Chapter 4 Digital signal processing
4.1 Logic gates

Learning objectives:

→ Recognise the functions of various logic gates.

→ Determine the truth table for a combination of logic gates.

→ Use Boolean algebra to design logic gate combinations.

Logic gates and truth tables

You make decisions all the time. Each decision involves making a choice of some sort. A logical decision is one with an outcome that depends on the input conditions. For example, suppose you are about to go out and you are thinking about whether or not you should lock up your home when you go. Table 1 shows the possible input conditions and the outcome or output for each input condition. Each input and output state is entered as 0 for No and 1 for Yes in the table. Such a table is called a **truth table**.

Table 1 *A truth table*

Inputs		Output
Am I the only person in at the moment?	**Will I be out for more than a few minutes?**	**Should I lock up?**
0	0	0
0	1	0
1	0	0
1	1	1

1 = Yes 0 = No

A logic gate is a digital circuit designed to make a decision. Any logic gate has one or more input terminals and one output terminal. The voltage at the output terminal (i.e., the output state) depends on the voltage at the input terminals (i.e., the input states) according to the type of gate. The output state for each combination of input states can be displayed using a truth table. We do not need to consider how a logic gate works, only what it does. Figure 1 shows the circuit symbol and the truth table of some common logic gates.

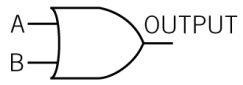
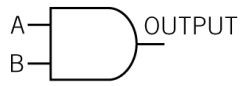
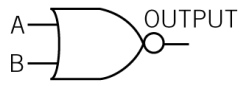
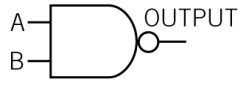
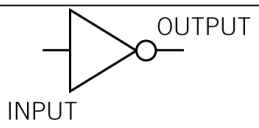
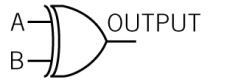
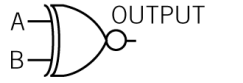
Gate	Symbol	Function (High voltage = 1, Low voltage = 0)	Truth Table INPUTS A	B	OUTPUT
OR		OUTPUT = 1 if A OR B = 1	0 0 1 1	0 1 0 1	0 1 1 1
AND		OUTPUT = 1 if A AND B = 1	0 0 1 1	0 1 0 1	0 0 0 1
NOR		OUTPUT = 0 if A OR B = 1	0 0 1 1	0 1 0 1	1 0 0 0
NAND		OUTPUT = 0 if A AND B = 1	0 0 1 1	0 1 0 1	1 1 1 0
NOT		OUTPUT = 1 if INPUT = 0 OUTPUT = 0 if INPUT = 1	0 1		1 0
EXOR (exclusive OR)		OUTPUT = 1 if A unequal to B	0 1 0 1	0 0 1 1	0 1 1 0
EXNOR (exclusive NOR)		OUTPUT = 1 if A equal to B	0 1 0 1	0 0 1 1	1 0 0 1

Figure 1 *Logic gates*

Boolean algebra

The output of a logic gate or a combination of logic gates can be expressed in terms of the inputs using Boolean algebra, where a plus sign represents OR, a dot represents AND, and a bar above an input or output symbol represents NOT. For example, the output of:

- a NOT gate with input A is represented by $\overline{A}$
- a gate with two inputs, A and B, is represented by:

  $A + B$ for an OR gate

  $A.B$ for an AND gate

  $\overline{A + B}$ for a NOR gate

  $\overline{A.B}$ for a NAND gate

  $A \oplus B$ for an EOR gate

Using the above representations, the logic circuit of a logic system can be deduced from its truth table. For example, consider the logic system for a door alarm in a two-door car

that sounds only when one or both of the two doors are opened and the engine is running. Figure 2 shows the truth table and a suitable logic circuit for this purpose. The truth table can be summed up by the condition:

the output = 1 if input A OR input B = 1 AND input C = 1

This can be written using Boolean algebra as:

output = (A + B).C

(a) truth table

A	B	C	Q
0	0	0	0
1	0	0	0
0	1	0	0
1	1	0	0
0	0	1	0
1	0	1	1
0	1	1	1
1	1	1	1

(b) using an OR gate and an AND gate

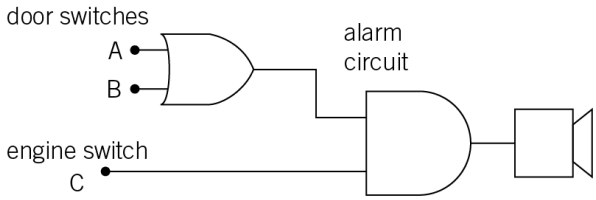


Figure 2 *A logic gate combination*

NOR gates only or NAND gates only are often used to construct other logic gates. For example:

- A NOT gate can be constructed by joining the two inputs of a NOR gate or a NAND gate together.
- A 2-input OR gate can be constructed by connecting the output of a 2-input NOR gate to the input of a NOT gate (which can be constructed from a NOR gate as explained above).
- A 2-input AND gate can be constructed by connecting the output of a 2-input NAND gate to the joined together inputs of another 2-input NAND gate.

Summary questions

1 Draw and complete the truth table for each of the logic gate combinations in Figure 3.

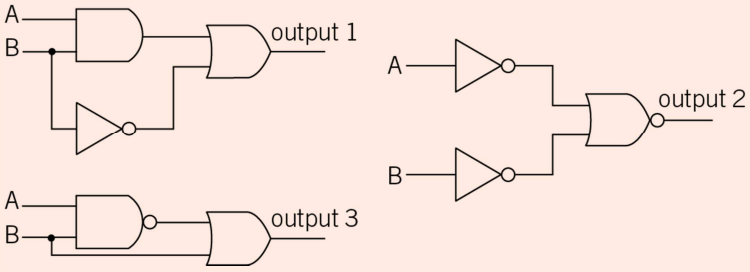


Figure 3

2 A freezer has a proximity sensor P and a temperature sensor T. The temperature in the freezer must not exceed a maximum temperature θ_{MAX} set by a dial and the freezer door is closed.

The output of T is logic 1 if the temperature is above θ_{MAX}, and logic 0 if the temperature is at or below θ_{MAX}.

The output of P is logic 1 if the freezer lid is closed, and logic 0 if the lid is open.

The two sensors are to be connected to a logic circuit C that will operate an alarm that sounds when the output of the logic circuit C = 1.

a Construct a truth table for a logic circuit to which T and P should be connected.

b Draw a suitable logic circuit for the system.

3 **a** Draw a truth table for the Boolean expression $C = \overline{\overline{A} + \overline{B}}$

b **i** Hence show that $\overline{\overline{A} + \overline{B}} = A.B$

ii Use the equation in **i** to show how an OR gate can be constructed from NAND gates.

c **i** Construct a truth table to show that $A \oplus B = \overline{A}.B + A.\overline{B}$

ii Use the expression above to construct an EOR gate using three 2-input NAND gates and two NAND gates used as NOT gates.

4 A petrol pump delivers either unleaded petrol or diesel fuel through separate hoses from separate tanks when the unleaded nozzle (A) or the diesel nozzle (B) is lifted from its holder and the nozzle trigger is pulled. A '1' from a nozzle trigger opens a valve and starts a pump at the appropriate fuel tank. No fuel is delivered if both nozzles are activated at the same time. Design a logic system for the pump.

4.2 Sequential logic

Learning objectives:

→ Explain how an *n*-bit counter works.

→ Describe a BCD counter and a Johnson counter.

→ Explain what is meant by the mark-to-space ratio of an astable multivibrator.

Binary counters

Sequential logic circuits store data and use it to give an output at a later time. A binary counter is an example of a sequential logic circuit because it stores and displays the number of pulses it receives. **Binary counters** are designed to count pulses and display the number counted as a binary number. A binary counter consists of a series of identical circuits called **T-type flip-flops** that each have two outputs labelled Q and $\overline{Q}$, a clock input Ck or $\overline{Ck}$, and a reset input R, as shown in Figure 1. Note the symbol for a $\overline{Ck}$ input. For a Ck input, the circle is omitted.

