Part II

Training 2

4 Timers

Types of timers

- TP - Timer is a trigger;
- TON - Timer On Delay;
- TOF - Turn-Off Delay.

All timers have the following parameters:

IN input which starts the timer (BOOL type);

PT (Preset Time) (TIME type) variable;

Q state of the timer (BOOL type);

ET countering time (TIME type).

See correct assignment of TIME constants in help files of CoDeSys. In order to use timer you need to declare it as one of the types described above.

4.1 TP

TP(IN, PT, Q, ET) is a trigger.

Write in the declaration part of the POU

```
VAR
    tmrTP1:TP;
END_VAR
```

In the code part of the programme press **F2** and using the Input Assistant find your timer in Local Variables. Select it.

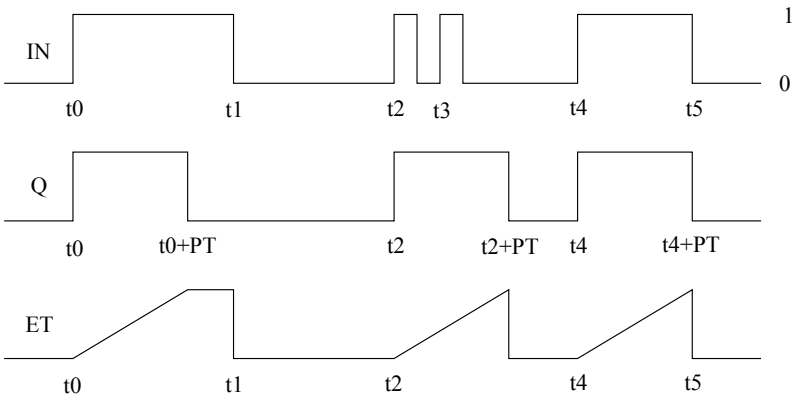

Figure 1: Graphic Display of the TP Time Sequence

As soon as IN becomes TRUE, the time will begin to be counted in milliseconds in ET until its value is equal to PT. It will then remain constant.
Q is TRUE as from IN has got TRUE and ET is less than or equal to PT. Otherwise it is FALSE.
Q returns a signal for the time period given in PT, see Fig. 1.
First, declare new variables in declaration part

```
trigUp:  BOOL;
tTP_Val:  TIME;
```

In the coding part set the next parameters

```
tmrTP1(IN:=your_input, PT:=T#5s, Q=>trigUp, ET=>tTP_Val);
your_output :=trigUP;
```

Download program to the PLC.

4.2 TON

TON(IN, PT, Q, ET) implements a turn-on delay.
Write in the declaration part of the the POU

```
VAR
    tmrTON1:TON;
END_VAR
```

As soon as IN becomes TRUE, the time will begin to be counted in milliseconds in ET until its value is equal to PT. It will then remain constant.
Q is TRUE when IN is TRUE and ET is equal to PT. Otherwise it is FALSE.
Thus, Q has a rising edge when the time indicated in PT in milliseconds has run out.

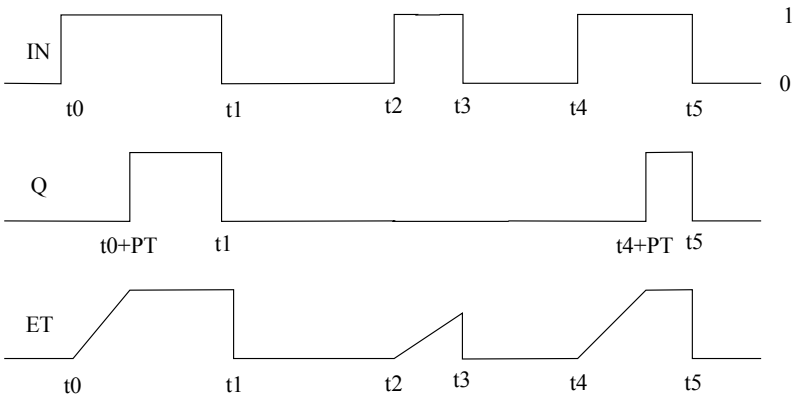

Figure 2: Graphic Display of the TON Time Sequence

4.3 TOF

TOF(IN, PT, Q, ET) implements a turn-off delay.

```
VAR
    tmrTOF1:TOF;
END_VAR
```

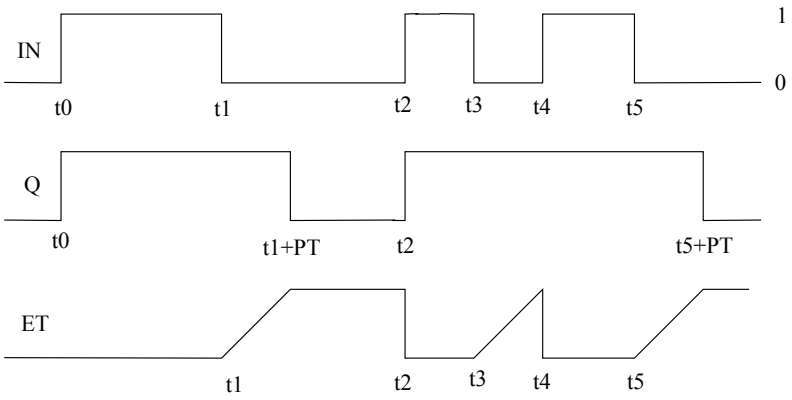

Figure 3: Graphic Display of the TOF Time Sequence

As soon as IN becomes FALSE, in ET the time will begin to be counted in milliseconds in ET until its value is equal to PT. It will then remain constant.

Q is FALSE when IN is FALSE and ET equal PT. Otherwise it is TRUE.

Thus, Q has a falling edge when the time indicated in PT in milliseconds has run out.

