

---

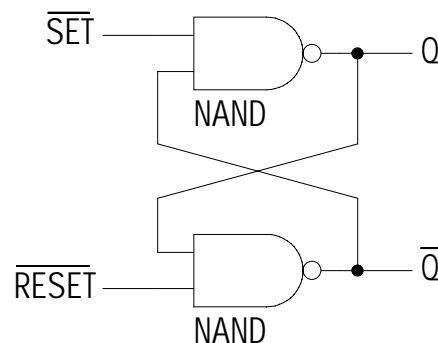
# 2 : BISTABLES

In this Chapter, you will find out about bistables which are the fundamental building blocks of electronic counting circuits.

## Set-reset bistable

A **bistable** circuit, also called a **latch**, or '**flip-flop**', has two stable states. The output of a bistable can be either logic 1, or logic 0, according to signals received at its inputs.

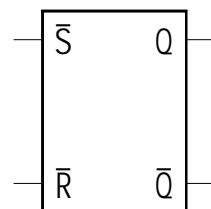
One of the simplest bistable circuits consists of two NAND gates:



As you can see, there are two inputs, the  $\overline{\text{SET}}$  and  $\overline{\text{RESET}}$ . The 'bars' over these symbols indicate that these inputs are '**active LOW**'. In other words, each input must be held HIGH and pulsed LOW when you want something to happen.  $Q$  and  $\overline{Q}$  are the outputs of the circuit and normally have opposite logic states.

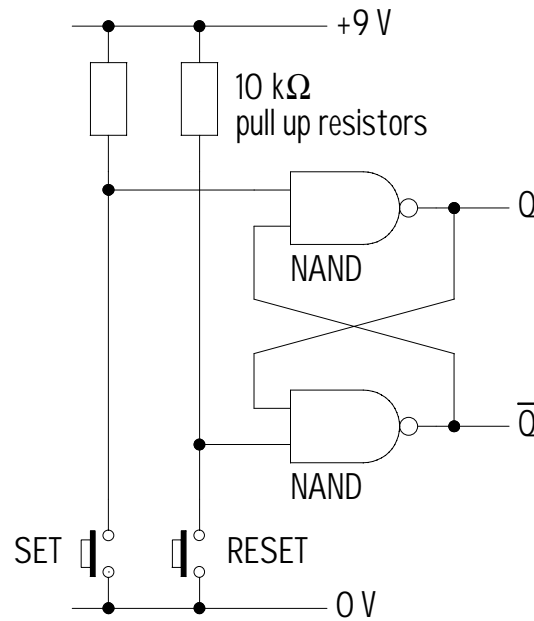
The bistable is SET when  $Q=1$  and  $\overline{Q}=0$ , and RESET when  $Q=0$  and  $\overline{Q}=1$ .

This kind of bistable is called a **set-reset bistable**,  $\overline{S} \overline{R}$  bistable, or sometimes  $\overline{R} \overline{S}$  bistable, latch or flip-flop, and is often represented by a simplified symbol, as follows:

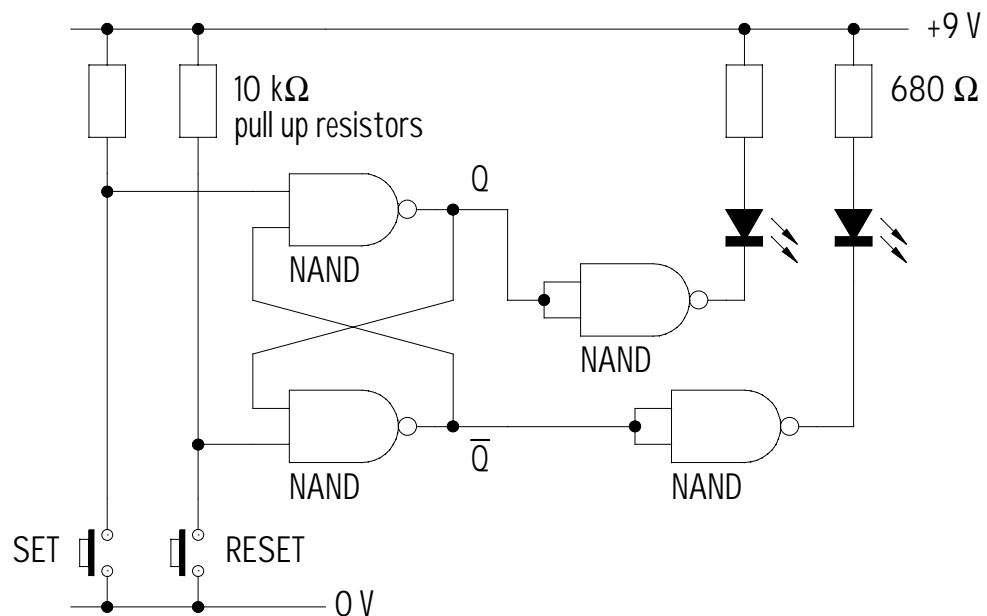


*Symbol for an  $\overline{S} \overline{R}$  bistable*

A practical circuit for a NAND gate bistable needs switches to provide logic signals. Since the  $\overline{\text{SET}}$  and  $\overline{\text{RESET}}$  inputs are to be held HIGH and pulsed LOW, you need switches with pull up resistors, as indicated in the circuit diagram which follows:



It would be useful to monitor the Q and  $\bar{Q}$  outputs with LEDs which illuminate when each output is HIGH:

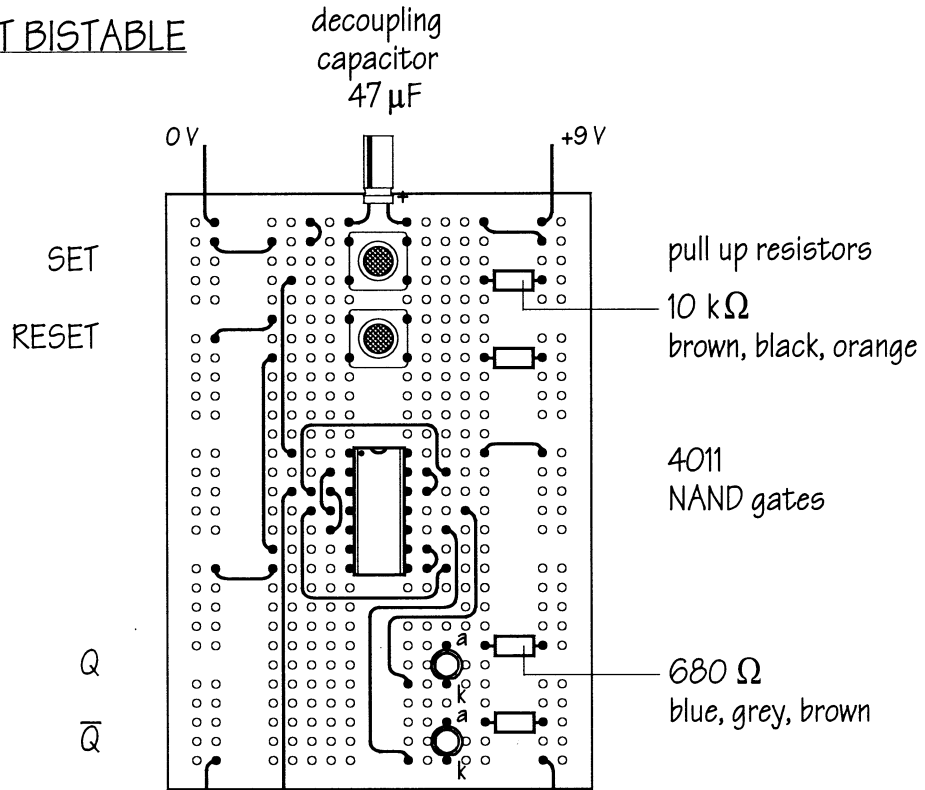


Remember that the outputs of the NAND gates driving the LEDs do not produce valid logic signals.

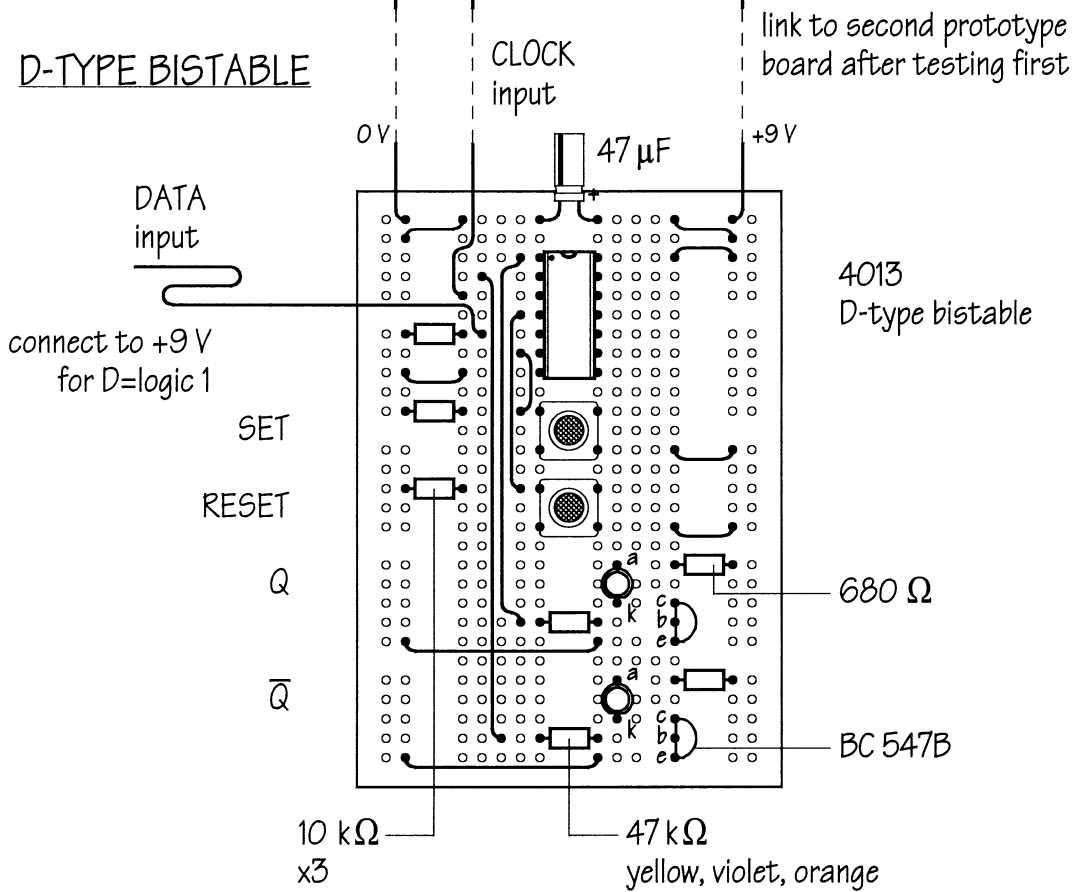
As you can see, the whole circuit requires four NAND gates and can therefore be assembled using a single 4011 CMOS integrated circuit. The prototype board layout for this is shown at the top of DS 2.1. Go ahead and build the circuit, following the layout carefully.

