

LamaPLC: Simatic S7 SCL commands: Timer / counter functions



Counter operations

IEC Counters

Counters - similar to IEC times - can be called in two ways, embedded or with a separate DB. I don't feel like describing this separately, please take a look at the times: [IEC Timers](#).

>> [Back to LamaPLC main menu \(SCL commands\)](#)

CTU: Count up

The function that only counts upwards.

The CTU counts the positive edges of the "CU:" parameter. As soon as it reaches the value of "PV" with the calculation, the output "Q" changes to "TRUE". The function continues to count. The function can be restarted with the "R" input.

CTU types:

- CTU_SINT / CTU_USINT: the calculation goes with the variable types [SINT](#), [USINT](#)
- CTU_INT / CTU_UINT: the calculation goes with the variable types [INT](#), [UINT](#)
- CTU_DINT / CTU_UDINT: the calculation goes with the variable types [DINT](#), [UDINT](#)
- CTU_LINT / CTU_ULINT: the calculation goes with the variable types [LINT](#), [ULINT](#) (only on S7-1500 PLC)


FB			
CTU (
	CU: BOOL , input		Counter signal: The function monitors the positive edge of this binary signal. The counter counted, if signal change from "FALSE" to "TRUE".
	R: BOOL , input		Reset: Reset CV value.
	PV: INT , input		Set value to Q: Value at which the Q output is set

	Q: BOOL ,	output	Result: When the counter reaches "PV".
	CV: INT / CHAR / WCHAR / DATE	output	Current counter value
);			

Example

<pre> 1 // timer set "secTimerQ" in every second 2 #secTimer(IN:=NOT(#secTimerQ), 3 PT:=t#1s, 4 ET=>#secTimerET); 5 6 #secTimerQ := #secTimer.Q; 7 8 // secTact change state every second 9 IF #secTimerQ THEN 10 #secTact := NOT (#secTact); 11 END_IF; 12 13 // testCounter count every 2 second (its works with positive edge 14 // Q will "true" after 48 second 15 #testCounter(CU:=#secTact, 16 PV:=24, 17 Q=>#testCounterQ, 18 CV=>#testCounterInt); 19 20 // actual state from testCounter 21 #tempInt := #testCounterInt; </pre>	<table border="1"> <tr><td>#secTimerQ</td><td>FALSE</td></tr> <tr><td>#secTimerET</td><td>T#739MS</td></tr> <tr><td>#secTimerQ</td><td>FALSE</td></tr> <tr><td>Result</td><td>FALSE</td></tr> <tr><td>#secTact</td><td>FALSE</td></tr> <tr><td>#secTact</td><td>FALSE</td></tr> <tr><td>#testCounterQ</td><td>TRUE</td></tr> <tr><td>#testCounterInt</td><td>141</td></tr> <tr><td>#tempInt</td><td>141</td></tr> </table>	#secTimerQ	FALSE	#secTimerET	T#739MS	#secTimerQ	FALSE	Result	FALSE	#secTact	FALSE	#secTact	FALSE	#testCounterQ	TRUE	#testCounterInt	141	#tempInt	141
#secTimerQ	FALSE																		
#secTimerET	T#739MS																		
#secTimerQ	FALSE																		
Result	FALSE																		
#secTact	FALSE																		
#secTact	FALSE																		
#testCounterQ	TRUE																		
#testCounterInt	141																		
#tempInt	141																		

The example program below can be downloaded here:


ctu_example.scl

Here is a description of how to import the downloaded program to the TIA Portal:
[Import source code to the TIA portal.](#)

>> [Back to LamaPLC main menu \(SCL commands\)](#)

CTD: Count down

The function that only counts downwards.

The CTD counts the positive edges of the "CD" parameter, the current counter value of the CV parameter is decremented by one. The function can be restarted with "LD" input, in this case, the countdown starts again from the "PV" value. If the current counter value is less than or equal to zero, the "Q" parameter is set to signal state "TRUE".

