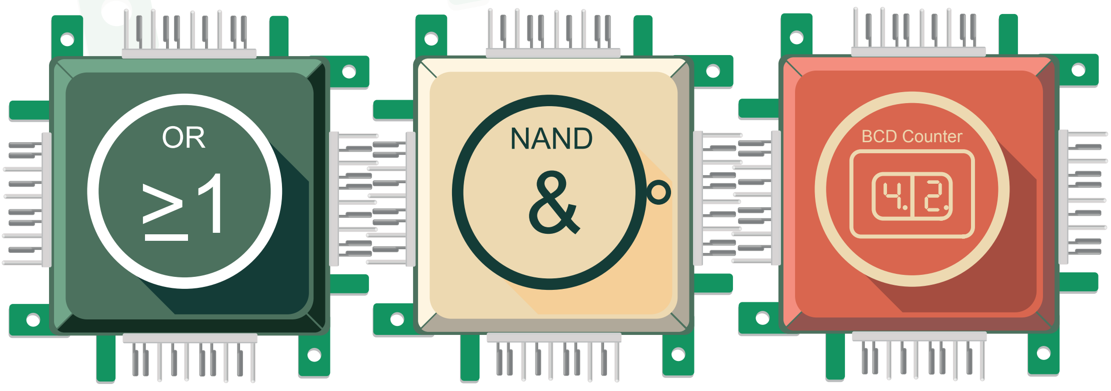
# Logic Set

Experimental kit by Brick'R'knowledge
Experimentierkasten von Brick'R'knowledge



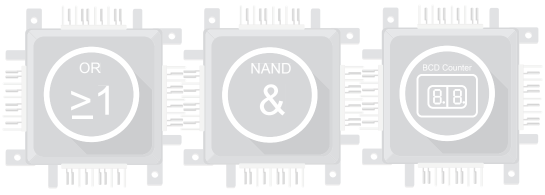
www.brickrknowledge.com

# Imprint

Brick'R'knowledge Logic Set Manual
Rev. 1.0
Date: 07.04.2017

**ALLNET® GmbH Computersysteme**
Brick'R'knowledge
Maistraße 2
D-82110 Germering

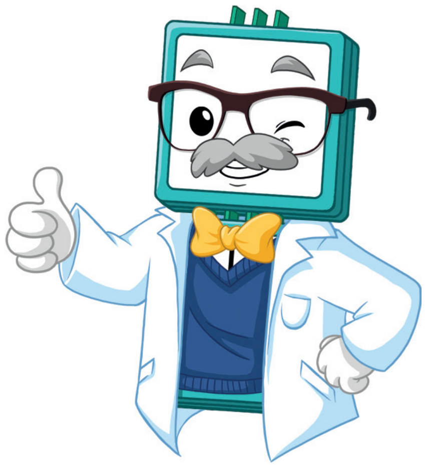
All of the information that are included in this manual has been put together with great care and to the best of our knowledge. Even though it errors can occur. We are always thankful for any feedback about possible mistakes. Please send your feedback to: info@brickrknowledge.de.



# Contents

## Preface

The Brick'R'knowledge experimental system has been launched at the HAM Radio faire on the 28th june of 2014 by Rolf-Dieter Klein (ham radio name: DM7RDK). The special thing of our electronic sets is that all of the bricks are connected via a plug system in which all parts are structually identical (hermaphrodite). This way it is possible to also realize tricky circuits. Furthermore you can plug the individual bricks together in different angles. For the return of the ground (0 Volt) there are even two contacts! This way you can build compact circuits in which the ground brick anables a stable voltage supply of the bricks. Another special feature is that you can explain and document these circuits in an easy way.



I wish you lots of fun with the Logic Set

*Rolf-Dieter Klein*

## List of references:

- Elektronik-Kompendium by Patrick Schnabel: http://www.elektronik-kompendium.de/sites/dig/index.htm
- Elektroniktutor by Detlef Mietke: http://www.elektroniktutor.de/digital1.html
- Wikipedia: https://www.wikipedia.de

You will find comprehensive basic knowledge about digital circuit technology and the laws of Boolean algebra.

# 1. Safety instructions

Note: Never connect the bricks directly to the main power supply (115V/230V). There might be danger to life!

Please only use the included power supply bricks. The voltage of our power supply is 9V, which is not a health hazard. Please

also ensure that no openly wires are in contact with the main power outlets. Otherwise there might be a danger of hazardous electric shocks. Never look straight into LEDs, since this may damage your eye retina. Please remove the power supply brick everytime you finished expimenting, to avoid the risk of an electric fire.

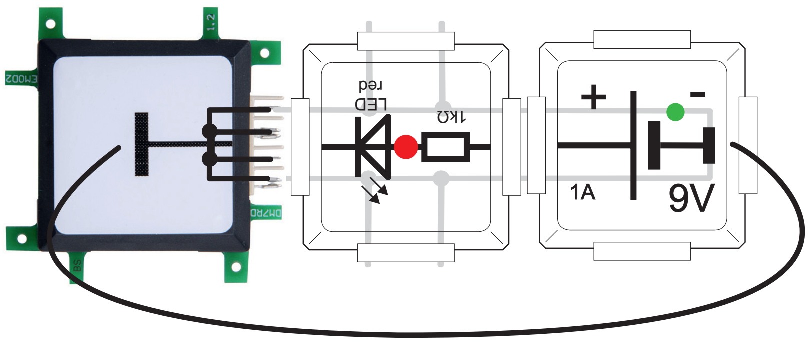
Do not swallow parts of the electronic set. If you did please contact a doctor immediately!

# 2. Basics of the Brick'R'knowledge system

## 2.1 The ground brick

One of the most important bricks is the so called ground brick. The ground brick has one connector with for contacts. Usually the middle two contacts are used for signal or power connection. But the outer contacts are intended for the so called ground level. Which means technically a level of 0V. The ground brick connects the both inner contacts with the outer contacts. Therefore it is possible to allow for a current return flow towards the 0V of a power supply invisible to the schematic symbols outside. The power supply of course must also beconnected at one pole (usually the minus pole) to the ground using the ground brick.

Figure 1: The ground brick



## 2.2 Die Spannungsversorgung

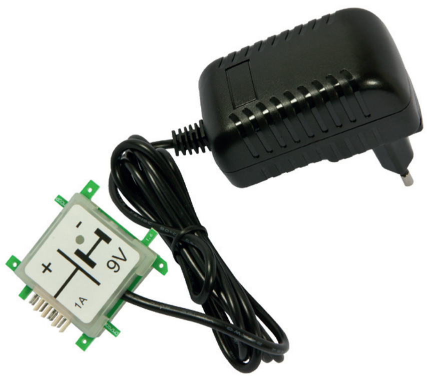


Figure 2: The power supply

The power supply of the Logic Set is achieved by the included 9V power supply brick. The power supply provides a stabilzed direct-current voltage of 9V and a maximum current of 1A. If overload occurs, the power supply turns off, meaning that it is short-circuit-proof.

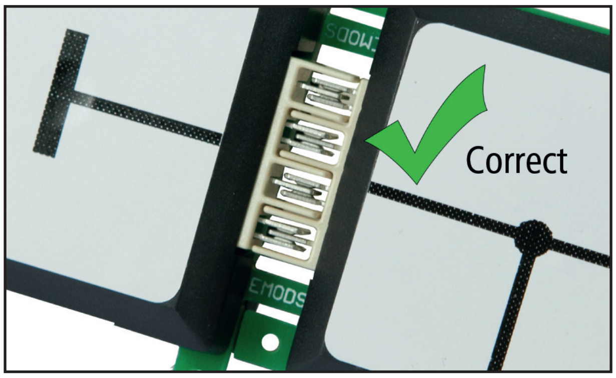
Gates, flip-flops and counter bricks are active components. For those bricks we use the clasical CMOS technology (for details see chapter 4.4) and they need a different power supply. Theoretically you can plug them to any stabilized direct-circuit power source in the range of +5V to +15V. Even though we recommend to use the included power supply brick.

Please always check the circuit before adding the power supply brick. After every experiment, please remove the power supply from the main supply.

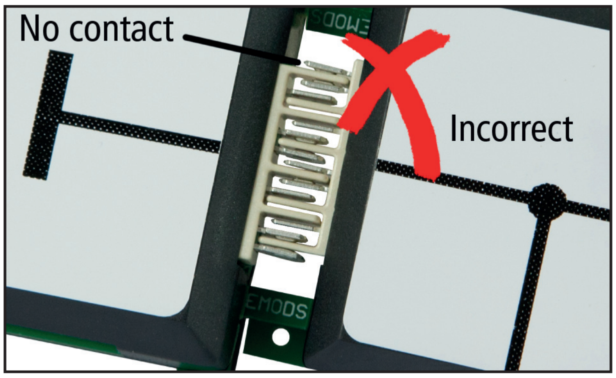
Alternatively we offer a battery brick (ALL-BRICK-0001), using a 9V block battery.

## 2.3 The connectors

Please connect the bricks correctly, otherwise the connection will have an open or short circuit.



Plugged together correctly



Not correctly plugged together

Figure 3: The connectors

The image on the left shows an example of a correctly plugged connection. The connection consists of small contacts, that stuck mechanically and transmit the electrical energy.

The image on the right shows an incorrect connection. As you can see, the metal contacts are interrupted by the plastic pins. This allows no current flow.

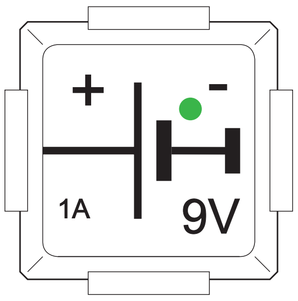
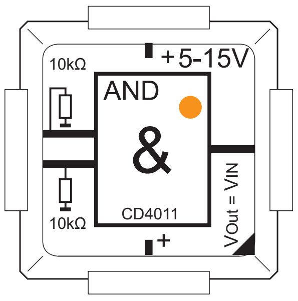
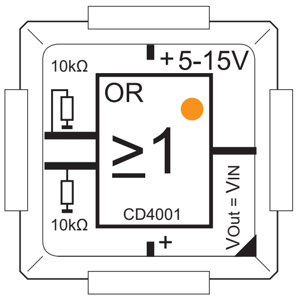
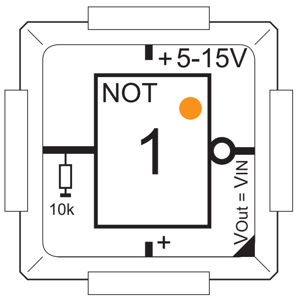
Caution: It is important to check the correct connection of the bricks. If they are not connected correctly this can lead to a short circuit or a malfunction of the circuit. If the connection is not working correctly, the current takes the lowest resistance way back to the power source, which might result in a short circuit, because the only resistance that has to be overcome, is the internal resistance of the voltage source.

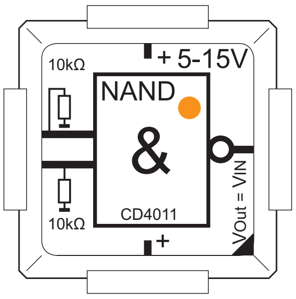
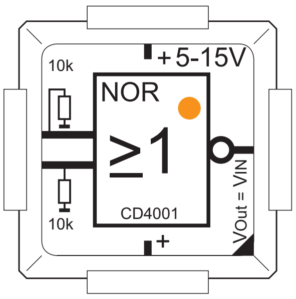
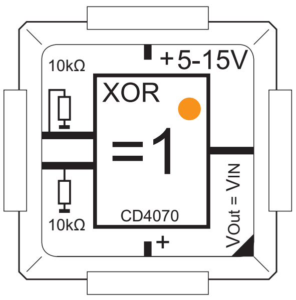
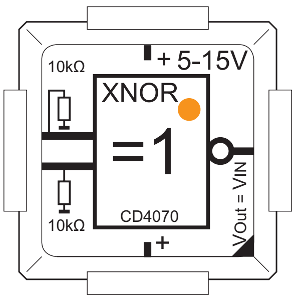
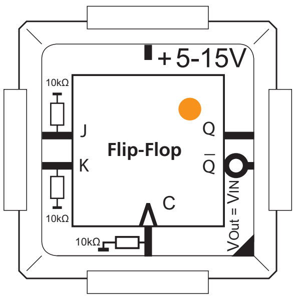
**Important: Always check the correct position of the contacts!**

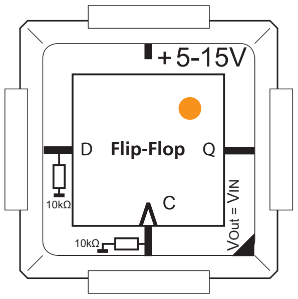
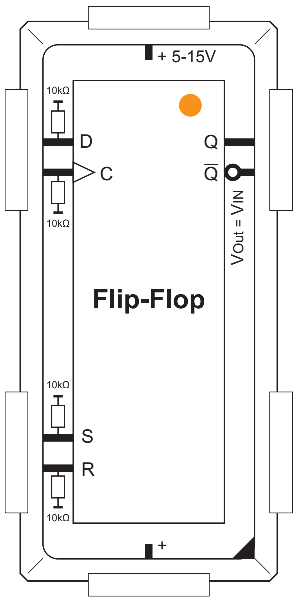
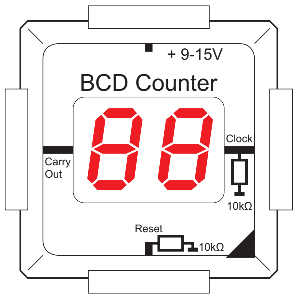
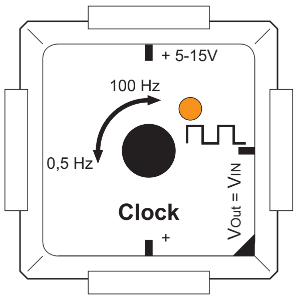
# 3. The Logic Set

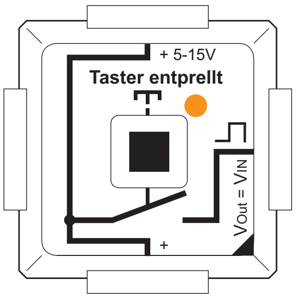
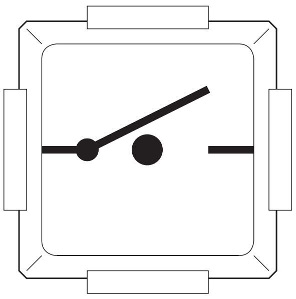
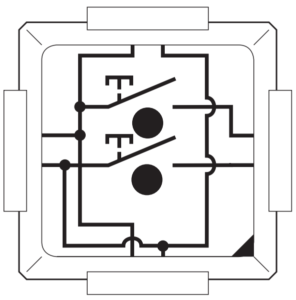
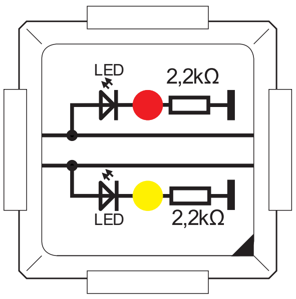
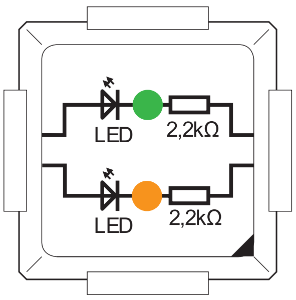
The task of the Brick'R'knowledge Logic Set is to teach the basics of the digital circuit technology in an easy and understandable way. This is done on one hand by this detailed, didactical manual and on the other hand by the practice-oriented plugging together and experimenting with the bricks. We start with easy logical linkings to half- and full-adders, to flip-flop- circuits and a BCD counter. As clock we use the famous Timer 555.

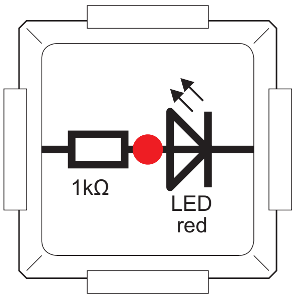
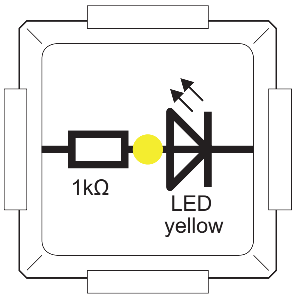
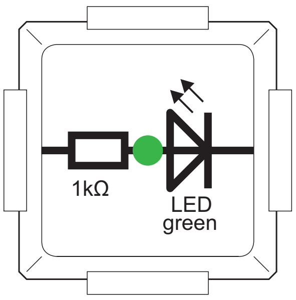
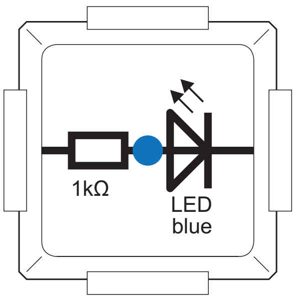
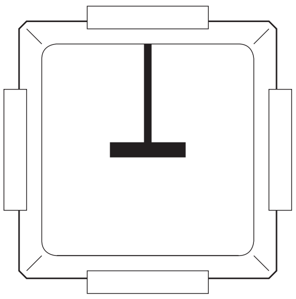
## 3.1 Bricks of the Logic Set

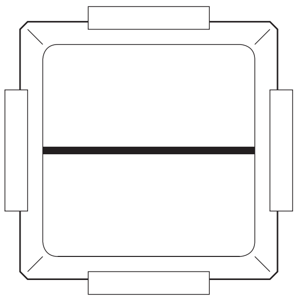
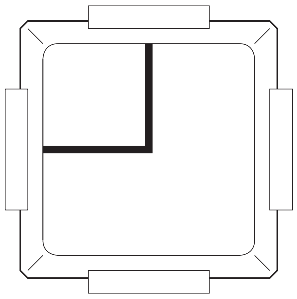
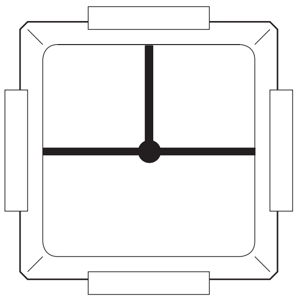
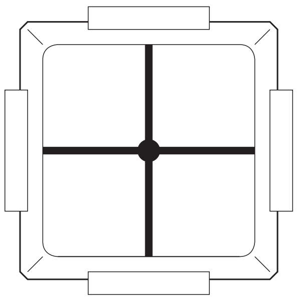
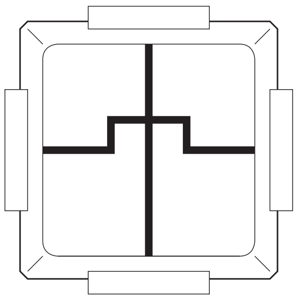
| Scheme | Qty. | Art.-Nr. / Brick ID | Short description |
|---|---|---|---|
|  | 1 | Art.-Nr.: 118627<br>Brick-ID: ALL-BRICK-0221 | **9V power supply**<br>The power supply adapter provides a stabilized DC voltage of 9V with a maximum short-circuit-proof current flow of 1A. The ground is connected to the minus pole, so that no further ground brick must be used on this pole to complete the circuit. A LED signals the correct insertion of the brick. To reduce the risk of electronic malfunctions, the power-supply-brick should be disconnected after an experiment. |
|  10kΩ<br>+5-15V<br>AND<br>&<br>10kΩ<br>CD4011<br>VOut = VIN<br>+ | 2 | Art.-Nr.: 128276<br>Brick-ID: ALL-BRICK-0437 | **AND gate**<br>This gate realises a logical AND linkage with two inputs based on 1/4 CD4011. The inputs are connected with ground via a 10kΩ pulldown resistor. Supply voltage: +5..15V. |
|  10kΩ<br>+5-15V<br>OR<br>≥1<br>10kΩ<br>CD4001<br>VOut = VIN<br>+ | 1 | Art.-Nr.: 128277<br>Brick-ID: ALL-BRICK-0438 | **OR gate**<br>This gate realises a logical OR linkage with two inputs based on 1/4 CD4011. The inputs are connected with ground via a 10kΩ pulldown resistor. Supply voltage: +5..15V. |
|  | 1 | Art.-Nr.: 135015<br>Brick-ID: ALL-BRICK-0634 | **Inverter**<br>The inverter negates the signal at the output. The inputs are connected with ground via a 10kΩ pulldown resistor. Supply voltage: +5..15V. |

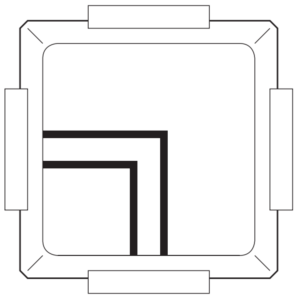
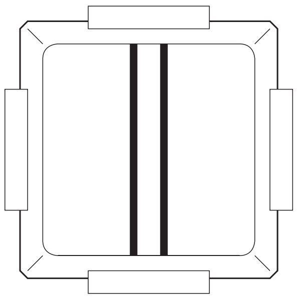
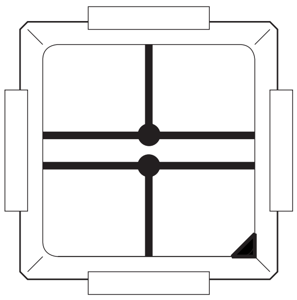
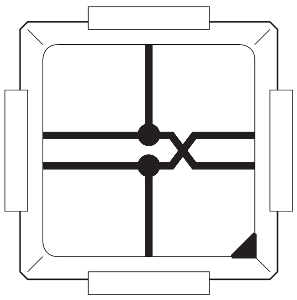
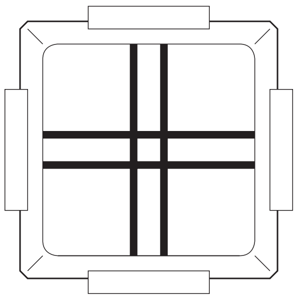
| Scheme | Qty. | Art.-Nr. / Brick ID | Short description |
|---|---|---|---|
|  10kΩ<br>+ 5-15V<br>NAND<br>&<br>CD4011<br>10kΩ<br>+<br>VOut = VIN | 1 | Art.-Nr.: 113684<br>Brick-ID: ALL-BRICK-0057 | **NAND gate**<br>This gate realises a logical NAND linkage with two inputs based on 1/4 CD4011. The inputs are connected with ground via a 10kΩ pulldown resistor. Supply voltage: +5..15V. |
|  10k<br>+ 5-15V<br>NOR<br>≥1<br>CD4001<br>10k<br>+<br>VOut = VIN | 2 | Art.-Nr.: 113685<br>Brick-ID: ALL-BRICK-0058 | **NOR gate**<br>This gate realises a logical NOR linkage with two inputs based on 1/4 CD4011. The inputs are connected with ground via a 10kΩ pulldown resistor. Supply voltage: +5..15V. |
|  10kΩ<br>+ 5-15V<br>XOR<br>=1<br>CD4070<br>10kΩ<br>+<br>VOut = VIN | 2 | Art.-Nr.: 128278<br>Brick-ID: ALL-BRICK-0439 | **XOR gate**<br>This gate realises a logical XOR linkage with two inputs based on 1/4 CD4011. The inputs are connected with ground via a 10kΩ pulldown resistor. Supply voltage: +5..15V. |
|  10kΩ<br>+ 5-15V<br>XNOR<br>=1<br>CD4070<br>10kΩ<br>+<br>VOut = VIN | 1 | Art.-Nr.: 128279<br>Brick-ID: ALL-BRICK-0440 | **XNOR gate**<br>This gate realises a logical XNOR linkage with two inputs based on 1/4 CD4011. The inputs are connected with ground via a 10kΩ pulldown resistor. Supply voltage: +5..15V. |
|  + 5-15V<br>10kΩ<br>J<br>Flip-Flop<br>Q<br>K<br>Q̄<br>C<br>10kΩ<br>10kΩ<br>VOut = VIN | 4 | Art.-Nr.: 113683<br>Brick-ID: ALL-BRICK-0056 | **JK flip-flop**<br>This JK flip-flop (one edge-controlled) is based on the 1/2 CD4027. The JK and clock inputs are connected with ground via a 10kΩ resistor. Supply voltage: +5..15V.<br><br>Notice: You should not connect any further load e.g. a LED to the clock input C. |