Look at the way in which the push button switches are connected on the prototype board. The placement of the link wires to the inputs of the 4011 NAND gates exploits the way in which the contacts are

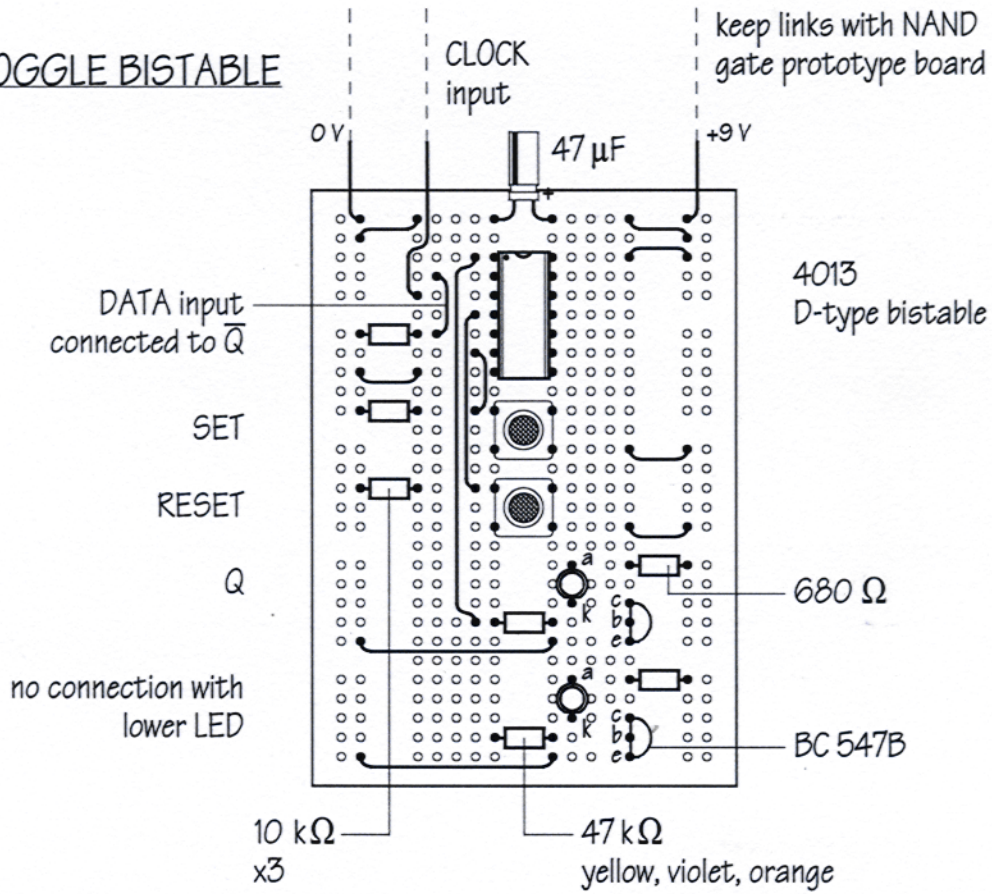
# SET-RESET BISTABLE



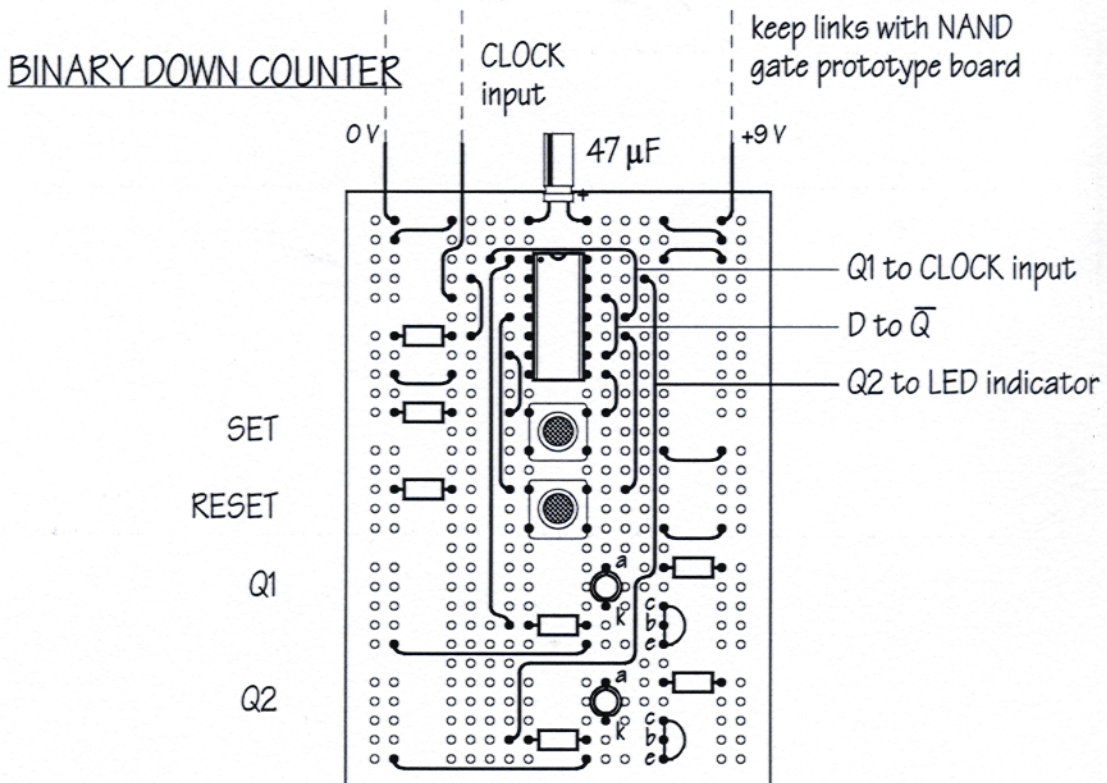
# D-TYPE BISTABLE



## TOGGLE BISTABLE



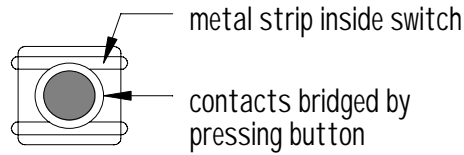
## BINARY DOWN COUNTER



---

arranged inside the switch. As you can see in the diagram below, the top two pins of the switch are internally connected by a metal strip:

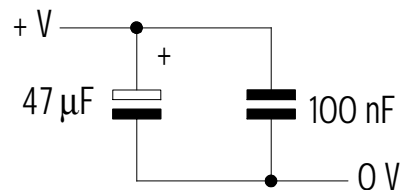
miniature tactile switch



The bottom two pins are internally connected in the same way. As a result, either pin from each pair can be used to make connections to other parts of the circuit. This is a common arrangement for push button switches intended for use on printed circuit boards, PCBs, and helps to simplify the design of PCB layouts for keyboard circuits.

An important new component has been added to the prototype board layout. This is the 47  $\mu\text{F}$  **decoupling capacitor** connected across the power supply from +9 V to 0 V. Why is it there? The answer is that the decoupling capacitor prevents the transfer of rapid or transient signals, or 'glitches', from one part of the circuit to another along the power supply connections. CMOS i.c.'s are relatively vulnerable to this sort of interference and the addition of 47  $\mu\text{F}$ , or 100  $\mu\text{F}$  decoupling capacitors, connected as close to the power supply pins of individual integrated circuits as possible, is recommended.

Sometimes, you will see a decoupling arrangement like this:



This arrangement looks strange because the values of capacitors connected in parallel add together:

$$C_{\text{total}} = C_1 + C_2 \dots$$

