

**EURO**PROT +

**Factory configured macros  
for the user logic**



**Document ID: VERSION 1.0**  
Budapest, November 2011.

User's manual version information

<b>Version</b>	<b>Date</b>	<b>Modification</b>	<b>Compiled by</b>
Version 1.0	11.11.2011.	First edition	Petri

## CONTENTS

1	User logic in the EuroProt+ devices.....	4
1.1	Application of the user logic.....	4
2	The factory defined macros.....	5
2.1	The list of the factory defined macros.....	5
2.1.1	Extended AND gates .....	6
2.1.2	Extended OR gates .....	6
2.1.3	NAND gate.....	7
2.1.4	NOR gate.....	8
2.1.5	XOR gate .....	8
2.1.6	XNOR gate .....	9
2.1.7	RS flip-flop .....	10
2.1.8	Non volatile NVRS flip-flop .....	10
2.1.9	DQ (D) flip-flop (DQ).....	11
2.1.10	Non volatile DQ (D) flip-flop (NVDQ).....	12
2.1.11	REdge Rising edge detection .....	14
2.1.12	FEdge Falling edge detection .....	14
2.1.13	Application of the general time “Timer “.....	15
2.1.14	Timer macros.....	23
2.1.15	State Superv for disconnecter or circuit breaker state supervision .....	26

# 1 User logic in the EuroProt+ devices

## 1.1 Application of the user logic

The application of the user logic is described in the document “**EuroCAP configuration tool for the EuroProt+ devices**”.

The logic belonging to the function blocks are factory edited; here the user has the possibility to extend the functionality of the device with “Programmable Logic Controller (PLC)”-like functions.

When opening the Logic Editor, the toolbox offers the available elements to be involved in the user logic (See the marked part of Figure 1-1.)

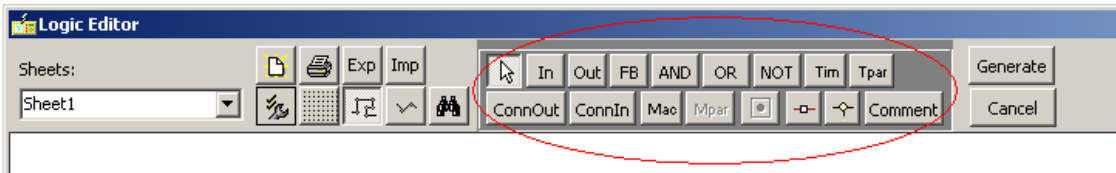






Figure 1-1 The available tools of the Logic Editor

The Logic Editor is a powerful tool available for the user to compose user logic, based on the binary signals of a factory configuration.

The available tools are symbolized with icons as follows:

- **In** The inputs of the user logic can be:
  - Non-filtered optocoupler inputs
  - Optocoupler inputs
  - Graphed input statuses
  - Logic parameter variables
  - Matrix columns
- **Out** The outputs of the user logic can be:
  - Contacts
  - Graphed output statuses
- **FB** The function blocks configured in the device can be placed on the sheet and the related binary inputs and outputs can be included in the user logic.
- **AND OR NOT** Selection of simple AND, OR, NOT “gates” for processing the binary signals.
- **ConnOut ConnIn** These icons are used to transfer signals from one sheet to another.
- **Tim Tpar** Selection of timers and related timer parameters for processing the binary signals.
-  This icon inserts a new node in the signal flow.
-  This icon inserts a new junction in the signal flow.
- **Mac Mpar** These icons serve the purpose of processing “macros” (pre-defined parts of the logic chart) to simplify the most common operations.
-  Macro compilation if the MACRO\_EDIOR sheet is active.

The individual elements can be placed on the worksheet with the usual drag-and-drop techniques. The connections among the elements are edited with interconnecting lines.

After editing, the logic diagram must be compiled using the  button; the results are the generated logic equations to be processed by the processor, but invisible for the user.

This description explains the application of pre-defined macros.

## 2 The factory defined macros

The “macros” are elements of the user logic, which are composed of basic elements of the toolbox. These basic elements are simple AND, OR, NOT “gates” for processing the binary signals, timers, and other macros as well.

The aim of the macros is to simplify the editing procedure performed by the user, offering edited parts of the user logic, which can be applied repeatedly, or offering pre-tested parts which perform sophisticated tasks in the user logic.

### 2.1 The list of the factory defined macros

The list of the factory defined macros is extending continuously. This description explains the macros available at the date of compiling this document.

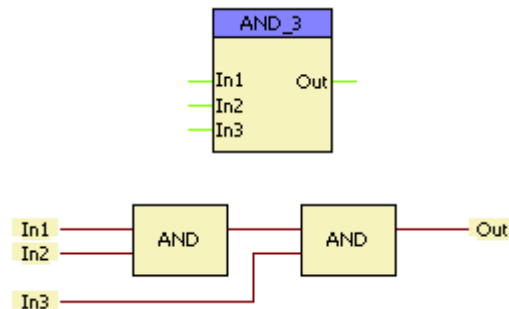
- AND gates with extended number of input signals (AND\_2, AND\_3, AND\_4, AND\_5, AND\_6, AND\_7, AND\_8)
- OR gates with extended number of input signals (OR\_2, OR\_3, OR\_4, OR\_5, OR\_6, OR\_7, OR\_8, OR\_9)
- NAND gate
- NOR gate
- XOR gate
- XNOR gate
- RS flip-flop
- DQ flip-flop
- NVRS flip-flop
- NVDQ flip-flop
- REdge Rising edge detection
- FEdge Falling edge detection
- PickDly timer with pick-up delay
- DropDly timer with drop-off delay
- IMP impulse generation
- StateSuperv status supervision

## 2.1.1 Extended AND gates

AND\_2, AND\_3, AND\_4, AND\_5, AND\_6, AND\_7, AND\_8

These are pre-edited macros composed of simple AND gates. The number of configured inputs is indicated by the name. As an example the AND\_3 is an AND gate using three input signals. The output is true only if all input signals are in true state.

The symbol and the logic diagram of an extended AND gate are as follows:



*Figure 2-1 Example: The symbol and the logic scheme of an AND gate with three inputs*

The Truth Table of an extended AND gate is as follows:

Inputs			Output
In1	In2	In3	Out
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	0
1	0	1	0
1	1	0	0
1	1	1	1

*Table 2-1 The Truth Table of an extended AND gate*

AND gates with further input extensions are constructed similarly.

## 2.1.2 Extended OR gates

OR\_2, OR\_3, OR\_4, OR\_5, OR\_6, OR\_7, OR\_8, OR\_9

These are pre-edited macros composed of simple OR gates. The number of configured inputs is indicated by the figure in the name. As an example the OR\_3 is an OR gate using three input signals. The output is false only if all input signals are in false state.

The symbol and the logic diagram of an extended OR gate are as follows:

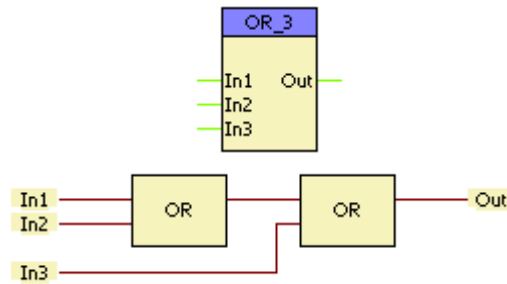


Figure 2-2 Example: The symbol and the logic scheme of an OR gate with three inputs

The Truth Table of an extended OR gate is as follows:

Inputs			Output
In1	In2	In3	Out
0	0	0	0
0	0	1	1
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	1
1	1	0	1
1	1	1	1

Table 2-2 The Truth Table of an extended OR gate

OR gates with further input extensions are constructed similarly.

