Ladder Diagram Petri Nets: Discrete Event Systems

José Carlos Quezada Quezada,
Ernesto Flores García, Joselito Medina Marín,
Jorge Bautista López and Víctor Quezada Aguilar

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.75753

Abstract

Ladder diagram language (LD) is a common programming language in industry to develop control algorithms of discrete event systems (DESs). Besides, it is one of the five programming languages supported by the International Electrotechnical Commission through the IEC-61131-3 standard. On the other hand, Petri net (PN) theory is both a graphical and mathematical tool used to model discrete event systems, particularly in this study, control lines used in industrial algorithms. Control algorithms in LD are generally developed based on the experience of control system programmers. Therefore, it is still a relevant problem how to formalize the current and new control algorithms. In this chapter, are analyzed lines in LD used more frequently in control algorithms. Additionally, an element-to-element transformation methodology from a LD program to a PN model is proposed.

Keywords: control algorithms, discrete event systems, ladder diagram language, model, petri nets

1. Introduction

LD language is one of the five languages contemplated in the standard IEC-61131-3 [1], its use in the industry is due to its similarity with the electrical diagrams, and its behavior is based mainly on the electromechanical relay, but LD language also has the capacity to include logical functions blocks. The others languages are: function block diagram (FBD), instructions list (IL), structured text (ST) and sequential function char (SFC).

There are two types of control lines that are analyzed and converted into PN structures: the logical AND, OR, AND-OR, auto-loop and interlocking, which have both discrete inputs and outputs. The logic blocks such as timer, counter and comparator have all analog inputs, but their control output is discrete.

The main motive or need to model the control algorithms in LD is because they are developed mainly based on the experience of programmers in industrial control [2, 3], so it is important to propose approaches that help guarantee the safe control algorithms applied in machines or industrial processes, and the theory of PN [4] allows modeling the basic control lines used in the LD algorithms. Different approaches have been presented to provide a solution to analyze, model and simulate control algorithms developed in LD with PN or vice versa [5–11].

Physical or discrete memory signals can have two states (activated or deactivated, 0 or 1, etc.), so, we propose a distribution of these signals to PN structure that can model both states, but only one active at a time. On the other hand, the cyclic operation of PLC generates cyclic evaluation of the control algorithm in function of the states of physical input and memory signals. This behavior must be considered to avoid accumulation of tokens in places of PN structures, for which reason marking conditions are proposed in places that represent physical or memory outputs of PN structures of control lines in LD. Likewise, cyclic evaluation of control lines generates the energized and de-energized behavior of coils; therefore, it is also necessary to restore conditions of PN structures of each control line in LD, conditioning the marking in function of the input places [12, 13].

To convert control lines with analog inputs, places where their marking is a data (color in colored Petri nets) are included [9], which may be changing depending on the logic control algorithm. Conditioned transitions are proposed for their firing depending on the behavior of the control block in respective LD.

Based on analysis of the control lines, we propose the definition of a PN for discrete event systems in LD (LDPN), with which PN structures of control lines in LD are generated.

2. Control lines in LD to discrete event systems

The LD language has as its operating principle the behavior of an electromechanical relay, with the option of including function blocks. The standards IEC-61131-3 define LD like "*modeling networks of simultaneous functioning electromechanical elements, such as relay contacts and coils, timers, counters, etc.*" The control lines analyzed are the logic AND, OR, AND–OR, auto-loop, interlocking, timers, counters and mathematic comparisons. The first five logical have discrete inputs and output. Meanwhile in the logical of timers, counters and mathematical comparisons have analog inputs and discrete outputs.

The run of control algorithm in PLC is cyclic, and it mainly performs five actions such as reading of physical inputs, copy status of physical inputs, evaluation of the control algorithm with previous copy, copy of the status of physical outputs and sending of these statuses to physical modules.

2.1. Control lines both discrete inputs and outputs

Figure 1 shows the control line of logic AND, when all contacts In_1, In_2, …, In_n allow electric power flow, then Out1 coil is energized. Eq. (1) is the model corresponding.

$$\mathrm{Out1} = \mathrm{In\_1}\,\&\&\,\mathrm{In\_2}\,\&\&\ldots\mathrm{In\_n} \tag{1}$$

Figure 2 shows the control line of logic OR, when any contact In_1, In_2, …, In_n allows electric power flow, then Out1 coil is energized, its model is stand for the Eq. (2).

$$\mathrm{Out1} = \mathrm{In\_1} \parallel \mathrm{In\_2} \parallel\ldots\mathrm{In\_n} \tag{2}$$

Figure 3 shows the control line of logic AND–OR. When the contacts In_1, In_2, …, In_n or the contacts In_1, In_3, …, In_n allow electric power flow, then Out1 coil is energized. Eq. (3) is the model corresponding.

$$\mathrm{Out1} = (\mathrm{In\_1}\,\&\&\,In\_2\,\&\&\ldots In\_n) \parallel (In\_1\,\&\&\,In\_3\,\&\&\ldots In\_n) \tag{3}$$

Figure 4 shows the control line of logic auto-loop. When the contacts In_1, In_2, …, In_n or the contacts Out1, In_2, …, In_n allow electric power flow, then Out1 coil is energized. Eq. (4) is the model corresponding.

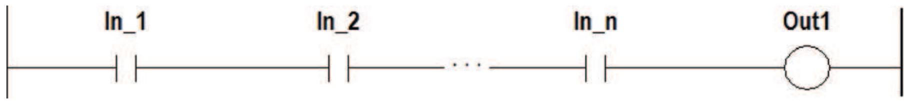

Figure 1. Control line of logic AND.

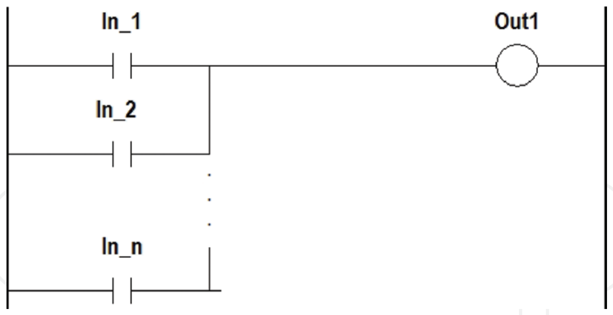

Figure 2. Control line of logic OR.

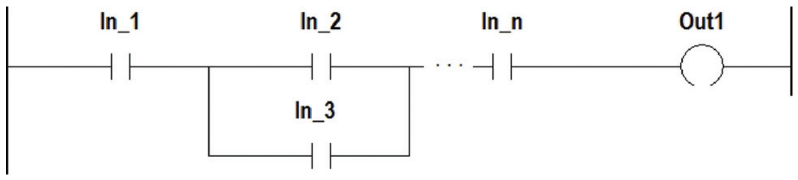

Figure 3. Control line of logic AND–OR.

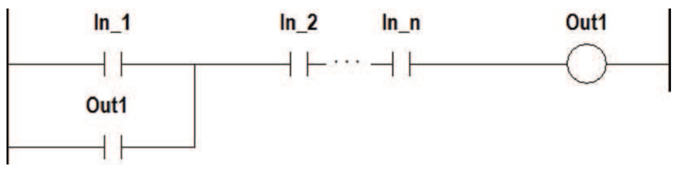

Figure 4. Control line of logic auto-loop.

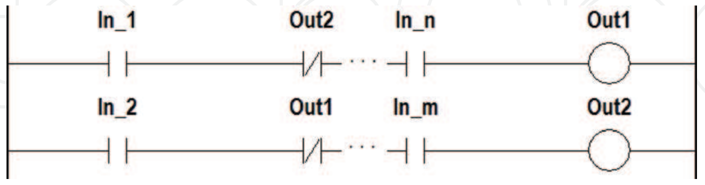

Figure 5. Control line of logic interlocking.

$$\mathrm{Out1} = (\mathrm{In\_1} \,\&\,\& In\_2 \,\&\,\& In\_n) \parallel (Out1 \,\&\,\& In\_2 \,\&\,\& In\_n) \tag{4}$$

Figure 5 shows the control line of logic interlocking, when the contacts In_1, ~Out2, …, In_n allow electric power flow, then Out1 coil is energized, and it blocks the energizing of Out2 coil. If Out2 coil is energized first, then Out1 coil cannot be energized. Eq. (5) is the model corresponding.

$$\mathrm{Out1} = \mathrm{In\_1} \,\&\,\& \overline{\mathrm{Out2}} \,\&\,\&\ldots\ In\_n; \mathrm{Out2} = \mathrm{In\_2} \,\&\,\& \overline{\mathrm{Out1}} \,\&\,\&\ldots\ In\_m \tag{5}$$

2.2. Control lines with analog inputs and discrete output

Figure 6 shows the standard function block of on-delay timer (TON) and its timing diagram of the functional [1]. The signals *Preset_time* and *Elapsed_time* are analog. If the contact In_1 allows electric energy flow, when *Elapsed_time* adds base time and if *Elapsed_time* is equal or greater than *Preset_time*, then Out1 coil is energized. Eq. (6) depicts the logic model of the block TON.

$$\text{If } (\mathrm{In\_1} = 1 \,\&\,\& \mathrm{ET} \geq \mathrm{PT}), \text{ then Out1} = 1 \tag{6}$$

Restart condition: If In_1 = 0, then ET = 0 and Out1 = 0.