Here  $C_{\text{total}} = 47 + 0.1 = 47.1 \mu\text{F}$ . Since the tolerance of a 47  $\mu\text{F}$  capacitor is likely to be as much as 20%, it seems pointless to add such a small capacitor in parallel. In fact, the behaviour of *practical* polarized and non-polarized capacitors is different, with non-polarised capacitors working more effectively for high frequency signals.

Think about decoupling capacitors as part of the 'recipe' for getting CMOS circuits to work and don't forget to put them in.

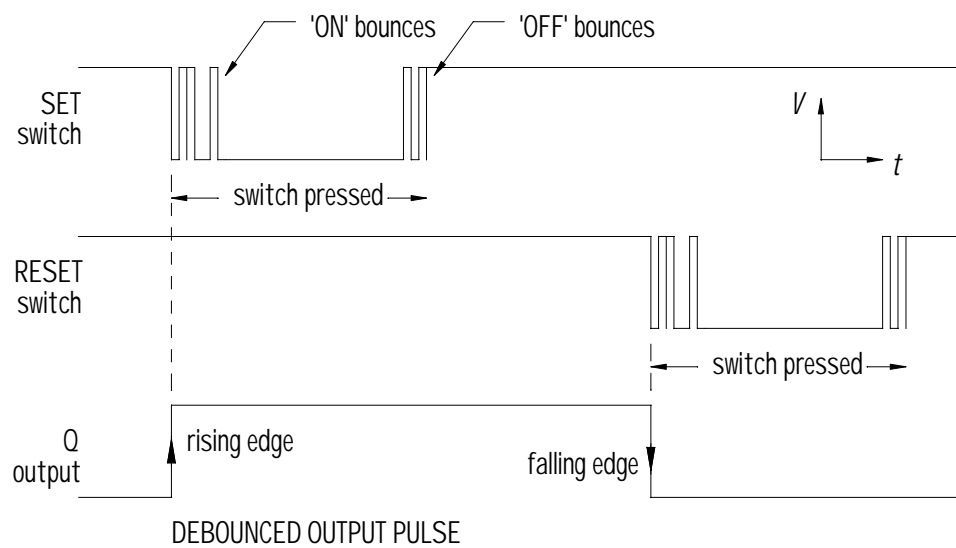
Test your set-reset bistable.

- What happens when you operate the SET push button?
- What happens when you operate the RESET push button?
- What happens when you operate both buttons at once?

In this last condition, *both* Q and  $\bar{Q}$  LEDs should illuminate. This is often described as a 'disallowed state'. The bistable isn't damaged, but it is impossible to predict which output will remain HIGH when the buttons are released.

A set-reset bistable is often used as a 'memory' device, for example in a burglar alarm, which remains ON once it has been triggered.

A different application for a set-reset bistable is in '**debouncing**' a switch. The metal contacts inside a switch are springy and often bounce when the switch state is altered. Contacts may close, not once, but several times. With an oscilloscope, to monitor voltage/time changes, this is what you might see at the inputs and outputs of the bistable:



Although, the signals at the  $\overline{\text{SET}}$  and  $\overline{\text{RESET}}$  inputs change many times, the Q output changes just once from LOW to HIGH, and once from HIGH to LOW. In this way, you can produce a 'debounced' output pulse. Pressing the SET switch gives a rising edge, pressing the RESET switch gives a falling edge.