### 2.1.3 NAND gate

This is an AND gate with the output inverted. (NAND = Not AND)

The output is true if input In1 AND input In2 are NOT both true:  $Out = NOT(In1 AND In2)$

The symbol and the logic diagram of a NAND gate are as follows:

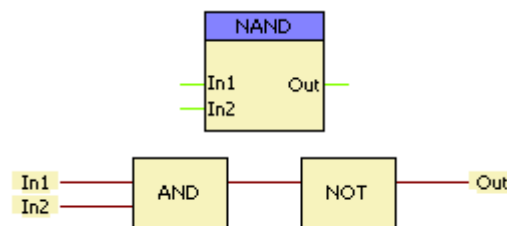


Figure 2-3 The symbol and the logic scheme of a NAND gate

The Truth Table of a NAND gate is as follows:

Inputs		Output
In1	In2	Out
0	0	1
0	1	1
1	0	1
1	1	0

Table 2-3 The Truth Table of a NAND gate

### 2.1.4 NOR gate

This is an OR gate with the output inverted. (NOR = Not OR)

The output is true if NOT input In1 OR input In2 are true:  $Out = NOT(In1 OR In2)$

The symbol and the logic diagram of a NOR gate are as follows:

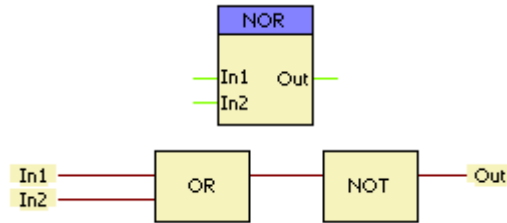


Figure 2-4 The symbol and the logic scheme of a NOR gate

The Truth Table of a NOR gate is as follows:

Inputs		Output
In1	In2	Out
0	0	1
0	1	0
1	0	0
1	1	0

Table 2-4 The Truth Table of a NOR gate

### 2.1.5 XOR gate

In this Exclusive OR (XOR) gate the output is true if the inputs are different. This gate can have only two input signals.

The output is true if input In1 OR input In2 is true, but not when both of them are true:  
 $Out = (In1 AND NOT In2) OR (NOT In1 AND In2)$

The symbol and the logic diagram of an XOR gate are as follows:

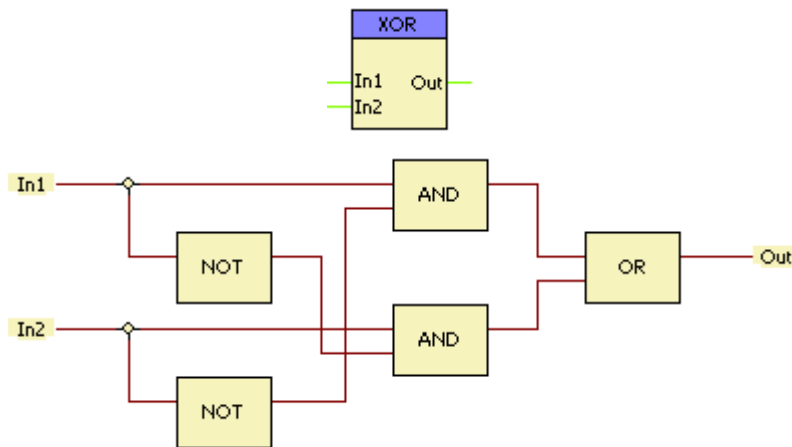


Figure 2-5 The symbol and the logic scheme of an XOR gate



The Truth Table of an XOR gate is as follows:

Inputs		Output
In1	In2	Out
0	0	0
0	1	1
1	0	1
1	1	0

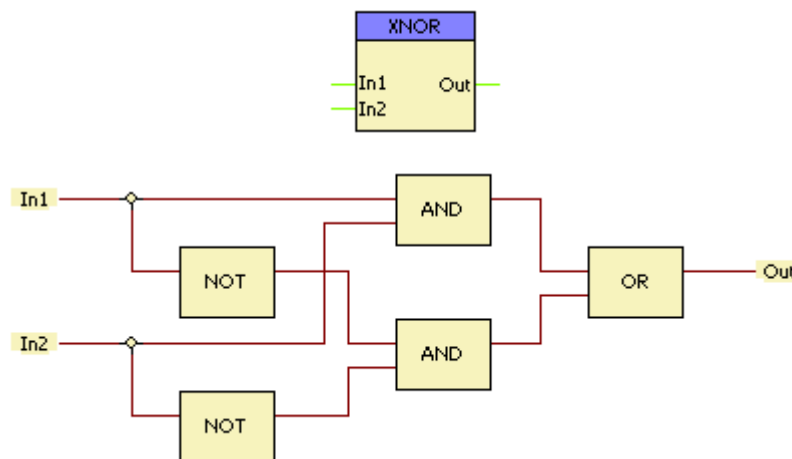
*Table 2-5 The Truth Table of an XOR gate*

### 2.1.6 XNOR gate

In this Exclusive NOR gate the output is true if the inputs are the same (both true or both false). Basically this gate is an inverted XOR gate. Also this gate can have only two input signals.

The output is true if input In1 AND input In2 is true, or input In1 AND input In2 is false:  
 $Out = (In1 \text{ AND } In2) \text{ OR } (\text{NOT } In1 \text{ AND } \text{NOT } In2)$

The symbol and the logic diagram of a XNOR gate are as follows:



*Figure 2-6 The symbol and the logic scheme of an XNOR gate*

The Truth Table of an XNOR gate is as follows:

Inputs		Output
In1	In2	Out
0	0	1
0	1	0
1	0	0
1	1	1

*Table 2-6 The Truth Table of an XNOR gate*

## 2.1.7 RS flip-flop

An RS flip-flop is the simplest possible memory element.

The inputs R and S are referred to as the Reset and Set inputs, respectively. The active state of the S input sets the Q output to true state. The active state of the R input resets the status of Q to false.

The symbol and the logic diagram of an RS flip-flop are as follows:

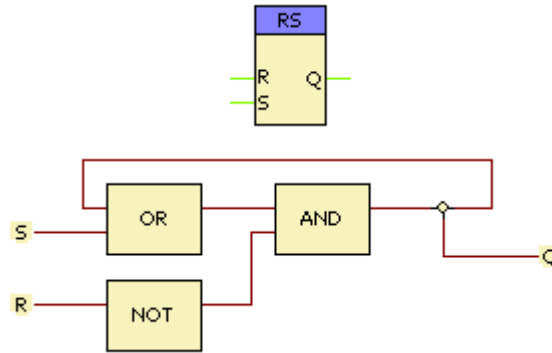


Figure 2-7 The symbol and the logic scheme of an RS flip-flop

The Truth Table of an RS flip-flop is as follows:

Inputs		Output	Comment
R	S	Q	
0	0	Q	Hold state
0	1	1	Set
1	0	0	Reset
1	1	(0)	Avoid

Table 2-7 The Truth Table of an RS flip-flop

## 2.1.8 Non volatile NVRS flip-flop

This is the non-volatile version of the simple RS flip-flop. If the power supply of the device is switched off and on again then the value of Q, as it was assigned before switching off, is preserved till the subsequent set or reset operation.

The inputs R and S are referred to as the Reset and Set inputs, respectively. The active state of the S input sets the Q output to true state. The active state of the R input resets the status of Q to false.

