommand \multicolumn [2]
  {
    \tl_gclear:N \g__tblr_multicolumn_num_tl
    \tl_gclear:N \g__tblr_multicolumn_spec_tl
    \tl_map_inline:nn {#2}
      {
        \bool_lazy_and:nnF
          { \tl_if_single_token_p:n {##1} }
          { \token_if_eq_charcode_p:NN ##1 | }
          { \tl_put_right:Nn \g__tblr_multicolumn_spec_tl {,##1} }
      }
    \peek_meaning:NTF \multirow
      { \tl_gset:Nn \g__tblr_multicolumn_num_tl {#1} }
      { \tblr_set_cell:nV { c = #1 } \g__tblr_multicolumn_spec_tl }
  }

\NewTableCommand \multirow [3] [m]
  {
    \tl_if_eq:nnTF {#1} {c}
      { \tl_set:Nn \l_tmpa_tl {, m} }
      {
        \tl_if_eq:nnTF {#1} {t}
          { \tl_set:Nn \l_tmpa_tl {, h} }
          { \tl_if_eq:nnTF {#1} {b}
            { \tl_set:Nn \l_tmpa_tl {, f} }
            { \tl_set:Nn \l_tmpa_tl {, #1} }
          }
      }
    \tl_if_eq:nnF {#3} {*}
      { \tl_if_eq:nnF {#3} {=} { \tl_put_right:Nn \l_tmpa_tl {, wd=#3} } }
    \tl_if_empty:NTF \g__tblr_multicolumn_num_tl
      { \tblr_set_cell:nV { r = #2 } \l_tmpa_tl }
      {
        \tl_put_left:NV \l_tmpa_tl \g__tblr_multicolumn_spec_tl
        \exp_args:Nx \tblr_set_cell:nV
          { c = \g__tblr_multicolumn_num_tl, r = #2 } \l_tmpa_tl
        \tl_gclear:N \g__tblr_multicolumn_num_tl
      }
  }

%%% ------------------------------------------------------------
%%  \section{Set Columns and Rows}
%%% ------------------------------------------------------------

%% \SetColumns command for setting every column in the table
\NewTableCommand \SetColumns [2] []
  {
    \tblr_set_every_column:nn {#1} {#2}
  }

%% We put all code inside a group to avoid affecting other table commands
\cs_new_protected:Npn \tblr_set_every_column:nn #1 #2
  {
    \group_begin:
```

```
    \int_step_inline:nn { \c@colcount }
      {
        \int_set:Nn \c@colnum {##1}
        \tblr_set_column:nn {#1} {#2}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_every_column in different ways
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_every_column_aux:n #1
  {
    \tl_if_head_is_group:nTF {#1}
      { \tblr_set_every_column:nn #1 }
      { \tblr_set_every_column:nn {} {#1} }
  }

%% \SetColumn command for current column or each cells in the column

\NewTableCommand \SetColumn [2] []
  {
    \tblr_set_column:nn {#1} {#2}
  }

\cs_new_protected:Npn \tblr_set_column:nn #1 #2
  {
    \keys_set:nn { tblr-column } {#2}
  }

\cs_new_protected:Npn \tblr_set_column:nnn #1 #2 #3
  {
    \group_begin:
    \__tblr_get_childs:nx {#1} { \int_use:N \c@colcount }
    \clist_map_inline:Nn \l_tblr_childs_clist
      {
        \int_set:Nn \c@colnum {##1}
        \tblr_set_column:nn {#2} {#3}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_column in different ways
%% Note that #1 always includes an outer pair of braces
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_column_aux:nn #1 #2
  {
    \tl_if_head_is_group:nTF {#2}
      { \tblr_set_column:nnn #1 #2 }
      { \tblr_set_column:nnn #1 {} {#2} }
  }
\cs_generate_variant:Nn \__tblr_set_column_aux:nn { Vn }

\keys_define:nn { tblr-column }
  {
    l .code:n = \__tblr_set_key_for_every_column_cell:nnn
                  { \int_use:N \c@colnum } { halign } {l},
```

```
    c .code:n = \__tblr_set_key_for_every_column_cell:nnn
                  { \int_use:N \c@colnum } { halign } {c},
    r .code:n = \__tblr_set_key_for_every_column_cell:nnn
                  { \int_use:N \c@colnum } { halign } {r},
    t .code:n = \__tblr_set_key_for_every_column_cell:nnn
                  { \int_use:N \c@colnum } { valign } {t},
    p .code:n = \__tblr_set_key_for_every_column_cell:nnn
                  { \int_use:N \c@colnum } { valign } {t},
    m .code:n = \__tblr_set_key_for_every_column_cell:nnn
                  { \int_use:N \c@colnum } { valign } {m},
    b .code:n = \__tblr_set_key_for_every_column_cell:nnn
                  { \int_use:N \c@colnum } { valign } {b},
    h .code:n = \__tblr_set_key_for_every_column_cell:nnn
                  { \int_use:N \c@colnum } { valign } {h},
    f .code:n = \__tblr_set_key_for_every_column_cell:nnn
                  { \int_use:N \c@colnum } { valign } {f},
    bg .code:n = \__tblr_set_key_for_every_column_cell:nnn
                  { \int_use:N \c@colnum } { background } {#1},
    fg .code:n = \__tblr_preto_text_for_every_column_cell:n { \color{#1} },
    font .code:n = \__tblr_preto_text_for_every_column_cell:n { #1 \selectfont },
    wd .code:n = \__tblr_data_gput:nene { column }
                  { \int_use:N \c@colnum } { width } { \dim_eval:n {#1} },
    co .code:n = \__tblr_data_gput:nene { column }
                  { \int_use:N \c@colnum } { coefficient } {#1},
    leftsep .code:n = \__tblr_data_gput:nene { column }
                  { \int_use:N \c@colnum } { leftsep } { \dim_eval:n {#1} },
    rightsep .code:n = \__tblr_data_gput:nene { column }
                  { \int_use:N \c@colnum } { rightsep } { \dim_eval:n {#1} },
    colsep .meta:n = { leftsep = #1, rightsep = #1},
    leftsep+ .code:n = \__tblr_data_gadd_dimen_value:nene { column }
                  { \int_use:N \c@colnum } { leftsep } { \dim_eval:n {#1} },
    rightsep+ .code:n = \__tblr_data_gadd_dimen_value:nene { column }
                  { \int_use:N \c@colnum } { rightsep } { \dim_eval:n {#1} },
    colsep+ .meta:n = { leftsep+ = #1, rightsep+ = #1},
    unknown .code:n = \__tblr_column_unknown_key:V \l_keys_key_str,
  }

%% #1: column number; #2: key; #3: value
\cs_new_protected:Npn \__tblr_set_key_for_every_column_cell:nnn #1 #2 #3
  {
    \int_step_inline:nn { \c@rowcount }
      {
        \__tblr_data_gput:neenn { cell } {##1} {#1} {#2} {#3}
      }
  }

\cs_new_protected:Npn \__tblr_preto_text_for_every_column_cell:n #1
  {
    \int_step_inline:nn { \c@rowcount }
      {
        \__tblr_cell_preto_text:nen {##1} { \int_use:N \c@colnum } {#1}
      }
  }

\cs_new_protected:Npn \__tblr_appto_text_for_every_column_cell:n #1
  {
    \int_step_inline:nn { \c@rowcount }
```

```
      {
        \__tblr_cell_appto_text:nen {##1} { \int_use:N \c@colnum } {#1}
      }
  }

\regex_const:Nn \c__tblr_is_number_key_regex { ^[\+\-]? (\d+|\d*\.\d+)$ }

\cs_new_protected:Npn \__tblr_column_unknown_key:n #1
  {
    \regex_match:NnTF \c__tblr_is_number_key_regex {#1}
      {
        \__tblr_data_gput:nene { column }
          { \int_use:N \c@colnum } { coefficient } {#1}
      }
      {
        \regex_match:NnTF \c__tblr_is_color_key_regex {#1}
          {
            \__tblr_set_key_for_every_column_cell:nnn
              { \int_use:N \c@colnum } { background } {#1}
          }
          {
            \tl_set_rescan:Nnn \l__tblr_v_tl {} {#1}
            \__tblr_data_gput:nene { column }
              { \int_use:N \c@colnum } { width } { \dim_eval:n { \l__tblr_v_tl } }
          }
      }
  }
\cs_generate_variant:Nn \__tblr_column_unknown_key:n { V }

%% \SetRows command for setting every row in the table
\NewTableCommand \SetRows [2] []
  {
    \tblr_set_every_row:nn {#1} {#2}
  }

%% We put all code inside a group to avoid affecting other table commands
\cs_new_protected:Npn \tblr_set_every_row:nn #1 #2
  {
    \group_begin:
    \int_step_inline:nn { \c@rowcount }
      {
        \int_set:Nn \c@rownum {##1}
        \tblr_set_row:nn {#1} {#2}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_every_row in different ways
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_every_row_aux:n #1
  {
    \tl_if_head_is_group:nTF {#1}
      { \tblr_set_every_row:nn #1 }
      { \tblr_set_every_row:nn {} {#1} }
  }

%% \SetRow command for current row or each cells in the row
```

```
\NewTableCommand \SetRow [2] []
  {
    \tblr_set_row:nn {#1} {#2}
  }

\cs_new_protected:Npn \tblr_set_row:nn #1 #2
  {
    \keys_set:nn { tblr-row } {#2}
  }

\cs_new_protected:Npn \tblr_set_row:nnn #1 #2 #3
  {
    \group_begin:
    \__tblr_get_childs:nx {#1} { \int_use:N \c@rowcount }
    \clist_map_inline:Nn \l_tblr_childs_clist
      {
        \int_set:Nn \c@rownum {##1}
        \tblr_set_row:nn {#2} {#3}
      }
    \group_end:
  }

%% Check the number of arguments and call \tblr_set_row in different ways
%% Note that #1 always includes an outer pair of braces
%% This function is called when parsing table specifications
\cs_new_protected:Npn \__tblr_set_row_aux:nn #1 #2
  {
    \tl_if_head_is_group:nTF {#2}
      { \tblr_set_row:nnn #1 #2 }
      { \tblr_set_row:nnn #1 {} {#2} }
  }
\cs_generate_variant:Nn \__tblr_set_row_aux:nn { Vn }

\keys_define:nn { tblr-row }
  {
    l .code:n = \__tblr_set_key_for_every_row_cell:nnn
                  { \int_use:N \c@rownum } { halign } {l},
    c .code:n = \__tblr_set_key_for_every_row_cell:nnn
                  { \int_use:N \c@rownum } { halign } {c},
    r .code:n = \__tblr_set_key_for_every_row_cell:nnn
                  { \int_use:N \c@rownum } { halign } {r},
    t .code:n = \__tblr_set_key_for_every_row_cell:nnn
                  { \int_use:N \c@rownum } { valign } {t},
    p .code:n = \__tblr_set_key_for_every_row_cell:nnn
                  { \int_use:N \c@rownum } { valign } {t},
    m .code:n = \__tblr_set_key_for_every_row_cell:nnn
                  { \int_use:N \c@rownum } { valign } {m},
    b .code:n = \__tblr_set_key_for_every_row_cell:nnn
                  { \int_use:N \c@rownum } { valign } {b},
    h .code:n = \__tblr_set_key_for_every_row_cell:nnn
                  { \int_use:N \c@rownum } { valign } {h},
    f .code:n = \__tblr_set_key_for_every_row_cell:nnn
                  { \int_use:N \c@rownum } { valign } {f},
    bg .code:n = \__tblr_set_key_for_every_row_cell:nnn
                  { \int_use:N \c@rownum } { background } {#1},
    fg .code:n = \__tblr_preto_text_for_every_row_cell:n { \color{#1} },
    font .code:n = \__tblr_preto_text_for_every_row_cell:n { #1 \selectfont },
```

```
    ht .code:n = \__tblr_data_gput:nene { row } { \int_use:N \c@rownum }
                  { height } { \dim_eval:n {#1} },
    co .code:n = \__tblr_data_gput:nene { row } { \int_use:N \c@rownum }
                  { coefficient } {#1},
    abovesep .code:n = \__tblr_data_gput:nene { row } { \int_use:N \c@rownum }
                        { abovesep } { \dim_eval:n {#1} },
    belowsep .code:n = \__tblr_data_gput:nene { row } { \int_use:N \c@rownum }
                        { belowsep } { \dim_eval:n {#1} },
    rowsep .meta:n = { abovesep = #1, belowsep = #1},
    abovesep+ .code:n = \__tblr_data_gadd_dimen_value:nene { row }
                  { \int_use:N \c@rownum } { abovesep } { \dim_eval:n {#1} },
    belowsep+ .code:n = \__tblr_data_gadd_dimen_value:nene { row }
                  { \int_use:N \c@rownum } { belowsep } { \dim_eval:n {#1} },
    rowsep+ .meta:n = { abovesep+ = #1, belowsep+ = #1},
    nobreak .code:n = \__tblr_prop_gput:nxx { row }
                  { [\int_eval:n {\c@rownum - 1}] / nobreak } { true },
    unknown .code:n = \__tblr_row_unknown_key:V \l_keys_key_str,
  }

%% #1: row number; #2: key; #3: value
\cs_new_protected:Npn \__tblr_set_key_for_every_row_cell:nnn #1 #2 #3
  {
    \int_step_inline:nn { \c@colcount }
      {
        \__tblr_data_gput:neenn { cell } {#1} {##1} {#2} {#3}
      }
  }

\cs_new_protected:Npn \__tblr_preto_text_for_every_row_cell:n #1
  {
    \int_step_inline:nn { \c@colcount }
      {
        \__tblr_cell_preto_text:enn { \int_use:N \c@rownum } {##1} {#1}
      }
  }

\cs_new_protected:Npn \__tblr_appto_text_for_every_row_cell:n #1
  {
    \int_step_inline:nn { \c@colcount }
      {
        \__tblr_cell_appto_text:enn { \int_use:N \c@rownum } {##1} {#1}
      }
  }

\cs_new_protected:Npn \__tblr_row_unknown_key:n #1
  {
    \regex_match:NnTF \c__tblr_is_number_key_regex {#1}
      {
        \__tblr_data_gput:nene { row } { \int_use:N \c@rownum }
          { coefficient } {#1}
      }
      {
        \regex_match:NnTF \c__tblr_is_color_key_regex {#1}
          {
            \__tblr_set_key_for_every_row_cell:nnn
              { \int_use:N \c@rownum } { background } {#1}
          }
```

```
          {
          \tl_set_rescan:Nnn \l__tblr_v_tl {} {#1}
            \__tblr_data_gput:nene { row } { \int_use:N \c@rownum }
              { height } { \dim_eval:n { \l__tblr_v_tl } }
          }
      }
  }
\cs_generate_variant:Nn \__tblr_row_unknown_key:n { V }

%%% ------------------------------------------------------------
%%  \section{Column Types and Row Types}
%%% ------------------------------------------------------------

%% Some primitive column/row types

\str_const:Nn \c_tblr_primitive_colrow_types_str { Q | < > }
\tl_new:N \g__tblr_expanded_colrow_spec_tl

\exp_args:Nc \NewDocumentCommand { tblr_primitive_column_type_ Q } { O{} }
  {
    \keys_set:nn { tblr-column } { #1 }
    \int_incr:N \c@colnum
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_column_type_ Q } { O{} }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { Q[#1] }
    \__tblr_expand_colrow_spec_next:N
  }

\exp_args:Nc \NewDocumentCommand { tblr_primitive_row_type_ Q } { O{} }
  {
    \keys_set:nn { tblr-row } { #1 }
    \int_incr:N \c@rownum
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_row_type_ Q } { O{} }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { Q[#1] }
    \__tblr_expand_colrow_spec_next:N
  }

\exp_args:Nc \NewDocumentCommand { tblr_primitive_column_type_ | } { O{} }
  {
    \vline [#1]
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_column_type_ | } { O{} }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { |[#1] }
    \__tblr_expand_colrow_spec_next:N
  }

\exp_args:Nc \NewDocumentCommand { tblr_primitive_row_type_ | } { O{} }
  {
    \hline [#1]
```

```
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_row_type_ | } { O{} }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { |[#1] }
    \__tblr_expand_colrow_spec_next:N
  }

