The karnaughmap package*

Michael Vonbun
m.vonbun@gmail.com

April 20, 2015

1 Introduction

This package uses TikZ to typeset Karnaugh maps in an easy way. Though the map size is restricted to four variables (as for v2.0), it is easier and customizable compared to karnaugh.tex.

2 Installation

Install this package like any other LaTeX package.

The end of this file contains the installation file (.ins) code in case the provided ins-file got lost.

For style creation run `latex karnaughmap.ins`. For doc/manual creation run `latex karnaughmap.dtx` afterwards.

3 Dependencies

As of karnaughmap v2.0 this package depends on

- tikz
- xkeyval
- ifthen
- xstring

*This document corresponds to karnaughmap v2.0, dated 2015/04/20.

4 Usage

4.1 Karnaugh Map Macros

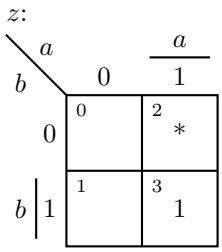
`\karnaughmap` `\karnaughmap[`⟨*key=value*⟩`]{`⟨*entries*⟩`}`

Use this macro inside of a `tikzpicture`-environment to typeset a Karnaugh map of a corresponding truth table.

Example Assume the truth table with inputs a and b and output z is

a	b	z
0	0	0
0	1	0
1	0	*
1	1	1

Then, the corresponding Karnaugh map with default layout is:

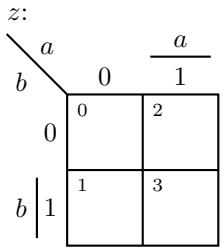


Which was typeset using:

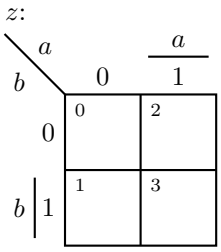
```
\begin{tikzpicture}[thick]
   \karnaughmap{00*1}
\end{tikzpicture}
```

To produce empty maps, there are two possibilities. First is to use the key `defaultmap` described later and omit the mandatory input, the second is by providing a scalar mandatory argument that is equal the number of cells desired, i.e. 4,8, or 16.

`\karnaughmap{4}`

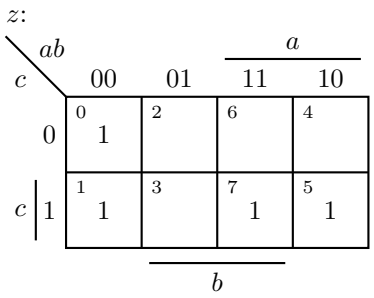


`\karnaughmap[defaultmap=4]{}`



Note that it is also possible to provide the input string with blanks:

```
\begin{tikzpicture}[thick]
   \karnaughmap{1100 0101}
\end{tikzpicture}
```

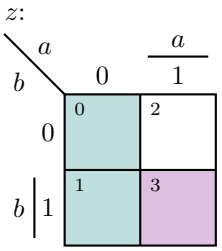


This is especially useful when larger maps are typeset as it increases readability of the LaTeX 2_ε code.

`\karnaughmapcolorfield` `\karnaughmapcolorfield[`⟨*key=value*⟩`]{`⟨*map size*⟩`}{`⟨*field string*⟩`}{`⟨*color*⟩`}`

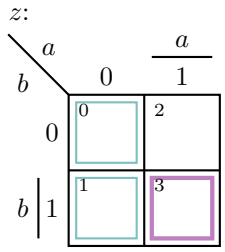
Use this macro inside of a `tikzpicture`-environment to color a Karnaugh map of size `map size`. Use hexadecimal values to address a certain field of the map in the `field string`.

```
\begin{tikzpicture}[thick]
\karnaughmapcolorfield{2}{01}{teal!50}%
\karnaughmapcolorfield{2}{3}{violet!50}%
\karnaughmap{4}
\end{tikzpicture}
```



The keys for this macro allow for two different styles: *filled* `fill` and *outlined* `outline`. The default is filled. In addition, TikZ keys are forwarded to the `\draw` or `\fill` command, allowing a good adjustment of styles.

```
\begin{tikzpicture}[thick]
\karnaughmapcolorfield[outline]{2}{01}{teal!50}%
\karnaughmapcolorfield[outline,ultra thick]{2}{3}{violet!50}%
\karnaughmap{4}
\end{tikzpicture}
```



4.2 Style Customizations

For style customization a key-value system is provided.

\setkarnaughmap `\setkarnaughmap[`⟨*key=value*⟩`]` This is the interface function for setting the keys. To restore the defaults call `\setkarnaughmap`.

The defaults are

```
\setkarnaughmap[omitzeros, defaultmap=16,
   function=z, variables=abcd, dontcare=*].
```

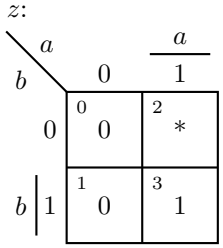
Key-Value List The following table is an overview over all options provided. The exact behavior and description of the keys are described below.

Key	Prefix	Family	Type	Default
omitzeros	KV	karnaughmap	boolean	true
omitones	KV	karnaughmap	boolean	true
omitdontcares	KV	karnaughmap	boolean	true
omitbinaries	KV	karnaughmap	boolean	true
omitvariables	KV	karnaughmap	boolean	true
omitidx	KV	karnaughmap	boolean	true
binaryidx	KV	karnaughmap	boolean	true
omitnegated	KV	karnaughmap	boolean	true
outline	KV	karnaughmap	boolean	true
fill	KV	karnaughmap	boolean	true
opacity	KV	karnaughmap	command	0.5
defaultmap	KV	karnaughmap	command	16
function	KV	karnaughmap	command	z
variables	KV	karnaughmap	command	abcd
dontcare	KV	karnaughmap	command	*
color	KV	karnaughmap	ordinary	black
draw	KV	karnaughmap	ordinary	karnaughmapColor
labelcolor	KV	karnaughmap	ordinary	karnaughmapColor
xshift	KV	karnaughmap	command	0
yshift	KV	karnaughmap	command	0

Selective Entry Printing The following keys control the printing of different types of Karnaugh map cell entries.

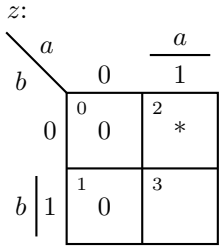
`omitzeros` `omitzeros[`⟨*=**true**/false*⟩`]` Toggle between printing and omitting zero-valued entries in the Karnaugh map.

```
\karnaughmap[omitzeros=false]{00*1}
```



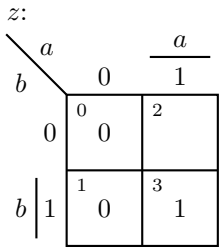
`omitones` `omitones[`⟨*=true/**false***⟩`]` Toggle between printing and omitting one-valued entries in the Karnaugh map.

```
\karnaughmap[omitzeros=false,omitones]{00*1}
```



`omitdontcares` `omitdontcares[`⟨*=true/**false***⟩`]` Toggle between printing and omitting don't care entries in the Karnaugh map.

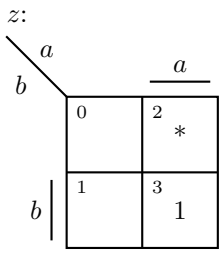
```
\karnaughmap[omitzeros=false,omitdontcares]{00*1}
```



Karnaugh Map Style The following keys control the printing of different types of Karnaugh map labels.