The symbol and the logic diagram of an NVRS flip-flop are practically the same as those of the RS flip-flop:

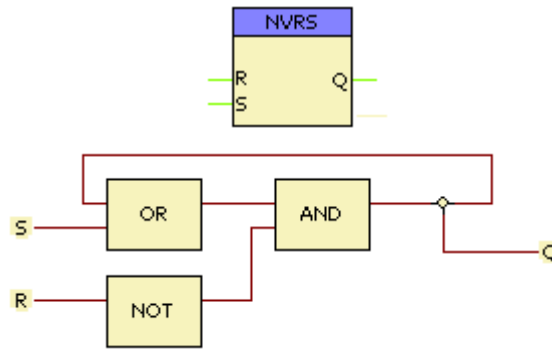


Figure 2-8 The symbol and the logic scheme of a (non volatile) RS flip-flop

The Truth Table of an NVRS flip-flop is the same as that of the RS flip-flop:

Inputs		Output	Comment
R	S	Q	
0	0	Q	Hold state
0	1	1	Set
1	0	0	Reset
1	1	(0)	Avoid

Table 2-8 The Truth Table of a (non volatile) RS flip-flop

The only difference is that here the output status value (Q) is stored in the non-volatile memory. If the power supply is switched off and then on again, the value is kept until the subsequent valid S or R status change.

### 2.1.9 DQ (D) flip-flop (DQ)

The D flip-flop is the most common flip-flop in use today. It is better known as *data* or *delay* flip-flop (as its output Q looks like a delay of input D).

The Q output takes on the state of the D input at the moment of a positive edge at the CLK (clock) input. It is called the D flip-flop for this reason, since the output takes the value of the D input or *data* input, and keeps it for one clock cycle. The D flip-flop can be interpreted as a primitive memory cell. Whenever the clock pulses, the value of D is assigned to the Q output, and the value is kept for one clock cycle.

The symbol and the logic diagram of a D flip-flop are as follows:

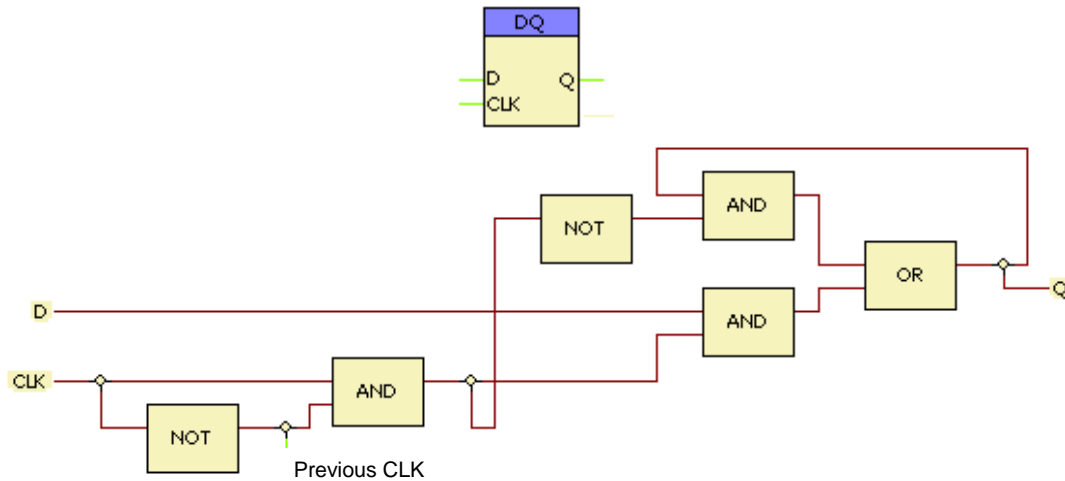


Figure 2-9 The symbol and the logic scheme of a DQ (D) flip-flop

The rising edge of the CLK signal is detected if the state is logic 1 and the previous state is logic 0.

At that moment Q gets the state of D. (NOTE: the previous state is kept by arranging the sequence of the evaluation of the gates!)

If no rising edge of the CLK signal is detected then Q output keeps the state until the next rising edge of the CLK signal.

The Truth Table of a D flip-flop is as follows:

Inputs		Output	Comment
CLK	D	Q	
At rising edge	0	0	Reset
At rising edge	1	1	Set
Any other state	x	Qprev	Keeps previous state

Table 2-9 The Truth Table of a DQ (D) flip-flop

### 2.1.10 Non volatile DQ (D) flip-flop (NVDQ)

This is the non-volatile version of the simple DQ (D) flip-flop. If the power supply of the device is switched off and on again then the value of Q, as it was assigned before switching off, is preserved till the subsequent rising edge of the CLK signal.

The D flip-flop is the most common flip-flop in use today. It is better known as *data* or *delay* flip-flop (as its output Q looks like a delay of input D).

The Q output takes on the state of the D input at the moment of a positive edge at the CLK (clock) input. It is called the D flip-flop for this reason, since the output takes the value of the D input or *data* input, and keeps it for one clock cycle. The D flip-flop can be interpreted as a primitive memory cell. Whenever the clock pulses, the value of D is assigned to the Q output, and the value is kept for one clock cycle.

The symbol and the logic diagram of a non volatile D flip-flop are as follows:

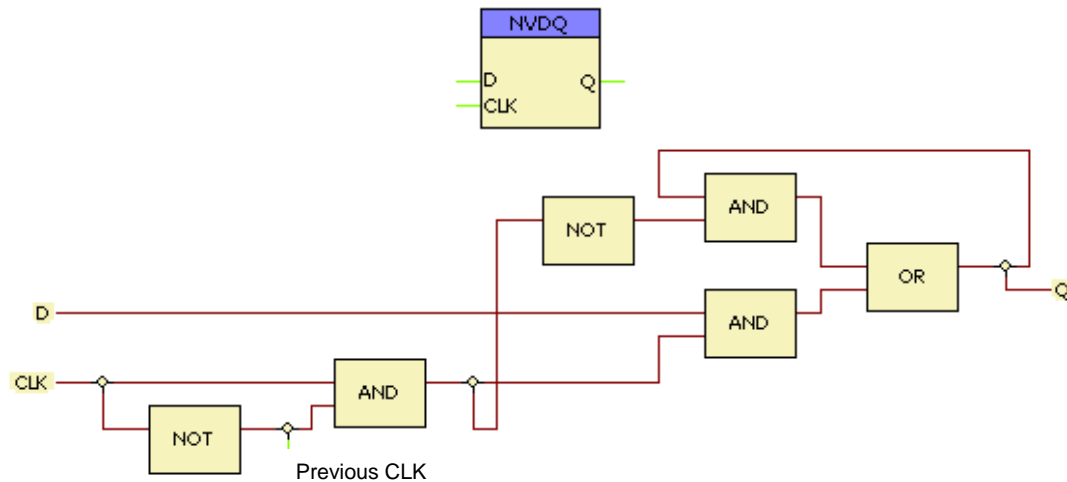


Figure 2-10 The symbol and the logic scheme of a (non volatile) DQ (D) flip-flop

The rising edge of the CLK signal is detected if the state is logic 1 and the previous state is logic 0.

At that moment Q gets the state of D. (NOTE: the previous state is kept by arranging the sequence of the evaluation of the gates!)

If no rising edge of the CLK signal is detected then Q output keeps the state until the next rising edge of the CLK signal.

The Truth Table of a D flip-flop is as follows:

Inputs		Output	Comment
CLK	D	Q	
At rising edge	0	0	Reset
At rising edge	1	1	Set
Any other state	x	Qprev	Keeps previous state

Table 2-10 The Truth Table of a (non volatile) DQ (D) flip-flop

### 2.1.11 REdge Rising edge detection

This macro can be used to detect the rising edge of the input signal. The output is a short (1 ms) pulse at the time of the rising edge.

The symbol and the logic diagram of a rising edge detector macro are as follows:

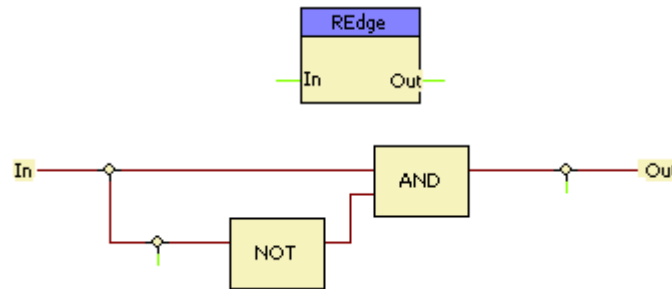


Figure 2-11 The symbol and the logic scheme of a rising edge detection

(NOTE: the previous state is kept by arranging the sequence of the evaluation of the gates!)