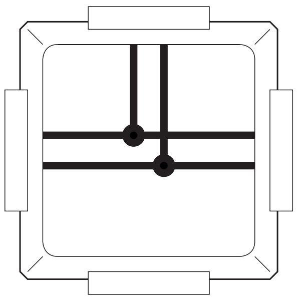
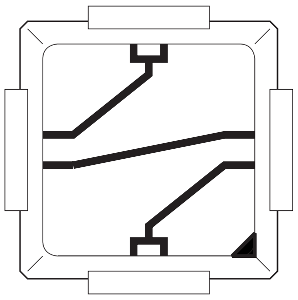
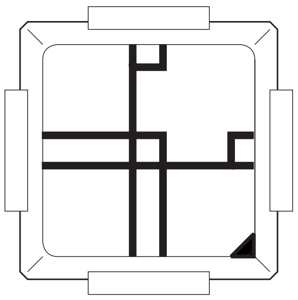
| Scheme | Qty. | Art.-Nr. / Brick ID | Short description |
|---|---|---|---|
|  +5-15V<br>D<br>Flip-Flop<br>Q<br>10kΩ<br>C<br>10kΩ<br>VOut = VIN | 4 | Art.-Nr.: 135012<br>Brick-ID: ALL-BRICK-0632 | **D flip-flop**<br>This is a D flip-flop (edge-controlled). The data and clock inputs are connected with ground via a 10kΩ pulldown resistor. Supply voltage: +5..15V.<br><br>Notice: You should not connect any further load e.g. a LED to the clock input C. |
|  + 5-15V<br>10kΩ<br>D<br>Q<br>C<br>Q̄<br>10kΩ<br>VOut = VIN<br>Flip-Flop<br>10kΩ<br>S<br>R<br>10kΩ<br>+ | 1 | Art.-Nr.: 135014<br>Brick-ID: ALL-BRICK-0633 | **D flip-flop with set and reset**<br>This is a D flip-flop (edge-controlled) with a set and reset input in the 2x1 format. All inputs are connected with ground via a 10kΩ pulldown resistor. Supply voltage: +5..15V.<br><br>Notice: You should not connect any further load e.g. a LED to the clock input C. |
|  + 9-15V<br>BCD Counter<br>Clock<br>Carry Out<br>10kΩ<br>Reset<br>10kΩ | 2 | Art.-Nr.: 135011<br>Brick-ID: ALL-BRICK-0631 | **BCD counter**<br>The BCD counter is based on a synchronous counter block CD4518. With this brick it is very easy to build a two-digit BCD counter. The clock and reset inputs are connected with ground via a 10kΩ pulldown resistor. For cascading several BCD counters, the carry out of the ten´s digit is lead through. Clock input and output are low active. Supply voltage: +9..15V. |
|  | 1 | Art.-Nr.:  137826<br>Brick-ID: ALL-BRICK-0643 | **Clock**<br>The clock brick is the clock source for the logic bricks. The circuit is based on the universal timer 555. You can change the square wave signal from ca. 0,5 Hz to 100 Hz by turning the poten-tiometer knob. To create single square wave signals you can also use the debounced push button. Voltage supply: +5..15V. |

| Scheme | Qty. | Art.-Nr. / Brick ID | Short description |
|---|---|---|---|
|  | 1 | Art.-Nr.: 137825<br>Brick-ID: ALL-BRICK-0641 | **Push button debounced**<br>With this special push button it is very easy to create a debounced clock signal for flip-flop circuits. Alternatively you can also use the clock brick. By using the debounced push button, there is enough time to understand the individual steps of a sequential circuit. The supply voltage is +5..15V. |
|  | 3 | Art.-Nr.: 113644<br>Brick-ID: ALL-BRICK-0017 | **Push button**<br>The push button is an electromechanical switch that allows a conductive connection only during pressing and holding the button. At the moment of release, the connection will open again and the button returns to its inital position. Note: This push button does not create a debounced signal. By using this brick within logic circuits it can lead to wrong pulses (see chapter 5.2, page 33). |
|  | 1 | Art.-Nr.: 137824<br>Brick-ID: ALL-BRICK-0642 | **Push button dual**<br>This brick contains 2 push buttons. With this brick it´s convenient to control gates or flip-flops. Additionally the two signals are seperated from each other. Connect the supply to the upper or lower connector to save time and wire bricks. |
|  | 1 | Art.-Nr.: 125693<br>Brick-ID: ALL-BRICK-0410 | **LED dual grounded, red/yellow & signal through-connected**<br>There are two LEDs (red and yellow) in this brick. Additionally there is one series resistor (2.2KOhm) used per piece that protects the LEDs. They are optimated for 2mA. Both of the resistors are grounded. Additionally the signals are transfered to the end. |
|  | 1 | Art.-Nr.: 125682<br>Brick-ID: ALL-BRICK-0409 | **LED dual grounded, green/orange & signal through-connected**<br>There are two LEDs (green and orange) in this brick. Additionally there is one series resistor (2.2KOhm) used per piece that protects the LEDs. They are optimated for 2mA. Both of the resistors are grounded. Additionally the signals are transfered to the end. |

| Scheme | Qty. | Art.-Nr. / Brick ID | Short description |
|---|---|---|---|
|  | 1 | Art.-Nr.: 113636<br>Brick-ID: ALL-BRICK-0009 | **LED red**<br>This is a red LED with an included series resistor of 1kΩ. The cathode is displayed by the bar at the LED symbol. |
|  | 1 | Art.-Nr.: 113638<br>Brick-ID: ALL-BRICK-0011 | **LED yellow**<br>This is a yellow LED with an included series resistor of 1kΩ. The cathode is displayed by the bar at the LED symbol. |
|  | 1 | Art.-Nr.: 113639<br>Brick-ID: ALL-BRICK-0012 | **LED green**<br>This is a green LED with an included series resistor of 1kΩ. The cathode is displayed by the bar at the LED symbol. |
|  | 1 | Art.-Nr.: 113637<br>Brick-ID: ALL-BRICK-0010 | **LED blue**<br>This is a blue LED with an included series resistor of 1kΩ. The cathode is displayed by the bar at the LED symbol. |
|  | 4 | Art.-Nr.: 113630<br>Brick-ID: ALL-BRICK-0003 | **Wire ground**<br>The ground is responsible for leading and returning the voltage. Put the ground brick on the end of each circuit in order to close it. It connects the middle connectors with the two external ground lines. |

| Scheme | Qty. | Art.-Nr. / Brick ID | Short description |
|---|---|---|---|
|  | 10 | Art.-Nr.: 113631<br>Brick-ID: ALL-BRICK-0004 | **Wire straight**<br>This straight connector brick connects two opposite bricks. |
|  | 8 | Art.-Nr.: 113632<br>Brick-ID: ALL-BRICK-0005 | **Wire corner**<br>You can connect your brick with the corner brick across the corner. |
|  | 8 | Art.-Nr.: 113633<br>Brick-ID: ALL-BRICK-0006 | **Wire T-crossing**<br>With the T-crossing brick you can connect your bricks like with each other like a "T". |
|  | 3 | Art.-Nr.: 113634<br>Brick-ID: ALL-BRICK-0007 | **Wire crossing connected**<br>The crossing brick connects all four connectors with each other. |
|  | 7 | Art.-Nr.: 113635<br>Brick-ID: ALL-BRICK-0008 | **Wire crossing not connected**<br>This brick connects the opposite connectors. Left to right and up to down without connected the two signals. |

| Scheme | Qty. | Art.-Nr. / Brick ID | Short description |
|---|---|---|---|
|  | 1 | Art.-Nr.: 113681<br>Brick-ID: ALL-BRICK-0054 | **Wire double corner**<br>This is a corner brick for two seperated signals. |
|  | 1 | Art.-Nr.: 113676<br>Brick-ID: ALL-BRICK-0049 | **Wire double straight**<br>This is a straight connector brick for two sepera-ted signals. |
|  | 2 | Art.-Nr.: 122443<br>Brick-ID: ALL-BRICK-0381 | **Wire T-crossing double left/right**<br>This is a dual T-crossing left/right which is used to seperate or connect signals. |
|  | 1 | Art.-Nr.: 113675<br>Brick-ID: ALL-BRICK-0048 | **Wire double crossed**<br>With this brick you can forward and cross sepe-rated signals. |
|  | 1 | Art.-Nr.: 113678<br>Brick-ID: ALL-BRICK-0051 | **Wire double crossing not connected**<br>This brick makes it possible to cross seperated signals without connecting them. |

| Scheme | Qty. | Art.-Nr. / Brick ID | Short description |
|---|---|---|---|
|  | **4** | Art.-Nr.: 113679<br>Brick-ID: ALL-BRICK-0052 | **Wire double T-crossing left**<br>This is a t-crossing left brick for seperated signals. Alternatively we also offer a Wire double t-crossing brick right (article number 113680, brick ID ALL-BRICK-0053). |
|  | **4** | Art.-Nr.: 122442<br>Brick-ID: ALL-BRICK-0380 | **Wire double duplexer left/right**<br>This is a duplexer left/right to seperate signals. |
|  | **5** | Art.-Nr.: 113677<br>Brick-ID: ALL-BRICK-0050 | **Wire double special**<br>This is a special wire brick to distribute sepera-ted signals. A typical use for this brick are logic circuits and circuits with controllers e.g. the Arduino brick. |

# 4. Basics of digital circuitry

## 4.1 What does digital mean?

The term „digital“ derives from Latin and means finger. Young children learn how to count and calculate by using their fingers. They serve as an elementary arithmetic unit. We will focus on the binary digital technology that only knows two possible signal states:

- logical zero „0“ or „L“ (English low) and logical one „1“
- or H (English high)

). In contrast to it there is the term „analogue“ (for example analogue vs. digital camera). We can easily explain the difference by comparing a slant with a staircase. While a slant has a continuous change of height a staircase has little discrete steps. Similar to digital information processing that differentiates only between logical zero „0“ and „1“, in analogue technology signals are processed value-continuously.

The main components of digital circuitries are logic gates like NOT, AND and OR. You can build all kinds of other gates, numerators, flip-flops etc. Complex circuits include storage switching circuits and processors as well as freely programmable logic components.

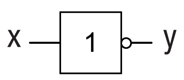
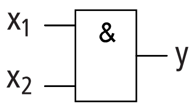
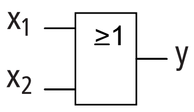
In case a high voltage level is indicated as „1“ and a low voltage level as „0“ we speak of positive logic, If a high voltage level is indicated as „0“ and a low voltage level „1“ we speak of negative logic.

## 4.2 Logical functions

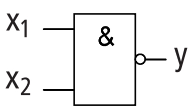
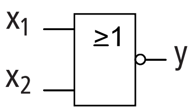
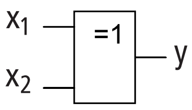
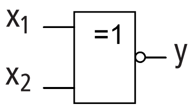
Logical functions (also known as „Boolean functions“) are labled as logic gates or simply as gates. There are the three basic logical functions NOT, AND and OR of which consist the functions NAND, NOR, XOR and XNOR. Each logical element can be described by a switch function. Common forms of presentation for this include the Boolean equation and the truth table.

When a digital circuit consists only of logical elements without feedback of output and input, we speak of a clear combinational logic or a combinatorial circuit (compare chapter 4.5, page 21). In addition to logical functions, digital circuits can include storing elements like flip-flops that work synchronously or statefully. As soon as at least one output is directed to the input we speak of sequential circuit or machine (compare 4.6, page 21).

**Basic functions**

| Circuit symbol | Designation | Equation | Truth table |
|---|---|---|---|
|  | NOT (Negation) | $y = \overline{x}$ | x ‖ y<br>0 ‖ 1<br>1 ‖ 0 |
|  | AND (Conjunction) | $y = x_1 \wedge x_2$ | $x_1$ \| $x_2$ ‖ y<br>0 \| 0 ‖ 0<br>0 \| 1 ‖ 0<br>1 \| 0 ‖ 0<br>1 \| 1 ‖ 1 |
|  | OR (Disjunction) | $y = x_1 \vee x_2$ | $x_1$ \| $x_2$ ‖ y<br>0 \| 0 ‖ 0<br>0 \| 1 ‖ 1<br>1 \| 0 ‖ 1<br>1 \| 1 ‖ 1 |

Composed functions

| Circuit symbol | Designation | Equation | Truth table |
|---|---|---|---|
|  | NAND (Exclusion) | $y = \overline{x_1 \wedge x_2}$ | $\begin{array}{c\|c\|\|c} x_1 & x_2 & y \\ \hline 0 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{array}$ |
|  | NOR (Nihilition) | $y = \overline{x_1 \vee x_2}$ | $\begin{array}{c\|c\|\|c} x_1 & x_2 & y \\ \hline 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 0 \end{array}$ |
|  | XOR (Anticoincidence) | $y = (\overline{x_1} \wedge x_2) \vee (x_1 \wedge \overline{x_2})$ | $\begin{array}{c\|c\|\|c} x_1 & x_2 & y \\ \hline 0 & 0 & 0 \\ 0 & 1 & 1 \\ 1 & 0 & 1 \\ 1 & 1 & 0 \end{array}$ |
|  | XNOR (Equivalent) | $y = (x_1 \wedge x_2) \vee (\overline{x_1} \wedge \overline{x_2})$ | $\begin{array}{c\|c\|\|c} x_1 & x_2 & y \\ \hline 0 & 0 & 1 \\ 0 & 1 & 0 \\ 1 & 0 & 0 \\ 1 & 1 & 1 \end{array}$ |

**Please note: The display of circuit symbols complies with IEC 60617-12.**

## 4.3 Comparing number systems

| | Binary number | | | | Decimal number | | Hexadecimal number* |
|---|---|---|---|---|---|---|---|
| Value of digits | **8** ($2^3$) | **4** ($2^2$) | **2** ($2^1$) | **1** ($2^0$) | **10** ($10^1$) | **1** ($10^0$) | **1** ($16^0$) |
| Zero | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| One | 0 | 0 | 0 | 1 | 0 | 1 | 1 |
| Two | 0 | 0 | 1 | 0 | 0 | 2 | 2 |
| Three | 0 | 0 | 1 | 1 | 0 | 3 | 3 |
| Four | 0 | 1 | 0 | 0 | 0 | 4 | 4 |
| Five | 0 | 1 | 0 | 1 | 0 | 5 | 5 |
| Six | 0 | 1 | 1 | 0 | 0 | 6 | 6 |
| Seven | 0 | 1 | 1 | 1 | 0 | 7 | 7 |
| Eight | 1 | 0 | 0 | 0 | 0 | 8 | 8 |
| Nine | 1 | 0 | 0 | 1 | 0 | 9 | 9 |
| Ten | 1 | 0 | 1 | 0 | 1 | 0 | A |
| Eleven | 1 | 0 | 1 | 1 | 1 | 1 | B |
| Twelve | 1 | 1 | 0 | 0 | 1 | 2 | C |
| Thirteen | 1 | 1 | 0 | 1 | 1 | 3 | D |
| Fourteen | 1 | 1 | 1 | 0 | 1 | 4 | E |
| Fifteen | 1 | 1 | 1 | 1 | 1 | 5 | F |

The respective number system is indicated by its basis: the basis is lowered in parenthesis, for example 1100(2) = 12(10) = C(16).

In addition to the binary system with the basis 2, the hexadecimal system with the basis 16 is frequently used in digital technology. Binary numbers are relatively long and difficult to overlook. As 16 is a potency of 2, it is especially easy to convert binary numbers into hexadecimal numbers. In this case, four digits of the binary numbers are replaced by one hexadecimal digit – which increases greater clarity. The hexadecimal digits with the value 0-9 are represented by the number symbols 0-9. The values 10-15 are represented by the captial letters A-F - which is why they are clearly readable.

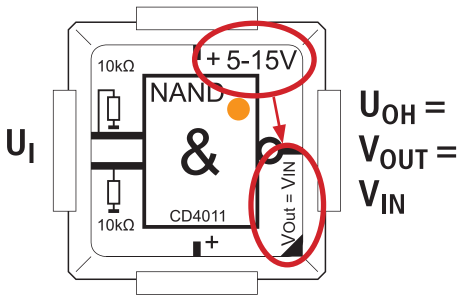
Hence, you can easily determine that ADF5(16) is bigger than ABF1(16) while the respective binary numbers 1010110111110101(2) and 1010101111110001(2) are not as transparent.

The Latin-Greek combination of words in "Hexadecimal" derives from Greek hexa "six" and Latin decem "ten". Also compare: https://en.wikipedia.org/wiki/Hexadecimal

## 4.4 Logic level

Voltage levels are defined to process and display digital signals. Regarding binary coded signals, two voltage ranges are enough that represent the information. These are called High Level (or H Level, High or H) and Low Level (L Level, Low or L).

The High Level, that is the higher voltage, is equivalent to the operating voltage (e.g. 9 V with bricks); the Low Level being the lower voltage is at 0 V (usually called ground). The more or less steep transition from Low to High Level  is also called rising or positive  edge and the transition from High to Low Level is called falling or negative edge. The change between both levels has to occur with a minimum edge steepness that is usually defined in the manufacturer's data sheet.



Regarding our Logic bricks, the output level complies with the supply voltage that varies between +5V and +15V (exception BCD Counter Brick: 9-15V). When using the provided power supply adapter or a 9V monobloc battery it is 9V. Consequently, the output High Level Uoh corresponds approximately to the supply voltage (as usual in CMOS components).

Output level of the bricks

### 4.4.1 Voltage ranges

The voltage ranges are prescribed by the different "Logic Families". Logic Families are concidered to be electronic circuits (ICs) that have identical logic levels, temporal behaviours, driver properties, manufacturing processes etc.

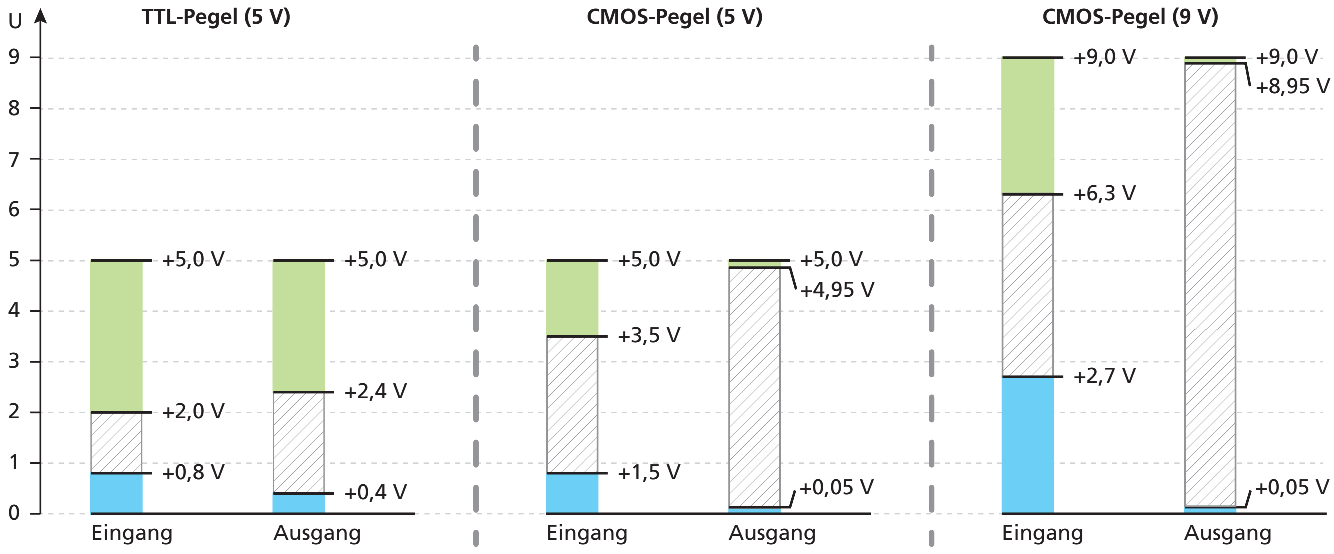


Figure 5: Logic level for TTL and CMOS technology

To demonstrate both logic values we have relatively huge level ranges (blue and green in the graphic). This makes sense for real logic circuits to recognize the conditions in spite of tolerance.  The range between both input level ranges of a logic gate, that is, between UIL and UIH is not permitted (prohibited range), the signal range cannot clearly be assigned to a logic value (hatched graphic). This is why there is a minimum output voltage UOH guaranteed for a High Level range on the output and there is a minimum input voltage UIH requested on the input. The output voltage UOH is always higher than the input voltage UIH. The difference UOH – UIH that ensures the circuits' operational safety is called static interference distance. Low Level ranges have a maximum output voltage UOL, the maximum input voltage UIL and the static interference distance UIL – UOL.

## 4.4.2 Logic states

The level information L and H must not be mixed up with the logic states 0 and 1. L and H indicate the real voltage level. For example 0V (Low) or 5V (High). With these level information we refer to the real electric voltage level of a circuit. When you want to describe the logic functioning of a circuit, the level information have to be assigned to the logic states. We differentiate between positive and negative logic.

**Positive logic**

When using positive logic, the logic 0 corresponds to the low level and the logic 1 corresponds to the high level.

**Negative logic**

When using the negative logic, the logic 0 corresponds to the high level while the logic 1 corresponds to the low level.

Indications of low active signals are usually marked with an overscore. Alternatively, asterisks or forward slashes are prefixed or follow the indication. The spelling RESET,

*RESET and /RESET all indicate that the signal RESET is low active.

## 4.5 Combinational circuit

The term combinational circuit indicates a combinational circuit that consists of simple basic gates (for example AND, OR , NOT). One or more output variable depend on one or more input variables. There aren't any feedbacks, that means, the gate's output is not reconducted to its input. You could also call this an oblivious act. Signal propagation delays are neglected on the logic leve (compare ch. "Glitches" on p. 22)

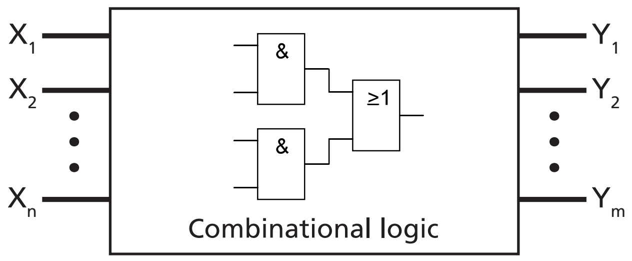


Image 6: Combinatorial circuit

## 4.6 Sequential circuit

You need circuits with a "memory" in order to solve a problem that does not only depend on a snapshot. This means, you need a circuit that combines input variables at a certain time with values that had occured before that time. Due to the outputs' feedback to the inputs, the circuit gets a storing character. We receive a so called sequential circuit or finite state machine.

Such a sequential circuit consists of a memory block (flip-flop) and a block with combinational logic. The combinational circuit part develops the requested effect Zt+1 and the output variables  Y1 out of the current memory block’s state and the input variables X.

This part is called Mealy-Automat.

While the output variables Y1 can change in between a state, the outputs Y2 are clocked and free from possible error pulses (so called “Moore Automat”).

A sequential circuit is synchronous when the inputs and the feedback are synchronized via clock signals (right image), otherwise they are "asynchronous" (left image).

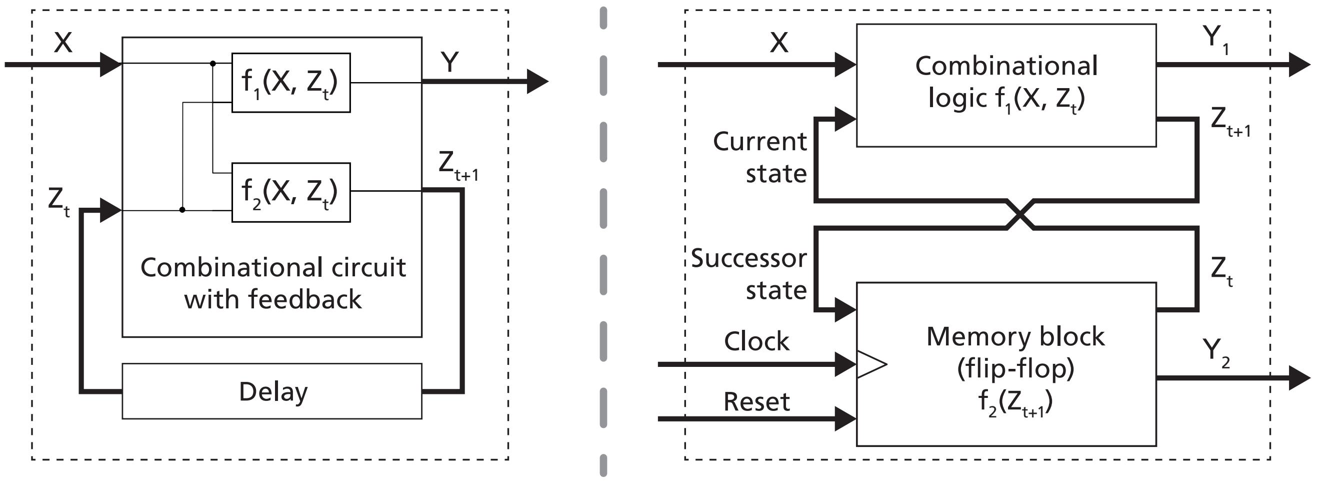


Image 7: Asynchronous sequential circuit (left) and synchronous sequential circuit (right)

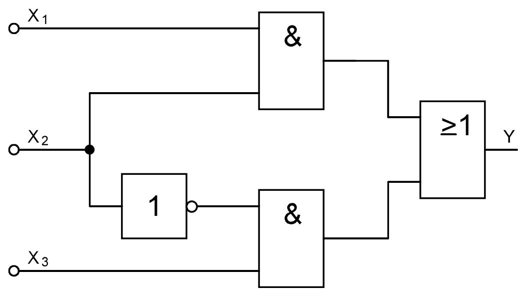
In reality, each combinational circuit generates its outcome only after some time that depends on its development. In order to combine input values with the correct buffered values at a fixed time, the input signals and the feedback outputs are synchronzied via a flip-flop and a clock.

The frequency of the clock signal needs to be big enough for all combinational circuits in the sequential circuit to finish their calculations, this means, for all propagation delays involved to be elapsed.

## 4.7 Glitches

In the field of electronics "glitches" are a short-time false statement in logical circuits and a temporary falsification of a boolean function. This happens since the signal delays of the individual gates are never completely equal. The frequency of glichtes is rising with the complexity and the miniaturization of the circuits and faster clock pulses. Glitches are a significant problem at the development of modern electronical circuits and fast micro processors.

**Example for the development of glitches:**

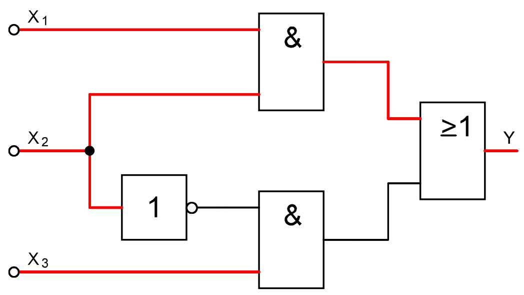


**The circuit**

There is a circuit that has three inputs: $x_3$. The output is supposed to deploy the value "1" if at least one of the conditions is fulfilled.

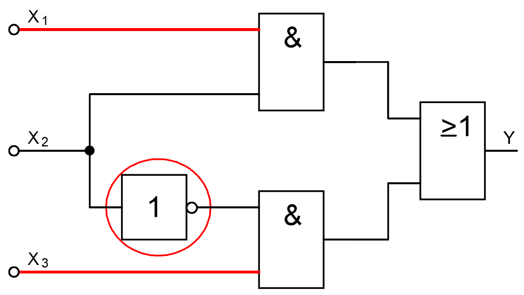
- $x_2$ and $x_1$ are at the same time „1“ OR
- $x_2$ is „0“ and $x_3$ at the same time „1“

If not at least one of the two conditions apply, y should deploy a "0".



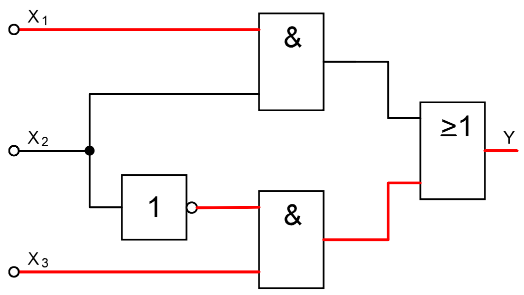
**State 1 - the circuit delivers a 1 as desired**

The circuit is now in state 1. According to our specifications, the first condition applies because $x_2$ and $x_1$ are "1". The junctions that transmit the information "1



**State 2 - the inverter causes a glitch**

In the state 2, $x_2 = 0$ and $x_1 = 1$. The circuit should still deploy a "1". The inverter needs some time to register the converting of the $x_2$-Signals from "0" to "1". For a short time $x_2 = 0$, as well as $\overline{x}_2 = 0$. The state is is handled as if none of the conditions apply and therefore it deploysj a "0". This situation is called "glitch".



**State 3 - the circuit delivers the correct value again**

After some time - in our case a couple of nanoseconds - the circuit is in state 3: the inverter has processes the new information. The now deployed "1" goes into the AND gate which deploys (again after a few nanoseconds) a "1" as well. Now the OR linkage at y deploys the desired "1".

Figure 8: Development of glitches

In practice, the time difference also exists in gates of the same type and with different lengths. If you want to know the exact values of the function, you have to wait some time until all signals are stable. This fact limits the clock frequency of modern processors significantially.

The impact of glitches in synchronous circuit designs can be prevented through downstream D-Flipflops. The outputs of combinational circuit parts, which consist of diverse logic gates with various terms, need to adapt applicable states just after the clock edge accepts the output values in the D-Flipflops. In the time between two clock edges, any number of glitches can occur through runtime effects in the combinational part, since intermediate states are not noted by the downstream of D-Flipflop.

The method to always equip the outputs of combinational circuit parts with D-Flipflops is one of the essential bases for stable, digital circuit designs in complex all programmable FPGAs (see chapter 4.8.3, page 23)

## 4.8 Programmable logic blocks

Conventional, non-programmable logic bricks have firmly defined functions. These bricks can be bought as integrated circuit (IC), and are also built into the logic bricks. One of the most known logic gates is the NAND gate with type designation 7400 (TTL technology) or 4011 (CMOS technology). By contrast, PLD (programmable logic devices) get their function by appropriate programming of the circuit developer (so called personalization). Some chip manufacturers offer free software tools for circuit designs, implementation (place & route) and simulation.

We created a short overview for the ones who'd like to dip deeper into this topic - especially regarding the FPGA elements: (rising complexity):

### 4.8.1 Programmable Array Logic (PAL) and Generic Array Logic (GAL)

A PAL is a programmable AND array with a fixed OR array. The GAL is in contrast to a PAL rewriteable. The personalization takes place by a programming unit by the circuit developer. In technology, the successor is the CPLD (see next chapter)

### 4.8.2 Complex Programmable Logic Device (CPLD)

A CPLD is made of blocks, which can be connected. The main element is the so called programmable logic arrangement (PLA), that consists of an AND array and an OR array, while both arrays can be programmed. More, an input and output block as well as a programmable feedback exist. Usually, for every I/O pin is a flipflop on hand. This AND/OR matrix allows any combinational link. Due to the structure the throughput times of the CPLD can be - unlike those of the FPGA - defined precisely. Another disparity to the FPGA is the permanent programming by the circuit developer, so the functionality remains even after turning off the supply. The CPLD needn't be refreshed every time the devices is started.