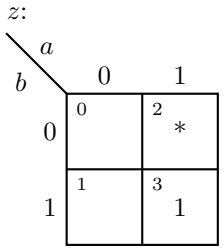
`omitbinaries` `omitbinaries[`⟨*true/**false***⟩`]` Toggle between printing and omitting the binary encoding on top and left of the Karnaugh map.

```
\karnaughmap[omitbinaries]{00*1}
```



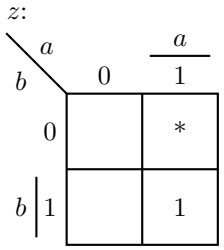
`omitvariables` `omitvariables[`⟨=*true*/***false***⟩`]` Toggle between printing and omitting the variable labeling of rows and columns around the Karnaugh map.

```
\karnaughmap[omitvariables]{00*1}
```



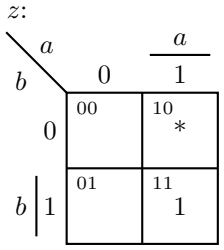
`omitidx` `omitidx[`⟨=*true*/***false***⟩`]` Toggle between printing and omitting index values besides the cell entries in the Karnaugh map.

```
\karnaughmap[omitidx]{00*1}
```



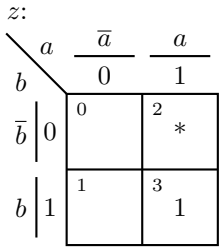
`binaryidx` `binaryidx[`⟨=*true*/***false***⟩`]` Toggle between decimal and binary indices.

```
\karnaughmap[binaryidx]{00*1}
```



`omitnegated` `omitnegated[`⟨=***true***/*false*⟩`]` Toggle between typesetting the negated in addition to the non-negated variable labels.

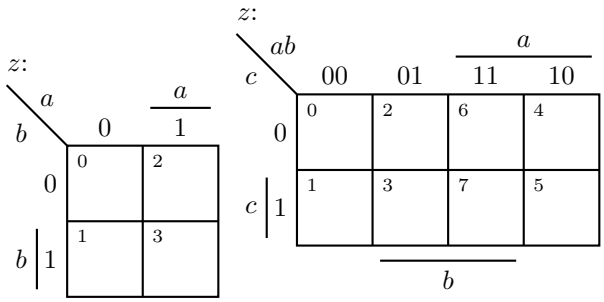
```
\karnaughmap[omitnegated=false]{00*1}
```



Map Customizations The following keys can be used to customize the Karnaugh maps in various ways.

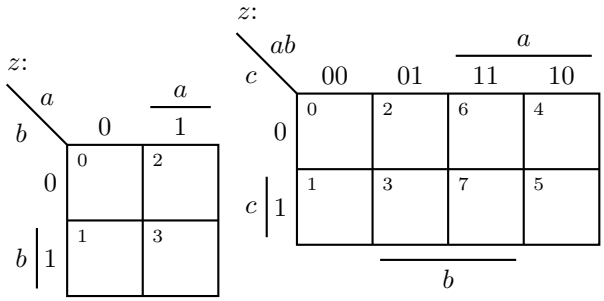
`defaultmap` `defaultmap[`⟨*=**16***⟩`]` Set the default map size (4,8, or 16), i.e. this map size is used if the mandatory argument of `\karnaughmap{}` is empty. Thus, empty maps can be produced.

```
\karnaughmap[defaultmap=4]{}, \karnaughmap[defaultmap=8]{}
```



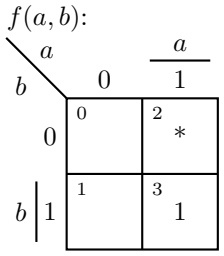
Note that the same functionality can be achieved by providing a scalar mandatory input

```
\karnaughmap{4}, \karnaughmap[defaultmap=8]{}
```

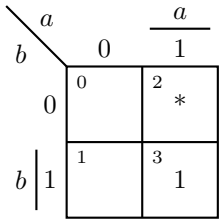


`function` `function[`⟨*=**z***⟩`]` Set the function label used to label the Karnaugh map.

```
\karnaughmap[function={f(a,b)}]{00*1}
```

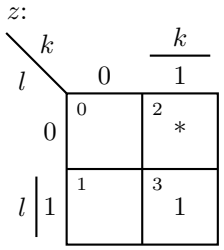



```
\karnaughmap[function=]{00*1}
```

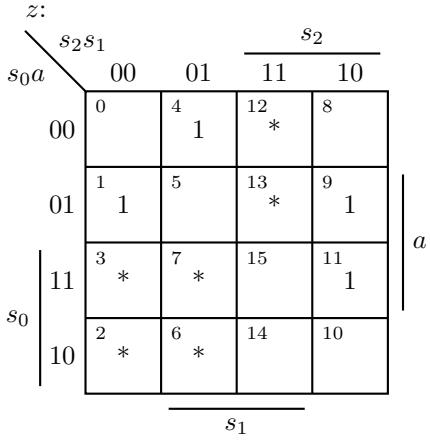


`variables` `variables[`⟨=***abcd***⟩`]` Set the variable names used in the Karnaugh map. Note that you have to provide a single string of variables or variables embraced by curly braces.

```
\karnaughmap[variables=kl]{00*1}
```

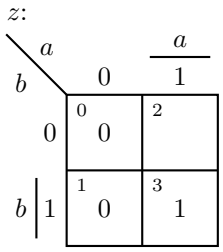



```
\karnaughmap[variables={{s_2}{s_1}{s_0}{a}}]{01** 10** 0101 **00}
```



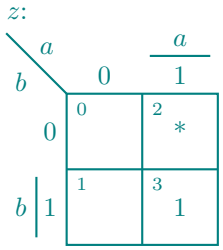
`dontcare` `dontcare[⟨=*⟩]` Set the don't care character. This is only needed if you provide the `omitdontcares` key and use don't care characters other than `*` in the `\karnaughmap{x}` inputstring.

```
\karnaughmap[dontcare=X,omitdontcares,omitzeros=false]{00X1}
```



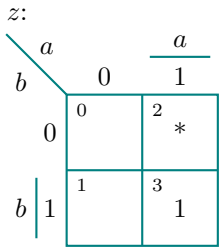
`color` `color[⟨=black⟩]` Set the color of the *entire* Karnaugh map, including labels, entries, etc.

```
\karnaughmap[color=teal]{00*1}
```



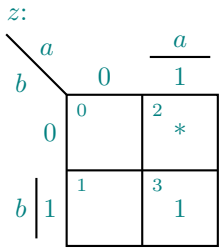
`draw` `draw[⟨=same as color⟩]` Set the color of the Karnaugh map *only*, i.e. the lines that are drawn. The labels are not affected.

```
\karnaughmap[draw=teal]{00*1}
```



`labelcolor` `labelcolor[⟨=same as color⟩]` Set the color of the labels of the Karnaugh map *only*. The lines are not affected.

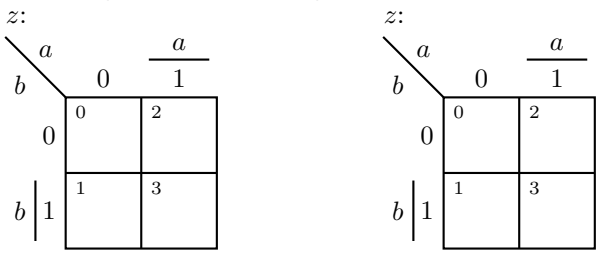
```
\karnaughmap[labelcolor=teal]{00*1}
```



Multiple Karnaugh Maps in one TikZ Picture To typeset more than one Karnaugh map into one TikZ Picture, use `xshift` and `yshift`, just as usual n TikZ.

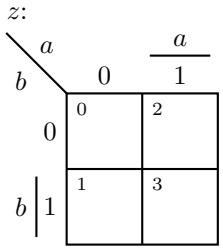
`xshift` `xshift[⟨=0⟩]` Shift the Karnaugh map in x-direction within the TikZ picture.