\exp_args:Nc \NewDocumentCommand { tblr_primitive_column_type_ > } { O{} m }
  {
    \tl_if_blank:nF {#1}
      {
        \__tblr_data_gput:nene
          { column }
          { \int_use:N \c@colnum } { leftsep }
          { \dim_eval:n {#1} }
      }
    \tl_if_blank:nF {#2}
      {
        \__tblr_preto_text_for_every_column_cell:n {#2}
      }
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_column_type_ > } { O{} m }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { >[#1]{#2} }
    \__tblr_expand_colrow_spec_next:N
  }

\exp_args:Nc \NewDocumentCommand { tblr_primitive_row_type_ > } { O{} m }
  {
    \tl_if_blank:nF {#1}
      {
        \__tblr_data_gput:nene { row } { \int_use:N \c@rownum }
          { abovesep } { \dim_eval:n { #1 } }
      }
    \tl_if_blank:nF {#2}
      {
        \__tblr_preto_text_for_every_row_cell:n {#2}
      }
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_row_type_ > } { O{} m }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { >[#1]{#2} }
    \__tblr_expand_colrow_spec_next:N
  }

\exp_args:Nc \NewDocumentCommand { tblr_primitive_column_type_ < } { O{} m }
  {
    \tl_if_blank:nF {#1}
      {
        \__tblr_data_gput:nene { column }
          { \int_eval:n {\c@colnum - 1} } { rightsep } { \dim_eval:n {#1} }
      }
    \tl_if_blank:nF {#2}
      {
```

```
        \group_begin:
        \int_decr:N \c@colnum
        \__tblr_appto_text_for_every_column_cell:n {#2}
        \group_end:
      }
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_column_type_ < } { O{} m }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { <[#1]{#2} }
    \__tblr_expand_colrow_spec_next:N
  }

\exp_args:Nc \NewDocumentCommand { tblr_primitive_row_type_ < } { O{} m }
  {
    \tl_if_blank:nF {#1}
      {
        \__tblr_data_gput:nene { row } { \int_eval:n {\c@rownum - 1} }
          { belowsep } { \dim_eval:n {#1} }
      }
    \tl_if_blank:nF {#2}
      {
        \group_begin:
        \int_decr:N \c@rownum
        \__tblr_appto_text_for_every_row_cell:n {#2}
        \group_end:
      }
    \__tblr_execute_colrow_spec_next:N
  }
\exp_args:Nc \NewDocumentCommand { tblr_row_type_ < } { O{} m }
  {
    \tl_gput_right:Nn \g__tblr_expanded_colrow_spec_tl { <[#1]{#2} }
    \__tblr_expand_colrow_spec_next:N
  }

%% \NewColumnType/\NewRowType command and predefined column/row types

\str_new:N \g_tblr_used_column_types_str
\str_gset_eq:NN \g_tblr_used_column_types_str \c_tblr_primitive_colrow_types_str

\str_new:N \g_tblr_used_row_types_str
\str_gset_eq:NN \g_tblr_used_row_types_str \c_tblr_primitive_colrow_types_str

\bool_new:N \g__tblr_colrow_spec_expand_stop_bool
\tl_new:N \g__tblr_column_or_row_tl

\msg_new:nnn { tabularray } { used-colrow-type }
  { #1 ~ type ~ name ~ #2 ~ has ~ been ~ used! }

\NewDocumentCommand \NewColumnType { m O{0} o m }
  {
    \tl_set:Nn \g__tblr_column_or_row_tl { column }
    \__tblr_new_column_or_row_type:nnnn {#1} {#2} {#3} {#4}
  }

\NewDocumentCommand \NewRowType { m O{0} o m }
```

```
  {
    \tl_set:Nn \g__tblr_column_or_row_tl { row }
    \__tblr_new_column_or_row_type:nnnn {#1} {#2} {#3} {#4}
  }

\NewDocumentCommand \NewColumnRowType { m O{0} o m }
  {
    \tl_set:Nn \g__tblr_column_or_row_tl { column }
    \__tblr_new_column_or_row_type:nnnn {#1} {#2} {#3} {#4}
    \tl_set:Nn \g__tblr_column_or_row_tl { row }
    \__tblr_new_column_or_row_type:nnnn {#1} {#2} {#3} {#4}
  }

\cs_new_protected:Npn \__tblr_new_column_or_row_type:nnnn #1 #2 #3 #4
  {
    \str_if_in:cnTF { g_tblr_used_ \g__tblr_column_or_row_tl _types_str } {#1}
      {
        \tl_if_eq:NnTF \g__tblr_column_or_row_tl { row }
          { \msg_warning:nnnn { tabularray } { used-colrow-type } { Row } {#1} }
          { \msg_warning:nnnn { tabularray } { used-colrow-type } { Column } {#1} }
        \str_log:c { g_tblr_used_ \g__tblr_column_or_row_tl _types_str }
      }
      {
        \__tblr_make_xparse_arg_spec:nnN {#2} {#3} \l__tblr_a_tl
        \exp_args:NcV \NewDocumentCommand
          { tblr_ \g__tblr_column_or_row_tl _type_ #1 } \l__tblr_a_tl
          {
            \bool_gset_false:N \g__tblr_colrow_spec_expand_stop_bool
            \tl_gput_right:Nf \g__tblr_expanded_colrow_spec_tl {#4}
            \__tblr_expand_colrow_spec_next:N
          }
        \str_gput_right:cn
          { g_tblr_used_ \g__tblr_column_or_row_tl _types_str } {#1}
      }
  }

\NewColumnRowType { l } { Q[l] }
\NewColumnRowType { c } { Q[c] }
\NewColumnRowType { r } { Q[r] }

\NewColumnType { t } [1] { Q[t,wd=#1] }
\NewColumnType { p } [1] { Q[p,wd=#1] }
\NewColumnType { m } [1] { Q[m,wd=#1] }
\NewColumnType { b } [1] { Q[b,wd=#1] }
\NewColumnType { h } [1] { Q[h,wd=#1] }
\NewColumnType { f } [1] { Q[f,wd=#1] }

\NewRowType { t } [1] { Q[t,ht=#1] }
\NewRowType { p } [1] { Q[p,ht=#1] }
\NewRowType { m } [1] { Q[m,ht=#1] }
\NewRowType { b } [1] { Q[b,ht=#1] }
\NewRowType { h } [1] { Q[h,ht=#1] }
\NewRowType { f } [1] { Q[f,ht=#1] }

\NewColumnRowType { X } [1][] { Q[co=1,#1] }
```

```
\NewColumnRowType { ! } [1] { |[text={#1}] }
\NewColumnRowType { @ } [1] { <[Opt]{} |[text={#1}] >[Opt]{} }
\NewColumnRowType { * } [2] { \prg_replicate:nn {#1} {#2} }


\cs_new_protected:Npn \__tblr_parse_colrow_spec:nn #1 #2
  {
    \tl_gset:Nn \g__tblr_column_or_row_tl {#1}
    \tl_gset:Nn \g__tblr_expanded_colrow_spec_tl {#2}
    \__tblr_expand_colrow_spec:N \g__tblr_expanded_colrow_spec_tl
    \__tblr_execute_colrow_spec:N \g__tblr_expanded_colrow_spec_tl
  }

%% Expand defined column/row types

\cs_new_protected:Npn \__tblr_expand_colrow_spec:N #1
  {
    \bool_do_until:Nn \g__tblr_colrow_spec_expand_stop_bool
      {
        \LogTblrTracing { colspec, rowspec }
        \bool_gset_true:N \g__tblr_colrow_spec_expand_stop_bool
        \tl_set_eq:NN \l_tmpa_tl #1
        \tl_gclear:N #1
        \exp_last_unbraced:NV
          \__tblr_expand_colrow_spec_next:N \l_tmpa_tl \scan_stop:
      }
  }

\msg_new:nnn { tabularray } { unexpandable-colrow-type }
  { Unexpandable ~ command ~ #2 inside ~ #1 ~ type! }

\msg_new:nnn { tabularray } { unknown-colrow-type }
  { Unknown ~ #1 ~ type ~ #2! }

\cs_new_protected:Npn \__tblr_expand_colrow_spec_next:N #1
  {
    \token_if_eq_catcode:NNTF #1 \scan_stop:
      {
        \token_if_eq_meaning:NNF #1 \scan_stop:
          {
            \msg_error:nnVn { tabularray } { unexpandable-colrow-type }
              \g__tblr_column_or_row_tl {#1}
          }
      }
      {
        \str_if_in:cnTF { g_tblr_used_ \g__tblr_column_or_row_tl _types_str } {#1}
          { \cs:w tblr_ \g__tblr_column_or_row_tl _type_  #1 \cs_end: }
          {
            \msg_error:nnVn { tabularray } { unknown-colrow-type }
              \g__tblr_column_or_row_tl {#1}
            \str_log:c { g_tblr_used_ \g__tblr_column_or_row_tl _types_str }
          }
      }
  }

%% Execute primitive column/row types
```

```
\cs_new_protected:Npn \__tblr_execute_colrow_spec:N #1
  {
    \tl_if_eq:NnTF \g__tblr_column_or_row_tl { row }
      { \int_set:Nn \c@rownum {1} }
      { \int_set:Nn \c@colnum {1} }
    \exp_last_unbraced:NV \__tblr_execute_colrow_spec_next:N #1 \scan_stop:
  }

\cs_new_protected:Npn \__tblr_execute_colrow_spec_next:N #1
  {
    \token_if_eq_meaning:NNF #1 \scan_stop:
      { \cs:w tblr_primitive_ \g__tblr_column_or_row_tl _type_  #1 \cs_end: }
  }

%%% ------------------------------------------------------------
%%  \section{Tabularray Environments}
%%% ------------------------------------------------------------

\tl_new:N \l__tblr_env_name_tl
\bool_new:N \l__tblr_math_mode_bool

\NewDocumentEnvironment { tblr } { O{c} m +b }
  {
    \tl_set:Nn \l__tblr_env_name_tl { tblr }
    \mode_if_math:TF
      { \bool_set_true:N \l__tblr_math_mode_bool }
      { \bool_set_false:N \l__tblr_math_mode_bool }
    \buildtblr {#1} {#2} {#3}
  } { }

%% Read, split and build the table

\cs_new_protected:Npn \buildtblr #1 #2 #3
  {
    \mode_leave_vertical:
    \int_gincr:N \g_tblr_level_int
    \__tblr_clear_prop_lists:
    \__tblr_clear_text_lists:
    \__tblr_enable_table_commands:
    \__tblr_split_table:n { #3 }
    \LogTblrTracing { command }
    \bool_if:NT \g__tblr_use_intarray_bool { \__tblr_initial_table_data: }
    \__tblr_initial_table_spec:
    \LogTblrTracing { table }
    \__tblr_parse_table_spec:n { #2 }
    \__tblr_execute_table_commands:
    \__tblr_disable_table_commands:
    \__tblr_calc_cell_and_line_sizes:
    \__tblr_build_whole:n { #1 }
    \int_gdecr:N \g_tblr_level_int
  }

%% Insert and remove braces for nesting environments inside cells
%% These make line split and cell split workable
%% We need to replace N times for N level nestings
\regex_const:Nn \c__tblr_insert_braces_regex
```

```
  {
    \c{begin} \cB\{ (\c[^BE].*) \cE\} (.*?) \c{end} \cB\{ (\c[^BE].*) \cE\}
  }
\tl_const:Nn \c__tblr_insert_braces_tl
  {
    \c{begin} \cB\{ \cB\{ \1 \cE\} \2 \c{end} \cE\} \cB\{ \3 \cE\}
  }
\regex_const:Nn \c__tblr_remove_braces_regex
  {
    \c{begin} \cB\{ \cB\{ (.*?) \c{end} \cE\}
  }
\tl_const:Nn \c__tblr_remove_braces_tl
  {
    \c{begin} \cB\{ \1 \c{end}
  }
\cs_new_protected:Npn \__tblr_insert_braces:N #1
  {
    \regex_replace_all:NVN \c__tblr_insert_braces_regex \c__tblr_insert_braces_tl #1
    \regex_replace_all:NVN \c__tblr_insert_braces_regex \c__tblr_insert_braces_tl #1
  }
\cs_new_protected:Npn \__tblr_remove_braces:N #1
  {
    \regex_replace_all:NVN \c__tblr_remove_braces_regex \c__tblr_remove_braces_tl #1
    \regex_replace_all:NVN \c__tblr_remove_braces_regex \c__tblr_remove_braces_tl #1
  }

%% Split table content to cells and store them
%% #1: table content

\seq_new:N \l_tblr_lines_seq

\cs_new_protected:Npn \__tblr_split_table:n #1
  {
    \int_zero:N \c@rowcount
    \int_zero:N \c@colcount
    \__tblr_split_table_to_lines:nN { #1 } \l_tblr_lines_seq
    \__tblr_split_lines_to_cells:N \l_tblr_lines_seq
  }

%% Split table content to a sequence of lines
%% #1: table content, #2: resulting sequence of lines
\cs_new_protected:Npn \__tblr_split_table_to_lines:nN #1 #2
  {
    \tl_set:Nn \l_tmpa_tl { #1 }
    \__tblr_insert_braces:N \l_tmpa_tl
    \seq_set_split:NnV \l_tmpa_seq { \\ } \l_tmpa_tl
    \seq_clear:N #2
    \seq_map_inline:Nn \l_tmpa_seq
      {
        \tl_if_head_eq_meaning:nNTF {##1} *
          {
            \tl_set:Nn \l__tblr_b_tl { \SetRow{nobreak} }
            \tl_set:Nx \l__tblr_c_tl { \tl_tail:n {##1} }
            \tl_trim_spaces:N \l__tblr_c_tl %% Ignore spaces between * and [dimen]
            \tl_log:N \l__tblr_c_tl
            \tl_if_head_eq_meaning:VNT \l__tblr_c_tl [
              {
```

```
                \tl_put_right:Nn \l__tblr_b_tl { \RowBefore@AddBelowSep }
              }
            \tl_put_right:NV \l__tblr_b_tl \l__tblr_c_tl
            \seq_put_right:NV #2 \l__tblr_b_tl
          }
          {
            \tl_if_head_eq_meaning:nNTF { ##1 } [
              { \seq_put_right:Nn #2 { \RowBefore@AddBelowSep ##1 } }
              { \seq_put_right:Nn #2 { ##1 } }
          }
      }
    \int_set:Nn \c@rowcount { \seq_count:N #2 }
  }

%% Treat \\[dimen] command
\NewTableCommand \RowBefore@AddBelowSep [1] []
  {
    \IfValueT { #1 }
      {
        \__tblr_data_gadd_dimen_value:nene { row }
          { \int_eval:n {\c@rownum - 1} } { belowsep } {#1}
      }
  }

%% Split table lines to cells and store them
%% #1: sequence of lines
\cs_new_protected:Npn \__tblr_split_lines_to_cells:N #1
  {
    \seq_map_indexed_function:NN #1 \__tblr_split_one_line:nn
    \LogTblrTracing { text }
  }

%% Split one line into cells and store them
%% #1: row number, #2 the line text
\cs_new_protected:Npn \__tblr_split_one_line:nn #1 #2
  {
    \seq_set_split:Nnn \l_tmpa_seq { & } { #2 }
    \int_set:Nn \c@rownum {#1}
    \int_zero:N \c@colnum
    \seq_map_inline:Nn \l_tmpa_seq
      {
        \tl_set:Nn \l_tmpa_tl { ##1 }
        \__tblr_remove_braces:N \l_tmpa_tl
        \int_incr:N \c@colnum
        \__tblr_extract_table_commands:N \l_tmpa_tl
        \__tblr_text_gput:neV { text } { [#1][\int_use:N \c@colnum] } \l_tmpa_tl
        \__tblr_add_multicolumn_empty_cell:
      }
    %% Decrease row count by 1 if the last row has only one empty cell text
    %% We need to do it here since the > or < column type may add text to cells
    \bool_lazy_and:nnTF
      { \int_compare_p:nNn {\c@colnum} = {1} }
      { \tl_if_empty_p:N \l_tmpa_tl }
      { \int_decr:N \c@rowcount }
      {
        \__tblr_prop_gput:nnx
          {row} { [#1] / cell-number } { \int_use:N \c@colnum }
```