### 4.8.3 Field Programmable Gate Array (FPGA)

A FPGA consists - similar to the CPLD - of connected, but quit complex blocks. The possibility to connect these blocks is in contrast to the CPLD enlarged. By specific configuration of the intern existing elements, in the FPGA many different circuits and functions can be realised. The range of circuits reaches from less complex projects, like e.g. a simple synchronic meter or decoder, to high complicated circuits, such as memory controller or digital signal processing. Modern FPGA often contain integrated function blocks, like RAM, PLL or whole CPU cores.

FPGA are used in every field of digital technologies, especially in working areas with fast signal processing and high flexibility in changes. So also subsequent changes and improvements in the implemented circuit are possible without causing high costs or time-consuming hardware changes. The high flexibility is perfect for prototypes and for cheap productions of smaller and medium series.

The personalization is carried out "in the field" by the user either by reading the configuration data from an external non-volatile memory such as a PROM (Programmable Read Only Memory) - in this case, the FPGA is "master" or by downloading the data from the computer In the FPGA (slave mode).

After switching on the power supply, the FPGA is initially "dumb". The module is ready for operation only after the so-called configuration phase of a few milliseconds has been completed. FPGAs can be reconfigured at any time in the circuit without a programming device.

### 4.8.3.1 Basic structure of FPGAs

The basic structure of FPGAs is a matrix arrangement of configurable logic blocks (CLBs) as well as input / output blocks (IOBs). The latter are usually connected to the pins on the chip. See Fig. 9 on page 24. Between the blocks (CLBs and IOBs) is a lattice of connection paths, also called the interconnect area consisting of a hierarchy of horizontal and vertical "Lines". The inputs and outputs of the blocks are connected to these lines. Remote connections are "wired" through programmable connection points in the switch matrices of the grid. In this way, signal routing (also called routing) is possible over the entire chip.

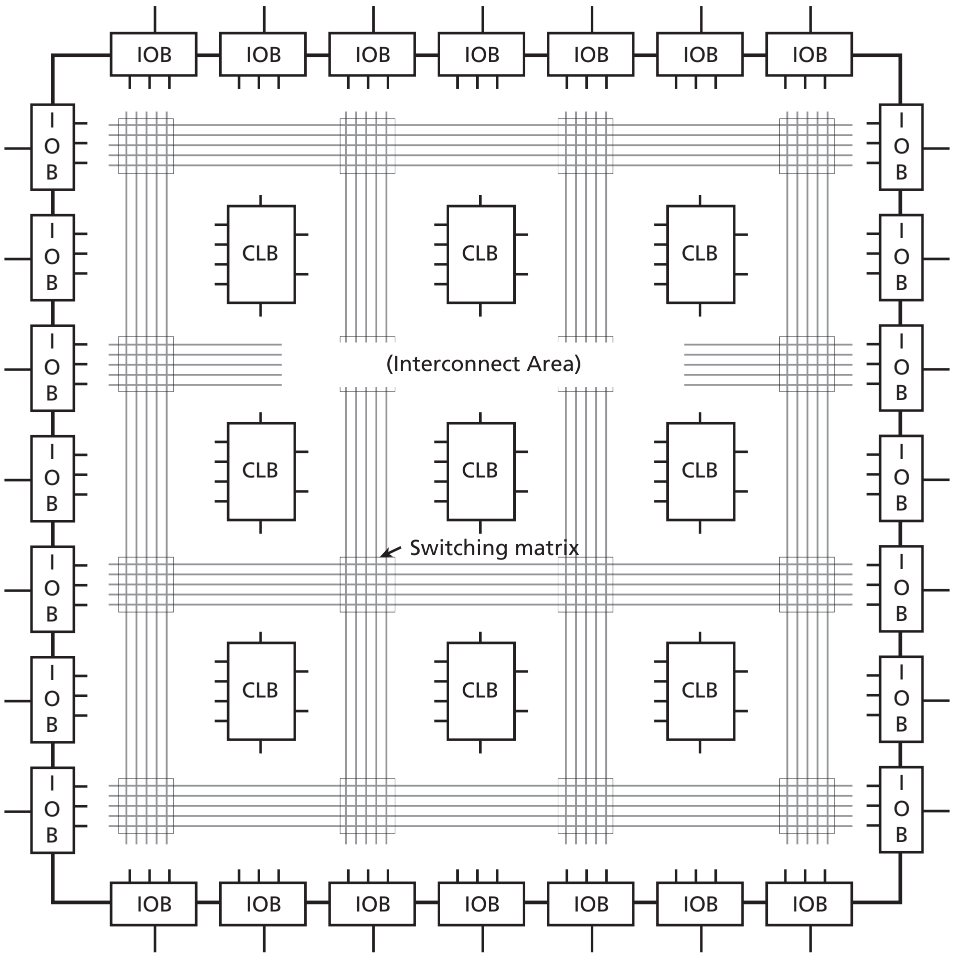


Figure 9: Basic structure of a FPGA

### 4.8.3.2 Configurable logic blocks

The basic structure of an FPGA is an array of configurable blocks with a programmable truth table (LUT) and a 1-bit register (D-flip-flop). Depending on the number of available inputs, the LUT can implement any n-digit binary function. Common are LUT structures with four binary inputs and more. In addition to the LUTs, multiplexers in the base blocks allow very fast local signal paths, for example for the integration or bypassing of the flip-flop, for feedback of its output, for connecting neighboring blocks and the like.

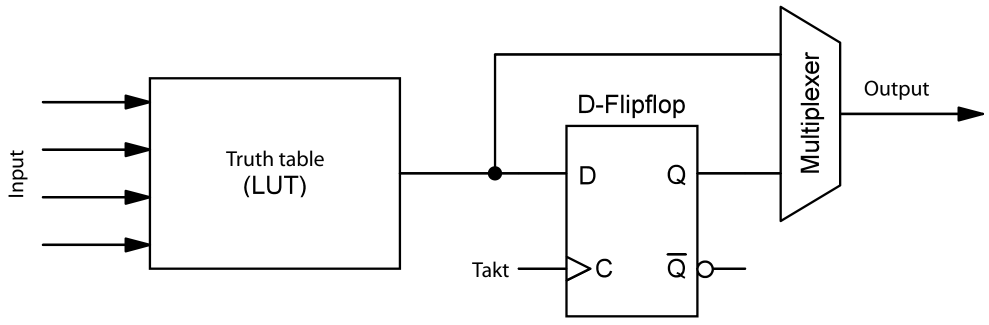


Figure 10: Logic block example

### 4.8.3.3 Input / output blocks

The input / output blocks (IOBs) form the interface to the outside world. The FPGA's connections are connected to the switching matrix. Depending on the application, the voltage level of the inputs and outputs can be adapted to the respective interface standard (TTL, LVDS, etc.). In addition, the drive current of the outputs and the edge slope of the output signals, as well as Tristate buffers (pins that can be switched to high impedance) can be activated to set up a bus architecture.

# 5. Circuit examples

## 5.1 Basic circuits

### 5.1.1 Inverter

The inverter brick realises a NOT function and is therefore also called neation or inverting. This means that the input signal is inverted. If the input of the inverter brick receives a high level, you receive a low level at the output and the other way around.

In our brick circuit we connected a push button and the power supply to the input of the inverter brick. The inverter brick has a built-in pull-down resistor so that it knows what to do if the push button is not pressed.

By pushing the button, there is high level at the input. Therefore the output is low and the LED switches off.

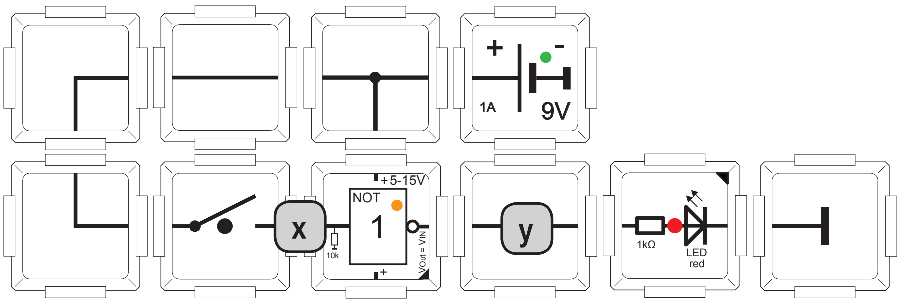
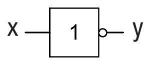


Figure 11: Inverter as brick circuit

| Circuit symbol | Description | Equation | Truth table |
|---|---|---|---|
|  | NOT (Negation) | $y = \overline{x}$ | x ‖ y<br>0 ‖ 1<br>1 ‖ 0 |

### 5.1.2 AND gate

The AND gate realises a logical AND linkage of two or more inputs. Compared to the NAND gate, the ouput is not inverted.

In our example, the two inputs of the AND gates are connected with the supply voltage via push buttons. As long as no button is pressed, both of the inputs are connected to ground (low level) via the pulldown resistors.  This means in the case of an AND linkage that the ouput deploys a defined level. The LED only starts to light up if both push buttons are pressed and the output changes (as you can see in the truth table) to a high level.

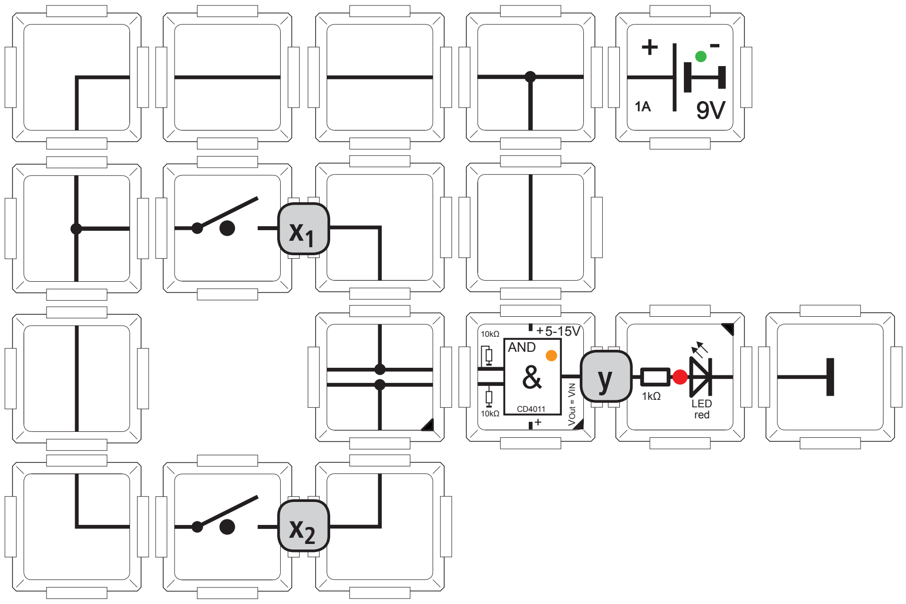
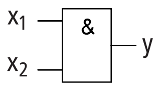


Figure 12: AND gate as brick circuit

| Circuit symbol | Description | Equation | Truth table |
|---|---|---|---|
|  | (Conjunction) | $y = x_1 \wedge x_2$ | |

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 1 |

You can also build a AND circuit with two push buttons that are connect in series. Only if you push both buttons ($x_1$ und $x_2$) are pressed, the LED (y) lights up. Try this fuction with the help of the truth table above.

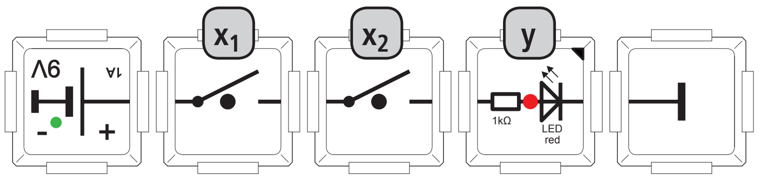


Figure 13: Alternative AND circuit with push button

## 5.1.3 OR gate

The OR gate realises a logical OR linkage with two or more inputs. Compared to the NOR gate, the outputs are not inverted.

In our example, the two inputs of the OR gates are connected with the supply voltage via push buttons. As long as no button is pressed, both of the inputs are connected to ground (low level) via the pulldown resistors.  This means in the case of an OR linkage that the ouput deploys a defined level. The LED only starts to light up at least one push button is pressed and the output changes (as you can see in the truth table) to a high level.

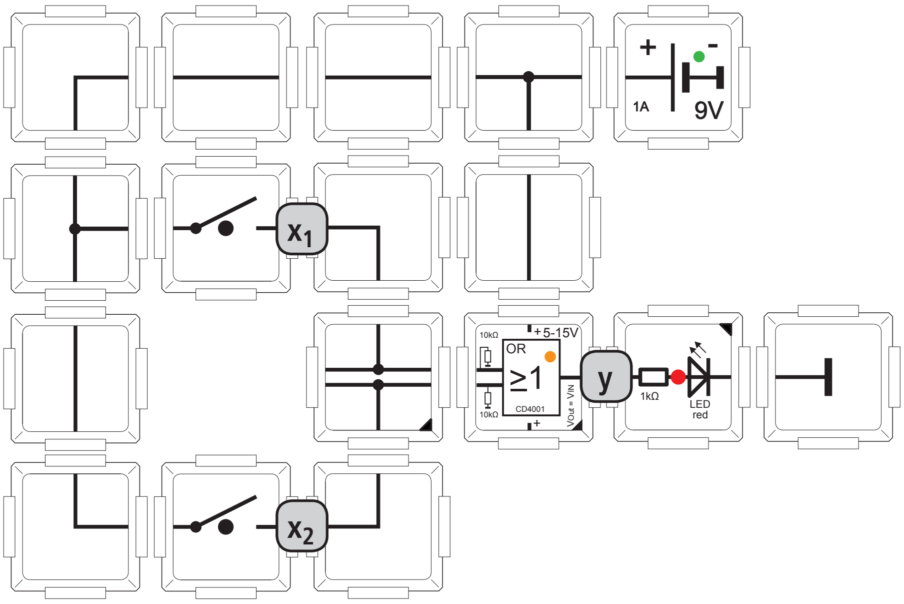
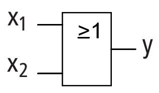


Figure 14: OR gate as brick circuit

| Circuit symbol | Description | Equation | Truth table |
|---|---|---|---|
|  | OR (Disjunction) | $y = x_1 \lor x_2$ | see below |

| $x_1$ | $x_2$ | $y$ |
|---|---|---|
| 0 | 0 | 0 |
| 0 | 1 | 1 |
| 1 | 0 | 1 |
| 1 | 1 | 1 |

An OR circuit can also be built with two push buttons that are connected in parallel. As soon as you press at least one button ($x_1$ or $x_2$) the LED (y) lights up. Try this funcationality with the help of the truth table above.

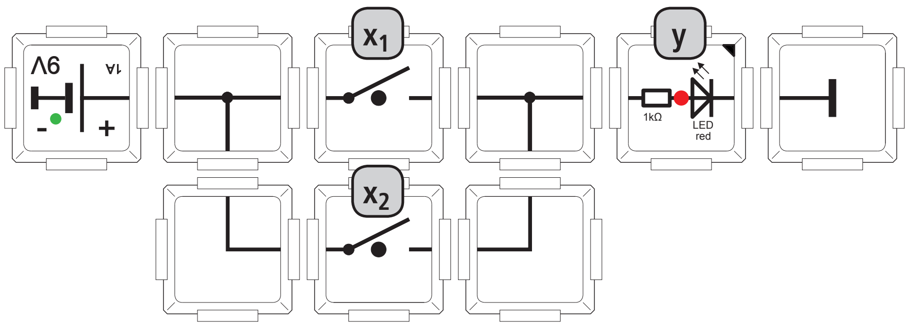


Figure 15: Alternative OR circuit with push button

## 5.1.4 NAND gate

The NAND gate realises a logical AND linkage of two or more inputs with an inverted output. Besides the NOR gate, the NAND gate is the most common type. There is no function that can not be built with these to gates! This shows why those two types are so common in practice.

In our example, the two inputs of the NAND gate are connected with the supply voltage via push buttons. As long as no button is pressed, both of the inputs are connected to ground (low level) via the pulldown resistors. The LED light sup because the inverted output deploys a high level. The LED only switches off if both of the push button are pressed and the output changes (as you can see in the truth table) to a low level.

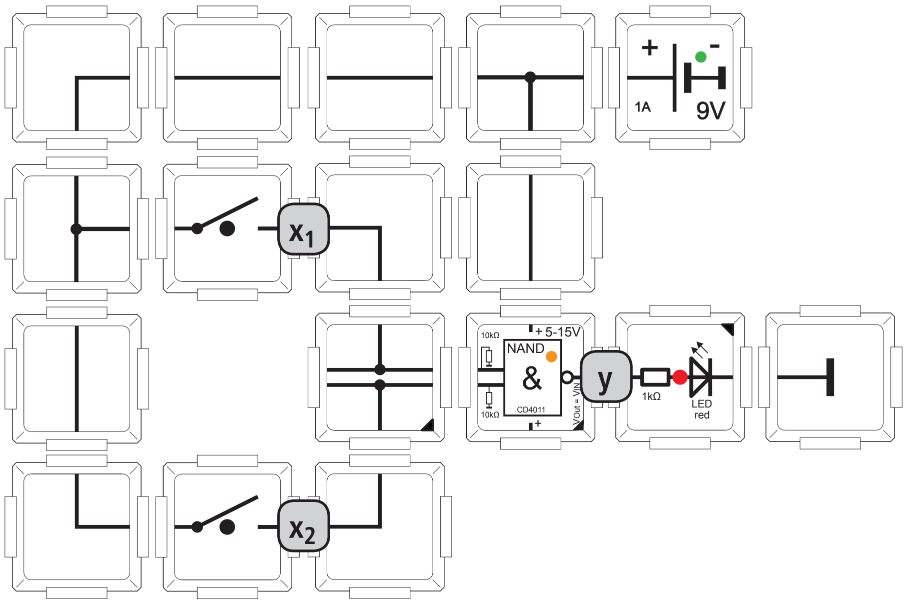
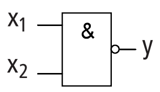


Figure 16: NAND gate as brick circuit

| Circuit symbol | Description | Equation | Truth table | | |
|---|---|---|---|---|---|
| | | | $x_1$ | $x_2$ | y |
|  | NAND (Exclusion) | $y = \overline{x_1 \wedge x_2}$ | 0 | 0 | 1 |
| | | | 0 | 1 | 1 |
| | | | 1 | 0 | 1 |
| | | | 1 | 1 | 0 |

Alternatively you can also build the XOR circuit in a more compact way - the functionality is the same:

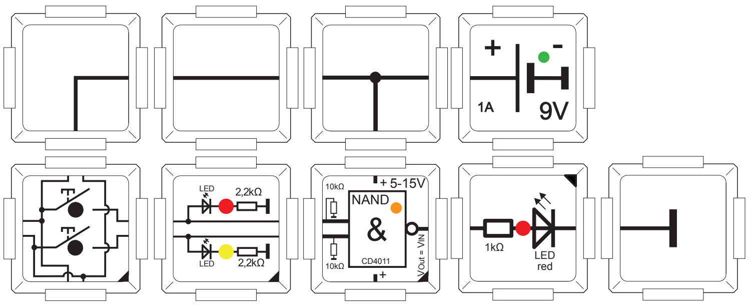


Figure 17: Alternative NAND circuit

## 5.1.5 NOR gates

The NOR gate realises a logical OR linkage of two or more inputs with inverted output.

In our example, the two inputs of the NOR gate are connected with the supply voltage via push buttons. As long as no button is pressed, both of the inputs are connected to ground (low level) via the pulldown resistors. The LED lights up because the inverted output deploys a high level. As long as at least one of the two buttons is pressed, the output changes to low level and the LED switches off.

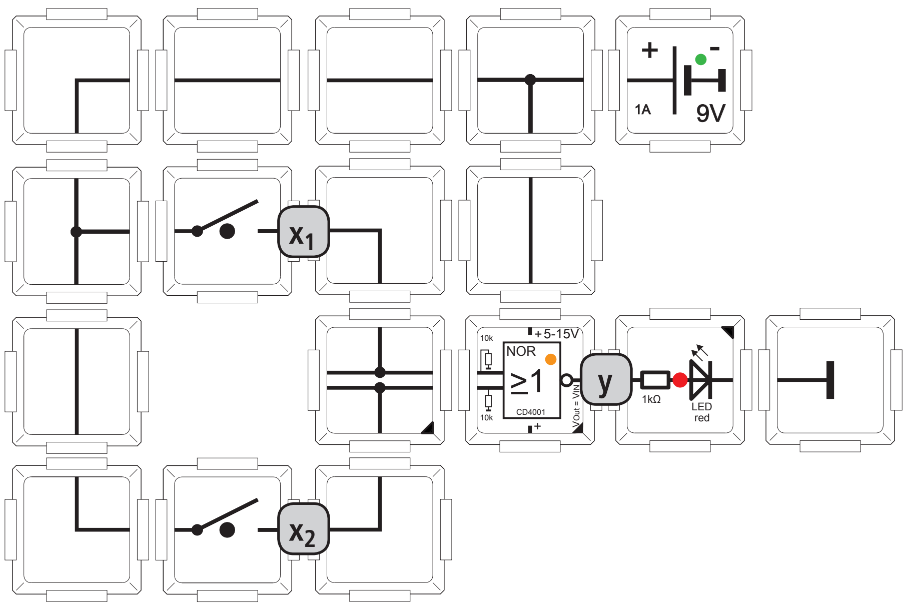
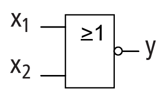


Figure 18: NOR gate as brick circuit

| Circuit symbol | Description | Equation | Truth table |
|---|---|---|---|
|  | NOR (Nihilition) | $y = \overline{x_1 \vee x_2}$ | |

| $x_1$ | $x_2$ | y |
|---|---|---|
| 0 | 0 | 1 |
| 0 | 1 | 0 |
| 1 | 0 | 0 |
| 1 | 1 | 0 |

Alternatively you can also build the XOR circuit in a more compact way - the functionality is the same:

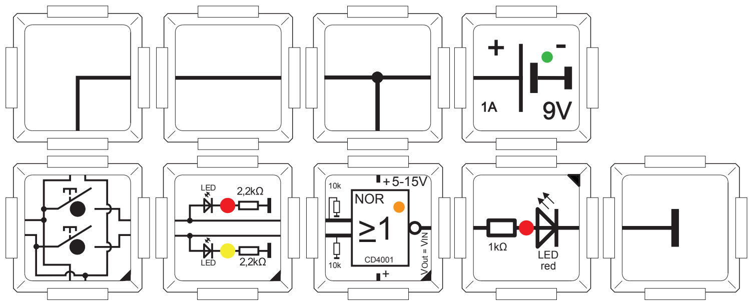


Figure 19: Alternative NOR circuit

## 5.1.6 XOR gate

The XOR gate is very common in the electronical field. It realises a logical exclusive-or connection with two or more inputs. An easy example is a common toggle switch e.g. the hall light at home. Each switch can be used to switch the light on or off. It doesn´t matter which switch you use first. In everyday conversation you would proabably refer to the XOR gate as "either...or". The XOR linkage is an important part of the adder, that we will get to know in chapter 5.3 "digital counter".

In this example, the two inputs of the XOR gate are connected with the supply voltage via push buttons. As long as no button is pushed, both of the inputs are connected to ground (low level) via the two integrated pulldown resistors. The LED does not light up because the output issues a low level. Once "either" button 1 "or" button 2 is pressed, the output is changing to high level and the LED lights up. If no or both buttons are pressed the output changes bach to low level referred to the truth table.

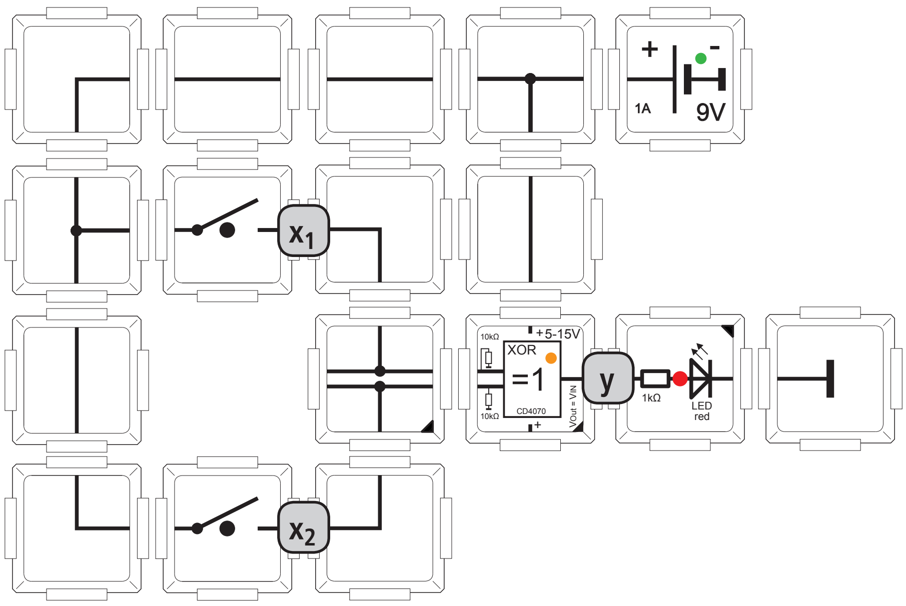
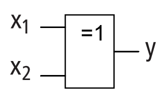


Figure 20: XOR gate as brick circuit

| Circuit symbol | Decription | Equation | Truth table | | |
|---|---|---|---|---|---|
| | | | $x_1$ | $x_2$ | y |
|  | XOR (Antivalence) | $y = (\overline{x_1} \wedge x_2) \vee (x_1 \wedge \overline{x_2})$ | 0 | 0 | 0 |
| | | | 0 | 1 | 1 |
| | | | 1 | 0 | 1 |
| | | | 1 | 1 | 0 |

Alternatively you can also build the XOR circuit in a more compact way - the functionality is the same:

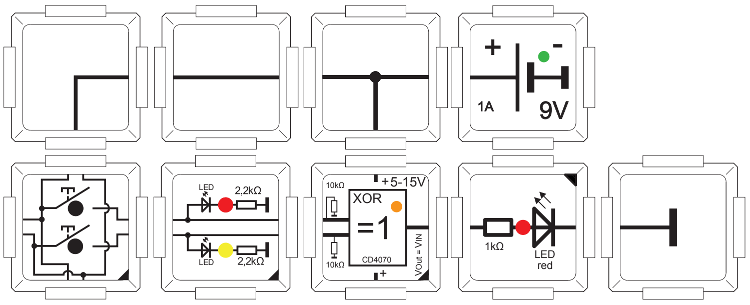


Figure 21: Alternative XOR circuit

## 5.1.7 XNOR gates

The XNOR gate realises logical exclusive-or linkage of two or more inputs. Besides the NAND or NOR gates it is a gate with inverting output. The inverting exclusive-or linkage signalises the equality of the inputs. This can be used for a comparison of different bit patterns. In the case of a bit pattern equality (both inputs "0" or both inputs "1") an interruption signal can  be sent to the computer or an acustical or opticalal alarm can be activated.

In our example both of the inputs of the XNOR gate are connected via a button with the supply voltage. As long as no button is pressed, both of the inputs are connected to ground (low level) via the pulldown resistors.  This means in the case of a XNOR linkage that the ouput issues a high level. Our red "alarm" LED lights up. The same applies when both of the buttons are pressed at the same time. As soon as only one button is pushed, the output changes back to low level and the LED turns off.