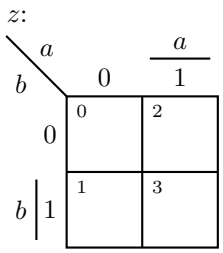
```
\karnaughmap{4}\karnaughmap[xshift=5cm]{4}
```



`yshift` `yshift[⟨=0⟩]` Shift the Karnaugh map in x-direction within the TikZ picture.

```
\karnaughmap{4}\karnaughmap[yshift=5cm]{4}
```

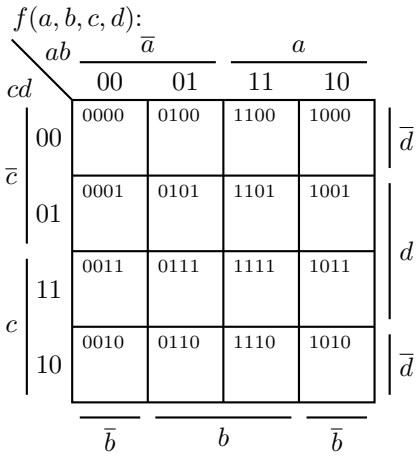




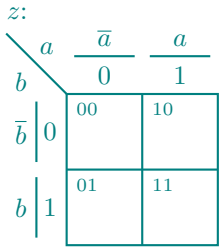
5 Examples

5.1 Basic Usage Examples:

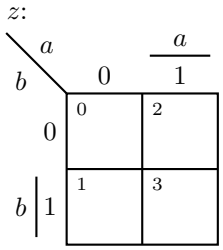
```
\karnaughmap[defaultmap=16,binaryidx,omitnegated=false]{}
```




```
\karnaughmap[defaultmap=4,binaryidx,omitnegated=false,color=teal]{}
```



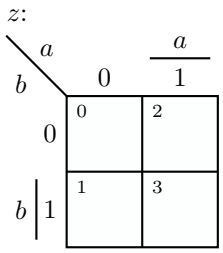

```
\karnaughmap{4}
```



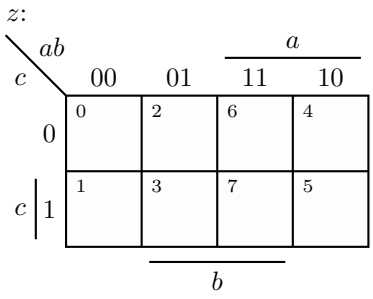
5.2 Field coloring

```
\begin{tikzpicture}[thick]
\foreach \x [count=\xi] in {0,1,2,3} {%
   \pgfmathparse{-20+30*\xi}%
```

```
   \karnaughmapcolorfield{2}{\x}{teal!\pgfmathresult}%
}%
\karnaughmap{2}
\end{tikzpicture}
```

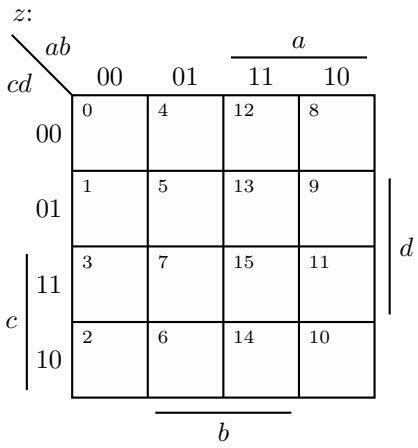



```
\begin{tikzpicture}[thick]
\foreach \x [count=\xi] in {0,1,2,3} {%
   \pgfmathparse{-20+30*\xi}%
   \karnaughmapcolorfield{3}{\x}{teal!\pgfmathresult}%
}%
\foreach \x [count=\xi] in {4,5,6,7} {%
   \pgfmathparse{-20+30*\xi}%
   \karnaughmapcolorfield{3}{\x}{purple!\pgfmathresult}%
}%
\karnaughmap{8}
\end{tikzpicture}
```




```
\begin{tikzpicture}[thick]
\foreach \x [count=\xi] in {0,1,2,3} {%
   \pgfmathparse{-20+30*\xi}%
   \karnaughmapcolorfield{4}{\x}{teal!\pgfmathresult}%
}%
\foreach \x [count=\xi] in {4,5,6,7} {%
   \pgfmathparse{-20+30*\xi}%
   \karnaughmapcolorfield{4}{\x}{purple!\pgfmathresult}%
```

```
}%
\foreach \x [count=\xi] in {8,9,a,b} {%
   \pgfmathparse{-20+30*\xi}%
   \karnaughmapcolorfield{4}{\x}{olive!\pgfmathresult}%
}%
\foreach \x [count=\xi] in {c,d,e,f} {%
   \pgfmathparse{-20+30*\xi}%
   \karnaughmapcolorfield{4}{\x}{violet!\pgfmathresult}%
}%
\karnaughmap[defaultmap=16]{}%
\end{tikzpicture}
```



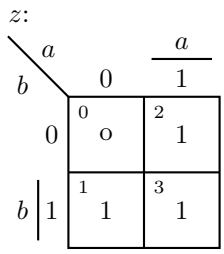
6 Field Allocation Reference

This package provides maps with full control up to *4* variables, which is equivalent to a map size of *16* fields.
For bigger maps, the package provides just basic typesetting, i.e. just the column and row encoding and the, possibly filled matrix.
However, the biggest map supported yet has 256 fields.

The reference can be read as follows: the first row contains the value of the rows, the columns are then likewise enumerated. By this procedure, you can check if everything went right (debugging) and got an easy reference.

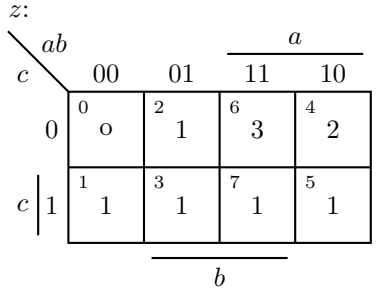
6.1 2 Variables – 4 Fields

```
\begin{tikzpicture}[thick]
  \karnaughmap[variables=ab]{o1 11 }
\end{tikzpicture}
```



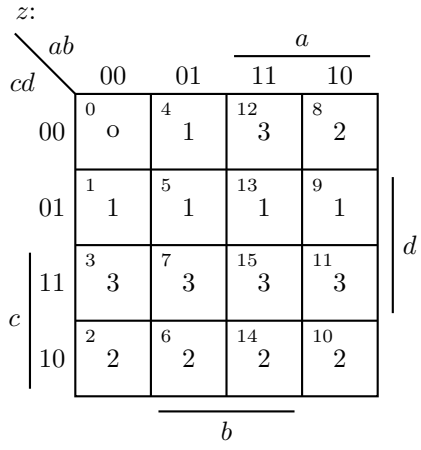
6.2 3 Variables – 8 Fields

```
\begin{tikzpicture}[thick]
  \karnaughmap[variables=abc]{o1 11 21 31 }
\end{tikzpicture}
```



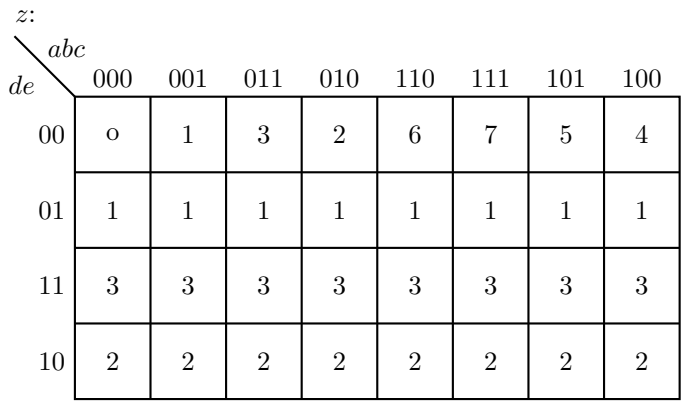
6.3 4 Variables – 16 Fields

```
\begin{tikzpicture}[thick]
  \karnaughmap[variables=abcd]{o123 1123 2123 3123 }
\end{tikzpicture}
```



6.4 5 Variables – 32 Fields

```
\begin{tikzpicture}[thick]
  \karnaughmap[variables=abcde]{0000 0001 0010 0011 0100 0101 0110 1111}
\end{tikzpicture}
```



6.5 6 Variables – 64 Fields

```
\begin{tikzpicture}[thick]
  \karnaughmap[variables=abcdef]{o1234567 11234567 21234567 31234567
                                 41234567 51234567 61234567 71234567 }
\end{tikzpicture}
```

z: abc / def	000	001	011	010	110	111	101	100
000	o	1	3	2	6	7	5	4
001	1	1	1	1	1	1	1	1
011	3	3	3	3	3	3	3	3
010	2	2	2	2	2	2	2	2
110	6	6	6	6	6	6	6	6
111	7	7	7	7	7	7	7	7
101	5	5	5	5	5	5	5	5
100	4	4	4	4	4	4	4	4