```
        \int_compare:nT { \c@colnum > \c@colcount }
          {
            \int_set_eq:NN \c@colcount \c@colnum
          }
      }
  }

%% Add empty cells after the \multicolumn span cell
\cs_new_protected:Npn \__tblr_add_multicolumn_empty_cell:
  {
    \int_step_inline:nn { \l__multicolumn_cell_number_int - 1 }
      {
        \int_incr:N \c@colnum
        \__tblr_text_gput:nen { text }
          { [\int_use:N \c@rownum][\int_use:N \c@colnum] } { }
      }
  }

%%% ------------------------------------------------------------
%%  \section{Extract Table Commands from Cell Text}
%%% ------------------------------------------------------------

%% Extract table commands defined with \NewTableCommand from cell text

\clist_gset:Nn \g__tblr_table_commands_unbrace_next_clist {\multirow, \multicolumn}
\bool_new:N \l__tblr_table_command_unbrace_next_bool
\int_new:N \l__multicolumn_cell_number_int
\tl_new:N \l__tblr_saved_table_commands_before_cell_text_tl
\tl_new:N \l__tblr_saved_cell_text_after_table_commands_tl

\cs_new_protected:Npn \__tblr_extract_table_commands:N #1
  {
    \tl_clear:N \l__tblr_saved_table_commands_before_cell_text_tl
    \tl_clear:N \l__tblr_saved_cell_text_after_table_commands_tl
    \int_set:Nn \l__multicolumn_cell_number_int {1}
    \exp_last_unbraced:NV \__tblr_extract_table_commands_next:w #1 \scan_stop:
    \tl_if_empty:NF \l__tblr_saved_table_commands_before_cell_text_tl
      {
        \__tblr_prop_gput:nxV { command }
          {[\int_use:N \c@rownum][\int_use:N \c@colnum]}
          \l__tblr_saved_table_commands_before_cell_text_tl
      }
    \tl_set_eq:NN #1 \l__tblr_saved_cell_text_after_table_commands_tl
  }

%% #1 maybe a single token or multiple tokens given in braces
\cs_new_protected:Npn \__tblr_extract_table_commands_next:w #1
  {
    \clist_if_in:NnTF \g__tblr_table_commands_clist { #1 }
      {
        \clist_if_in:NnTF \g__tblr_table_commands_unbrace_next_clist { #1 }
          { \bool_set_true:N \l__tblr_table_command_unbrace_next_bool }
          { \bool_set_false:N \l__tblr_table_command_unbrace_next_bool }
        \token_if_eq_meaning:NNTF #1 \multicolumn
          { \__tblr_extract_multicolumn_command:Nn #1 }
          { \__tblr_extract_one_table_command:N #1 }
```

```
      }
      {
        \tl_if_single_token:nTF {#1}
          {
            \token_if_eq_meaning:NNF #1 \scan_stop:
              { \__tblr_save_real_cell_text:w #1 }
          }
          { \__tblr_save_real_cell_text:w {#1} }
      }
  }

\cs_new_protected:Npn \__tblr_extract_multicolumn_command:Nn #1 #2
  {
    \int_set:Nn \l__multicolumn_cell_number_int {#2}
    \__tblr_extract_one_table_command:N #1 {#2}
  }

\cs_new_protected:Npn \__tblr_extract_one_table_command:N #1
  {
    \int_set:Nn \l__tblr_a_int
      { \cs:w g__tblr_table_cmd_ \cs_to_str:N #1 _arg_numb_tl \cs_end: }
    \tl_put_right:Nn \l__tblr_saved_table_commands_before_cell_text_tl {#1}
    \int_compare:nNnTF {\l__tblr_a_int} < {0}
      {
        \int_set:Nn \l__tblr_a_int { \int_abs:n {\l__tblr_a_int} - 1 }
        \peek_charcode:NTF [
          { \__tblr_extract_table_command_arg_o:w }
          { \__tblr_extract_table_command_arg_next: }
      }
      { \__tblr_extract_table_command_arg_next: }
  }

\cs_new_protected:Npn \__tblr_extract_table_command_arg_o:w [#1]
  {
    \tl_put_right:Nn \l__tblr_saved_table_commands_before_cell_text_tl { [#1] }
    \__tblr_extract_table_command_arg_next:
  }

\cs_new_protected:Npn \__tblr_extract_table_command_arg_m:n #1
  {
    \tl_put_right:Nn \l__tblr_saved_table_commands_before_cell_text_tl { {#1} }
    \__tblr_extract_table_command_arg_next:
  }

\cs_new_protected:Npn \__tblr_extract_table_command_arg_next:
  {
    \int_compare:nNnTF {\l__tblr_a_int} > {0}
      {
        \int_decr:N \l__tblr_a_int
        \__tblr_extract_table_command_arg_m:n
      }
      {
        \bool_if:NTF \l__tblr_table_command_unbrace_next_bool
          { \__tblr_last_unbraced:Nn \__tblr_extract_table_commands_next:w }
          { \__tblr_extract_table_commands_next:w }
      }
  }
```

```
\cs_new_protected:Npn \__tblr_last_unbraced:Nn #1 #2 { #1 #2 }

%% The outermost set of braces of cell text #1 will be removed
\cs_new_protected:Npn \__tblr_save_real_cell_text:w #1 \scan_stop:
  {
    \tl_set:Nn \l__tblr_saved_cell_text_after_table_commands_tl {#1}
  }

%%% ------------------------------------------------------------
%%  \section{Initial Table Specifications}
%%% ------------------------------------------------------------

\prop_gset_from_keyval:Nn \g__tblr_default_tblr_table_prop
  {
    stretch = 1,
    rulesep = 2pt,
  }

\prop_gset_from_keyval:Nn \g__tblr_default_tblr_rows_prop
  {
    abovesep = 2pt,
    belowsep = 2pt,
    @row-height = 0pt,
    @row-head = 0pt,
    @row-foot = 0pt,
    @row-upper = 0pt,
    @row-lower = 0pt,
  }

\prop_gset_from_keyval:Nn \g__tblr_default_tblr_columns_prop
  {
    leftsep = 6pt,
    rightsep = 6pt,
    width = -1pt, % column width unset
    coefficient = 0, % column coefficient unset
    @col-width = 0pt,
  }

\prop_gset_from_keyval:Nn \g__tblr_default_tblr_cells_prop
  {
    halign = l,
    valign = t,
    width = -1pt, % cell width unset
    rowspan = 1,
    colspan = 1,
    omit = 0,
  }

\prop_gset_from_keyval:Nn \g__tblr_default_tblr_hlines_prop
  {
    @hline-count = 0,
  }

\prop_gset_from_keyval:Nn \g__tblr_default_tblr_vlines_prop
  {
    @vline-count = 0,
```

```
  }

\cs_new_protected:Npn \__tblr_initial_table_spec:
  {
    \prop_map_inline:cn { g__tblr_default_ \l__tblr_env_name_tl _table_prop }
      {
        \__tblr_prop_gput:nxn { table } { ##1 } {##2}
      }
    \int_step_variable:nNn { \c@rowcount } \l__tblr_i_tl
      {
        \prop_map_inline:cn { g__tblr_default_ \l__tblr_env_name_tl _rows_prop }
          {
            \__tblr_data_gput:nVnn { row } \l__tblr_i_tl {##1} {##2}
          }
        \prop_map_inline:cn { g__tblr_default_ \l__tblr_env_name_tl _hlines_prop }
          {
            \__tblr_text_gput:nen { hline } { [\l__tblr_i_tl] / ##1 } {##2}
          }
        \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
          {
            \prop_map_inline:cn
              { g__tblr_default_ \l__tblr_env_name_tl _cells_prop }
              {
                \__tblr_data_gput:neeen { cell }
                  { \l__tblr_i_tl } { \l__tblr_j_tl } {##1} {##2}
              }
          }
      }
    \prop_map_inline:cn { g__tblr_default_ \l__tblr_env_name_tl _hlines_prop }
      {
        \__tblr_text_gput:nen { hline }
          { [\int_eval:n { \c@rowcount + 1}] / ##1 } {##2}
      }
    \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
      {
        \prop_map_inline:cn { g__tblr_default_ \l__tblr_env_name_tl _columns_prop }
          {
            \__tblr_data_gput:nenn { column } { \l__tblr_j_tl } {##1} {##2}
          }
        \prop_map_inline:cn { g__tblr_default_ \l__tblr_env_name_tl _vlines_prop }
          {
            \__tblr_text_gput:nen { vline } { [\l__tblr_j_tl] / ##1 } {##2}
          }
      }
    \prop_map_inline:cn { g__tblr_default_ \l__tblr_env_name_tl _vlines_prop }
      {
        \__tblr_text_gput:nen { vline }
          { [\int_eval:n { \c@colcount + 1}] / ##1 } {##2}
      }
    \keys_set:nv { tblr } { l__tblr_default_ \l__tblr_env_name_tl _tl }
  }

\tl_new:N \l__tblr_default_tblr_tl

%% #1: env name; #2: options
\NewDocumentCommand \SetTabularrayDefault { O{tblr} m }
  {
```

```
    \tl_put_right:cn { l__tblr_default_ #1 _tl } { , #2 }
  }
\cs_new_eq:NN \SetTblrDefault \SetTabularrayDefault

%%% ------------------------------------------------------------
%%  \section{Parse Table Specifications}
%%% ------------------------------------------------------------

\clist_new:N \g__tblr_table_known_keys_clist
\clist_gset:Nn \g__tblr_table_known_keys_clist
  {
    long, colspec, rowspec, width, hspan, stretch,
    column, row, cell, vline, hline, columns, rows, cells, vlines, hlines,
    leftsep, rightsep, colsep, abovesep, belowsep, rowsep, rulesep,
  }

\bool_new:N \l__tblr_long_table_bool

\keys_define:nn { tblr }
  {
    long .bool_set:N = \l__tblr_long_table_bool,
    colspec .code:n = \__tblr_parse_colrow_spec:nn { column } {#1},
    rowspec .code:n = \__tblr_parse_colrow_spec:nn { row } {#1},
    width .code:n = \__tblr_keys_gput:nx { width } { \dim_eval:n {#1} },
    hspan .code:n = \__tblr_keys_gput:nn { hspan } {#1},
    stretch .code:n = \__tblr_keys_gput:nn { stretch } {#1},
    columns .code:n = \__tblr_set_every_column_aux:n {#1},
    rows    .code:n = \__tblr_set_every_row_aux:n {#1},
    cells   .code:n = \__tblr_set_every_cell_aux:n {#1},
    hlines  .code:n = \__tblr_set_every_hline_aux:n {#1},
    vlines  .code:n = \__tblr_set_every_vline_aux:n {#1},
    leftsep .code:n = \tblr_set_every_column:nn { } { leftsep = #1 },
    rightsep .code:n = \tblr_set_every_column:nn { } { rightsep = #1 },
    colsep .meta:n = { leftsep = #1, rightsep = #1 },
    abovesep .code:n = \tblr_set_every_row:nn { } { abovesep = #1 },
    belowsep .code:n = \tblr_set_every_row:nn { } { belowsep = #1 },
    rowsep .meta:n = { abovesep = #1, belowsep = #1 },
    rulesep .code:n = \__tblr_keys_gput:nn { rulesep } {#1},
    unknown .code:n = \__tblr_table_special_key:Vn \l_keys_key_str {#1},
  }

\regex_const:Nn \c__tblr_split_key_name_regex { ^ ( [a-z] + ) ( . * ) }

\cs_new_protected:Npn \__tblr_table_special_key:nn #1 #2
  {
    \regex_extract_once:NnNT \c__tblr_split_key_name_regex {#1} \l_tmpa_seq
      {
        \tl_set:Nx \l__tblr_a_tl { \seq_item:Nn \l_tmpa_seq {2} }
        \tl_set_rescan:Nnx \l__tblr_b_tl {} { \seq_item:Nn \l_tmpa_seq {3} }
        \cs:w __tblr_set_ \l__tblr_a_tl _aux:Vn \cs_end: \l__tblr_b_tl {#2}
      }
  }
\cs_generate_variant:Nn \__tblr_table_special_key:nn { Vn }

%% If the first key name is known, treat #1 is the table spec;
%% otherwise, treat #1 as colspec.
```

```
\regex_const:Nn \c__tblr_first_key_name_regex { ^ \s * ( [A-Za-z\-] + ) }

\cs_new_protected:Npn \__tblr_parse_table_spec:n #1
  {
    \regex_extract_once:NnNTF \c__tblr_first_key_name_regex {#1} \l_tmpa_seq
      {
        \clist_if_in:NxTF \g__tblr_table_known_keys_clist
          { \seq_item:Nn \l_tmpa_seq {2} }
          { \keys_set:nn { tblr } {#1} }
          { \__tblr_parse_colrow_spec:nn { column } {#1} }
      }
      { \__tblr_parse_colrow_spec:nn { column } {#1} }
  }

\cs_new_protected:Npn  \__tblr_keys_gput:nn #1 #2
  {
    \__tblr_prop_gput:nnn { table } {#1} {#2}
  }
\cs_generate_variant:Nn \__tblr_keys_gput:nn { nx }

%%% ------------------------------------------------------------
%%  \section{Typeset and Calculate Sizes}
%%% ------------------------------------------------------------

%% Calculate the width and height for every cell and border

\cs_new_protected:Npn \__tblr_calc_cell_and_line_sizes:
  {
    \__tblr_make_strut_box:
    \__tblr_calculate_line_sizes:
    \__tblr_calculate_cell_sizes:
    \LogTblrTracing { cell, row, column, hline, vline }
    \__tblr_compute_extendable_column_width:
    \__tblr_adjust_sizes_for_span_cells:
  }

%% make strut box from stretch option of the table

\box_new:N \l__tblr_strut_ht_box
\box_new:N \l__tblr_strut_dp_box

\cs_new_protected:Npn \__tblr_make_strut_box:
  {
    \tl_set:Nx \l__tblr_s_tl { \__tblr_prop_item:ne { table } { stretch } }
    \hbox_set:Nn \l__tblr_strut_ht_box
      { \vrule height \l__tblr_s_tl \box_ht:N \strutbox width ~ 0pt }
    \hbox_set:Nn \l__tblr_strut_dp_box
      { \vrule depth \l__tblr_s_tl \box_dp:N \strutbox width ~ 0pt }
  }

%% Calculate the thickness for every hline and vline
\cs_new_protected:Npn \__tblr_calculate_line_sizes:
  {
    %% We need these two counters in executing hline and vline commands
    \int_zero:N \c@rownum
```

```
    \int_zero:N \c@colnum
    \int_step_inline:nn { \c@rowcount + 1 }
      {
        \int_incr:N \c@rownum
        \int_zero:N \c@colnum
        \int_step_inline:nn { \c@colcount + 1 }
          {
            \int_incr:N \c@colnum
            \int_compare:nNnT { ##1 } < { \c@rowcount + 1 }
              {
                \__tblr_measure_and_update_vline_size:nn { ##1 } { ####1 }
              }
            \int_compare:nNnT { ####1 } < { \c@colcount + 1 }
              {
                \__tblr_measure_and_update_hline_size:nn { ##1 } { ####1 }
              }
          }
      }
  }

%% Measure and update thickness of the vline
%% #1: row number, #2 column number
\cs_new_protected:Npn \__tblr_measure_and_update_vline_size:nn #1 #2
  {
    \dim_zero:N \l__tblr_w_dim
    \tl_set:Nx \l__tblr_n_tl
      { \__tblr_text_item:ne { vline } { [#2] / @vline-count } }
    \int_compare:nNnT { \l__tblr_n_tl } > {0}
      {
        \tl_set:Nx \l__tblr_s_tl
          { \__tblr_prop_item:ne { table } { rulesep } }
        \int_step_inline:nn { \l__tblr_n_tl }
          {
            \vbox_set_to_ht:Nnn \l__tblr_b_box {1pt}
              {
                \__tblr_get_vline_segment_child:nnnnn
                  {#1} {#2} {##1} {1pt} {1pt}
              }
            \tl_set:Nx \l__tblr_w_tl { \dim_eval:n { \box_wd:N \l__tblr_b_box } }
            \__tblr_text_gput_if_larger:nee { vline }
              { [#2](##1) / @vline-width } { \l__tblr_w_tl }
            \dim_add:Nn \l__tblr_w_dim { \l__tblr_w_tl }
            \dim_add:Nn \l__tblr_w_dim { \l__tblr_s_tl }
          }
        \dim_add:Nn \l__tblr_w_dim { - \l__tblr_s_tl }
      }
    \__tblr_text_gput_if_larger:nee { vline }
      { [#2]/ @vline-width } { \dim_use:N \l__tblr_w_dim }
  }