Figure 7 shows the standard function block of off-delay timer (TOF) and its timing diagram of the functional [1]. If the contact In_1 allows energy power, then the Out1 coil is energized, and the *Elapsed_time* variable is set to zero. When the In_1 signal is equal to zero, the *Elapsed_time* variable adds base time and if *Elapsed_time* is equal or greater than *Present_time*, then Out1 coil is de-energized. Eq. (7) shows the logic model of block TOF.

$$\text{If } (\mathrm{In\_1} = 0 \,\&\,\& \mathrm{ET} \leq \mathrm{PT}), \text{ then Out1} = 0 \tag{7}$$

Restart condition: If In_1 = 1, then ET = 0 and Out1 = 1.

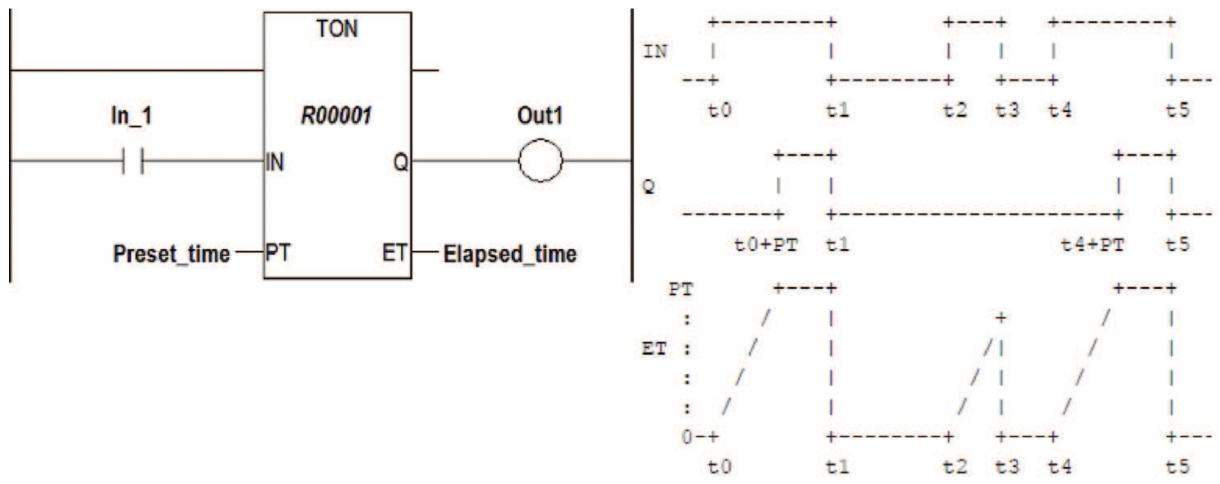

Figure 6. On-delay timer.

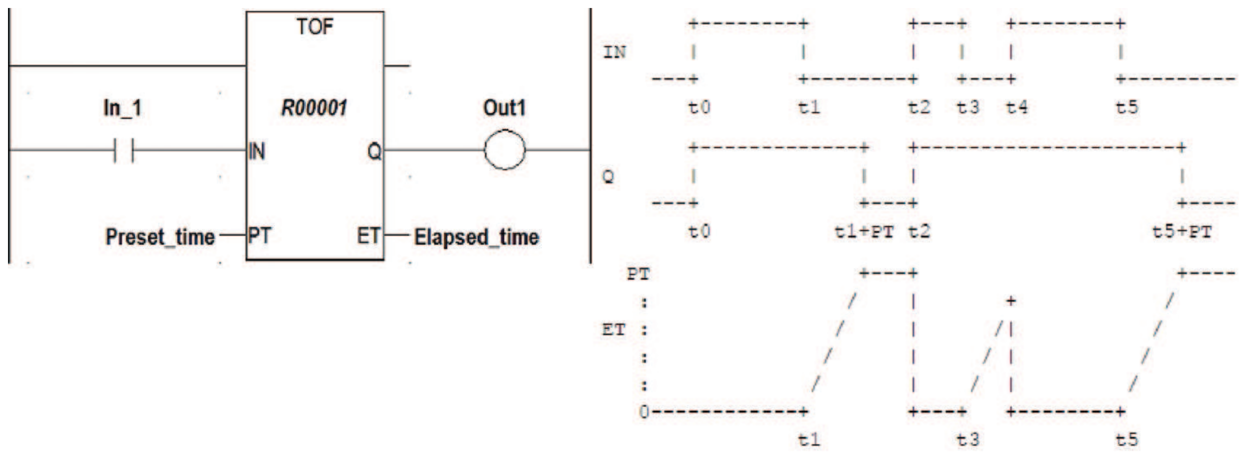

Figure 7. Off-delay timer.

Figure 8 shows two counter function blocks: (1) up-counter and (2) down-counter. In both blocks, the contact In_1 is the pulse to counter, that is and positive transition is detected; if the contact In_2 allows electric energy flow, then Out1 coil is de-energized, and the *Current_value* variable is set to zero in up-counter and to *Preset_value* in down-counter. In up-counter, if *Current_value* is equal or greater than *Preset_value*, then Out1 coil is energized. In down counter, if *Current_value* is equal to zero, then Out1 coil is energized. Eqs. (8) and (9) are logic models of the counters, respectively.

$$\text{if In\_1 } (\uparrow), \text{then CV} = \text{CV} + 1; \text{if } (\text{In\_2} = 0 \,\&\,\& \, CV \geq PV), \, then \;\, Out1 = 1 \tag{8}$$

Restart condition: If In_2 = 1, then CV = 0

$$\text{if In\_1 } (\uparrow), \text{then CV} = \text{CV} - 1; \text{if } (\text{In\_2} = 0 \,\&\,\& \, CV \leq 0), \text{then Out1} = 1 \tag{9}$$

Restart condition: If In_2 = 1, then CV = PV.

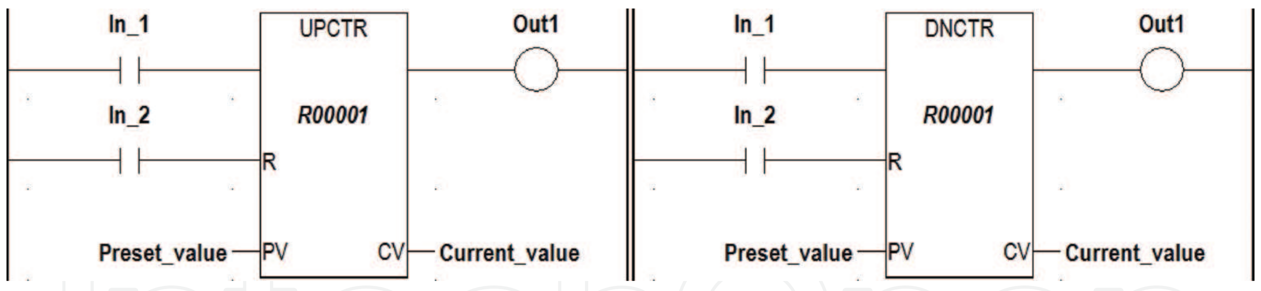

Figure 8. Counters. a) Counter up, b) Counter down.

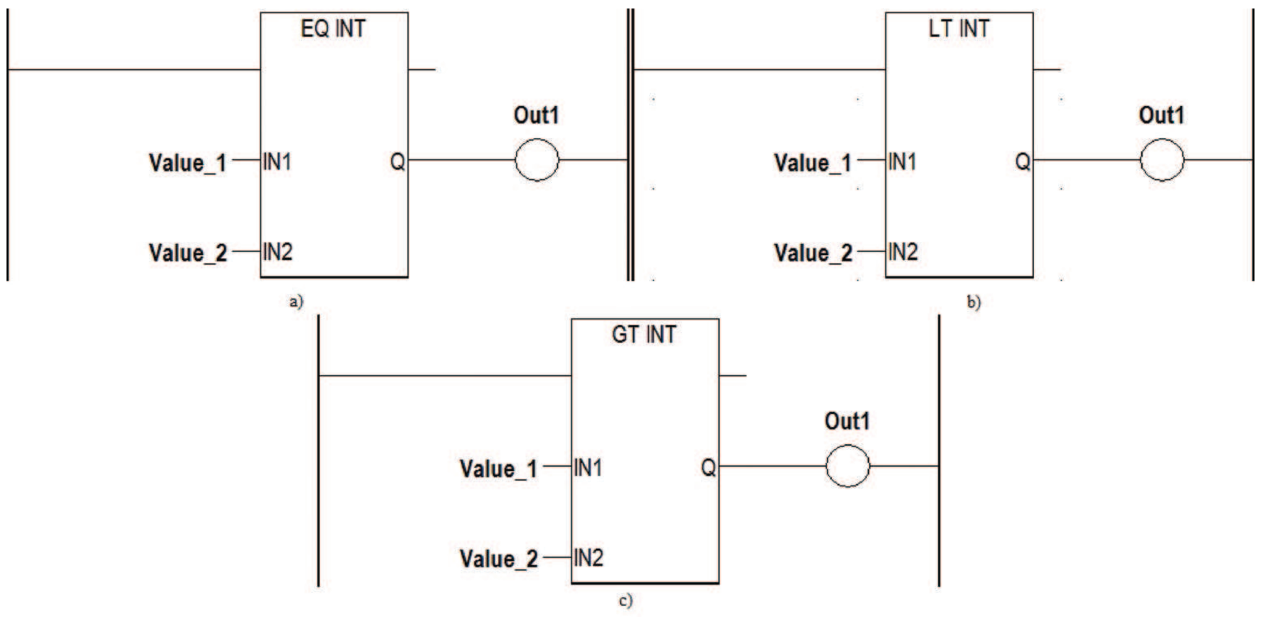

Figure 9. Mathematical comparisons. a) Relation, equal to, b) Relation, lower than, c) Relation, greater than.