## D-type bistable

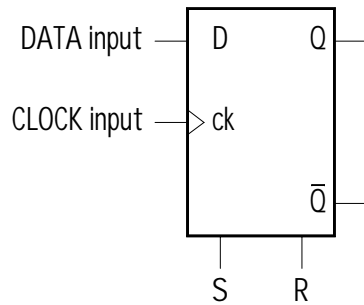
For electronic counting, a more sophisticated type of bistable is needed. The diagram at the top of the next page shows the connections for a **D-type bistable**. As you can see, there are four different inputs:

- The DATA input is held at logic 1 or logic 0 and can be changed from one logic state to the other at any time.
- The small triangle inside the symbol indicates that the CLOCK input is **edge-triggered**. This is an important property. The CLOCK input of a D-type bistable responds only to rising edges,

that is, a rapid change from LOW to HIGH. Slow changes and sustained HIGH or LOW signals have no effect.

- The S and R inputs are the SET and RESET. The absence of 'bars' above these inputs indicates that they are 'active HIGH'. In other words, S and R should be held LOW and pulsed HIGH to SET or RESET the bistable.

Q and  $\bar{Q}$  are the outputs as before:

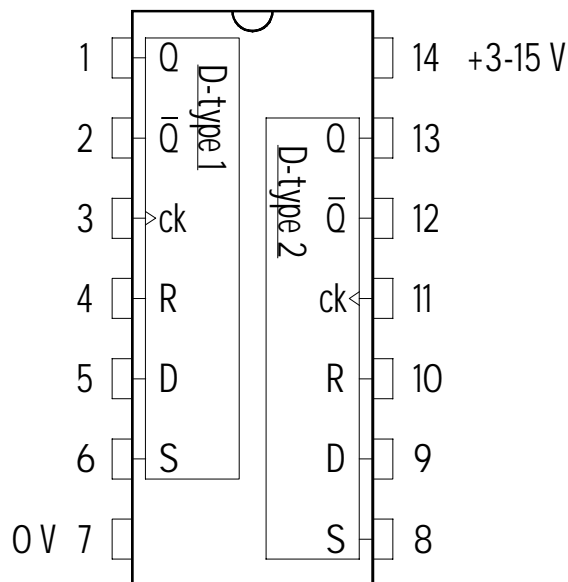


*Symbol for a D-type bistable*

To describe the action of a D-type bistable, you can say that **the logic state at the D-input is transferred to the Q-output on the rising edge of the clock.**

This statement is easier to understand once you have built and tested a prototype circuit. This is shown on the lower half of DS 2.1, and uses a new CMOS integrated circuit, the 4013. In fact, a 4013 contains two separate D-type bistables with pin connections as follows:

### 4013 D-type bistables



For the present, you are going to use the D-type bistable connected to pins 1-6, leaving the other bistable unconnected. (In any final circuit, all unused CMOS inputs *must* be connected either to +9 V or

---

0 V, but you can usually work on prototype board without worrying about the unused parts of the integrated circuit.)

Build the D-type circuit, following DS 2.1 carefully. Connect the power supply and the CLOCK input from the first prototype board as shown, then investigate the action of the D-type:

- Verify the action of the D-type SET and RESET switches.

Whenever these switches are pressed, the output of the D-type will change accordingly. The SET and RESET inputs are '**level-sensitive**', rather than edge-triggered.

- What happens if the SET and RESET are pressed simultaneously?

This is a disallowed state, as with the NAND gate bistable.

- Press the RESET switch to force  $Q=0$ , then connect the DATA input to +9 V, using a 'flying lead' from the prototype board, see DS 2.1.
- Press the RESET switch on the NAND gate circuit (first prototype board) and then press the SET switch on the NAND gate circuit. This produces a rising edge at the CLOCK input of the D-type bistable, provided you have connected the two prototype boards together.

At the *exact moment* that the rising edge occurs, the Q-output of the D-type bistable will go HIGH.

- Disconnect the flying lead from +9 V. Because of the pull down resistor, the DATA input immediately returns to LOW. Does this have any effect on the outputs of the D-type?
- Repeat the RESET/SET sequence with the NAND gate circuit.

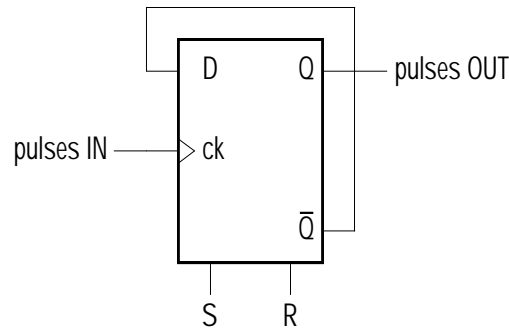
Again, the output of the D-type changes at the *exact moment* that the rising edge arrives. This is what is meant by edge-triggered behaviour.

Continue to investigate the D-type until you are confident that you understand what each of the inputs and outputs will do.

## Start counting . . .

Don't take your carefully assembled prototype board circuits to bits. You will need them both for the practical work described in this section.