%% Get text of a vline segment
%% #1: row number, #2: column number; #3: index number; #4: height; #5: depth
%% We put all code inside a group to avoid conflicts of local variables
\cs_new_protected:Npn \__tblr_get_vline_segment_child:nnnnn #1 #2 #3 #4 #5
  {
    \group_begin:
    \tl_set:Nx \l__tblr_w_tl
      { \__tblr_text_item:ne { vline } { [#1][#2](#3) / wd } }
```

```
    \tl_if_empty:NF \l__tblr_w_tl { \dim_set:Nn \rulewidth { \l__tblr_w_tl } }
    \tl_set:Nx \l__tblr_d_tl
      { \__tblr_text_item:ne { vline } { [#1][#2](#3) / @dash } }
    \tl_set:Nx \l__tblr_a_tl { \tl_head:N \l__tblr_d_tl }
    \tl_set:Nx \l__tblr_b_tl { \tl_tail:N \l__tblr_d_tl }
    \exp_args:NV \tl_if_eq:NNTF \l__tblr_a_tl \@tblr@dash
      {
        \__tblr_get_vline_dash_style:N \l__tblr_b_tl
        \xleaders \l__tblr_b_tl \vfil
      }
      {
        \hbox_set:Nn \l__tblr_d_box { \l__tblr_b_tl }
        \box_set_ht:Nn \l__tblr_d_box {#4}
        \box_set_dp:Nn \l__tblr_d_box {#5}
        \box_use:N \l__tblr_d_box
      }
    \group_end:
  }
\cs_generate_variant:Nn \__tblr_get_vline_segment_child:nnnnn { nnnxx }

%% Measure and update thickness of the hline
%% #1: row number, #2 column number
\cs_new_protected:Npn \__tblr_measure_and_update_hline_size:nn #1 #2
  {
    \dim_zero:N \l__tblr_h_dim
    \tl_set:Nx \l__tblr_n_tl
      { \__tblr_text_item:ne { hline } { [#1] / @hline-count } }
    \int_compare:nNnT { \l__tblr_n_tl } > {0}
      {
        \tl_set:Nx \l__tblr_s_tl
          { \__tblr_prop_item:ne { table } { rulesep } }
        \int_step_inline:nn { \l__tblr_n_tl }
          {
            \hbox_set_to_wd:Nnn \l__tblr_b_box {1pt}
              { \__tblr_get_hline_segment_child:nnn {#1} {#2} {##1} }
            \tl_set:Nx \l__tblr_h_tl
              {
                \dim_eval:n
                  { \box_ht:N \l__tblr_b_box + \box_dp:N \l__tblr_b_box }
              }
            \__tblr_text_gput_if_larger:nee { hline }
              { [#1](##1) / @hline-height } { \l__tblr_h_tl }
            \dim_add:Nn \l__tblr_h_dim { \l__tblr_h_tl }
            \dim_add:Nn \l__tblr_h_dim { \l__tblr_s_tl }
          }
        \dim_add:Nn \l__tblr_h_dim { - \l__tblr_s_tl }
      }
    \__tblr_text_gput_if_larger:nee { hline }
      { [#1] / @hline-height } { \dim_use:N \l__tblr_h_dim }
  }

%% Get text of a hline segment
%% #1: row number, #2: column number; #3: index number
\cs_new_protected:Npn \__tblr_get_hline_segment_child:nnn #1 #2 #3
  {
    \group_begin:
    \tl_set:Nx \l__tblr_w_tl
      { \__tblr_text_item:ne { hline } { [#1][#2](#3) / wd } }
```

```
    \tl_if_empty:NF \l__tblr_w_tl { \dim_set:Nn \rulewidth { \l__tblr_w_tl } }
    \tl_set:Nx \l__tblr_d_tl
      { \__tblr_text_item:ne { hline } { [#1][#2](#3) / @dash } }
    \tl_set:Nx \l__tblr_a_tl { \tl_head:N \l__tblr_d_tl }
    \tl_set:Nx \l__tblr_b_tl { \tl_tail:N \l__tblr_d_tl }
    \exp_args:NV \tl_if_eq:NNTF \l__tblr_a_tl \@tblr@dash
      {
        \__tblr_get_hline_dash_style:N \l__tblr_b_tl
        \xleaders \l__tblr_b_tl \hfil
      }
      { \l__tblr_b_tl \hfil }
    \group_end:
  }

%% current cell alignments
\tl_new:N \g__tblr_cell_halign_tl
\tl_new:N \g__tblr_cell_valign_tl
\tl_new:N \g__tblr_cell_middle_tl

\tl_const:Nn \c__tblr_valign_h_tl { h }
\tl_const:Nn \c__tblr_valign_m_tl { m }
\tl_const:Nn \c__tblr_valign_f_tl { f }
\tl_const:Nn \c__tblr_valign_t_tl { t }
\tl_const:Nn \c__tblr_valign_b_tl { b }

\tl_const:Nn \c__tblr_middle_t_tl { t }
\tl_const:Nn \c__tblr_middle_m_tl { m }
\tl_const:Nn \c__tblr_middle_b_tl { b }

%% #1: row number; #2: column number
\cs_new_protected:Npn \__tblr_get_cell_alignments:nn #1 #2
  {
    \group_begin:
    \tl_gset:Nx \g__tblr_cell_halign_tl
      { \__tblr_data_item:neen { cell } {#1} {#2} { halign } }
    \tl_set:Nx \l__tblr_v_tl
      { \__tblr_data_item:neen { cell } {#1} {#2} { valign } }
    \tl_case:NnF \l__tblr_v_tl
      {
        \c__tblr_valign_t_tl
          {
            \tl_gset:Nn \g__tblr_cell_valign_tl {m}
            \tl_gset:Nn \g__tblr_cell_middle_tl {t}
          }
        \c__tblr_valign_m_tl
          {
            \tl_gset:Nn \g__tblr_cell_valign_tl {m}
            \tl_gset:Nn \g__tblr_cell_middle_tl {m}
          }
        \c__tblr_valign_b_tl
          {
            \tl_gset:Nn \g__tblr_cell_valign_tl {m}
            \tl_gset:Nn \g__tblr_cell_middle_tl {b}
          }
      }
      {
        \tl_gset_eq:NN \g__tblr_cell_valign_tl \l__tblr_v_tl
```

```
        \tl_gclear:N \g__tblr_cell_middle_tl
      }
    \group_end:
  }

%% current cell dimensions
\dim_new:N \g__tblr_cell_wd_dim
\dim_new:N \g__tblr_cell_ht_dim
\dim_new:N \g__tblr_cell_head_dim
\dim_new:N \g__tblr_cell_foot_dim

%% Calculate the width and height for every cell
\cs_new_protected:Npn \__tblr_calculate_cell_sizes:
  {
    %% You can use these two counters in cell text
    \int_zero:N \c@rownum
    \int_zero:N \c@colnum
    \int_step_inline:nn { \c@rowcount }
      {
        \int_incr:N \c@rownum
        \int_zero:N \c@colnum
        \tl_set:Nx \l__tblr_h_tl
           { \__tblr_data_item:nen { row } { \int_use:N \c@rownum } { height } }
        \tl_if_empty:NF \l__tblr_h_tl
          {
            \__tblr_data_gput:nenV { row } { \int_use:N \c@rownum }
              { @row-height } \l__tblr_h_tl
          }
        \int_step_inline:nn { \c@colcount }
          {
            \int_incr:N \c@colnum
            \__tblr_measure_cell_update_sizes:nnNNNN
              { \int_use:N \c@rownum }
              { \int_use:N \c@colnum }
              \g__tblr_cell_wd_dim
              \g__tblr_cell_ht_dim
              \g__tblr_cell_head_dim
              \g__tblr_cell_foot_dim
          }
      }
  }

%% Measure and update natural dimensions of the row/column/cell
%% #1: row number; #2 column number; #3: width dimension;
%% #4: total height dimension; #5: head dimension; #6: foot dimension
\cs_new_protected:Npn \__tblr_measure_cell_update_sizes:nnNNNN #1 #2 #3 #4 #5 #6
  {
    \__tblr_get_cell_alignments:nn {#1} {#2}
    \hbox_set:Nn \l_tmpa_box { \__tblr_get_cell_text:nn {#1} {#2} }
    \__tblr_update_cell_size:nnNNNN {#1} {#2} #3 #4 #5 #6
    \__tblr_update_row_size:nnNNN {#1} {#2} #4 #5 #6
    \__tblr_update_col_size:nN {#2} #3
  }

%% #1: row number, #2: column number
\cs_new_protected:Npn \__tblr_get_cell_text:nn #1 #2
  {
```

```
    \int_compare:nNnTF { \__tblr_data_item:neen { cell } {#1} {#2} { omit } } > {0}
      {
        \dim_gzero:N \g__tblr_cell_wd_dim
        \dim_gzero:N \g__tblr_cell_ht_dim
        \dim_gzero:N \g__tblr_cell_head_dim
        \dim_gzero:N \g__tblr_cell_foot_dim
      }
      { \__tblr_get_cell_text_real:nn { #1 } { #2 } }
  }

%% Get cell text, #1: row number, #2: column number
%% If the width of the cell is not set, split it with \\ and compute the width
%% Therefore we always get a vbox for any cell
\cs_new_protected:Npn \__tblr_get_cell_text_real:nn #1 #2
  {
    \group_begin:
    \tl_set:Nx \l__tblr_c_tl { \__tblr_text_item:ne { text } {[#1][#2]} }
    \tl_set:Nx \l__tblr_w_tl
      { \__tblr_data_item:neen { cell } {#1} {#2} { width } }
    \dim_compare:nNnT { \l__tblr_w_tl } < { 0pt } % cell width unset
      {
        \int_compare:nNnT
          { \__tblr_data_item:neen { cell } {#1} {#2} { colspan } } < {2}
          {
            \tl_set:Nx \l__tblr_w_tl
              { \__tblr_data_item:nen { column } {#2} { width } }
          }
      }
    \dim_compare:nNnT { \l__tblr_w_tl } < { 0pt } % column width unset
      {
        \bool_if:NTF \l__tblr_math_mode_bool
          {
            \hbox_set:Nn \l_tmpa_box { $\l__tblr_c_tl$ }
            \tl_set:Nx \l__tblr_w_tl { \box_wd:N \l_tmpa_box }
          }
          {
            \tl_set_eq:NN \l_tmpb_tl \l__tblr_c_tl
            \__tblr_insert_braces:N \l_tmpb_tl
            \seq_set_split:NnV \l_tmpa_seq { \\ } \l_tmpb_tl
            \tl_set:Nn \l__tblr_w_tl { 0pt }
            \seq_map_variable:NNn \l_tmpa_seq \l_tmpa_tl
              {
                \__tblr_remove_braces:N \l_tmpa_tl
                \hbox_set:Nn \l_tmpa_box { \l_tmpa_tl }
                \tl_set:Nx \l__tblr_w_tl
                  { \dim_max:nn { \l__tblr_w_tl } { \box_wd:N \l_tmpa_box } }
              }
          }
      }
    \__tblr_get_vcell_and_sizes:NN \l__tblr_c_tl \l__tblr_w_tl
    \group_end:
  }

%% #1: cell text; #2: box width
\cs_new_protected:Npn \__tblr_get_vcell_and_sizes:NN #1 #2
  {
    \group_begin:
    \vbox_set_top:Nn \l_tmpa_box { \__tblr_make_vcell_text:nN #1 #2 }
```

```
    \vbox_set:Nn \l_tmpb_box { \__tblr_make_vcell_text:nN #1 #2 }
    \dim_gset:Nn \g__tblr_cell_wd_dim { \box_wd:N \l_tmpb_box }
    \dim_gset:Nn \g__tblr_cell_ht_dim
      { \box_ht:N \l_tmpb_box + \box_dp:N \l_tmpb_box }
    \dim_gset:Nn \g__tblr_cell_head_dim { \box_ht:N \l_tmpa_box }
    \dim_gset:Nn \g__tblr_cell_foot_dim { \box_dp:N \l_tmpb_box }
    \tl_case:Nn \g__tblr_cell_valign_tl
      {
        \c__tblr_valign_h_tl
          { \box_use:N \l_tmpa_box }
        \c__tblr_valign_m_tl
          {
            \tl_case:Nn \g__tblr_cell_middle_tl
              {
                \c__tblr_middle_t_tl
                  { \box_use:N \l_tmpa_box }
                \c__tblr_middle_m_tl
                  {
                    \tl_set:Nx \l__tblr_b_tl
                      {
                        \dim_eval:n
                          {
                            ( \g__tblr_cell_ht_dim - \g__tblr_cell_head_dim
                                                   - \g__tblr_cell_foot_dim ) / 2
                          }
                      }
                    \box_set_ht:Nn \l_tmpb_box
                      { \g__tblr_cell_head_dim + \l__tblr_b_tl }
                    \box_set_dp:Nn \l_tmpb_box
                      { \g__tblr_cell_foot_dim + \l__tblr_b_tl }
                    \box_use:N \l_tmpb_box
                  }
                \c__tblr_middle_b_tl
                  { \box_use:N \l_tmpb_box }
              }
          }
        \c__tblr_valign_f_tl
          { \box_use:N \l_tmpb_box }
      }
    \group_end:
  }

\cs_new_eq:NN \__tlbr_halign_l: \raggedright
\cs_new_eq:NN \__tlbr_halign_c: \centering
\cs_new_eq:NN \__tlbr_halign_r: \raggedleft

%% #1: cell text; #2: box width
\cs_new_protected:Npn \__tblr_make_vcell_text:nN #1 #2
  {
    \dim_set:Nn \tex_hsize:D { #2 }
    \@arrayparboxrestore
    \cs:w __tlbr_halign_ \g__tblr_cell_halign_tl : \cs_end:
    \mode_leave_vertical:
    \box_use:N \l__tblr_strut_ht_box
    \bool_if:NTF \l__tblr_math_mode_bool { $#1$ } { #1 }
    \box_use:N \l__tblr_strut_dp_box
  }
```

```
%% #1: total height dimension; #2: head dimension; #3: foot dimension;
%% #4: tl for resulting upper size; #5: tl for resulting lower size

\tl_new:N \l__tblr_middle_body_tl

\cs_new_protected:Npn \__tblr_get_middle_cell_upper_lower:NNNNN #1 #2 #3 #4 #5
  {
    \tl_case:Nn \g__tblr_cell_middle_tl
      {
        \c__tblr_middle_t_tl
          {
            \tl_set:Nx #4 { \dim_use:N #2 }
            \tl_set:Nx #5 { \dim_eval:n { #1 - #2 } }
          }
        \c__tblr_middle_m_tl
          {
            \tl_set:Nx \l__tblr_middle_body_tl { \dim_eval:n { #1 - #2 - #3 } }
            \tl_set:Nx #4 { \dim_eval:n { #2 + \l__tblr_middle_body_tl / 2 } }
            \tl_set:Nx #5 { \dim_eval:n { #3 + \l__tblr_middle_body_tl / 2 } }
          }
        \c__tblr_middle_b_tl
          {
            \tl_set:Nx #4 { \dim_eval:n { #1 - #3 } }
            \tl_set:Nx #5 { \dim_use:N #3 }
          }
      }
  }