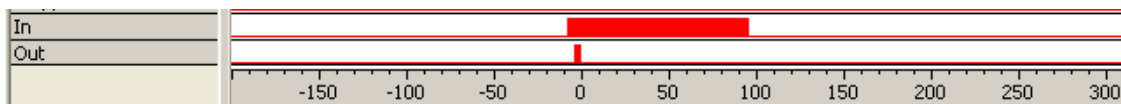


Figure 2-12 The time diagram of a rising edge detection

### 2.1.12 FEdge Falling edge detection

This macro can be used to detect the falling edge of the input signal. The output is a short (1 ms) pulse at the time of the falling edge.

The symbol and the logic diagram of a falling edge detector macro is as follows:

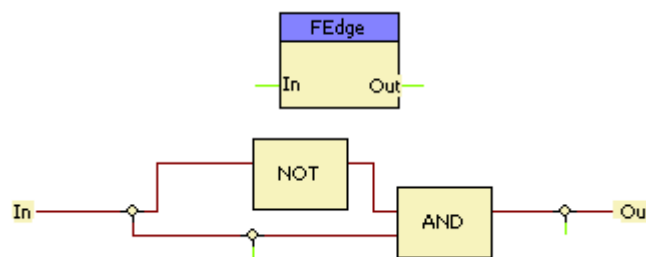


Figure 2-13 The symbol and the logic scheme of a falling edge detection

(NOTE: the previous state is kept by arranging the sequence of the evaluation of the gates!)

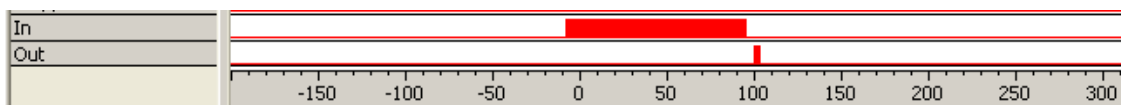


Figure 2-14 The time diagram of a rising edge detection

### 2.1.13 Application of the general “Timer “

The symbol of a general timer is as follows:

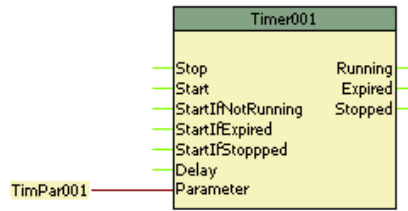


Figure 2-15 The symbol of a general timer

Table 2-11 shows the binary input status signals of the general timer.

Binary input signals	Explanation
Stop	This signal sets the timer in “Stopped” state
Start	This signal can start the timer
StartIfNotRunning	This signal can start the timer if it is not running, i.e. it is in “Expired” or in “Stopped” state
StartIfExpired	This signal can start the timer if it is in “Expired” state
StartIfStopped	This signal can start the timer if it is in “Stopped” state
Delay	The rising edge on this input starts the timer, resulting logic 1 status of the “Runing” output. The output status “Expired” is set only if the starting pulse is longer than the timer parameter. If the starting pulse resets then the “Stopped” output gets logic 1, and both other outputs reset.
Parameter	The choice of the parameter is explained below

Table 2-11 The binary input status signals of the timer

NOTE: The general rule is that only one kind of starting the timer (Start or StartIfNotRunning or StartIfExpired or StartIfStopped or Delay) may be used. If more than one starting condition is programmed to start the timer then the behavior of the timer depends also on the sequence of the generated logic equations. These details are not explained here.

The timer parameter can be either a constant value or a parameter can be assigned to the timer, according to the choice in the Figure below.

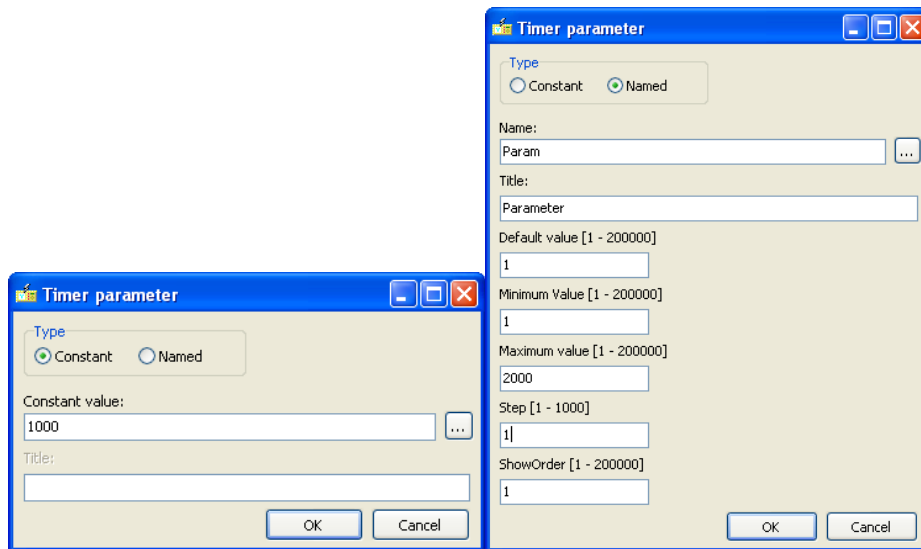


Figure 2-16 The timer parameter configuration

Table 2-12 shows the binary output status signals of the general timer.

Binary output signals	Explanation
Running	The output is logic 1 if the timer is running
Expired	The output is logic 1 if the timer is expired
Stopped	The output is logic 1 if the timer is neither running nor expired

Table 2-12 The binary output status signals of the timer

The following paragraphs explain the application of the inputs and outputs of the general timer.

(NOTE: for the most common timer applications macros are prepared to make the composing procedure of the user logic simpler. These timer macros are explained after the description of the general application.)

### 2.1.13.1 Using the “Start” input

The active state of the Start input resets the time counter and sets the state of the timer to “Running”, independently of the output status. The output status can be “Running” or “Expired” or “Stopped”.

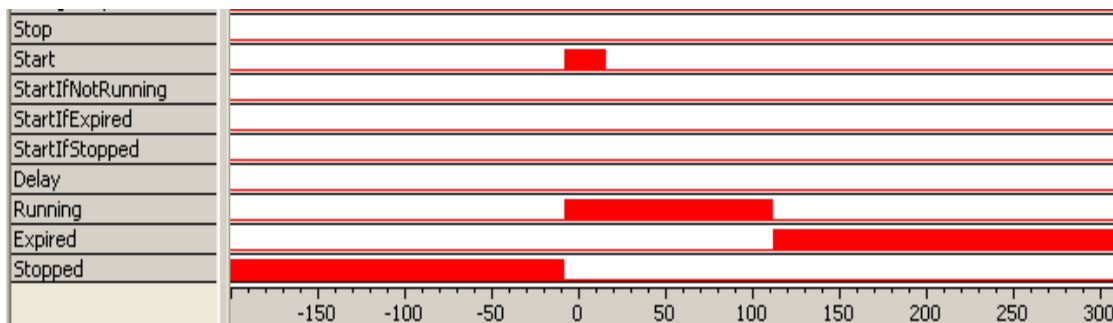


Figure 2-17 Starting the timer in “Stopped” state

The timer is in *Stopped* state.

Until the *Start* signal is in active state, it restarts the timer. The time measurement is started at the falling edge of the *Start* signal.



During the continuous restart and the time measurement state the *Running* output is active.  
 At timeout the *Running* state resets and the *Expired* state gets active.

