INTL JOURNAL OF ELECTRONICS AND TELECOMMUNICATIONS, 2018, VOL. 64, NO. 2, PP. 249-254
Manuscript received May 9, 2017; revised April, 2018. DOI: 10.24425/119584

Modelling of Parameterized Discrete Control Algorithms With Use of Finite State Machines in TIA Portal Environment

Grzegorz Andrzejewski and Wojciech Zając

***Abstract*—There exist numerous modelling techniques and representation methods for digital control algorithms, aimed to achieve required system or process parameters, e.g. precision of process modelling, control quality, fulfilling the time constrains, optimisation of consumption of system resources, or achieving a trade-off between number of parameters.**

This work illustrates usage of Finite State Machines (FSM) modelling technique to solve a control problem with parameterized external variables. The structure of this work comprises six elements. The FSM is presented in brief and discrete control algorithm modelling is discussed. The modelled object and control problem is described and variables are identified. The FSM model is presented and control algorithm is described. The parameterization problem is identified and addressed, and the implementation in PLC programming LAD language is presented. Finally, the conclusion is given and future work areas are identified.

***Keywords*—Finite State Machine, parameterised discrete control, process modelling, Ladder Diagram, TIA Portal**

I. Finite State Machine

FNITE State Machine, FSM, or Finite State Automaton, or in short Finite Automaton, is a mathematical apparatus allowing to model the behaviour of processes that can be described by discrete values of parameters (variables) and by distinguishing particular states of the system, in which the discrete variables take particular values. In other words, it is a mathematical model of behaviour of the system described by a finite number of discrete states and activities (transitions), triggered by certain conditions and changing one or more of discrete variables or parameters of the system.

The FSM model describes the state of an object by specifying the values of particular system variables or parameters at certain moments or intervals of time (states), and by defining the table of changes of system variables in subsequent discrete states (the transition table).

Let's define the Finite State Machine with Moore type outputs as in equation (1) [1]:

$$FSM = \{S, F, X, Y, \delta, \lambda\} \tag{1}$$

where:
$S=\{s_1,\ldots,s_j\}$ is non-empty and finite set of states;
F is non-empty set of directed arcs such that $F \subset (S \times S)$;
$X=\{x_1,\ldots,x_i\}$ is a finite and non-empty set of inputs;
$Y=\{y_1,\ldots,y_k\}$ is a finite set of outputs;
δ is a function assigning to each arc of a subset of the input δ: $F \rightarrow X$;
λ is a function assigning to each state a subset of the output λ: $S \rightarrow Y$.

Authors are with The Jacob of Paradies University, Gorzów Wielkopolski, Poland (e-mails: {GAndrzejewski; WZajac}@ajp.edu.pl).

II. Modelling the FSM automaton

To model of the Finite Automaton operation algorithm it is necessary to identify and describe the components of equation (1). An example of the FSM algorithm is presented in Fig. 1.

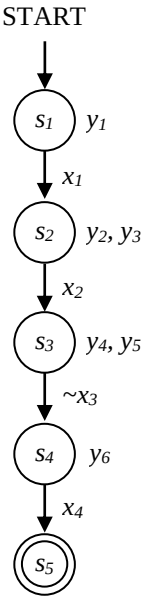

Fig. 1. An example of FSM algorithm.

For the algorithm to change a state it is necessary to fulfil a certain transition condition: setting or resetting a discrete input variable, occurrence of a particular combination of signals, etc. Changing the state of the automate causes changes in values of discrete output variable.

The FSM example in the Fig. 1. After starting, it waits in the state s_1 for trigger condition x_1 to be fulfilled. In this state, the y_1 output signal is active. Activation of x_1 input causes FSM to change the state to s_2, in which there are activated y_2 and y_3 outputs. If the input x_2 is set, the machine changes the state to s_3, with y_4 and y_5 are active.