Electronic counting depends on **toggle bistables**. This isn't an entirely new variety of bistable you have to learn about but is easily made by connecting the  $\bar{Q}$ -output of a D-type bistable to its DATA input, as shown in the diagram which follows:



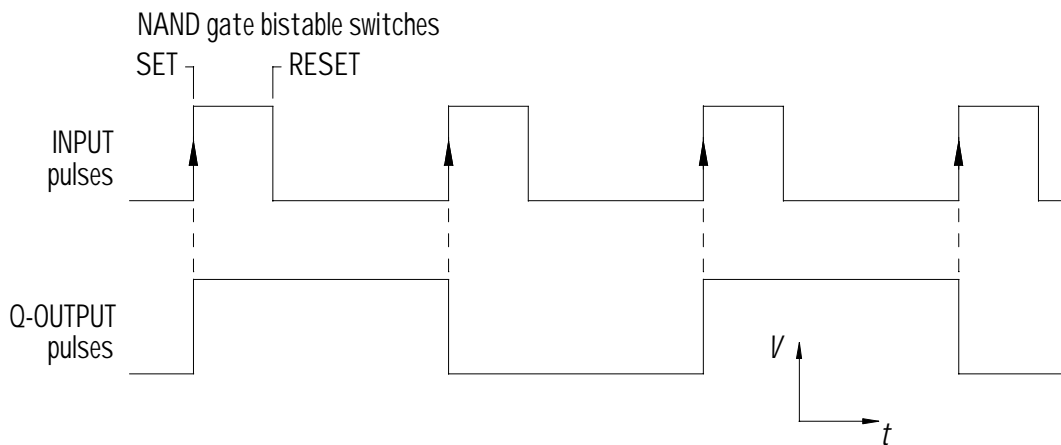
*A toggle bistable*

The special property of a toggle bistable is that its output changes state every time an input pulse arrives.

You can convert your D-type bistable circuit to a toggle bistable by removing one link and inserting another. Do this now, following the layout at the top of DS 2.2. There is a new link from  $\bar{Q}$  to the DATA input and the connection to the second LED has been removed. Keep the connections with the NAND gate bistable prototype circuit.

- RESET the NAND gate bistable and then press SET and RESET in sequence in order to deliver a series of input pulses to the CLOCK input of the toggle bistable.

The output of the toggle bistable changes state for each input pulse received. Strictly, for each rising edge received. With an oscilloscope, the changes observed are like this:

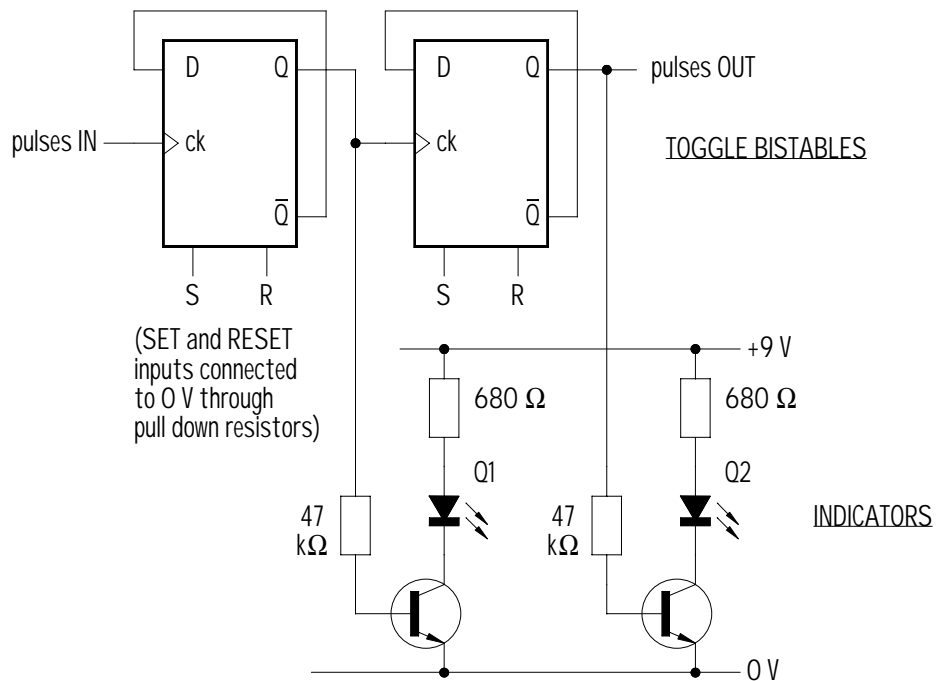


As you can see, the Q-output changes at the beginning of each input pulse. You should be able to see this happen when you press the NAND gate bistable SET button.

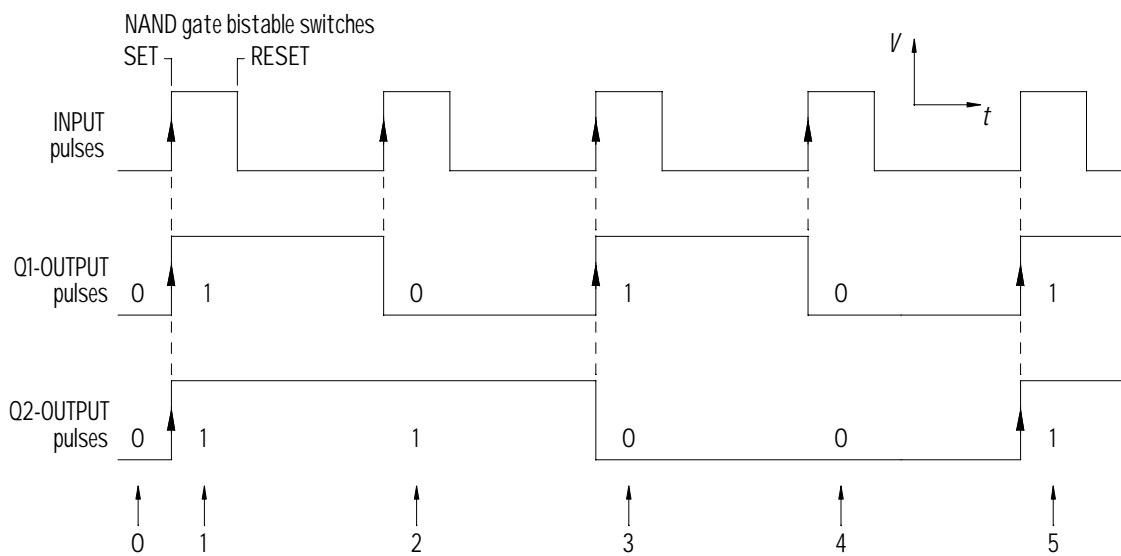
- Count the input pulses in the diagram above.  
Answer: 4
- Count the output pulses in the diagram above.  
Answer: 2