6.6 7 Variables – 128 Fields

```
\begin{tikzpicture}[thick]
  \karnaughmap[variables=abcdefg]{o1234567 11234567 21234567 31234567
                                  41234567 51234567 61234567 71234567
                                  81234567 91234567 a1234567 b1234567
                                  c1234567 d1234567 e1234567 f1234567 }
\end{tikzpicture}
```

z: $abcd$ / efg	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101	1111	1110	1010	1011	1001	1000
000	o	1	3	2	6	7	5	4	c	d	f	e	a	b	9	8
001	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
011	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
010	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
110	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
111	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
101	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
100	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4

6.7 8 Variables – 256 Fields

```
\begin{tikzpicture}[thick]
  \karnaughmap[variables=abcdefg]{o123456789abcdef 1123456789abcdef
                                  2123456789abcdef 3123456789abcdef
                                  4123456789abcdef 5123456789abcdef
                                  6123456789abcdef 7123456789abcdef
                                  8123456789abcdef 9123456789abcdef
                                  a123456789abcdef b123456789abcdef
                                  c123456789abcdef d123456789abcdef
                                  e123456789abcdef f123456789abcdef }
\end{tikzpicture}
```

z: efg \ $abcd$	0000	0001	0011	0010	0110	0111	0101	0100	1100	1101	1111	1110	1010	1011	1001	1000
0000	o	1	3	2	6	7	5	4	c	d	f	e	a	b	9	8
0001	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
0011	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3	3
0010	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2
0110	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6	6
0111	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7	7
0101	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5	5
0100	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4	4
1100	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c	c
1101	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d	d
1111	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f	f
1110	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e	e
1010	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a	a
1011	b	b	b	b	b	b	b	b	b	b	b	b	b	b	b	b
1001	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9	9
1000	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8	8

7 Macros for Internal Usage

These macros are not intended to be used by the user, rather these are auxiliary functions invoked by the main macro `\karnaughmap{x}`.

Nevertheless, for documentation these macros are described here briefly.

8 Implementation

8.1 Load Dependencies

```
\RequirePackage{tikz}
\RequirePackage{xkeyval}
\RequirePackage{ifthen}
\RequirePackage{xstring}
```

8.2 Allocate Counters

```
\newcounter{karnaughmapIdxCounter}
\newcounter{karnaughmapStrCounter}
\setcounter{karnaughmapIdxCounter}{0}
\setcounter{karnaughmapStrCounter}{0}
```

8.3 Key-Val Interface

Switches Simple switches are used to control basic layout options of the Karnaugh map.

These boolean switches control if specific entries – i.e. zeros, ones, and don't cares – are printed or not (omitted).

```
\define@boolkeys{karnaughmap}{omitzeros, omitones, omitdontcares}[true]
```

These boolean switches control the layout of the Karnaugh map. The `omitbinaries` switch is used to control if binary encoding is typeset left and on top of the Karnaugh map. The `omitvariables` switch is used to control if the Karnaugh map is labeled with the variables. The `omitidx` switch controls the typesetting of decimal indices within the cells of the Karnaugh map. The `binaryidx` switch controls whether the indices are decimal or binary. The `omitnegated` switch controls whether the negated variable labels are typeset as well.

```
\define@boolkeys{karnaughmap}{omitbinaries, omitvariables, omitidx, binaryidx, omitnegated}[tru
```

These boolean switches control the behavior of the highlighting function.

```
\define@boolkey{karnaughmap}{outline}[true]{\setkeys{karnaughmap}{fill=false}}
\define@boolkeys{karnaughmap}{fill}[true]
\define@cmdkey{karnaughmap}{opacity}[0.5]{}
```

Customizations In contrast to the boolean switches these keys are used to customize the Karnaugh map.

This key controls the default size of the Karnaugh map. It is set to 16, i.e. a 4 variable map is typeset.

```
\define@cmdkey{karnaughmap}{defaultmap}[16]{}
```

This key controls the Karnaugh map function label. The default is 'z'.

```
\define@cmdkey{karnaughmap}{function}[z]{}
```

This key controls the variables. Provide the variables as a string. Maybe more control over variables will be added in a later version. The default is 'abcd'

```
\define@cmdkey{karnaughmap}{variables}[abcd]{}
```

Define the standard "'Don't Care"' character. The default is '*'.

```
\define@cmdkey{karnaughmap}{dontcare}[*]{}
```

Define the color. For desired functionality both the draw and color have to be set.

```
\define@key{karnaughmap}{color}[black]{%
  \colorlet{karnaughmapColor}{#1}%
  \colorlet{karnaughmapMapColor}{#1}}
```

Define the draw color, i.e. the color of the map itself. Default is the same as color.

```
\define@key{karnaughmap}{draw}[karnaughmapColor]{\colorlet{karnaughmapMapColor}{#1}}
```

Define the label color. By setting the color only, the draw remains unchanged.

```
\define@key{karnaughmap}{labelcolor}[karnaughmapColor]{%
  \colorlet{karnaughmapColor}{#1}}
```

Define the shifts.

```
\define@cmdkeys{karnaughmap}{xshift,yshift}[0]
```

Set Key Defaults The default is a map with full information, i.e. binary and variable labels as well as decimal indices. The zeros are omitted by default.

`\setkarnaughmap` Simple interface function.

```
\newcommand{\setkarnaughmap}[1][]{%
\ifthenelse{\equal{#1}{}}{%
```

Change the default behavior by uncommenting the specific key.

```
\setkeys{karnaughmap}{%
```

omitones, omitdontcares, omitbinaries, omitvariables, omitidx, binaryidx

```
omitzeros, omitnegated}
```

The key-definition defaults are assigned here. See the definition of the keys above for the defaults.

```
\setkeys{karnaughmap}{defaultmap, function, variables, dontcare, color, draw, labelcolor, xshif
```

If the optional input is provided set the keys accordingly.

```
}{
\setkeys{karnaughmap}{#1}
}}
```

Assign the defaults during style file call.

```
\setkarnaughmap
```

8.4 Auxiliary Functions

These function are called by the main function to facilitate code execution.

\karnaughmapPrintIndex `\karnaughmapPrintIndex[`⟨*numBits*⟩`]`

This macro typesets the indices inside the cells. If the switch `binaryidx` is true binary instead of decimal indices are typeset. The number provided as optional argument is used to determine the number of bits used.

Every time the macro is called, it prints the counter value of counter `karnaughmapIdxCounter` and increments the counter thereafter.

The size of the indices is very small (scriptsize).

```
\newcommand{\karnaughmapPrintIndex}[1][4]{%
\scriptsize%
\ifKV@karnaughmap@binaryidx
```

Typeset binary indices.

```
  \ifthenelse{\equal{#1}{2}}{%
    \ifcase\value{karnaughmapIdxCounter}%
      00\or 01\or 10\or 11
    \fi
  }{}%
  \ifthenelse{\equal{#1}{3}}{%
    \ifcase\value{karnaughmapIdxCounter}%
      000\or 001\or 010\or 011\or 100\or 101\or 110\or 111
    \fi
  }{}%
  \ifthenelse{\equal{#1}{4}}{%
    \ifcase\value{karnaughmapIdxCounter}%
      0000\or 0001\or 0010\or 0011\or 0100\or 0101\or 0110\or 0111\or
      1000\or 1001\or 1010\or 1011\or 1100\or 1101\or 1110\or 1111
  \fi
  }{}%
\else
```

Typeset decimal indices.

```
$\thekarnaughmapIdxCounter$%
\fi
\stepcounter{karnaughmapIdxCounter}%
}
```

\karnaughmapPrintValue `\karnaughmapPrintValue{`⟨*inputstring*⟩`}`

This macro typesets the entries of the Karnaugh map. Every time the macro is called, it increments the counter `karnaughmapStrCounter` and prints the character at position of the counter value of its mandatory input.

```
\newcommand{\karnaughmapPrintValue}[2][]{%
  \stepcounter{karnaughmapStrCounter}%
  \StrChar{#2}{\thekarnaughmapStrCounter}}
```

\karnaughmapPrintCellContents `\karnaughmapPrintCellContents[`⟨*node-position*⟩`]{`⟨*x-offset*⟩`}{`⟨*y-offset*⟩`}{`⟨*size*⟩`}{`⟨*rowsize*⟩`}{`⟨*function*⟩`}`

This macro is used to provide the positions of the cells of the map, as they are not straight forward (e.g. order is 0-1-3-2 for first column of 4x4 maps). It takes 5 mandatory and one optional arguments as specified above.