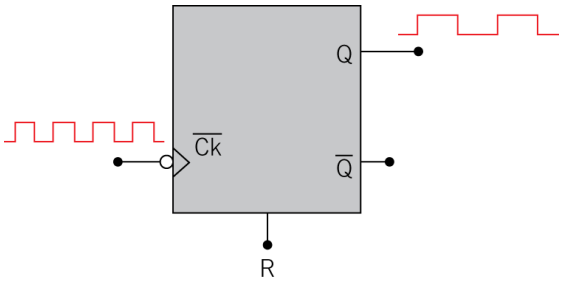


Figure 1 *A T-type flip-flop*

Note that:

- $\overline{Q}$ is the complement of Q, which means that when Q = 0, $\overline{Q}$ = 1, and when Q = 1, $\overline{Q}$ = 0.
- If a pulse is applied at input Ck, the rising edge (0→1) causes Q and $\overline{Q}$ to change. If a pulse is applied at input $\overline{Ck}$, the falling edge (1→0) causes Q and $\overline{Q}$ to change.
- When a pulse is applied to R, the Q output is set to 0 and the $\overline{Q}$ output is set to 1.

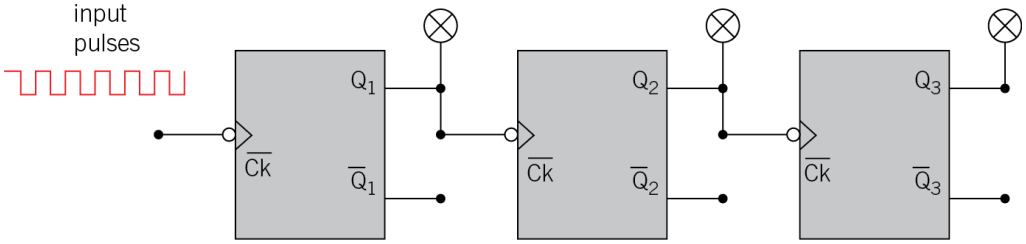
Consider the changes that take place in a flip-flop when two successive pulses are applied at $\overline{Ck}$ after the flip-flop has been reset:

- The falling edge of the first pulse causes output Q to change to 1 and output $\overline{Q}$ to change to 0.
- The falling edge of the second pulse causes output Q to change back to 0 and output $\overline{Q}$ to change back to 1.

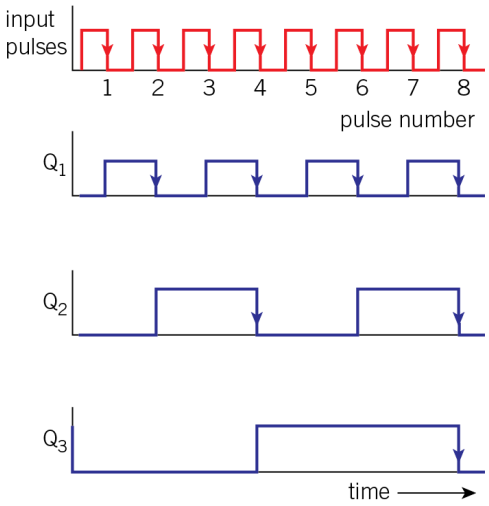
Therefore, a single pulse is produced at output Q for every two pulses applied to the clock input. In continuous operation, to produce *n* pulses at output Q, 2^n pulses need to be received at $\overline{Ck}$.

In a **binary counter**, the output Q of each flip-flop is connected to the clock input of the next flip-flop, as shown in Figure 2. A pulse-producer circuit is connected to the clock input of the first flip-flop. The reset terminals of the flip-flops are connected together so that all the flip-flops have a common reset terminal.

(a) circuit (reset terminals not shown)



(b) input and output states



pulse number	Q_3	Q_2	Q_1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

Figure 2 *A 3-bit binary counter*

In Figure 2, the $\overline{Ck}$ input of each flip-flop is used. As a result of using the $\overline{Ck}$ inputs instead of the Ck inputs, the counter **counts up** instead of down. When pulses are supplied to the counter after it has been reset, the falling edge (1→0) of each pulse applied to a flip-flop causes its outputs to change.