Example 5 *Set time on 7 s. delay*

Read the HELP for the Timers. Start timer with a switch input. Use a timer.
Add a physical output, which would indicate the timer output signal (output of the controller).

- Check timer's work switching an input signal and tracking the output of the controller.
- Observe output of the timer and the running time /Monitoring mode/.
- What would happened if input signal is shorter than the Preset Value?

Example 6 *Set an output signal which starts with an input and remains ON for 4 s.*

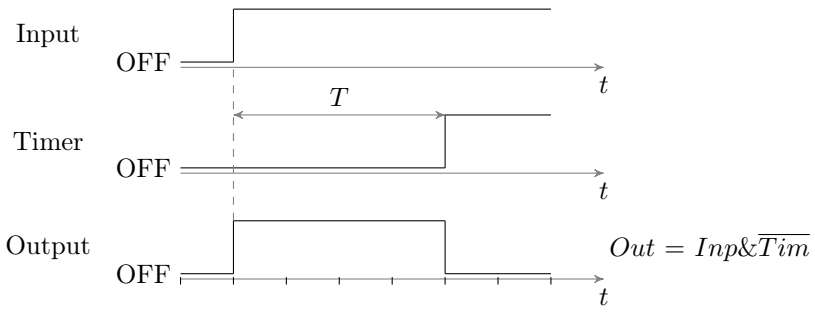

Figure 4: Delayed reaction

Choose the input and output signals for program monitoring.

- What would happened if input signal is shorter than the Preset Value?
- How to manage a long output signal then input signal is short?

Example 7 *The output signal starts (ON) 2 s. after the input signal and stops (OFF) 1 s. after the input is OFF.*

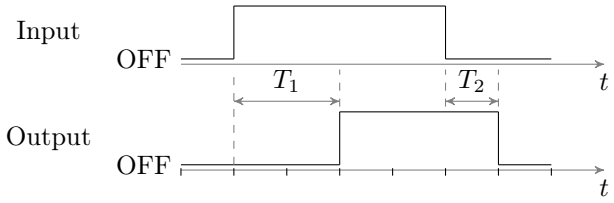

Figure 5: Input and output delayed reactions

5 Counters

Types of counters:

- CTU;
- CTD;
- CTUD.

5.1 CTU

Function block Incrementer: CV will be raised by 1. (see HELP of CoDeSys).

Write in the declaration part of the the POU

```
VAR
    cCTU1:CTU;
END_VAR
```

Write in the programm part

```
cCTU1(CU:= VarBOOL1, RESET:=VarBOOL2 , PV:= VarINT1);
```

5.2 CTD

Function Block Decrementer: CV will be lowered by 1. (see HELP of CoDeSys).

Write in the declaration part of the the POU

```
VAR
    cCTD1:CTD;
END_VAR
```

Write in the programm part

```
cCTD1(CD:= VarBOOL1, LOAD:=VarBOOL2, PV:= VarINT1);
```

5.3 CTUD

Function Block Incrementer/Decrementer.

Write in the declaration part of the the POU

```
VAR
    cCTUD1:CTUD;
END_VAR
```

Write in the program part

```
cCTUD1(CD:= VarBOOL1, LOAD:=VarBOOL3, PV:= VarINT1);
```

Example 8 *Counting*

1. Create a program that counts the pulses from input switch (increments them) and resets the counter.

   Observe counter work using Monitoring Mode.

   - What would happened if the accumulated number of pulses is greater than the counter's preset value?

2. Add to the previous part of the program for the same variable the Decrement possibility. In that case one input signal increments the value and another - decrements it.

Example 9 *Wrapping Machine*

Pieces product (bottles) come off the production line, they are countered by the sensor QP (see Fig. 6).

Then you have 9 pieces of the product the output OUT is launched for the 2 s., which stops the production line and puts the product into the box.

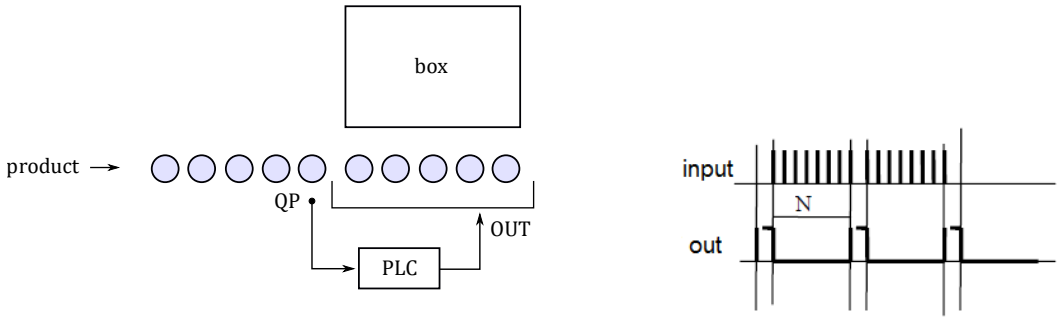

Figure 6: Wrapping Machine

Simulate a delivery of the product by the signal 1 Hz (use function block BLINK).

Function block BLINK generates a pulsating signal. Write in the declaration part of the the POU

```
VAR
    bottles:  BLINK;
END_VAR
```

Write in the program part

```
bottles(ENABLE:=_, TIMELOW:=t#1s, TIMEHIGH:=t#1s, OUT=>_ );
```

Write a program.

Example 10 *Operations with numeric data*

1. Set a constant value 5 to variable A and 2 to variable B
2. Counter (C) counts the number of impulses from Input module (\%IX._)
3. $D = A + C$; $E = B \times D$;

Compare E with constant 18. Results of comparison are presented as binary signals (<,=,>). Add Inputs that reset C, D and E values.