The toggle bistable *divides* the number of input pulses by two. This is really the same as *counting* the input pulses. To make this clear, you

can now add a second toggle bistable to the first, as shown in the layout at the bottom of DS 2.2. Add the links following the layout carefully. Your circuit is now:



The output of the first D-type is a pulse waveform with rising edges and these trigger the second bistable as follows:



You have added a second divide-by-two circuit, with one Q2-output pulse for every *four* input pulses.

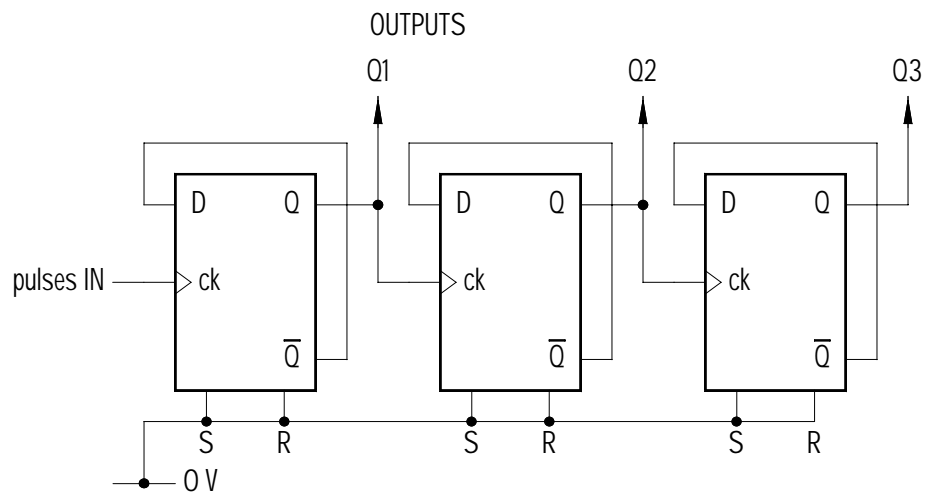
It is helpful to convert these waveforms into truth table format, with each line in the truth table corresponding to one of the numbered arrows at the bottom of the diagram:

<i>pulse number</i>	<i>Q2 output</i>	<i>Q1 output</i>
0	0	0
1	1	1
2	1	0
3	0	1
4	0	0
5	1	1

repeated  
sequence

In the table, the Q1 output is on the right because it is the **least significant bit, LSB**, of a binary number. The counting sequence for the circuit is 11 10 01 00, in descending binary order. You have built a **2-bit binary down counter**. There are 2 D-types and  $2^2=4$  different output states.

How could you make a 3-bit binary down counter? Easy! add another toggle bistable:

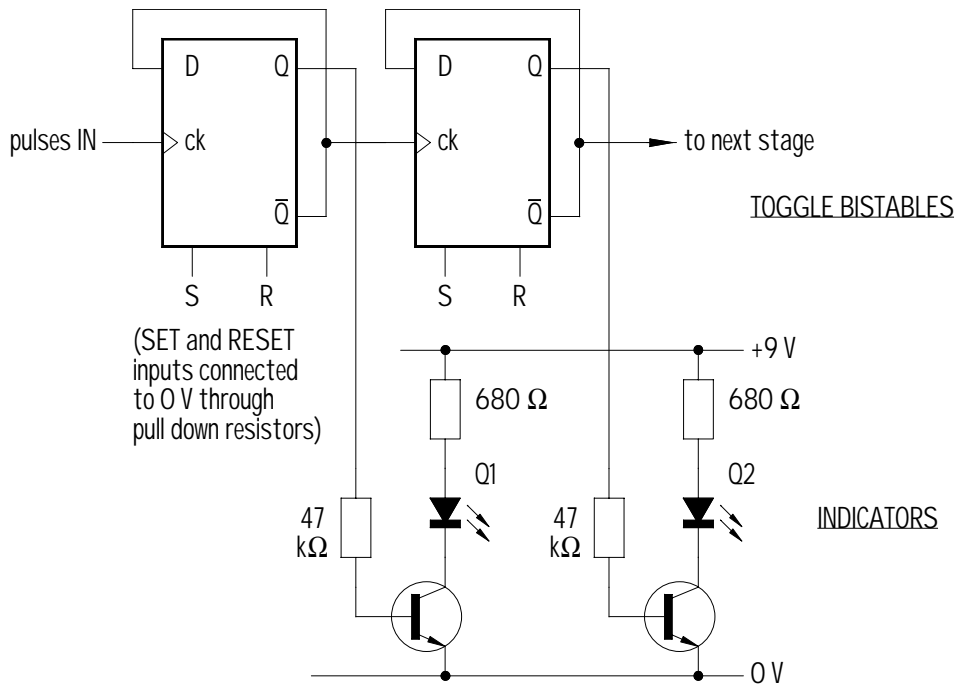


*3-bit binary down counter*

How many output states would this have. The calculation is  $2^3=8$  output states. For a 4-bit counter ( $2^4=16$  output states) you would add a fourth bistable, and so on.