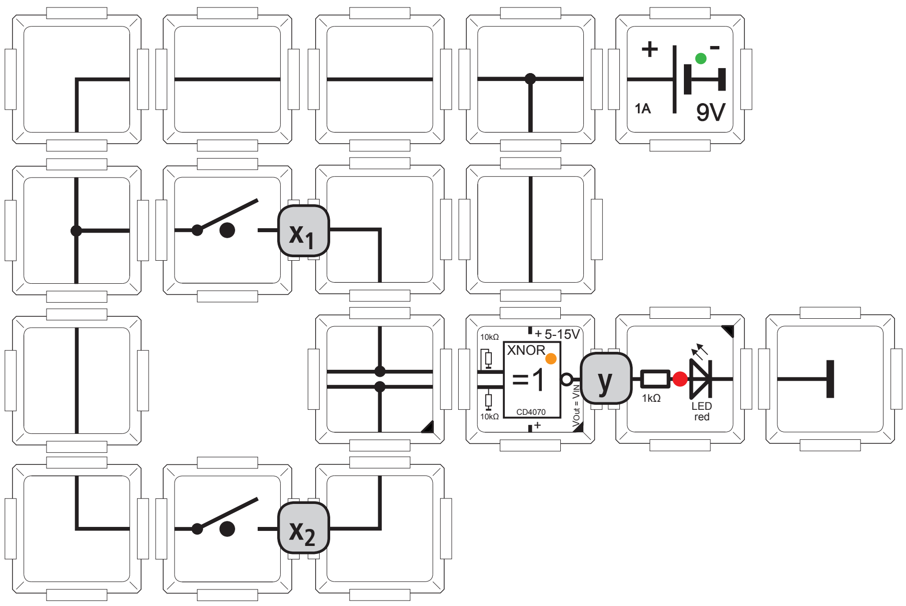
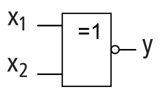


Figure 22: XNOR gate as brick circuit

| Circuit symbol | Description | Equation | Truth table | | |
|---|---|---|---|---|---|
| | | | $x_1$ | $x_2$ | $y$ |
|  | XNOR (Equivalence) | $y = (x_1 \wedge x_2) \vee (\overline{x_1} \wedge \overline{x_2})$ | 0 | 0 | 1 |
| | | | 0 | 1 | 0 |
| | | | 1 | 0 | 0 |
| | | | 1 | 1 | 1 |

Alternatively you can also build the XNOR circuit in a more compact way - the functionality is the same:

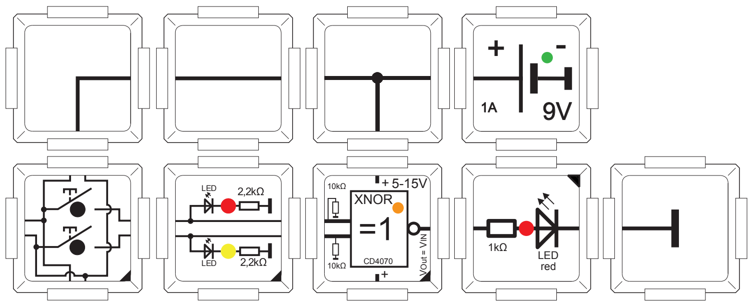


Figure 23: Alternative XNOR circuit

## 5.2 Debounced circuit

Push buttons and switches have the disadvantage that the operation of the mechanical contact (a spring is often used) causes multiple opening or closing. In the digital technology we call this desruptive effect "chatter" or "bouncing".

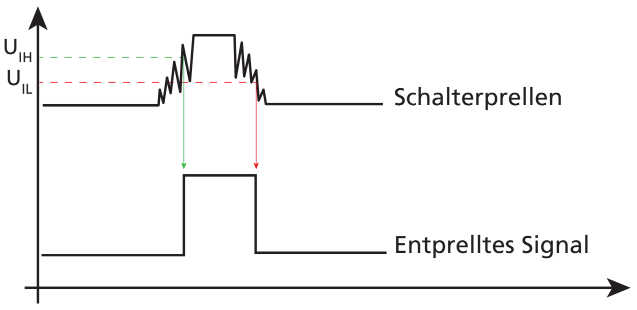


Figure 24: Typical intererence pulse because of contact bounce

But there is an easy solution for this problem by using a normal RS flip-flop. Basically a RS flip-flop can be built optionally with two NOR or NAND gates. Here we use the version with NOR gates because the NOR bricks already include a pull-down resistor at the inputs. Because of the fast switching speed of the RS flip-flop, at first contact  it rests stable in this state and saves the logical value until the inputs are wired differently. Because of the pull-down resistors, the blank inputs always have a stable state.

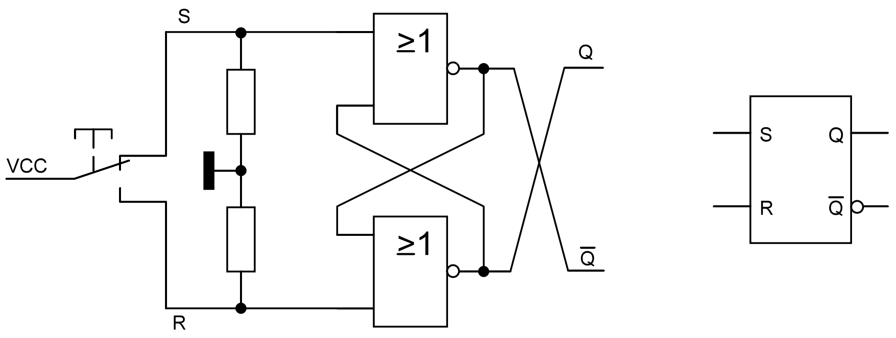
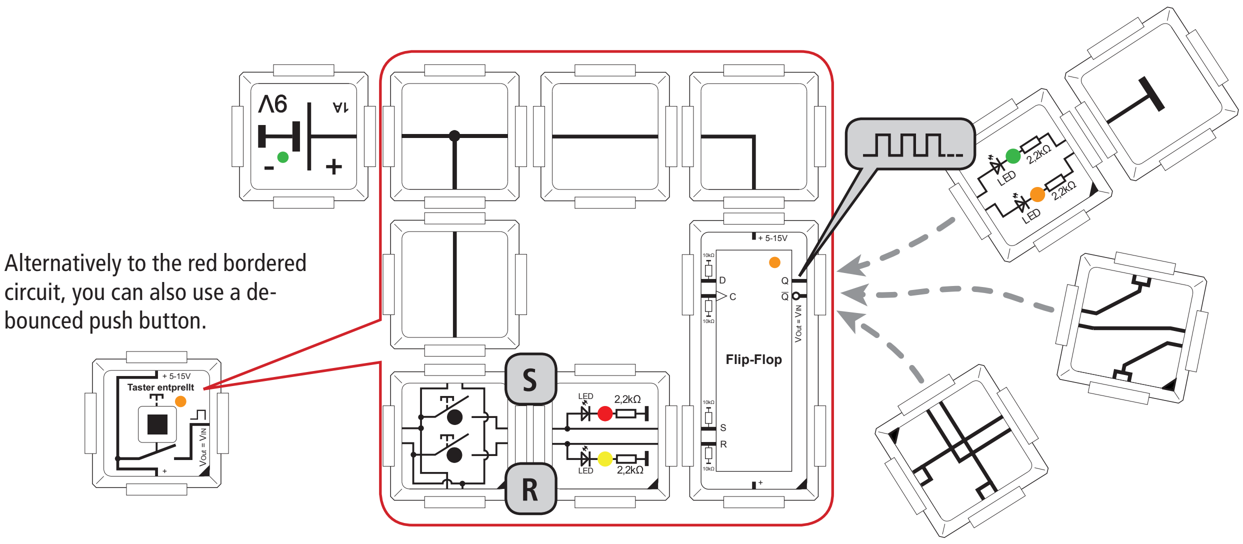


Figure 25: Debounced circuit built with NOR gates (left) and RS flip-flop (right)

Press the push buttons S and R by turns to get a debounced clock pulse at the outputs Q. You can use a clock for this experiment as well.



9V
1A
Alternatively to the red bordered circuit, you can also use a debounced push button.
Taster entprellt
+ 5-15V
S
R
LED
2,2kΩ
Flip-Flop
D
C
Q
S
R

Figure 26: Debounced circuit with RS flio-flop brick (bordered red) or with debounced push button (left)

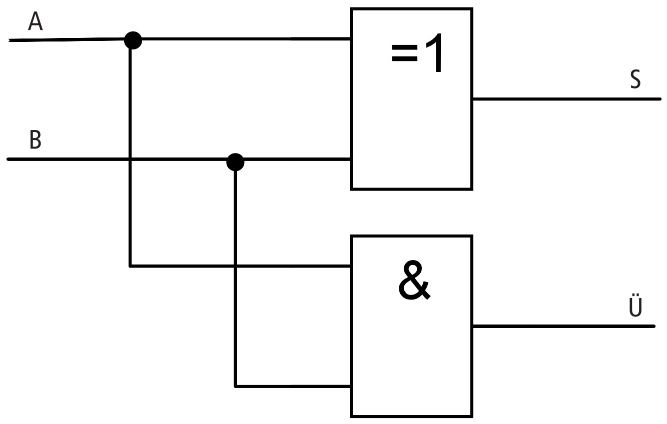
## 5.3 Digital counter

By using the already knows logic gates you can build digital circuits to operate arithmetic operations in the binary number system. This digital counter is also knows as ALU (arithmetic logic unit). It is able to connect two inputs with fundamental, arithmetic and logical operations. If you combine an ALU with a control unit (condition follower) and a result register, you receive a so called central processing unit, known as CPU. This means that the heart of every CPU is a ALU and then again the heart of every ALU is a adder. In the simplest case, two single-digit binary number A and B can be added. In the first step we will build a 1-bit half-adder, that we will extend to a full-adder in the second step.

### 5.3.1 1-bit half-adder

The 1-bit half-adder (HA) is the easiest counter circuit and can add two single-digit numbers with each other. The input A is added to the input B. The sum S is generated with the XOR linkage and the carry out (Ü) with a AND linkage. In principle this is the same way of adding as with the written addition in the decimal number system. Remember: If you reach 10, the next higher digit is noted as a carry out.

The two push buttons represent the summands A and B that should be added. By pressing one of the buttons, you receive the binary value "1": the red sum LED at the output of the XOR gate light up (1+0=1). As soon as both of the push buttons are pressed, meaning that we count 1+1, the sum is "0" and the carry out is "1" (the green LED lights up).



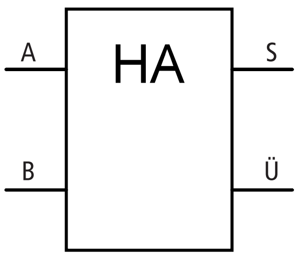
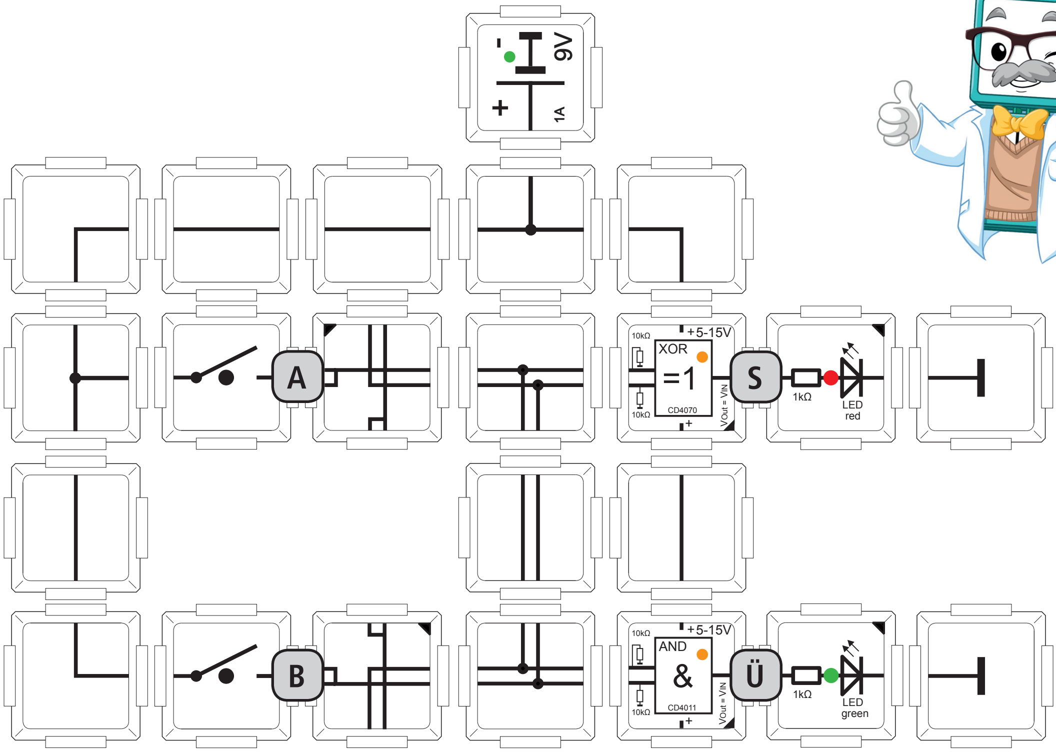


Figure 27: Circuit of a half-adder (left) and its block diagram (right)

**Truth table of a 1-bit half-adder**

| Summand A | Summand B | Sum S | Carry over Ü |
|---|---|---|---|
| 0 | 0 | 0 | 0 |
| 0 | 1 | 1 | 0 |
| 1 | 0 | 1 | 0 |
| 1 | 1 | 0 | 1 |



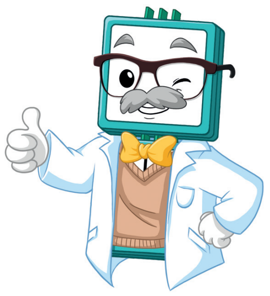


Figure 28: 1-bit half-adder as brick circuit

### 5.3.2 1-bit full-adder

If you want to add muti-digit binary numbers, the carry out of each lower digital has to be included into the sum S. We call this additional summand "carry out input". $Ü_E$. Schaltungstechnisch kombinieren wir zwei Halbaddierer (HA) zu einem sog. Volladdierer (VA). Damit können wir zu den beiden Summanden A und B zusätzlich den Übertrag $Ü_E$ addieren. Das Ergebnis wird als Summe S (rote LED) und als Übertragsausgang $U_A$ (grüne LED) angezeigt.

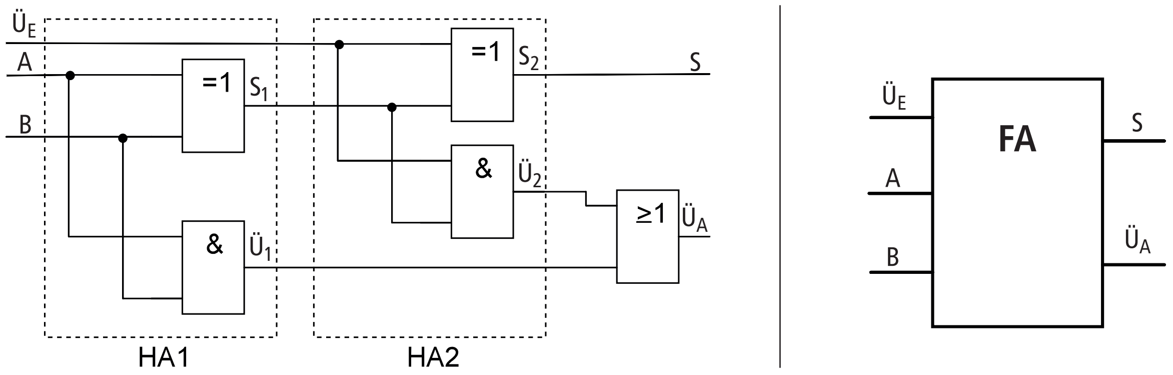


Figure 29: Circuit of a full-adder (left) and block diagram

**Truth table for a 1-bit full-adder**

| Carry out input $Ü_E$ | Summand A | Summand B | Sum S | Carry out output $Ü_A$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 |
| 0 | 1 | 0 | 1 | 0 |
| 0 | 0 | 1 | 1 | 0 |
| 0 | 1 | 1 | 0 | 1 |
| 1 | 0 | 0 | 1 | 0 |
| 1 | 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 0 | 1 |

|  |  |  |  |  |
|---|---|---|---|---|
| 1 | 1 | 1 | 1 | 1 |

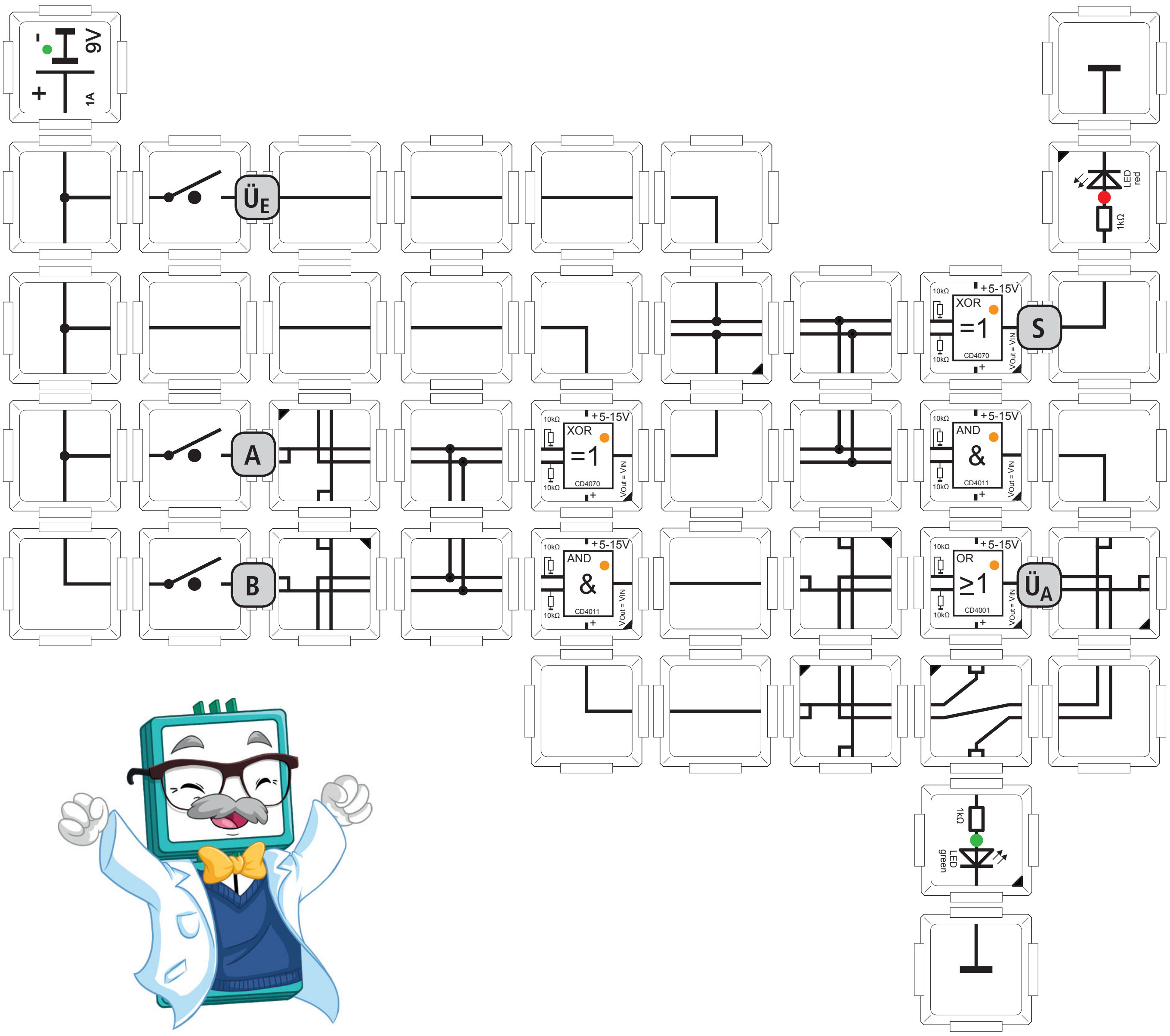


Figure 30: 1-bit full-adder as brick circuit

### 5.3.3 4-bit full-adder

The following block diagram shows a 4-bit full-adder (VA0 to VA3), that is built with four 1-bit full-adders.

Der Eingang $Ü_{E0}$ kann mit Masse verbunden werden, da für die niederwertigste Stelle kein Übertrag berücksichtigt werden muss. Die Überträge von VA0 bis VA3 werden jeweils durch die Verbindung der Übertragausgänge $Ü_{An}$ mit den Übertragseingängen des jeweils höherwertigeren Addierers $Ü_{En+1}$ weitergereicht. $Ü_{A3}$ wird hier nicht weiter verwendet.

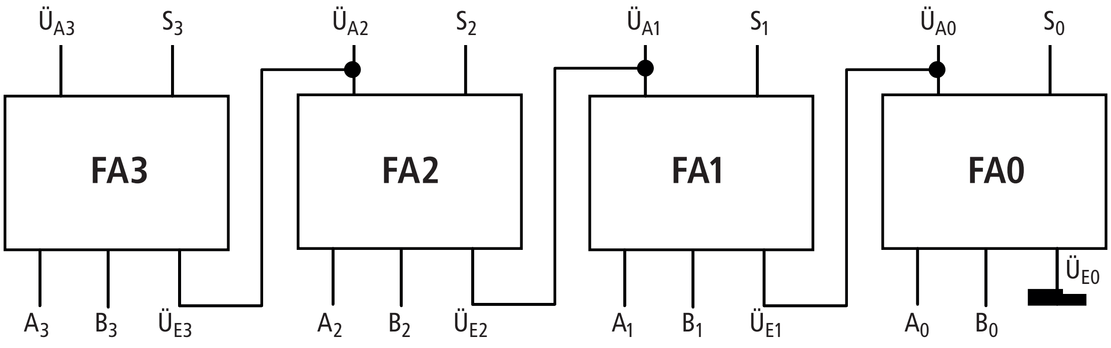
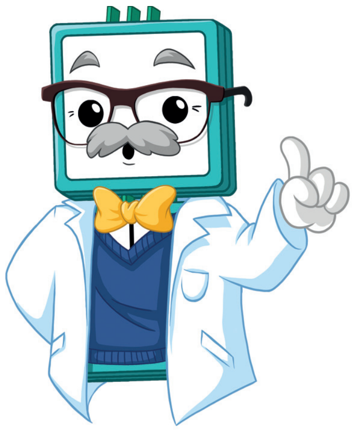


Figure 31: Block diagram of a 4-bit full-adder

**By the way: In the field of digital technology it is common to start the indices for bus connections or logical units with "0". The background ist also that the valence of the digits is displayed as power of two in the binary system: 2n – starting with n = 0. Here is an example:**

| | | | |
|---|---|---|---|
| $2^3 = 8$ | $2^2 = 4$ | $2^1 = 2$ | $2^0 = 1$ |

**Due to the complexity of the circuit, it is not possible to build the 4-bit full-adder with the bricks of the logic set.**



## 5.4 Flip-flops

Suppose you are waiting for a message but currently you are not at home. Thanks to the letterbox this is no problem because the postman can simply post the letter into it. This way, the message doesn´t get lost and you can simply get the mail from the mailbox and read it when you have time. You could also call the postbox a "buffer". The red flag shows you if the postbox contains a message or not. You can imagine a flip-flop, which is the smallest digital storage element, in a simular way to this postbox.

The term "flip-flop" indicates in the digital technology an electronical circuit that can have two stable electrical states (0 and 1). By changing the input signals, the flip-flop can change its state from one to the other.

Therefore we have the possibility to store a digital information (1 bit), as long as there is a supply voltage. The flip-flop is the heart of sequential circuits and is used in various types. In micro processors, several flip-flops are connected in parallel. The width of a register shows us how many flip-flops are connected in parallel in the register. Common units are: 8 bit = 1 byte; 16 bits = 1 word; 32 bits = 1 Longword.

Often the RS flip-flop is also compared to a mechanical rocker:

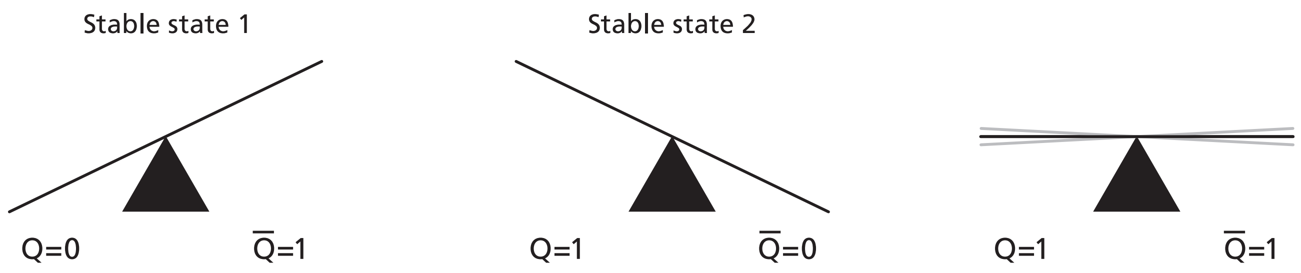


Figure 32: Flip-flop compared to a rocker

A rocker has two stable states: Either its left end or its right end toches the floor. By using the appropriate physical effort, you can change its state. In a flip-flop that happens if you active the inputs, R = 0, S= 1 or R = 1, S = 1. If you don´t use any physical effort to change the state of the rocker, it rests in the last state: the state is therefore stored. If the rocker is in a horizontal orientation, it has a metastable state (R = 1 and S = 1): In practice, the rocker can never rest in this position. At some point it will change to one or the other side. You can´t even tell into which state it changes because you never know all of the exact interferences. A clocked flip-flop is equivalent to the rocker, but the physical effort can only work at a time, that is determined by an external clock signal.

## 5.4.1 Types of flip-flops

Various flip-flops differ in terms of input amount and its behaviour. They have differently working inputs and only work in specifically defined conditions. An easy flip-flop only has to inputs and two outputs. Clocked flip-flops have an additonal clock input.

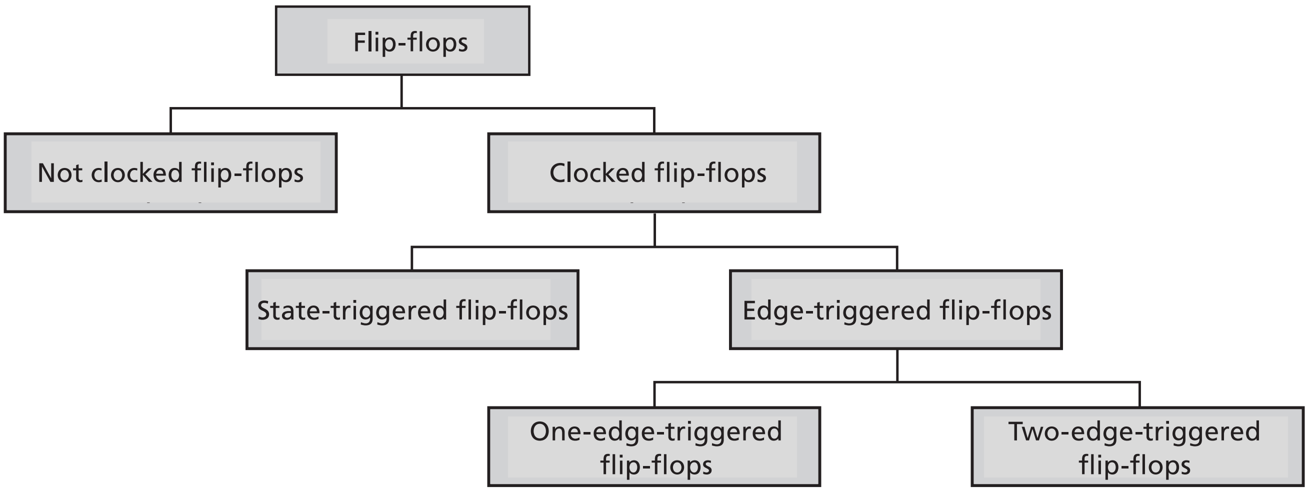
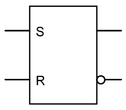

Flip-flops
Not clocked flip-flops
Clocked flip-flops
State-triggered flip-flops
Edge-triggered flip-flops
One-edge-triggered flip-flops
Two-edge-triggered flip-flops

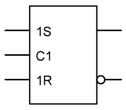
Abb. 1: Flipflop-Typen

### 5.4.1.1 Not clocked flip-flops



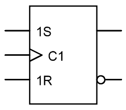
Flip-flops without a clock input are completey independet of a clock. Its set and reset inputs (S and R) are always adressable. See also chapter 5.4.2 on page 40.