The transition condition for the system to change the state from s_3 to s_4 is that x_3 input is not active. In s_4 the y_6 output is activated. Finally, the occurrence of x_4 input signal will cause the machine to go to s_5 state, reset all outputs and stop the operation.

III. CONTROL OBJECT

As an example of control object with parameterized variables, there was taken a cross-roads model with two traffic flows and corresponding streetlights (Fig. 2), according to appropriate regulations [2, 3].

The streetlights are controlled by signals as follows:

- Red A - red light in A flow,
- Yellow A - yellow light in A flow,
- Green A - green light in A flow,
- Red B - red light in B flow,
- Yellow B - yellow light in B flow,
- Green B - green light in B flow.

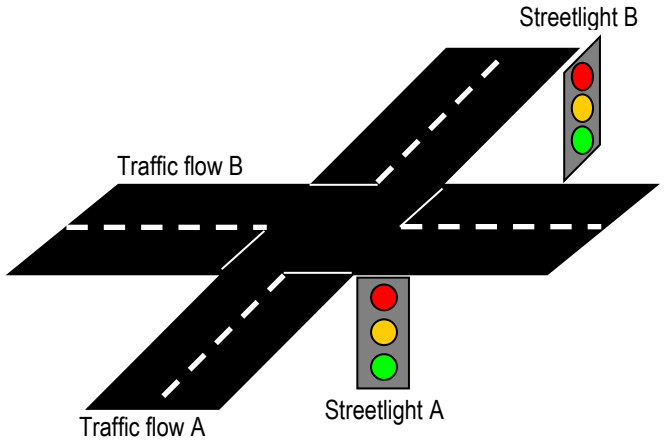

Fig. 2. Cross-roads model as control object.

In FMS description of the object control algorithm there can be identified seven states s_1 to s_7. Each state has individual set of active outputs. In Table I there is presented relation between states and activated output signals.

Table II presents logical values of particular discrete outputs in subsequent states of FSM.

In Fig. 3. there is presented a FSM graph, modelling the control object behaviour.

TABLE I
FSM STATES AND ACTIVE SIGNALS

State	s_1	s_2	s_3	s_4	s_5	s_6	s_7
Active Signals	Green A Red B	Yellow A Red B	Red A Yellow B Red B	Red A Green B	Red A Yellow B	Red A Yellow A Red B	-

TABLE II
LOGICAL VALUES OF SYSTEM OUTPUTS IN PARTICULAR FSM STATES

Signals/states	s_1	s_2	s_3	s_4	s_5	s_6	s_7
Red A	0	0	1	1	1	1	0
Yellow A	0	1	0	0	0	1	0
Green A	1	0	0	0	0	0	0
Red B	1	1	1	0	0	1	0
Yellow B	0	0	1	0	1	0	0
Green B	0	0	0	1	0	0	0

