

TikZ Codeblocks v.0.13

Adrian Salamon

2020-04-06

Inhaltsverzeichnis

1	Präambel	2
1.1	Editoren mit graphischer Programmierung	3
1.2	Alternativpakete	3
1.3	Installation und Benutzung	3
2	Beispielcode	4
2.1	English Codeexample	5
3	Bausteine und Befehle	6
3.1	Bausteinclassen nach PXT-Calliope Editor	6
3.2	Skalierung	7
3.3	Farben	7
3.3.1	Lokale Farbänderung	7
3.3.2	Globale Farbänderung	8
3.3.3	Keine Farben (print)	8
3.4	Boxen	8
3.5	Bilder/LED-Matrix	9
3.6	Strukturen	10
3.6.1	Verzweigungen	10
3.6.2	Schleifen	11
3.6.3	Branches, loops and the english language	12
4	Positionierung der Nodes	12
4.1	Manuelles Positionieren	12
4.1.1	Verschieben der Zacken bei manueller Einrückung	13
4.2	Automatisches Einrücken	13
5	Dekorationen	14
5.1	Puzzleoptik	14
5.2	Symbole und kleine Elemente	14
6	FAQ	15
6.1	Wie kann ich für ein komplettes Dokument eine Farbe umdefinieren?	15
6.2	Eine nested Box ist zu hoch/zu niedrig	15
6.3	Ich will einen Node innerhalb eines Nodes setzen	15
6.4	Mein Block ist sehr klein und deswegen verformt	15

7	Mehr Beispielcode	16
7.1	Bsp: Calliope Smart Home	16

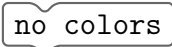
Versionshistorie

Version	Datum	Autor(en)	Änderungen
0.1	2017-07-06	A. Salamon	published
0.11	2017-07-18	A. Salamon	update: shapes, translations and fix typos
0.12	2018-04-03	A. Salamon	update: LED colors
0.13	2020-04-06	A. Salamon, A. Wagner	bug-fix: wrongly shifed nodes

- Translations
 - via **babel** for structures (if/wenn, then/dann...)
 - marcos and styles are now (partly) also in english¹

- new Shapes



- Package option **print** gets rid of all colors: 
- Open-Roberta colors are now more accurate and usable next to the standard colors. (see 3.1.)

- fixed LED-Color for NEPO-Editor.



1 Präambel

Diese Sammlung von TikZ Stilen und Kommandos soll helfen, grafische Codeblöcke, wie beim Calliope mini Editor, der Sprache NEPO von Open Roberta oder der Programmierumgebung Scratch zu setzen. Sie ersetzt **nicht** die Auseinandersetzung mit TikZ und der entsprechenden Syntax. Die Positionierung, Benennung und Referenzierung der Elemente muss weiterhin manuell gestaltet werden.

Dies ist eine Entwicklerversion. Bezeichner und Paketname können in späteren Versionen noch variieren.

Fragen nehme ich gerne per adriansalamon@gmail.com entgegen.

¹Feel free to contact me, if you are an english speaker and want to use everything in english.

1.1 Editoren mit graphischer Programmierung

`tikzcodeblocks` wurde erstellt, um Quelltexte auf Blockbasis für den Microcontroller Calliope mini zu setzen. Alle verwendeten Farben und Gestaltungen sind daher standardmäßig vom PXT Editor von Calliope inspiriert. Als alternative Farbgebung ist eine Variation nach Open Roberta implementiert (siehe 3.1). Es lässt sich jedoch auch Code mit Farb- und Formgebungen anderer Editoren setzen.

Editor	URL
PXT - Calliope	http://pxt.calliope.cc/
Open Roberta	https://lab.open-roberta.org/
Scratch	https://scratch.mit.edu/projects/editor

Tabelle 2: Tabelle von Editoren mit graphischen Codeblöcken.

1.2 Alternativpakete

Während der Entwicklung dieses Paketes wurde das Paket `scratch`² veröffentlicht. Damit lässt sich graphischer Code in der Optik von Scratch sehr einfach setzen. Die Dokumentation des Pakets ist zur Zeit auf Französisch verfügbar. Ein Unterschied zwischen den Paketen `scratch` und `tikzcodeblocks` ist m. E. vor allem folgender: `scratch` lässt sich **nicht** in andere TikZ Umgebungen einbinden. Die einzelnen Objekte sind nicht als Nodes³ referenzierbar. Die Syntax ist jedoch deutlich schmaler als bei `tikzcodeblocks`. Ein Blick lohnt sich bestimmt für die meisten interessierten Nutzer.

1.3 Installation und Benutzung

`tikzcodeblocks`

Das Paket wird über die üblichen L^AT_EX-Paketmanager installiert (z. B. T_EXLive) und per `\usepackage{tikzcodeblocks}` in die Präambel des gewünschten Dokuments eingebunden.

