Technical Note 125

AC500 V3 PLC – Bistable, Counter, Timer & Trigger

Description and usage of standard library function blocks

The standard library contains useful function blocks. This document provides hints about when and how to use them. The string functions of the standard library are not included in this document.

PART 1 - BISTABLE FUNCTION BLOCKS

The standard library contains two bistable function blocks. Also known as RS flip-flop.

- RS: Bistable reset-dominant latch
- SR: Bistable set-dominant latch

RS and SR are bistable function blocks. These blocks are used to hold a signal, when e.g., a pushbutton is pressed. In this example there are two pushbuttons for starting and stopping a machine. Would there be direct contact from the pushbutton to the machine, the machine is only running as long the pushbutton is pressed. To hold the signal even if the start button is not pressed anymore an RS function block can be used. Another pushbutton is then used. To reset the signal again.

The RS function block is an edge triggered function block, this means it is only important when the button was pushed and not how long.

The table below shows the behavior of the RS function block. Q is the output value. The table shows a timeline which can be read from left to right.

Cycle	1	2	3	4	5	6	7	8	9	10	11	12	13
Set	0	1	1	0	0	0	0	1	1	1	1	0	1
Reset	0	0	0	0	0	1	0	0	0	1	0	0	0
Q	0	1	1	1	1	0	0	1	1	0	0	0	1

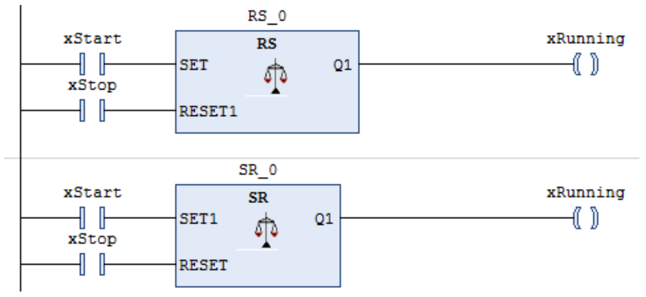
In the 2nd cycle the Set input becomes TRUE, so the output Q is also TRUE. Independent of Set is TRUE or FALSE the output Q will remain TRUE until in the 6th cycle the reset becomes TRUE. Then the output Q is reset. With a new Set input in cycle 8 the output becomes TRUE again. It remains true until in the 10th cycle the Reset is TRUE. Even if the Reset becomes false in the 11th cycle again the output remains FALSE. Only a rising edge of the Set input will activate the output again as visible in the 13th cycle.

What happens if Set and Reset are triggered at the same time? This depends on the usage of the RS function block is Reset dominant. This means in this condition the output is reset.

The SR function block is Set dominant. All other functionality is the same as described above.

The examples below show how to use the function block in different programming languages.