- The falling edge of the flip-flop's first pulse sets Q_1 at 1 (the Q output of the first flip-flop) without changing the other outputs Q_2 and Q_3. Binary counter output $Q_3\,Q_2\,Q_1 = 001$, which is the binary number for 1.
- The second pulse sets Q_1 at 0, so the falling edge at Q_1 (1→0) changes the output at Q_2 from 0 to 1 without changing Q_3. Binary counter output $Q_3\,Q_2\,Q_1 = 010$, which is the binary number for 2.
- The third pulse makes $Q_1 = 1$ without changing the other outputs Q_2 and Q_3. Binary counter output $Q_3\,Q_2\,Q_1 = 011$, which is the binary number for 3.
- The fourth pulse makes $Q_1 = 0$, so the falling edge at Q_1 (1→0) changes Q_2 from 1 to 0, which changes Q_3 from 0 to 1. The binary counter output is therefore changed to $Q_3\,Q_2\,Q_1 = 100$, which is the binary number for 4.

Table 1 shows these changes and subsequent changes for eight pulses. The falling edge of the eighth pulse causes all the Q outputs to change from 1 to 0, so subsequent pulses cause the counting sequence to repeat after every eight pulses.

With three falling-edge flip-flops, the counter counts up from 0 to 7 repeatedly. A further falling-edge flip-flop would enable it to count from 0 to 15 (because decimal 15 = 1111).

Table 1

Pulse number	Q_3	Q_2	Q_1
0	0	0	0
1	0	0	1
2	0	1	0
3	0	1	1
4	1	0	0
5	1	0	1
6	1	1	0
7	1	1	1
8	0	0	0

In general, a binary counter with *n* flip-flops will count from 0 to $2^n - 1$ before it repeats the counting sequence. Each of the 2^n output states represents a different *n*-bit binary number. The **modulus** of a counter is the number of output states it goes through in each counting sequence before the sequence repeats. A 3-bit binary counter is thus a modulo-8 counter.

If the Ck inputs had been used in Figure 2 instead of the $\overline{\text{Ck}}$ inputs, the counter would count down instead of up. This is because the rising edge of each pulse applied to a flip-flop causes the outputs to change.

- The rising edge of the first pulse applied to the first flip-flop of the counter causes Q_1 to change from 0 to 1, and this change causes Q_2 and Q_3 to change from 0 to 1. Therefore the binary counter output $Q_3\,Q_2\,Q_1$ is 111 (= decimal 7).
- The rising edge of the second pulse causes Q_1 to change from 1 to 0, but there would be no change for Q_2 and Q_3. Therefore the binary counter output $Q_3\,Q_2\,Q_1$ changes to 110 (= decimal 6).
- The rising edge of the third pulse causes Q_1 to change from 0 to 1 and therefore changes from 1 to 0, but there would be no change for Q_3. Therefore the binary counter output $Q_3\,Q_2\,Q_1$ would be 101 (= decimal 5).

Subsequent pulses cause the binary counter output to count down from

$$101 \rightarrow 100 \rightarrow 011 \rightarrow 010 \rightarrow 001 \rightarrow 000 \text{ then back to } 111,$$

so the countdown cycle repeats.

A falling-edge counter counts up; a rising-edge counter counts down

BCD counters

BCD counters are 4-bit binary counters that count up from 0 and automatically reset to 0 when the binary count reaches 1010, which is the decimal number 10. This happens when $Q_4\,Q_3\,Q_2\,Q_1 = 1010$. Thus a BCD counter counts from 0 to 9 repeatedly. It is therefore a modulo-10 counter because the output counts through 10 states before it resets. The flip-flops in a binary counter have a common reset terminal R that resets all the Q outputs to 0 when a pulse is applied to R. In a BCD counter as shown in Figure 3, the reset terminal is activated by the output of an 2-input AND gate whose inputs are connected to Q_4 and Q_2.