`codeblocks`

Mit `\begin{tikzpicture}[codeblocks]` werden die Codeblock-Stile in die jeweilige `tikzpicture`-Umgebung geladen. Es setzt das Verhalten für `every Node`.

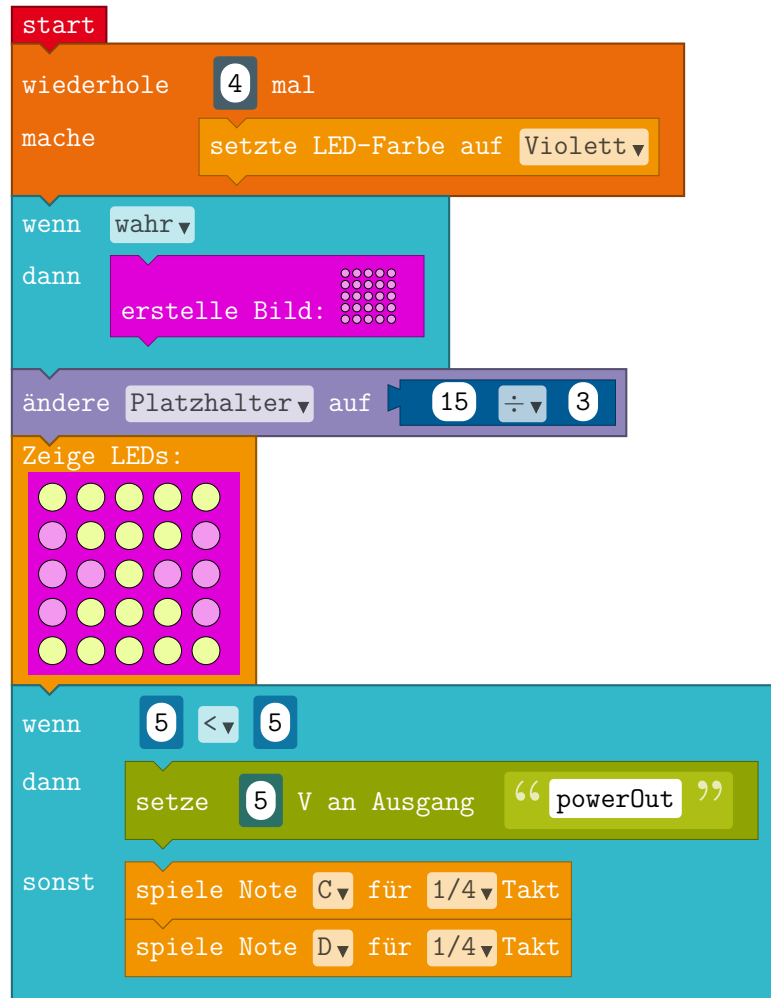
```
1 | \documentclass{standalone}
2 | \usepackage{tikzcodeblocks}
3 | \begin{document}
4 |   \begin{tikzpicture}[codeblocks] Hello World;
5 |     \node[mathe]{Hello World;};
6 |   \end{tikzpicture}
7 | \end{document}
```

²<https://www.ctan.org/pkg/scratch?lang=de>.

³In dieser Dokumentation wird die Bezeichnung **Node** für die Knoten in TikZ verwendet.

2 Beispielcode

Eine Beispiel zeigt Ergebnisse der Nutzung des Pakets. Der Quellcode zur Erstellung der Vektorgrafik folgt unter dem Beispiel.



```

1 | \begin{tikzpicture}[codeblocks , openroberta , scale=.90]
2 |
3 | \node[start , pinlow](start){start};
4 | \schleife[unter={start}{0}{0}]{wiederhole}{\intbox{4} mal}{\node[
5 |   aktion]{setzte LED-Farbe auf \dropdown{Violett}};}{schl1}
6 | \wenndann[unter={schl1}{0}{0}]{\dropdown{wahr}}{\node[bild]{
7 |   erstelle Bild: \bild[0.3]{\emptyled}};}{verz1}
8 | \node[variablen , unter={verz1}{0}{0}](plz1){ändere \dropdown{
9 |   Platzhalter} auf \tikz\node[mathe , boden , keinezacken , puzzleteil
10 | ]{\intbox{15}\dropdown{\$div\$}\intbox{3}};};
11 | \node[aktion , unter={plz1}{0}{0}](bild1){Zeige LEDs:\
12 | \bild{
13 |   \X \X \X \X \X \\
14 |   \0 \X \X \X \0 \\
15 |   \0 \0 \X \0 \0 \\
16 |   \0 \X \X \X \0 \\
17 |   \X \X \X \X \X \\
18 | }
19 | };

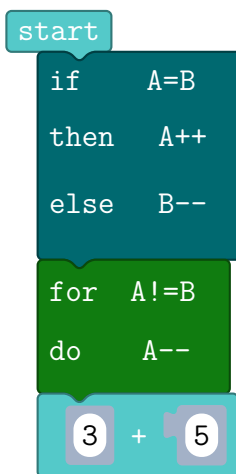
```

```

16
17 \wenndannsonst [unter={bild1}{0}{0}]
18 {\intbox{5}\dropdown{<}\intbox{5}}
19 {\node[sensor]{setze \intbox{5}V an Ausgang \stringbox{powerOut
    }};}
20 {\node[aktion,draw,](akt1){spiele Note \dropdown{C} für \dropdown
    {1/4}\,Takt};
21 \node[aktion,draw,pinhigh,unter={akt1}{0}{0}](akt2){spiele Note
    \dropdown{D} für \dropdown{1/4}\,Takt};}
22 {wds1}
23
24 \end{tikzpicture}