Figure 9 shows the standard comparison function blocks: a) equal to, b) lower than and c) greater than. In all the blocks, two analog signals are compared, and depending on result is energized or de-energized Out1 coil. The logic models, respectively, are specified in Eqs. (10–12).

$$\text{If Value\_1} = \text{Value\_2, then Out1} = 1 \tag{10}$$

$$\text{If Value\_1} < \text{Value\_2, then Out1} = 1 \tag{11}$$

$$\text{If Value\_1} > Value\_2,\ then\ Out1 = 1 \tag{12}$$

3. Model of control lines in PN

In this section, the bases of the PN theory are indicated, and the discrete-LDPN network, which is the basis for generating the PN structures of the control lines in LD, is defined.

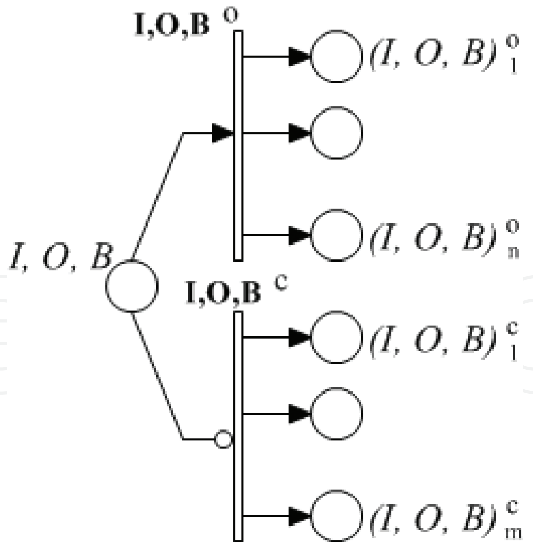

Figure 10. Distribution of discrete signals in PN.

Likewise, the conditions for marking places and triggering transitions are described to model the cyclical evaluation behavior of the control algorithm in PLC.

3.1. Petri nets

PN are a graphic and mathematic tool mean to modeling DES behavior. Graphically, a PN uses circles in order to represent places, rectangles to represent transitions and arcs with arrow or circle to link the inputs and output places with a transition. The relation between places and transition can be represented mathematically by means of an incidence matrix. For a PN with n transitions and m places, its incidence matrix $A = [a_{ij}]$ is an integer number matrix representing the weighting of the input and output arcs; a_{ij}^{+} represents the weighting of output arcs from transitions and a_{ij}^{-} represents input arcs to transitions. Eq. (13) represents how the incidence matrix values are obtained.

$$a_{ij} = a_{ij}^{+} - a_{ij}^{-} \tag{13}$$

To model the dynamic behavior of DES, PN has the state equation, which shows the marking in the net sequentially from initial marking M_{k-1} and when applying a firing vector u_k to the transpose of the respective incidence matrix A^T, respectively. Eq. (14) shows the relationship between them.

$$M_k = M_{k-1} + A^T u_k \tag{14}$$

3.2. LDPN: Discrete event systems

In an LD control algorithm, a discrete signal can have n contacts normally open and m contacts normally closed. The work in [12] shows a representation of discrete signals used in LD to PN, which is the base of conversion of control lines that have both discrete inputs and outputs. On the other hand, evaluation of control algorithm in PLC is cyclical, which generates two important conditions to consider in the PN model; the cyclical evaluation in PN would generate accumulation of marks in the places, and in function of the logic, marking and consuming of theses in places that represent coils in the LD. This last condition is also necessary to restore the information of places in PN that represent physical analog signals or memory registers. **Figure 10** shows the distribution of discrete signals in PN, and Eq. (15) its interpretation. Only one transition can be enabled at a time; if the input place does not have a mark, then the transition $(\mathbf{I},\mathbf{O},\mathbf{B})^{\mathbf{c}}$ is enabled for inhibitor arc. Eq. (16) is the generalization of the marking of I, O y B places.

$$I_i = \left\{I_n^o \cup I_m^c\right\}; O_o = \left\{O_n^o \cup O_m^c\right\}; B_b = \left\{B_n^o \cup B_m^c\right\} \tag{15}$$

where the subscripts n and m are not necessarily equal.

$$M(I,O,B) = \begin{bmatrix} 0 \\ 1 \end{bmatrix}, then \left\{ \begin{matrix} M(I,O,B)^o = 0 \; and \; M(I,O,B)^c = 1 \\ M(I,O,B)^o = 1 \; and \; M(I,O,B)^c = 0 \end{matrix} \right\} \tag{16}$$

Considering symbols of [3], for a pre-set and post-set of places, are defined:

$*t = \{p : (p,t) \in F\}$, the set of input places of t.

$*t = \{p : (t,p) \in F\}$, the set of output places of t.

For tokens accumulation problem in input places, the Eqs. (17) and (18) are proposed. Both equations are is in function of the marking of inputs places and of output place. Eq. (17) is for structures with logic AND, and Eq. (18) for logic OR.

$$(O,B)(t*) = \left\{\prod M(*t) = 1 \,\&\,\& \,(O,B)(t*) = 0\right\} \tag{17}$$

$$(O,B)(t*) = \left\{\sum M(*t) = 1 \,\&\,\& \,(O,B)(t*) = 0\right\} \tag{18}$$

In the same way, to consume token in output place and restoring conditions of PN structures, Eqs. (19) and (20) are defined, which are in function of both marking input places and output places.

$$RC(t*) = \left\{\prod M(*t) = 0 \,\&\,\&(O,B)(t*) = 1\right\} \tag{19}$$

$$RC(t*) = \left\{\sum M(*t) = 0 \,\&\,\&(O,B)(t*) = 1\right\} \tag{20}$$

From the above, the ladder diagram Petri net: discrete event systems is defined as it is shown in **Table 1**.

Eq. 16 to distribution of signals, Eqs. 17 and 18 to accumulate tokens and Eqs. 19 and 20 to restart conditions should be evaluated after each marking of the net M_{k+1} to update the marking of LDPN and simulate cycled behavior of PLC. Marking of input places I is in function of discrete sensors states.

LDPN considers the following transition rules to dynamic behavior:

- In initial conditions of LDPN, inhibitor arcs enable transitions and put token in its output places O and/or B in PN model with both inputs and outputs discrete. In AI places restart condition of data.
- All output places (O and B) of the PN model are binary, only one can token.
- All transitions enabled should be fired in one some evaluation. To PN model with both inputs and outputs discrete, transition fired T consume unique token $\mathbf{W}(P, T) = 1$ of each input place **P** of **T** and put to unique token $\mathbf{W}(T, P) = 1$ to each output place **T** of **P**. For PN model with some analog input place and output place discrete, the transition **T** should be fired when it satisfies the respective condition (*if* - *then*) and put to unique token in each output place **T** of **P**.
- To update, marking should be applied Eqs. 16–20.

3.3. Model of control lines both discrete inputs and outputs

Figure 11 shows the PN model of logic AND, if input places I_1^o, O_3^c *y* B_2^o have a token, then $\mathbf{L_1}$ transition is enabled. The $\mathbf{L_1}$ firing puts a token at place $O_{1.}$ When are updates the marking of input places, the Eq. (17) disables the $\mathbf{L_1}$ transition, avoiding a token more in output place O_1.

A Discrete-LDPN is a 5-tuple (**P, T, W, F, M₀**), where:

$\mathbf{P} = \{I \cup O \cup B \cup AI \cup AR \cup RC\}$ is a finite set of places, where:
$I = \{I_1, I_2, \ldots, I_i\}$ is a finite set of places that represent discrete physical inputs,
$O = \{O_1, O_2, \ldots, O_o\}$ is a finite set of places that represent discrete physical outputs,
$B = \{B_1, B_2, \ldots, B_b\}$ is a finite set of places that represent discrete memory signals,
$AI = \{AI_1, AI_2, \ldots, AI_{ai}\}$ is a finite set of places that represent analog physical inputs,
$AR = \{AR_1, ARI_2, \ldots, AR_{ar}\}$ is a finite set of places that represent analog memory signals,
$RC = \{RC_1, RC_2, \ldots, RC_{rc}\}$ is a finite set of places to restart condition of the nets and its marking it in function of the states of inputs and outputs of control line type.