By default, it evaluates `function` as contents of a node at the top left of each cell of a Karnaugh map of size `size` with `rowsize` rows. By providing the bias (x- and y-offset), the provided point is shifted accordingly. The alignment of the node is controlled by the optional argument.

```
\newcommand{\karnaughmapPrintCellContents}[6][]{%
```

If map is 2x2, the order is pretty simple:
0 2
1 3

```
\ifthenelse{\equal{#4}{4}}{%
  \def\karnaughmapPCCColumnSpecifier{++( 0,-1) node[#1] {#6}}%
  \path  (0, #5)
    ++(#2,#3) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+1) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
  ;
}{}
```

If map is 2x4, the order is more complex:
0 2 6 4
1 3 7 5

```
\ifthenelse{\equal{#4}{8}}{%
  \def\karnaughmapPCCColumnSpecifier{++( 0,-1) node[#1] {#6}}%
  \path (0, #5)
    ++(#2,#3) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+1) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+2,+1) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-1,+1) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
  ;
}{}
```

If map is 4x4, the same ordering as before is present in both x- and y-direction:
0 4 12 8
1 5 13 9
3 7 15 11
2 6 14 10

```
\ifthenelse{\equal{#4}{16}}{%
  \def\karnaughmapPCCColumnSpecifier{%
    ++( 0,-1) node[#1] {#6}
```

```
    ++( 0,-2) node[#1] {#6}
    ++( 0,+1) node[#1] {#6}
  }%
  \path (0, #5)
    ++(#2,#3) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+2) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+2,+2) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-1,+2) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
  ;
}{}
```

If map is 4x8, the ordering as before has to be extended in x-direction, but from the middle in reverse order:
0 4 12 8 24 28 20 16
1 5 13 9 25 29 21 17
3 7 15 11 27 31 23 29
2 6 14 10 26 30 22 18

```
\ifthenelse{\equal{#4}{32}}{%
  \def\karnaughmapPCCColumnSpecifier{%
    ++( 0,-1) node[#1] {#6}
    ++( 0,-2) node[#1] {#6}
    ++( 0,+1) node[#1] {#6}
  }%
  \path (0, #5)
    ++(#2,#3) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+2) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+2,+2) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-1,+2) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+5,+2) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-1,+2) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-2,+2) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+2) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
  ;
}{}
```

If map is 8x8, the ordering as before has to be extended in y-direction, too.
0 8 24 16 48 56 40 32
1 9 25 17 49 57 41 33

3 11 27 19 51 59 43 35
2 10 26 18 50 58 42 34
6 14 30 22 54 62 46 38
7 15 31 23 55 63 47 39
5 13 29 21 53 61 45 37
4 12 28 20 52 60 44 36

```
\ifthenelse{\equal{#4}{64}}{%
  \def\karnaughmapPCCColumnSpecifier{%
    ++( 0,-1) node[#1] {#6}
    ++( 0,-2) node[#1] {#6}
    ++( 0,+1) node[#1] {#6}
    ++( 0,-5) node[#1] {#6}
    ++( 0,+1) node[#1] {#6}
    ++( 0,+2) node[#1] {#6}
    ++( 0,-1) node[#1] {#6}
  }%
  \path (0, #5)
    ++(#2,#3) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+2,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-1,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+5,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-1,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-2,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
  ;
}{}
```

If map is 8x16, the ordering is:

0 8 24 16 48 56 40 32
1 9 25 17 49 57 41 33
3 11 27 19 51 59 43 35
2 10 26 18 50 58 42 34
6 14 30 22 54 62 46 38
7 15 31 23 55 63 47 39
5 13 29 21 53 61 45 37
4 12 28 20 52 60 44 36

```
\ifthenelse{\equal{#4}{128}}{%
  \def\karnaughmapPCCColumnSpecifier{%
    ++( 0,-1) node[#1] {#6}
    ++( 0,-2) node[#1] {#6}
```

```
    ++( 0,+1) node[#1] {#6}
    ++( 0,-5) node[#1] {#6}
    ++( 0,+1) node[#1] {#6}
    ++( 0,+2) node[#1] {#6}
    ++( 0,-1) node[#1] {#6}
  }%
  \path (0, #5)
    ++(#2,#3) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+2,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-1,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+5,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-1,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-2,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+10,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-1,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-2,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-5,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+2,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-1,+5) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
  ;
}{}
```

If map is 16x16, the ordering is:

0 8 24 16 48 56 40 32
1 9 25 17 49 57 41 33
3 11 27 19 51 59 43 35
2 10 26 18 50 58 42 34
6 14 30 22 54 62 46 38
7 15 31 23 55 63 47 39
5 13 29 21 53 61 45 37

4 12 28 20 52 60 44 36

```
\ifthenelse{\equal{#4}{256}}{%
  \def\karnaughmapPCCColumnSpecifier{%
    ++( 0,-1) node[#1] {#6}
    ++( 0,-2) node[#1] {#6}
    ++( 0,+1) node[#1] {#6}
    ++( 0,-5) node[#1] {#6}
    ++( 0,+1) node[#1] {#6}
    ++( 0,+2) node[#1] {#6}
    ++( 0,-1) node[#1] {#6}

    ++( 0,-10) node[#1] {#6}
    ++( 0,+1) node[#1] {#6}
    ++( 0,+2) node[#1] {#6}
    ++( 0,-1) node[#1] {#6}
    ++( 0,+5) node[#1] {#6}
    ++( 0,-1) node[#1] {#6}
    ++( 0,-2) node[#1] {#6}
    ++( 0,+1) node[#1] {#6}
  }%
  \path (0, #5)
    ++(#2,#3) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+2,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-1,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+5,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-1,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-2,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+10,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-1,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-2,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-5,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(+1,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
```

```
    ++(+2,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
    ++(-1,+10) node[#1] {#6}
      \karnaughmapPCCColumnSpecifier
  ;
}{}
}
```

`\karnaughmapHighlightField` Macro that defines the highlighting. `\karnaughmapHighlightField[`⟨*key=value*⟩`]{`⟨*coordinates*⟩`}{`⟨*color name*⟩`}`

```
\newcommand{\karnaughmapHighlightField}[3][]{%
\ifKV@karnaughmap@fill
  \path[fill=#3,opacity=\cmdKV@karnaughmap@opacity,#1] (#2) rectangle ++(1,1);%
\fi
\ifKV@karnaughmap@outline
  \draw[#3,#1] (#2) ++ (0.1,0.1) rectangle ++(0.8,0.8);%
\fi
}
```

`\karnaughmapShadeMapfieldTWO` Macro to shade square two input karnaugh map fields. `\karnaughmapShadeMapfieldTWO[`⟨*key=value*⟩`]{`⟨*decimal field number*⟩`}{`⟨*color name*⟩`}`

Optional macro is used as key-val interface for option processing, mandatory ones specify the color used for highlighting and the fields that are to be highlighted as decimal integers from 1 to $2^{\text{map size}} - 1$.

```
\newcommand{\karnaughmapShadeMapfieldTWO}[3][]{%
\begin{scope}[scale=1]
  \ifthenelse{\equal{#2}{0}}{\karnaughmapHighlightField[#1]{0,1}{#3}}{}%
  \ifthenelse{\equal{#2}{1}}{\karnaughmapHighlightField[#1]{0,0}{#3}}{}%
  \ifthenelse{\equal{#2}{2}}{\karnaughmapHighlightField[#1]{1,1}{#3}}{}%
  \ifthenelse{\equal{#2}{3}}{\karnaughmapHighlightField[#1]{1,0}{#3}}{}%
\end{scope}
}
```