```

2.1 English Codeexample



```

1 \begin{otherlanguage}{english}
2 \begin{tikzpicture}[codeblocks]
3 \moveindent{
4 \node[start,pinlow](one){start};
5 }
6 \ifthenelseblocks [under={one}{1}{0}]{A=B}{\node{A++};}{\node{B
    -{-}-};}{two}
7 \loopblocks [under={two}{0}{0}]{for}{A!=B}{\node{A-{-}-};}{three}
8 \node [under={three}{0}{0},aktion,pinhigh]{\intbox{3}+\intbox[
    puzzlepiece]{5}};
9 \end{tikzpicture}
10 \end{otherlanguage}

```

3 Bausteine und Befehle






































3.1 Baustein-Klassen nach PXT-Calliope Editor

Die Klassen der Programmierbausteine werden nach folgendem Muster als TikZ-Style angegeben.

```
\node[STYLE] {Inhalt};
```

`openroberta`

Die Defaulteinstellung orientiert sich an den Farben und Formen des Calliope-PXT-Editors. Mithilfe des Stils `openroberta` können alternative Farb- und Formdefinitionen geladen werden, die sich an der NEPO-Umgebung von Open-Roberta orientieren. Der Stil kann auch für eine ganze TikZ-Umgebung verwendet werden.

Stilname	pxt (Standard)	openroberta
grundlage		
eingabe		
schleife		
logik		
musik		
led		
platzhalter		
mathe		
funk		
motor		
zeichenkette		
spiel		
bild		
pins		
konsole		
steuerung		
bluetooth		
start		
aktion		
sensor		

Stilname	pxt (Standard)	openroberta
kontrolle		
liste		
farbe		
bild		
variablen		
funktion		
nachricht		

! Hinweis: In PXT und im Open-Roberta-Editor heißen die entsprechenden Einträge für Zeichenketten „Texte“. Der Style „Text“ ist von TikZ jedoch bereits intern belegt und wird hier deswegen als „zeichenkette“ verwendet. Standardmäßig wird das Paket mit der PXT-Farbinformation geladen, um die Farben des PXT-Editors für Calliope zu verwenden.⁴ Die Stile unterscheiden sich voneinander nur durch ihre Farben. `openroberta` lädt standardmäßig noch den Stil `eckig`.

3.2 Skalierung

Bsp: Skalierung Die Skalierung des gesamten Bildes ist mit dem TikZ-Boardmittel `scale` möglich.

```

1 \begin{tikzpicture}[codeblocks , scale=0.7]
2   \node[variablen]{Hello World;};
3 \end{tikzpicture}
4 %
5 \begin{tikzpicture}[codeblocks , scale=1.3]
6   \node[variablen]{Hello World;};
7 \end{tikzpicture}

```

3.3 Farben

3.3.1 Lokale Farbänderung

Lokale Überschreibungen sind – wie üblich – durch Angabe einer Farbe bei den entsprechenden Node-Attributen möglich.

Bsp: Lokale Farbänderung

```
\node[fill=black,draw=red]{schwarzer Hintergrund - roter Rahmen}
```

⁴Damit beide Stile problemlos ineinander überführt werden können, ohne dass bestimmte Stile in anderen Kontexten undefiniert sind, wurden einige Stile doppelt zugeordnet: `pxt-grundlagen = openroberta-start`, `pxt-musik = openroberta-aktion`, etc.

3.3.2 Globale Farbänderung

`\setcolor` Der Befehl `\setcolor{<farbreferenz>}{<hexfarbcode>}` lässt zu, Farben dokumentenweit umzudefinieren.

Bsp: Globale Farbänderung

Alte Farbe

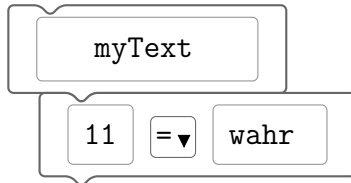
```
\setcolor{logik-color}{ff0000}
```

Neue Farbe!

3.3.3 Keine Farben (print)

Mit der Paketoption `\usepackage[print]{tikzcodeblocks}` lassen sich alle Farben entfernen. Es werden lediglich Umrisse, sowie ausgefüllte LEDs gezeichnet. Die Schriftfarbe ist schwarz.