### 5.4.1.2 One-edge-triggered flip-flops



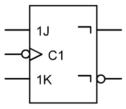
In the one-edge-triggered flip-flop the set and reset inputs are only adressable if the clock signal at the clock input C1 changes its state. This change is displayed by the triangle on the graphic symbol. See also chapter 5.4.3 on page 41.

### 5.4.1.3 Einflankengesteuerte Flipflops



Beim einflankengesteuerten Flipflop ist der Setz- und Rücksetzeingang (1S und 1R) nur bei Änderung der Flanke am Takteingang C1 wirksam. Die Störanfälligkeit wird heruntergesetzt. Die Taktflankensteuerung wird im Schaltzeichen durch das Dreieck gekennzeichnet. Siehe auch Kap. 5.4.3 auf Seite 41.

### 5.4.1.4 Two-edge-triggered flip-flops



The two-edge-triggered registers the input states at the first change of the clock signal and issues the signal at the second change of the clock signal. The change of the clock signal is also called clock edge. This way, the susceptibility is minimized. The clock edge-triggering is displayed by the triangle on the graphic symbol. See more in chapter 5.4.4 on page 42.

Flip-flops whose input signals appear delayed in time at the output, are also called Master-Slave-flip-flops. This is displayed by the rectangular corners at the output. A well-known example for this type of flip-flop is the so called JK-Master-Slave-flip-flop.

In the following chapters you will get to know the flip-flops that are included in the Brick'R'knowledge Logic Set.

## 5.4.2 RS flip-flop

The RS flip-flop is an easy not clock-triggered flip-flop that can be built with two NOR gates. This circuit is also called "NOR flip-flop". In principle, this circuit can also be built with two NAND flip-flops, but this would need negated inputs.

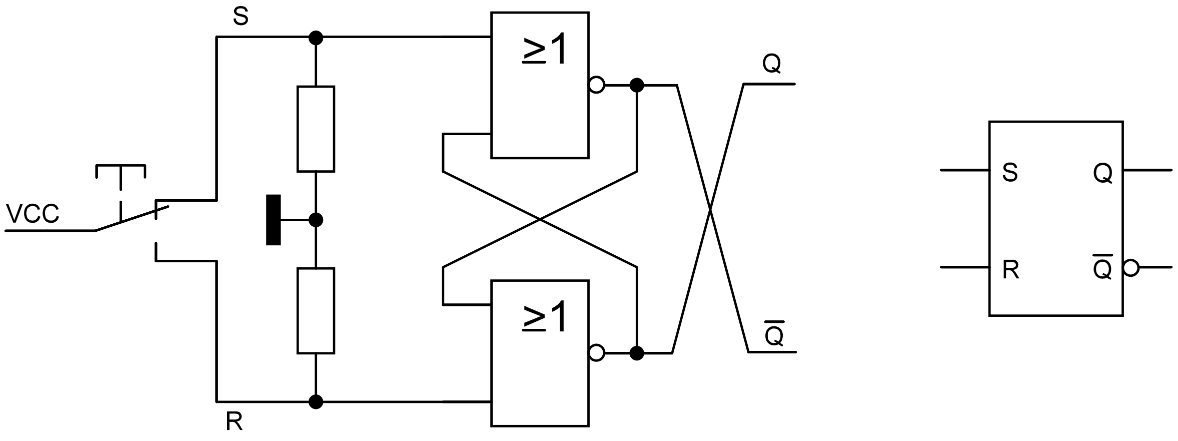


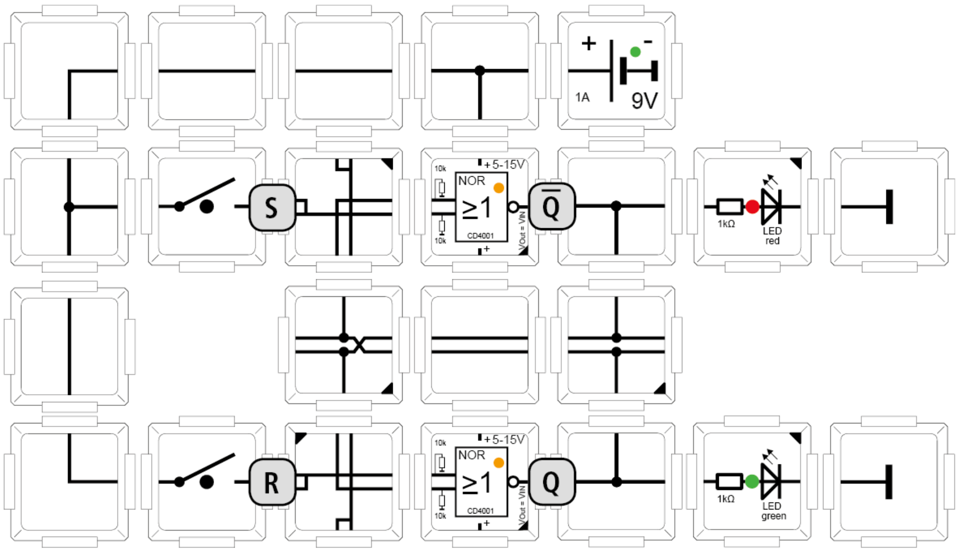
Figure 34: Flip-flop built with two NOR gates (left) and the circuit symbol of the RS flip-flop (right)

In the cicuit symbol of the RS flip-flops, the inputs are called S (set) and R (reset). Q is negated to Q. Please consider that the outputs Q and $\overline{Q}$ are crossed!

**Truthtable of not edge-triggered RS flip-flops**

| Set input S | Reset output R | Output Q | Output $\overline{Q}$ | Discription |
|---|---|---|---|---|
| 1 | 0 | 1 | 0 | Set (SET) |
| 0 | 0 | Q | $\overline{Q}$ | No change |
| 0 | 1 | 0 | 1 | Reset (RESET) |
| 1 | 1 | 0 | 0 | Prohibited state |

Beachte, dass sich in der folgenden Brick-Schaltung Ausgang Q unten und Ausgang $\overline{Q}$ is above!



- Set: By pressing the push button at the S input, the output Q is set to "1".
- No change: As long as no push button is pressed, the outputs don´t change.
- Reset: By pressing the push button at the R input, the output Q is set to "0".
- Prohibited

### 5.4.3 D flip-flop

The D flip-flop consists of a RS flip-flop, whereby the reset input is connected with the set input via an inverter. This prevents the occurrence of the undefined, metastable state.

The D flip-flop is available as state-trigerred and edge-triggered flip-flop, as in this Logic Set. If a D flip-flop also containes RS inputs, it is also possible two set and reset it clock-independtly (asynchronous). See also in chapter 5.2 on page 33).

The D flip-flop is the basic element of the read-only memory. In practice, many D flip-flops are usually connected in parallel and synchronized via a common clock. Such an arrangement of mostly 4, 8, 16 or 32 D flipflops is called a register. A practical application is, for example, a status register which is to be read in order to receive information about a device or a command register to transmit commands to a device. The only input is called a data input. In our case, the value at the data input ("0" or "1") with the positive edge at the clock input C is accepted.

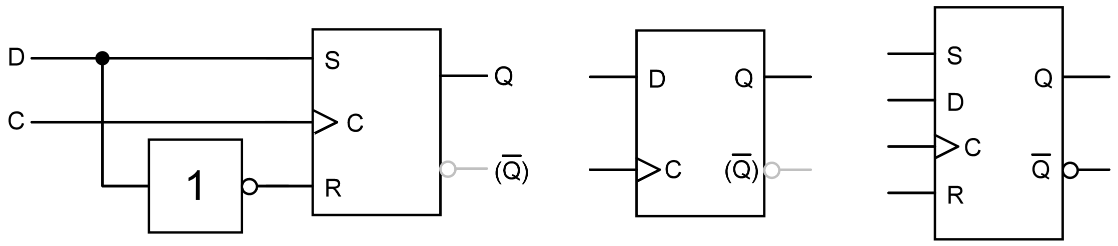


Figure 36: D flip-flop built with an clock-triggered RS flip-flop (left), circuit symbol of a normal D flip-flop (middle) and with two RS inputs (right).

Consider that the standard D flip-flop brick which is included in the Logic Set does not contain a $\overline{Q}$-output.

**Truthtable of a positive edge-trigerred D flip-flop**

| Data input D | Clock input C | Output Q | (Output $\overline{Q}$) | Discription |
|---|---|---|---|---|
| 1 | $0 \rightarrow 1$ | 1 | (0) | Set |
| 0 | $0 \rightarrow 1$ | 0 | (1) | Reset |
| 1 | $1 \rightarrow 0$ | Q | ($\overline{Q}$) | No change |
| 0 | $1 \rightarrow 0$ | Q | ($\overline{Q}$) | No change |

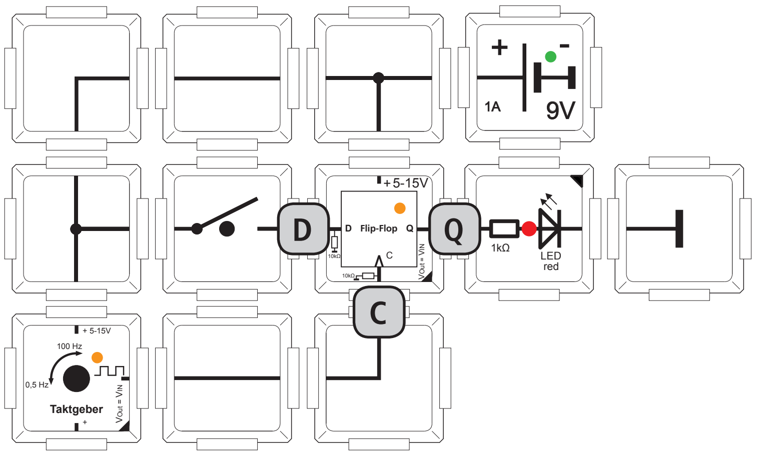


Figure 37: Brick circuit with standard D flip-flop

**D flip-flop with asynchronous set and reset input**

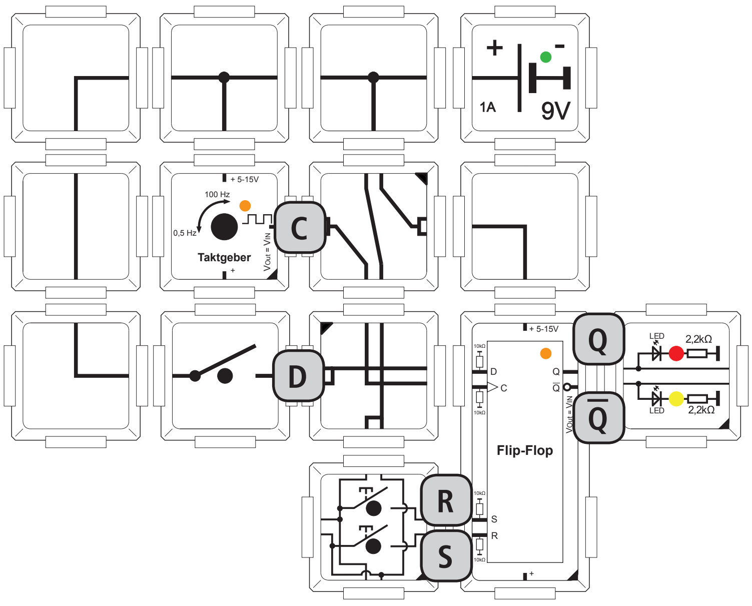


Figure 38: Brick circuit with D flip-flop and asynchronous set and reset input

## 5.4.4 JK flip-flop

The JK flip-flop is available as edge-triggered and state-triggered flip-flop. The Brick'R'knowledge Logic Set contains four (one-)edge-triggered JK flip-flops. If the clock edge receives a positive singal, it changes the input state, if both of the inputs J and K also receive a "1". This behaviour is called "Toggling", meaning switching. On a T flip-flop there are also two connected J and K inputs which are available for toggling.

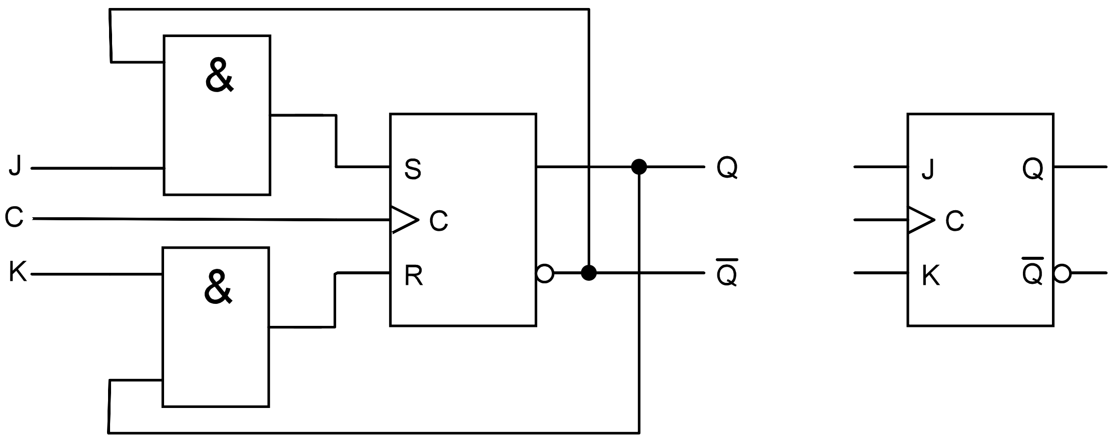


Figure 39: Structure of a JK flip-flop (left) and circuit symbol for one-edge-triggered JK flip-flop (right).

You need a square wave signal at the clock input of the JK flip-flop. The JK flip-flop brick always reacts to a positive clock edge, therefore to the change from "0" to "1" at the clock input C. The two input J and K are control inputs. The outputs Q and $\overline{Q}$ are controlled depending on the control of the inputs J and K.

**Truthtable for positive edge-triggered JK flip-flops**

| Input J | Input K | Clock input C | Output Q | Output $\overline{Q}$ | Disciption |
|---|---|---|---|---|---|
| 0 | 0 | $0 \rightarrow 1$ | Q | $\overline{Q}$ | No change |
| 1 | 0 | $0 \rightarrow 1$ | 1 | 0 | Set |
| 0 | 1 | $0 \rightarrow 1$ | 0 | 1 | Reset |
| 1 | 1 | $0 \rightarrow 1$ | $\overline{Q}$ | Q | Toggle outputs |

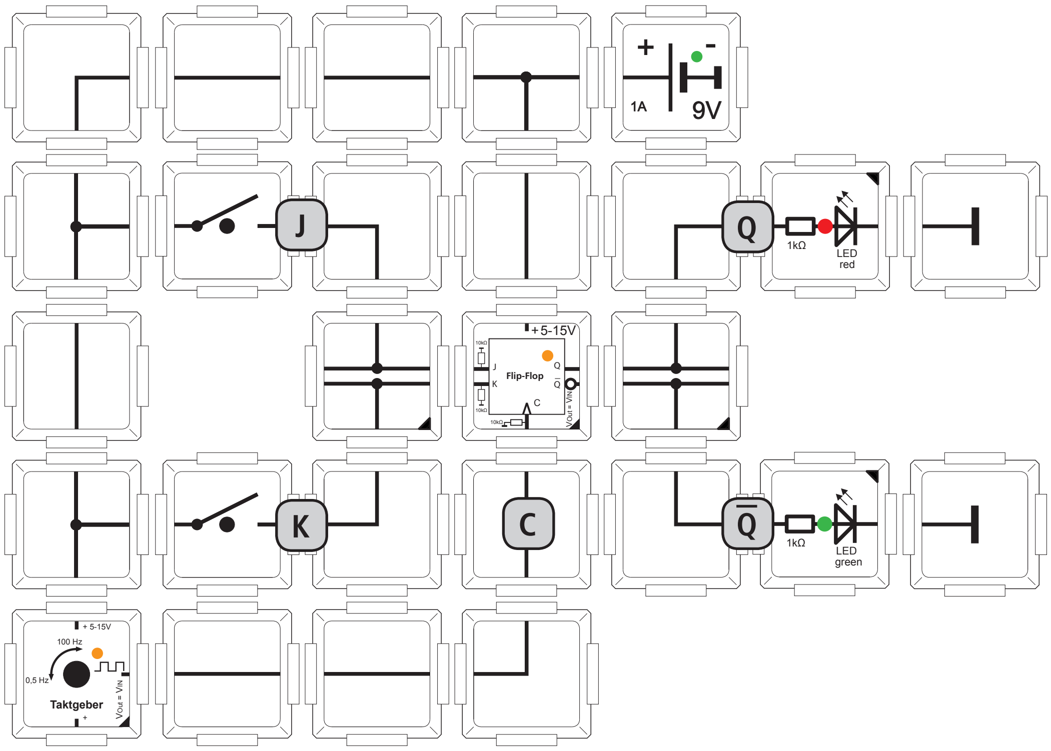


Figure 40: JK flip-flop as brick circuit

- No change: If no push button is pressed, the outputs Q and $\overline{Q}$ rest unchanged.
- Set: By pressing the push button at the J input, the output Q is set to 1 and the output $\overline{Q}$ is set to "0".
- Reset: By pressing the push button at the K input (button J not pressed), the output Q is set to "0" and the output $\overline{Q}$ is set to "1".
- Toggling outputs: If both push buttons are pressed, the outputs toggle. This means for the Q output: a "1" at the clock edge becomes a "0" and a "0" at the clock edge becomes a "1". The output $\overline{Q}$ is equivalent inverted.

## Alternative JK flip-flop circuit

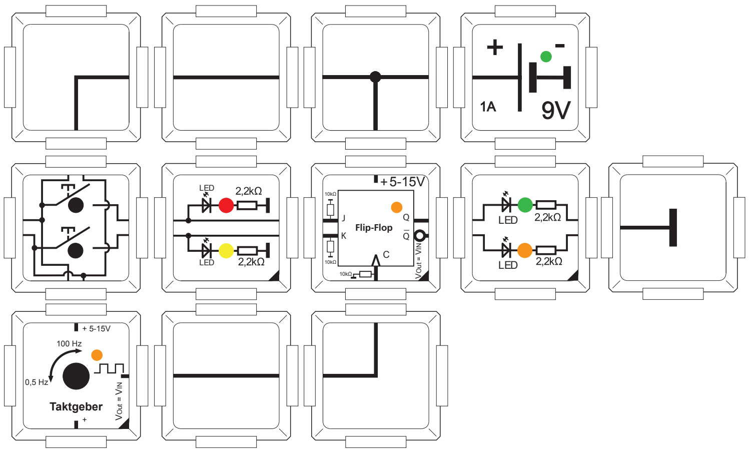


Figure 41: Alternative JK flip-flop circuit

## 5.5 Shift register

Shift registers consist of flip-flops that are connected in series and are pulsed synchronously. The bit-by-bit shifting is -besides the addition (see also chapter 5.3 on page 34)- one of the elementary operations of an arithmetic unit. If an imported binary number is shifted one digit to the right, the decimal result is equal to a division with 2. For multiplying, the binary number has to be shifted to the left. Depending on the application, shift registers have different bit width and the shifting to the left or right can be loaded in series or in parallel.  For a lot of applications shift registers as integrated circuits (ICs) are suitbale because they already include most of the standard functions.

Possible fields of applications:

- Arithmetic operations (multiplication, divison)
- Ring buffers
- Serial-parallel transformers and parallel-serial transformers
- Delay of signals

### 5.5.1 Shift register with serial charging

If a shift registers is loaded with a "1" at the D input, this value is shifted one digit (flip-flop) with each clock.

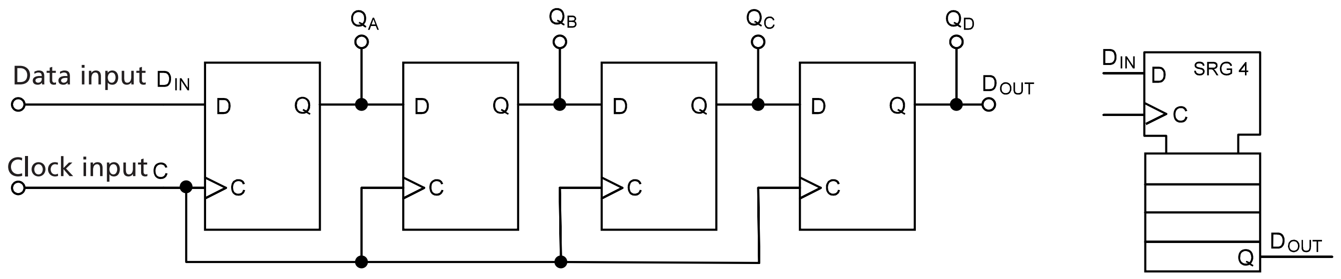


Figure 42: 4 bit shift register built with a D flip-flop with serial input and output (left), circuit symbol (right)

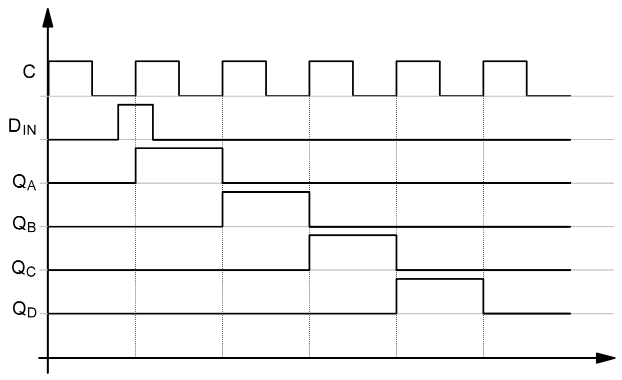


Figure 43: Timing diagram of a shift register

In the following brick circuit, the shift register is built with four D flip-flops. As long as the push button is pressed, the level at the data input of the first flip-flop is high. At the next positive edge of the clock C, the "1" is adopted and is exported at $Q_A$. ZAt the same time, the level of  $Q_A$ an den Dateneingang des zweiten Flipflops weitergegeben. This principle continues until after four clock cycles, when the 4-bit data word is in the shift register ($Q_A$ to $Q_D$). $Q_D$ is in our example the serial data output  $D_{OUT}$. Alternatively it could also be connected with $D_{IN}$ to build a ring register in which the bits are shifted in a circle.

In our case, the pulse is generated with a clock brick. On the inside of the brick you can find the well known Timer 555. You can change the clock frequency between 0,5 to 100 Hz.

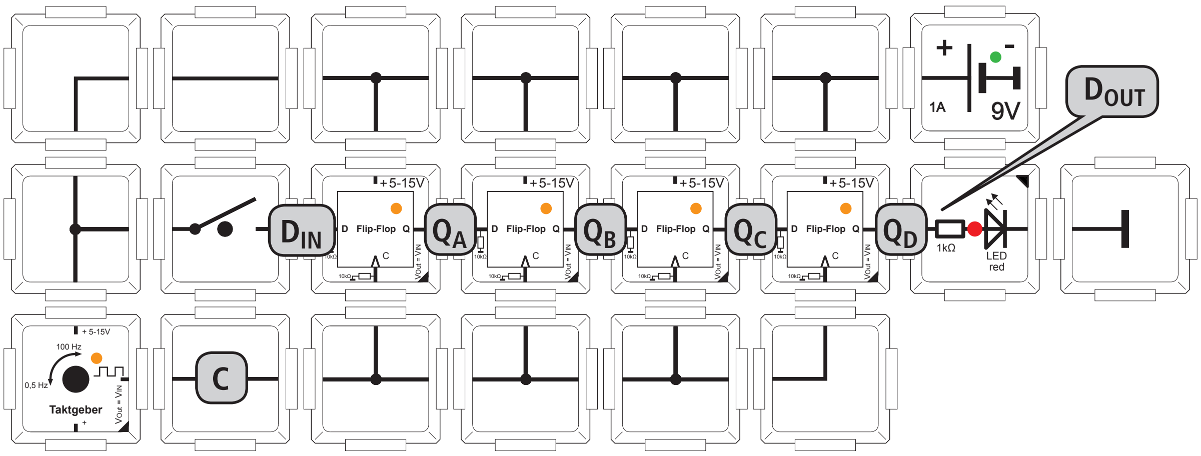


Figure 44: Shift register as brick circuit

## 5.6 Counter

Flip-flop circuits are perfectly suited as digital counters. The binary counters are differed in control and direction of counting. In case of pulsing, you differ between two basic counting types: the asynchronous and synchronous counters. The clock signal should always be debounced to avoid counting errors caused by contact bounces. See more in chapter 5.2 on page 33.

In synchronous counters, all memory flip-flops are pulsed by the same counting pulse. There is no addition of the signal delays as with asynchronous counters caused by the pulse transfer from step to step. The output levels on all steps occur at the same time after the clock pulse edge. Compared to the asynchronous contai-ners the synchronous method allows significant higher maximum counting frequencies. Wheras the circuit effort is a lot higher with synchronous counters.

The clock control of asynchronous counters occurs for all counting modules. The counting pulse only controls the first fliop-flop. All following flip-flops are being controlled by the output level of its predecessors. As it is like with all asynchronous operations, chaotic circuits with runtime problems can occur. In contrast, the gate effort is getting smaller. Asynchronous counters are also callled "ripple counters" because the control pulse between the flip-flops reminds of a wave.

Syncronous as well as asynchonous counters are availabe as upwards and downwards counters. By adding an additional circuit you can change between the counting directions and limit the counting sector. When required, the counting value can be issued in different coded styles (e.g. binary, BCD, gray counter). See also in the chart on page 18.

The universal basic element of a binary counter is a JK flip-flop which is built as a toggle flip-flop. Each flip-flop is equivalent to a binary digit and stores one bit. By connecting the flip-flops in a suitable way, you can create multi-digit binary counters (see more in the following chapters).

The capacity K of a binary counter depends on the amount n of the individual momories (flip-flops). The following formular applies to a binary counter:

$K = 2^n - 1$. You can find the values for n = 2 to 10 in the chart below:

**Dependence on counter depth and counting capacity**

| Amount of flip-flops n / (Counter depth in bits) | Counting capacity (K) |
|---|---|
| 2 | 3 |
| 3 | 7 |
| 4 | 15 |
| 5 | 31 |
| 6 | 63 |
| 7 | 127 |
| 8 | 255 |
| 9 | 511 |
| 10 | 1023 |

In practice mostly 4 or 8 bit counters are used. When required two 8 bit counters can be combined to one 16 bit counter for example. In modern circuit design, counters are integrated as logic blocks in programmable modules e.g. FPGAs (Field-Programmable Gate-Array). See also in chapter 4.8 on page 23.