`\karnaughmapShadeMapfieldTHREE` Macro to shade three input karnaugh map fields. `\karnaughmapShadeMapfieldTHREE[`⟨*key=value*⟩`]{`⟨*decimal field number*⟩`}{`⟨*color name*⟩`}`

Optional macro is used as key-val interface for option processing, mandatory ones specify the color used for highlighting and the fields that are to be highlighted as decimal integers from 1 to $2^{\text{map size}} - 1$.

```
\newcommand{\karnaughmapShadeMapfieldTHREE}[3][]{%
\begin{scope}[scale=1]
  \ifthenelse{\equal{#2}{0}}{\karnaughmapHighlightField[#1]{0,1}{#3}}{}%
  \ifthenelse{\equal{#2}{1}}{\karnaughmapHighlightField[#1]{0,0}{#3}}{}%
  \ifthenelse{\equal{#2}{2}}{\karnaughmapHighlightField[#1]{1,1}{#3}}{}%
  \ifthenelse{\equal{#2}{3}}{\karnaughmapHighlightField[#1]{1,0}{#3}}{}%
  \ifthenelse{\equal{#2}{4}}{\karnaughmapHighlightField[#1]{3,1}{#3}}{}%
  \ifthenelse{\equal{#2}{5}}{\karnaughmapHighlightField[#1]{3,0}{#3}}{}%
```

```
  \ifthenelse{\equal{#2}{6}}{\karnaughmapHighlightField[#1]{2,1}{#3}}{}%
  \ifthenelse{\equal{#2}{7}}{\karnaughmapHighlightField[#1]{2,0}{#3}}{}%
\end{scope}
}
```

`\karnaughmapShadeMapfieldFOUR` Macro to shade square four input karnaugh map fields.
`\karnaughmapShadeMapfieldFOUR[`⟨*key=value*⟩`]{`⟨*decimal field number*⟩`}{`⟨*color name*⟩`}`

Optional macro is used as key-val interface for option processing, mandatory ones specify the color used for highlighting and the fields that are to be highlighted as decimal integers from 1 to $2^{\text{map size}} - 1$.

```
\newcommand{\karnaughmapShadeMapfieldFOUR}[3][]{%
\begin{scope}[scale=1]%
```

1st column

```
\ifthenelse{\equal{#2}{0}}{\karnaughmapHighlightField[#1]{0,3}{#3}}{}%
\ifthenelse{\equal{#2}{1}}{\karnaughmapHighlightField[#1]{0,2}{#3}}{}%
\ifthenelse{\equal{#2}{2}}{\karnaughmapHighlightField[#1]{0,0}{#3}}{}%
\ifthenelse{\equal{#2}{3}}{\karnaughmapHighlightField[#1]{0,1}{#3}}{}%
```

2nd column

```
\ifthenelse{\equal{#2}{4}}{\karnaughmapHighlightField[#1]{1,3}{#3}}{}%
\ifthenelse{\equal{#2}{5}}{\karnaughmapHighlightField[#1]{1,2}{#3}}{}%
\ifthenelse{\equal{#2}{6}}{\karnaughmapHighlightField[#1]{1,0}{#3}}{}%
\ifthenelse{\equal{#2}{7}}{\karnaughmapHighlightField[#1]{1,1}{#3}}{}%
```

4th column

```
\ifthenelse{\equal{#2}{8}}{\karnaughmapHighlightField[#1]{3,3}{#3}}{}%
\ifthenelse{\equal{#2}{9}}{\karnaughmapHighlightField[#1]{3,2}{#3}}{}%
\ifthenelse{\equal{#2}{a}\OR\equal{#2}{A}}{\karnaughmapHighlightField[#1]{3,0}{#3}}{}%
\ifthenelse{\equal{#2}{b}\OR\equal{#2}{B}}{\karnaughmapHighlightField[#1]{3,1}{#3}}{}%
```

3rd column

```
\ifthenelse{\equal{#2}{c}\OR\equal{#2}{C}}{\karnaughmapHighlightField[#1]{2,3}{#3}}{}%
\ifthenelse{\equal{#2}{d}\OR\equal{#2}{D}}{\karnaughmapHighlightField[#1]{2,2}{#3}}{}%
\ifthenelse{\equal{#2}{e}\OR\equal{#2}{E}}{\karnaughmapHighlightField[#1]{2,0}{#3}}{}%
\ifthenelse{\equal{#2}{f}\OR\equal{#2}{F}}{\karnaughmapHighlightField[#1]{2,1}{#3}}{}%
\end{scope}
}
```

8.5 Main Functions

Here come the main functions to typeset the map and to mark certain cells (not implemented yet).

`\karnaughmap` The main macro takes one mandatory and one optional argument:
`[`⟨*key=value*⟩`]{`⟨*inputstring*⟩`}`.

The optional is used as key-val inteface, the mandatory is empty or holds the entry-pattern (inputstring) according to the corresponding ordered truth table of the desired logic function.

```
\newcommand{\karnaughmap}[2][]{%
```

If the optional input is not empty, assign the key-val.

```
\ifthenelse{\equal{#1}{}}{}{\setkeys{karnaughmap}{#1}}
```

For easy access store the mandatory argument as variable `\karnaughmapCellEntries`.

```
\def\karnaughmapCellEntries{#2}%
%     \end{macrocode}
%
% Check length of mandatory argument and store the result in |\karnaughmapSize|.
%
% If the mandatory argument is empty,
% use the default map size provided by the key-val interface (|defaultmap| key).
%    \begin{macrocode}
\ifthenelse{\equal{#2}{}}{%
  \def\karnaughmapSize{\cmdKV@karnaughmap@defaultmap}
}{%
```

Else, determine the length of the input string to determine the map size.

Before calculating the string length, remove any blanks in the input string. Usage of blanks in the input string can increase readability, e.g. when groups of four are provided.

```
\IfSubStr{\karnaughmapCellEntries}{ }{%
\StrDel[0]{\karnaughmapCellEntries}{ }[\karnaughmapCellEntries]
}{}
```

Now, calculate the string length. This is the old version. `\StrLen{#2}[\karnaughmapSize]%`

```
  \StrLen{\karnaughmapCellEntries}[\karnaughmapSize]%
```

If the input is a scalar number, i.e. it is ¡ 1000 (decimal), it is treated as number and is used to determine the map size. In this case adapt the variable `\karnaughmapCellEntries` accordingly. `\ifthenelse{\equal{\karnaughmapSize}{1}}{%`

```
  \ifthenelse{\karnaughmapSize < 4}{%
    \def\karnaughmapSize{#2}\def\karnaughmapCellEntries{}}{}%
}
```

Parse the input string and omit printing of '0', '1', or '*' entries if the switches say so.

```
\ifKV@karnaughmap@omitzeros
  \StrSubstitute[0]{\karnaughmapCellEntries}{0}{ }[\karnaughmapCellEntries]
\fi
\ifKV@karnaughmap@omitones
  \StrSubstitute[0]{\karnaughmapCellEntries}{1}{ }[\karnaughmapCellEntries]
\fi
\ifKV@karnaughmap@omitdontcares
  \StrSubstitute[0]{\karnaughmapCellEntries}{\cmdKV@karnaughmap@dontcare}{ }[\karnaughmapCellEn
\fi
```

Initialize internal counters.

```
\setcounter{karnaughmapIdxCounter}{0}
\setcounter{karnaughmapStrCounter}{0}
```