Bsp: print



3.4 Boxen

Boxen werden innerhalb von Codeblöcken verwendet, um bestimmte Platzhalter und Datentypen zu kennzeichnen. Die Farben werden dabei teilweise in Abhängigkeit zum Parent (durch Durchsichtigkeit) gesetzt.

`\dropdown`
`\intbox`
`\stringbox`
`\boolbox`

Code	PXT	Open Roberta
<code>\dropdown{Dropdown}</code>	Dropdown ▼	Dropdown ▼
<code>\intbox{5}</code>		
<code>\stringbox{Text}</code>		
<code>\boolbox{wahr}</code>		

`\intbox`, `\stringbox`, `\boolbox` haben alle als optionales Argument die Möglichkeit Stile hinzuzufügen.

Bsp: optionales Argument bei Boxen



```
\intbox{42}
```



```
\intbox[puzzleteil]{42}
```

Bsp: Boxen mit Kapselung



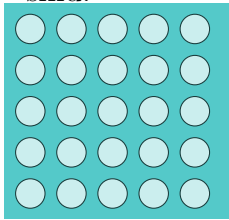
```
\node[eingabe]{Meine \dropdown{Dropdown}-Box mit Wert \intbox{5}};
```


! **Hinweis:** Bei der Verwendung des Stils `openroberta` ändern sich auch entsprechend die Farben der int-, string- und boolboxen.

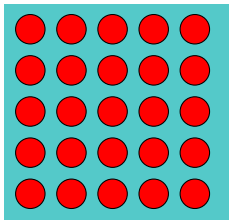
3.5 Bilder/LED-Matrix


`\bild` Mit `\bild[skalierungsfaktor]{Inhalt}` lassen sich LED-Matrizen setzen. Es erwartet einen Tabelleninhalt. Jede Zeile muss entsprechend per `\\` beendet werden. Zeilen und Spalten können dabei unbegrenzt sein. Dabei gelten weiterhin folgende Befehle:

`\emptyled` • `\emptyled` setzt eine 5×5 LED-Matrix, bei der alle LEDs ausgeschaltet sind.



`\fullled` • `\fullled` setzt eine 5×5 LED-Matrix, bei der alle LEDs angeschaltet sind.



`\X` • `\X` repräsentiert darin eine angeschaltete LED 

`\0` • `\0` repräsentiert darin eine ausgeschaltete LED 

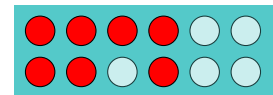
• Die Kombination von `\bild{}`, `\X`, und `\0` ergibt schließlich alle möglichen LED-Matrizen:

Bsp: Beliebige Matrix

```

1 | \bild{
2 |   \X \X \X \X \0 \0 \\
3 |   \X \X \0 \X \0 \0 \\
4 | }

```

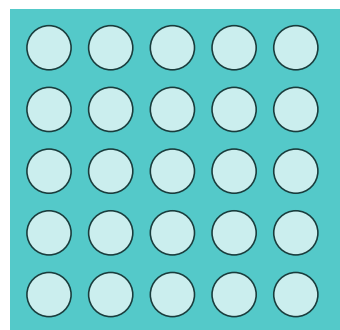


`\bild` • Mit dem optionalen Argument kann ein Skalierungsfaktor angegeben werden.

Bsp: Bildskalierung



`\bild[0.4]{\emptyled}`



`\bild[1.5]{\emptyled}`

! **Hinweis:** `\X`, `\0` sowie `\emptyled` und `\fullled` können nur innerhalb des `\bild`-Kommandos verwendet werden.

3.6 Strukturen

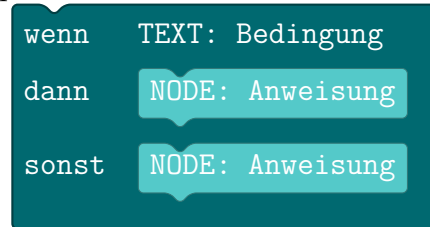
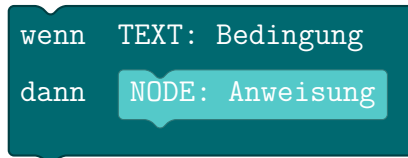
Strukturen helfen, die Positionierung von Nodes zu vereinfachen. Dafür können Verzweigungen und Schleifen verwendet werden. Damit muss nur noch in seltenen Fällen eine manuelle Positionierung von Nodes vorgenommen werden. Intern werden Tabellen verwendet.

3.6.1 Verzweigungen

`\wenndann` Über den Befehl `\wenndann[\langle TikZ-Stil \rangle]{\langle TEXT: Bedingung \rangle}{\langle NODE: Anweisung \rangle}{\langle TEXT: Nodename \rangle}` wird die Kontrollstruktur Verzweigung abgebildet. Der `logik-/bzw. kontrolle`-Stil wird automatisch gesetzt. Das letzte Argument ist die Bezeichnung des eigenen Nodennamen, damit nachfolgende Codeblöcke dies bei ihrerer Positionierung referenzieren können.