$\mathbf{T} = \left\{\mathbf{I}^{c|o}, , \mathbf{O}^{c|o}, , \mathbf{B}^{c|o}, \mathbf{L}, \mathbf{AI}, \mathbf{RC}\right\}$ is a finite set of transitions, where:
$\mathbf{I}^{c|o} = \left\{\mathbf{I}_1^{c|o}, \mathbf{I}_2^{c|o}, \ldots, \mathbf{I}_i^{c|o}\right\}$ is a finite set of transitions that have discrete physical inputs,
$\mathbf{O}^{c|o} = \left\{\mathbf{O}_1^{c|o}, \mathbf{O}_2^{c|o}, \ldots, \mathbf{O}_o^{c|o}\right\}$ is a finite set of transitions that have discrete physical outputs,
$\mathbf{B}^{c|o} = \left\{\mathbf{B}_1^{c|o}, \mathbf{B}_2^{c|o}, \ldots, \mathbf{B}_b^{c|o}\right\}$ is a finite set of transitions that have discrete memory signals,
$\mathbf{L} = \{\mathbf{L}_1, \mathbf{L}_2, \ldots, \mathbf{L}_l\}$ is a finite set of transitions that may have places of discrete signals,
$\mathbf{AI} = \{\mathbf{AI}_1, \mathbf{AI}_2, \ldots, \mathbf{AI}_{ai}\}$ is a finite set of transitions that can have discrete and/or analog signals, its fire condition it in function of mathematics or logics restrictions.
$\mathbf{RC} = \{\mathbf{RC_1}, \mathbf{RC_2}, \ldots, \mathbf{RC_{rc}}\}$ is a finite set of transitions that have input place RC to restart condition of PN structure.

$\mathbf{F} \subseteq (\mathrm{P} \times \mathrm{T}) \cup (\mathrm{T} \times \mathrm{P})$ is a set of arcs.

$\mathbf{W} = \mathbf{F} \rightarrow \{1\}$, all weights of the arcs are equal to 1.

$$\mathbf{M}_0 = \begin{cases} \mathrm{P} \rightarrow \{0,1\}, & \text{discrete signal.} \\ \mathrm{P} \rightarrow \{\mathrm{Z}\ (16\ \text{bit integer})\}, & \text{analog signal.} \end{cases}$$

Table 1. Definition of LDPN: Discrete event systems.

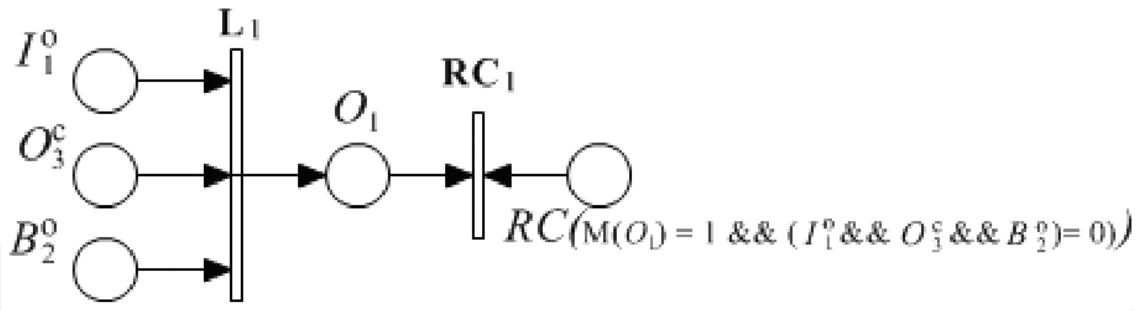

Figure 11. PN model of logic AND.

By Eq. (19), the marking of place *RC* is in function of both marking input places and output place.

Figure 12 shows the PN model of logic OR, if any input places I_1^c, O_5^o *y* B_2^o have a token, then $\mathbf{L_1}$, $\mathbf{L_2}$ or $\mathbf{L_3}$ transition is enabled, respectively. If the transition enabled is fired, then a token is put at place O_1. When are updates the marking of input places, the Eq. (18) disables the $\mathbf{L_1}$, $\mathbf{L_2}$ and $\mathbf{L_3}$ transitions, avoiding a toke more in output place O_1. By Eq. (20), the marking of place *RC* is in function of both marking input places and output place.

Figure 13 shows the PN model of logic AND-OR, output place can get token from L_1 or L_2 transitions, in function of marking of input places I_1^o, O_3^c *y* B_2^o or I_1^o, O_3^c *y* O_7^c have a token, respectively. The $\mathbf{L_1}$ or $\mathbf{L_2}$ firing puts a token at place O_1. When are updates the marking of input places, the Eqs. (17) and (18) disables the $\mathbf{L_1}$ and $\mathbf{L_2}$ transitions, avoiding a token more in output place O_1. The marking of place *RC* is in function of both marking input places of $\mathbf{L_1}$ and $\mathbf{L_2}$ transitions and output place O_1 based on Eqs. (19) and (20) to restart condition.

Figure 14 shows the PN model of logic auto-loop. In this model, it is necessary that $\mathbf{L_1}$ transition to be enabled and fired set a token in the output place O_1, enabling the $\mathbf{O_1^o}$ transition, which consumes the token of O_1 and sets a token in the place O_1^o, enabling the $\mathbf{L_2}$ and holding a token in O_1. The restart condition of the model auto-loop is in function of the Eqs. (19) and (20).

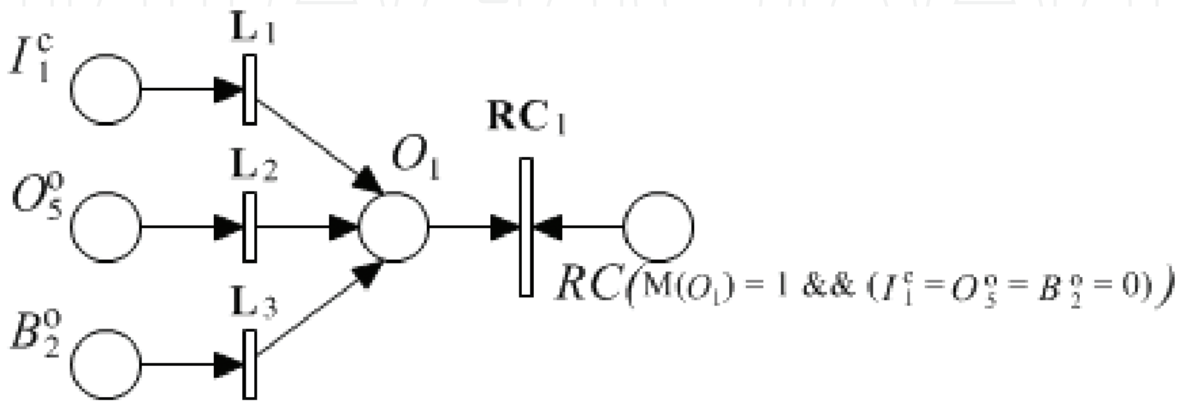

Figure 12. PN model of logic OR.

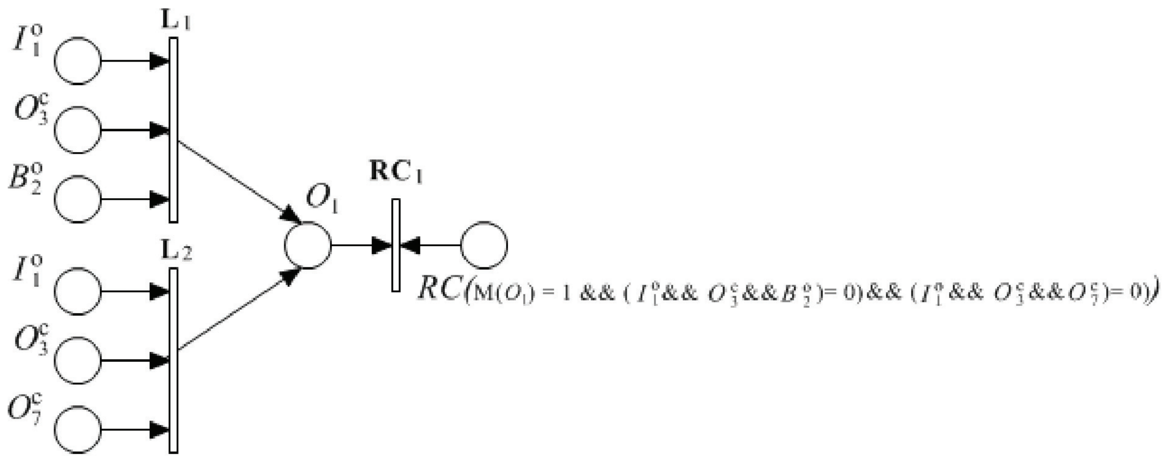
I_1^o
O_3^c
B_2^o
I_1^o
O_3^c
O_7^c
L1
L2
O_1
RC1
$RC(M(O_1) = 1 \&\& (I_1^o \&\& O_3^c \&\& B_2^o) = 0) \&\& (I_1^o \&\& O_3^c \&\& O_7^c) = 0))$

Figure 13. PN model of logic AND–OR.