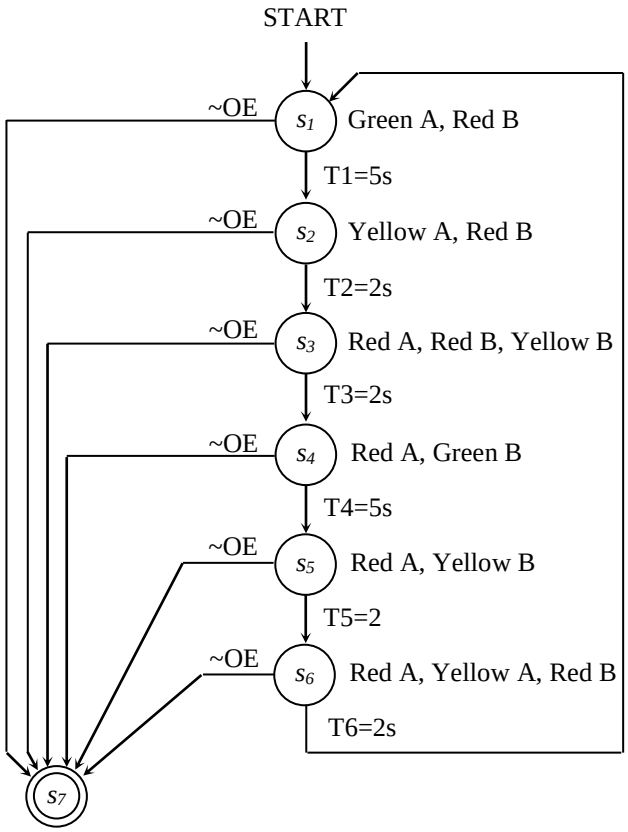

Fig. 3. FSM graph of streetlights control.

The presented system changes the FMS states on the timing conditions. Therefore it was necessary to introduce timing signals T1 to T6.

For research purposes, there were assigned short times, 2 and 5 seconds, as convenient for implementation verification. They will be transformed to variables on further research steps and changed.

Except for T1-T6 input signals, there was introduced Operation Enable (OE) input signal. If it is active, the FMS, once started, performs the control algorithm in infinite loop. Deactivation of the OE causes the algorithm to change the FSM state to s_7 and to reset all output signals.

Presented algorithm was modeled in Ladder Diagram language [3, 4, 5] and implemented on PLC platform (Siemens SIMATIC S7-1200 series) [6, 7].