`\wenndannsonst` `\wenndannsonst[\langle TikZ-Stil \rangle]{\langle TEXT: Bedingung \rangle}{\langle NODE: Dann-Anweisung \rangle}{\langle NODE: Sonst-Anweisung \rangle}{\langle TEXT: Nodename \rangle}` verhält sich analog zu `\wenndann`, ist jedoch um einen Sonst-Block, der mit Nodes gefüllt wird, erweitert. Automatisch ergänzt werden die Wörter „wenn“, „dann“ und „sonst“.

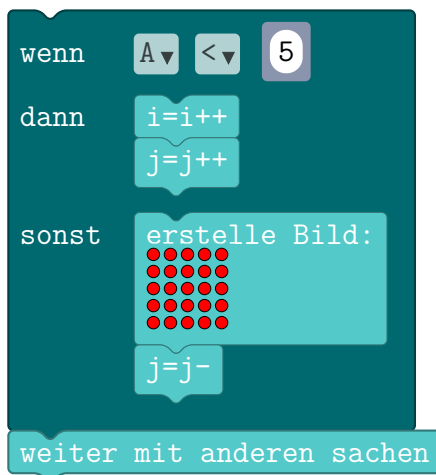
! **Hinweis:** Das optionale Argument ist oft notwendig, um die Verzweigung richtig in Relation zu vorherstehenden Blöcken zu positionieren. Siehe hierfür 4.1.



Deutsch: `\wenndann`
English: `\ifthenblocks`

`\wenndannsonst`
`\ifthenelseblocks`

Bsp: Verzweigungen



```
1 || \begin{tikzpicture}[codeblocks]
2 || \wenndannsonst[draw]
```

```

3 | {\dropdown{A}~\dropdown{<}\,\intbox{5}} %wenn
4 | { \node[aktion](akt1){i=i++};
5 |   \node[farbe,unter={akt1}{0}{0}](akt2){j=j++};} %
   |   dann
6 | { \node[aktion,](akt1){erstelle Bild:\
7 |   \bild[0.4]{\fullled}
8 |   };
9 |   \node[farbe,unter={akt1}{0}{0}](akt2){j=j--};
10 | } % sonst-ende
11 |   {eins}; % eigener Name
12 | \node[aktion,unter={eins}{0}{0}]{weiter mit anderen
   |   sachen};
13 | \end{tikzpicture}

```

! **Hinweis:** Sollen Nodes in Textfeldern gesetzt werden, so muss `\tikz` vorgeschoben werden.

3.6.2 Schleifen

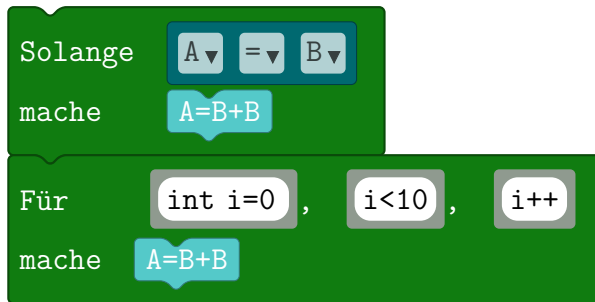
`\schleife` Die Schleife nach dem Muster `\schleife[<TikZ-Stil>]{<TEXT: Für/solange/etc.>}{<TEXT: Bedingung>}{<NODE: Anweisung>}{<TEXT: Nodename>}` ist eine weitere vordefinierte Struktur. Automatisch ergänzt wird der Begriff „mache“.

Bsp: Muster: Schleifen

TEXT: Für/solange/etc TEXT: Bedingung
 mache NODE: Anweisung

deutsch `\schleife[<STIL>]{<PRE>}{<BED>}{<ANW>}{<NAME>}`
 english `\loopblocks[<STYLE>]{<PRE>}{<COND>}{<INST>}{<NAME>}`

Bsp: Schleifen



```

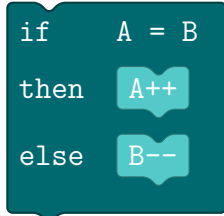
1 | \schleife[draw]{Solange}{\tikz\node[logik,keinezacken
   |   ]{\dropdown{A}~\dropdown{=}~\dropdown{B}}};}{
2 |   \node[aktion]{A=B+B};}{schl1}
3 |
4 | \schleife[draw,unter={schl1}{0}{0}]{Für}{\intbox{int i
   |   =0}, \intbox{i<10}, \intbox{i++}}{
5 |   \node[aktion]{A=B+B};}{schl2}

```

3.6.3 Branches, loops and the english language

If you want to use this package in an english document just load `\usepackage[english]{babel}` in the preamble. It will automatically set the outer words for branches and loops in english.

