

Counters and timers

1 Counters

1.1 Counter functions

1.2 Asynchronous counters

1.3 Synchronous counters

1.4 Bidirectional counters

1.5 Other counter inputs and outputs

1.6 Shift registers

1.7 Feedback shift registers

2 Timers

2.1 Asynchronous timing circuits

2.2 Synchronous timing circuits

2.3 Synchronous edge detector

2.4 Pulse width modulator

Counters and timers

Counters are used to count pulses. The counter state is a set of values of n output signals, usually in a binary code. The counter changes state with each input pulse and gradually goes through a cycle of M states. **Timers** are used to generate time functions. In response to the input pulse, it generates an output pulse of a precisely defined length. Very often the timers are based on counters.

1 Counters

1.1 Counter functions

Counters are very simple sequential circuits used to count pulses. The counter changes state with each input pulse and gradually goes through a cycle of M different states. After M impulses, the cycle is repeated, but the number of repetitions is not maintained. Fig. 1 shows a simple graph describing the transitions between counter states. The states are marked with numbers. Each state corresponds to one combination of output signal values. This combination of values can be understood as a binary number, or its decimal equivalent. Then it is completely natural to assign these numbers to the states of the counter so that with each input pulse the number increases - the counter **counts up**. The transitions graph of such a counter is on the left in Figure 1, below it is the symbol of the counter with the abbreviation CTR (Counter) - its input is marked *UP*. The other obvious option is to sort the states in descending order - counter **counts down** - Fig. 1 middle, input labeled *DN* (Down). The state assignment, however, may be different, for example there are counters counting in **Gray code** (code with change in one variable).

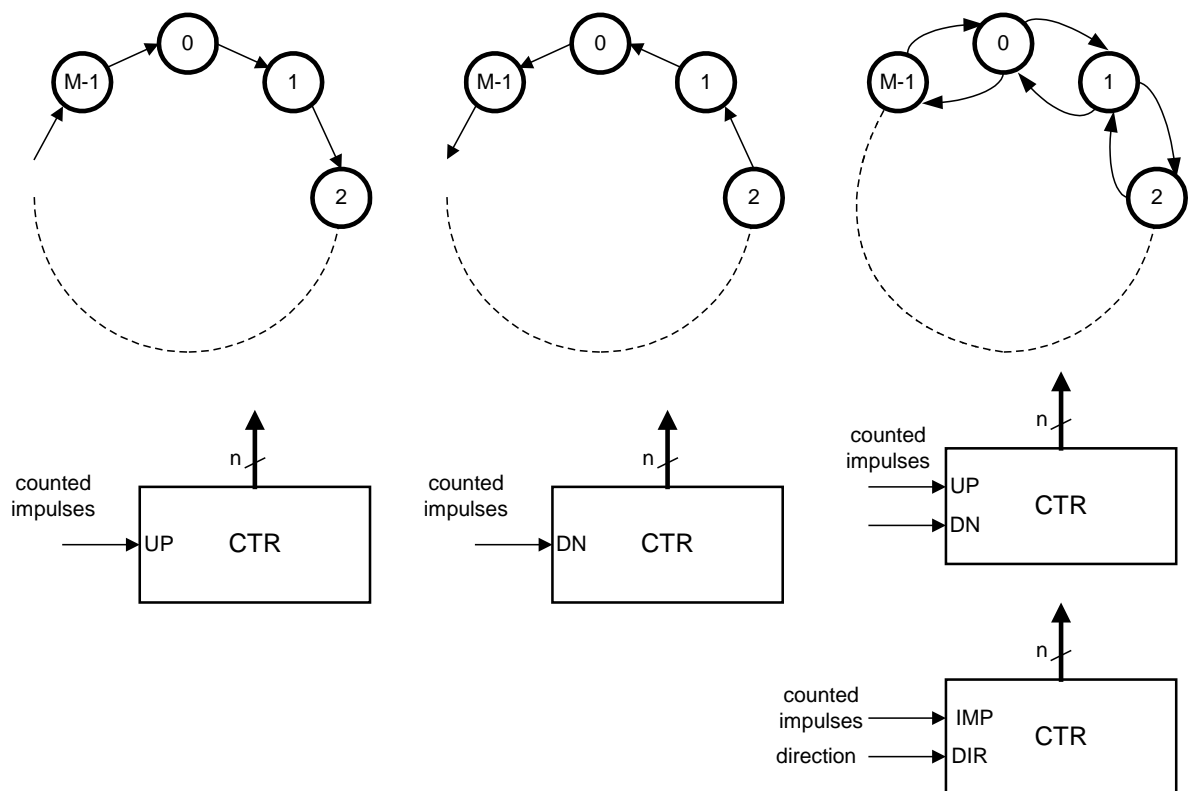


Fig. 1: Graphs of transitions and symbolic symbols for various kinds of the counter

In Fig. 1 in the right is a **bidirectional** counter (UP-DOWN). There are two variants - with special input pulses for counting up and counting down, and with one input IMP for pulses and an input DIR for controlling the counting direction. The counter also has a name **reversible**.

It is common that the outputs of counters are brought out straight from the internal triggers - this option ensures that at the outputs do not exist false pulses (see chapter Synchronous sequential circuits, Fig. 5 and corresponding text). This also means that the internal code is identical to the output code given by the counter type.

A counter that counts in a binary code and with a cycle of length $M = 2^k$ is called **binary**. The counter with a cycle $M = 10$ is called **decimal**, it counts in a BCD (Binary Coded Decimal) code. For counters with cycles of other lengths, the counter name is used “**modulo M**”. There are also counters counting in another code - for example in **Gray's** code.

The sequence of states of a binary counter counting up is shown in the example of a three-bit counter. Three signals Q_2 , Q_1 , Q_0 are required to encode 8 states. Arranged in this order (i.e. Q_2 = MSB, most significant bit) will express a binary number of three bits. Figure 2 shows the state transition induced by the input pulses.

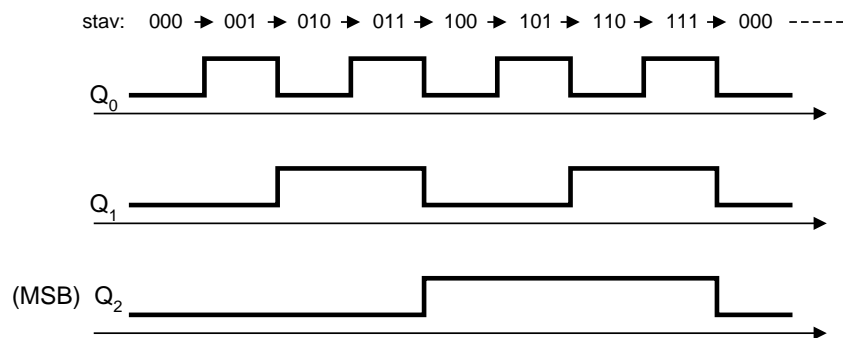


Fig. 2: Waveform of binary counter signals

Evidently the pulse frequency at Q_0 is half the frequency of the input pulses, at Q_1 it is a quarter of it, at Q_2 it is its eighth, etc. The counter can therefore be used as a **frequency divider**.