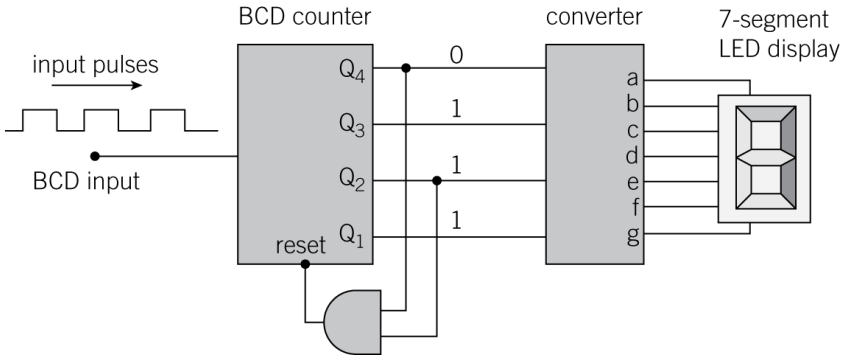


Figure 3 *A BCD counter in use*

The four outputs of a BCD counter may be connected via a converter to a seven-segment LED display. The converter has four inputs and seven outputs and is designed to display the BCD outputs as a decimal number on the LED display.

Notes

1. The output from the AND gate can be connected to the output of a second BCD counter. Each count on this counter display represents 10 pulses received. The two displays would therefore repeatedly count from 0 to 99.
2. The counter in Figure 3 can be modified to count up to any desired number. This is achieved by connecting the appropriate Q outputs to a 2- or 3-input AND gate whose output gate is connected to the reset pin. For example, connecting Q_1, Q_2, and Q_3 to a 3-input AND gate would give a modulo-7 counter (i.e., counts from 0 to 6, then resets).

Johnson counters

Johnson counters are counters in which the Q bit of each flip-flop is shifted to the next flip-flop by the falling edge of a clock pulse. The flip-flops are different to T-type flips and are called D-type flip-flops because they have a data input D as well as a clock input. Each clock pulse applied to a D-type flip-flop stores the data bit at the Q output. In a Johnson counter, the Q output of each flip-flop is connected to the D input of the next flip-flop, and the $\overline{Q}_4$ output of the last flip-flop in the series is connected to the D input of the first one. Figure 4 shows the arrangement.

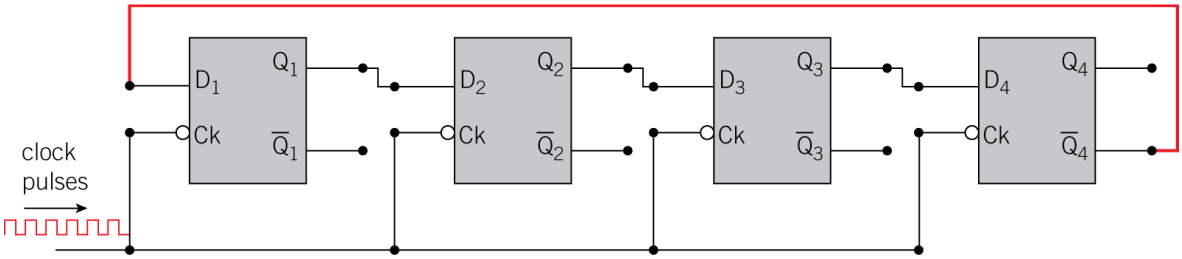


Figure 4 *A Johnson counter (reset terminals not shown)*

Before the clock pulses are applied, the Q outputs are reset to 0 (and the $\overline{Q}$ outputs are reset to 1) by applying a '1' to the common reset terminal. When the clock pulses are applied, the sequence of changes shown in Table 2 takes place in two stages.