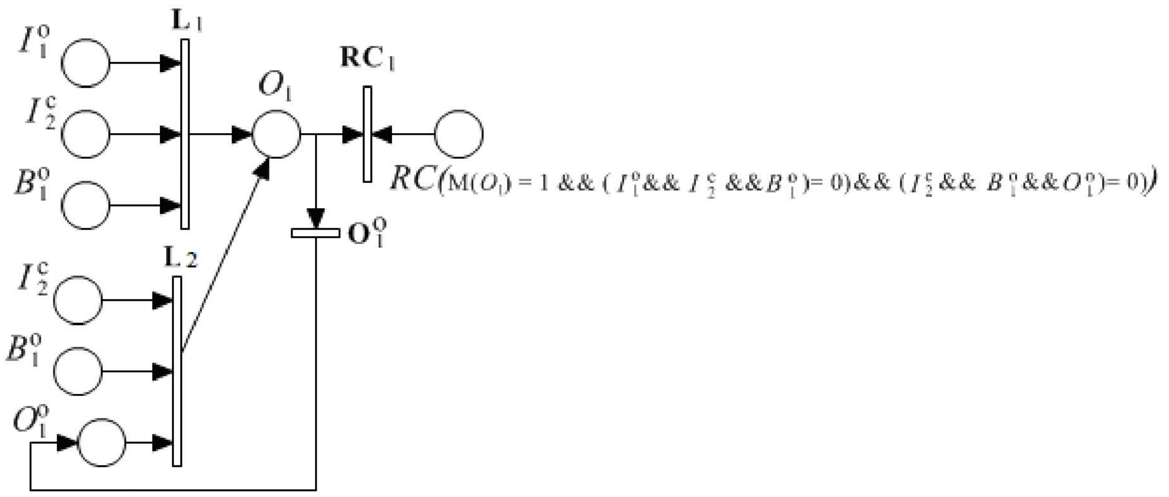
I_1^o
I_2^c
B_1^o
I_2^c
B_1^o
O_1^o
L1
L2
O_1
RC1
$\mathbf{O_1^o}$
$RC(M(O_1) = 1 \&\& (I_1^o \&\& I_2^c \&\& B_1^o) = 0) \&\& (I_2^c \&\& B_1^o \&\& O_1^o) = 0))$

Figure 14. PN model of logic auto-loop.

Figure 15 shows the PN model of logic interlocking. Both places O1 and O2 enable the $\mathbf{O_1^c}$ and $\mathbf{O_2^c}$ transitions by the inhibitor arcs, placing a token in input places O_1^c and O_2^c of $\mathbf{L_1}$ and $\mathbf{L_2}$ transitions, respectively. If $\mathbf{L_1}$ or $\mathbf{L_2}$ transition is firing first disables the other transition by the inhibitor arc. The restart condition places RC_1 and RC_2 are in function of Eq. (19).

3.4. Model of control lines with analog inputs and output discrete

Figure 16 shows the PN model of on-delay timer. The BT and PT are variables to determine base time and preset time, respectively. The marking of the place AI_2 is a data analog to store the sum ET = ET + BT. The marking of the place O_1 is in function of firing of the AI_2 transition, which depends on the condition $if(I_1 = 1 \& \& ET \geq PT)$. To restart condition of the places O_1 and AI_2 are in function of I_1^c.

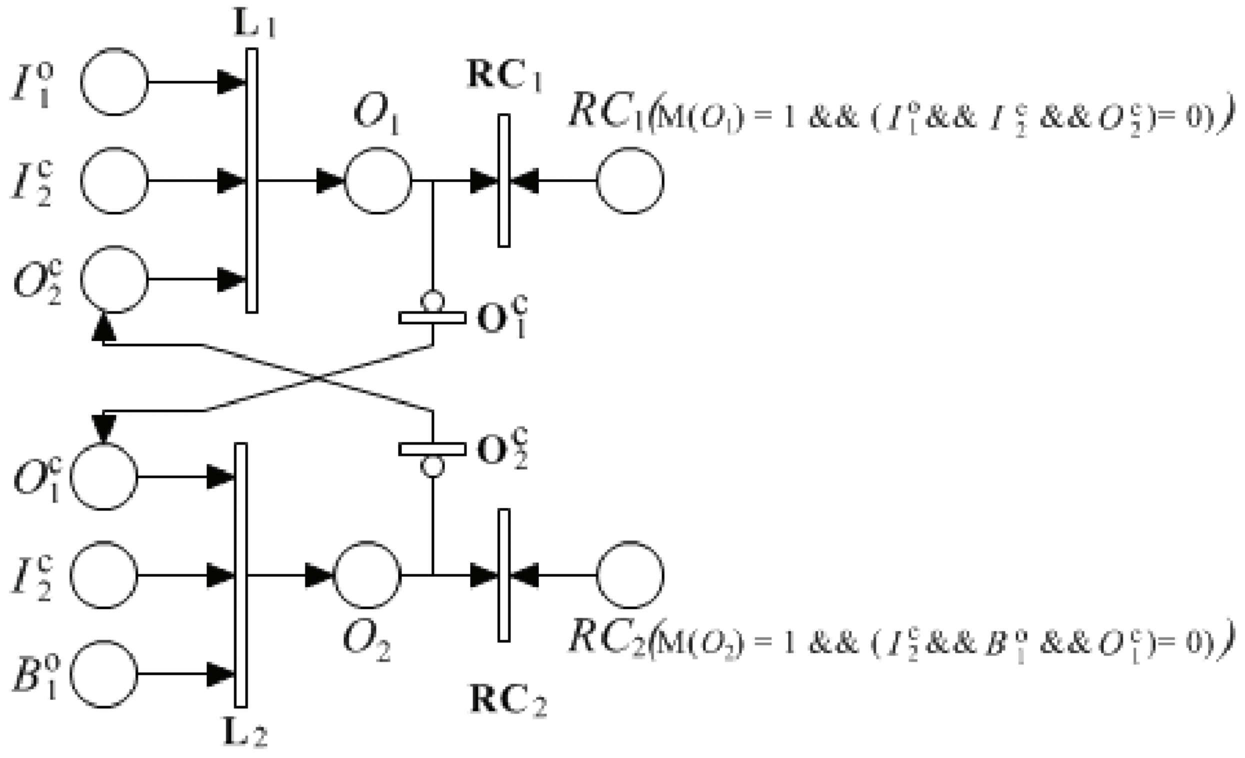
L1
I_1^o
I_2^c
O_2^c
O_1
RC1
$RC_1(M(O_1) = 1 \&\& (I_1^o \&\& I_2^c \&\& O_2^c) = 0)$
O_1^c
O_2^c
O_1^c
I_2^c
B_1^o
O_2
$RC_2(M(O_2) = 1 \&\& (I_2^c \&\& B_1^o \&\& O_1^c) = 0)$
RC2
L2

Figure 15. PN model of logic interlocking.

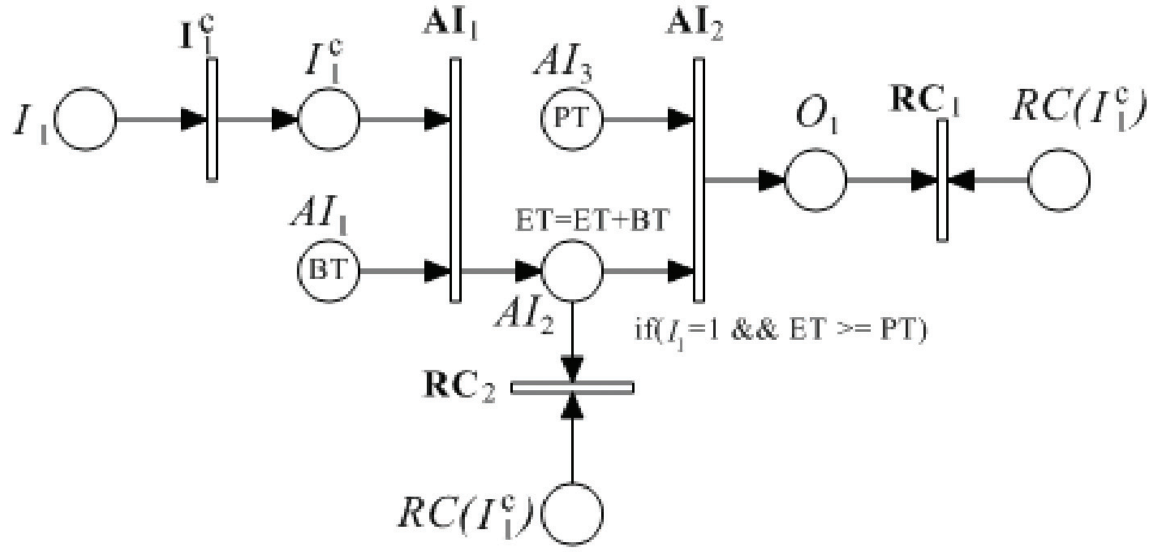

Figure 16. PN model of on-delay timer.