%% Update natural dimensions of the cell
%% #1: row number; #2 column number; #3: width dimension;
%% #4: total height dimension; #5: head dimension; #6: foot dimension
\cs_new_protected:Npn \__tblr_update_cell_size:nnNNNN #1 #2 #3 #4 #5 #6
  {
    \group_begin:
    \tl_set:Nx \l__tblr_c_tl
      { \__tblr_data_item:neen { cell } {#1} {#2} { colspan } }
    \int_compare:nNnT { \l__tblr_c_tl } > {1}
      {
        \__tblr_data_gput:neene { cell } {#1} {#2} { @cell-width } {\dim_use:N #3}
        \dim_gzero:N #3 % don't affect column width
      }
    \tl_set:Nx \l__tblr_r_tl
      { \__tblr_data_item:neen { cell } {#1} {#2} { rowspan } }
    \int_compare:nNnT { \l__tblr_r_tl } > {1}
      {
        \tl_case:Nn \g__tblr_cell_valign_tl
          {
            \c__tblr_valign_h_tl
              {
                \tl_set:Nx \l__tblr_u_tl { \dim_use:N #5 }
                \tl_set:Nx \l__tblr_v_tl { \dim_eval:n { #4 - #5 } }
                %% Update the head size of the first span row here
                \__tblr_data_gput_if_larger:nene
                  { row } {#1} { @row-head } { \dim_use:N #5 }
              }
            \c__tblr_valign_f_tl
              {
```

```
                \tl_set:Nx \l__tblr_u_tl { \dim_eval:n { #4 - #6 } }
                \tl_set:Nx \l__tblr_v_tl { \dim_use:N #6 }
                %% Update the foot size of the last span row here
                \__tblr_data_gput_if_larger:nene
                  { row }
                  { \int_eval:n { #1 + \l__tblr_r_tl - 1 } }
                  { @row-foot }
                  { \dim_use:N #6 }
              }
            \c__tblr_valign_m_tl
              {
                \__tblr_get_middle_cell_upper_lower:NNNNN
                  #4 #5 #6 \l__tblr_u_tl \l__tblr_v_tl
              }
          }
        \__tblr_data_gput:neenV { cell } {#1} {#2} { @cell-height } \l__tblr_u_tl
        \__tblr_data_gput:neenV { cell } {#1} {#2} { @cell-depth } \l__tblr_v_tl
        %% Don't affect row sizes
        \dim_gzero:N #4
        \dim_gzero:N #5
        \dim_gzero:N #6
      }
    \group_end:
  }

%% Update size of the row. #1: row number; #2: column number;
%% #3: total height dimension; #4: head dimension; #5: foot dimension
\cs_new_protected:Npn \__tblr_update_row_size:nnNNN #1 #2 #3 #4 #5
  {
    \group_begin:
    %% Note that \l__tblr_h_tl may be empty
    \tl_set:Nx \l__tblr_h_tl
      { \__tblr_data_item:nen { row } {#1} { @row-height } }
    \tl_if_eq:NNTF \g__tblr_cell_valign_tl \c__tblr_valign_m_tl
      {
        \tl_set:Nx \l__tblr_a_tl
          { \__tblr_data_item:nen { row } {#1} { @row-upper } }
        \tl_set:Nx \l__tblr_b_tl
          { \__tblr_data_item:nen { row } {#1} { @row-lower } }
        \__tblr_get_middle_cell_upper_lower:NNNNN
          #3 #4 #5 \l__tblr_u_tl \l__tblr_v_tl
        \dim_compare:nNnT { \l__tblr_u_tl } > { \l__tblr_a_tl }
          {
            \tl_set_eq:NN \l__tblr_a_tl \l__tblr_u_tl
            \__tblr_data_gput:nenV { row } {#1} { @row-upper } \l__tblr_a_tl
          }
        \dim_compare:nNnT { \l__tblr_v_tl } > { \l__tblr_b_tl }
          {
            \tl_set_eq:NN \l__tblr_b_tl \l__tblr_v_tl
            \__tblr_data_gput:nenV { row } {#1} { @row-lower } \l__tblr_b_tl
          }
        \dim_compare:nNnT
          { \l__tblr_a_tl + \l__tblr_b_tl } > { \l__tblr_h_tl + 0pt }
          {
            \__tblr_data_gput:nene { row } {#1} { @row-height }
              { \dim_eval:n { \l__tblr_a_tl + \l__tblr_b_tl } }
          }
      }
```

```
      {
        \tl_set:Nx \l__tblr_e_tl
          { \__tblr_data_item:nen { row } {#1} { @row-head } }
        \tl_set:Nx \l__tblr_f_tl
          { \__tblr_data_item:nen { row } {#1} { @row-foot } }
        \dim_compare:nNnT {#4} > {\l__tblr_e_tl}
          {
            \__tblr_data_gput:nene { row } {#1} { @row-head } { \dim_use:N #4 }
          }
        \dim_compare:nNnT {#5} > {\l__tblr_f_tl}
          {
            \__tblr_data_gput:nene { row } {#1} { @row-foot } { \dim_use:N #5 }
          }
        \tl_set:Nx \l__tblr_x_tl { \dim_max:nn {#4} { \l__tblr_e_tl } }
        \tl_set:Nx \l__tblr_y_tl { \dim_max:nn {#5} { \l__tblr_f_tl } }
        \dim_compare:nNnT
          { #3 - #4 - #5 } > { \l__tblr_h_tl - \l__tblr_x_tl - \l__tblr_y_tl }
          {
            \__tblr_data_gput:nene { row } {#1} { @row-height }
              {
                \dim_eval:n
                  {
                    \l__tblr_x_tl
                    + \dim_use:N #3 - \dim_use:N #4 - \dim_use:N #5
                    + \l__tblr_y_tl
                  }
              }
          }
      }
    \group_end:
  }

%% Update size of the column. #1: column number; #2: width dimension

\cs_new_protected:Npn \__tblr_update_col_size:nN #1 #2
  {
    \tl_set:Nx \l_tmpb_tl
      { \__tblr_data_item:nen { column } {#1} { @col-width } }
    \bool_lazy_or:nnT
      { \tl_if_empty_p:N \l_tmpb_tl }
      { \dim_compare_p:nNn { \dim_use:N #2 } > { \l_tmpb_tl } }
      {
        \__tblr_data_gput:nene { column } {#1} { @col-width } { \dim_use:N #2 }
      }
  }

%%% ------------------------------------------------------------
%%  \section{Calculate and Adjust Extendable Columns}
%%% ------------------------------------------------------------

%% Compute column widths when there are some extendable columns

\dim_new:N \l__column_target_dim
\prop_new:N \l__column_coefficient_prop
\prop_new:N \l__column_natural_width_prop
\prop_new:N \l__column_computed_width_prop
```

```
\msg_new:nnn { tabularray } { table-width-too-small }
  { Table ~ width ~ is ~ too ~ small, ~ need ~ #1 ~ more! }

\cs_new_protected:Npn \__tblr_compute_extendable_column_width:
  {
    \__tblr_collect_extendable_column_width:
    \dim_compare:nNnTF { \l__column_target_dim } < { 0pt }
      {
        \msg_warning:nnx { tabularray } { table-width-too-small }
          { \dim_abs:n { \l__column_target_dim } }
      }
      {
        \prop_if_empty:NF \l__column_coefficient_prop
          { \__tblr_adjust_extendable_column_width: }
      }
  }

\cs_new_protected:Npn \__tblr_collect_extendable_column_width:
  {
    \tl_set:Nx \l_tmpa_tl { \__tblr_prop_item:nn {table} {width} }
    \tl_if_empty:NTF \l_tmpa_tl
      { \dim_set_eq:NN \l__column_target_dim \linewidth }
      { \dim_set:Nn \l__column_target_dim { \l_tmpa_tl } }
    \prop_clear:N \l__column_coefficient_prop
    \prop_clear:N \l__column_natural_width_prop
    \prop_clear:N \l__column_computed_width_prop
    \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
      {
        \tl_set:Nx \l__tblr_a_tl
          { \__tblr_data_item:nen { column } { \l__tblr_j_tl } { width } }
        \tl_set:Nx \l__tblr_b_tl
          { \__tblr_data_item:nen { column } { \l__tblr_j_tl } { coefficient } }
        \tl_set:Nx \l__tblr_c_tl
          { \__tblr_data_item:nen  { column } { \l__tblr_j_tl } { @col-width } }
        \dim_compare:nNnTF { \l__tblr_a_tl } < { 0pt } % column width unset
          {
            \dim_compare:nNnTF { \l__tblr_b_tl pt } = { 0pt }
              { \dim_sub:Nn \l__column_target_dim { \l__tblr_c_tl } }
              {
                \prop_put:Nxx \l__column_coefficient_prop
                  { \l__tblr_j_tl } { \l__tblr_b_tl }
                \prop_put:Nxn \l__column_computed_width_prop
                  { \l__tblr_j_tl } { 0pt }
                \dim_compare:nNnF { \l__tblr_b_tl pt } > { 0pt }
                  {
                    \prop_put:Nxx \l__column_natural_width_prop
                      { \l__tblr_j_tl } { \l__tblr_c_tl }
                  }
              }
          }
          { \dim_sub:Nn \l__column_target_dim { \l__tblr_a_tl } }
        \tl_set:Nx \l__tblr_a_tl
          { \__tblr_text_item:ne { vline } { [\l__tblr_j_tl] / @vline-width } }
        \tl_set:Nx \l__tblr_b_tl
          { \__tblr_data_item:nen { column } { \l__tblr_j_tl } { leftsep } }
        \tl_set:Nx \l__tblr_c_tl
          { \__tblr_data_item:nen { column } { \l__tblr_j_tl } { rightsep } }
        \dim_set:Nn \l__column_target_dim
```

```
          { \l__column_target_dim - \l__tblr_a_tl - \l__tblr_b_tl - \l__tblr_c_tl }
      }
    \tl_set:Nx \l__tblr_a_tl
      {
        \__tblr_text_item:ne { vline }
          { [\int_eval:n {\c@colcount + 1}] / @vline-width }
      }
    \tl_if_empty:NF \l__tblr_a_tl
      { \dim_sub:Nn \l__column_target_dim { \l__tblr_a_tl } }
    \LogTblrTracing { target }
  }

%% If all columns have negative coefficients and small natural widths,
%% \l__column_coefficient_prop will be empty after one or more rounds
\cs_new_protected:Npn \__tblr_adjust_extendable_column_width:
  {
    \bool_while_do:nn
      { \dim_compare_p:nNn { \l__column_target_dim } > { \hfuzz } }
      {
        \prop_if_empty:NTF \l__column_coefficient_prop
          { \__tblr_adjust_extendable_column_width_negative: }
          { \__tblr_adjust_extendable_column_width_once: }
      }
    \prop_map_inline:Nn \l__column_computed_width_prop
      {
        \__tblr_data_gput:nnne { column } {##1} { width } {##2}
        \__tblr_data_gput:nnnn { column } {##1} { @col-width } { 0pt }
      }
    \__tblr_calculate_cell_sizes:
  }

%% We use dimen register, since the coefficient may be a decimal number
\cs_new_protected:Npn \__tblr_adjust_extendable_column_width_once:
  {
    \dim_zero:N \l_tmpa_dim
    \prop_map_inline:Nn \l__column_coefficient_prop
      {
        \dim_add:Nn \l_tmpa_dim { \dim_abs:n { ##2 pt } }
      }
    \tl_set:Nx \l__tblr_w_tl
      { \dim_ratio:nn { \l__column_target_dim } { \l_tmpa_dim } }
    \dim_zero:N \l__column_target_dim
    \prop_map_inline:Nn \l__column_coefficient_prop
      {
        \tl_set:Nx \l__tblr_a_tl
          { \dim_eval:n { \dim_abs:n { ##2 pt } * \l__tblr_w_tl } }
        \dim_compare:nNnTF { ##2 pt } > { 0pt }
          {
            \__tblr_add_dimen_value:Nnn
              \l__column_computed_width_prop { ##1 } { \l__tblr_a_tl }
          }
          {
            \tl_set:Nx \l__tblr_b_tl
              { \prop_item:Nn \l__column_natural_width_prop { ##1 } }
            \tl_set:Nx \l__tblr_c_tl
              { \prop_item:Nn \l__column_computed_width_prop { ##1 } }
            \dim_compare:nNnTF { \l__tblr_a_tl + \l__tblr_c_tl } > { \l__tblr_b_tl }
              {
```

```
                \prop_put:Nnx \l__column_computed_width_prop
                  { ##1 } { \l__tblr_b_tl }
                \dim_add:Nn \l__column_target_dim
                  { \l__tblr_a_tl + \l__tblr_c_tl - \l__tblr_b_tl }
                \prop_remove:Nn \l__column_coefficient_prop { ##1 }
              }
              {
                \__tblr_add_dimen_value:Nnn
                  \l__column_computed_width_prop { ##1 } { \l__tblr_a_tl }
              }
          }
      }
    \LogTblrTracing { target }
  }

\cs_new_protected:Npn \__tblr_adjust_extendable_column_width_negative:
  {
    \dim_zero:N \l_tmpa_dim
    \prop_map_inline:Nn \l__column_natural_width_prop
      { \dim_add:Nn \l_tmpa_dim { ##2 } }
    \tl_set:Nx \l_tmpa_tl
      { \dim_ratio:nn { \l__column_target_dim } { \l_tmpa_dim } }
    \dim_zero:N \l__column_target_dim
    \prop_map_inline:Nn \l__column_natural_width_prop
      {
        \tl_set:Nx \l_tmpb_tl { \dim_eval:n { ##2 * \l_tmpa_tl } }
        \__tblr_add_dimen_value:Nnn
          \l__column_computed_width_prop { ##1 } { \l_tmpb_tl }
      }
    \LogTblrTracing { target }
  }

%%% ------------------------------------------------------------
%%  \section{Calculate and Adjust Multispan Cells}
%%% ------------------------------------------------------------

%% Compute and adjust widths when there are some span cells.
%% By default, we will compute column widths from span widths;
%% but if we set table option "hspan = minimal",
%% we will compute span widths from column widths.

\cs_new_protected:Npn \__tblr_adjust_sizes_for_span_cells:
  {
    \__tblr_prop_if_in:nnT {table} {colspan}
      {
        \__tblr_collect_column_widths_skips:
        \str_if_eq:xnTF
          { \__tblr_prop_item:ne {table} {hspan} } {minimal}
          {
            \__tblr_set_span_widths_from_column_widths:
          }
          {
            \__tblr_collect_span_widths:
            \__tblr_set_column_widths_from_span_widths:
          }
        \LogTblrTracing { column }
        \__tblr_calculate_cell_sizes:
```

```
      }
    \__tblr_prop_if_in:nnT {table} {rowspan}
      {
        \__tblr_collect_row_heights_skips:
        \__tblr_collect_span_heights:
        \__tblr_set_row_heights_from_span_heights:
        \LogTblrTracing {row}
      }
  }

\prop_new:N \l__tblr_col_item_skip_size_prop
\prop_new:N \l__tblr_col_span_size_prop
\prop_new:N \l__tblr_row_item_skip_size_prop
\prop_new:N \l__tblr_row_span_size_prop

\cs_new_protected:Npn \__tblr_collect_column_widths_skips:
  {
    \prop_clear:N \l__tblr_col_item_skip_size_prop
    \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
      {
        \int_compare:nNnTF { \l__tblr_j_tl } > { 1 }
          {
            \prop_put:Nxx \l__tblr_col_item_skip_size_prop { skip[\l__tblr_j_tl] }
              {
                \dim_eval:n
                  {
                    \__tblr_data_item:nen { column }
                      { \int_eval:n { \l__tblr_j_tl - 1 } } { rightsep }
                    +
                    \__tblr_text_item:ne { vline }
                      { [\l__tblr_j_tl] / @vline-width }
                    +
                    \__tblr_data_item:nen { column } { \l__tblr_j_tl } { leftsep }
                  }
              }
          }
          {
            \prop_put:Nxn \l__tblr_col_item_skip_size_prop { skip[\l__tblr_j_tl] }
              { 0pt }
          }
        \prop_put:Nxx \l__tblr_col_item_skip_size_prop { item[\l__tblr_j_tl] }
          { \__tblr_data_item:nen { column } { \l__tblr_j_tl } { @col-width } }
      }
    \__tblr_do_if_tracing:nn { cellspan }
      { \prop_log:N \l__tblr_col_item_skip_size_prop }
  }