Bsp: English example `\usepackage[english]{babel}`



```

1 | \begin{otherlanguage}{english}
2 |   \begin{tikzpicture}[codeblocks]
3 |     \wenndannsonst{A = B}{\node[aktion]{A++};}{\node[
4 |       aktion]{B-{}-};}{name}
5 |   \end{tikzpicture}
6 | \end{otherlanguage}

```

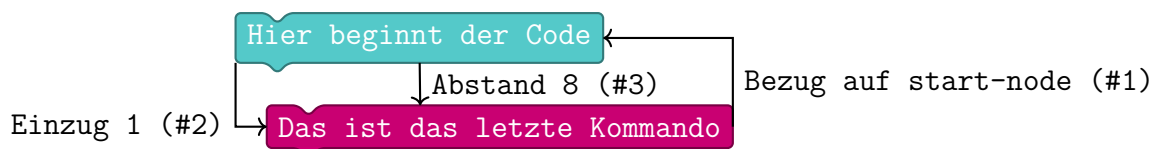
`\ifthenblocks` `\ifthenelseblocks` `\loopblocks` To fit the commands to the english language please you can use `\ifthenblocks`, `\ifthenelseblocks` and `\loopblocks`. Notice the added „blocks“, because the \LaTeX Command `\ifthenelse` is already used by the `ifthen`-Package.

english	german
<code>\ifthenblocks</code>	<code>\wenndann</code>
<code>\ifthenelseblocks</code>	<code>\wenndannsonst</code>
<code>\loopblocks</code>	<code>\schleife</code>

4 Positionierung der Nodes

4.1 Manuelles Positionieren

unter Mit konsequenter Verwendung der Strukturen ist manuelles Einrücken selten notwendig. Jedoch ist das Aneinanderketten der Nodes unabdinglich. Hierfür wird der Stil `unter={\langle NODE \rangle}{\langle X-Einzug-Faktor \rangle}{\langle Y-Einzug-Faktor \rangle}` verwendet. Hierbei wird der Einzug als Ankerpunkt jeweils relativ zum Vorgänger gesetzt.



```

1 | \node[grundlage] (drueber) {Hier beginnt der Code};
2 | \node[eingabe, unter={drueber}{1}{8}] (drunter){Das
   |   ist das letzte Kommando};

```

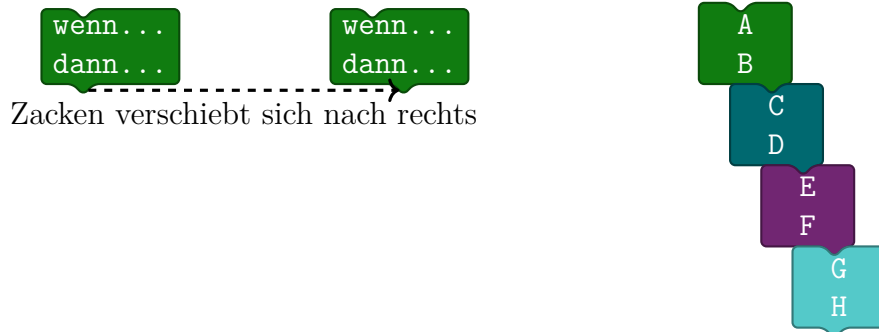
! Hinweis: Dank der Verwendung von Kontrollstrukturen (siehe 3.6) ist manuelles Einrücken in der Regel nicht notwendig.

4.1.1 Verschieben der Zacken bei manueller Einrückung

`\einruecken` Über das Kommando `\einruecken{\langle nodes \rangle}` lässt sich der untere Zacken eines Nodes um genau einen Einzug verschieben (siehe: 4.1;). Die obere Ausbuchtung bleibt an ihrem normalen Platz.

`\moveindent` The english equivalent to `\einruecken` is `\moveindent`.

Bsp: Verschachteltes Einrücken

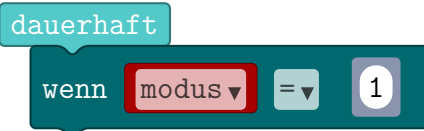


```

1 | \begin{tikzpicture}[codeblocks ,minimum width=1.2cm]
2 | \einruecken{\node[kontrolle](eins){A\ B};
3 |   \einruecken{\node[logik ,unter={eins}{1}{0}](zwei){C\ D};
4 |     \einruecken{\node[mathe ,unter={zwei}{1}{0}](drei){E\ F};}
5 |       \node[aktion ,unter={drei}{1}{0}](vier){G\ H};
6 |   }
7 | }
8 | \end{tikzpicture}

```

Bsp: einruecken



```

1 | \einruecken{
2 |   \node[grundlage ,pinlow](grund){dauerhaft};
3 | }
4 | \node[logik ,unter={grund}{1}{0}](wenn1){wenn \tikz\node
   |   [platzhalter ,boden ,keinezacken]{\dropdown{modus}}; \
   |   dropdown{=} \intbox{1} };

```

4.2 Automatisches Einrücken

Siehe Strukturen 3.6!