CTD types:

- CTD_SINT / CTD_USINT: the calculation goes with the variable types [SINT](#), [USINT](#)
- CTD_INT / CTD_UINT: the calculation goes with the variable types [INT](#), [UINT](#)
- CTD_DINT / CTD_UDINT: the calculation goes with the variable types [DINT](#), [UDINT](#)
- CTD_LINT / CTD_ULINT: the calculation goes with the variable types [LINT](#), [ULINT](#) (only on S7-1500 PLC)


FB			
CTD (
	CD: BOOL ,	input	Counter signal: The function monitors the positive edge of this binary signal. The counter dountcounted, if signal change from "FALSE" to "TRUE".
	LD: BOOL ,	input	Reset: Load PV value to CV. (Counter newstart)
	PV: INT ,	input	Downcounter start value: The value from which the counter counts down. It can be reloaded with the LD parameter.
	Q: BOOL ,	output	Result: If the current counter value is less than or equal to zero, the parameter is set to signal state "TRUE".
	CV: INT / CHAR / WCHAR / DATE	output	Current counter value
);			

Example

```

9  □ #secTimer(IN:=NOT(#secTimerQ),
10     PT:=t#1s,
11     ET=>#secTimerET);
12
13  #secTimerQ := #secTimer.Q;
14
15  // secTact change state every second
16  □ IF #secTimerQ THEN
17     #secTact := NOT (#secTact);
18  END_IF;
19
20  // testCounter count down every 2 second (its works with positive edge
21  // Q will "true" at 0
22  // CV cannot be less than 0 because it is an unsigned SINT counter (CTD_USINT)
23  □ #testCounter(CD:=#secTact,
24     LD:=NOT(#init),
25     PV:=100,
26     Q=>#testCounterQ,
27     CV=>#testCounterInt);
28
29  #init := TRUE;
30
31  // actual state from testCounter
32  #tempInt := #testCounterInt;
    
```

#secTimerQ	FALSE
#secTimerET	T#952MS
#secTimerQ	FALSE
Result	FALSE
#secTact	TRUE
#secTact	TRUE
#init	TRUE
#init	TRUE
#testCounterQ	TRUE
#testCounterInt	0
#tempInt	0



The example program below can be downloaded here:

ctd_example.scl



Here is a description of how to import the downloaded program to the TIA Portal:
[Import source code to the TIA portal.](#)

>> [Back to LamaPLC main menu \(SCL commands\)](#)

CTUD: Count up

The function that counts up- and downwards.

The CTUD counts up the positive edges of the “CU” input, and counts down with “CD” input. As soon as it reaches the value of “PV” with the calculation, the output “QU” changes to “TRUE”. The function continues to count. The function can be restarted with the “R” input. The downcounts function can be restarted with “LD” input, in this case, the countdown starts again from the “PV” value. If the current counter value is less than or equal to zero, the “QD” parameter is set to signal state “TRUE”.

CTUD types:

- CTUD_SINT / CTUD_USINT: the calculation goes with the variable types [SINT](#), [USINT](#)
- CTUD_INT / CTUD_UINT: the calculation goes with the variable types [INT](#), [UINT](#)
- CTUD_DINT / CTUD_UDINT: the calculation goes with the variable types [DINT](#), [UDINT](#)
- CTUD_LINT / CTUD_ULINT: the calculation goes with the variable types [LINT](#), [ULINT](#) (only on S7-1500 PLC)

FB			
CTUD			
(
	CU: BOOL , input		Counter up signal: The function monitors the positive edge of this binary signal. The counter counted up, if signal change from “FALSE” to “TRUE”.
	CD: BOOL , input		Counter down signal: The function monitors the positive edge of this binary signal. The counter downcounted, if signal change from “FALSE” to “TRUE”.
	R: BOOL , input		Reset upcounter: Reset upcounter CV value.
	LD: BOOL , input		Reset downcounter: Load PV value to CV. (Downcounter newstart)
	PV: INT , input		Set value to Q / startpoint for downcounter Value at which the Q output is set / The value from which the counter counts down. It can be reloaded with the LD parameter.
	QU: BOOL , output		Result: When the counter reaches “PV”.
	QD: BOOL , output		Result: If the current counter value is less than or equal to zero, the parameter is set to signal state “TRUE”.