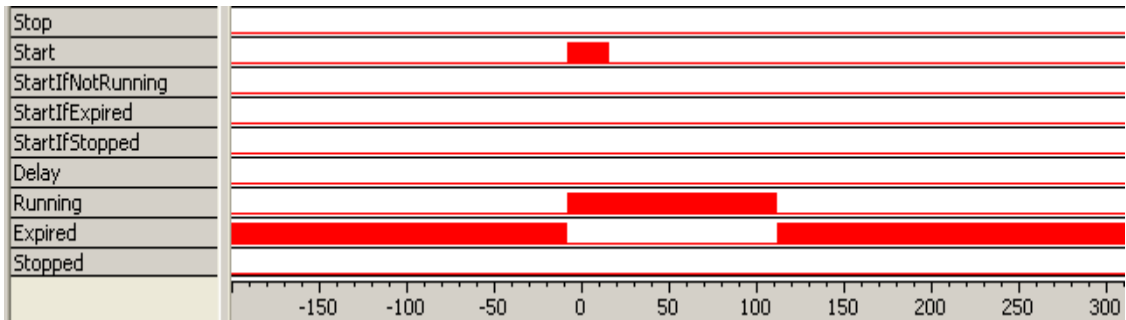


Figure 2-18 Starting the timer in “Expired” state

The initial state does not influence the operation using the “**Start**” input.

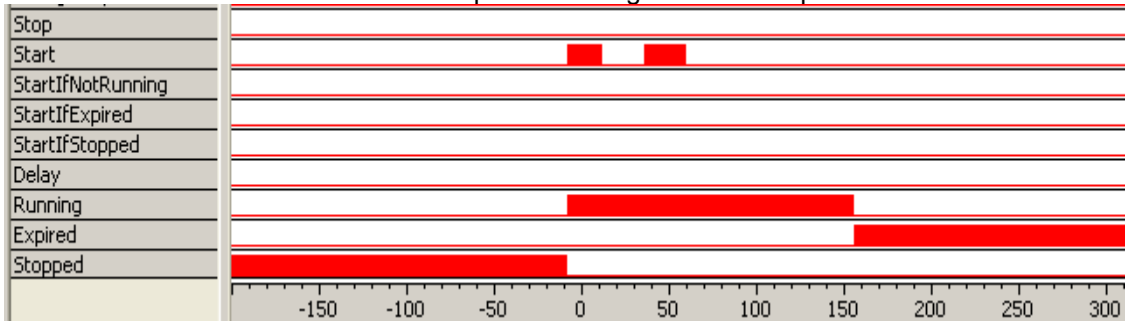


Figure 2-19 (Re)starting the timer in “Running” state

If the start impulse is received during the “Running” state then the time measurement is restarted at the falling edge of each subsequent “Start” impulse.

### 2.1.13.2 Using the “Stop” input

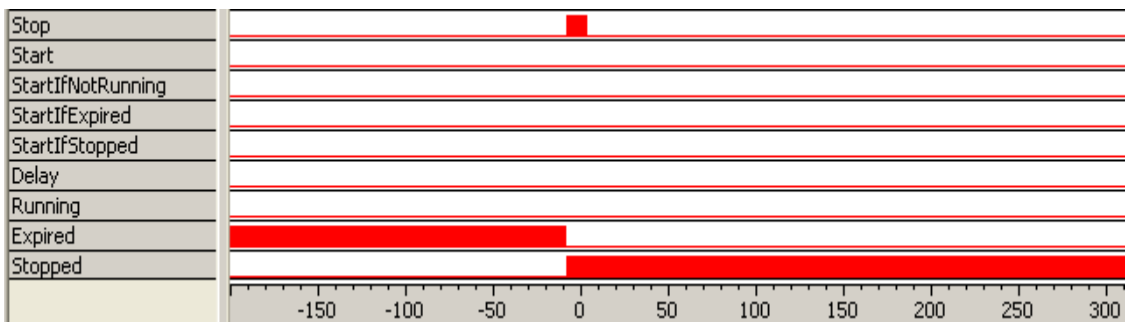


Figure 2-20 Using the “Stop” input of a timer in “Expired” state

The rising edge of the “Stop” signal resets the timer and the “Stopped” output gets in logic 1 state.

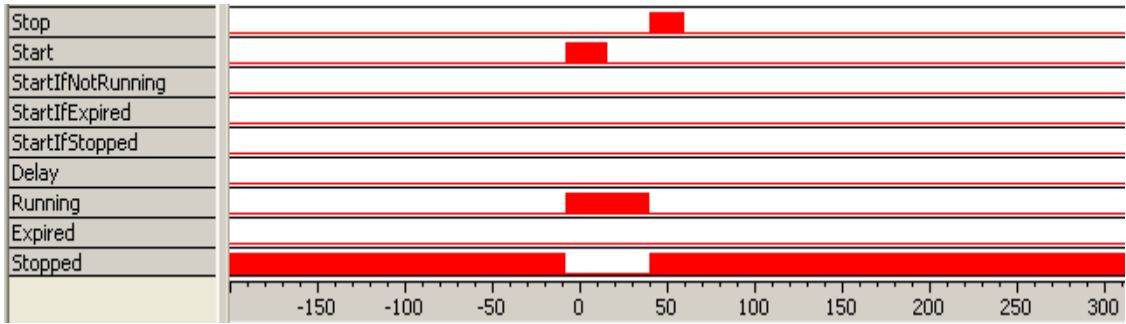


Figure 2-21 Using the “Stop” input of a timer in “Running” state

If the “Stop” signal is received during the running state of the timer then the time measurement stops immediately, the timer resets from the “Running” state and gets into “Stopped” state, and it never gets to “Expired” state.

### 2.1.13.3 Using the “StartIfNotRunning” input

The active state of the “StartIfNotRunning” input resets the time counter and sets the state of the timer to “Running”, if the output status is “Expired” or “Stopped”. In “Running” state the “StartIfNotRunning” input has no influence on the operation of the timer.

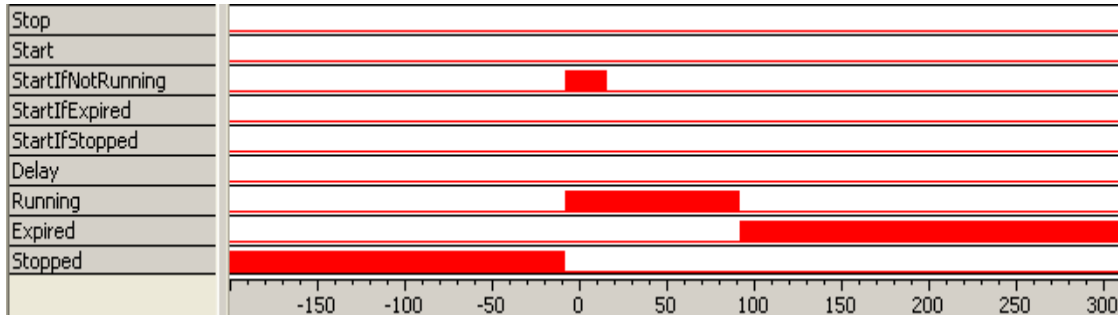


Figure 2-22 Using the “StartIfNotRunning” input of a timer in “Stopped” state

In “Stopped” state the starting with “StartIfNotRunning” input is active.

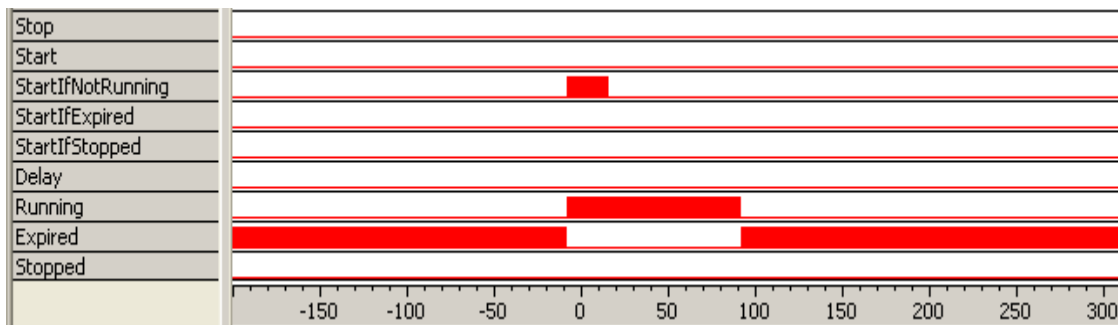


Figure 2-23 Using the “StartIfNotRunning” input of a timer in “Expired” state

In “Expired” state the starting with “StartIfNotRunning” input is active.