1.2 Asynchronous counters

The counter can be implemented very simply by combining the asynchronous **T** triggers (latches). The following Fig. 3 applies to the counter from the previous example.

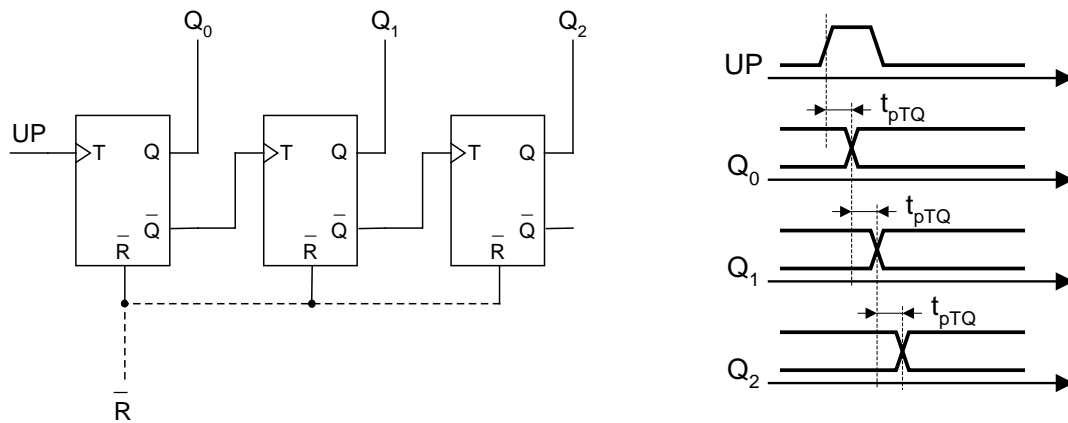


Fig. 3: Three-bit asynchronous counter and delay of individual signals

For the required function, it is necessary for the first trigger to toggle with each *UP* pulse, for the second trigger to toggle when the first trigger toggles from 1 to 0, etc. If the triggers toggle on the leading edge at T input (as indicated in the figure), the required signal waveform can be achieved by connecting the input T with \bar{Q} of the previous trigger. If they toggle on the falling edge, it would be necessary to connect T with Q instead of \bar{Q} . It is clear from the time diagram that the toggling takes place gradually along the triggers from left to right like a **wave**, always with a delay t_{pTQ} . The longest path happens during the transition from the state 11 ... 11 to 00 ... 00, when the wave passes through all n triggers - hence the name **asynchronous counter**. The stabilization time (of all output signals) is long, its maximum is $n \cdot t_{pTQ}$. This is a disadvantage of the asynchronous counter in applications that require a short stabilization time. However, this is not a defect in the application as a **frequency divider** - only the first trigger must have the highest toggling frequency, the second one only works at half that frequency, etc.

By simply changing the connections between the triggers, i.e. from Q to T instead of from \bar{Q} to T, a **down** counting can be achieved.

1.3 Synchronous counters

For synchronous counters, all triggers are toggled at the same time. The general scheme is shown in Fig. 4. Input (i.e. counted pulses) is introduced into the joined CLK inputs of the triggers. Input signals for flip-flops are generated in control (combination) circuits based on the state of the triggers. This gives the possibility to use any types of triggers (i.e. not only D according to the picture, but also T, JK), but always **edge** controlled. The choice of a suitable type of a trigger does not affect the function of the whole counter, but it does affect its connection and **complexity**.

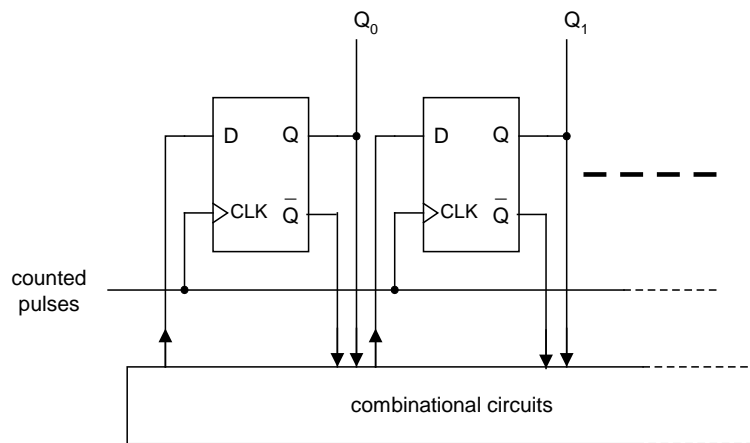


Fig. 4: Simplified connection of a synchronous counter

Due to the simultaneous action of input pulses on all triggers, the **stabilization time** of the new state of the synchronous counter is given regardless of the number of triggers by the delay time t_{pCQ} , i.e. from the *CLK* edge to the change of the state at the trigger's output. This time is **almost** the same for all triggers in the counter - small deviations are due to production tolerances. If the input pulses meet the conditions for the slope of the edges and the length of the pulses, **no false pulses** are generated at the outputs.

However, if the counter is a signal source for the next combinational circuit, deviations in the delay times of the individual triggers may cause a false output in that combinational circuit. This is a typical case of signal **concurrency**, which will be further discussed in the chapter on sequential circuits.

Counters are very common in digital systems. In particular, they are binary counters. These are available as building blocks of medium integration in many variants. However, there are exceptions with less common codes or unusual performance. We will notice some.

The solution of a **three-bit binary** synchronous counter counting up will be shown first. The **transitions table** (Fig. 5) lists all the current states of the counter and the following states ("old" state and "new" state). The counter status is given by a combination of signal values Q_2 , Q_1 , Q_0 ($Q_2 = \text{MSB}$). On the right, the **excitation functions** of individual triggers, necessary to initiate the given transitions, are determined. Triggers *T* and *D* are considered.

state:		excitation function					
old	new	function					
$Q_2 Q_1 Q_0$	$\rightarrow Q_2 Q_1 Q_0$	T_2	T_1	T_0	D_2	D_1	D_0
0 0 0	\rightarrow 0 0 1	0	0	1	0	0	1
0 0 1	\rightarrow 0 1 0	0	1	1	0	1	0
0 1 0	\rightarrow 0 1 1	0	0	1	0	1	1
0 1 1	\rightarrow 1 0 0	1	1	1	1	0	0
1 0 0	\rightarrow 1 0 1	0	0	1	1	0	1
1 0 1	\rightarrow 1 1 0	0	1	1	1	1	0
1 1 0	\rightarrow 1 1 1	0	0	1	1	1	1
1 1 1	\rightarrow 0 0 0	1	1	1	0	0	0

Fig. 5: Table of transitions and excitation functions of a three-bit binary counter for counting up

The counter with T-type triggers must have a value of 1 at input T_i in every state where the output Q_i is to change with a CLK pulse. Therefore, for T_2 :

$$T_2 = \bar{Q}_2 \cdot Q_1 \cdot Q_0 + Q_2 \cdot Q_1 \cdot \bar{Q}_0 = Q_1 \cdot Q_0$$

as the output Q_2 must change in the states 011 and 111.