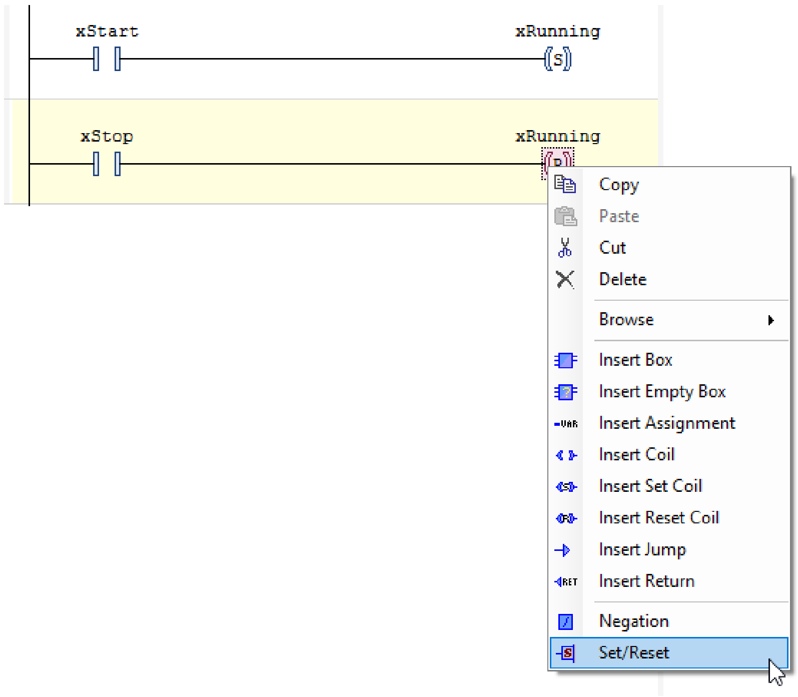
Ladder Diagram



Structured Text

```
RS_0(SET:= xStart, RESET1:= xStop, Q1=> xRunning);
SR_0(SET1:= xStart, RESET:= xStop, Q1=> xRunning);
```

In contrast to using the function block, the output itself can be changed to Set or Reset. This can be done by right clicking on the output and changing Set/Reset.



Set/Reset outputs are always reset dominant and in contrast to the function block level triggered and not edge triggered. In contrast to the edge triggered function blocks below the behavior will be different when using the Set/Reset on the outputs. The table below highlights the difference in the 11th cycle.

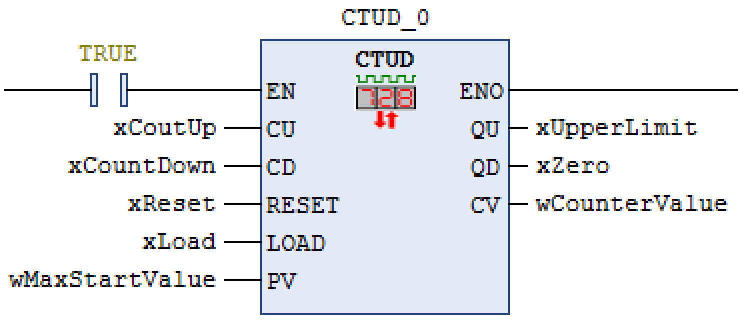
Cycle	1	2	3	4	5	6	7	8	9	10	11	12	13
Set	0	1	1	0	0	0	0	1	1	1	1	0	1
Reset	0	0	0	0	0	1	0	0	0	1	0	0	0
Q	0	1	1	1	1	0	0	1	1	0	**1**	0	1

PART 2 - COUNTER

The standard library contains three counters

- CTD: Counter Down
- CTU: Counter Up
- CTUD: Counter Up Down

Here only the CTUD counter is described, which combines the features of the up and down counter.



The screenshot above shows the call of the CTUD block in a ladder logic program.

The function block has five inputs

- CU: Rising edge increments CV by one
- CD: Rising edge decrements CV by one
- RESET: TRUE Reset CV to 0
- LOAD: TRUE set CV to start value PV
- PV: Start value loaded for decrementing/ upper limit for incrementing

The function block has three outputs

- QU: TRUE, if the upper limit is reached: CV >= PV
- QD: TRUE, if CV = 0
- CV: Current counter value

The same logic as described above is shown below in Structured Text.

```
CTUD_0(
      CU:= xCountUp,
      CD:= xCountDown  ,
      RESET:= xReset,
      LOAD:= xLoad,
      PV:= wMaxStartValue,
      QU=> xUpperLimit,
      QD=> xZero,
      CV=> wCounterValue);
```

PART 3 - TIMER

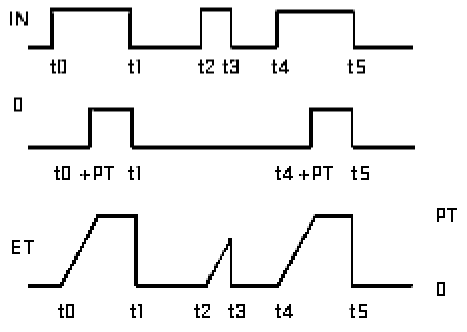
The standard library contains three timers

- TON: Timer with turn on delay
- TOF: Timer with turn off delay
- TP: Pulse Timer

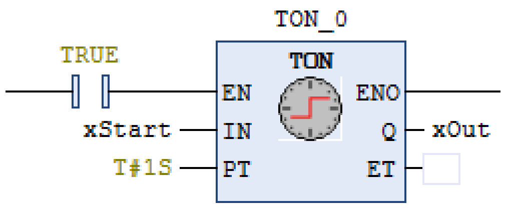
TON

The block has a Boolean IN input and the delay time PT as inputs. If the input IN is TRUE for more than the delay time PT the output Q will become true. The output ET shows how long the Input IN is already set.