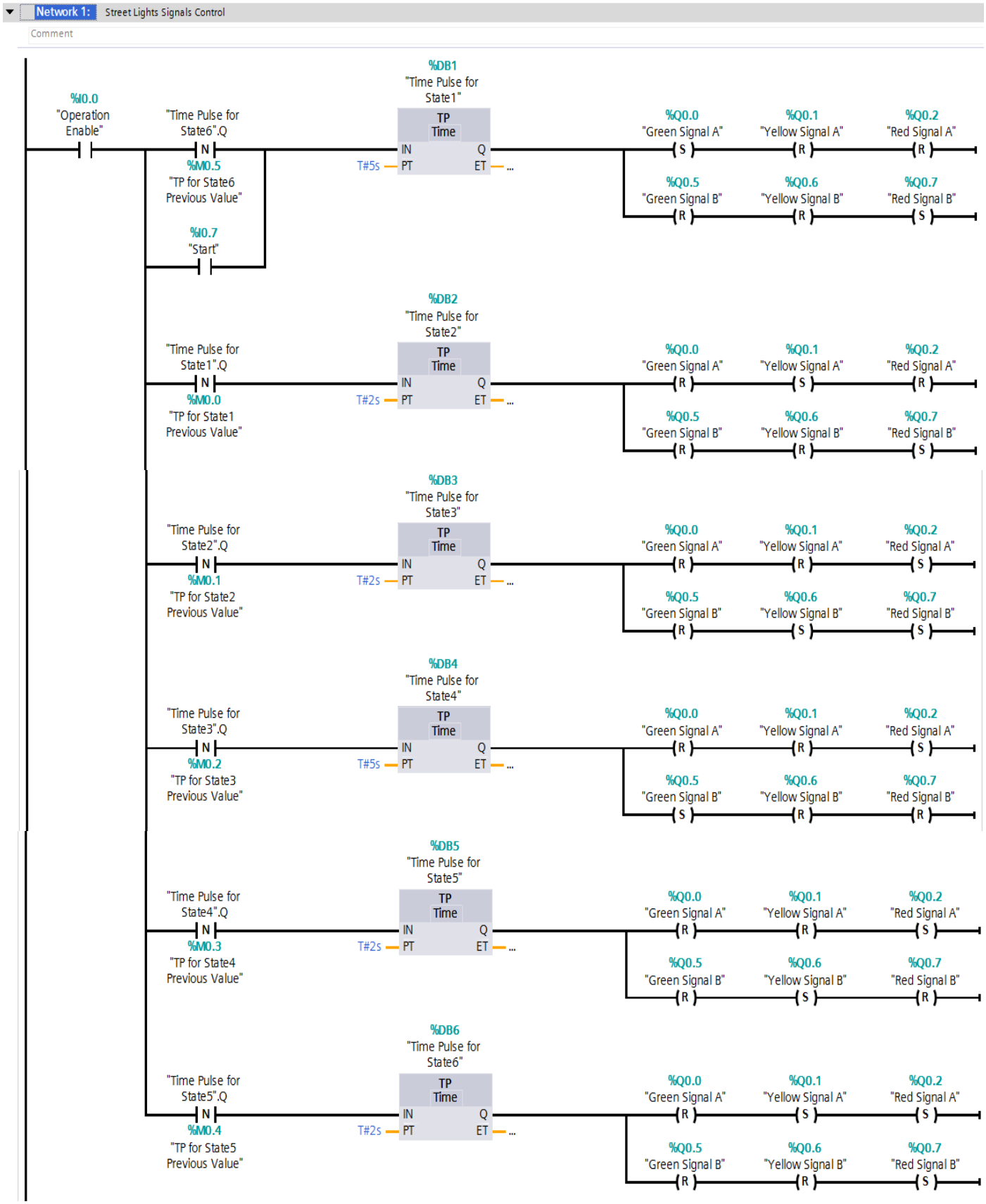

Fig. 4. Control network for streetlight signals.

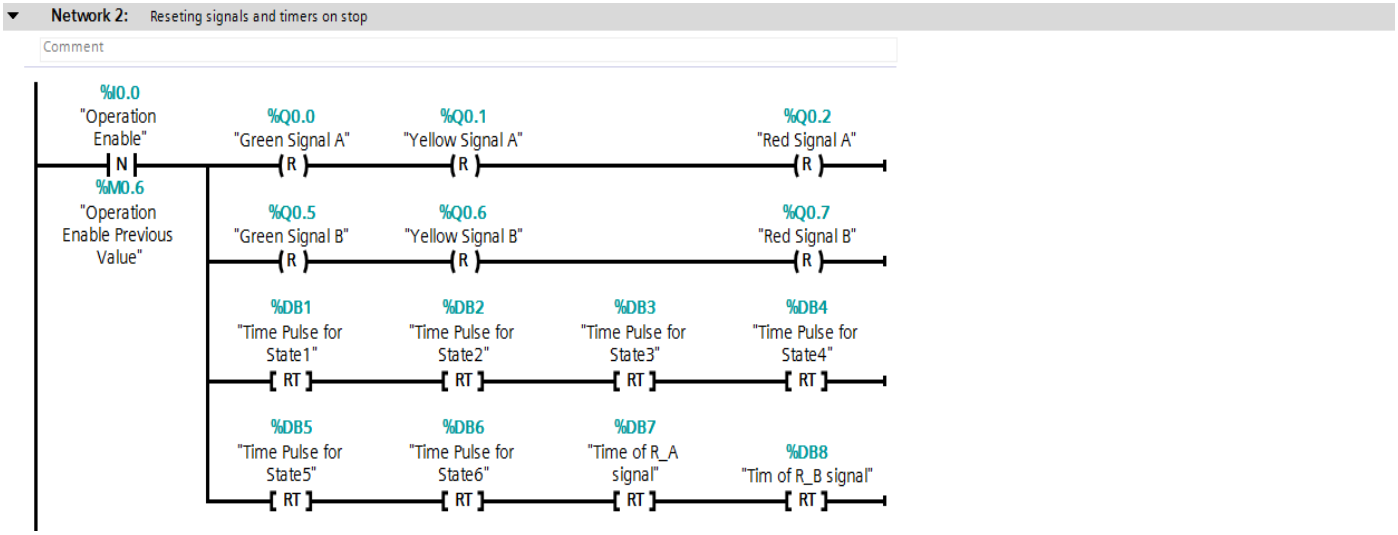

Fig. 5. Control network for system stop.