CV: INT / CHAR / WCHAR / DATE	output	Current counter value
);		

>> [Back to LamaPLC main menu \(SCL commands\)](#)

Timer operations

IEC timers

IEC timers (TP, TON, TOF) should definitely be called embedded from the FBs, the description of which can be found here: [Call FBs](#)

In the case of embedding, the functions must be called from the “static” block of the calling FB (just type “TON” for the “data type”):

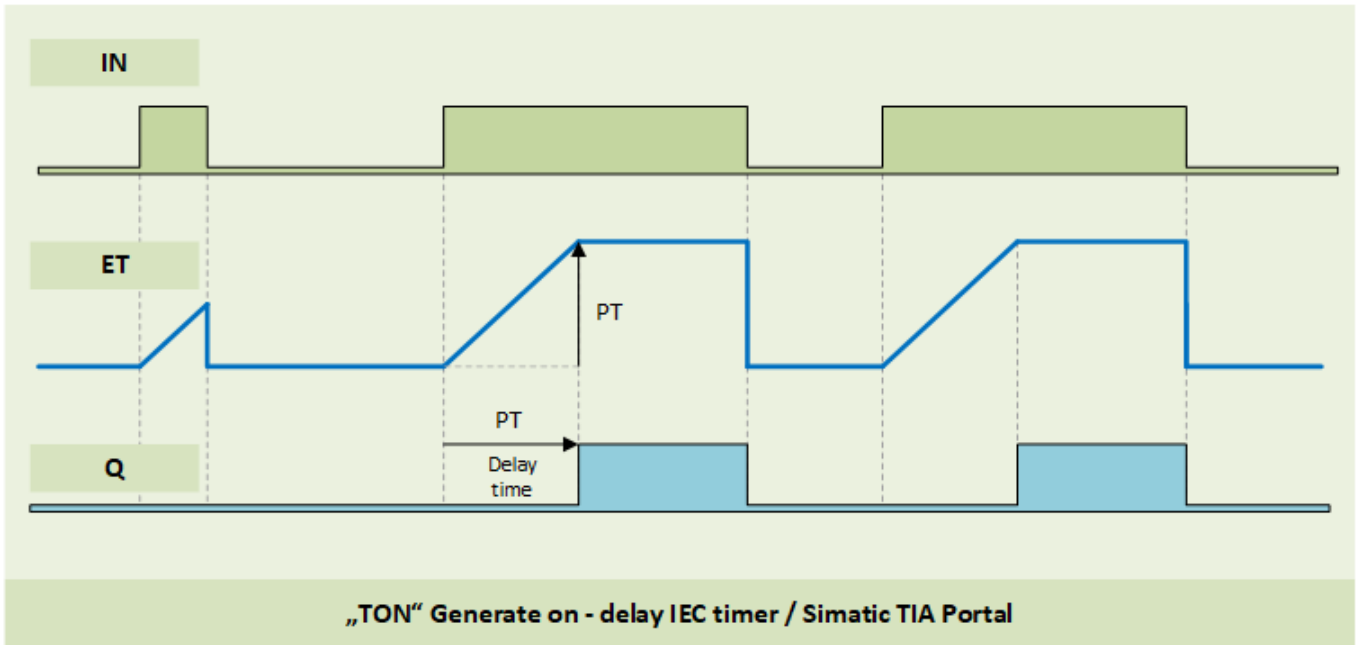
<div data-bbox="113 898 1230 1182"> <ul style="list-style-type: none"> ▼ Static ▀ ▶ time_1 TON_TIME Non-retain <input checked="" type="checkbox"/> ▀ ▶ time_2 TON_TIME Non-retain <input checked="" type="checkbox"/> ▼ Temp ▀ start Bool <input type="checkbox"/> ▀ <Add new> ▼ Constant </div>	<p>In the case of embedding, the functions must be called from the “static” block of the calling FB (just type “TON” for the “data type”. FBs must be called in the program in much the same way, of course the DB call must be omitted.</p>																																																					
<div data-bbox="113 1189 1230 1653"> <p>No condition defined.</p> <table border="1" style="width: 100%; border-collapse: collapse;"> <thead> <tr> <th>IF...</th> <th>CASE... OF...</th> <th>FOR... TO DO...</th> <th>WHILE.. DO...</th> <th>(*...*)</th> <th>REGION</th> </tr> </thead> <tbody> <tr> <td>1</td> <td></td> <td></td> <td></td> <td></td> <td></td> </tr> <tr> <td>2</td> <td colspan="5">▀ #time_1 (IN:=#start,</td> </tr> <tr> <td>3</td> <td colspan="5"> PT:=t#12s);</td> </tr> <tr> <td>4</td> <td colspan="5"></td> </tr> <tr> <td>5</td> <td colspan="5">▀ #time_2 (IN := #start,</td> </tr> <tr> <td>6</td> <td colspan="5"> PT := t#24s);</td> </tr> <tr> <td>7</td> <td colspan="5"></td> </tr> <tr> <td>8</td> <td colspan="5"></td> </tr> </tbody> </table> </div>		IF...	CASE... OF...	FOR... TO DO...	WHILE.. DO...	(*...*)	REGION	1						2	▀ #time_1 (IN:=#start,					3	PT:=t#12s);					4						5	▀ #time_2 (IN := #start,					6	PT := t#24s);					7						8				
IF...	CASE... OF...	FOR... TO DO...	WHILE.. DO...	(*...*)	REGION																																																	
1																																																						
2	▀ #time_1 (IN:=#start,																																																					
3	PT:=t#12s);																																																					
4																																																						
5	▀ #time_2 (IN := #start,																																																					
6	PT := t#24s);																																																					
7																																																						
8																																																						