From the table we find that T_1 should have the value 1 whenever $Q_0 = 1$, i.e.:

$$T_1 = Q_0$$

Finally, we find that T_0 should have the value 1 always, i.e.:

$$T_0 = 1$$

The counter with D-type triggers must have a value of 1 at input D_i in every state where the output Q_i is to stay in 1 or to change to 1 with a CLK pulse. Therefore (after adjustment):

$$D_2 = Q_2 \bar{Q}_1 + Q_2 \bar{Q}_0 + \bar{Q}_2 Q_1 Q_0$$

$$D_1 = Q_1 \bar{Q}_0 + \bar{Q}_1 Q_0$$

$$D_0 = \bar{Q}_0$$

Obviously, the solution with **T triggers** is much simpler than with D circuits - but it is valid only for **binary** counters (!).

Example:

Solution of a synchronous **three-bit Gray** counter counting up:

state:			excitation function					
old	->	new						
$Q_2 Q_1 Q_0$	->	$Q_2 Q_1 Q_0$	T_2	T_1	T_0	D_2	D_1	D_0
0 0 0	->	0 0 1	0	0	1	0	0	1
0 0 1	->	0 1 1	0	1	0	0	1	1
0 1 1	->	0 1 0	0	0	1	0	1	0
0 1 0	->	1 1 0	1	0	0	1	1	0
1 1 0	->	1 1 1	0	0	1	1	1	1
1 1 1	->	1 0 1	0	1	0	1	0	1
1 0 1	->	1 0 0	0	0	1	1	0	0
1 0 0	->	0 0 0	1	0	0	0	0	0

Fig. 6: Table of transitions and excitation functions of a three-bit Gray counter for counting up

Using the same procedure as explained above, we can get the excitation functions for T_2 , T_1 , T_0 or D_2 , D_1 , D_0 , and after their minimization (in maps), the following expressions have been obtained:

$$\begin{aligned}
T_2 &= Q_2 \cdot \bar{Q}_1 \cdot \bar{Q}_0 + \bar{Q}_2 \cdot Q_1 \cdot \bar{Q}_0 \\
T_1 &= \bar{Q}_2 \cdot \bar{Q}_1 \cdot Q_0 + Q_2 \cdot Q_1 \cdot Q_0 \\
T_0 &= \bar{Q}_2 \cdot \bar{Q}_1 \cdot Q_0 + \bar{Q}_2 \cdot Q_1 \cdot Q_0 + Q_2 \cdot \bar{Q}_1 \cdot Q_0 + Q_2 \cdot Q_1 \cdot \bar{Q}_0 \\
D_2 &= Q_1 \cdot \bar{Q}_0 + Q_2 \cdot Q_0 \\
D_1 &= Q_1 \cdot \bar{Q}_0 + \bar{Q}_2 \cdot Q_0 \\
D_0 &= \bar{Q}_2 \cdot \bar{Q}_1 + Q_2 \cdot Q_1
\end{aligned}$$

The variant with **D triggers** looks much simpler, so we will prefer it.

Various counters can be solved in a similar way. As an example, transition tables of a binary counter down, counter modulo 6 up, and binary counter up with saturation are given:

$S^t \dots \rightarrow S^{t+1}$					
1	1	1	1	1	0
1	1	0	1	0	1
...
0	0	1	0	0	0
0	0	0	1	1	1

$S^t \dots \rightarrow S^{t+1}$					
0	0	0	0	0	1
0	0	1	0	1	0
...
1	0	0	1	0	1
1	0	1	0	0	0

$S^t \dots \rightarrow S^{t+1}$					
0	0	0	0	0	1
0	0	1	0	1	0
...
1	1	0	1	1	1
1	1	1	1	1	1

Fig. 7: Transition tables of different counters
left: binary down, middle: modulo 6 up, right: binary up with saturation

The saturation arithmetic is used in many tasks and the **saturation counter** is a part of it. Numbers in the range of 0 to 2^{n-1} can be expressed using n bits. However, if the range is exceeded when adding two numbers, an error will occur. The same applies to counting pulses by the counter - from the maximum value, by adding 1 pulse, the counter does not get to a larger number, but to zero. This might result in a big error. It would be better if the counter from the highest number would not get any further, but it would be blocked or **saturated**. Of course, the error occurs again, but much smaller. Normally, bidirectional counters with saturation are constructed in this way. The one-way counter must be reset after saturation, there is no other way. A bidirectional counter, saturated in one direction, can count in the other direction and thus get out of the saturation.

1.4 Bidirectional counters

A two-way counter can be easily designed by combining the design of an up-counting counter and a down-counting counter. There is one common group of triggers for both directions, but different combinational circuits, one for each direction. Multiplexers, switched by the DIR signal, are inserted into the triggers' inputs, and the respective signals are always fed to the inputs belonging to the required counting direction - see Fig. 8 and Fig. 1. Figure 8 also shows alternative additional circuits (RS plus AND gate) for counter control by UP and DN pulse inputs. The correct timing of these pulses must be respected to avoid metastability in the RS circuit.

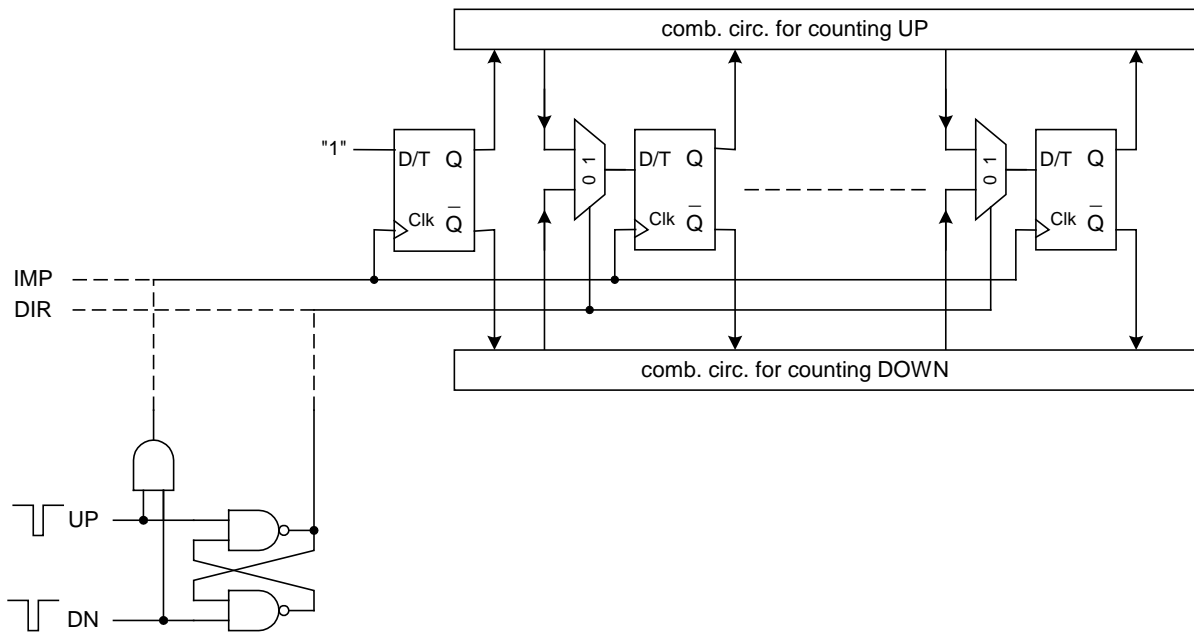


Fig. 8: Principle of bidirectional counter

1.5 Other counter inputs and outputs

Resetting and loading the counter

It is often necessary to reset the counter time by time. The only exception may be a frequency divider, where the initial state does not make sense. The CLR (clear) input, sometimes referred to as RESET or NULL, is used for **resetting**.