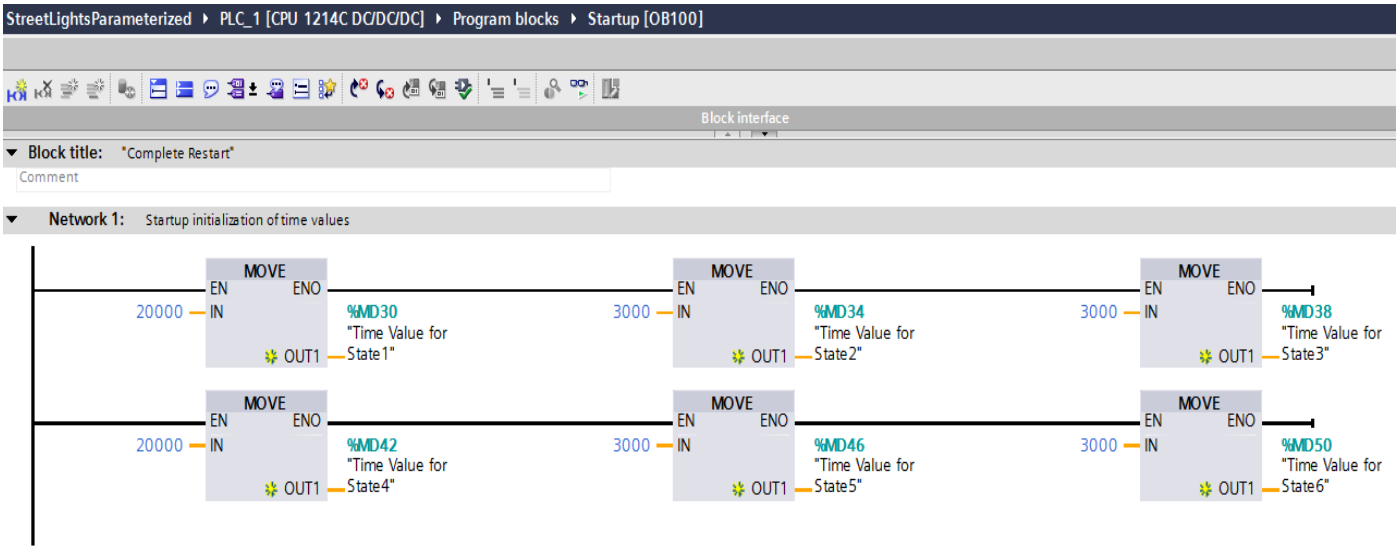

Fig. 6. Startup block of modified program.

IV. LAD IMPLEMENTATION

The FMS algorithm was implemented as Step7 LAD program in SIEMENS Totally Integrated Automation Portal environment, for target platform Siemens SIMATIS S7-1200 PLC. The initial version with no parameterisation is presented in Fig. 4. and 5.

Fig. 4. presents main control network for streetlight signals. The "Operation Enable" signal allows the system to be set into a state of being ready to start operation. When the signal is active, the "Start" signal initiates transition of the system to the s_1 state, by starting the timer "Time Pulse for State 1". When started, the timer enables its output and invokes "Set" coils for "Green Signal A" and "Red Signal B" and Reset coils for "Yellow Signal A", "Red Signal A", "Green Signal B" and "Yellow Signal B". The length of time pulse is determined by PT field of the "Time Pulse for State1" data block (DB1). When the time elapses, the timer disables its output.

"Time Pulse for State 2" timer is initiated by LAD language N contact, detecting negative edge of the Q output signal of "Time Pulse for State1" data block. When the "Time Pulse for State1" disables the output, "Time Pulse for State2" timer starts, enabling the Set coils for "Yellow Signal A" and "Red Signal B" and Reset coils for "Green Signal A", "Red Signal A", "Green Signal B" and "Yellow Signal B". This is equivalent to putting the system into s_2 state. The length of time pulse is determined by PT field of the "Time Pulse for State2" data block (DB2).