Stage 1:

- The first clock pulse changes Q_1 from 0 to 1 because $\overline{Q}_4 = 1$ and D_1 is connected to $\overline{Q}_4$.
- The second clock pulse keeps Q_1 at 1 because $\overline{Q}_4$ is still 1, and shifts the '1' at Q_1 onto Q_2.
- The third clock pulse keeps Q_1 at 1 because $\overline{Q}_4$ is still 1, and shifts the '1's at Q_1 and Q_2 onto Q_2 and Q_3.
- The fourth clock pulse keeps Q_1 at 1 because $\overline{Q}_4$ is still 1, and shifts the '1's at Q_1, Q_2, and Q_3 onto Q_2, Q_3, and Q_4. So $\overline{Q}_4$ changes to 0.

Stage 2:

- The fifth clock pulse changes Q_1 from 1 to 0 because $\overline{Q}_4 = 0$ and D_1 is connected to $\overline{Q}_4$.
- The sixth clock pulse keeps Q_1 at 0 because $\overline{Q}_4$ is still 0, and shifts the '0' at Q_1 onto Q_2.
- The seventh clock pulse keeps Q_1 at 0 because $\overline{Q}_4$ is still 0, and shifts the '0's at Q_1 and Q_2 onto Q_2 and Q_3.
- The eighth clock pulse keeps Q_1 at 0 because $\overline{Q}_4$ is still 0, and shifts the '0's at Q_1, Q_2, and Q_3 onto Q_2, Q_3, and Q_4. So $\overline{Q}_4$ changes to 1, and stages 1 and 2 are repeated.

Table 2 *A Johnson counter sequence*

Input pulse number	Q_1	Q_2	Q_3	Q_4
0	0	0	0	0
1	**1**	0	0	0
2	**1**	**1**	0	0
3	**1**	**1**	**1**	0
4	**1**	**1**	**1**	**1**
5	0	**1**	**1**	**1**
6	0	0	**1**	**1**
7	0	0	0	**1**
8	0	0	0	0
9	**1**	0	0	0
10	**1**	**1**	0	0
11	**1**	**1**	**1**	0
12	**1**	**1**	**1**	**1**
13	0	**1**	**1**	**1**
14	0	0	**1**	**1**
15	0	0	0	**1**
16	0	0	0	0
17	**1**	0	0	0
18	**1**	**1**	0	0

The outputs Q_1, Q_2, Q_3, and Q_4 go through a full cycle of changes every eight pulses. Each output is continuously at '1' for half of each cycle at different times during the cycle. For example, if clock pulses are supplied at 400 Hz, each cycle of eight pulses would take 0.02 s, and each output would be at '1' for 0.01 s and '0' for 0.01 s. Therefore the outputs would produce pulses at a frequency of 50 Hz. However, the pulses from each output would be one clock pulse behind the output from the preceding output, and therefore adjacent outputs would be one-eighth of a cycle out of phase.

A moving LED display is a simple application of this Johnson counter. Each output state Q_1, Q_2, Q_3, and Q_4 supplied to a suitable combination of logic gates can be used to produce a '1', which can light an LED. Therefore by cycling through the eight output states repeatedly, eight LEDs in a row can be switched on repeatedly one at a time.

The astable oscillator

The clock pulses supplied to counters and to other electronic circuits such as shift registers ensure that bytes are moved through digital systems at the correct times in a logical sequence. The clock pulses are provided by an **astable** oscillator. This is a digital circuit that repeatedly switches its output voltage between two voltage levels, as shown in Figure 5.

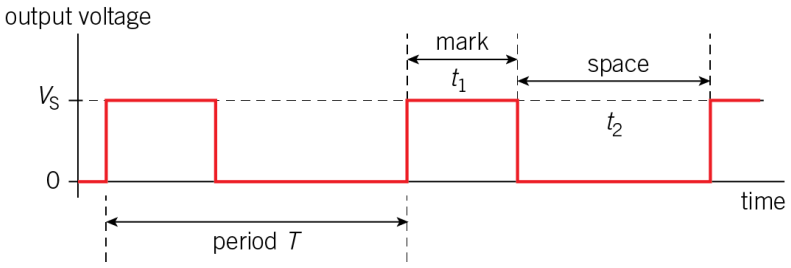


Figure 5 *The astable multivibrator output voltage*

1 The **period**, T, of the output waveform is the time taken for one complete cycle. In Figure 5, this is shown as the time interval from the rising edge of the first pulse to the rising edge of the next pulse. However, the time period T can be from any point on the waveform to the next corresponding point.