test_3_DB				
Name	Data type	Start value	Monitor value	Re
Input				
Output				
InOut				
Static				
time_1	TON_TIME			
PT	Time	T#0ms	T#0MS	
ET	Time	T#0ms	T#0MS	
IN	Bool	false	FALSE	
Q	Bool	false	FALSE	
time_2	TON_TIME			
PT	Time	T#0ms	T#0MS	
ET	Time	T#0ms	T#0MS	
IN	Bool	false	FALSE	
Q	Bool	false	FALSE	

The image shows the data of the embedded FBs in the IDB of the calling FB. The great advantage of this method is that the program is easy to import / export / copy, as you only need one FB definition.

TON: Generate on-delay

The TON “switch-on delay” function. It monitors the positive edge of the binary signal at the “IN” input. If the signal persists for a specified “PT” time, it outputs “Q”. The timer is most commonly used to delay signals, for example: - in the case of a field signal, we want to make sure that not only one interference signal is received - in case of level signals we want to get a certain signal (many level signals “wobble”)



„TON“ Generate on - delay IEC timer / Simatic TIA Portal

lamaplc.com

FB	TON (
	IN: BOOL ,	input	Start signal: The function monitors the positive edge of this binary signal. The timer start, if signal is “TRUE”.
	PT: TIME ,	input	Delay time: The delay time, example: t#12s
	Q: BOOL ,	output	Result: When the time is up, the result will be “TRUE”.