### 5.6.1 Asynchronous 4 bit binary counter

Asynchronous counters don´t have a combined clock. The flip-flops are connected in series, this means that the output of the first flip-flop is connected with the clock input of the second flip-flop and so on. Because of the signal transit time, each flip-flop receives the clock pulse edge at another time at the clock input. The signal delays of the individual memory parts add up. This effect is especially important at binary counters with a big counting depth. To achive a correct counting, the clock time always has to be longer that the total duration of the signal through the complete counter.

If the shortest switching time (in seconds) of n equal flip-flops is known you can calculate the maximum frequency $f_g$ of the counting pulse in Hertz [Hz] with this formula:

$$f_g = \frac{1}{(n+1) \times t}$$

The singal duration in flip-flops with CMOS or TTL technology only takes a few nano seconds. The bigger the counting depts (e.g. 16 bits) the longer it takes the signal to go from the first to the last flip-flop. This duration of counting impulses can lead to failures and thus to counting errors. The higher the counting frequency, the more easily problems occur. As long as you only count in the range up to 100Hz - as with our clock brick - a 4 bit asynchronous counter is no problem.

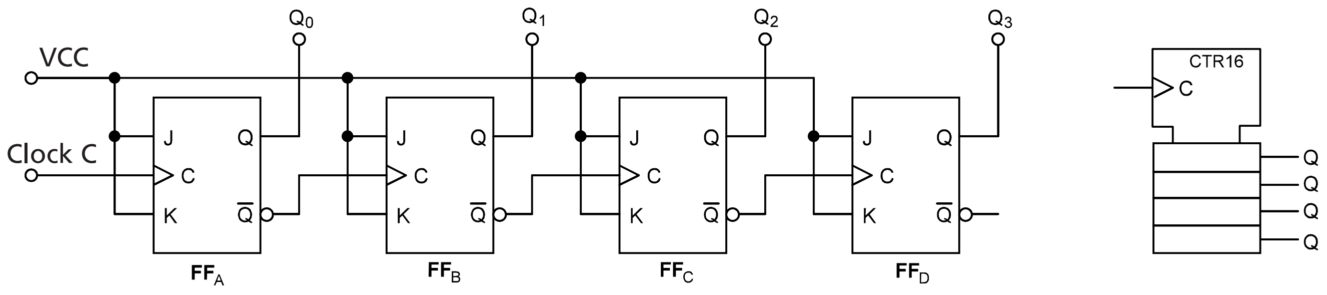


Figure 45: Asynchronous 4 bit binary counter (left), circuit symbol (right)

#### 5.6.1.1 Asynchronous upwards counter

The following brick circuits represents an asynchronous upwards counter. It´s easy to observe the counting operation with the colorful LED bricks that are connected to the Q output of the flip-flops (see also the timing diagram figure 47). The series connection of the flip-flop steps is achieved by connecting the output $\overline{Q}_0$ of the first step with the clock input C on the second step etc. Since we are using an one-edge-triggered JK flip-flop, each transfer from the set to the reset state causes a positive edge at the $\overline{Q}$-output. Since the J input as well as the K input is connected with high level, this causes a toggling of the folloing flip-flops.

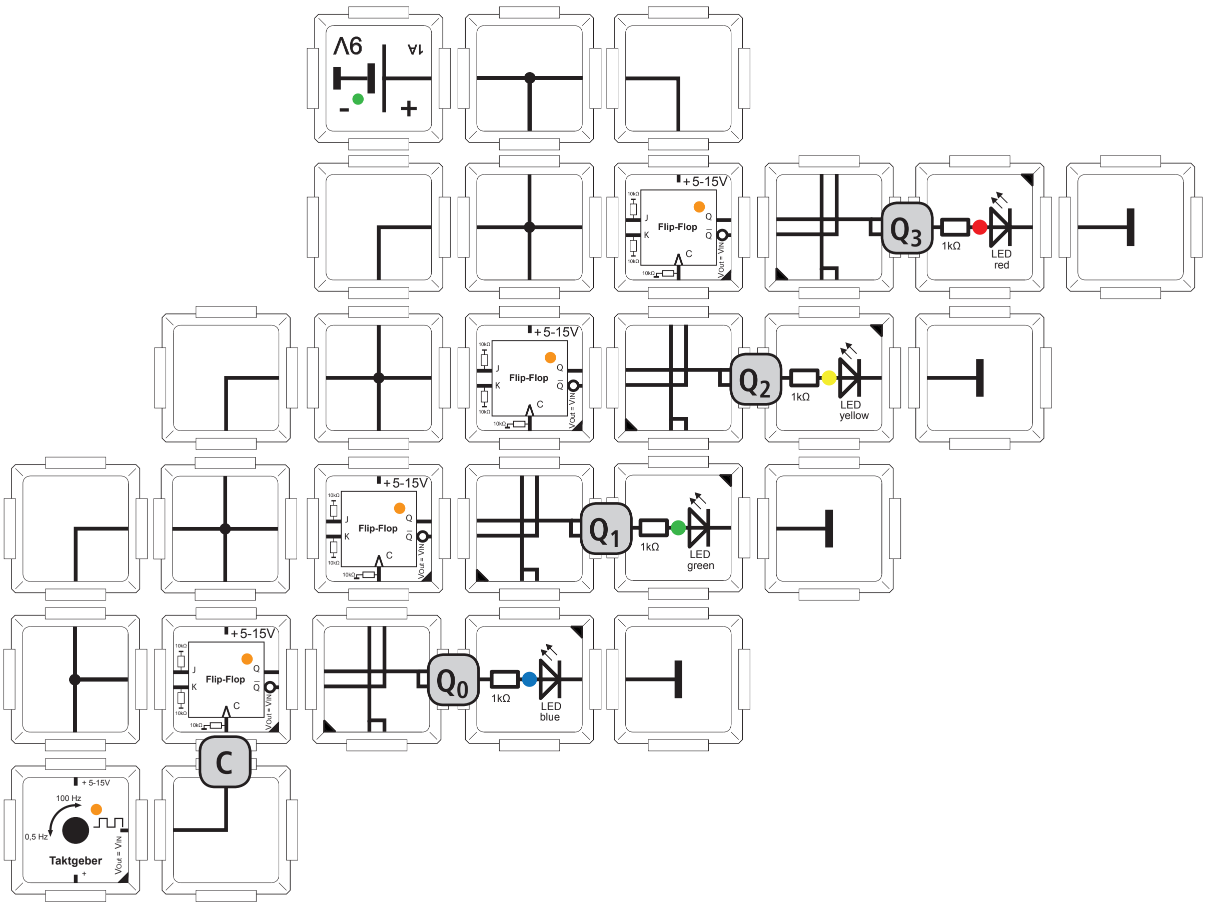


Figure 46: Asynchronous 4 bit binary upwards counter built with JK flip-flops as brick circuit

In the following diagram you can see a 4 bit binary upwards counter that counts from 0 to 15 and then starts at 0 again. Please consider that you need a debounced signal at the clock input C. By using the clock brick this is always gururanteed. Alternatively you can use the debounced push button to understand the counting steps better.

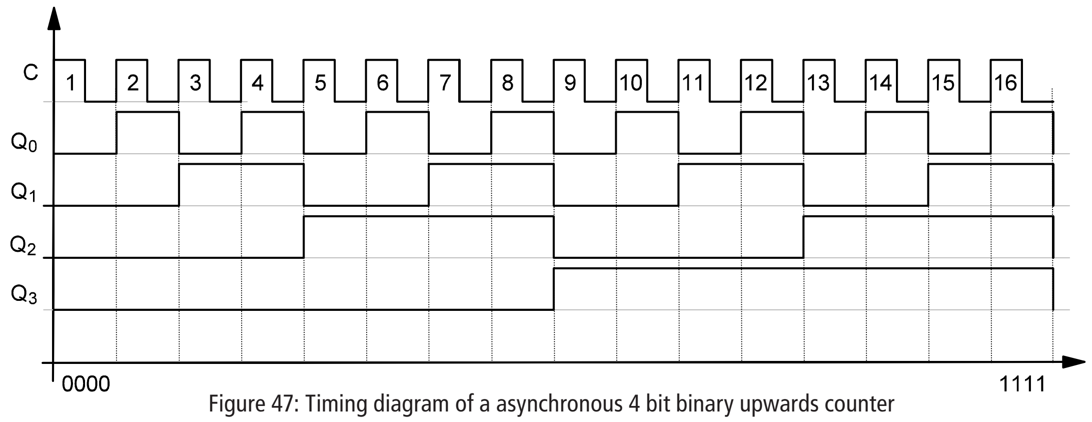


Figure 47: Timing diagram of a asynchronous 4 bit binary upwards counter

### 5.6.1.2 Asynchronous downwards counter

The following brick circuit represents a asynchronous downwards counter. The counting operation can be observed by the orange LEDs inside of the flip-flop that is connected to the Q output (see also timing diagram figure 49). The series connection of the flip-flop steps is achieved by connecting the output $Q_0$ of the first step with the clock input C on the second step etc. Since we are using an one-edge-triggered JK flip-flop, each transfer from the set to the reset state causes a positive edge at the Q output. Since the J input as well as the K input is connected with high level, this causes a toggling of the folloing flip-flops.

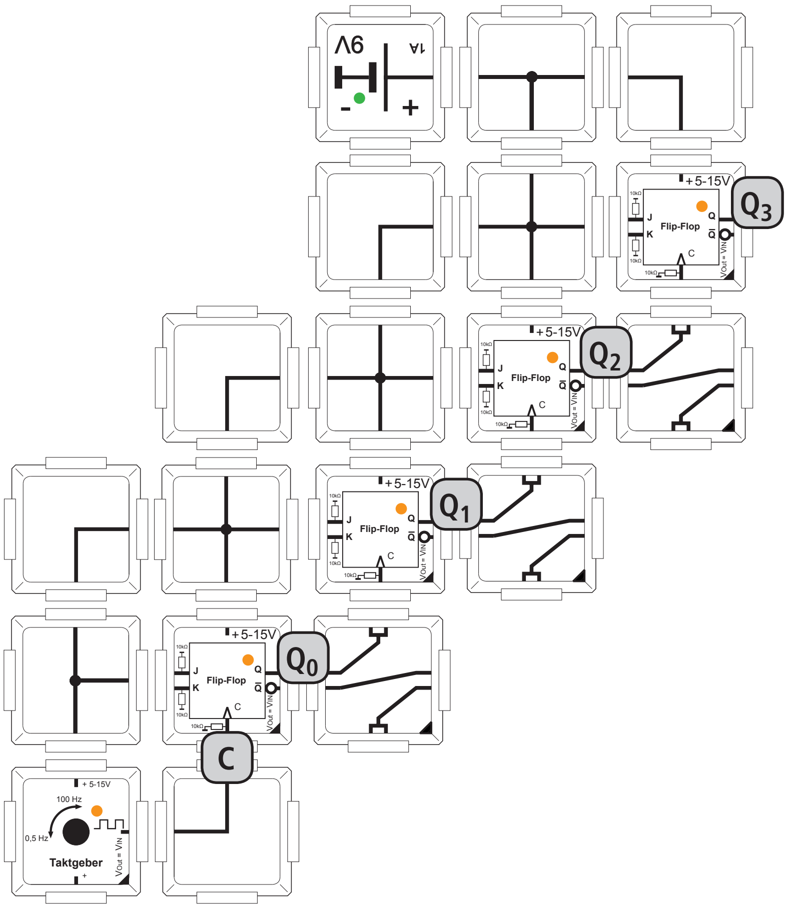


Figure 48: Asynchonous 4 bit binary downwards counter built with JK flip-flops as brick circuit

In the following diagram you can see a 4 bit binary downwards counter that counts from 15 to 0 and then starts at 0 again. Please consider that you need a debounced signal at the clock input C. By using the clock brick this is always guraranteed. Alternatively you can use the debounced push button to understand the counting steps better.

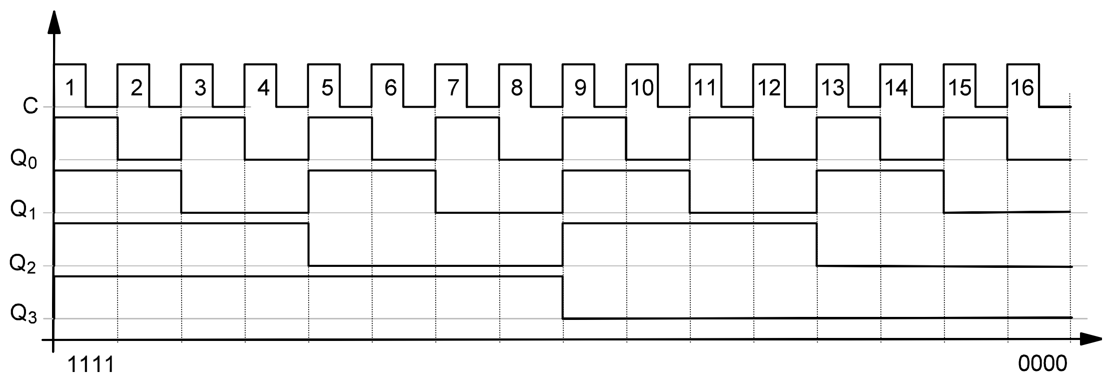


Figure 49: Timing diagram of a asynchronous 4 bit binary downwards counter

**Please consider that no additional load e.g. a LED should be connected to the clock input C of the JK flip-flops.**

### 5.6.2 Asynchronous 4 bit BCD counter

BCD counters are based on a 4 bit binary counter. If you convert binary numbers into decimal numbers for display purposes you can also use binary numbers in the decimal system. For the numbers 0 to 9 of each decade you need a 4 bit binary counter. Since it however only resets to 0 after the 16th counting run, you need a additional circuit to achieve a reset after 10 runs. Binary counters with this property are called BCD counters (binary coded decimal). BCD counters are available as forwards, backwords and switchable counters.

The basic circuit arrangement of the asynchronous BCD counter is similar to the asynchronous binary counter. However the BCD counter has to be extended so that it counts back to 0 after the 9th counter status. Remeber that right now we only look at one decimal digit. In the BCD counter, the output $Q_1$ should not switch to "high" after the 10th input pulse. Therefore its J input $J_B$ is connected to the $\overline{Q}$-output of the flip-flop D ($\overline{Q_3}$). The flip-flop D does not shift until the 8th pulse at the $\overline{Q}$-output from high to low. Until then the flip-flop B works like a toggle flip-flop and resets not until the 8th pulse back to low. The flip-flop C works unchanged as a frequency divider.

Flip-flop D is getting pulsed by the output $\overline{Q}$ of the flip-flop A. It can only be set to high if the J inputJ-Eingang $J_B$ receives a high level via the AND gate. This happens after the 6th pulse, when $Q_1$ and $Q_2$ are set. However at this time, the pulse $Q_0$ is low for the flip-flop D, which means that its output can not be set. With the falling edge of $Q_0$ after the 8th pulse, $Q_3$ is set. The binary value 1000 is equivalent to the decimal number 8. At this time, the level at the J input of flip-flop D changes to low. This has no impact until the end of the 9th pulse, because only now the flip-flop D, which is controlled by $Q_0$ is set back to low after the 10th pulse. The binary value shows 0000.

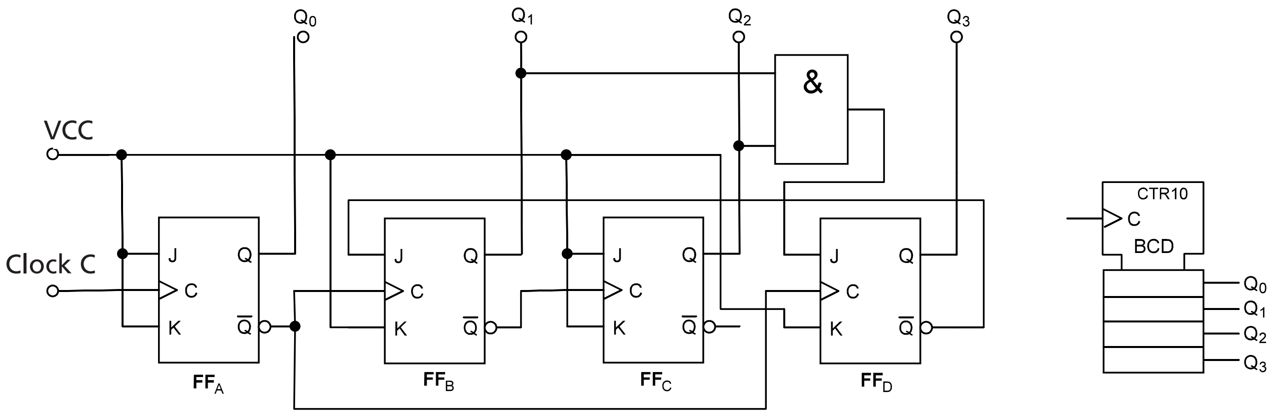
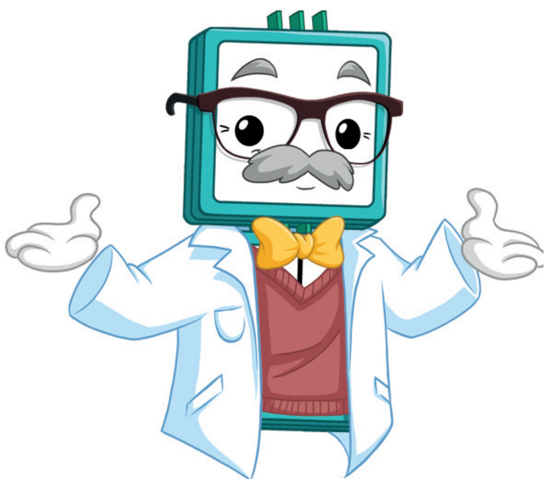
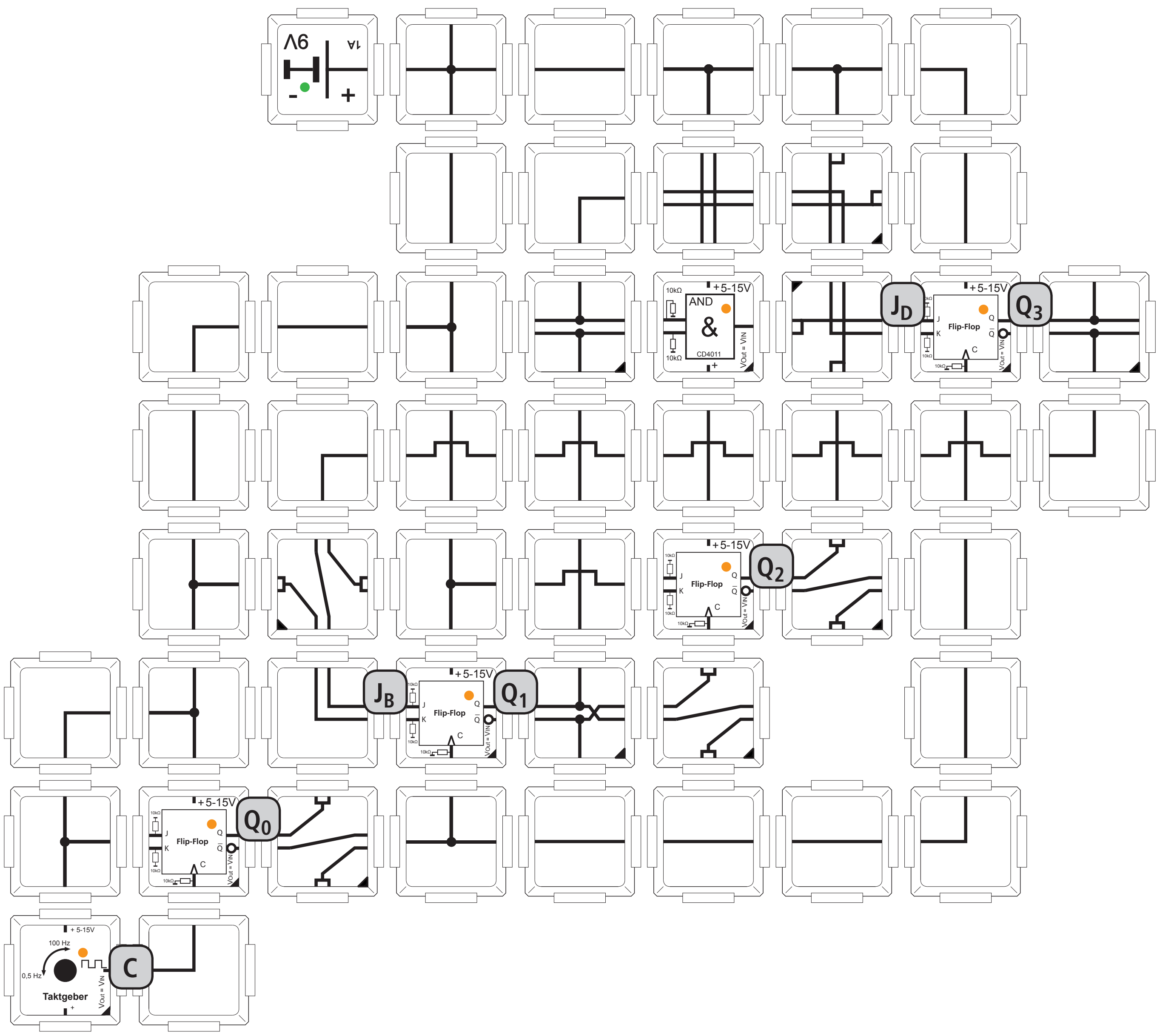


Figure 50: 4 bit BCD counter (left), circuit symbol (right)





Asynchronous 4 bit BCD counter built with JK flip-flops as brick circuit

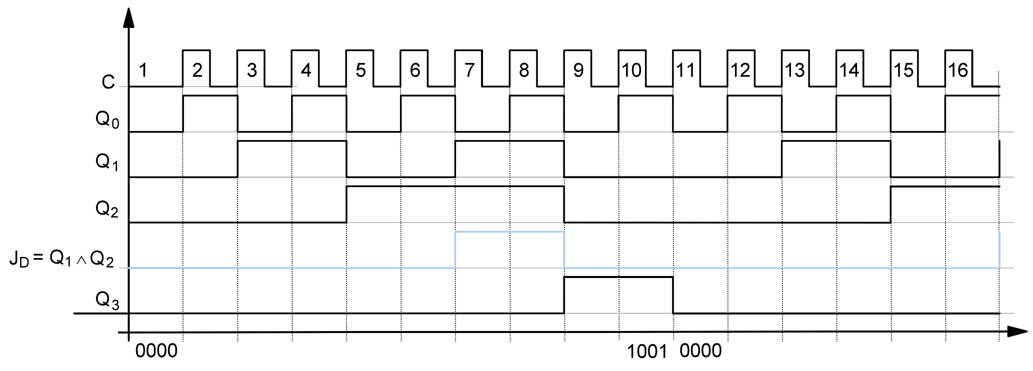


Figure 52: Timing diagram of an asynchronous 4 bit BCD counter

**Please consider that no additional load e.g. a LED should be connected to the clock input C of the JK flip-flops.**

## 5.6.3 Synchronous 3 bit binary counter

Synchronous counters have a clearer structure and is easily expandable. Its combinatorial circuit is more complex, but there are no run time problems.

When the pulse changes, the input levels of each memory step decides if the flip-flop is reset or rests in its previous state. Since all information at its inputs have to be set at the beginning of the clock pulse, synchronous counters can´t be built with T flip-flops. RS flip-flops are only limitedly suitable as well because the circuit effort is too extensive due to the additional circuit.

The most suitable flip-flop is therefore the JK-Master-Slave flip-flop which J and K inputs can be controlled via additional gates. In our example we use a one-edge controlled JK flip-flop. The J and K inputs of the first flip-flop have a high level which equals the toggling operation. The output controls the connected J and K inputs of the following stage. The J and K inputs of all following flip-flops are connected with the AND gate which interprets the output levels of the previous memory step.

Due to better clarity the folloing example for a syncronous counter is only 3 bits broad. This way it its possible to count up to 7.

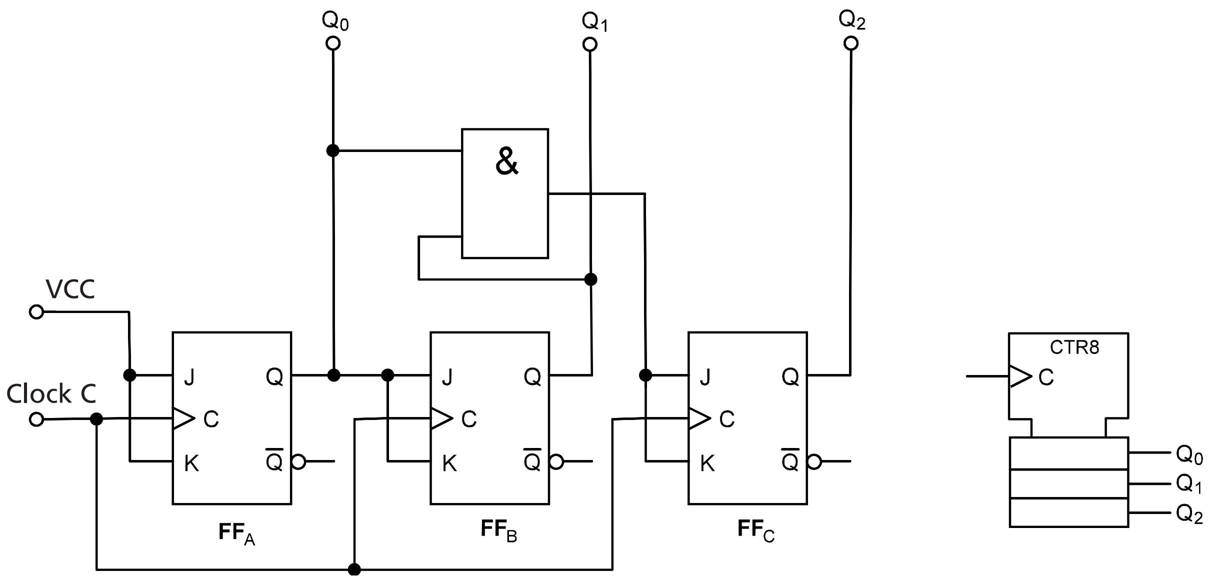


Figure 53: Synchronous 3 bit binary upwards counter (left), circuit symbol (right)

**State table for a 3 bit synchronous upwards counter**

| Clock | $Q_2$ | $Q_1$ | $Q_0$ | Equation |
|---|---|---|---|---|
| 1 | 0 | 0 | 0 | |
| 2 | 0 | 0 | 1 | |
| 3 | 0 | 1 | 0 | |
| 4 | 0 | 1 | 1 | $Q_0 \wedge Q_1$ |
| 5 | 1 | 0 | 0 | |
| 6 | 1 | 0 | 1 | |
| 7 | 1 | 1 | 0 | |
| 8 | 1 | 1 | 1 | |

The counter has drei memory flip-flops and can count binary from 000 up to 111. With the help of the status table for the outputs Q0 to Q2 the gate circuit can be explained. Flip-flop A switches over at each pulse which results a halving of the clock frequency. Flip-flop B is only set new if output $Q_0$ his high, otherwise the previous state is saved. Flip-flop C has to be set as soon as $Q_0$ and $Q_1$ have a high level at the same time.

The new output levels appears after the rising clock edge. All output signal are set synchronously, meaning at the same time. Compared to the asynchronous adders, the signal delays of the individual memory steps do not add up.