Note that the frequency of the oscillator $= \frac{1}{T}$.

2 The **mark**, t_1, of the waveform is the time interval in each cycle when the output is high (i.e., at $+V_S$). The mark is also called the **pulse width** of the waveform. The **space**, t_2, of the waveform is the time interval in each cycle when the output is low (i.e., at zero volts).

Hence $T = t_1 + t_2$.

The **mark-to-space ratio** of the waveform $= \frac{t_1}{t_2}$.

- A large mark-to-space ratio means that the output voltage is high for most of each cycle.
- A small mark-to-space ratio means that the output is low for most of each cycle.

The **duty cycle** is the proportion or percentage of a complete cycle when the output is high. In terms of the mark t_1 and space t_2, therefore, the duty cycle as a percentage is equal to $\frac{t_1}{t_1 + t_2} \times 100\% = \frac{t_1 / t_2}{\frac{t_1}{t_2} + 1} \times 100\%$. Hence the duty cycle is equal to the ratio $\left(\frac{\text{MSR}}{\text{MSR} + 1}\right) \times 100\%$, where MSR is the mark-to-space ratio. The duty cycle is also equal to $\frac{t_1}{T} \times 100\%$.

Examples include:

- A mark-to-space ratio of 0.1 means that the duty cycle is 9.1% because the output voltage is high for $\frac{0.1}{1.1}(=9.1\%)$ of each cycle.
- A mark-to-space ratio of 1.0 means that the duty cycle is 50% because the output voltage is high for $\frac{1}{2}(=50\%)$ of each cycle.
- A mark-to-space ratio of 10 means that the duty cycle is 91% because the output voltage is high for $\frac{10}{11}(=91\%)$ of each cycle.

An astable oscillator circuit includes a capacitor that charges and discharges through a network of resistors. The frequency of its output waveform, called its **running frequency**, can be changed by altering the capacitance and resistance values in the circuit. For example, the 555 timer is a single chip that can be used as an astable oscillator by connecting it to a network including a capacitor and two resistors, as shown in Figure 6. The period and mark-to-space ratio are determined by the values of the capacitance of the capacitor and the resistance of each resistor. The larger these values are, the longer the period of the waveform is and the lower its frequency is. The resistance values determine the mark-to-space ratio as described below.

Link

The time constant was looked at in Topic 23.3, Charging and discharging a capacitor through a fixed resistor, in Year 2 of the *AQA Physics* student book.

More about the 555 timer and flip-flops

The 555 timer

You do not need to know about a particular circuit or a specific device (e.g., 555 chip). But the information here about the 555 chip will give you a greater appreciation of how an astable oscillator is designed to provide a particular running frequency.

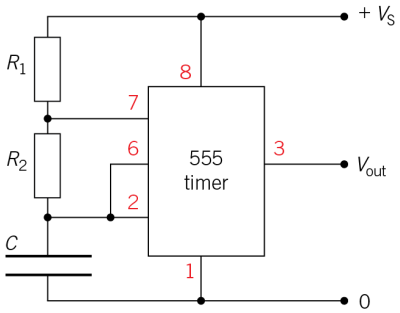


Figure 6 *An astable oscillator*

The capacitance and resistance values affect the frequency because the oscillator circuit contains flip-flops that trigger each other after a short delay in which the change of output voltage charges or discharges the capacitor through the resistor network. The time constant of a resistor–capacitor circuit is equal to $RC \ln 2$. Thus the time taken to charge and discharge a capacitor in the external network of a 555 timer is governed by the values of resistance and capacitance of the resistors and the capacitor in the network.

The mark, t_1, of the waveform $= (R_1 + R_2)C \ln 2$

The space, t_2, of the waveform $= R_2C \ln 2$

The D-type flip-flop

The notes here and for the T-type flip-flop will give you a deeper understanding of counting circuits.