Figure 17 shows the PN model of logic off-delay timer. The place to restart condition RC and fire of $\mathbf{RC_1}$ is putting a token in output place O_1 that is the initial condition of the structure PN. When the $\mathbf{I_1^c}$ transition is fired put a token in place I_1^c, which enables $\mathbf{AI_1}$ transition to allow the sum of time $ET = ET + BT$. The fire of A$\mathbf{I_2}$ transition is in function of $if(I_1 = 0 \& \& ET \leq PT)$, if

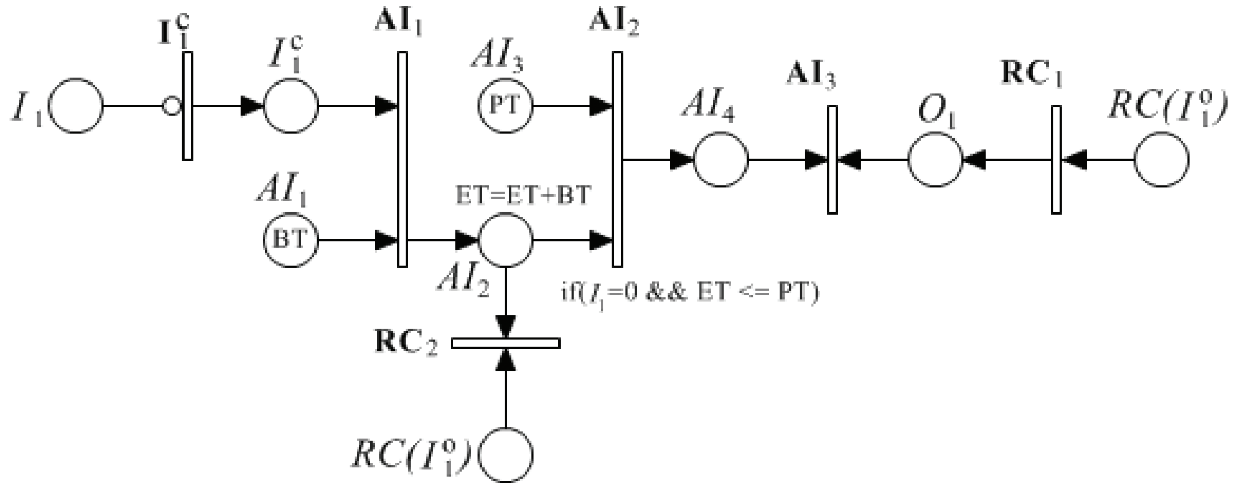

Figure 17. PN model of logic off-delay timer.

it is fired, then put a token in place AI_4, which enables $\mathbf{AI_3}$ transition and consumes the token of place O_1.

Figure 18 shows the PN model of logic up-counter and **Figure 19** to PN model of logic down-counter. In both models, the marking of the place I_1^o is in function of $M(I_1^o) = \{M(I_1(*t)) = 0\,\&\&\,M(I_1(t*)) = 0\}$, which is to detect a positive transition in the marking, respectively. In place AI_1 are added the tokens (positive transition), *if* $(CV \geq PV)$ then $\mathbf{AI_1}$ transition is enabled, and its fire put a token in place O_1. If the place $RC(I_2^o)$ has a token, then it is consumed the token of the place O1 and $CV = 0$.

Figure 19 shows the PN model of logic down-counter, which has similar behavior to up-counter, just that if one token in place I_1^o consume one token of place AI_1, *if* $(CV \leq 0)$, then the fire of $\mathbf{AI_1}$ transition puts a token in place O_1.

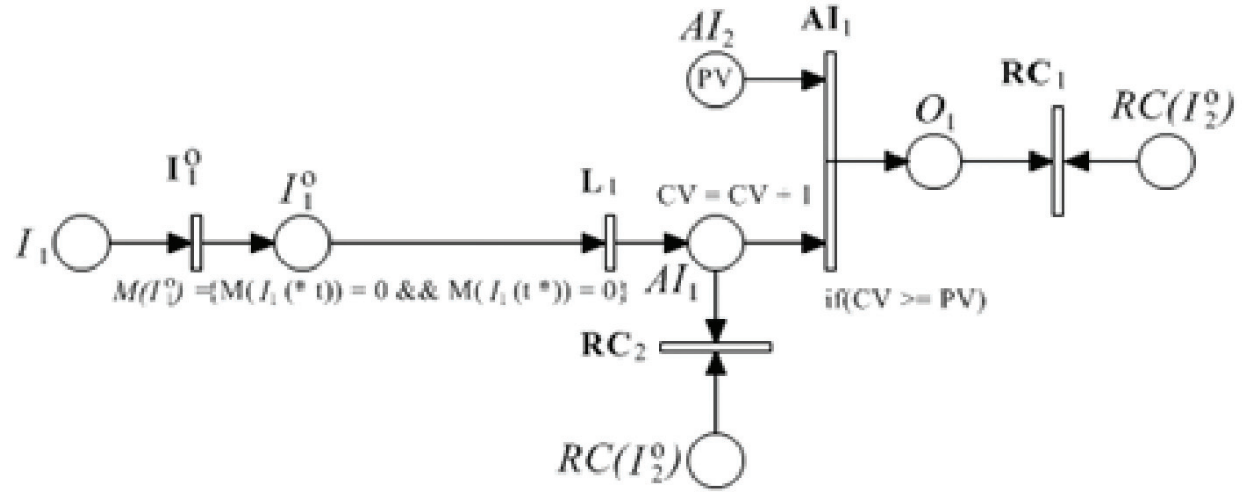

Figure 18. PN model of logic up-counter.

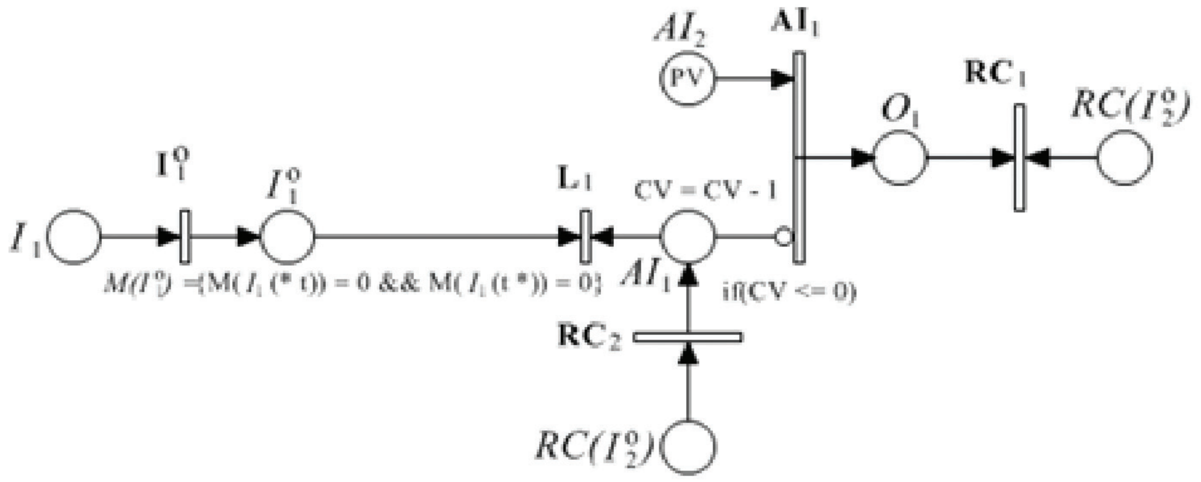

Figure 19. PN model of logic down-counter.

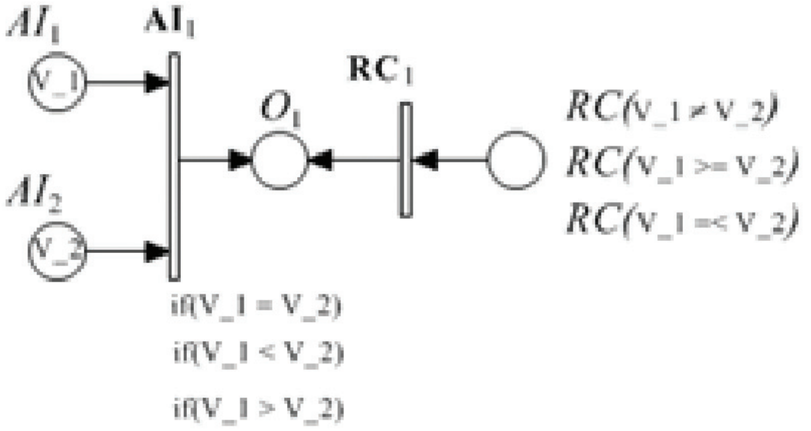

Figure 20. PN model of logic of comparisons.

Figure 20 shows the PN model of logic of comparisons of two analog places. Enabling and firing of $\mathbf{AI_1}$ is in function of $if(V_1 = V_2); if(V_1 < V_2); if(V_1 > V_2);$ according to the comparison. Similarly, the marking of place RC is in function of $RC(V_1 \neq V_2); RC(V_1 \geq V_2);$ $RC(V_1 \leq V_2),$ respectively.