**Please consider that no additional load e.g. a LED should be connected to the clock input C of the JK flip-flops.**

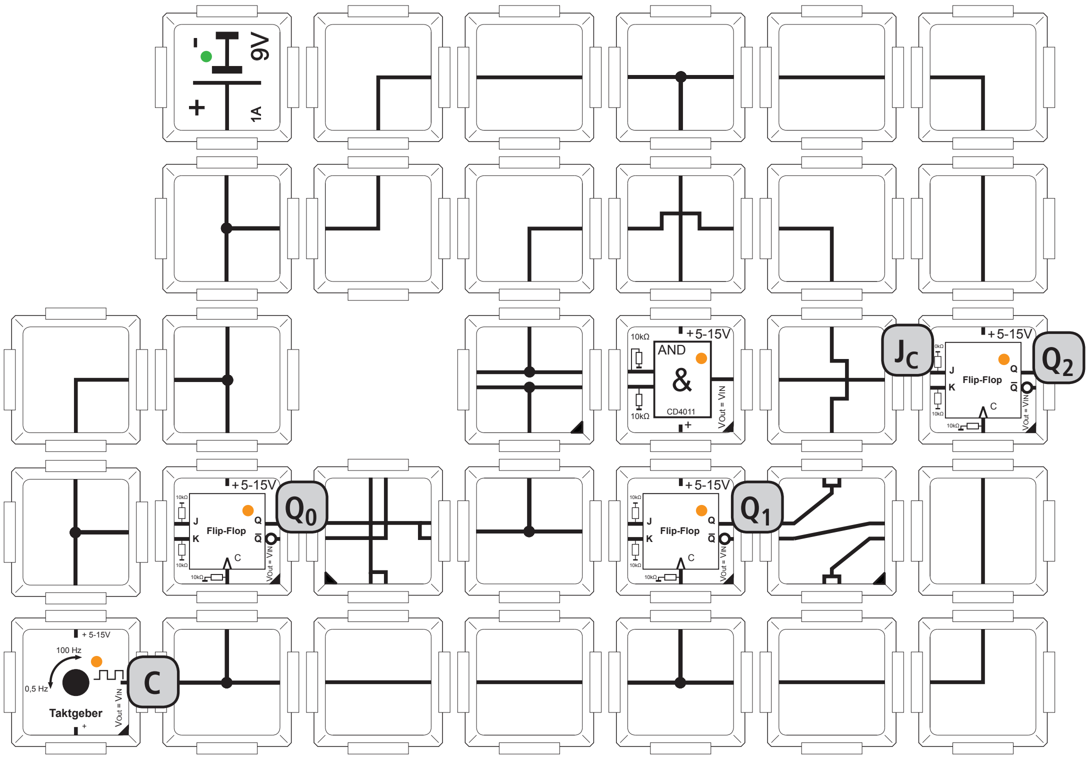


Figure 54: Synchronous 3 bit binary counter built with JK flip flops as brick circuits

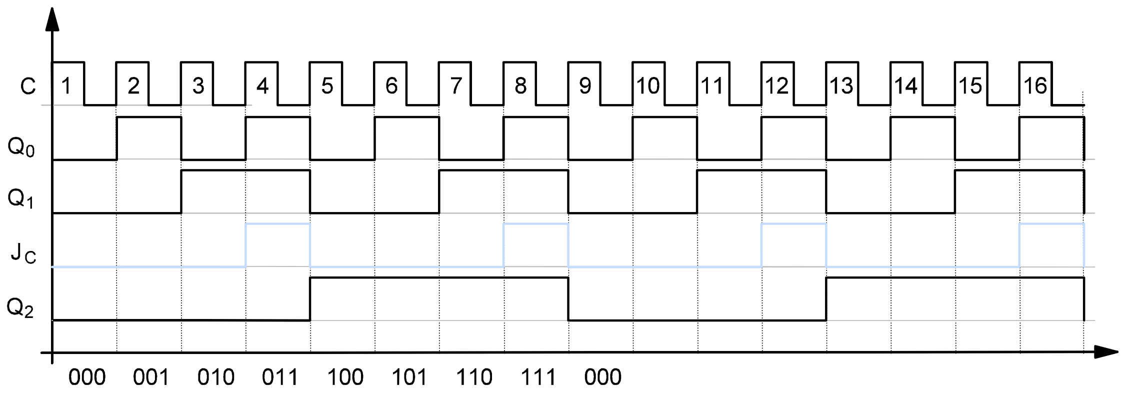


Figure 55: Timing diagram of a synchronous 3 bit binary upwards counter

**Alternative: downwards counter**
By adding LEDs at the $\overline{Q}$-outputs, you can transform the counter into a downwards counter.

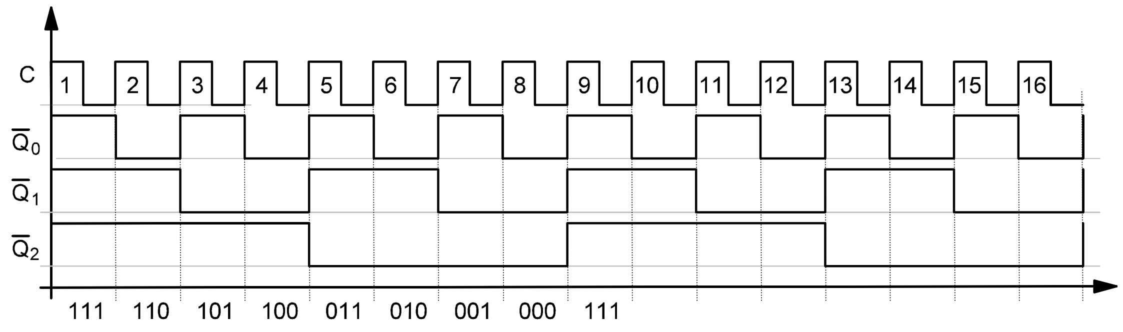


Figure 56: Timing diagram of a synchronous 3 bit binary downwards counter

## 5.6.4 BCD counter with carry out and reset

In this example we use the Brick'R'knowledge BCD counter for the first time. Internally they are built with two synchronous BCD counters that are are connected asynchronously via a transmission bit. By connecting them in series, you can count up to multi-digit numbers. The BCD-to-7-segment-display decoding is already integrated into the BCD counter brick.

The following figure shows the wiring of a BCD counter brick with two different possibilities of pulsing. On the left side you can see the version with a debounced button and on the right side the version with the clock brick that can deploys pulse frequencies up to 100 Hz.

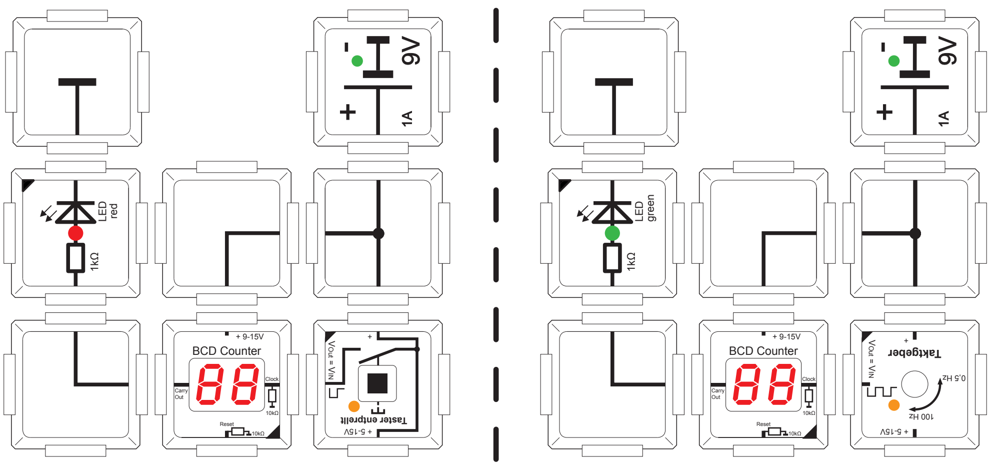


Figure 57: BCD counter with debounced button (left) and clock brick (right)

In our brick example we will build one 4-digit BCD counter by cascading two of the BCD counter bricks. The clock signal $\overline{\text{Clock}}$ and the transmission signals ($\overline{\text{Carry 1}}$ and $\overline{\text{Carry 2}}$) are displayed by the LEDs. An asynchronous reset (high active) is possible via the push button. The block diagram for our circuit looks as follows.

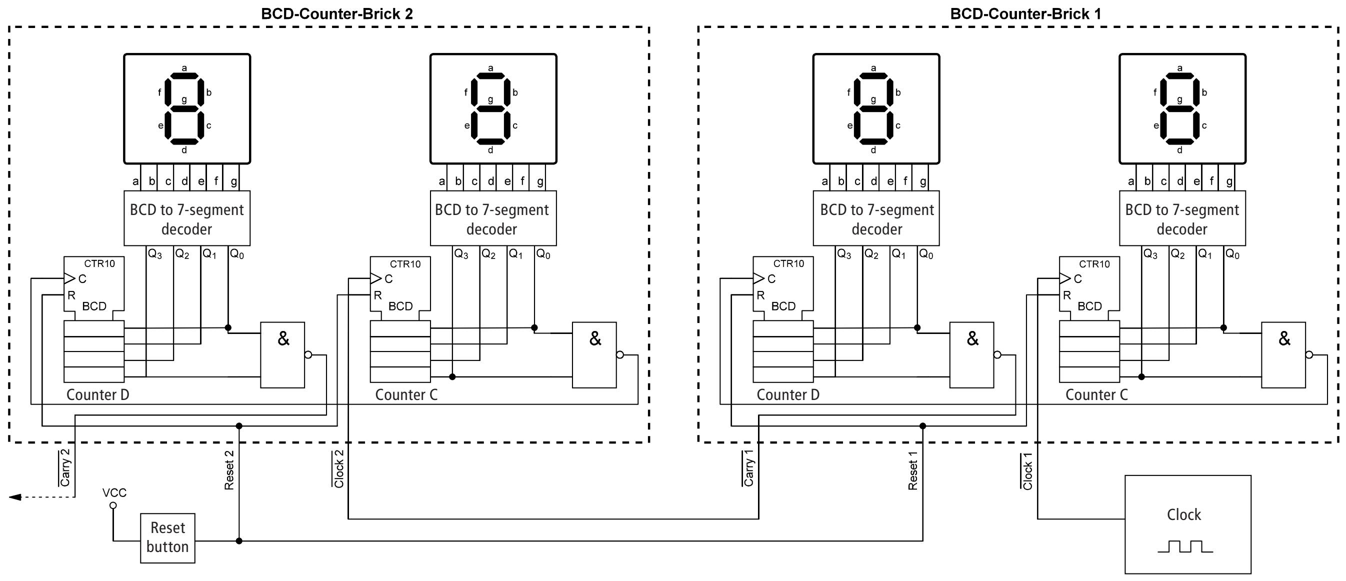


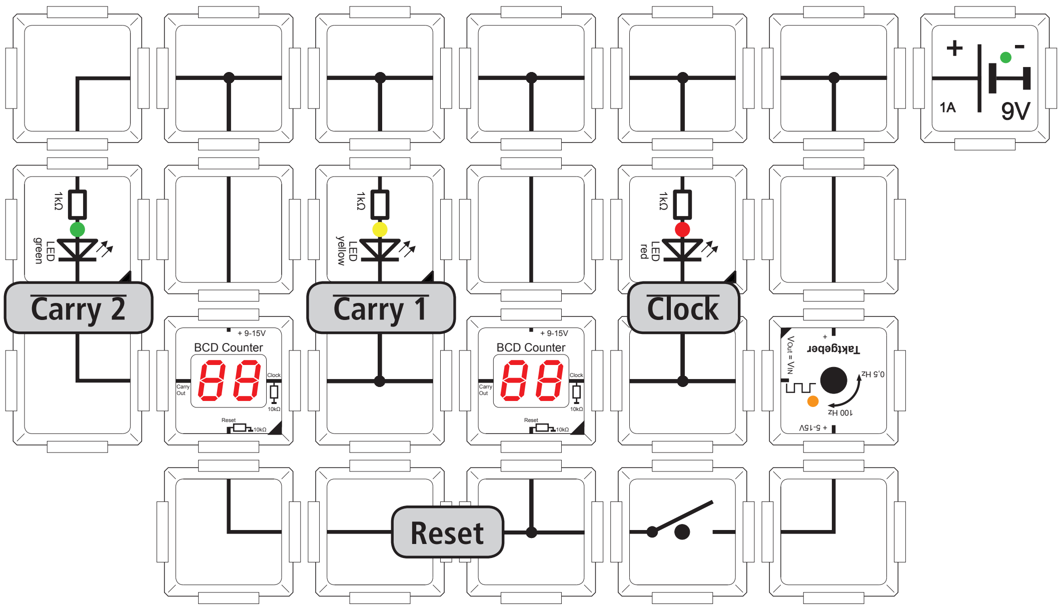
Figure 58: Block diagram of a 4 digits BCD counter

As soon as the BCD counter bricks reach the maximum number 99 they deploy the carry out. Since this signal is low active, we have to correctly write "$\overline{\text{Carry Out}}$" to show the negation. For being able to distinguish between the two carry out output of the brick circuit above,we write $\overline{\text{Carry 1}}$ and $\overline{\text{Carry 2}}$.

The $\overline{\text{Carry Out}}$-signal is in the case of our brick BCD counter 10 pulses long. This means that the carry out will be annouced at the number 90 already. You can see the this annoucement by the output because it changes to low and switches back to high (meaning not active) when the number changes from 99 to 00.

Consider that the $\overline{\text{Carry 2}}$-LED of the left, higher-grade counter brick is getting active at the maximum frequency of the clock brick for the first time after ca. 1 minute and 40 seconds. This means that the $\overline{\text{Carry 2}}$-output deploys a low level for a short time so that the LED flashes briefly.

The clock input "$\overline{\text{Clock}}$" is low active, this means that it counts on negative clock impulses.



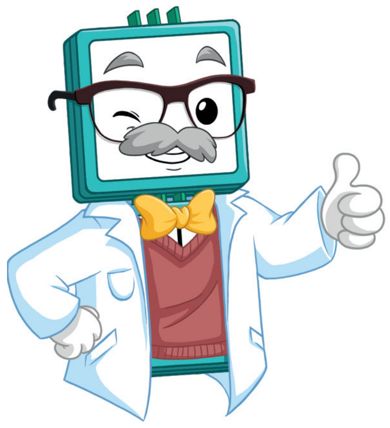


Figure 59: 4 digits BCD counter with carry out outputs and reset input

## 5.7 Frequency divider

Frequency dividers are circuits that divide the frequencies of signals at a certain splitting ratio. Most of the frequency dividers have a fixed integral splitting ratio. Besides that, you can find adjustable frequency dividers that have an additional input that sets the splitting ratio.

In principle, every counter is also a frequency divider. Example circuits:

- Asynchronous binary counter: see more in chapter 5.6.1 on page 46
- Synchronous binary counter: see more in chapter 5.6.3 on page 51

One single flip-flop already achieves a frequency division at a ratio of 2 : 1 (output $\overline{Q}_0$). With two flip-flops you can build a frequency divider at a ratio of 4 : 1 (output $\overline{Q}_1$), etc.

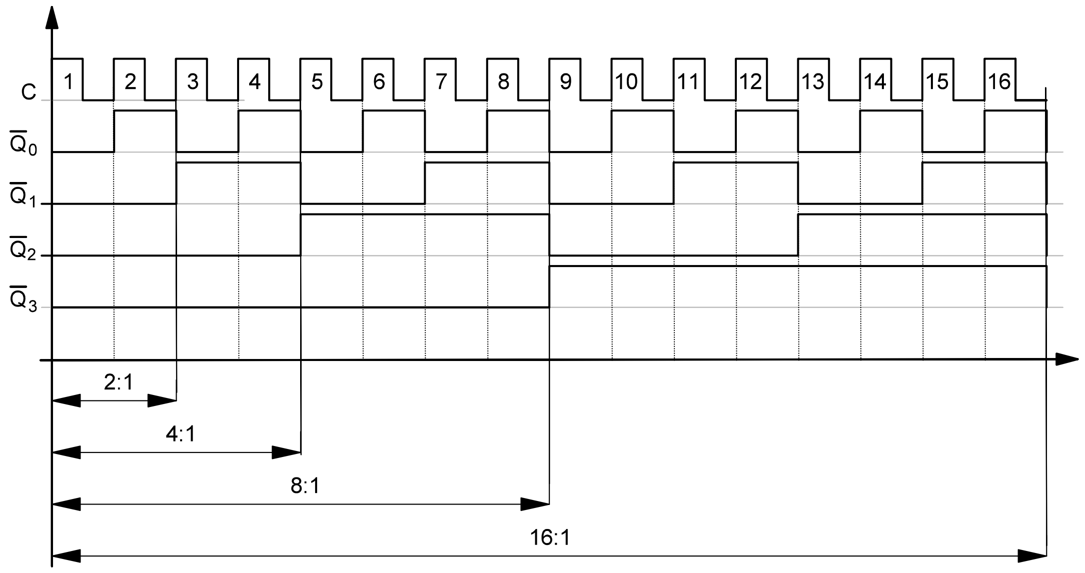


Figure 60: Timing diagram of a frequency counter (see also 4 bit binary counter in chapter 5.6.1 on page 46)

At a clock pulse of 100 Hz (Maximum frequency of out clock brick) you reseive the following frequencies at the $\overline{Q}$-outputs:

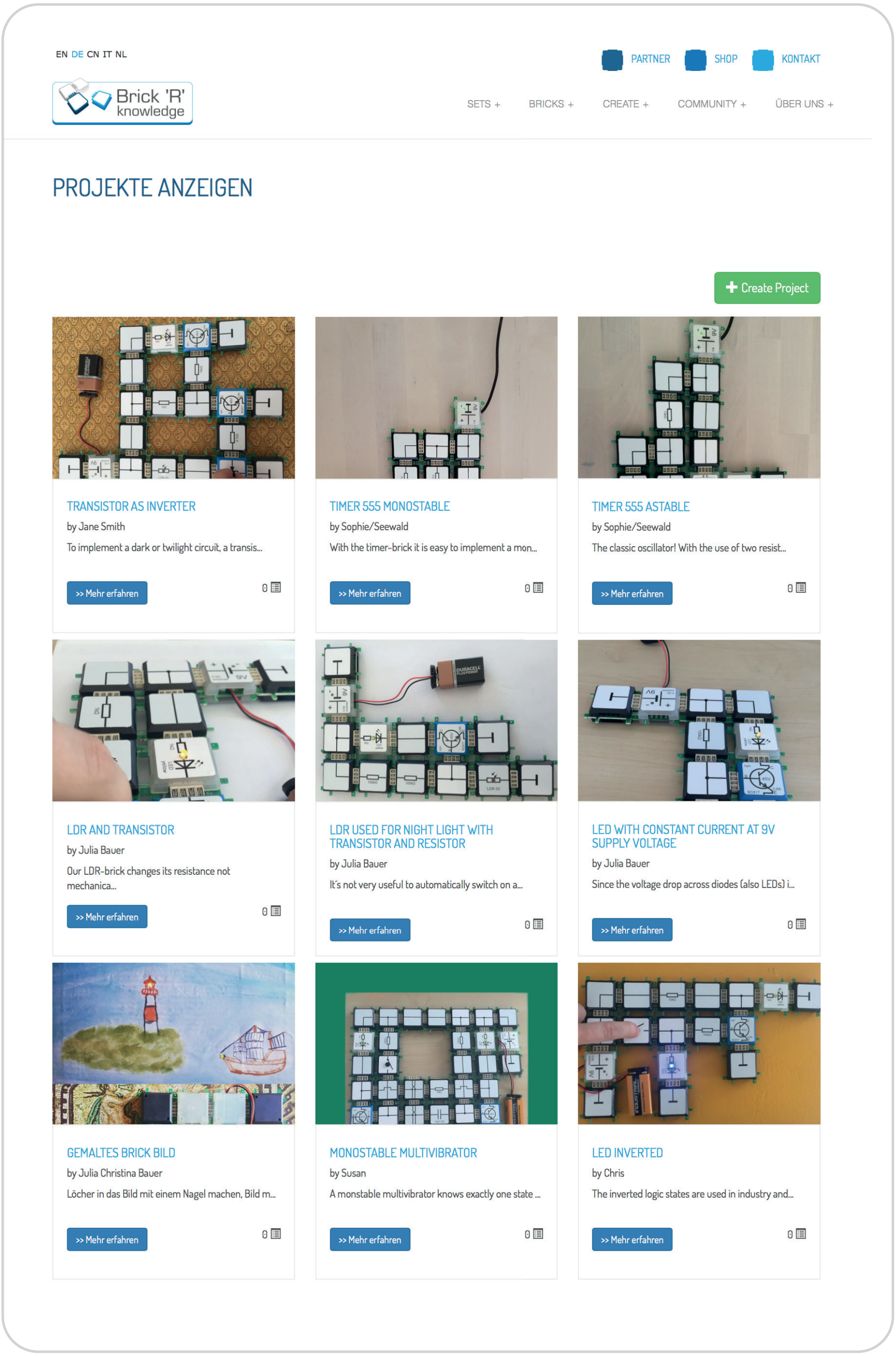
| Teilverhältnis | 2:1 | 4:1 | 8:1 | 16:1 |
|---:|:---:|:---:|:---:|:---:|
| **Output** | Output $\overline{Q}_0$ | Output $\overline{Q}_1$ | Output $\overline{Q}_2$ | Output $\overline{Q}_3$ |
| **Frequency** | 50 Hz | 25 Hz | 12,5 Hz | 6,25 Hz |

# 6. Brick Community

The brick universe is expanding: You can  nd more ideas, experiments and bricks at fairs, on our website, on youtoube or social media networks. Boost your creativity!

## More projects

By clicking on "Create" you can try out experiments from other users or show your own cool circuits.



EN DE CN IT NL

PARTNER SHOP KONTAKT

SETS + BRICKS + CREATE + COMMUNITY + ÜBER UNS +

PROJEKTE ANZEIGEN

+ Create Project

TRANSISTOR AS INVERTER
by Jane Smith
To implement a dark or twilight circuit, a transis...
>> Mehr erfahren 0

TIMER 555 MONOSTABLE
by Sophie/Seewald
With the timer-brick it is easy to implement a mon...
>> Mehr erfahren 0

TIMER 555 ASTABLE
by Sophie/Seewald
The classic oscillator! With the use of two resist...
>> Mehr erfahren 0

LDR AND TRANSISTOR
by Julia Bauer
Our LDR-brick changes its resistance not mechanica...
>> Mehr erfahren 0

LDR USED FOR NIGHT LIGHT WITH TRANSISTOR AND RESISTOR
by Julia Bauer
It's not very useful to automatically switch on a...
>> Mehr erfahren 0

LED WITH CONSTANT CURRENT AT 9V SUPPLY VOLTAGE
by Julia Bauer
Since the voltage drop across diodes (also LEDs) i...
>> Mehr erfahren 0

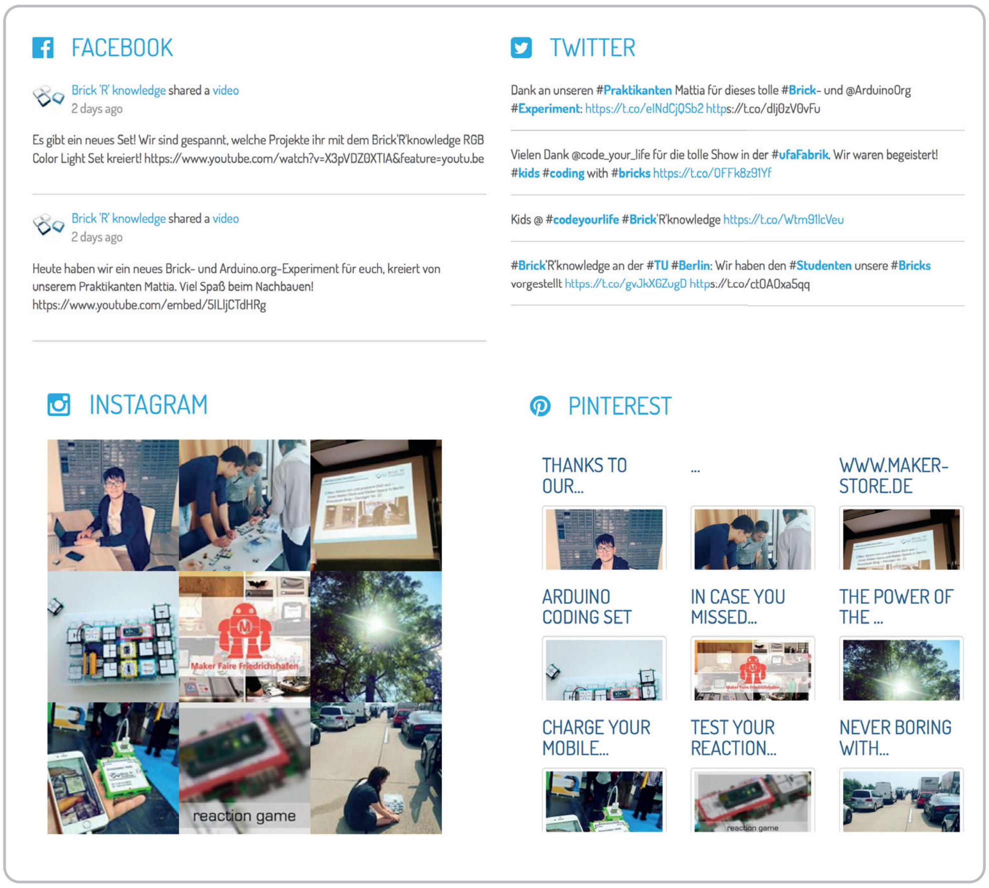
GEMALTES BRICK BILD
by Julia Christina Bauer
Löcher in das Bild mit einem Nagel machen, Bild m...
>> Mehr erfahren 0

MONOSTABLE MULTIVIBRATOR
by Susan
A monstable multivibrator knows exactly one state ...
>> Mehr erfahren 0

LED INVERTED
by Chris
The inverted logic states are used in industry and...
>> Mehr erfahren 0

## Social Media

By clicking on "Community" you can  nd all of our social media networks. Stay up-to-date!