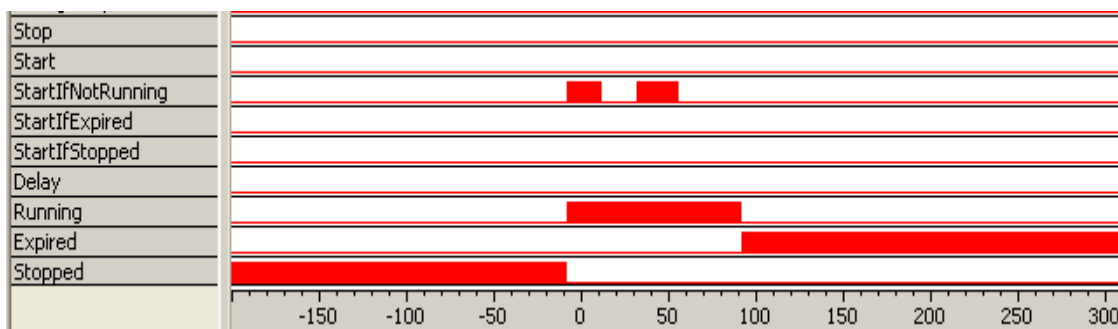


Figure 2-24 Using the “StartIfNotRunning” input of a timer in “Running” state

“StartIfNotRunning” in “Running” state has no influence on the operation of the timer, the running state is not prolonged.

### 2.1.13.4 Using the “StartIfExpired” input

The active state of the “StartIfExpired” input resets the time counter and sets the state of the timer to “Running”, if the output status is “Expired”. In “Running” or “Stopped” states the “StartIfNotRunning” input has no influence on the operation of the timer.

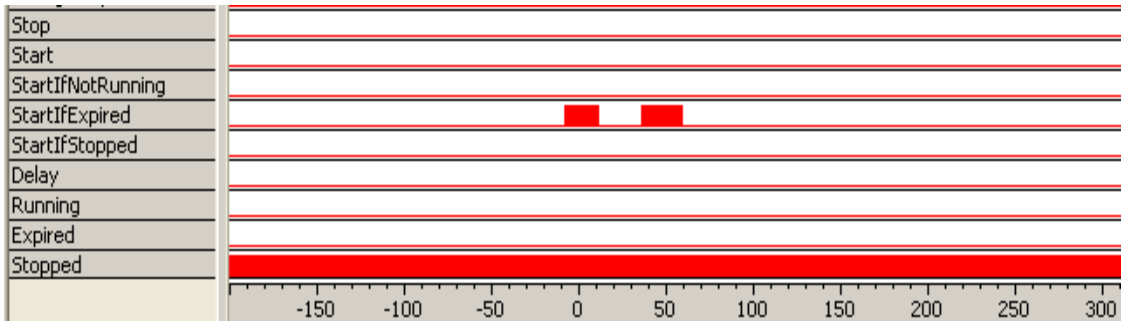


Figure 2-25 Using the “StartIfExpired” input of a timer in “Stopped” state

“StartIfExpired” input in “Stopped” state does not start the timer.

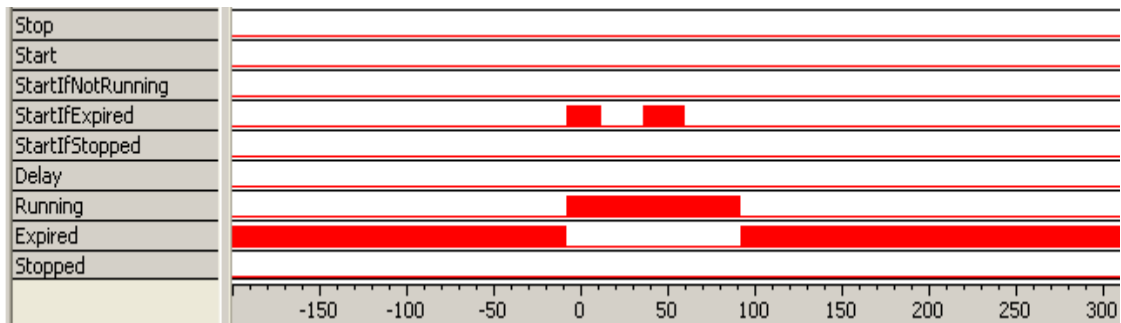


Figure 2-26 Using the “StartIfExpired” input of a timer in “Expired” state

“StartIfExpired” input in “Expired” state starts the timer, but the second pulse, which is received in “Running” state, has no influence.

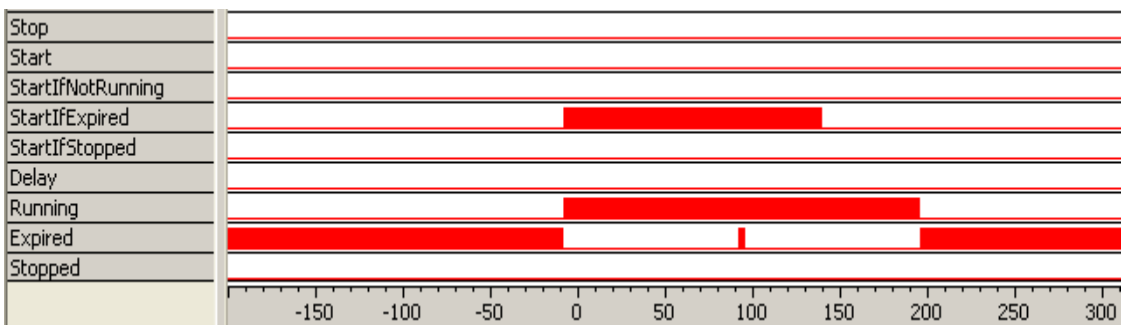


Figure 2-27 Using long pulse on the “StartIfExpired” input of a timer

If the starting pulse is longer than the parameter setting then after the first timeout the timer gets in “Expired” state, consequently the timer is restarted.

(NOTE: The ‘initial’ “Expired” state can be set only by using another starting input. Usually avoid using more than one starting input of a timer.)

### 2.1.13.5 Using the “StartIfStopped” input

The active state of the “StartIfStopped” input resets the time counter and sets the state of the timer to “Running”, if the output status is “Stopped”. In “Running” or “Expired” states the “StartIfStopped” input has no influence on the operation of the timer.

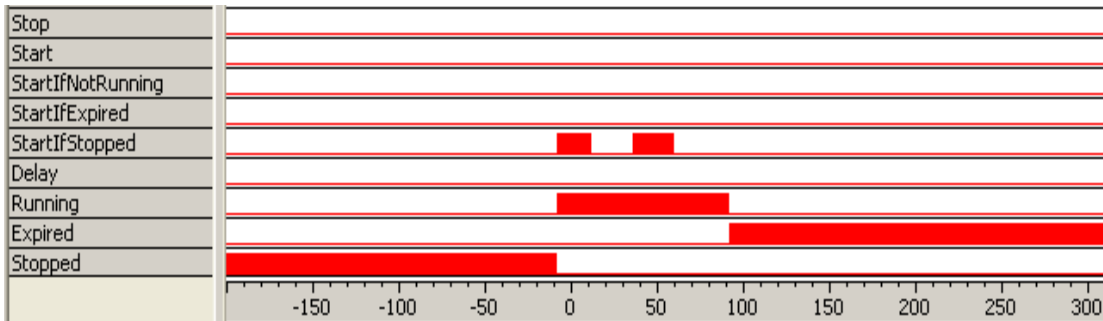


Figure 2-28 Using the “StartIfStopped” input of a timer in “Stopped” state

In “Stopped” state the “StartIfStopped” input starts the timer. The repeated start command has no influence, since the state at that time is already “Running”.