In other cases, it is necessary to **preload** a counter to some value before the counting starts. The input of the initial state is controlled the LD (Load) input. Data for presetting is fed to the DI (Data IN) input.

Asynchronous inputs of R and S are used for resetting and setting - see chapter Triggers and metastability. Since these inputs are asynchronous, clearing or preloading the counter takes precedence over other input signals and is possible at any time. Figure 9 shows a counter with all described inputs.

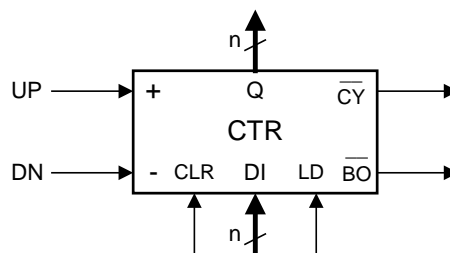


Fig. 9: Counter with reset and data load inputs

Carry and Borrow from the counter

Carry is an output signal informing about the transition from state 11 ... 11 to state 00 ... 00 when counting up, and **Borrow** is an output signal informing about the transition from state 00 ... 11 to state when counting down. The Carry or Borrow output is used for cascade connection with other counters, or for other purposes - e.g. in timers, filters, modulators, etc.

The simplest is **asynchronous carry**, used in asynchronous counters, consisting only in outputting the signal from the highest trigger. Asynchronous carry prolongs the stabilizing time of the whole counter, as it must gradually go through all stages - see Fig. 10.

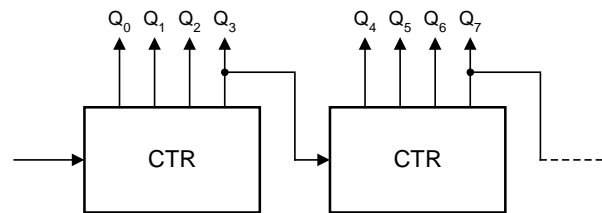


Fig. 10: Asynchronous carry from the counter

For **synchronous cascading**, a separate output is created in the counter, on which a short **CY** (carry) pulse is issued at the appropriate transition. The principle is shown in Figure 11 for a counter counting up. In the case of a counter with counting down, the principle remains the same, only instead of state 11 ... 11 the state 00 ... 00 is distinguished - negated outputs of triggers are introduced into the product. The output signal is then referred to as **BO** (Borrow). For reversible counters with UP and DN inputs, both **CY** and **BO** are generated.

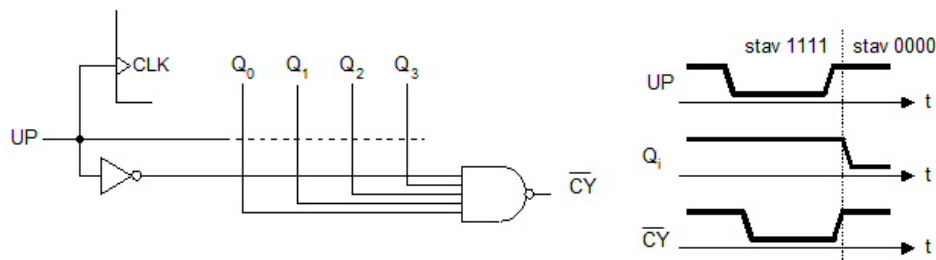


Fig. 11: Circuits for generating synchronous carry from the counter

Synchronous carry allows faster operation than asynchronous carry. From Fig. 11 it is clear that the carry pulse is generated by the counted pulse (**UP**) when the counter is still in the state 11 ... 11, i.e. before the transition from 11 ... 11 to 00 ... 00 is finished. This increases the speed of cascaded counters.

The Fig. 12 shows the connection of synchronous reversible counters to cascade with all the previously mentioned inputs and outputs.

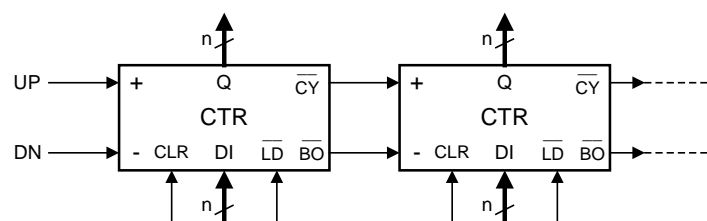


Fig. 12: Cascading synchronous reversible counters

1.6 Shift registers

By connecting type D triggers in a cascade with a common distribution of clock pulses, a **shift register** is created. The basic connection and graphic symbol are shown in Fig. 13.

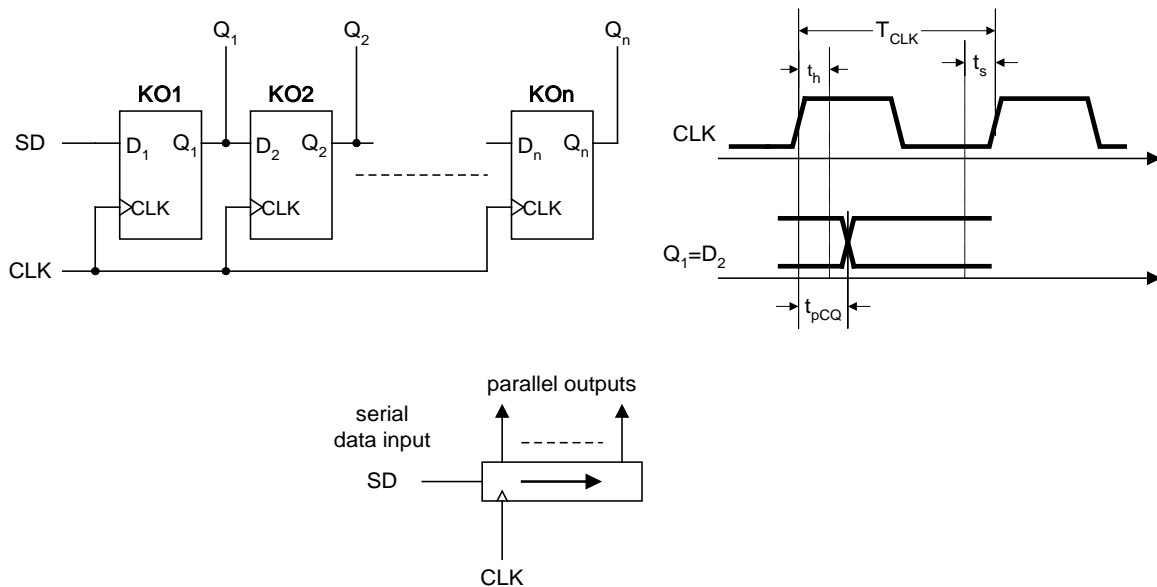


Fig. 13: Shift register

By a clock pulse, the state at input D of the trigger KO1 moves to its output Q₁, but at the same time the original state at output Q₁ of the trigger KO1 moves to output Q₂ of the trigger KO2, etc. As a result, the entire register contents **shifts to the right**, on the first position in the left is inserted the input data, and after the last trigger in the right it is **lost**. However, this simple operation is not self-evident and requires **proper timing**, as shown in the Fig. 13 on the right. The following applies:

$$t_{pCQ} \geq t_h$$

$$T_{CLK} \geq t_{pCQ} + t_s$$

In the arrangement according to the previous figure, all triggers must be **edge** controlled. Level control cannot be used, because during the time when $CLK = 1$, the relation $Q_i = D_i$ would apply to all triggers. The result would be the entire register filled with the value at the SD input.