\cs_new_protected:Npn \__tblr_collect_row_heights_skips:
  {
    \prop_clear:N \l__tblr_row_item_skip_size_prop
    \int_step_variable:nNn { \c@rowcount } \l__tblr_i_tl
      {
        \int_compare:nNnTF { \l__tblr_i_tl } > { 1 }
          {
            \prop_put:Nxx \l__tblr_row_item_skip_size_prop { skip[\l__tblr_i_tl] }
              {
                \dim_eval:n
```

```
                {
                  \__tblr_data_item:nen { row }
                    { \int_eval:n {\l__tblr_i_tl - 1} } { belowsep }
                  +
                  \__tblr_text_item:ne { hline }
                    { [\l__tblr_i_tl] / @hline-height }
                  +
                  \__tblr_data_item:nen { row } { \l__tblr_i_tl } { abovesep }
                }
            }
          }
          {
            \prop_put:Nxn \l__tblr_row_item_skip_size_prop { skip[\l__tblr_i_tl] }
              { 0pt }
          }
        \__tblr_collect_one_row_height:NN \l__tblr_i_tl \l__tblr_h_tl
        \prop_put:Nxx \l__tblr_row_item_skip_size_prop
          { item[\l__tblr_i_tl] } { \l__tblr_h_tl }
      }
    \__tblr_do_if_tracing:nn { cellspan }
      { \prop_log:N \l__tblr_row_item_skip_size_prop }
  }

%% #1: row number; #2: tl with result
\cs_new_protected:Npn \__tblr_collect_one_row_height:NN #1 #2
  {
    \tl_set:Nx #2 { \__tblr_data_item:nen { row } {#1} { @row-height } }
  }

\cs_new_protected:Npn \__tblr_collect_span_widths:
  {
    \prop_clear:N \l__tblr_col_span_size_prop
    \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
      {
        \int_step_variable:nNn { \c@rowcount } \l__tblr_i_tl
          {
            \tl_set:Nx \l__tblr_a_tl
              {
                \__tblr_data_item:neen { cell }
                  { \l__tblr_i_tl } { \l__tblr_j_tl } { colspan }
              }
            \int_compare:nNnT { \l__tblr_a_tl } > {1}
              {
                \__tblr_put_if_larger:Nxx \l__tblr_col_span_size_prop
                  {
                    ( \l__tblr_j_tl -
                      \int_eval:n {\l__tblr_j_tl + \l__tblr_a_tl - 1} )
                  }
                  {
                    \__tblr_data_item:neen { cell }
                      { \l__tblr_i_tl } { \l__tblr_j_tl } { @cell-width }
                  }
              }
          }
      }
    \__tblr_do_if_tracing:nn { cellspan }
      { \prop_log:N \l__tblr_col_span_size_prop }
  }
```

```
\prop_new:N \l__tblr_row_span_to_row_prop

\cs_new_protected:Npn \__tblr_collect_span_heights:
  {
    \prop_clear:N \l__tblr_row_span_to_row_prop
    \prop_clear:N \l__tblr_row_span_size_prop
    \int_step_variable:nNn { \c@rowcount } \l__tblr_i_tl
      {
        \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
          {
            \tl_set:Nx \l__tblr_a_tl
              {
                \__tblr_data_item:neen { cell }
                  { \l__tblr_i_tl } { \l__tblr_j_tl } { rowspan }
              }
            \int_compare:nNnT { \l__tblr_a_tl } > {1}
              {
                \tl_set:Nx \l__tblr_v_tl
                  {
                    \__tblr_data_item:neen { cell }
                      { \l__tblr_i_tl } { \l__tblr_j_tl } { valign }
                  }
                \tl_if_eq:NnT \l__tblr_v_tl { h }
                  {
                    \tl_set:Nx \l__tblr_h_tl
                      {
                        \__tblr_data_item:nen { row }
                          { \l__tblr_i_tl } { @row-head }
                      }
                    \__tblr_data_gput:neenV { cell }
                      { \l__tblr_i_tl } { \l__tblr_j_tl } { @cell-height }
                      \l__tblr_h_tl
                  }
                \tl_if_eq:NnT \l__tblr_v_tl { f }
                  {
                    \tl_set:Nx \l__tblr_d_tl
                      {
                        \__tblr_data_item:nen
                          { row }
                          { \int_eval:n { \l__tblr_i_tl + \l__tblr_a_tl - 1 } }
                          { @row-foot }
                      }
                    \__tblr_data_gput:neenV { cell }
                      { \l__tblr_i_tl } { \l__tblr_j_tl } { @cell-depth }
                      \l__tblr_d_tl
                  }
                \__tblr_put_if_larger:Nxx \l__tblr_row_span_size_prop
                  {
                    ( \l__tblr_i_tl -
                      \int_eval:n {\l__tblr_i_tl + \l__tblr_a_tl - 1} )
                  }
                  {
                    \dim_eval:n
                      {
                        \__tblr_data_item:neen { cell }
                          { \l__tblr_i_tl } { \l__tblr_j_tl } { @cell-height }
                        +
                        \__tblr_data_item:neen { cell }
```

```
                        { \l__tblr_i_tl } { \l__tblr_j_tl } { @cell-depth }
                    }
                }
              \prop_put:Nxx \l__tblr_row_span_to_row_prop
                { [\l__tblr_i_tl][\l__tblr_j_tl] }
                { \int_eval:n {\l__tblr_i_tl + \l__tblr_a_tl - 1} }
            }
        }
    }
  \__tblr_do_if_tracing:nn { cellspan }
    {
      \prop_log:N \l__tblr_row_span_to_row_prop
      \prop_log:N \l__tblr_row_span_size_prop
    }
}

%% Compute and set column widths from span widths
\cs_new_protected:Npn \__tblr_set_column_widths_from_span_widths:
{
  \__tblr_calc_item_sizes_from_span_sizes:xNN
    { \int_use:N \c@colcount }
    \l__tblr_col_item_skip_size_prop
    \l__tblr_col_span_size_prop
  \__tblr_set_all_column_widths:
}

%% Compute and set row heights from span heights
\cs_new_protected:Npn \__tblr_set_row_heights_from_span_heights:
{
  \__tblr_calc_item_sizes_from_span_sizes:xNN
    { \int_use:N \c@rowcount }
    \l__tblr_row_item_skip_size_prop
    \l__tblr_row_span_size_prop
  \__tblr_set_all_row_heights:
}

%% See page 245 in Chapter 22 of TeXbook
%% #1: total number of items
%% #2: prop list with item sizes and skip sizes; #3: prop list with span sizes
\cs_new_protected:Npn \__tblr_calc_item_sizes_from_span_sizes:nNN #1 #2 #3
{
  \int_step_variable:nNn { #1 } \l__tblr_j_tl
    {
      \dim_set:Nn \l__tblr_w_dim
        {
          \prop_item:Ne #2 { item[\l__tblr_j_tl] }
        }
      \int_step_variable:nNn { \l__tblr_j_tl - 1 } \l__tblr_i_tl
        {
          \tl_set:Nx \l__tblr_a_tl
            { \prop_item:Ne #3 { (\l__tblr_i_tl-\l__tblr_j_tl) } }
          \tl_if_empty:NF \l__tblr_a_tl
            {
              \int_step_variable:nnNn
                { \l__tblr_i_tl } { \l__tblr_j_tl - 1 } \l__tblr_k_tl
                {
                  \__tblr_do_if_tracing:nn { cellspan }
```

```
                    {
                      \tl_log:x
                        { \l__tblr_j_tl : \l__tblr_i_tl -> \l__tblr_k_tl }
                    }
                  \tl_set:Nx \l_tmpa_tl
                    {
                      \prop_item:Ne #2 { itemskip[\l__tblr_k_tl] }
                    }
                  \tl_set:Nx \l__tblr_a_tl
                    { \dim_eval:n { \l__tblr_a_tl - \l_tmpa_tl } }
                }
              \dim_compare:nNnT { \l__tblr_a_tl } > { \l__tblr_w_dim }
                {
                  \dim_set:Nn \l__tblr_w_dim { \l__tblr_a_tl }
                }
            }
        }
      \prop_put:Nxx #2
        { item[\l__tblr_j_tl] } { \dim_use:N \l__tblr_w_dim }
      \int_compare:nNnT { \l__tblr_j_tl } < { #1 }
        {
          \tl_set:Nx \l_tmpb_tl
            {
              \prop_item:Ne #2
                { skip[\int_eval:n { \l__tblr_j_tl + 1} ] }
            }
          \dim_add:Nn \l__tblr_w_dim { \l_tmpb_tl }
          \prop_put:Nxx #2
            { itemskip[\l__tblr_j_tl] } { \dim_use:N \l__tblr_w_dim }
        }
    }
  \__tblr_do_if_tracing:nn { cellspan } { \prop_log:N #2 }
}
\cs_generate_variant:Nn \__tblr_calc_item_sizes_from_span_sizes:nNN { x }

\cs_new_protected:Npn \__tblr_set_all_column_widths:
{
  \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
    {
      \__tblr_data_gput:nene { column }
        { \l__tblr_j_tl } { @col-width }
        { \prop_item:Ne \l__tblr_col_item_skip_size_prop { item[\l__tblr_j_tl] } }
    }
}

\cs_new_protected:Npn \__tblr_set_all_row_heights:
{
  \int_step_variable:nNn { \c@rowcount } \l__tblr_i_tl
    {
      \tl_set:Nx \l__tblr_h_tl
        {
          \__tblr_data_item:nen { row } { \l__tblr_i_tl } { @row-head }
        }
      \tl_set:Nx \l__tblr_d_tl
        {
          \__tblr_data_item:nen { row } { \l__tblr_i_tl } { @row-foot }
        }
      \tl_set:Nx \l__tblr_a_tl
```

```
          {
            \prop_item:Ne \l__tblr_row_item_skip_size_prop { item[\l__tblr_i_tl] }
          }
        \__tblr_collect_one_row_height:NN \l__tblr_i_tl \l__tblr_t_tl
        \__tblr_data_gput:nene { row }
          { \l__tblr_i_tl } { @row-height } { \l__tblr_a_tl }
      }
  }

\cs_new_protected:Npn \__tblr_get_span_key_row_col:w [#1][#2]
  {
    \tl_set:Nn \l__tblr_i_tl {#1}
    \tl_set:Nn \l__tblr_j_tl {#2}
  }

%% Compute and set span widths from column widths
\cs_new_protected:Npn \__tblr_set_span_widths_from_column_widths:
  {
    \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
      {
        \int_step_variable:nNn { \c@rowcount } \l__tblr_i_tl
          {
            \tl_set:Nx \l__tblr_a_tl
              {
                \__tblr_data_item:neen { cell }
                  { \l__tblr_i_tl } { \l__tblr_j_tl } { colspan }
              }
            \int_compare:nNnT { \l__tblr_a_tl } > {1}
              {
                \__tblr_calc_span_widths:xxN
                  { \l__tblr_j_tl }
                  { \int_eval:n { \l__tblr_j_tl + \l__tblr_a_tl - 1 } }
                  \l__tblr_w_dim
                \__tblr_data_gput:neene { cell }
                  { \l__tblr_i_tl } { \l__tblr_j_tl } { width }
                  { \dim_use:N \l__tblr_w_dim }
              }
          }
      }
  }

%% Cell is spanned from col #1 to col #2, #3 is the return dim
\cs_new_protected:Npn \__tblr_calc_span_widths:nnN #1 #2 #3
  {
    \dim_zero:N #3
    \int_step_inline:nnn { #1 } { #2 }
      {
        \tl_set:Nx \l_tmpa_tl
          { \prop_item:Ne \l__tblr_col_item_skip_size_prop { skip[##1] } }
        \tl_set:Nx \l_tmpb_tl
          { \prop_item:Ne \l__tblr_col_item_skip_size_prop { item[##1] } }
        \dim_add:Nn #3 { \dim_eval:n { \l_tmpa_tl + \l_tmpb_tl } }
      }
  }
\cs_generate_variant:Nn \__tblr_calc_span_widths:nnN { xxN }

%%% ----------------------------------------------------------
```

```
%%  \section{Build the Whole Table}
%%% ------------------------------------------------------

\tl_new:N \__tlbr_vbox_align_tl
\tl_const:Nn \__tlbr_vbox_t_tl {t}
\tl_const:Nn \__tlbr_vbox_m_tl {m}
\tl_const:Nn \__tlbr_vbox_c_tl {c}
\tl_const:Nn \__tlbr_vbox_b_tl {b}

\box_new:N \l__tblr_table_box

%% #1: table alignment
\cs_new_protected:Npn \__tblr_build_whole:n #1
  {
    \bool_if:NTF \l__tblr_long_table_bool
      { \__tblr_build_long_table:n {#1} }
      { \__tblr_build_short_table:n {#1} }
  }

\dim_new:N \l__tblr_remain_height_dim
\tl_new:N \l__tblr_long_from_tl

\cs_new_protected:Npn \__tblr_build_long_table:n #1
  {
    %\dim_log:N \pagegoal
    %\dim_log:N \pagetotal
    \dim_set:Nn \l__tblr_remain_height_dim { \pagegoal - \pagetotal }
    \tl_set:Nn \l__tblr_long_from_tl {1}
    \int_step_variable:nNn { \c@rowcount } \l__tblr_i_tl
      {
        \dim_set:Nn \l_tmpa_dim
          {
            \__tblr_text_item:ne { hline } { [\l__tblr_i_tl] / @hline-height }
            +
            \__tblr_data_item:nen { row } { \l__tblr_i_tl } { abovesep }
            +
            \__tblr_data_item:nen { row } { \l__tblr_i_tl } { @row-height }
            +
            \__tblr_data_item:nen { row } { \l__tblr_i_tl } { belowsep }
          }
        \dim_compare:nNnTF
          { \l_tmpa_dim } > { \l__tblr_remain_height_dim }
          {
            \tl_log:N \l__tblr_i_tl
            \__tblr_build_page_table:nnx {#1}
              { \l__tblr_long_from_tl } { \int_eval:n { \l__tblr_i_tl - 1 } }
            \newpage
            \hbox{}\kern-\topskip\nobreak
            \leavevmode
            %\dim_log:N \pagegoal
            %\dim_log:N \pagetotal
            \dim_set:Nn \l__tblr_remain_height_dim
              { \pagegoal - \pagetotal - \l_tmpa_dim }
            \tl_set_eq:NN \l__tblr_long_from_tl \l__tblr_i_tl
          }
          {
            \dim_add:Nn \l__tblr_remain_height_dim { -\l_tmpa_dim }
```

```
          }
      }
    \__tblr_build_page_table:nnn {#1} { \l__tblr_long_from_tl } { \c@rowcount }
  }

\cs_new_protected:Npn \__tblr_build_page_table:nnn #1 #2 #3
  {
    \__tblr_build_one_table:nn {#2} {#3}
    \__tblr_halign_whole:Nn \l__tblr_table_box #1
  }
\cs_generate_variant:Nn \__tblr_build_page_table:nnn { nnx }

\cs_new_protected:Npn \__tblr_halign_whole:Nn #1 #2
  {
    \noindent
    \hbox_to_wd:nn { \linewidth }
      {
        \tl_if_eq:nnF {#2} {l} { \hfil }
        \box_use:N #1
        \tl_if_eq:nnF {#2} {r} { \hfil }
      }
  }

\cs_new_protected:Npn \__tblr_build_short_table:n #1
  {
    \__tblr_build_one_table:nn {1} {\c@rowcount}
    \__tblr_valign_whole:Nn \l__tblr_table_box #1
  }

%% #1: row from; #2: row to
\cs_new_protected:Npn \__tblr_build_one_table:nn #1 #2
  {
    \vbox_set:Nn \l__tblr_table_box
      {
        \int_step_variable:nnNn {#1} {#2} \l__tblr_i_tl
          {
            \hbox:n { \__tblr_build_hline:V \l__tblr_i_tl }
            \hrule height ~ 0pt % remove lineskip between hlines and rows
            \hbox:n { \__tblr_build_row:N \l__tblr_i_tl }
            \hrule height ~ 0pt
          }
        \hbox:n { \__tblr_build_hline:n { \int_eval:n {#2 + 1} } }
      }
  }