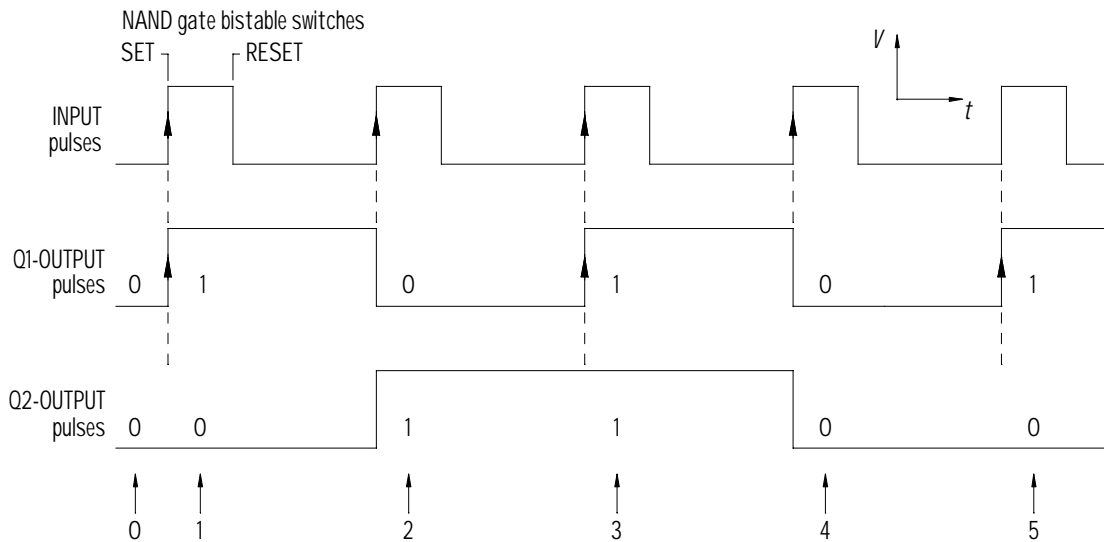
This is all very well, but how can you make an *up* counter, which is likely to be more useful for counting in general and for electronic clock circuits in particular? This is easier than it sounds. All you need to do is to rearrange the connections between the toggle bistables. Instead of connecting the Q-output to the CLOCK input of the next

stage, the  $\bar{Q}$ -output is connected to the CLOCK input of the next stage. The circuit becomes:



- Check back with the layout at the bottom of DS 2.2, and make this modification.

The waveforms at the counter outputs are changed to:



The truth table for the counter outputs is changed as well and the outputs now follow the repeated sequence 00 01 10 11, that is, binary numbers in ascending order:

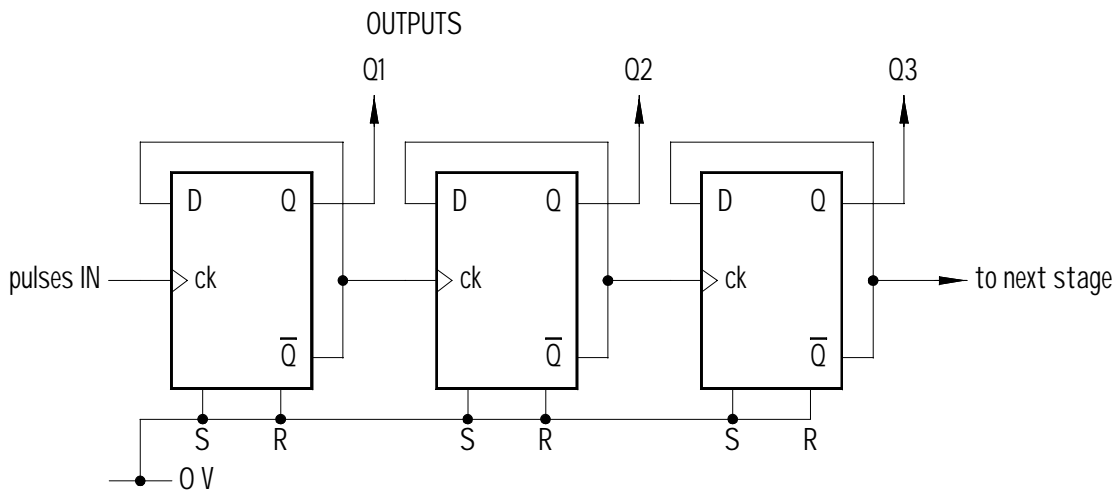
<i>pulse number</i>	<i>Q2 output</i>	<i>Q1 output</i>
0	0	0
1	0	1
2	1	0
3	1	1
4	0	0
5	0	1

repeated sequence

Notice again, that the Q1-output appears in the table at the right hand side because it is the LSB of the binary number.

- Confirm the behaviour of your counter by sending pulses to the CLOCK input from the NAND gate bistable, as before.

Here is the circuit for a 3-bit binary up counter:



*3-bit binary up counter*

To make a 4-bit binary up counter, you add a fourth toggle bistable connected in the same way, and so on.

To summarise:

DOWN counter: Q to CLOCK input of next stage

UP counter:  $\bar{Q}$  to CLOCK input of next stage

---

## What's next?

You will find out more about counters in due course. The counters described in this Chapter count only in powers of two,  $2^2=4$ ,  $2^3=8$ ,  $2^4=16$  and so on. How can you make a *decimal* counter, which counts in 10's? How could you link counters together to count up to 60, to count seconds or minutes, or to 12 or 24 for hours? Similarly, how can you process the outputs of counters to give a user-friendly display?

All of these questions have answers and, as you work through the next few Chapters, you will find out what they are.