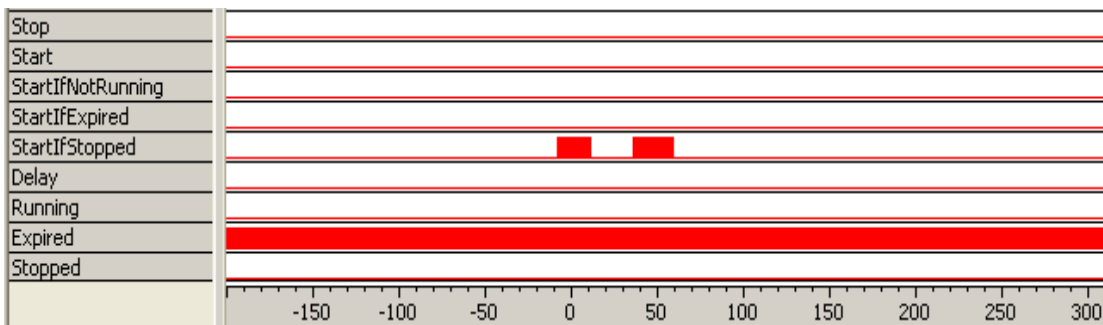


Figure 2-29 Using the “StartIfStopped” input of a timer in “Expired” state

In “Expired” state the “StartIfStopped” input cannot start the timer.

### 2.1.13.6 Using the “Delay” input

The joint application of the “Delay” input and the “Expired” output is the usual mode of operation of a timer. The “Delay” input is received from any protection functions as the signal “Protection started”. If the started state of the protection function is longer than the time delay setting then the status of the “Expired” output is applied as a TRIP command. If the circuit breaker opens and the fault is cleared then the “Protection started” signal resets on the “Delay” input. As a consequence the “Expired” state resets to zero and the TRIP command is withdrawn.

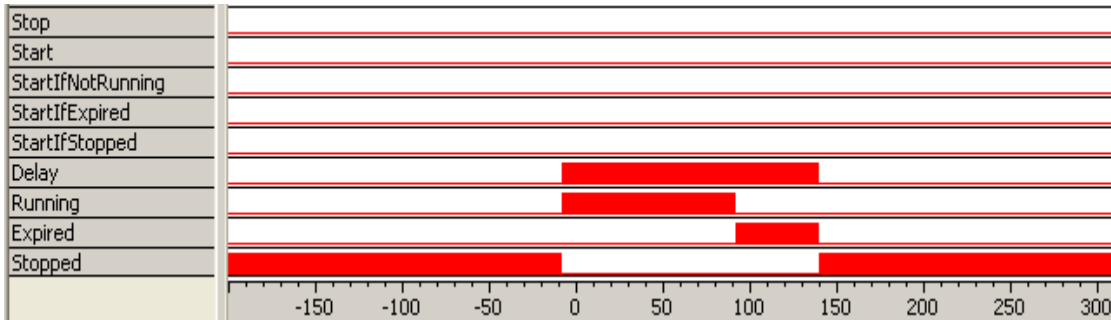


Figure 2-30 Using the “Delay” input of a timer in “Stopped” state

In “Stopped” or “Expired” state the rising edge of the signal on “Delay” input starts the timer. The running state is kept according to the parameter setting. At timeout of the timer the “Expired” output gets in logic 1 state, then at the end of the starting pulse the timer resets to “Stopped” state.

If however the “Protection started” state is shorter than the time delay setting then no trip command is generated. This is shown in the Figure below:

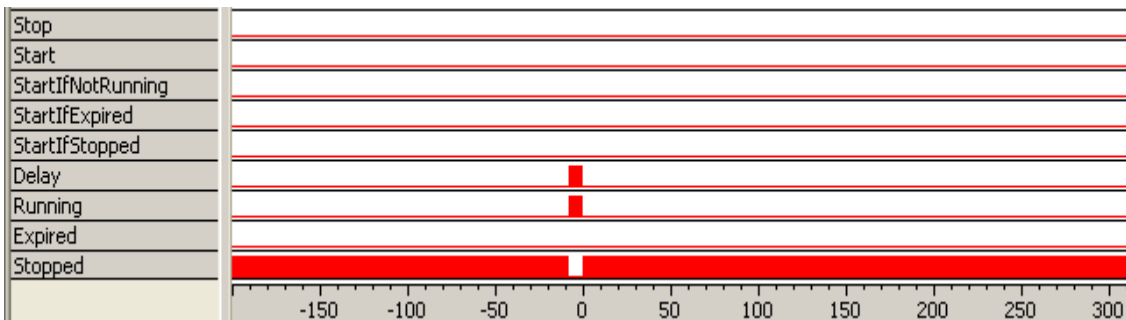


Figure 2-31 Using the “Delay” input of a timer with a short pulse

If the duration of the starting pulse on the “Delay” input is shorter than the parameter setting (in the Figure 10 ms and 150 ms respectively) then the timer does not reach the “Expired” state. At the falling edge of the starting pulse the “Stopped” state gets logic 1 at once.

## 2.1.14 Timer macros

These timer macros apply the general timer. The advantage of these macros is that the user does not need to select among several inputs and outputs to realize usual timer tasks.

### 2.1.14.1 PickDly Timer with pick-up delay

This is the usual application of a timer. The “In” input is received from any protection functions as the signal “**Protection started**”. If the started state of the protection function is longer than the time delay setting then the status of the “Out” output is applied as a TRIP command. If the circuit breaker opens and the fault is cleared then the “**Protection started**” signal resets on the “In” input. As a consequence the “Out” state resets to zero and the TRIP command is withdrawn.

If however the “**Protection started**” state (on “In” input) is shorter than the time delay setting then no trip command (on “Out” output) is generated.

The symbol and the logic diagram of a pick-up delay macro are as follows:

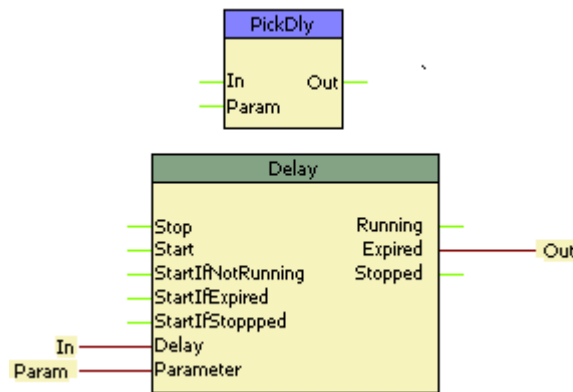


Figure 2-32 The symbol and the logic scheme of a pick-up delay macro

This macro is the simple application of the “Delay” input and the “Expired” output.



Figure 2-33 Using the pick-up delay macro with a short pulse

If the duration of the starting pulse on the “In” input is shorter than the parameter setting (in the Figure 20 ms and 150 ms respectively) then the timer does not reach the “Expired” state. No “Out” output signal is generated.

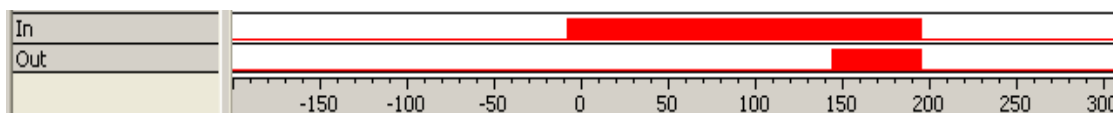


Figure 2-34 Using the pick-up delay macro with a long pulse

If however the duration of the starting pulse on the “In” input is longer than the parameter setting (in the Figure 200 ms and 150 ms respectively) then the timer generates the “Out” output signal.