\cs_new_protected:Npn \__tblr_valign_whole:Nn #1 #2
  {
    \group_begin:
    \tl_set:Nn \__tblr_vbox_align_tl {#2}
    \dim_set:Nn \l__tblr_t_dim { \box_ht:N #1 + \box_dp:N #1 }
    \tl_case:NnF \__tblr_vbox_align_tl
      {
        \__tblr_vbox_m_tl
          { \__tblr_valign_whole_middle:N #1 }
        \__tblr_vbox_c_tl
          { \__tblr_valign_whole_middle:N #1 }
        \__tblr_vbox_t_tl
```

```
          { \__tblr_valign_whole_top:N #1 }
        \__tlbr_vbox_b_tl
          { \__tblr_valign_whole_bottom:N #1 }
      }
      { \__tblr_valign_whole_middle:N #1 }
    \group_end:
  }

\cs_new_protected:Npn \__tblr_valign_whole_middle:N #1
  {
    \hbox:n { $ \m@th \tex_vcenter:D { \vbox_unpack_drop:N #1 } $ }
  }

\cs_new_protected:Npn \__tblr_valign_whole_top:N #1
  {
    \tl_set:Nx \l__tblr_a_tl
      { \__tblr_text_item:ne { hline } { [1] / @hline-height } }
    %% Note that \l__tblr_b_tl may be empty
    \tl_set:Nx \l__tblr_b_tl
      { \__tblr_prop_item:ne { table } { baseline } }
    \bool_lazy_or:nnTF
      { \dim_compare_p:nNn { \l__tblr_a_tl } = { 0pt } }
      { \int_compare_p:nNn { \l__tblr_b_tl + 0 } = { 1 } }
      {
        \dim_set:Nn \l__tblr_h_dim
          {
            \__tblr_data_item:nnn { row } {1} { abovesep }
            +
            ( \__tblr_data_item:nnn { row } {1} { @row-height }
              +
              \__tblr_data_item:nnn { row } {1} { @row-upper }
              -
              \__tblr_data_item:nnn { row } {1} { @row-lower }
            ) / 2
          }
        \dim_set:Nn \l__tblr_d_dim { \l__tblr_t_dim - \l__tblr_h_dim }
      }
      {
        \dim_set:Nn \l__tblr_h_dim { 0pt }
        \dim_set_eq:NN \l__tblr_d_dim \l__tblr_t_dim
      }
    \box_set_ht:Nn #1 { \l__tblr_h_dim }
    \box_set_dp:Nn #1 { \l__tblr_d_dim }
    \box_use_drop:N #1
  }

\cs_new_protected:Npn \__tblr_valign_whole_bottom:N #1
  {
    \tl_set:Nx \l__tblr_a_tl
      {
        \__tblr_text_item:ne { hline }
          { [\int_eval:n {\c@rowcount + 1}] / @hline-height }
      }
    %% Note that \l__tblr_b_tl may be empty
    \tl_set:Nx \l__tblr_b_tl
      { \__tblr_prop_item:ne { table } { baseline } }
    \bool_lazy_or:nnTF
```

```
      { \dim_compare_p:nNn { \l__tblr_a_tl } = { 0pt } }
      { \int_compare_p:nNn { \l__tblr_b_tl + 0 } = { \c@rowcount } }
      {
        \dim_set:Nn \l__tblr_d_dim
          {
            ( \__tblr_data_item:nen { row }
                { \int_use:N \c@rowcount } { @row-height }
              -
              \__tblr_data_item:nen { row }
                { \int_use:N \c@rowcount } { @row-upper }
              +
              \__tblr_data_item:nen { row }
                { \int_use:N \c@rowcount } { @row-lower }
            ) / 2
            +
            \__tblr_data_item:nnn { row } {1} { belowsep }
          }
        \dim_set:Nn \l__tblr_h_dim { \l__tblr_t_dim - \l__tblr_d_dim }
      }
      {
        \dim_set:Nn \l__tblr_d_dim { 0pt }
        \dim_set_eq:NN \l__tblr_h_dim \l__tblr_t_dim
      }
    \box_set_ht:Nn #1 { \l__tblr_h_dim }
    \box_set_dp:Nn #1 { \l__tblr_d_dim }
    \box_use_drop:N #1
  }

\dim_new:N \l__tblr_col_o_wd_dim
\dim_new:N \l__tblr_col_b_wd_dim

%% Build hline. #1: row number
\cs_new_protected:Npn \__tblr_build_hline:n #1
  {
    \int_step_inline:nn { \c@colcount }
      { \__tblr_build_hline_segment:nn { #1 } { ##1 } }
  }
\cs_generate_variant:Nn \__tblr_build_hline:n { x, V }

%% #1: row number, #2: column number
\cs_new_protected:Npn \__tblr_build_hline_segment:nn #1 #2
  {
    \tl_set:Nx \l__tblr_n_tl
      { \__tblr_text_item:ne { hline } { [#1] / @hline-count } }
    \tl_set:Nx \l__tblr_o_tl
      { \__tblr_text_item:ne { hline } { [#1][#2] / omit } }
    \__tblr_get_col_outer_width_border_width:nNN {#2}
      \l__tblr_col_o_wd_dim \l__tblr_col_b_wd_dim
    \tl_if_empty:NTF \l__tblr_o_tl
      {
        \int_compare:nNnT { \l__tblr_n_tl } > {0}
          { \__tblr_build_hline_segment_real:nn {#1} {#2} }
      }
      { \__tblr_build_hline_segment_omit:nn {#1} {#2} }
  }

%% #1: row number, #2: column number
```

```
\cs_new_protected:Npn \__tblr_build_hline_segment_omit:nn #1 #2
  {
    \skip_horizontal:n { \l__tblr_col_o_wd_dim - \l__tblr_col_b_wd_dim }
  }

%% #1: row number, #2: column number
\cs_new_protected:Npn \__tblr_build_hline_segment_real:nn #1 #2
  {
    \tl_set:Nx \l__tblr_s_tl
      { \__tblr_prop_item:ne { table } { rulesep } }
    \vbox_set:Nn \l__tblr_c_box
      {
        %% add an empty hbox to support vbox width
        \tex_hbox:D to \l__tblr_col_o_wd_dim {}
        \int_step_inline:nn { \l__tblr_n_tl }
          {
            \tl_set:Nx \l__tblr_h_tl
              { \__tblr_text_item:ne { hline } { [#1](##1) / @hline-height } }
            \hrule height ~ 0pt % remove lineskip
            \hbox_set_to_wd:Nnn \l__tblr_b_box { \l__tblr_col_o_wd_dim }
              {
                \tl_set:Nx \l__tblr_f_tl
                  { \__tblr_text_item:ne { hline } { [#1][#2](##1) / fg } }
                \tl_if_empty:NF \l__tblr_f_tl { \color{\l__tblr_f_tl} }
                \__tblr_get_hline_segment_child:nnn {#1} {#2} {##1}
              }
            \box_set_ht:Nn \l__tblr_b_box { \l__tblr_h_tl }
            \box_set_dp:Nn \l__tblr_b_box { 0pt }
            \box_use:N \l__tblr_b_box
            \skip_vertical:n { \l__tblr_s_tl }
          }
        \skip_vertical:n { - \l__tblr_s_tl }
      }
    \box_use:N \l__tblr_c_box
    \skip_horizontal:n { - \l__tblr_col_b_wd_dim }
  }

%% Read from table specifications and calculate the widths of row and border
%% column outer width = content width + colsep width + border width
%% #1: the column number, #2: outer width, #3: border width
\cs_new_protected:Npn \__tblr_get_col_outer_width_border_width:nNN #1 #2 #3
  {
    \dim_set:Nn #3
      { \__tblr_text_item:ne { vline } { [\int_eval:n {#1 + 1}] / @vline-width } }
    \dim_set:Nn #2
      {
        \__tblr_text_item:ne { vline } { [#1] / @vline-width }
        +
        \__tblr_data_item:nen { column } {#1} { leftsep }
        +
        \__tblr_data_item:nen { column } {#1} { @col-width }
        +
        \__tblr_data_item:nen { column } {#1} { rightsep }
        +
        #3
      }
  }
```

```
\dim_new:N \l__tblr_row_ht_dim
\dim_new:N \l__tblr_row_dp_dim
\dim_new:N \l__tblr_row_abovesep_dim
\dim_new:N \l__tblr_row_belowsep_dim

%% Build current row, #1: row number
\cs_new_protected:Npn \__tblr_build_row:N #1
  {
    \__tblr_get_row_inner_height_depth:VNNNN #1
      \l__tblr_row_ht_dim \l__tblr_row_dp_dim
      \l__tblr_row_abovesep_dim \l__tblr_row_belowsep_dim
    \vrule width ~ 0pt ~ height ~ \l__tblr_row_ht_dim ~ depth ~ \l__tblr_row_dp_dim
    \int_step_variable:nNn { \c@colcount } \l__tblr_j_tl
      {
        \__tblr_build_vline_segment:nn {#1} { \l__tblr_j_tl }
        \__tblr_build_cell:NN #1 \l__tblr_j_tl
      }
    \__tblr_build_vline_segment:nn {#1} { \int_eval:n {\c@colcount + 1} }
  }

%% Read from table specifications and calculate inner height/depth of the row
%% inner height = abovesep + above vspace + row upper
%% inner depth = row lower + below vspace + belowsep
%% #1: the row number; #2: resulting inner height; #3: resulting inner depth;
%% #4: restulting abovesep; #5: restulting belowsep.

\dim_new:N \l__row_upper_dim
\dim_new:N \l__row_lower_dim
\dim_new:N \l__row_vpace_dim

\cs_new_protected:Npn \__tblr_get_row_inner_height_depth:nNNNN #1 #2 #3 #4 #5
  {
    \dim_set:Nn #4
      { \__tblr_data_item:nen { row } {#1} { abovesep } }
    \dim_set:Nn #5
      { \__tblr_data_item:nen { row } {#1} { belowsep } }
    \dim_set:Nn \l__row_upper_dim
      {  \__tblr_data_item:nen { row } {#1} { @row-upper } }
    \dim_set:Nn \l__row_lower_dim
      {  \__tblr_data_item:nen { row } {#1} { @row-lower } }
    \dim_set:Nn \l__row_vpace_dim
      {
        ( \__tblr_data_item:nen { row } {#1} { @row-height }
          - \l__row_upper_dim - \l__row_lower_dim ) / 2
      }
    \dim_set:Nn #2 { #4 + \l__row_vpace_dim + \l__row_upper_dim }
    \dim_set:Nn #3 { \l__row_lower_dim + \l__row_vpace_dim + #5 }
  }
\cs_generate_variant:Nn \__tblr_get_row_inner_height_depth:nNNNN { V }

%% #1: row number, #2: column number
\cs_new_protected:Npn \__tblr_build_vline_segment:nn #1 #2
  {
    \tl_set:Nx \l__tblr_n_tl
      { \__tblr_text_item:ne { vline } { [#2] / @vline-count } }
    \tl_set:Nx \l__tblr_o_tl
      { \__tblr_text_item:ne { vline } { [#1][#2] / omit } }
```

```
    \tl_if_empty:NTF \l__tblr_o_tl
      {
        \int_compare:nNnT { \l__tblr_n_tl } > {0}
          { \__tblr_build_vline_segment_real:nn {#1} {#2} }
      }
      { \__tblr_build_vline_segment_omit:nn {#1} {#2} }
  }

%% #1: row number, #2: column number
\cs_new_protected:Npn \__tblr_build_vline_segment_omit:nn #1 #2
  {
    \tl_set:Nx \l__tblr_w_tl
      { \__tblr_text_item:ne { vline } { [#2] / @vline-width } }
    \skip_horizontal:N \l__tblr_w_tl
  }

%% #1: row number, #2: column number
%% We make every vline segment intersect with first hline below
%% to remove gaps in vlines around multirow cells
\cs_new_protected:Npn \__tblr_build_vline_segment_real:nn #1 #2
  {
    \tl_set:Nx \l__tblr_s_tl
      { \__tblr_prop_item:ne { table } { rulesep } }
    \tl_set:Nx \l__tblr_b_tl
      {
        \__tblr_text_item:ne { hline }
          { [\int_eval:n{#1 + 1}](1) / @hline-height }
      }
    \tl_if_empty:NT \l__tblr_b_tl { \tl_set:Nn \l__tblr_b_tl { 0pt } }
    \hbox_set:Nn \l__tblr_a_box
      {
        \int_step_inline:nn { \l__tblr_n_tl }
          {
            \tl_set:Nx \l__tblr_w_tl
              { \__tblr_text_item:ne { vline } { [#2](##1) / @vline-width } }
            \vbox_set_to_ht:Nnn \l__tblr_b_box
              { \dim_eval:n { \l__tblr_row_ht_dim + \l__tblr_row_dp_dim } }
              {
                \tl_set:Nx \l__tblr_f_tl
                  { \__tblr_text_item:ne { vline } { [#1][#2](##1) / fg } }
                \tl_if_empty:NF \l__tblr_f_tl { \color{\l__tblr_f_tl} }
                \__tblr_get_vline_segment_child:nnnxx {#1} {#2} {##1}
                  { \dim_eval:n { \l__tblr_row_ht_dim } }
                  { \dim_eval:n { \l__tblr_row_dp_dim + \l__tblr_b_tl } }
                \skip_vertical:n {  - \l__tblr_b_tl }
              }
            \box_set_wd:Nn \l__tblr_b_box { \l__tblr_w_tl }
            \box_use:N \l__tblr_b_box
            \skip_horizontal:n { \l__tblr_s_tl }
          }
        \skip_horizontal:n { - \l__tblr_s_tl }
      }
    \vbox_set:Nn \l__tblr_c_box { \box_use:N \l__tblr_a_box }
    \box_set_ht:Nn \l__tblr_c_box { \dim_use:N \l__tblr_row_ht_dim }
    \box_set_dp:Nn \l__tblr_c_box { \dim_use:N \l__tblr_row_dp_dim }
    \box_use:N \l__tblr_c_box
  }
```

```
\tl_new:N \l__tblr_cell_rowspan_tl
\tl_new:N \l__tblr_cell_colspan_tl
\dim_new:N \l__tblr_cell_wd_dim
\dim_new:N \l__tblr_cell_ht_dim

\cs_new_protected:Npn \__tblr_build_cell:NN #1 #2
  {
    \int_set:Nn \c@rownum {#1}
    \int_set:Nn \c@colnum {#2}
    \group_begin:
    \tl_set:Nx \l__tblr_w_tl
      { \__tblr_data_item:nen { column } {#2} { @col-width } }
    \tl_set:Nx \l__tblr_h_tl
      { \__tblr_data_item:nen { row } {#1} { @row-height } }
    \tl_set:Nx \l__tblr_x_tl
      { \__tblr_data_item:nen { column } {#2} { leftsep} }
    \tl_set:Nx \l__tblr_y_tl
      { \__tblr_data_item:nen { column } {#2} { rightsep } }
    \tl_set:Nx \l__tblr_cell_colspan_tl
      { \__tblr_data_item:neen { cell } {#1} {#2} { colspan } }
    \int_compare:nNnTF { \l__tblr_cell_colspan_tl } < {2}
      { \dim_set:Nn \l__tblr_cell_wd_dim { \l__tblr_w_tl } }
      {
        \__tblr_get_span_horizontal_sizes:NNNNN #1 #2
          \l__tblr_o_dim \l__tblr_cell_wd_dim \l__tblr_q_dim
      }
    \tl_set:Nx \l__tblr_cell_rowspan_tl
      { \__tblr_data_item:neen { cell } {#1} {#2} { rowspan } }
    \int_compare:nNnTF { \l__tblr_cell_rowspan_tl } < {2}
      { \dim_set:Nn \l__tblr_cell_ht_dim { \l__tblr_h_tl } }
      {
        \__tblr_get_span_vertical_sizes:NNNNN #1 #2
          \l__tblr_r_dim \l__tblr_cell_ht_dim \l__tblr_t_dim
      }
    \__tblr_get_cell_alignments:nn {#1} {#2}
    \__tblr_build_cell_background:NN #1 #2
    \__tblr_build_cell_content:NN #1 #2
    \group_end:
  }