One of the most common shift register applications is converting **serial** code to **parallel** code. Serial data is delivered to the SD input bit by bit and is always accompanied by a CLK pulse. After the appropriate number of pulses, the parallel data is available at the Q outputs. The opposite application, i.e. the conversion of **parallel** data to **serial** data, requires that the data shift register be preloaded and then the appropriate number of CLK pulses be delivered. Serial data are available from the last trigger.

For filling the data shift register, some registers are equipped with circuits for parallel data **loading** or **presetting**. Their solution is similar as for counters - see Fig. 14.

Shifting data in only one direction and by only one place is not the only alternative to shift registers. There are shift registers with the option to shift **left** or **right**, or with shifts by an optional **number of places**.

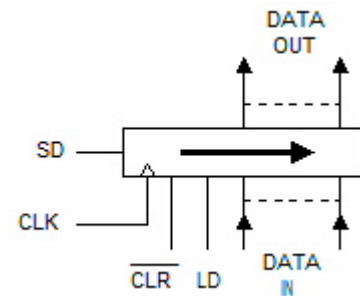


Fig. 14: Shift register control signals

The above options require an extensive multiplexer system. In addition to this variant, there is also a very universal variant with a data register and an array of **CMOS switches** - see Fig. 15. The switches are marked by dots on the crossing of wires. The complete array of switches allows not only shifts left and right by any number of positions, but also, for example, reversal of the bit order, literally "anything anywhere". This solution is utilized in arithmetic-logic units of computers optimized for digital signal processing.

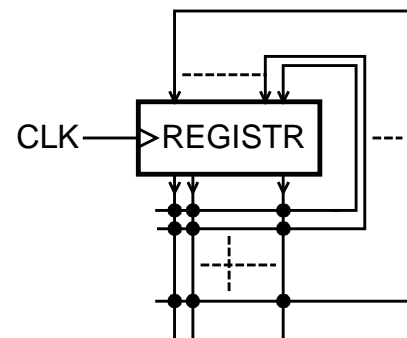


Fig. 15: Register with switching field as a shift register function

1.7 Feedback shift registers

If a signal from some outputs of the shift register is fed to the serial data input - either directly or after processing by a combinational circuit - a feedback is created, with which the shift register acquires new properties. The simplest case is a **circular register**, in which the feedback is conducted directly from the last trigger - see Fig. 16. The data entered in advance in the shift register then constantly rotates one place at a time with each clock (shift) pulse. Rotating circular register contents can be used e.g. for the generation of multiphase control pulses for running banners on display panels, and the like. The advantage of such generators is the ability of rapid reprogramming simply by inserting different initial content of the register.

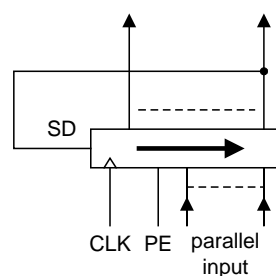


Fig. 16: Circular register with serial and parallel input

A more complex combinational circuit can also be involved in the feedback. Of particular importance is the involvement with **XOR** gates (non-equivalence). A simple XOR device has only two inputs, but they can be grouped because the associative law applies to the XOR function:

$$a \oplus b \oplus c \oplus d \oplus \dots = (a \oplus b) \oplus (d \oplus e) \oplus \dots, \text{ or also}$$

$$a \oplus b \oplus c \oplus d \oplus \dots = (((a \oplus b) \oplus c) \oplus d) \oplus \dots$$

The general principle is shown in Figure 17 on the left, specific examples of connections with a connection of two and four triggers are shown in the figures on the right.

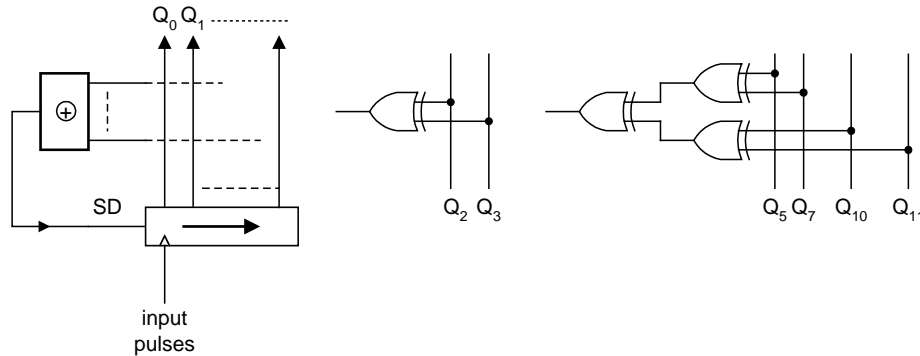


Fig. 17: Feedback via XOR gates

This whole circuit is called a **linear counter** and belongs to the category of linear sequential circuits. Their characteristic feature is that they are composed exclusively of three types of components - D triggers, XOR gates, and circuits for constant multiplication (but these are not present in the linear counter).

The linear counter cycles through a sequence of states, the length of which is always shorter than $2^N - 1$. Its length depends on the choice of shift register branches used for feedback. The rule is that for any length of the shift register, it is always possible to find at least one combination of branches so that the sequence of states is exactly $2^N - 1$. It is a so-called **sequence of maximum lengths**. This brings the linear counter closer to a binary counter with 2^N states in terms of the economy of the use of circuits. The feedback XOR gates form an odd parity generator. The only state that is missing in the sequence of the maximum length is the state $00 \dots 0$, because in that case the feedback circuit gives zero and after shifting the contents of the register there is again a state $00 \dots 0$ in it. For the correct function of the linear counter, it is necessary to set the **initial state** of the register other than $00 \dots 0$ after the power supply has been switched ON.

Further shown is Fig. 18, which specifies expressions in the feedback of the different length registers. Here Q_0 indicates the first trigger (closest to the serial input).

<u>N</u>	<u>expression for feedback</u>
2	$Q_0 \oplus Q_1$
3	$Q_1 \oplus Q_2$
4	$Q_2 \oplus Q_3$
5	$Q_2 \oplus Q_4$
6	$Q_4 \oplus Q_5$
7	$Q_5 \oplus Q_6$
8	$Q_3 \oplus Q_4 \oplus Q_5 \oplus Q_7$
9	$Q_4 \oplus Q_8$
10	$Q_6 \oplus Q_9$
11	$Q_8 \oplus Q_{10}$
12	$Q_5 \oplus Q_7 \oplus Q_{10} \oplus Q_{11}$
.....	