FACEBOOK

Brick 'R' knowledge shared a video
2 days ago

Es gibt ein neues Set! Wir sind gespannt, welche Projekte ihr mit dem Brick'R'knowledge RGB Color Light Set kreiert! https://www.youtube.com/watch?v=X3pVDZ0XTIA&feature=youtu.be

Brick 'R' knowledge shared a video
2 days ago

Heute haben wir ein neues Brick- und Arduino.org-Experiment für euch, kreiert von unserem Praktikanten Mattia. Viel Spaß beim Nachbauen!
https://www.youtube.com/embed/5ILljCTdHRg

TWITTER

Dank an unseren #**Praktikanten** Mattia für dieses tolle #**Brick**- und @ArduinoOrg #**Experiment**: https://t.co/elNdCjQSb2 https://t.co/dlj0zV0vFu

Vielen Dank @code_your_life für die tolle Show in der #**ufaFabrik**. Wir waren begeistert! #**kids** #**coding** with #**bricks** https://t.co/0FFk8z91Yf

Kids @ #**codeyourlife** #**Brick**'R'knowledge https://t.co/Wtm91lcVeu

#**Brick**'R'knowledge an der #**TU** #**Berlin**: Wir haben den #**Studenten** unsere #**Bricks** vorgestellt https://t.co/gvJkX6ZugD https://t.co/ct0A0xa5qq

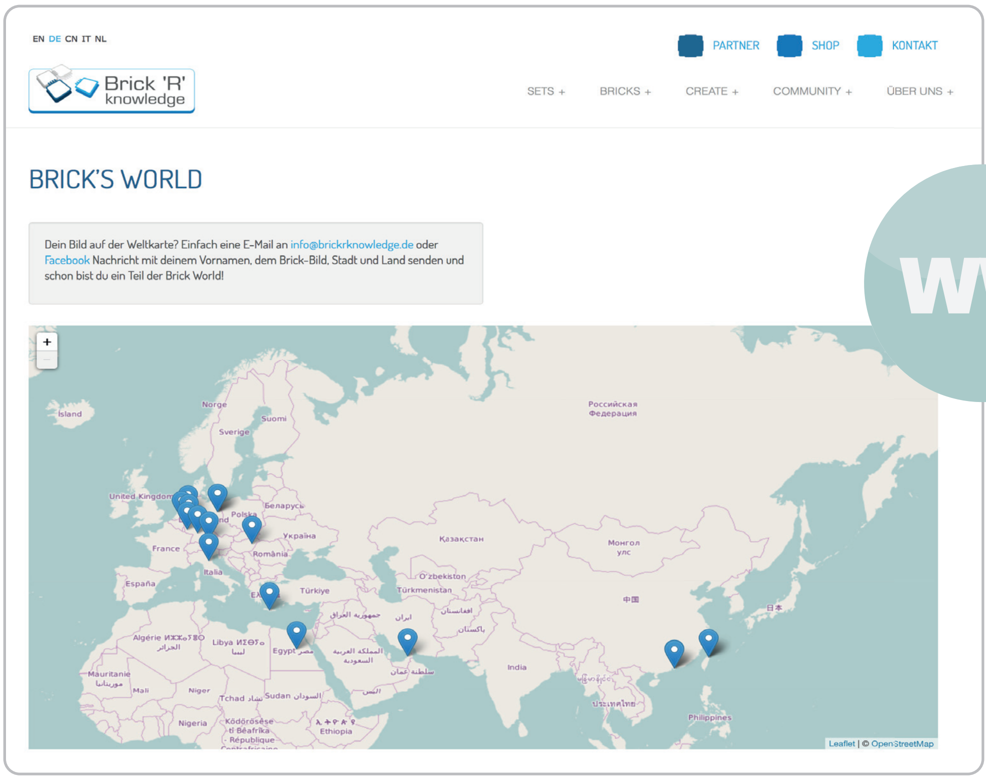
INSTAGRAM

PINTEREST

THANKS TO OUR...
...
WWW.MAKER-STORE.DE
ARDUINO CODING SET
IN CASE YOU MISSED...
THE POWER OF THE ...
CHARGE YOUR MOBILE...
TEST YOUR REACTION...
NEVER BORING WITH...

## Worldwide

At "Community" you can also  nd out where our bricks were already. Do you have a nice picture of bricks in your town? Just send it to us and soon you will  nde it on our website!



EN DE CN IT NL

PARTNER SHOP KONTAKT

Brick 'R' knowledge

SETS + BRICKS + CREATE + COMMUNITY + ÜBER UNS +

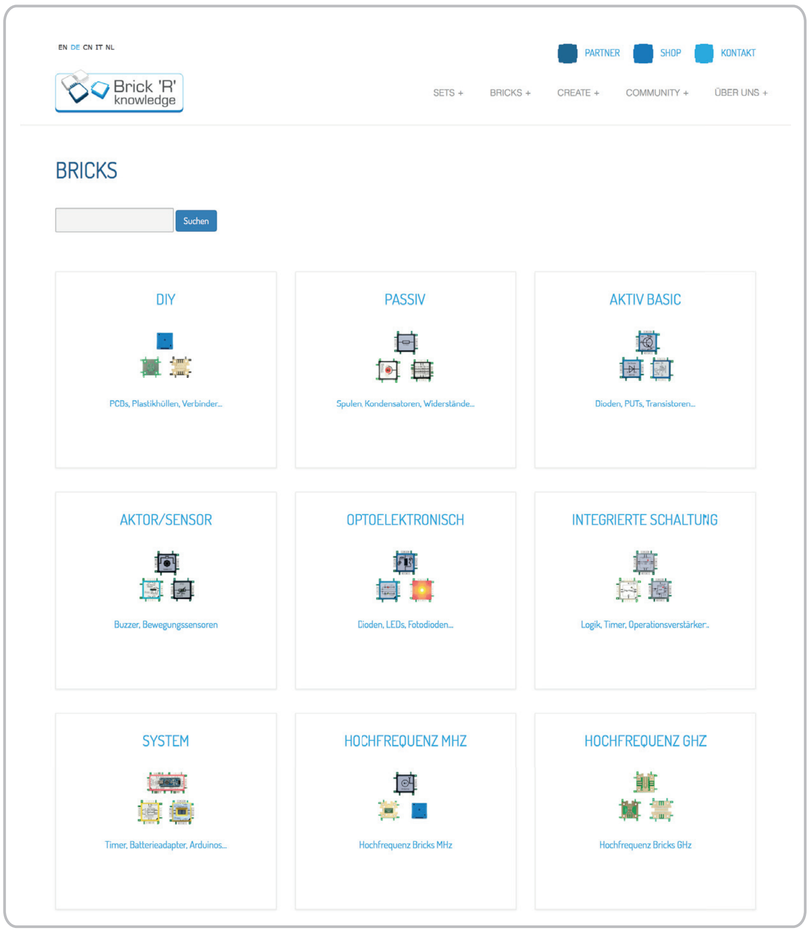
BRICK'S WORLD

Dein Bild auf der Weltkarte? Einfach eine E-Mail an info@brickrknowledge.de oder Facebook Nachricht mit deinem Vornamen, dem Brick-Bild, Stadt und Land senden und schon bist du ein Teil der Brick World!

WW

## Even more bricks!

By clicking on "Bricks" you can nd all of the available bricks with information and ideas for experiments.



EN DE CN IT NL

PARTNER SHOP KONTAKT

Brick 'R' knowledge

SETS + BRICKS + CREATE + COMMUNITY + ÜBER UNS +

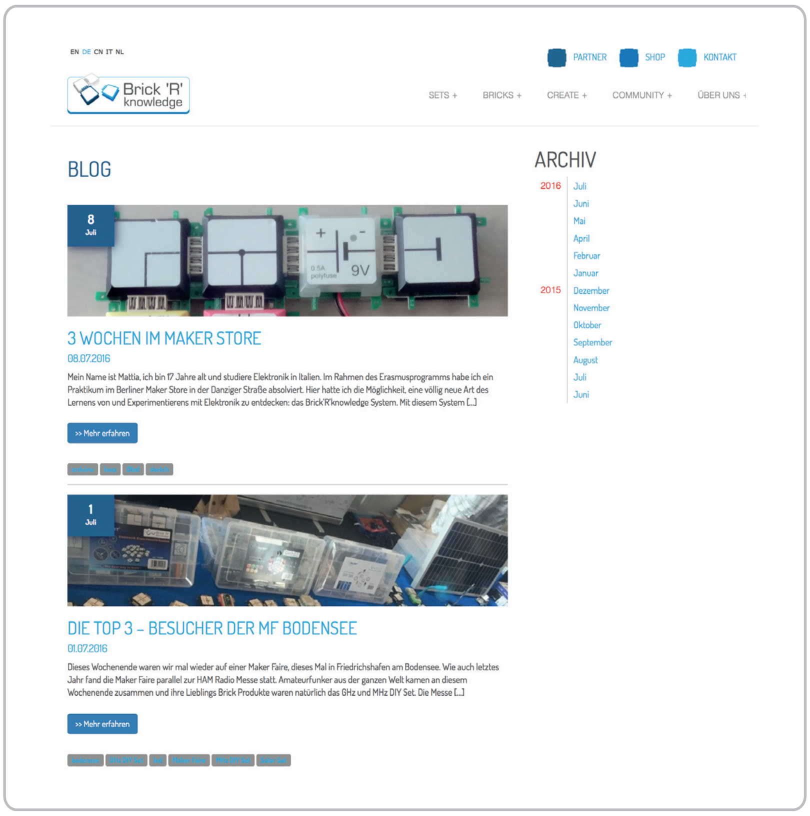
BRICKS

Suchen

| | | |
|---|---|---|
| DIY<br>PCBs, Plastikhüllen, Verbinder... | PASSIV<br>Spulen, Kondensatoren, Widerstände... | AKTIV BASIC<br>Dioden, PUTs, Transistoren... |
| AKTOR/SENSOR<br>Buzzer, Bewegungssensoren | OPTOELEKTRONISCH<br>Dioden, LEDs, Fotodioden... | INTEGRIERTE SCHALTUNG<br>Logik, Timer, Operationsverstärker... |
| SYSTEM<br>Timer, Batterieadapter, Arduinos... | HOCHFREQUENZ MHZ<br>Hochfrequenz Bricks MHz | HOCHFREQUENZ GHZ<br>Hochfrequenz Bricks GHz |

## Brick Blog

Each week we post a new blog post. You can read about our experiences at fairs, new circuits, funny stories and information of the world of electronics.



EN DE CN IT NL

PARTNER SHOP KONTAKT

Brick 'R' knowledge

SETS + BRICKS + CREATE + COMMUNITY + ÜBER UNS +

BLOG

8 Juli

3 WOCHEN IM MAKER STORE

08.07.2016

Mein Name ist Mattia, ich bin 17 Jahre alt und studiere Elektronik in Italien. Im Rahmen des Erasmusprogramms habe ich ein Praktikum im Berliner Maker Store in der Danziger Straße absolviert. Hier hatte ich die Möglichkeit, eine völlig neue Art des Lernens von und Experimentierens mit Elektronik zu entdecken: das Brick'R'knowledge System. Mit diesem System [...]

>> Mehr erfahren

arduino loop Oled sketch

1 Juli

DIE TOP 3 – BESUCHER DER MF BODENSEE

01.07.2016

Dieses Wochenende waren wir mal wieder auf einer Maker Faire, dieses Mal in Friedrichshafen am Bodensee. Wie auch letztes Jahr fand die Maker Faire parallel zur HAM Radio Messe statt. Amateurfunker aus der ganzen Welt kamen an diesem Wochenende zusammen und ihre Lieblings Brick Produkte waren natürlich das GHz und MHz DIY Set. Die Messe [...]

>> Mehr erfahren

bodensee GHz DIY Set led Maker Faire MHz DIY Set Solar Set

ARCHIV

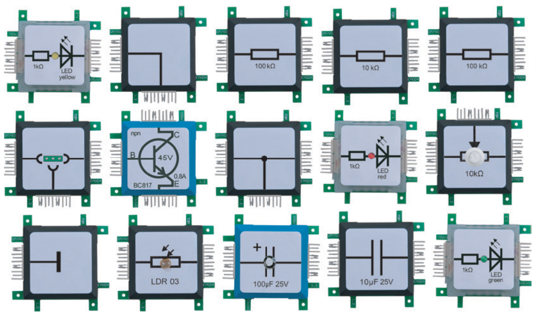
| | |
|---|---|
| 2016 | Juli |
| | Juni |
| | Mai |
| | April |
| | Februar |
| | Januar |
| 2015 | Dezember |
| | November |
| | Oktober |
| | September |
| | August |
| | Juli |
| | Juni |

## 7. Brick Sets im Überblick

### Basic Set

ALL-BRICK-0374

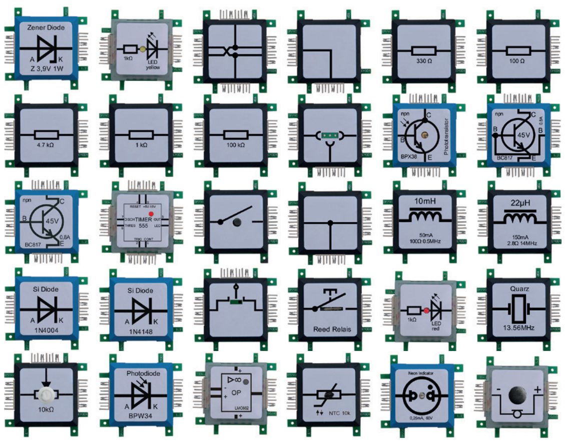
The basic set contains 19 selected bricks to o  er a fast and easy start into the world of Brick `R` knowledge as well as the possibility to create numerous circuits. The basic set is a perfect support for kids gaining their rst experiences with electronic and technical experiments.



### Advanced Set

ALL-BRICK-0223

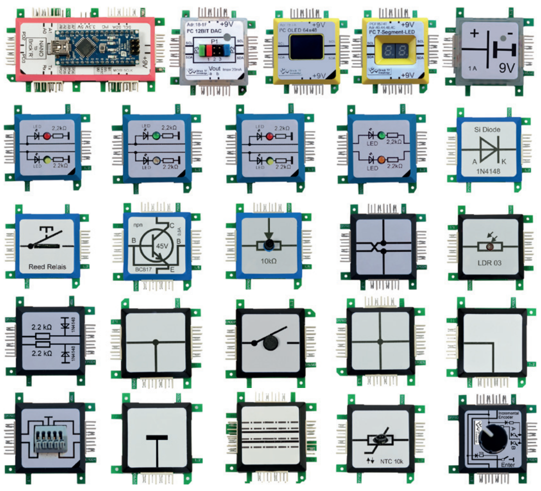
Our Advanced Set contains 111 components that allow you to build more complicated and complex solutions. Thanks to the educational system, knowledge can be gathered, so that not only you but also our next generation can prot from it. You can build individual circuits by plugging di  erent bricks together. Simple as well as complex electronic and technological topics can be experienced in a totally new way. Due to the open-source factor, you can create your own bricks and develop your own solutions.
Brick'R'knowledge isn't all about basic electronic engineering, also RF experiments can be realized, which makes it a unique system worldwide.



## Arduino Coding Set

ALL-BRICK-0414

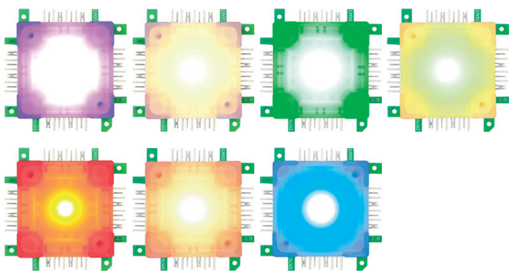
Get in touch with digital electronics and start understanding programming with the Arduino® Nano, which is included in the kit. It is our rst kit with digital components, such as 7-segment displays, OLED display, D/A converter or I2C Bricks, complementary to all analog bricks. To get you started with the popular microcontroller, we support you with various programming examples.



## 7 Color Light Set

ALL-BRICK-0398

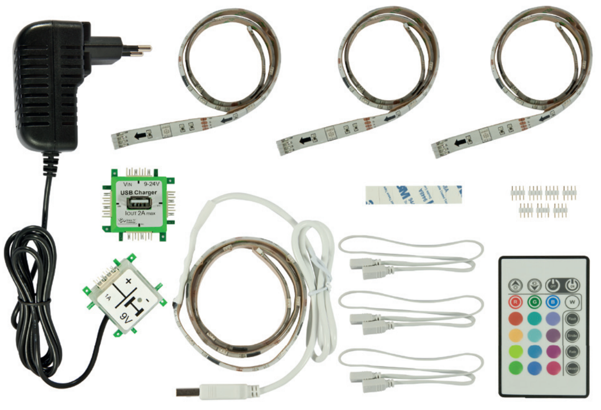
The 7 Color Light Set contains 28 LED bricks in 7 different colors to create stunning light effects in a horizontal and vertical architecture. The red, yellow, blue, orange, violet, green and warm white 1 watt LEDs are perfect for individual lighting characters or as a mobile lighting solution.



## RGB Color Light Set

ALL-BRICK-0619

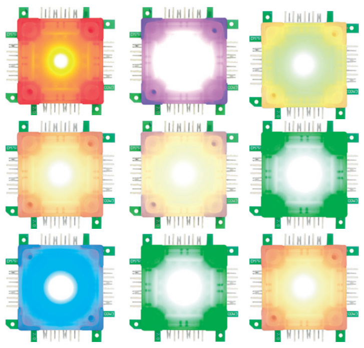
Create your light show! The RGB Color Light Set comes with four exible LED strips containing 36 LEDs in total that can be controlled with the included infrared remote control. You can glue, cut and connect the LED strips however you want. The infrared remote control has 16 different color keys and 4 light programs.



## Programmable LED Set

ALL-BRICK-0483

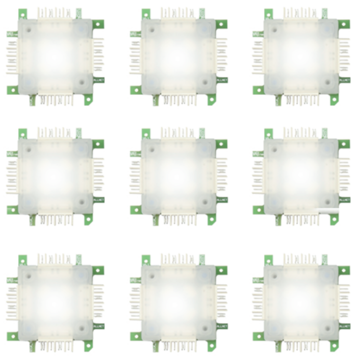
The kit contains 49 programmable and controllable RGB LED bricks, each with two or three connectors and a conjunction-brickfor Arduino management and power supply. Furthermore, the Brick'R'knowledge Programmable LED Set includes an Arduino adapter brick and an Arduino Nano. With this kit you can realize colorful LED animations and other individual ideas. And the best thing about it: by performing di erent projects, you can easily learn the programming of microcontrollers.



## Highpower LED Set

ALL-BRICK-0399

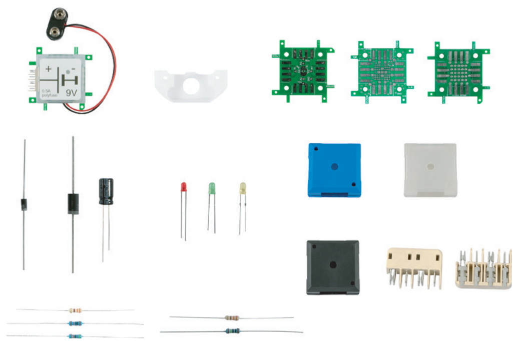
Powered by 1 Watts, each of the 50 High Power LED bricks contained in the kit irradiate the whole surrounding area in bright white. Build individual solutions in every imaginable architecture and invent Brick nightlights, Brick table lamps or any other creative illuminant. The power supply with 12V 8A supports the intensive luminosity to o er a stylish and cozy atmosphere. The High Power LED Set 50 allows you to deal with modern light design and simultaneously learn about electronics.



## DIY Set

ALL-BRICK-0397

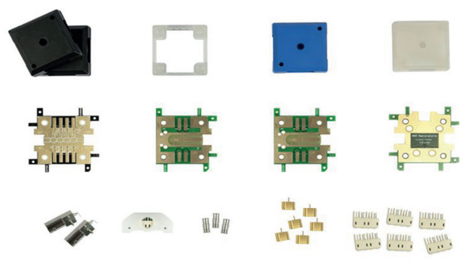
The DIY set goes even a step further. The included components offer a much more detailed insight into the brick architecture and allow even the production of individual bricks. The DIY set o ers an enormous exibility for the maker generation or for people creating individual bricks.



## MHz DIY Set

ALL-BRICK-0457

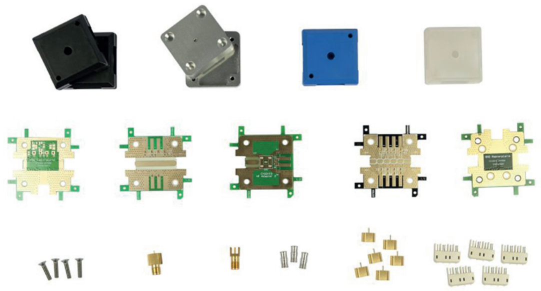
Individual challenging projects within the MHz frequency range can be created with the MHz DIY set. Three different grid and experimentation boards, BNC sockets, P-SMP plugs and suitable connectors make the kit perfect for any high frequency experiment. The kit contains hermaphrodite connectors and a soldering jig for SMD plugs to develop your own bricks or other components for the Brick system.



## GHz DIY Set

ALL-BRICK-0458

Realize advanced and complicated experiments in the high frequency range up to GHz frequencies. In addition, the kit offers four diferent PCBs, P-SMP, SMA sockets, P-SMP connectors and brick-specic hermaphrodite connectors. The GHz DIY Set is perfect for HAM radio operators and fans of measuring.



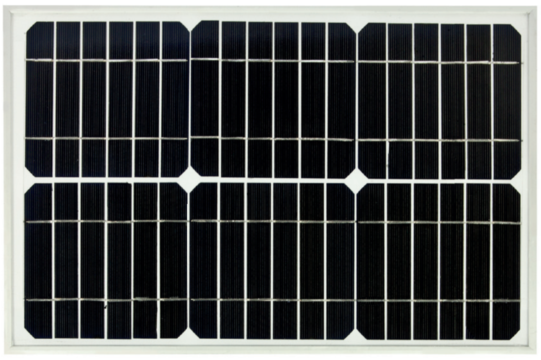
## Solar Set

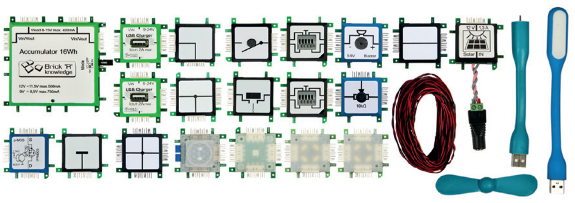
ALL-BRICK-0484

Das Solar Set von Brick'R'knowledge garantiert Experimentierspaß für die ganze Familie und bringt Kindern erneuerbare Energien auf spielerische Art und Weise näher.

- Wie funktioniert eine Solarzelle?
- Wie speichert ein Akku Strom?
- Wie baut man ein Nachtlicht mit Bewegungsmelder?

Auf diese und weitere Fragen gibt das Solar Set Antworten. Mit diesem Set sind Sie und Ihre Kinder offizielle Mitglieder der Maker-Generation.





## Measurement Set One enthält 4 Bricks ALL-BRICK-0637

Das Set ermöglicht es, mit Standardmessgeräten in Brick'R'knowledge Schaltungen Spannung, Stromstärke und andere Messgrößen einfach zu ermitteln. Das Messadapter-Set besteht aus folgenden Bricks: einem Messadapter mit 3 x 2 mm Buchse, einem Messadapter mit 4 mm Closed End GND in schwarz mit zusätzlicher Kabelklemme, einem Messadapter mit 4 mm Endpoint in gelb und einem Messadapter mit 4 mm Inline in rot.

## Measurement Set Two enthält 6 Bricks ALL-BRICK-0638

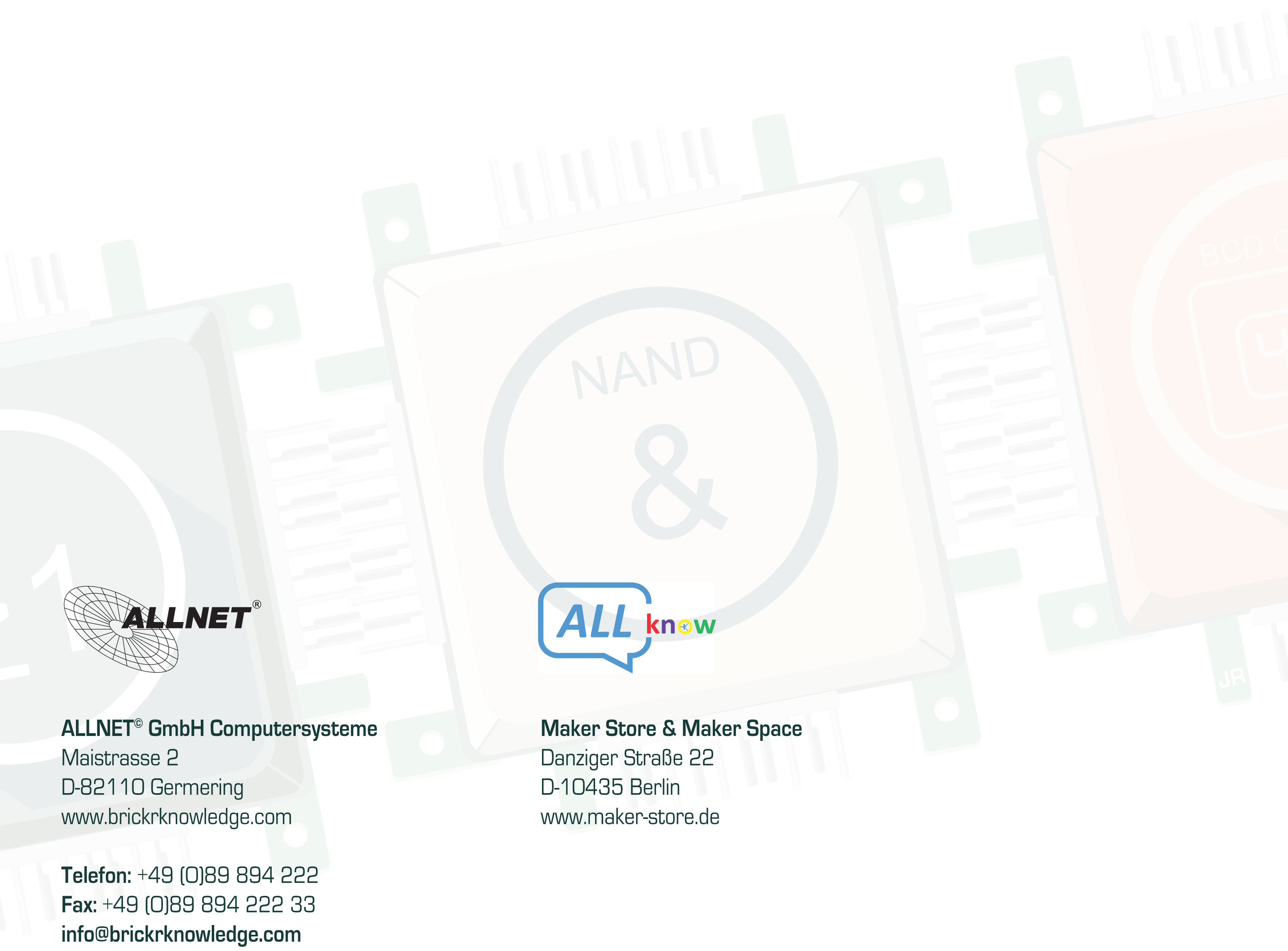
Das Set ermöglicht es, mit Standardmessgeräten in Brick'R'knowledge Schaltungen Spannung, Stromstärke und andere Messgrößen einfach zu ermitteln. Das Messadapter-Set besteht aus folgenden Bricks: zwei Messadapter mit 4 mm Closed End GND in schwarz, zwei Messadapter mit 4 mm Inline in rot und zwei Messadapter mit 4 mm Open End GND in schwarz.

## Internet of Things Set enthält 17 Bricks ALL-BRICK-0646

Mit dem Internet of Things Set ist es nun möglich, die Bricks via Internet zu kontrollieren. Mit dem enthaltenen IoT-Brick werden Sie beispielsweise lernen, Ihre erste Website zu bauen und I/O Pins mit Ihrem Smartphone zu steuern. Außerdem enthält das Set einen Temperatur- und Luftfeuchtigkeitssensor, dessen Werte Sie auf einem Display darstellen können: Der erste Schritt zur eigenen Home Automation!

Sie können auch Daten, wie zum Beispiel den Dollar-Kurs aus dem Internet abfragen und sich anzeigen lassen. Um die 7-Segmentanzeige anzusteuern, wird der sogenannte I$^2$C-Bus genutzt, den Sie auch bald kennenlernen. Das Internet der Dinge wartet darauf, von Ihnen entdeckt zu werden!

Brick 'R' knowledge



ALLNET®

ALL know

**ALLNET© GmbH Computersysteme**
Maistrasse 2
D-82110 Germering
www.brickrknowledge.com

**Telefon:** +49 (0)89 894 222
**Fax:** +49 (0)89 894 222 33
**info@brickrknowledge.com**

**Maker Store & Maker Space**
Danziger Straße 22
D-10435 Berlin
www.maker-store.de