ET: TIME	output	Elapsed time
-----------------	--------	---------------------

);

>> [Back to LamaPLC main menu \(SCL commands\)](#)

TON Examples

CALL TON

▼ Static				<input type="checkbox"/>
▀ time_1	TON_TIME		Non-retain	<input checked="" type="checkbox"/>
▀ time_2	TON_TIME		Non-retain	<input checked="" type="checkbox"/>
▀ time_3	TON_TIME		Non-retain	<input checked="" type="checkbox"/>
▼ Temp				<input type="checkbox"/>

```

1
2
3 // Embedded timer, minimum format
4 #time_1(IN:=#start,
5     PT:=t#12s);
6
7 // Embedded timer, full format
8 #time_2(IN := #start,
9     PT := t#24s,
10    Q=>#q,
11    ET=>#et);
12
13 // Embedded timer, full format with EN and ENO bits
14 #time_3(EN := #en_1,
15     IN := #start,
16     PT := t#24s,
17     Q => #q,
18     ET => #et,
19     ENO => #eno_1);
20
21 // Timer with DB, minimum format
22 "IEC_Timer_0_DB".TON(IN := #start,
23     PT := t#12s);
24

```

TON second impuls

The example program below can be downloaded here:



test_ton.scl

Here is a description of how to import the downloaded program to the TIA Portal:



[Import source code to the TIA portal.](#)

```
// author OB121 / Sandor Vamos
// ob121.com; 2022.04.13.
// TON example: sec impuls
//
// More information:
// https://www.ob121.com/doku.php?id=de:s7:scl_reference_timers
//
// FB "static" variables:
// secTakt: TON_TIME
// q, secChange: BOOL
// secCount: INT
// FB "temp" variables:
// tempInt: INT
// tempBool: BOOL

// TON, as sec impuls generator
#secTakt(IN:=NOT(#q),
        PT:=t#1s);

// For example, the "Q" parameter can be called
// directly when calling "secTakt",
// OR it can be referred to as:
#q := #secTakt.Q;

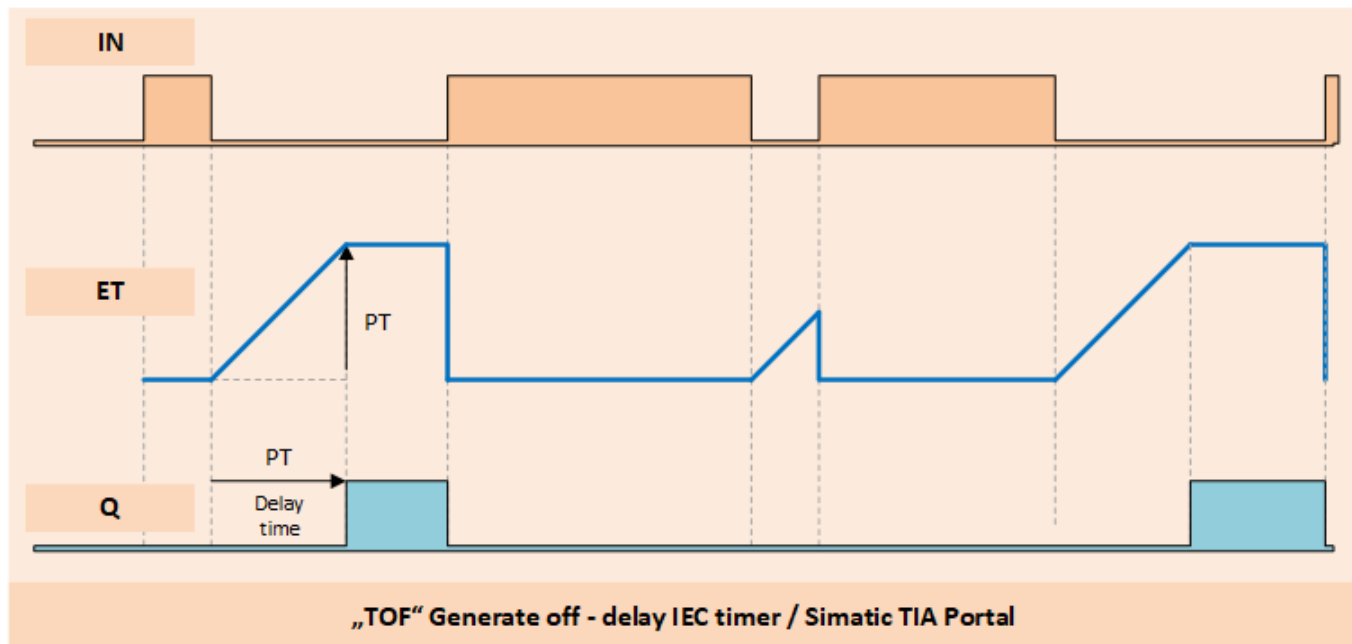
// when the time has elapsed, the block will be called:
// - secChange changes its status every second
// - secCount increases every second, in the range 0..9
IF #q THEN
    #secChange := NOT (#secChange);
    #secCount := #secCount + 1;
    IF #secCount > 9 THEN
        #secCount := 0;
    END_IF;
END_IF;

// monitor
#tempInt := #secCount;
#tempBool := #secChange;
```