4. Example

Figure 21 shows the control algorithm in LD of run of three motors sequentially [14]. The Start and Stop signals are physical inputs of type pushbutton. The Motor_1, Motor_2 and Motor_3 coils are physical outputs. The IR1, IR2 and IR3 variables are bits of memory. The first control line is logic of auto-loop, if Start variable is equal to one, then, the IR1 coil is energized and so it is hold by the contact IR1. It is also energized the Motor_1 coil, and the timer T1 and T2 begin counting time. In T1, if $ET \geq PT$, then, the IR2 and Motor_2 coils are energized. In T2, if $ET \geq PT$,

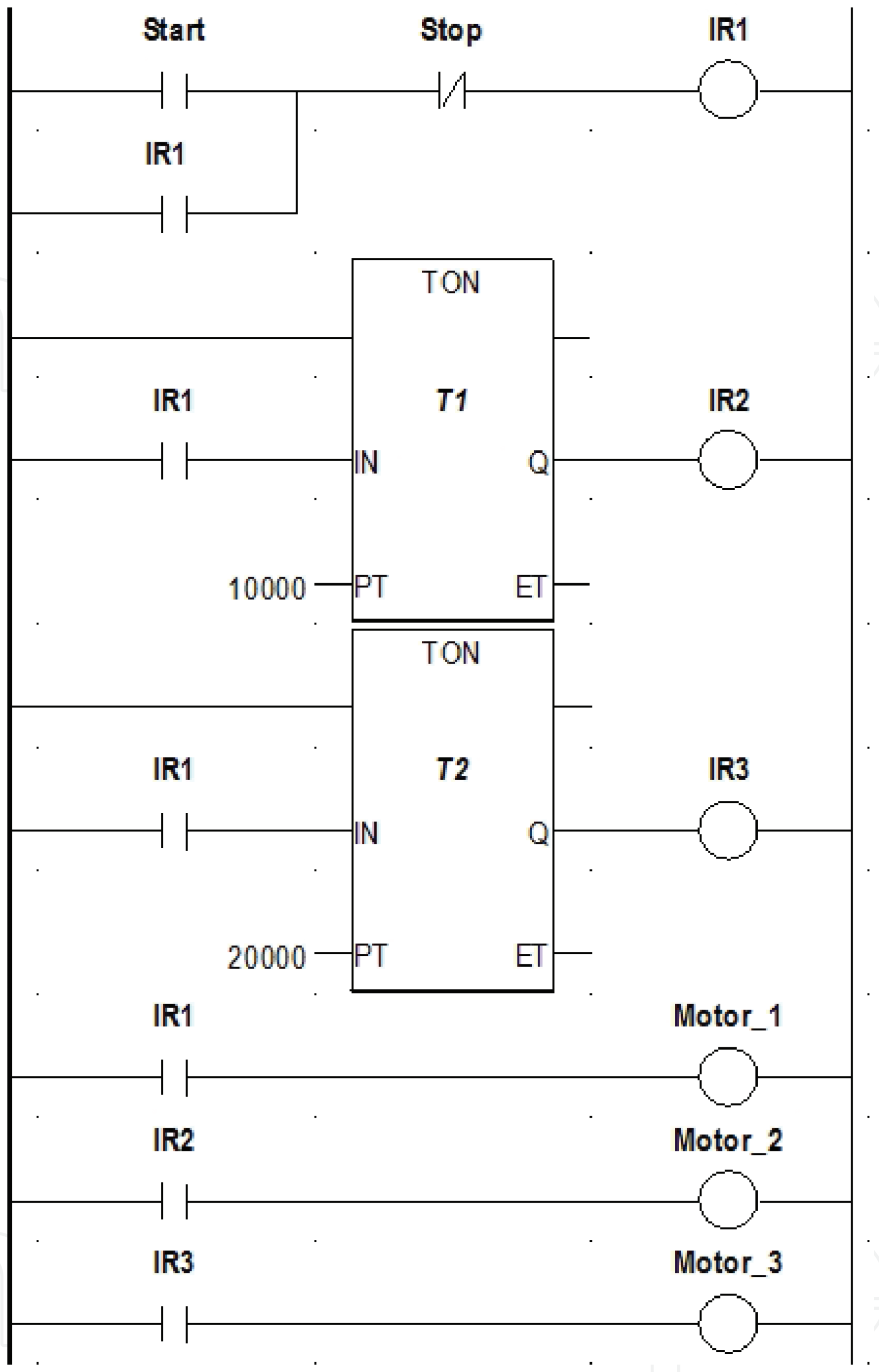

Figure 21. Run of three motors sequentially.

then, the IR3 and Motor_3 coils are energized. If Stop = 1, then, the IR1 coil is de-energized, and are restart conditions of the control algorithm. **Table 2** shows the equivalence of signals in function of the definition LDPN.

Figure 22 shows the LDPN to control algorithm of run of three motors sequentially. Restart conditions of the output places are in function of B_1^c by Eq. 19. The restarting condition of place B_1 is in function of input places I_1 and I_2 by Eqs. 19 and 20. Restart condition places RC_2 to RC_6

LD	**LDPN**
Start	I_1
Stop	I_2
Motor_1	O_1
Motor_2	O_2
Motor_3	O_3
IR1	B_1
IR2	B_2
IR3	B_3

Table 2. Equivalence of signals of control algorithm in LDPN.

are in function of the marking B_1^o, which are connected from $\mathbf{B_1^c}$. For complex control algorithms implies a larger graphic LDPN, it is advisable to indicate the marking function of the places *RC*. Eq. 21 shows the incidence matrix of the PN model respectively, where the conditioning *if-then* of transitions for reasons of space, which are indicated on corresponding figures, is omitted.

$$
\mathrm{A_{ij}} = \left[\begin{array}{c|cccccccccccccccccccccccccccc}
 & I_1 & I_1^o & I_2 & I_2^c & I_2^c & B_1 & B_1^o & B_1^o & B_1^o & B_1^o & AI_1 & AI_2 & AI_3 & AI_4 & AI_5 & AI_6 & O_1 & O_2 & O_3 & RC_1 & RC_2 & RC_3 & RC_4 & RC_5 & RC_6 \\
\mathbf{I_1^o} & -1 & 1 & & & & & & & & & & & & & & & & & & & & & & & \\
\mathbf{I_2^c} & & & -1 & 1 & 1 & & & & & & & & & & & & & & & & & & & & \\
\mathbf{B_1^o} & & & & & & -1 & 1 & 1 & 1 & 1 & & & & & & & & & & & & & & & \\
\mathbf{B_1^c} & & & & & & -1 & & & & & & & & & & & & & & & 1 & 1 & 1 & 1 & 1 \\
\mathbf{L_1} & & -1 & & -1 & & 1 & & & & & & & & & & & & & & & & & & & \\
\mathbf{L_2} & & & & & -1 & 1 & -1 & & & & & & & & & & & & & & & & & & \\
\mathbf{L_3} & & & & & & & & -1 & & & & & & & & & 1 & & & & & & & & \\
\mathbf{AI_1} & & & & & & & & & -1 & & \mathrm{BT}=1\mathrm{ms} & \mathrm{ET}+\mathrm{BT} & & & & & & & & & & & & & \\
\mathbf{AI_2} & & & & & & & & & & & & \mathrm{ET}+\mathrm{BT} & \mathrm{PT}=10^4 & & & & & 1 & & & & & & & \\
\mathbf{AI_3} & & & & & & & & & -1 & & & & & \mathrm{BT}=1\mathrm{ms} & \mathrm{ET}+\mathrm{BT} & & & & & & & & & & \\
\mathbf{AI_4} & & & & & & & & & & & & & & & \mathrm{ET}+\mathrm{BT} & \mathrm{PT}=2{*}10^4 & & & 1 & & & & & & \\
\mathbf{RC_1} & & & & & & -1 & & & & & & & & & & & & & & -1 & & & & & \\
\mathbf{RC_2} & & & & & & & & & & & & & & & & & -1 & & & & -1 & & & & \\
\mathbf{RC_3} & & & & & & & & & & & & & & & & & & -1 & & & & -1 & & & \\
\mathbf{RC_4} & & & & & & & & & & & & \mathrm{ET} & & & & & & & & & & & -1 & & \\
\mathbf{RC_5} & & & & & & & & & & & & & & & & & & & -1 & & & & & -1 & \\
\mathbf{RC_6} & & & & & & & & & & & & & & & \mathrm{ET} & & & & & & & & & & -1
\end{array}\right]
$$

The dynamic behavior of the PN model by run of three motors sequentially is described by the following marking. Fired the transitions with inhibitor arcs, initial marking $\mathrm{M_0}$ is:

$$
M_0 = \left[\begin{array}{ccccccccccccccccccccccccc}
I_1 & I_1^o & I_2 & I_2^c & I_2^c & B_1 & B_1^o & B_1^o & B_1^o & B_1^o & AI_1 & AI_2 & AI_3 & AI_4 & AI_5 & AI_6 & O_1 & O_2 & O_3 & RC_1 & RC_2 & RC_3 & RC_4 & RC_5 & RC_6 \\
0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 & 1ms & 0 & 10^4 & 1ms & 0 & 2{*}10^4 & 0 & 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1
\end{array}\right] \tag{21}
$$

If place I_1 has token, which enable the $\mathbf{I_1^o}$ transition, its fire puts a token in the place I_1^o, which enabled the $\mathbf{L_1}$ transition, its fire puts a token in the place B_1. In these conditions, by Eq. (16), the tokens in places of restarting conditions are consumed; the marking corresponding of LDPN is shown in Eq. (23).