\cs_new_protected:Npn \__tblr_build_cell_content:NN #1 #2
  {
    \hbox_set_to_wd:Nnn \l__tblr_a_box { \l__tblr_cell_wd_dim }
      {
        \tl_if_eq:NnF \g__tblr_cell_halign_tl {l} { \hfil }
        \__tblr_get_cell_text:nn {#1} {#2}
        \tl_if_eq:NnF \g__tblr_cell_halign_tl {r} { \hfil }
      }
    \vbox_set_to_ht:Nnn \l__tblr_b_box { \l__tblr_cell_ht_dim }
      {
        \tl_case:Nn \g__tblr_cell_valign_tl
          {
            \c__tblr_valign_m_tl
              {
                \vfil
                \int_compare:nNnT { \l__tblr_cell_rowspan_tl } < {2}
                  {
                    \box_set_ht:Nn \l__tblr_a_box
```

```
                    { \__tblr_data_item:nen { row } {#1} { @row-upper } }
                  \box_set_dp:Nn \l__tblr_a_box
                    { \__tblr_data_item:nen { row } {#1} { @row-lower } }
                }
              \box_use:N \l__tblr_a_box
              \vfil
            }
          \c__tblr_valign_h_tl
            {
              \box_set_ht:Nn \l__tblr_a_box
                { \__tblr_data_item:nen { row } {#1} { @row-head } }
              \box_use:N \l__tblr_a_box
              \vfil
            }
          \c__tblr_valign_f_tl
            {
              \vfil
              \int_compare:nNnTF { \l__tblr_cell_rowspan_tl } < {2}
                {
                  \box_set_dp:Nn \l__tblr_a_box
                    { \__tblr_data_item:nen { row } {#1} { @row-foot } }
                }
                {
                  \box_set_dp:Nn \l__tblr_a_box
                    {
                      \__tblr_data_item:nen
                        { row }
                        { \int_eval:n { #1 + \l__tblr_cell_rowspan_tl - 1 } }
                        { @row-foot }
                    }
                }
              \box_use:N \l__tblr_a_box
            }
        }
      \hrule height ~ 0pt %% zero depth
    }
  \vbox_set_to_ht:Nnn \l__tblr_c_box
    { \l__tblr_row_ht_dim - \l__tblr_row_abovesep_dim }
    {
      \box_use:N \l__tblr_b_box
      \vss
    }
  \skip_horizontal:n { \l__tblr_x_tl }
  \box_use:N \l__tblr_c_box
  \skip_horizontal:n { \l__tblr_y_tl - \l__tblr_cell_wd_dim + \l__tblr_w_tl }
}

\cs_new_protected:Npn \__tblr_build_cell_background:NN #1 #2
{
  \int_compare:nNnT { \__tblr_data_item:neen { cell } {#1} {#2} { omit } } = {0}
    {
      \group_begin:
      \tl_set:Nx \l__tblr_b_tl
        { \__tblr_data_item:neen { cell } {#1} {#2} { background } }
      \tl_if_empty:NF \l__tblr_b_tl
        {
          \__tblr_get_cell_background_width:NNN #1 #2 \l_tmpa_dim
          \__tblr_get_cell_background_depth:NNN #1 #2 \l_tmpb_dim
```

```
            \__tblr_build_cell_background:nnnn
              { \dim_use:N \l_tmpa_dim }
              { \l__tblr_row_ht_dim }
              { \dim_use:N \l_tmpb_dim }
              { \l__tblr_b_tl }
          }
        \group_end:
      }
  }

%% #1: row number; #2: column number; #3 resulting dimension
\cs_new_protected:Npn \__tblr_get_cell_background_width:NNN #1 #2 #3
  {
    \int_compare:nNnTF { \l__tblr_cell_colspan_tl } < {2}
      { \dim_set:Nn #3 { \l__tblr_x_tl + \l__tblr_w_tl + \l__tblr_y_tl } }
      {
        \dim_set:Nn #3 { \l__tblr_o_dim + \l__tblr_cell_wd_dim + \l__tblr_q_dim }
      }
  }

%% #1: row number; #2: column number; #3 resulting dimension
\cs_new_protected:Npn \__tblr_get_cell_background_depth:NNN #1 #2 #3
  {
    \int_compare:nNnTF { \l__tblr_cell_rowspan_tl } < {2}
      { \dim_set_eq:NN #3 \l__tblr_row_dp_dim }
      {
        \dim_set:Nn #3
          {
            \l__tblr_r_dim + \l__tblr_cell_ht_dim
                           + \l__tblr_t_dim - \l__tblr_row_ht_dim
          }
      }
  }

%% #1: width, #2: height, #3: depth, #4: color
\cs_new_protected:Npn \__tblr_build_cell_background:nnnn #1 #2 #3 #4
  {
    \hbox_set:Nn \l__tblr_a_box
      {
        \color {#4}
        \vrule width ~ #1 ~ height ~ #2 ~ depth ~ #3
      }
    \box_set_dp:Nn \l__tblr_a_box { 0pt }
    \box_use:N \l__tblr_a_box
    \skip_horizontal:n { - #1 }
  }

%% #1: row number; #2: column number; #3: dimen register for rowsep above.
%% #4: dimen register for total height; #5: dimen register for rowsep below.
%% We can use \l__tblr_row_item_skip_size_prop which was made before
\cs_new_protected:Npn \__tblr_get_span_vertical_sizes:NNNNN #1 #2 #3 #4 #5
  {
    \dim_set:Nn #3
      { \__tblr_data_item:nen { row } {#1} { abovesep } }
    \dim_zero:N #4
    \int_step_inline:nnn { #1 } { #1 + \l__tblr_cell_rowspan_tl - 2 }
      {
```

```
        \dim_add:Nn #4
          { \prop_item:Ne \l__tblr_row_item_skip_size_prop { itemskip[##1] } }
      }
    \dim_add:Nn #4
      {
        \prop_item:Ne \l__tblr_row_item_skip_size_prop
          { item[\int_eval:n { #1 + \l__tblr_cell_rowspan_tl - 1 }] }
      }
    \dim_set:Nn #5
      {
        \__tblr_data_item:nen { row }
          { \int_eval:n { #1 + \l__tblr_cell_rowspan_tl - 1 } } { belowsep }
      }
    %\tl_log:x { cell[#1][#2] ~:~ \dim_use:N #3, \dim_use:N #4, \dim_use:N #5 }
  }

%% #1: row number; #2: column number; #3: dimen register for colsep left.
%% #4: dimen register for total width; #5: dimen register for colsep right.
%% We can use \l__tblr_col_item_skip_size_prop which was made before
%% But when hspan=minimal, there are no itemskip in the prop list.
%% Therefore we need to calculate them from the sizes of items and skips
\cs_new_protected:Npn \__tblr_get_span_horizontal_sizes:NNNNN #1 #2 #3 #4 #5
  {
    \dim_set:Nn #3
      { \__tblr_data_item:nen { column } {#2} { leftsep } }
    \dim_zero:N #4
    \int_step_inline:nnn { #2 } { #2 + \l__tblr_cell_colspan_tl - 2 }
      {
        \dim_add:Nn #4
          { \prop_item:Ne \l__tblr_col_item_skip_size_prop { item[##1] } }
        \dim_add:Nn #4
          {
            \prop_item:Ne \l__tblr_col_item_skip_size_prop
              { skip[\int_eval:n { ##1 + 1 }] }
          }
      }
    \dim_add:Nn #4
      {
        \prop_item:Ne \l__tblr_col_item_skip_size_prop
          { item[\int_eval:n { #2 + \l__tblr_cell_colspan_tl - 1 }] }
      }
    \dim_set:Nn #5
      {
        \__tblr_data_item:nen { column }
          { \int_eval:n {#2 + \l__tblr_cell_colspan_tl - 1} } { rightsep }
      }
    %\tl_log:x { cell[#1][#2] ~:~ \dim_use:N #3, \dim_use:N #4, \dim_use:N #5 }
  }

%%% ------------------------------------------------------------
%%  \section{Tracing Tabularray}
%%% ------------------------------------------------------------

\NewDocumentCommand \SetTabularrayTracing { m }
  {
    \keys_set:nn { tblr-set-tracing } {#1}
  }
```

```
\cs_new_eq:NN \SetTblrTracing \SetTabularrayTracing

\bool_new:N \g__tblr_tracing_text_bool
\bool_new:N \g__tblr_tracing_command_bool
\bool_new:N \g__tblr_tracing_table_bool
\bool_new:N \g__tblr_tracing_column_bool
\bool_new:N \g__tblr_tracing_row_bool
\bool_new:N \g__tblr_tracing_cell_bool
\bool_new:N \g__tblr_tracing_vline_bool
\bool_new:N \g__tblr_tracing_hline_bool
\bool_new:N \g__tblr_tracing_colspec_bool
\bool_new:N \g__tblr_tracing_rowspec_bool
\bool_new:N \g__tblr_tracing_target_bool
\bool_new:N \g__tblr_tracing_cellspan_bool
\bool_new:N \g__tblr_tracing_intarray_bool

\keys_define:nn { tblr-set-tracing }
  {
    +text .code:n = \bool_gset_true:N \g__tblr_tracing_text_bool,
    -text .code:n = \bool_gset_false:N \g__tblr_tracing_text_bool,
    +command .code:n = \bool_gset_true:N \g__tblr_tracing_command_bool,
    -command .code:n = \bool_gset_false:N \g__tblr_tracing_command_bool,
    +table .code:n = \bool_gset_true:N \g__tblr_tracing_table_bool,
    -table .code:n = \bool_gset_false:N \g__tblr_tracing_table_bool,
    +column .code:n = \bool_gset_true:N \g__tblr_tracing_column_bool,
    -column .code:n = \bool_gset_false:N \g__tblr_tracing_column_bool,
    +row .code:n = \bool_gset_true:N \g__tblr_tracing_row_bool,
    -row .code:n = \bool_gset_false:N \g__tblr_tracing_row_bool,
    +cell .code:n = \bool_gset_true:N \g__tblr_tracing_cell_bool,
    -cell .code:n = \bool_gset_false:N \g__tblr_tracing_cell_bool,
    +vline .code:n = \bool_gset_true:N \g__tblr_tracing_vline_bool,
    -vline .code:n = \bool_gset_false:N \g__tblr_tracing_vline_bool,
    +hline .code:n = \bool_gset_true:N \g__tblr_tracing_hline_bool,
    -hline .code:n = \bool_gset_false:N \g__tblr_tracing_hline_bool,
    +colspec .code:n = \bool_gset_true:N \g__tblr_tracing_colspec_bool,
    -colspec .code:n = \bool_gset_false:N \g__tblr_tracing_colspec_bool,
    +rowspec .code:n = \bool_gset_true:N \g__tblr_tracing_rowspec_bool,
    -rowspec .code:n = \bool_gset_false:N \g__tblr_tracing_rowspec_bool,
    +target .code:n = \bool_gset_true:N \g__tblr_tracing_target_bool,
    -target .code:n = \bool_gset_false:N \g__tblr_tracing_target_bool,
    +cellspan .code:n = \bool_gset_true:N \g__tblr_tracing_cellspan_bool,
    -cellspan .code:n = \bool_gset_false:N \g__tblr_tracing_cellspan_bool,
    +intarray .code:n = \bool_gset_true:N \g__tblr_tracing_intarray_bool,
    -intarray .code:n = \bool_gset_false:N \g__tblr_tracing_intarray_bool,
    all .code:n = \__tblr_enable_all_tracings:,
    none .code:n = \__tblr_disable_all_tracings:,
  }

\cs_new_protected_nopar:Npn \__tblr_enable_all_tracings:
  {
    \bool_gset_true:N \g__tblr_tracing_text_bool
    \bool_gset_true:N \g__tblr_tracing_command_bool
    \bool_gset_true:N \g__tblr_tracing_table_bool
    \bool_gset_true:N \g__tblr_tracing_column_bool
    \bool_gset_true:N \g__tblr_tracing_row_bool
    \bool_gset_true:N \g__tblr_tracing_cell_bool
    \bool_gset_true:N \g__tblr_tracing_vline_bool
```

```
    \bool_gset_true:N \g__tblr_tracing_hline_bool
    \bool_gset_true:N \g__tblr_tracing_colspec_bool
    \bool_gset_true:N \g__tblr_tracing_rowspec_bool
    \bool_gset_true:N \g__tblr_tracing_target_bool
    \bool_gset_true:N \g__tblr_tracing_cellspan_bool
    \bool_gset_true:N \g__tblr_tracing_intarray_bool
  }

\cs_new_protected_nopar:Npn \__tblr_disable_all_tracings:
  {
    \bool_gset_false:N \g__tblr_tracing_text_bool
    \bool_gset_false:N \g__tblr_tracing_command_bool
    \bool_gset_false:N \g__tblr_tracing_table_bool
    \bool_gset_false:N \g__tblr_tracing_column_bool
    \bool_gset_false:N \g__tblr_tracing_row_bool
    \bool_gset_false:N \g__tblr_tracing_cell_bool
    \bool_gset_false:N \g__tblr_tracing_vline_bool
    \bool_gset_false:N \g__tblr_tracing_hline_bool
    \bool_gset_false:N \g__tblr_tracing_colspec_bool
    \bool_gset_false:N \g__tblr_tracing_rowspec_bool
    \bool_gset_false:N \g__tblr_tracing_target_bool
    \bool_gset_false:N \g__tblr_tracing_cellspan_bool
    \bool_gset_false:N \g__tblr_tracing_intarray_bool
  }

\NewDocumentCommand \LogTabularrayTracing { m }
  {
    \keys_set:nn { tblr-log-tracing } {#1}
  }
\cs_new_eq:NN \LogTblrTracing \LogTabularrayTracing

\keys_define:nn { tblr-log-tracing }
  {
    unknown .code:n = \__tblr_log_tracing:N \l_keys_key_str
  }

\cs_new_protected:Npn \__tblr_log_tracing:N #1
  {
    \bool_if:cT { g__tblr_tracing_ #1 _bool }
      { \cs:w __tblr_log_tracing _ #1 : \cs_end: }
  }

\cs_new_protected:Npn \__tblr_log_tracing_text:
  {
    \__tblr_text_log:n { text }
  }

\cs_new_protected:Npn \__tblr_log_tracing_command:
  {
    \__tblr_prop_log:n { command }
  }

\cs_new_protected:Npn \__tblr_log_tracing_table:
  {
    \__tblr_prop_log:n { table }
  }
```

```
\cs_new_protected:Npn \__tblr_log_tracing_column:
  {
    \__tblr_data_log:n { column }
  }

\cs_new_protected:Npn \__tblr_log_tracing_row:
  {
    \__tblr_data_log:n { row }
  }

\cs_new_protected:Npn \__tblr_log_tracing_cell:
  {
    \__tblr_data_log:n { cell }
  }

\cs_new_protected:Npn \__tblr_log_tracing_vline:
  {
    \__tblr_text_log:n { vline }
  }

\cs_new_protected:Npn \__tblr_log_tracing_hline:
  {
    \__tblr_text_log:n { hline }
  }

\cs_new_protected:Npn \__tblr_log_tracing_colspec:
  {
    \tl_if_eq:NnT \g__tblr_column_or_row_tl { column }
      { \tl_log:N \g__tblr_expanded_colrow_spec_tl }
  }

\cs_new_protected:Npn \__tblr_log_tracing_rowspec:
  {
    \tl_if_eq:NnT \g__tblr_column_or_row_tl { row }
      { \tl_log:N \g__tblr_expanded_colrow_spec_tl }
  }

\cs_new_protected:Npn \__tblr_log_tracing_target:
  {
    \dim_log:N \l__column_target_dim
    \prop_log:N \l__column_coefficient_prop
    \prop_log:N \l__column_natural_width_prop
    \prop_log:N \l__column_computed_width_prop
  }

\cs_new_protected:Npn \__tblr_log_tracing_cellspan:
  {
    \prop_log:N \l__tblr_col_item_skip_size_prop
    \prop_log:N \l__tblr_col_span_size_prop
    \prop_log:N \l__tblr_row_item_skip_size_prop
    \prop_log:N \l__tblr_row_span_size_prop
    \prop_log:N \l__tblr_row_span_to_row_prop
  }

\cs_new_protected:Npn \__tblr_do_if_tracing:nn #1 #2
```

```
  {
    \bool_if:cT { g__tblr_tracing_ #1 _bool } {#2}
  }

\ExplSyntaxOff
```