Fig. 18: Branch positions for the maximum sequence length N

As an example, the sequence of states for $N = 4$ and the initial state 0001 is shown below. The order of the triggers is $Q_0 Q_1 Q_2 Q_3$:

```
0001→1000→0100→0010→1001→1100→0110→1011→0101→1010→1101→1110→1111→0111→0011
└-----<-----┘
```

In the state 0001, $Q_2 = 0$ and $Q_3 = 1$, i.e. $Q_2 \oplus Q_3 = 1$, and 1 enters from the left into the shift register at position Q_0 . The next state after shifting to the right is therefore 1000, and similarly for other states. If we understand the state of the register as a decimal number ($Q_0 = \text{MSB}$), we get the sequence:

```
1 → 8 → 4 → 2 → 9 → 12 → 6 → 11 → 5 → 10 → 13 → 14 → 15 → 7 → 3
└-----<-----┘
```

The bit sequence at the output of any trigger is also interesting. On Q_0 it is as follows:

010011010111100 and repeatedly 010

This is close to the **random signal** - but it is periodic, i.e. not random, but **pseudo-random**. The generation of pseudo-random sequences is one of the frequent applications of linear counters.

2 Timers

The task of the timers is to change the length of the pulse applied to the input. It can be shortened to the required length, extended to the required length, or stabilized to this length. The three cases are shown in the following Figure 19. The circuit for **shortening the** pulse length shorter pulse length t_W , but transmits shorter pulses unchanged. Circuit for the **extension** of the pulse length transmits long pulses unchanged, and a short pulse length is extended to t_W . The circuit for pulse length **stabilization** generates a pulse of length t_W regardless of the length of the input pulse. Independence of the input pulse length means that the circuit is **edge controlled**.

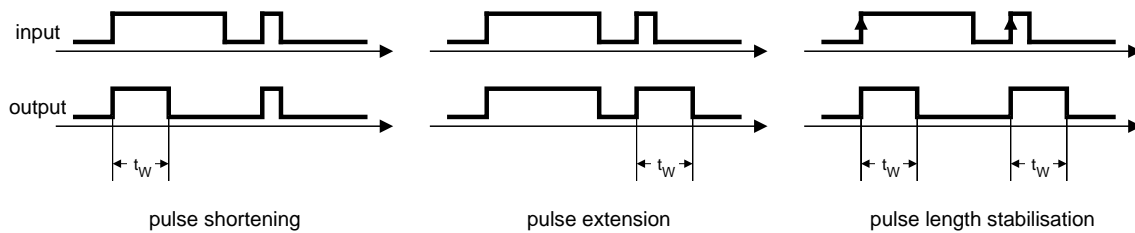


Fig. 19: Pulse length adjustment variants

The basic part of the timer is the delay element. It can be analog or digital. In the analog or **asynchronous** version, circuits with capacitors or chokes or elementary logic devices (inverters) are used. In digital or **synchronous** versions, counters with oscillators are used.

2.1 Asynchronous timing circuits

Circuits with RC cells can be used to delay, shorten or shift the pulses. The following Fig. 20 shows the simple circuit for the time delay t_D . The use of it is very limited, as the voltage profile behind the RC cell is gradual and unsuitable for the next digital device (possibility of oscillation). The required delay can also be achieved with one or more inverters in a cascade. The advantage is the steep edges of the pulses, but the delay can only be set in rough steps by individual t_{pd} . However, this is suitable for some applications.

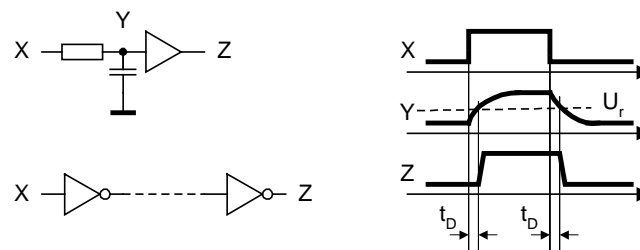


Fig. 20: Simple asynchronous circuits for pulse delay

By combining delay and logic functions, a number of circuits can be implemented that shorten pulses. In the figure on the left, it is a circuit that shortens the pulse by cutting off its end - in the figure on the right, on the contrary, the beginning of the pulse is delayed. Timing diagrams explain everything.

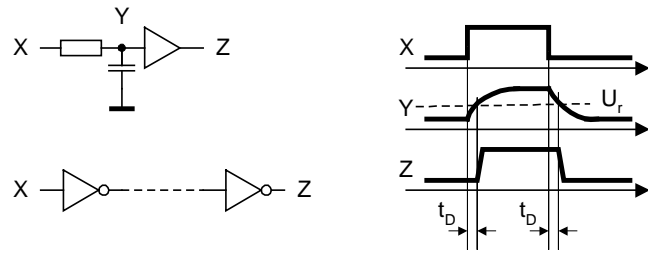


Fig. 21: Simple asynchronous circuits for pulse shortening

Using the XOR gate, a circuit can be assembled which generates a short pulse at each input state change and thus implements the function of an asynchronous **change detector** - see Fig. 22.

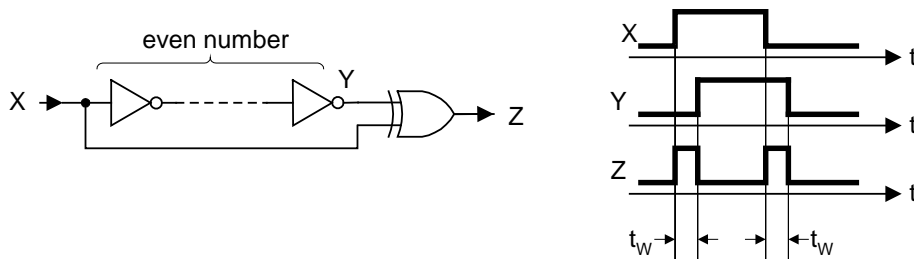


Fig. 22: Asynchronous change detector

The disadvantage of the gates as delay elements is the impossibility of **continuous** time setting (only by multiples of t_{pd}), low accuracy and timing instability. To keep the delay constant, it would be necessary to keep constant the supply voltage and temperature. However, there are applications where this simple solution is convenient. An example is shown in Fig. 23.

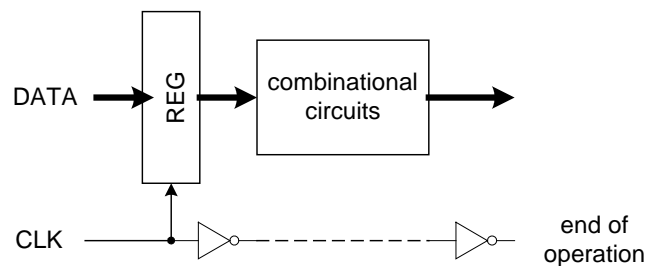


Fig. 23: Signaling end of operation

The combinational circuit is equipped with an input register. The entire stage should accept data for processing, process them, and signal the end of the operation. It would be possible to

monitor changes in the outputs of the combinational circuit, but the problem is that some operations may give identical outputs and then monitoring the changes would not work. The solution consists in including a delay line, which is guaranteed to have a **longer** delay than the monitored combination circuit. Since it is made on the same chip with the monitored circuit, the temperature and voltage dependences are the same and thus the delay in the delay line will always be in a fixed ratio to the delay in the monitored circuit.