A D-type flip-flop consists of a combination of logic gates which has two outputs Q and $\overline{Q}$, a data input D, a reset input R, and a clock pulse input $\overline{Ck}$ or Ck, as shown in Figure 7. When a pulse is applied to the clock input, the data bit at D is stored at Q. A '1' applied to R resets Q to 0. Note that $\overline{Q}$ is the complement of Q – in other words, when Q = 1, $\overline{Q} = 0$, and when Q = 0, $\overline{Q} = 1$.

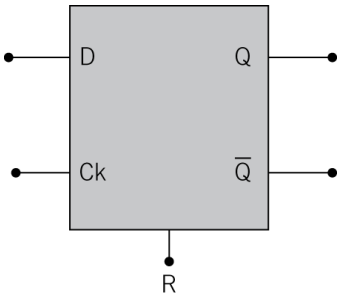


Figure 7 *A D-type flip-flop*

The T-type flip-flop

A T-type flip-flop is a modified D-type flip-flop in which the $\overline{Q}$ output is connected back to the D input, as shown in Figure 8. Output Q changes from 0 to 1 or 1 to 0 when a pulse (i.e., 0→1→0) is applied to the clock input. Each pulse is said to toggle or change the state of the output at Q. For this reason, the circuit is called a T-type flip-flop.

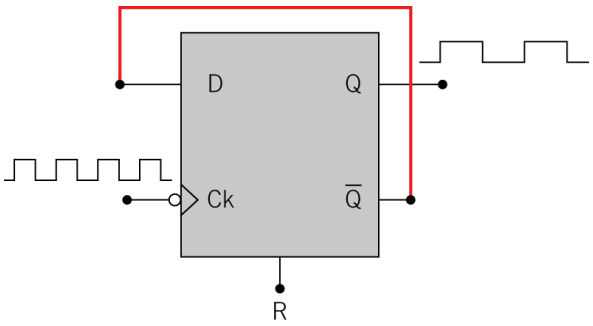


Figure 8 *Inside a T-type flip-flop*

Suppose Q = 0 when a pulse is applied to the clock input Ck.

- Because the bit at $\overline{Q}$ =1, then Q changes from 0 to 1 and $\overline{Q}$ from 1 to 0 as a result of the pulse being applied.
- Because the bit at $\overline{Q}$ is now 0, a second pulse applied at Ck causes Q to change from 1 to 0 and $\overline{Q}$ from 0 to 1.

Therefore the two clock pulses cause Q to change from 0 to 1 and back to 0. In other words, for every two pulses at Ck, a single pulse is produced at Q.

A T-type flip-flop that changes the output state when the input changes from 0 to 1 is called a rising-edge flip-flop. T-type flip-flops that change their output state when the input changes from 1 to 0 are called falling-edge flip-flops. In circuit diagrams, a falling-edge flip-flop is indicated by an open circle at its input, as shown in Figure 8.

QUESTION: Describe what would happen in Figure 8 if Q instead of $\overline{Q}$ was connected to D then clock pulses applied after the circuit was reset.

Summary questions

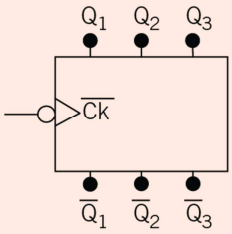
1 **a** Describe what is meant by a 4-bit binary up counter.

b Explain how a 4-bit binary counter is made to count down.

2 **a** Describe what is meant by a BCD counter and how it works.

b Explain how a binary counter can be adapted so that it counts from 0 to 6 repeatedly.

3 Figure 9 shows a block diagram of a 3-bit Johnson counter.



Q_1	Q_2	Q_3
0	0	0

Figure 9

a Copy and complete the table in Figure 9 to show how its output Q_1 Q_2 Q_3 changes from 000 to 111 when it is supplied continuously with pulses at its clock input.

b State and explain how the outputs change from 111 when it is supplied with two further pulses.

4 **a** Explain what is meant by the mark-to-space ratio of an astable waveform.

b An astable oscillator has a frequency of 25 kHz and its mark-to-space ratio is 0.6. Calculate:

i the period

ii the mark time

iii the duty cycle of the oscillations.