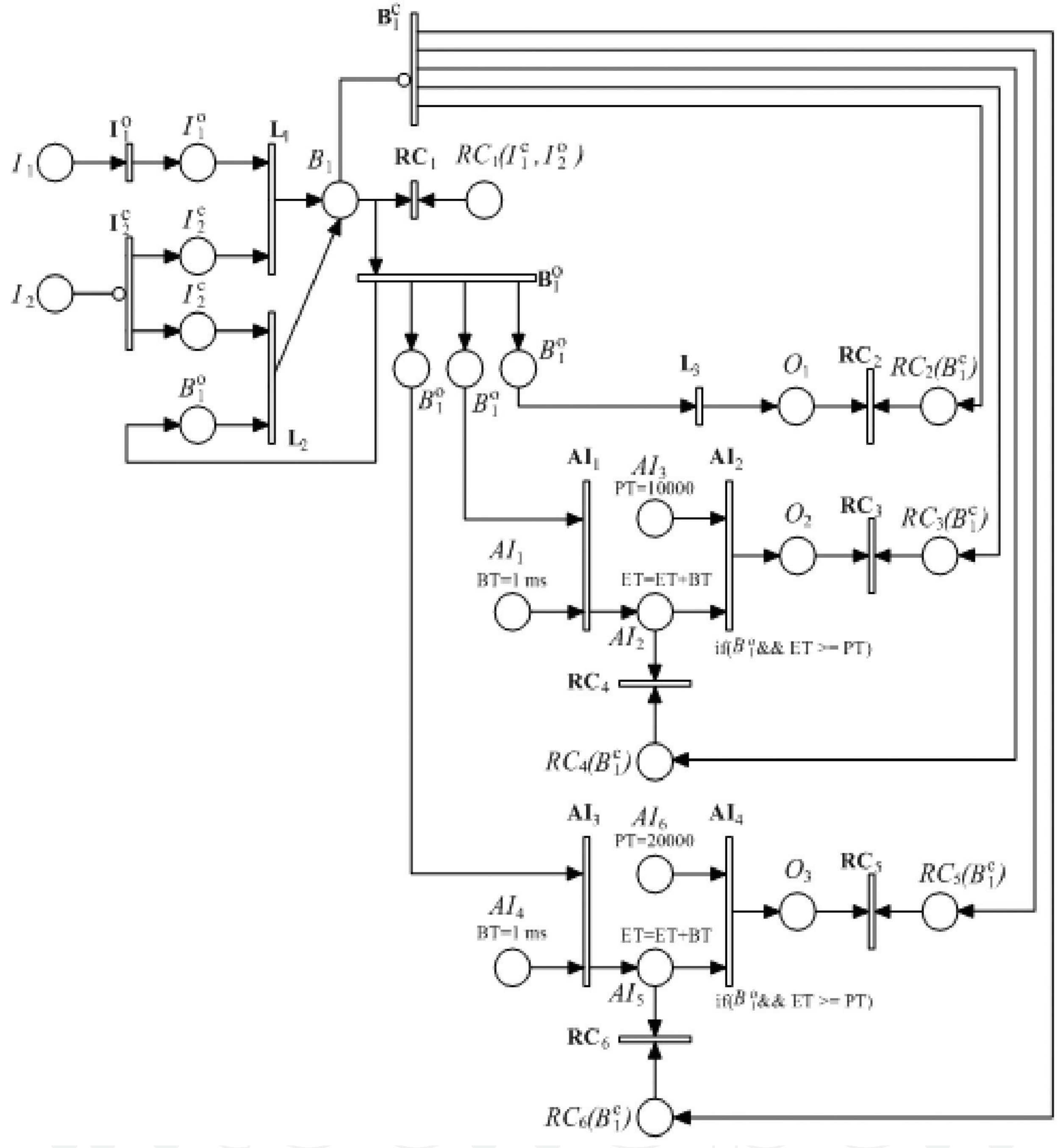

Figure 22. PN model of run of three motors sequentially.

$$M_1 = \begin{bmatrix} I_1 & I_1^o & I_2 & I_2^c & I_2^c & B_1 & B_1^o & B_1^o & B_1^o & B_1^o & AI_1 & AI_2 & AI_3 & AI_4 & AI_5 & AI_6 & O_1 & O_2 & O_3 & RC_1 & RC_2 & RC_3 & RC_4 & RC_5 & RC_6 \\ 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 & 1ms & 0 & 10^4 & 1ms & 0 & 2*10^4 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{22}$$

In these conditions, the $\mathbf{B_1^o}$ transition is enabled, its fire puts a token in four places B_1^o, this enables $\mathbf{L_2}$ transition, its fire puts a new token in the place B_1, it disables the fire of $\mathbf{L_1}$ and $\mathbf{L_2}$ transitions by Eqs. (17) and (18). Another place B_1^o enables $\mathbf{L_3}$ transition; its fire puts a token in the place O_1. The others two places B_1^o enable $\mathbf{AI_1}$ and $\mathbf{AI_3}$ transition to add the base time, respectively. Eq. (24) shows these conditions of LDPN, besides the update marking.

$$M_1 = \begin{bmatrix} I_1 & I_1^o & I_2 & I_2^c & I_2^c & B_1 & B_1^o & B_1^o & B_1^o & B_1^o & AI_1 & AI_2 & AI_3 & AI_4 & AI_5 & AI_6 & O_1 & O_2 & O_3 & RC_1 & RC_2 & RC_3 & RC_4 & RC_5 & RC_6 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 & 1 & 1 & 1 & 1ms & Add & 10^4 & 1ms & Add & 2*10^4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \tag{23}$$

In these conditions, if $ET \geq PT$, in both $\mathbf{AI_2}$ and $\mathbf{AI_4}$ transitions put a token in places O_2 and O_3, respectively. When place I_2^o has a token enabling the $\mathbf{RC_1}$ transition, its fire consumes a token in the place B_1, restarting condition in LDPN.

5. Conclusions

There are two types of control lines for discrete event systems: those with discrete inputs and outputs, and those with analog inputs and discrete output. Twelve logics that were analyzed and converted into Petri network models.

For dynamic behavior of the PN model proposed, constraints and equations for marking places and firing transitions are indicated to consider the problems of mark accumulation and the restarting condition of the structure PN.

LDPN to discrete event systems allow to model control lines used in LD language, and consequently, control algorithms development in LD, supporting that these are safe and reliable.

Each PN model is independent and can be interconnected in function of the control logic, as well as, the number of PN model that is needed can be integrated.

Author details

José Carlos Quezada Quezada[1]*, Ernesto Flores García[1], Joselito Medina Marín[2], Jorge Bautista López[3] and Víctor Quezada Aguilar[1]

*Address all correspondence to: jcarlos@uaeh.edu.mx

1 High Education School Tizayuca, Autonomous University of Hidalgo State, Mexico

2 Advanced Research Center in Industrial Engineering, Autonomous University of Hidalgo State, Mexico

3 Campus Zusmpango, Autonomous University of Mexico State, Mexico

References

[1] International Electrotechnical Commission, IEC 61131-3: Programmable Controllers: Programming Languages, International standard, 2nd ed, 2003

[2] Korotkin S, Zaidner G, Cohen B, Ellenbogen A, Arad M, Cohen Y. A petri net formal design methodology for discrete-event control of industrial automated systems, IEEE 26-th convention of electrical and electronics engineers in Israel; 2010. pp. 431-435. DOI: 10.1109/EEEI.2010.5662187

[3] John K-H, Tiegelkamp M. IEC 61131–3: Programming Industrial Automation Systems. 2nd ed. Springer; 2010

[4] Murata. Petri Nets: Properties, analysis and applications. Proceedings of the IEEE. 1989. pp. 541-580. DOI: 10.1109/5.24143

[5] Luo J, Zhang Q, Chen X, Zhou MC. Modeling and Race Detection of Ladder Diagrams via Ordinary Petri Nets. IEEE Transactions on Systems, Man and Cybernetics. DOI: 10.1109/TSMC.2016.2647219

[6] Năvrăpescu V, Deaconu I-D, Chirilă A-I, Deaconu A-S. Petri Net versus Ladder Diagram for Controlling a Process Automation. The 8th International symposium on advanced topics in electrical engineering. May 23–25, Bucharest, Romania, 2013. DOI: 10.1109/ATEE.2013.6563402

[7] Xuekum C, Lilian L, Pengfei Q. Method For Translating Ladder Diagram To Ordinary Petri Nets. 51st IEEE conference on decision and control; 2012. pp. 6716-6721. DOI: 10.1109/CDC.2012.6426901

[8] Zhang H, Jiang Y, Hung WN, Yang G, Gu M, Sun J. New strategies for reliability analysis of programmable logic controllers. Mathematical and Computer Modeling. 2012;**55**(7/8): 1916-1931. DOI: 10.1016/j.mcm.2011.11.050

[9] da Silva Oliveira EA, da Silva LD, Gorgonio K, Perkusich A, Martins AF. 9th IEEE international conference onObtaining formal models from ladder diagrams, industrial informatics (INDIN), 26-29 July, 2011; p. 796-801. DOI: 10.1109/INDIN.2011.6034994

[10] Lee J, Lee JS. Conversion of ladder diagram to petri net using module synthesis technique. International Journal of Modeling and Simulation. 2009;**29**(1):79-88. DOI: 10.1080/02286203.2009.11442513

[11] Grobelna I, Grobelny M, Adamski M. Petri Nets and Activity Diagrams in Logic Controller Specification—Transformation and Verification. 17th International Conference Mixed Design of Integrated Circuits and Systems, Wroclaw, Poland; 2010. pp. 607-612

[12] Quezada JC, Medina J, Flores E, Seck Tuoh JC, Solís AE. Simulation and validation of diagram ladder – Petri net. International Journal Advance Manufacturing Technology. 2017;**88**:1393-1405. DOI: 10.1007/s00170-016-8638-9

[13] Quezada JC, Medina J, Flores E, Seck Tuoh JC, Hernández N. Formal design methodology for transforming ladder diagram to Petri nets. International Journal Advance Manufacturing Technology. 2014;**73**:821-836. DOI: 10.1007/s00170-014-5715-9

[14] Bolton W. Programmable Logic Controllers. 5th ed. Elsevier Ltd.; 2009. pp. 222-223, ISBN: 978-1-85617-751-1