Timing elements based on gate delay are suitable for processing very short time functions - the typical delay of inverters on a printed circuit board is a few *ns*, on the chip tens of *ps*. However, for significantly longer times, an unacceptably long gate strings would be required.

For **long time** functions, connections based on integrators or capacitors charged by current sources, and precision analog comparators are used. These timing elements belong to the field of analog technology, but for the implementation of full-value timers they are supplemented by digital circuits. An example is shown in Fig. 24.

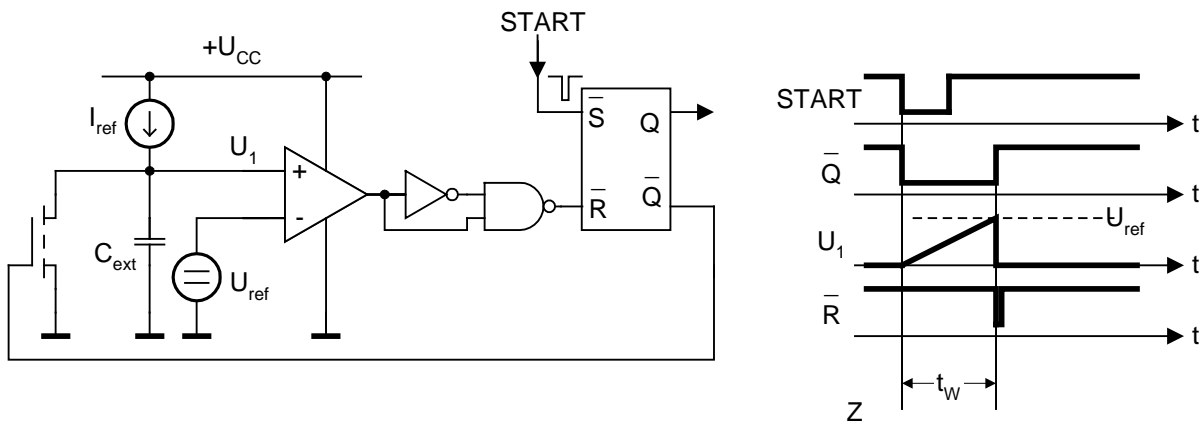


Fig. 24: Timer for long times based on the analog principle

The basis is a precise timing element and a trigger. At steady state $\bar{Q} = 1$, the transistor is switched ON and the timing capacitor is discharged. The $START$ pulse toggles the trigger, the transistor turns OFF and the capacitor is charged with current from the precise current source. So far, the analog comparator has a low voltage U_L at the output. When U_1 and U_{ref} become equal, the comparator gives voltage U_H , the trigger is switched to $Q = 0$, $\bar{Q} = 1$ via the pulse-shortening circuit, and the capacitor is reset again. The timing diagram shows the waveforms characterizing the operation of the circuit. The duration of the temporary state where $Q = 1$ and $\bar{Q} = 0$, is denoted as t_w . The whole circuit therefore has one stable state, when $Q = 0$, and one temporary state, when $Q = 1$. Hence the name **monostable** trigger, abbreviated **MKO**.

If another input pulse occurs in the circuit of Fig. 24 during the temporary state, it is ignored because the trigger is already toggled. The circuit is **Non-Retriggerable** and will be called **MKO-NR**. There are also monostable triggers of the second type, which start measuring time again after each $START$ pulse. The circuit is **Retriggerable** and will be called **MKO-R**. Fig. 25 shows time diagrams of both alternatives.

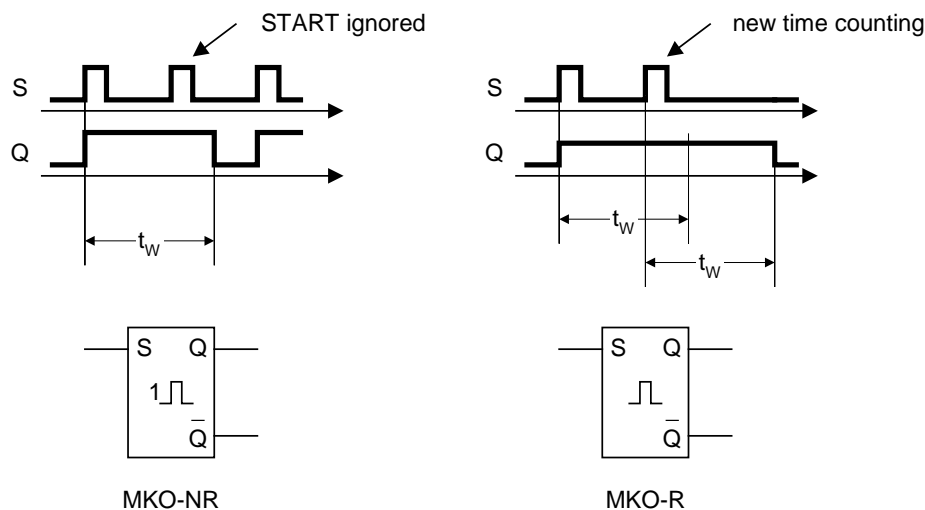


Fig. 25: Two types of MKO response to a START pulse

The periods achievable with these circuits are "long", but only relatively - a **few seconds**. The capacitor is large and cannot be integrated on chip. Really long times - even hours and more - can only be achieved with **synchronous timers**. These have the additional advantages of being able to program pulse lengths and being integrated.

2.2 Synchronous timing circuits

For synchronous timing circuits, timing is achieved by counting the clock pulses using counters. The output pulses generated by the synchronous timing circuits have a length or delay expressed in the number of cycles of the clock pulses and their accuracy depends only on the accuracy of the frequency of the generator of clock pulses. When controlled by a crystal, it can be very accurate (10^{-4} and better). The timing error is within one *CLK* period. Another advantage of the timing circuits in the synchronous version is that the output pulses, which are supplied to other parts of the digital system, have precisely **defined timing** with respect to the synchronizing *CLK* pulses distributed throughout the system from one central generator. The possibility of program setting the pulse length is also important.

The principle is shown in the following figure 26. The D-trigger is in the state 0, the CTR counter is permanently reset and does not count. The trigger is toggled by the *START* pulse, the resetting ends and the counter starts counting the *CLK* pulses. When the number in the counter matches the number that was previously written in the length register, the trigger is reset and thus the counter is reset and the circuit is ready to generate another pulse. However, false pulses may be generated in the comparator, and therefore the signal for resetting the trigger is formed as the product of the signal *K* with the inverted signal *CLK*. It is therefore active only in the **second half** of the *CLK* cycle, when the false impulses have already disappeared (see the chapter Transients in combinational circuits). Since the trigger remains in the state 1 for the duration of the output pulse, any repeated trigger pulses have no effect during this time. It is therefore a synchronous variant of **MKO-NR**.