The s_3 state is initiated by negative edge detection on "Time Pulse for State 2.Q" data field. It starts "Time Pulse for State 3", lasting for the time defined in the timer PT field. It enables "Red Signal A", "Yellow Signal B" and "Red Signal B" and disables signals "Green Signal A", "Yellow Signal A" and "Green Signal B".

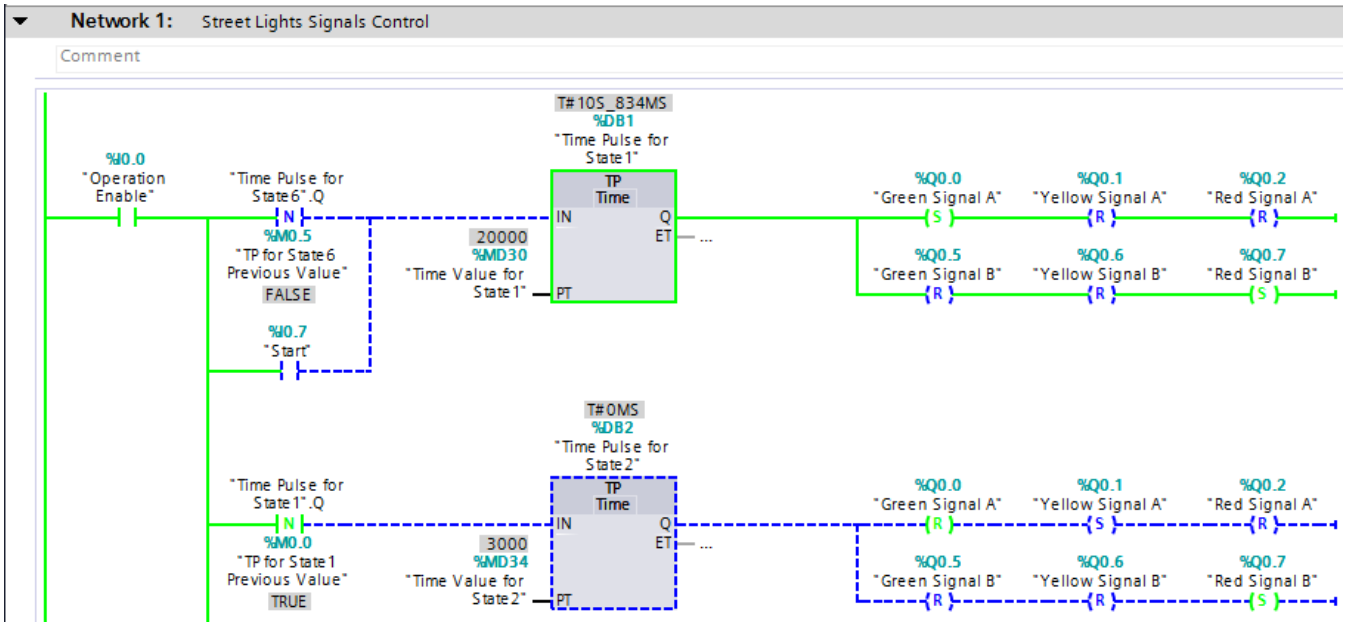

Fig. 7. Screenshot of modified program in operation.

"Time Pulse for State4", initiated by disabling the "Time Pulse for State3.Q" field, puts the system into s_4 state ("Green Signal A" disabled, "Yellow Signal A" disabled, "Red Signal A" enabled, "Green Signal B" enabled, "Yellow Signal B" disabled , "Red signal B" disabled).

The s_5 state is controlled by "Time Pulse for State5" timer, initiated by resetting "Time Pulse for State4.Q" signal. The timer sets "Red Signal A" and "Yellow Signal B and resets "Green Signal A", "Yellow Signal A",", "Green Signal B" and "Red signal B".