5 Dekorationen

5.1 Puzzleoptik

- `robertashape` Um die Verzahnung der einzelnen Elemente darzustellen, wird standardmäßig eine Puzzleoptik verwendet, welche die *vertikale* Beziehung der Bausteine zueinander verdeutlicht. Sie ist über die Form `robertashape` definiert.
- `puzzleteil` Soll die *horizontale* Beziehung von Bausteinen betont werden, so kann die Form `puzzleteil` verwendet werden.
- `keinezacken` Mit dem Stil `keinezacken` lassen sich alle vordefinierten Zacken entfernen. Dies ist v. A. bei verschachtelten Nodes notwendig. `keinezacken` ist ein Alias für `rectangle`.
- `eckig` Der Stil `eckig` entfernt alle runde Ecken und orientiert sich damit stärker an dem Erscheinungsbild von Open Roberta.
- `pinlow` Für Start- und Endbausteine sind die Formen `pinlow` und `pinhigh` definiert.
- `pinhigh`









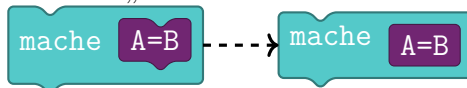
Shape	deutsch	english
		
	pinlow	pinlow
	pinhigh	pinhigh
	keinezacken	nopins
	eckig	square
	eckig, keinezacken	square, nopins
	puzzleteil	puzzlepiece
	puzzleteil, eckig	puzzlepiece, square




Tabelle 4: Übersicht über Blockformen

Bsp: `keinezacken` Der Stil „keinezacken“ ist z. B. bei verschachtelten Nodes notwendig:



```
1 || \node[aktion]{mache \tikz\node[mathe,keinezacken]{A=B};};
```



5.2 Symbole und kleine Elemente

- `\usb` `\usb` setzt ein .
- `\farbe` `\farbe{color}` setzt ein Quadrat mit der angegebenen Farbe . Zu verwenden auch in Nodes: 

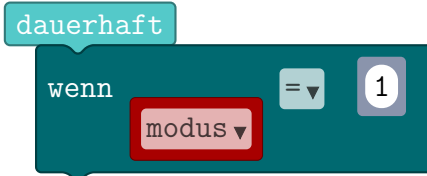
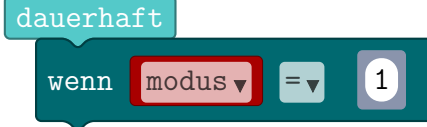
6 FAQ

6.1 Wie kann ich für ein komplettes Dokument eine Farbe umdefinieren?

`\setcolor`

Ist: 
Soll: 
Lösung: `\setcolor{logik-color}{ff0000}`

6.2 Eine nested Box ist zu hoch/zu niedrig

Ist: 
Soll: 
Lösung: Verwende den Hilfstyle boden:

```
1 | \einruecken{  
2 |   \node[grundlage, pinlow](grund){dauerhaft};  
3 | }  
4 | \node[logik, unter={grund}{1}{0}](wenn1){wenn \tikz  
   |   \node[platzhalter, boden, keinezacken]{\dropdown{  
   |   modus}}; \dropdown{=} \intbox{1} };
```

6.3 Ich will einen Node innerhalb eines Nodes setzten

Verwende eine verschachtelte TikZ-Umgebung, z. B. mit dem `\tikz`-Befehl:

außen: 

```
1 | \node[logik]{außen:  
2 |   \tikz\node[aktion]{innen};  
3 | };
```

6.4 Mein Block ist sehr klein und deswegen verformt

Ist: 
Soll: 

Lösung: Erweitere den Inhalt des Nodes um Whitespace/Phantome oder setze minimum width für den Node.

```
1 || \node[logik](one){2~~};  
2 || \node[logik, minimum width=1cm](one){3};
```

7 Mehr Beispielcode

7.1 Bsp: Calliope Smart Home

The image shows a sequence of Scratch-style code blocks for a Calliope Smart Home project:

- beim Start** (When started):
 - ändere modus auf 1** (Set mode to 1)
 - serial redirect to TX C17, RX C16, at baud rate 9600** (Configure serial communication)
- wenn Knopf A gedrückt dann** (When button A is pressed):
 - serial write line "test_line"** (Send a test message)
 - Warte 300000 μ** (Wait 300,000 microseconds)

serial on data received #

mach

ändere befehl auf serial read until #

wenn befehl = "aus"

dann

schreibe analogen Pin P2 auf 0

ändere modus auf 0

wenn befehl = "ein"

dann

ändere modus auf 1

wenn befehl = "solar"

dann

schreibe analogen Pin P2 auf 1023

ändere solar auf 1

wenn befehl = "solar_aus"

dann

schreibe analogen Pin P2 auf 0


```

1 \begin{tikzpicture}[codeblocks]
2 \einruecken{\node[grundlage, pinlow] (start){beim Start};}
3 \node[platzhalter, unter={start}{1}{0}] (plz1){ändere \dropdown{
4 \node[konsole, unter={plz1}{0}{0}] (ser1) {serial \redirect to
5 \end{tikzpicture}
6
7 \begin{tikzpicture}[codeblocks]
8 \wenn dann [eingabe]
9 {Knopf \dropdown{A} gedrückt}
10 {
11 \node[konsole] (ser1) {serial write line \stringbox{test\_line
12 \node[steuerung, unter={ser1}{0}{0}] (wait1){Warte  $\mu$  \
13 }
14 {buttonA}
15 \end{tikzpicture}
16
17
18 \begin{tikzpicture}[codeblocks]
19
20 \schleife[konsole]{\usb{}}{serial on data recived \usb{} \
21 \dropdown{\#}}{
22 \node[platzhalter, unter={start}{1}{0}] (plz1) {ändere \
23 \dropdown{befehl} auf \tikz\node[konsole, boden, keinezacken,
24 puzzleteil,]{\usb{}} serial read until \dropdown{\#}};};
25 \wenn dann [unter={plz1}{0}{0}]{\tikz\node[pins, boden, keinezacken
26 , puzzleteil]{\dropdown{befehl}}; \dropdown{=} \stringbox{aus
27 }}{
28 \node[pins] (pin1) {schreibe analogen Pin \dropdown{P2} auf
29 \intbox{0}};
30 \node[pins, unter={pin1}{0}{0}] (pin2) {ändere \dropdown{
31 modus} auf \intbox{0}};
32 }{wenn1}
33 \wenn dann [unter={wenn1}{0}{0}]{\tikz\node[pins, boden,
34 keinezacken, puzzleteil]{\dropdown{befehl}}; \dropdown{=} \
35 \stringbox{ein}}{
36 \node[pins] (pin2) {ändere \dropdown{modus} auf \intbox{1}};
37 }{wenn2}
38 \wenn dann [unter={wenn2}{0}{0}]{\tikz\node[pins, boden,
39 keinezacken, puzzleteil]{\dropdown{befehl}}; \dropdown{=} \
40 \stringbox{solar}}{
41 \node[pins] (pin3) {schreibe analogen Pin \dropdown{P2} auf \
42 \intbox{1023}};
43 \node[pins, unter={pin3}{0}{0}] (pin2) {ändere \dropdown{solar
44 } auf \intbox{1}};
45 }{wenn3}
46 \wenn dann [unter={wenn3}{0}{0}]{\tikz\node[pins, boden,
47 keinezacken, puzzleteil]{\dropdown{befehl}}; \dropdown{=} \
48 \stringbox{solar\_aus}}{
49 \node[pins] (pin4) {schreibe analogen Pin \dropdown{P2} auf \
50 \intbox{0}};
51 }{wenn4}

```



```

73         \0 \0 \0 \0 \0 \\  

74         \0 \0 \0 \X \0 \\  

75         \0 \0 \X \X \0 \\  

76         \0 \X \X \X \0 \\  

77         \X \X \X \X \0 \\  

78     }];}{wenn4}  

79  

80     \wenddann[unter={wenn3}{0}{0}]{\tikz\node[eingabe,boden,  

        keinezacken,puzzleteil]{Lichtstärke}; \dropdown{>}\  

        intbox{200} \dropdown{und} \tikz\node[eingabe,boden,  

        keinezacken,puzzleteil]{Lichtstärke}; \dropdown{\leq}\  

        intbox{255}]{\node[pins](pin5){schreibe analogen Ping \  

        dropdown{P1} auf \intbox{0}}};  

81     \node[grundlage,unter={pin5}{0}{0}](bild5){zeige LEDs\<\  

82     \bild{  

83         \0 \0 \0 \0 \X \\  

84         \0 \0 \0 \X \X \\  

85         \0 \0 \X \X \X \\  

86         \0 \X \X \X \X \\  

87         \X \X \X \X \X \\  

88     }];}{wenn5}  

89     }%dann außen  

90     {\node[pins](pin6){schreibe analogen Ping \dropdown{P1} auf \  

        intbox{0}}};  

91     \node[grundlage,unter={pin6}{0}{0}](bild5){zeige LEDs\<\  

92     \bild{  

93         \X \0 \0 \0 \X \\  

94         \0 \X \0 \X \0 \\  

95         \0 \0 \X \0 \0 \\  

96         \0 \X \0 \X \0 \\  

97         \X \0 \0 \0 \X \\  

98     }];}%sonst außen  

99     {wenn}  

100 }{schleife1}  

101 \end{tikzpicture}

```