The picture below shows the behavior of the TON function block. At t0 IN becomes TRUE, after the time PT is elapsed the output Q becomes true. At t1 IN becomes FALSE, Q follows immediately. Between t2 and t3 IN is again TRUE, but the time between is lower PT so Q remains FALSE. The behavior between t4 and t5 is the same as between t0 and t1.



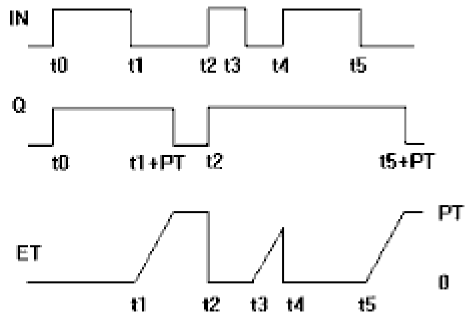
Below the TON is represented in ladder logic and structured text.



```
TON_0(IN:= xStart, PT:= T#1S, Q=> xOut, ET=> );
```

TOF

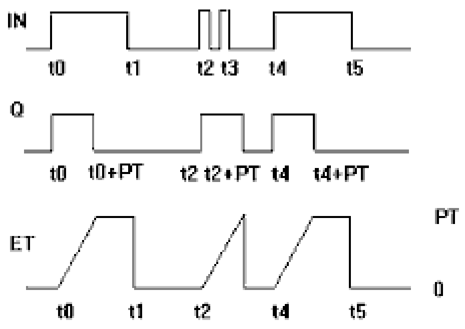
The delay timer works the same way as the TON function block, just the input signal is inverted. Whenever IN is FALSE for at least PT, the output Q becomes true.

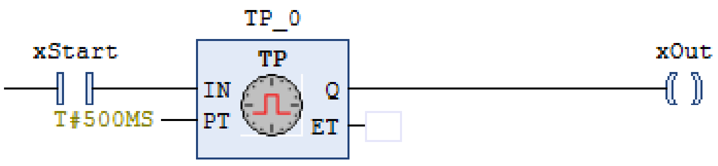


The inputs and the call is the same as in the TON function block.

TP

The pulse timer TP gives a pulse with a dedicated length (PT). The pulse is triggered with a rising edge at IN input. The output ET gives the elapsed time since the rising edge of the IN input. The picture below shows the behavior of the TP function block when IN is triggered.





```
TP_0(IN:= xStart, PT:= T#500MS,
     Q=> xOut);
```

The call of the function block in ladder logic and structured text is shown above.

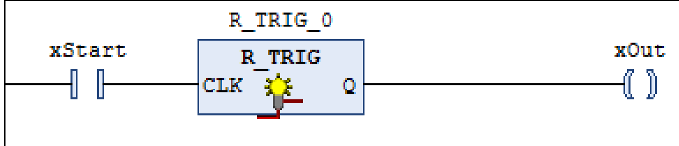
PART 4 - TRIGGER

The standard library contains two triggers

- F_TRIG: Falling edge detection of Boolean signal
- R_TRIG: Rising edge detection of Boolean signal

The trigger function blocks are used to detect rising edges of a signal. The output Q is TRUE for exactly one cycle, if an edge is detected. A use case could be a pushbutton. An action should be executed when the button is pressed. When directly connecting the input to the call, the action is called several times, if the pushbutton is pressed. When using the R_TRIG function block, the call is only performed once, when the rising edge is detected. Like the R_TRIG function block the F_TRIG function block can be used to detect falling edges. In the case of a pushbutton this is the moment when the button is released.

The call of the function block in ladder logic and structured text is shown below.

	`R_TRIG_0(CLK:= xStart, Q=> xOut);`