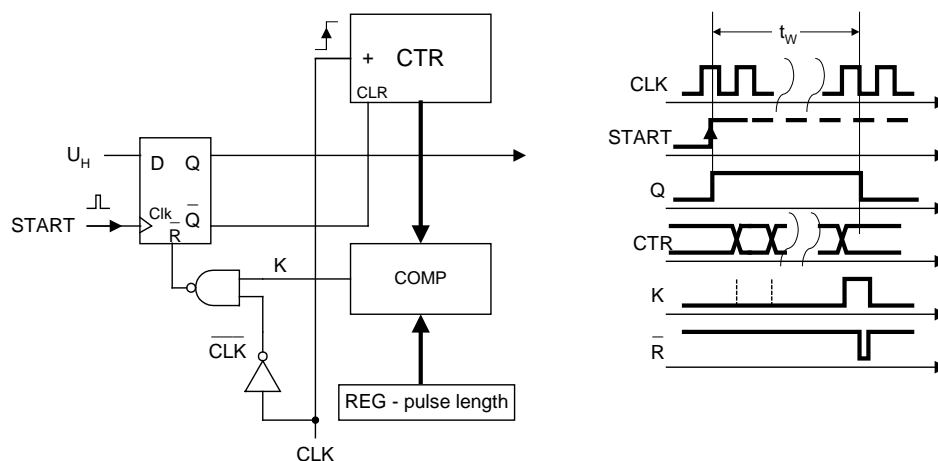


Fig. 26: Synchronous timer for longer pulses generation

The next figure 27 shows the principle of the synchronous variant **MKO-R**. A small change to the previous circuit will suffice. The trigger pulses can reset the counter here at any time, i.e. also **during** the duration of the output pulse. This starts the counting down again from zero.

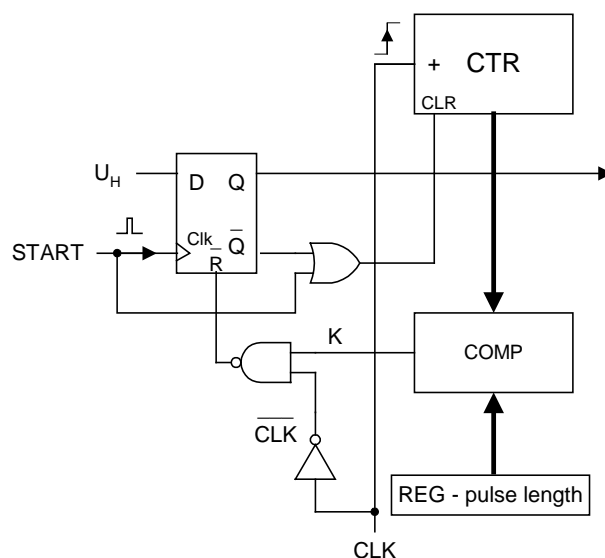


Fig. 27: Synchronous timer - retriggerable

Another connection of a synchronous MKO with a preset counter is shown in the following figure 28. In this case, the counter counts **down** from the preset value. When counting down to zero, an impulse is generated at its output BO, which resets the trigger. Before generating another pulse, the counter must be **preset** to the pulse length value again. For this reason, the connection is not suitable for a separate repetitive function, but is very often used in the peripheral circuits of microcomputers. The timer is set by **software**, but is **hardware-started**.

2.4 Pulse width modulator

Pulse Width Modulation (**PWM**) is a simple way to convert a number to an analog signal (mean voltage). Periodically generated pulses have a period T and a length (width) t_1 , so ideally the mean value of the voltage is:

$$U_m = U_1 \cdot \frac{t_1}{T} ,$$

where U_1 is the pulse amplitude and U_m its mean value. A low-pass filter, which smooths the pulse waveform, can be used as a demodulator. However, many devices controlled by a PWM signal have sufficient inertia in themselves, so that a filter is not needed (electromagnets, electric motors, electro-thermal devices, etc.). The filter or the inertia of the device itself is a very effective measure for suppressing possible interference in the transmitted PWM signal. A simple synchronous PWM modulator is shown in Fig. 30.

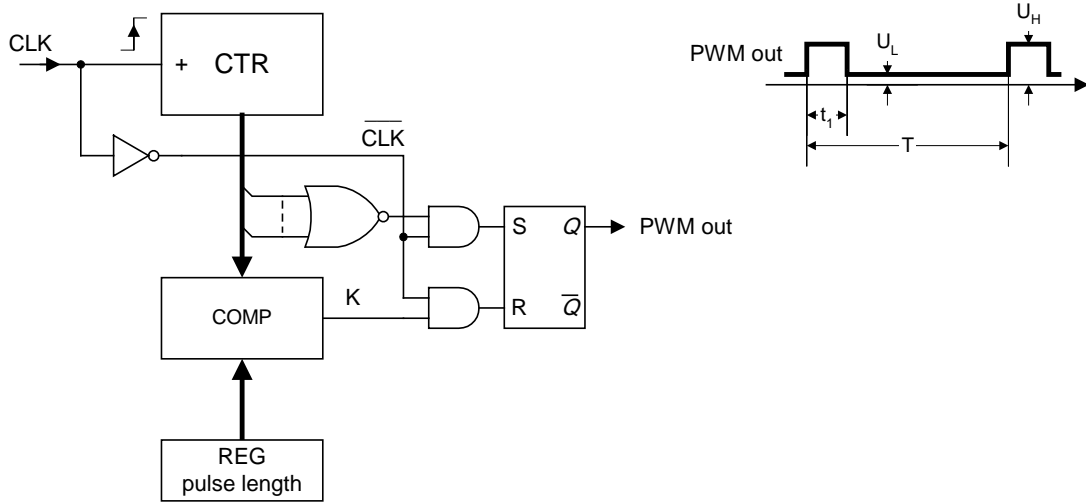


Fig. 30: Pulse width modulator

The CTR counter **modulo** M continuously counts the pulses and in the state 0 it sets the RS trigger to state 1. After reaching the state identical to the **number** N stored in the pulse length register, the comparator resets the trigger - this gives the pulse length $t_1 = N \cdot T_{CLK}$ (T_{CLK} is the CLK period). However, the counter continues to count, and after the last state of its M -state cycle it follows the state 0 and thus the trigger is again switched to state 1. Therefore,

$$T = M \cdot T_{CLK}$$

and the mean value of the PWM signal is proportional to N/M ratio.

The **relative error** of the ratio t_1/T is given by the number of counter states in the period T , or by the counter module. E.g. with an 8-bit counter with module 256, the relative error is approximately 0.4%. Each additional bit of the counter halves the error.

The finite number of counter states is not the only source of inaccuracies. Since the output signal is a digital signal with levels U_H and U_L , the exact relationship for U_m is:

$$U_m = (U_H - U_L) \cdot \frac{N}{M} + U_L$$

Since the voltages U_H and U_L do not represent constants, but whole ranges of values, U_m is expressed rather inaccurately. Therefore, to achieve higher accuracy, the demodulator must convert the digital PWM signal to the signal with more precisely defined levels via precision switches and precisely generated voltages.