Extract the map variables and alloc row and col sizes. In addition, store the number of variables in variable `\karnaughmapNumVar`.

```
\ifthenelse{\karnaughmapSize = 4}{%
  \StrChar{\cmdKV@karnaughmap@variables}{1}[\karnaughmapVarLabelA]
  \def\karnaughmapVarLabelB{}
  \StrChar{\cmdKV@karnaughmap@variables}{2}[\karnaughmapVarLabelC]
  \def\karnaughmapVarLabelD{}
  \def\karnaughmapNumRow{2}
  \def\karnaughmapNumCol{2}
  \def\karnaughmapNumVar{2}
}{}
\ifthenelse{\karnaughmapSize = 8}{%
  \StrChar{\cmdKV@karnaughmap@variables}{1}[\karnaughmapVarLabelA]
  \StrChar{\cmdKV@karnaughmap@variables}{2}[\karnaughmapVarLabelB]
  \StrChar{\cmdKV@karnaughmap@variables}{3}[\karnaughmapVarLabelC]
  \def\karnaughmapVarLabelD{}
  \def\karnaughmapNumRow{2}
  \def\karnaughmapNumCol{4}
  \def\karnaughmapNumVar{3}
}{}
\ifthenelse{\karnaughmapSize = 16}{%
  \StrChar{\cmdKV@karnaughmap@variables}{1}[\karnaughmapVarLabelA]
  \StrChar{\cmdKV@karnaughmap@variables}{2}[\karnaughmapVarLabelB]
  \StrChar{\cmdKV@karnaughmap@variables}{3}[\karnaughmapVarLabelC]
  \StrChar{\cmdKV@karnaughmap@variables}{4}[\karnaughmapVarLabelD]
  \def\karnaughmapNumRow{4}
  \def\karnaughmapNumCol{4}
  \def\karnaughmapNumVar{4}
}{}
```

The bigger karnaugh maps, i.e. 32, 64, 128 and 512, lack finer control possibilities. Rather, the variables are set such that they can be used as map descriptors, but not for some fancy stuff like indicating the variable blocks. This means, that we force to omit printing of the the indices (omitidx=true) and we force to omit printing of the variables.
For the variable macros to work, we set the B and D macro to empty and put the remaining variables in the A for column and C for row variables.

```
\ifthenelse{\karnaughmapSize = 32}{%
  \setkeys{karnaughmap}{omitidx=true, omitvariables=true}
  \StrMid{\cmdKV@karnaughmap@variables}{1}{3}[\karnaughmapVarLabelA]
  \def\karnaughmapVarLabelB{}
  \StrMid{\cmdKV@karnaughmap@variables}{4}{5}[\karnaughmapVarLabelC]
  \def\karnaughmapVarLabelD{}
  \def\karnaughmapNumRow{4}
  \def\karnaughmapNumCol{8}
  \def\karnaughmapNumVar{5}
}{}
\ifthenelse{\karnaughmapSize = 64}{%
  \setkeys{karnaughmap}{omitidx=true, omitvariables=true}
  \StrMid{\cmdKV@karnaughmap@variables}{1}{3}[\karnaughmapVarLabelA]
  \def\karnaughmapVarLabelB{}
  \StrMid{\cmdKV@karnaughmap@variables}{4}{6}[\karnaughmapVarLabelC]
```

```
  \def\karnaughmapVarLabelD{}
  \def\karnaughmapNumRow{8}
  \def\karnaughmapNumCol{8}
  \def\karnaughmapNumVar{6}
}{}
\ifthenelse{\karnaughmapSize = 128}{%
  \setkeys{karnaughmap}{omitidx=true, omitvariables=true}
  \StrMid{\cmdKV@karnaughmap@variables}{1}{4}[\karnaughmapVarLabelA]
  \def\karnaughmapVarLabelB{}
  \StrMid{\cmdKV@karnaughmap@variables}{5}{7}[\karnaughmapVarLabelC]
  \def\karnaughmapVarLabelD{}
  \def\karnaughmapNumRow{8}
  \def\karnaughmapNumCol{16}
  \def\karnaughmapNumVar{7}
}{}
\ifthenelse{\karnaughmapSize = 256}{%
  \setkeys{karnaughmap}{omitidx=true, omitvariables=true}
  \StrMid{\cmdKV@karnaughmap@variables}{1}{4}[\karnaughmapVarLabelA]
  \def\karnaughmapVarLabelB{}
  \StrMid{\cmdKV@karnaughmap@variables}{5}{8}[\karnaughmapVarLabelC]
  \def\karnaughmapVarLabelD{}
  \def\karnaughmapNumRow{16}
  \def\karnaughmapNumCol{16}
  \def\karnaughmapNumVar{8}
}{}
```

Basic table Typeset the basic table.

Embed the draw commands in a scope environment for style control.

```
  \begin{scope}[xshift=\cmdKV@karnaughmap@xshift,
                yshift=\cmdKV@karnaughmap@yshift,
                color=karnaughmapColor,
                draw=karnaughmapMapColor ]
```

Draw table boundaries.

```
  \draw (0,0) rectangle (\karnaughmapNumCol, \karnaughmapNumRow);
```

Draw column seperators.

```
  \foreach \x in {1,2,...,\karnaughmapNumCol} {%
    \ifthenelse{\equal{\x}{1}}{}{%
      \draw (\x-1,0) -- ++(0,\karnaughmapNumRow);%
    }%
  }%
```

Draw row seperators.

```
  \foreach \x in {1,2,...,\karnaughmapNumRow} {%
    \ifthenelse{\equal{\x}{1}}{}{%
      \draw (0,\x-1) -- ++(\karnaughmapNumCol,0);%
    }%
  }%
```

Draw map labels; Diagonal line and anchor first.

```
\draw (0,\karnaughmapNumRow) -- ++(-0.4,0.4) coordinate(identifier)
             -- ++(-0.4,0.4)
```

Function label at the top of the map, if the function label is non-empty the label is followed by a colon.

```
    node[yshift=7.5,xshift=-3.5,right] {%
      \ifthenelse{\equal{\cmdKV@karnaughmap@function}{}}{}{$\cmdKV@karnaughmap@function\colon$}
```

Variable label right and below the diagonal.

```
 \path (identifier) node[above right, xshift=-2.5] {$\karnaughmapVarLabelA\karnaughmapVarLabel
 \path (identifier) node[below left, yshift=0.5] {$\karnaughmapVarLabelC\karnaughmapVarLabelD$
```

Typeset binaries if `omitbinaries` switch is false;

```
\ifKV@karnaughmap@omitbinaries
\else
```

2 columns: anchors are above the top row `\karnaughmapNumRow` and in the middle of the cells at 0.5 and 1.5.

```
 \ifthenelse{\equal{\karnaughmapNumCol}{2}}{%
     \foreach \pos/\lab in {0.5/0, 1.5/1} {%
       \path (\pos, \karnaughmapNumRow) node[above] {$\lab$};%
     }%
 }{}%
```

4 columns: anchors are above the top row `\karnaughmapNumRow` and in the middle of the cells at 0.5 ... 3.5.

```
 \ifthenelse{\equal{\karnaughmapNumCol}{4}}{%
   \foreach \pos/\lab in {0.5/00, 1.5/01, 2.5/11, 3.5/10} {%
     \path (\pos, \karnaughmapNumRow) node[above] {$\lab$};%
   }%
 }{}%
```

8 columns: anchors are above the top row `\karnaughmapNumRow` and in the middle of the cells at 0.5 ... 7.5.

```
 \ifthenelse{\equal{\karnaughmapNumCol}{8}}{%
   \foreach \pos/\lab in {0.5/000, 1.5/001, 2.5/011, 3.5/010, 4.5/110, 5.5/111, 6.5/101, 7.5/1
     \path (\pos, \karnaughmapNumRow) node[above] {$\lab$};%
   }%
 }{}%
```

16 columns: anchors are above the top row `\karnaughmapNumRow` and in the middle of the cells at 0.5 ... 15.5.

```
 \ifthenelse{\equal{\karnaughmapNumCol}{16}}{%
   \foreach \pos/\lab in {0.5/0000, 1.5/0001, 2.5/0011, 3.5/0010, 4.5/0110, 5.5/0111, 6.5/0101
                          8.5/1100, 9.5/1101,10.5/1111,11.5/1110,12.5/1010,13.5/1011,14.5/1001
     \path (\pos, \karnaughmapNumRow) node[above] {$\lab$};%
   }%
 }{}%
```

2 rows: anchors are left of the first column with coordinate (0, x) and in the middle of the cells at 0.5 and 1.5.

```
  \ifthenelse{\equal{\karnaughmapNumRow}{2}}{%
      \foreach \pos/\lab in {0.5/0, 1.5/1} {%
        \path (0, \karnaughmapNumRow-\pos) node[left] {$\lab$};%
      }%
  }{}%
```