The next state, s_6, is initiated by negative edge of "Time Pulse for State5.Q" signal, starting "Time Pulse for State6" timer. In this state there are enabled "Yellow Signal A", "Red Signal A" and "Red signal B" and "Green Signal A", "Green Signal B" and "Yellow Signal B" are disabled.

When "Time Pulse for State6" disables its Q output, negative edge detection contact initiates "Time Pulse for State 1" timer and the operation cycle is repeated.

V. FSM PARAMETERISATION

To allow introducing variable times to the FMS algorithm, it is necessary to define six memory data blocks, for variables defining times for particular states. In the example, the variables are :

- "Time Value for State1",
- "Time Value for State2",
- "Time Value for State3",
- "Time Value for State4",
- "Time Value for State5",
- "Time Value for State6".

The time values are initially determined in the program. In the example, for the FSM to initially fill the data with time values there was user a special *Startup Block* of LAD language. If present in the program, the block is performed only once, as the first block of program. In the example in Fig. 6. the time variables are initiated with appropriate data (time in milliseconds).

In Fig. 7. there is presented a screenshot of a modified program during operation, with time values as PT input parameters of timer blocks.

During operation of the control system, time values can be changed manually by system operator or calculated by dedicated algorithms, e.g. of adaptive flow control [8].

The PLC platform used in this example, similarly to other modern PLCs, allows the operator to monitor the control process state and change the system variables with use of Human-Machine Interface panel. This can be used in the given example, to monitor or to change the states time values.

The other possibility is to employ the WWW server of PLC unit and create Web page that will allow to monitor and/or modify the system parameters with use of any popular web browser on PC/PG system connected via network to PLC.

VI. CONCLUSION AND FUTURE WORK

The paper gives an example of Finite State Machines (FSM) modelling technique to solve a control problem with parameterized external variables. The example was implemented and tested in PLC environment. The method proved to be efficient and straightforward to implement in LAD language.

Further research will be aimed to apply the technique in a system for adaptive traffic flow control.

References

[1] M. Adamski, A. Barkalov, „Architectural and sequential synthesis of digital devices”, University of Zielona Góra, 2006
[2] Regulation of the Ministry of Infrastructure dated 3 July 2003 of detailed technical specifications for signs and traffic signals and traffic safety equipment and conditions of their placing on the roads, OJ 2003 No. 220, item. 2181 (in Polish)
[3] K. Małecki, S. Jaszczak, R. Sokołowski, "Synthesis of Hardware and Software Traffic Control Simulator for a Particular Area", Measurement Automation and Control (Pomiary, Automatyka, Kontrola), no 7/2012, pp. 608-610 (in Polish) (2012)
[4] T. Łuba (red.), M. Rawski, P. Tomaszewicz, B. Zbierzchowski: „Programowalne Układy Przetwarzania Sygnałów i Informacji”, Wydawnictwa Komunikacji i Łączności, Warszawa 2008
[5] M. Adamski, J. Tkacz, Formal reasoning in logic design of reconfigurable controllers Proceedings of 11th IFAC/IEEE International Conference on Programmable Devices and Embedded Systems - PDeS 2012. Brno, Czech Republic (2012)
[6] A. Barkalov, L. Titarenko: „Logic synthesis for FSM-based control units”, Springer-Verlag, Berlin 2009, in Lecture Notes in Electrical Engineering, Vol. 53
[7] A. Barkalov, L. Titarenko, M. Kołopieńczyk, K. Mielcarek, G. Bazydło "Logic synthesis for FPGA-based finite state machines", Cham Heidelberg : Springer International Publishing Switzerland, 2016, Studies in Systems, Decision and Control, Vol. 38
[8] S. Iwan, K. Małecki, "Data Flows in the Integrated Urban Freight Transport Telematics System", Communications in Computer and Information Science, no. 0329, pp. 79-86 (2012)