[>> Back to LamaPLC main menu \(SCL commands\)](#)

TOF: Generate off-delay

The TOF “switch-off delay” function. It monitors the positive edge of the binary signal at the “IN” input. If the signal persists for a specified “PT” time, it outputs “Q”. The timer is most commonly used to delay signals, for example: - in the case of a field signal, we want to make sure that not only one interference signal is received - in case of level signals we want to get a certain signal (many level signals “wobble”)



lamaplc.com

FB			
TOF (
	IN: BOOL , input		Start signal: The function monitors the positive edge of this binary signal. The timer start, if signal is “TRUE”.
	PT: TIME , input		Delay time: The delay time, example: t#12s
	Q: BOOL , output		Result: When the time is up, the result will be “TRUE”.
	ET: TIME output		Elapsed time
);			

TOF Examples

Static			
time_1	TOF_TIME		Non-retain
time_2	TOF_TIME		Non-retain
time_3	TOF_TIME		Non-ret...

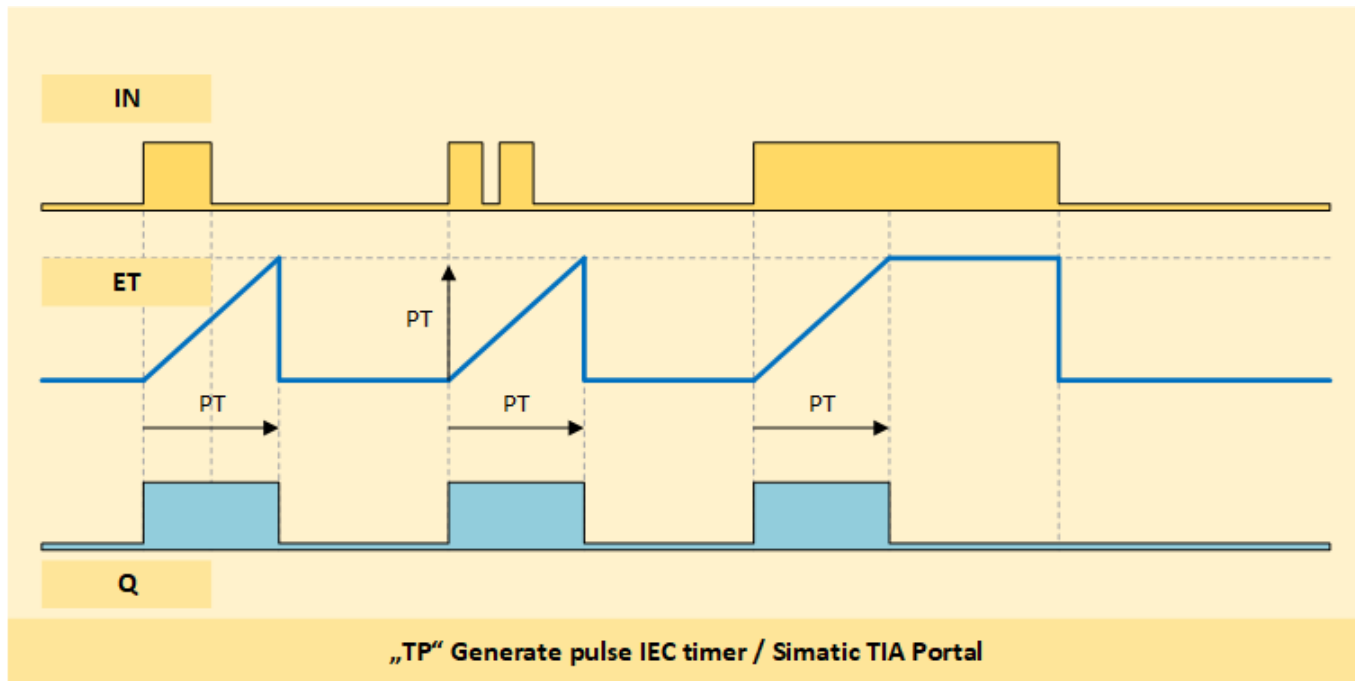
IF...	CASE... OF...	FOR... TO DO...	WHILE... DO...	(*...*)	REGION
-------	---------------	-----------------	----------------	---------	--------

```
1
2
3 // Embedded timer, minimum format
4 #time_1(IN:=#start,
5     PT:=t#12s);
6
7 // Embedded timer, full format
8 #time_2(IN := #start,
9     PT := t#24s,
10    Q=>#q,
11    ET=>#et);
12
13 // Embedded timer, full format with EN and ENO bits
14 #time_3(EN := #en_1,
15     IN := #start,
16     PT := t#24s,
17     Q => #q,
18     ET => #et,
19     ENO => #eno_1);
20
21 // Timer with DB, minimum format
22 #IEC_Timer_0_DB".TOF(IN := #start,
23     PT := t#12s);
24
25
26
```