4 rows: anchors are left of the first column with coordinate (0, x) and in the middle of the cells at 0.5 ... 3.5.

```
  \ifthenelse{\equal{\karnaughmapNumRow}{4}}{%
    \foreach \pos/\lab in {0.5/00, 1.5/01, 2.5/11, 3.5/10} {%
      \path (0, \karnaughmapNumRow-\pos) node[left] {$\lab$};%
    }%
  }{}%
```

8 rows: anchors are left of the first column with coordinate (0, x) and in the middle of the cells at 0.5 ... 7.5.

```
  \ifthenelse{\equal{\karnaughmapNumRow}{8}}{%
    \foreach \pos/\lab in {0.5/000, 1.5/001, 2.5/011, 3.5/010, 4.5/110, 5.5/111, 6.5/101, 7.5/1
      \path (-0.125, \karnaughmapNumRow-\pos) node[left] {$\lab$};%
    }%
  }{}%
```

16 rows: anchors are left of the first column with coordinate (0, x) and in the middle of the cells at 0.5 ... 15.5.

```
  \ifthenelse{\equal{\karnaughmapNumRow}{16}}{%
    \foreach \pos/\lab in {0.5/0000, 1.5/0001, 2.5/0011, 3.5/0010, 4.5/0110, 5.5/0111, 6.5/0101
                           8.5/1100, 9.5/1101,10.5/1111,11.5/1110,12.5/1010,13.5/1011,14.5/1001
      \path (-0.1875, \karnaughmapNumRow-\pos) node[left] {$\lab$};%
    }%
  }{}%
\fi
```

Typeset binary labels above and left of the map if switch `omitbinaries` is false.

Set base bias of variable labels; the top and left bias is modified depending on if the binary labels are set or not.

```
\def\karnaughmapVariableBaseBias{0.2}
\ifKV@karnaughmap@omitbinaries
```

Set bias to 0 if binaries are not typeset.

```
  \def\karnaughmapVariableTopBias{0}
  \def\karnaughmapVariableLeftBias{0}
\else
```

Set bias accordingly if binaries have 1 or 2 characters.

```
  \def\karnaughmapVariableTopBias{0.3}
  \ifthenelse{\equal{\karnaughmapVarLabelD}{}}{%
    \def\karnaughmapVariableLeftBias{0.2}
  }{%
    \def\karnaughmapVariableLeftBias{0.4}
  }
\fi
```

Typeset the variable labels around the map if switch `omitvariables` is false.

```
\ifKV@karnaughmap@omitvariables
\else
  \ifthenelse{\equal{\karnaughmapVarLabelA}{}}{%
  }{%
    \ifthenelse{\equal{\karnaughmapVarLabelB}{}}{%
      \draw (1.1, \karnaughmapNumRow + \karnaughmapVariableBaseBias + \karnaughmapVariableTopBi
        -- node[above] {$\karnaughmapVarLabelA$} ++(0.8, 0);
      \ifKV@karnaughmap@omitnegated
      \else
        \draw (0.1, \karnaughmapNumRow + \karnaughmapVariableBaseBias + \karnaughmapVariableTop
          -- node[above] {$\overline{\karnaughmapVarLabelA}$} ++(0.8, 0);
      \fi
    }{%
      \draw (2.1, \karnaughmapNumRow + \karnaughmapVariableBaseBias + \karnaughmapVariableTopBi
        -- node[above] {$\karnaughmapVarLabelA$} ++(1.8, 0);
      \draw (1.1, -\karnaughmapVariableBaseBias)
        -- node[below] {$\karnaughmapVarLabelB$} ++(1.8, 0);
      \ifKV@karnaughmap@omitnegated
      \else
        \draw (0.1, \karnaughmapNumRow + \karnaughmapVariableBaseBias + \karnaughmapVariableTop
          -- node[above] {$\overline{\karnaughmapVarLabelA}$} ++(1.8, 0);
        \draw (0.1, -\karnaughmapVariableBaseBias)
          -- node[below] {$\overline{\karnaughmapVarLabelB}$} ++(0.8, 0);
        \draw (3.1, -\karnaughmapVariableBaseBias)
          -- node[below] {$\overline{\karnaughmapVarLabelB}$} ++(0.8, 0);
      \fi
    }
  }
  \ifthenelse{\equal{\karnaughmapVarLabelC}{}}{%
  }{%
    \ifthenelse{\equal{\karnaughmapVarLabelD}{}}{%
      \draw (-\karnaughmapVariableBaseBias -\karnaughmapVariableLeftBias, 0.1)
        -- node[left] {$\karnaughmapVarLabelC$} ++(0, 0.8);
      \ifKV@karnaughmap@omitnegated
      \else
        \draw (-\karnaughmapVariableBaseBias -\karnaughmapVariableLeftBias, 1.1)
          -- node[left] {$\overline{\karnaughmapVarLabelC}$} ++(0, 0.8);
      \fi
    }{%
      \draw (-\karnaughmapVariableBaseBias -\karnaughmapVariableLeftBias, 0.1)
        -- node[left] {$\karnaughmapVarLabelC$} ++(0, 1.8);
      \draw (\karnaughmapNumCol + \karnaughmapVariableBaseBias, 1.1)
        -- node[right] {$\karnaughmapVarLabelD$} ++(0, 1.8);
      \ifKV@karnaughmap@omitnegated
      \else
        \draw (-\karnaughmapVariableBaseBias -\karnaughmapVariableLeftBias, 2.1)
          -- node[left] {$\overline{\karnaughmapVarLabelC}$} ++(0, 1.8);
        \draw (\karnaughmapNumCol + \karnaughmapVariableBaseBias, 0.1)
```

```
          -- node[right] {$\overline{\karnaughmapVarLabelD}$} ++(0, 0.8);
        \draw (\karnaughmapNumCol + \karnaughmapVariableBaseBias, 3.1)
          -- node[right] {$\overline{\karnaughmapVarLabelD}$} ++(0, 0.8);
      \fi
    }
  }
\fi
```

Typeset decimal index if switch `omitidx` is false.

```
\ifKV@karnaughmap@omitidx
\else
  \karnaughmapPrintCellContents[anchor=west]{0.0}{-0.2}{\karnaughmapSize}{\karnaughmapNumRow}{\
\fi
```

Typeset the cell entries.

```
\karnaughmapPrintCellContents{0.5}{-0.5}{\karnaughmapSize}{\karnaughmapNumRow}{\karnaughmapPrin
```

Close the scope

```
  \end{scope}
```

Finishing macro bracket.

```
}
```

`\karnaughmapcolorfield` The coloring macro takes two mandatory and one optional argument: `[`⟨*key=value*⟩`]{`⟨*map size*⟩`}{`⟨*fields*⟩`}{`⟨*color*⟩`}`.

```
\newcommand{\karnaughmapcolorfield}[4][]{%
\setkeys{karnaughmap}{fill,opacity}%
\ifthenelse{\equal{#1}{}}{}{\setkeys*{karnaughmap}{#1}}%
\ifthenelse{\equal{#3}{}}{}{%
\StrSplit{#3}{1}{\karnaughmapColorfieldCur}{\karnaughmapColorfieldRem}
\ifthenelse{\equal{#2}{2}}{%
\karnaughmapShadeMapfieldTWO[\XKV@rm]{\karnaughmapColorfieldCur}{#4}
}{}
\ifthenelse{\equal{#2}{3}}{%
\karnaughmapShadeMapfieldTHREE[\XKV@rm]{\karnaughmapColorfieldCur}{#4}
}{}
\ifthenelse{\equal{#2}{4}}{%
\karnaughmapShadeMapfieldFOUR[\XKV@rm]{\karnaughmapColorfieldCur}{#4}
}{}
\karnaughmapcolorfield[#1]{#2}{\karnaughmapColorfieldRem}{#4}
}
}
```

9 To Do

- Adjust the map label positions as well as the function label to fit the negated variant.
- Increase supported karnaugh map size: 5 and 6 variables should be feasible in a single karnaugh map

- Provide the possibility to permute the karnaugh map

Change History

Index

Numbers written in italic refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in roman refer to the code lines where the entry is used.

K

L

N

O