### 2.1.14.2 DropDly Timer with drop-off delay

The role of this macro is the prolongation (definition of the minimum duration) of an impulse generated by any of the protection functions.

The symbol and the logic diagram of a drop-off delay macro are as follows:

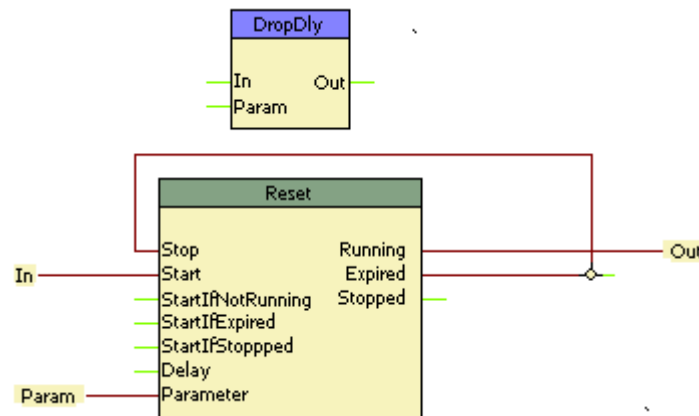


Figure 2-35 The symbol and the logic scheme of a drop-off delay macro

This macro is the simple application of the “Start” input and the “Running” output.

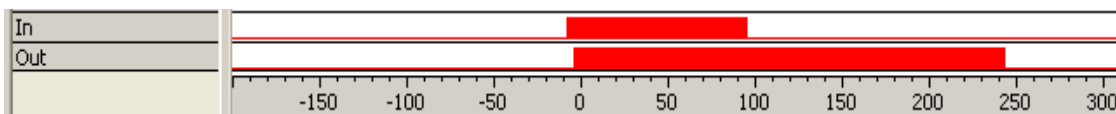


Figure 2-36 Using the drop-off delay macro to define the minimum duration of the pulse

The start input sets the timer in running state and resets the time counter. At the moment of the falling edge of the “In” signal the time measurement starts. The duration of the additional running state is defined by the timer parameter. At timeout the timer signals “Expired” state and this signal is used to reset the timer to “Stopped” state. (NOTE: the application of the “Stop” input is basically not needed, the timer performs the same operation if started either in “Expired” or in “Stopped” states.)



### 2.1.14.3 IMP impulse generation

This macro generates a pulse starting at the rising edge of the input signal and having a duration as defined by the dedicated timer parameter.

The symbol and the logic diagram of a macro for pulse generation are as follows:

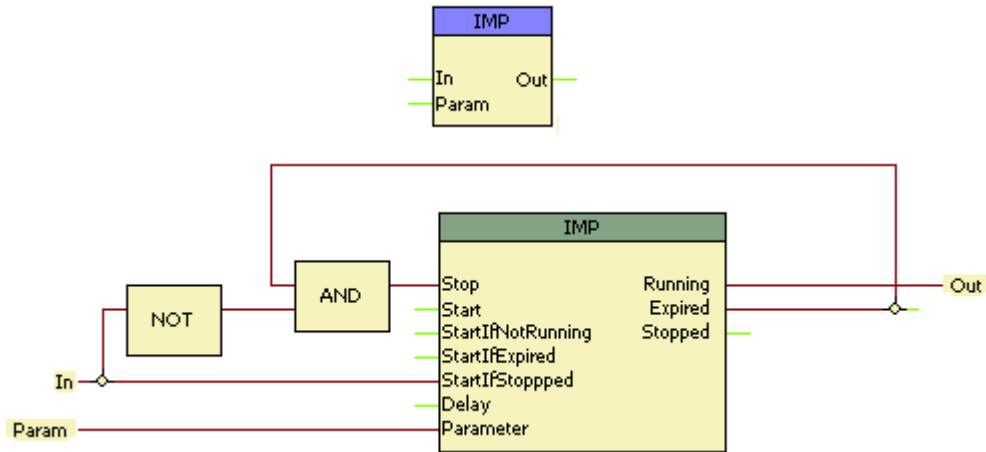


Figure 2-37 The symbol and the logic scheme of a macro for pulse generation

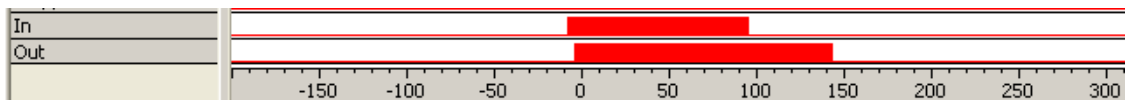


Figure 2-38 Using the impulse delay macro to prolong the duration of the pulse

If the duration of the “In” signal is shorter than the timer parameter value (in the Figure 100 ms and 150 ms respectively) then the pulse is prolonged according to the parameter setting.

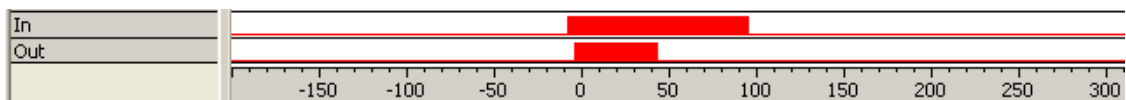


Figure 2-39 Using the impulse delay macro to limit the duration of the pulse

If the duration of the “In” signal is longer than the timer parameter value (in the Figure 100 ms and 50 ms respectively) then the pulse is terminated according to the parameter setting.

## 2.1.15 State Superv for disconnector or circuit breaker state supervision

A macro example: State supervision

Checking the dual status signals received from a circuit breaker or a disconnector is a common task. To simplify the tracking the logic scheme, this macro can be applied.

The symbol and the logic diagram of a macro for state supervision are as follows:

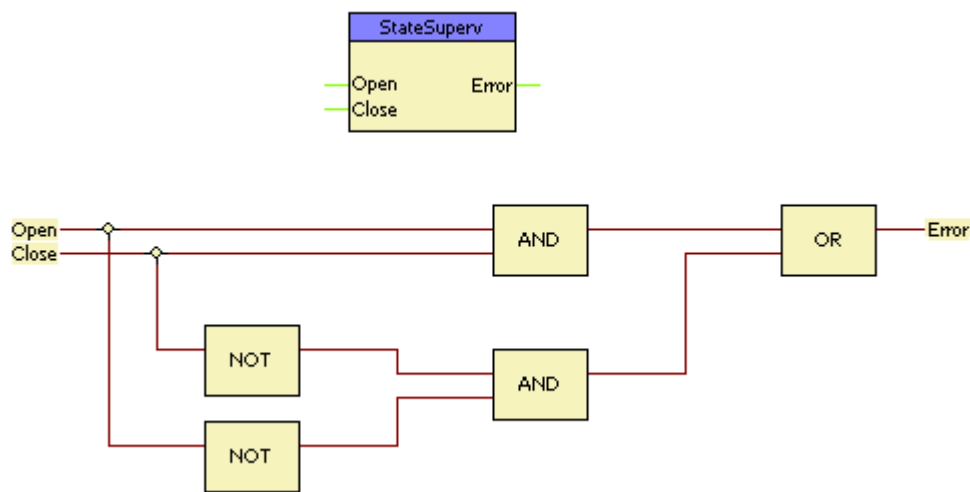


Figure 2-40 The symbol and the logic scheme of a macro for state supervision

The inputs are the “Open” and “Closed” status signals. At a moment only one of them can be logic 1. This macro outputs logic 1 to the “Error” output if both are logic 1 or both are logic 0.