[>> Back to LamaPLC main menu \(SCL commands\)](#)

TP: Generate pulse

The TP “pulse generator” function. It monitors the positive edge of the binary signal at the “IN” input, and generates pulses in effect equal “PT” length from this. It is commonly used to extend very short-lived signals because, for example, the cycle time of a WinCC HMI is typically 1 sec. If the signals are “extended” for 2 seconds, the HMI can also process them.



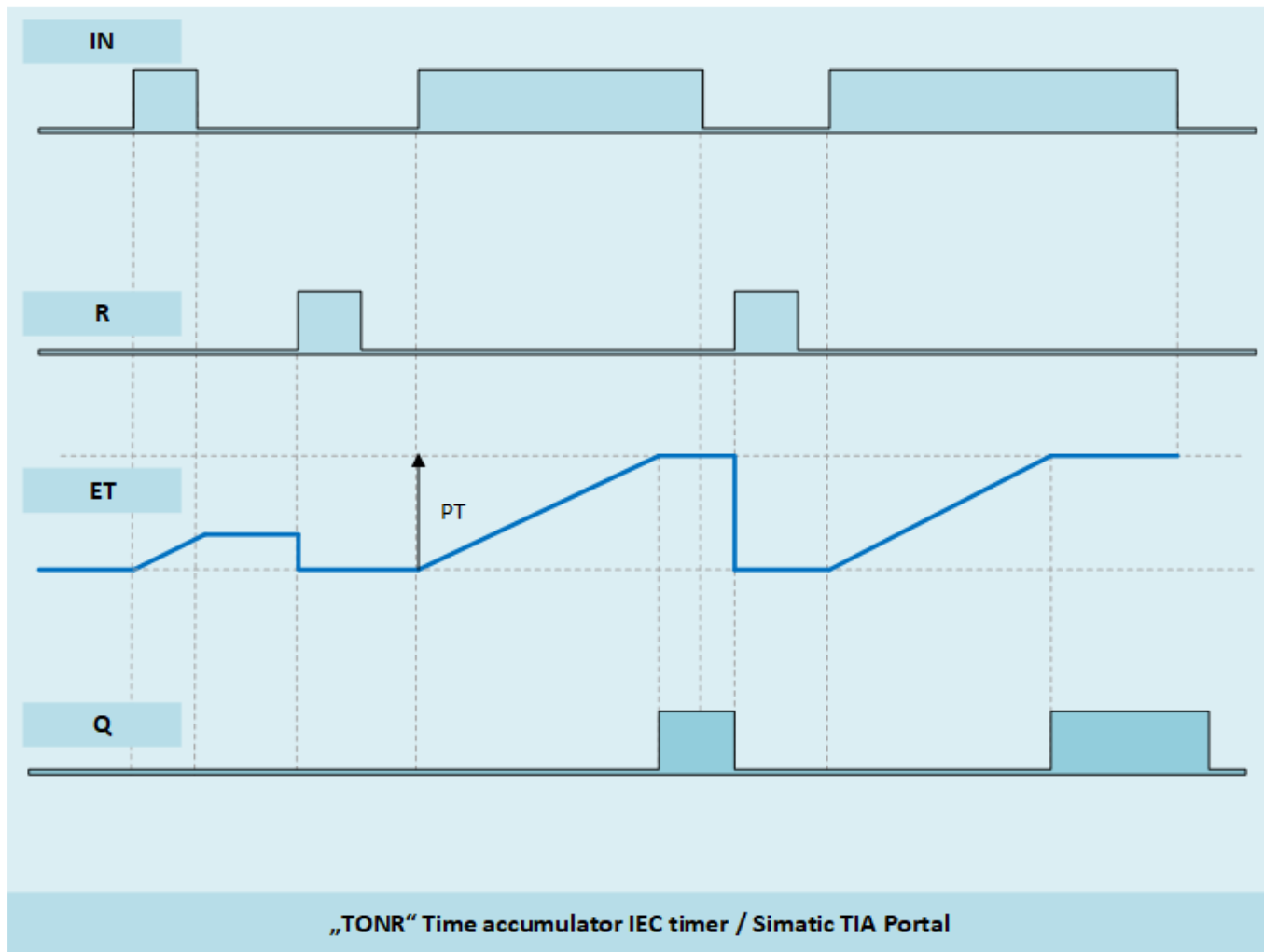
lamaplc.com

FB TP			
(
	IN: BOOL , input		Start signal: The function monitors the positive edge of this binary signal. The timer start, if signal is first time "TRUE".
	PT: TIME , input		Extending time: example: t#12s
	Q: BOOL , output		Result: "Q" will be TRUE for the duration of the signal extension.
	ET: TIME output		Elapsed time
);			

>> [Back to LamaPLC main menu \(SCL commands\)](#)

TONR: Time accumulator

The TONR "time accumulator" function. It practically collects the times, and if the input signal "IN" has already existed for the right amount of time, it indicates a result "Q". The timer function can be reset with the reset input "R" to set null the amount of time collected.



„TONR“ Time accumulator IEC timer / Simatic TIA Portal

lamaplc.com

FB			
TONR (
	IN: BOOL , input		Start signal: The function monitors the positive edge of this binary signal. The timer running, if signal is "TRUE".
	R: BOOL , input		Reset signal: The timer function can be reset with the reset input "R" to set null the amount of time collected.
	PT: TIME , input		Extending time: example: t#12s
	Q: BOOL , output		Result: After collecting the "PT" amount of time, the "Q" output will be "TRUE". The function can then be restarted with the "R" input.
	ET: TIME output		Elapsed / collected time
);

[Simatic](#), [SCL](#), [TIA](#), [Math](#), [CTU](#), [CTD](#), [CTUD](#), [IEC timers](#), [TP](#), [TON](#), [TOF](#), [counter](#), [timer](#)

This page has been accessed for: Today: 3, Until now: 221

From:

<https://www.lamaplc.com/> - **lamaPLC**

Permanent link:

https://www.lamaplc.com/doku.php?id=simatic:scl_commands_timer_counter

Last update: **2026/04/21 20:46**

