IMPLEMENTATION OF PETRI NETS INTO PLC BY USING PROGRAM SKETCHER

Luděk Chomát
Doctoral Degree Programee (1), FEEC BUT
E-mail: xchoma00@stud.feec.vutbr.cz

Supervised by: Petr Pivoňka
E-mail: pivonka@feec.vutbr.cz

ABSTRACT

Petri net is a graphic and mathematical tool for discrete and continuous system simulation and analysis. It is mostly used in domain of sequential control. Network states are often interconnected with inputs and outputs programmable logic controller that allows to control the real technological process. This document describes how to implement Petri net into the programmable logic controller. In this essay we introduce one of many available tools used to create Petri nets.

1. INTRODUCTION

In this chapter we will deal with implementation of Petri nets into programmable logic controller. Programmable logic controller is manufactured by company B&R but Petri net can be implemented on any platform. The Petri net can be implemented into the PLC in numeruous ways. We'll discuss one of them. Furthermore, special programme called Sketcher used for creation of Petri net and for PLC implementation will be shown. The last chapter deals with experimental verification of Petri net shown on an example. The Model transfer variant was chosen for PLC implementation (see *Figure 1*). Another suitable Petri net implementation is the Comparative model (shown at *Figure 2*).

1.1. TRANSFER MODEL

Model created on PC is transformed by an appropriate way into data structure that is transferred into the memory of programmable logic controller. After this, a command is sent to activate the service program that is implemented into programmable logic controller. Then this program performs Petri nets simulation autonomicaly. Because the simulation consists in fact only of integer number vectors addition, it is easy to program it on programmable logic controller .

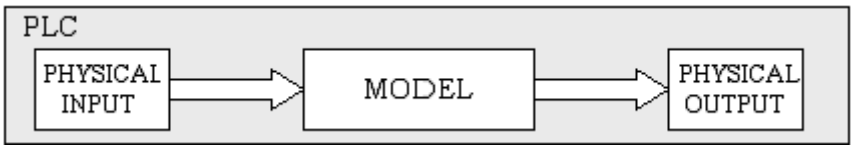


Figure 1: Petri net implementation

1.2. COMPARATIVE MODEL

The comparative model serves for control and comparison of theoretical and real result. The model is loaded both on programmable logic controller and on PC. Results are compared in certain intervals. Thanks to this comparison, we can avoid some collapses that could raise from a wrong controllers function.

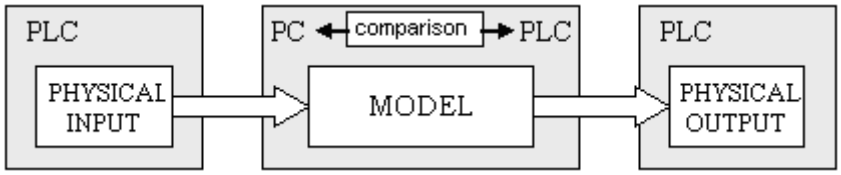


Figure 2: Petri net implementation

2. SIMULATION CORE DESCRIPTION

To implement the simulation core it was chosen to apply possibility C *(transfer model)*. This model is created on PC using program *Sketcher*. This model is transformed in a data structure that is transferred into programmable logic controller. Program *Sketcher* is a graphic tool on PC which serves for creation of Petri nets. The most important component of program *Sketcher* is the simulation core which processes Petri nets represented by a data structure. The simulation core is also uploaded on target platform. Its purpose there is to realize simulation of Petri net to its data from data structure. The flow diagram of simulation core is described in my thesis [1]. Simulation core performs all calculations concerned with Petri net simulation like placing tokens into places, starting P – timed, T – timed and firing transitions.

Petri nets data structure is saved on PC into a file and this file is transferred e.g. using FTP client on programmable logic controller. All information about Petri net are saved into a file. There are shown basic information in *table 1* that are contained in data structure (in created file). In fact this data structure can be viewed as in a simple script in specialized programming language for description of Petri nets.

Function	Function description	Example
AddPlace	Create place in net	*AddPlace 1 &DO_01 -1 ;*
AddToken	Create token in place	*AddAva 1 0 500 1 ;*
AddAva	Create P-timed in place	*AddTrans 1 &DI_01 0 ;*
AddTrans	Create transition in net	*AddTrans 1 &DI_01 0 ;*
AddFir	Set under which condition can be transition fired	*AddFir 1 0 2 1000 1 ;*
Connect	Connect places and transitions	*Connect 1 1 1 0 ;*

Table 1: List of Petri net scripting language functions

3. SKETCHER DEVELOPMENT ENVIRONMENT DESCRIPTION

This chapter deals with the simulation programme used to design Petri nets, its simulation and visualization. Programme Sketcher is based on work [1]. Simulation programme was created in MICROSOFT VISUAL C++ .NET 2003, where is also possible to use libraries MFC [5]. This library helped us to create better visualization. I'm sure this will be rewarded by every user.

3.1. PROGRAMME DESCRIPTION AND CONTROL

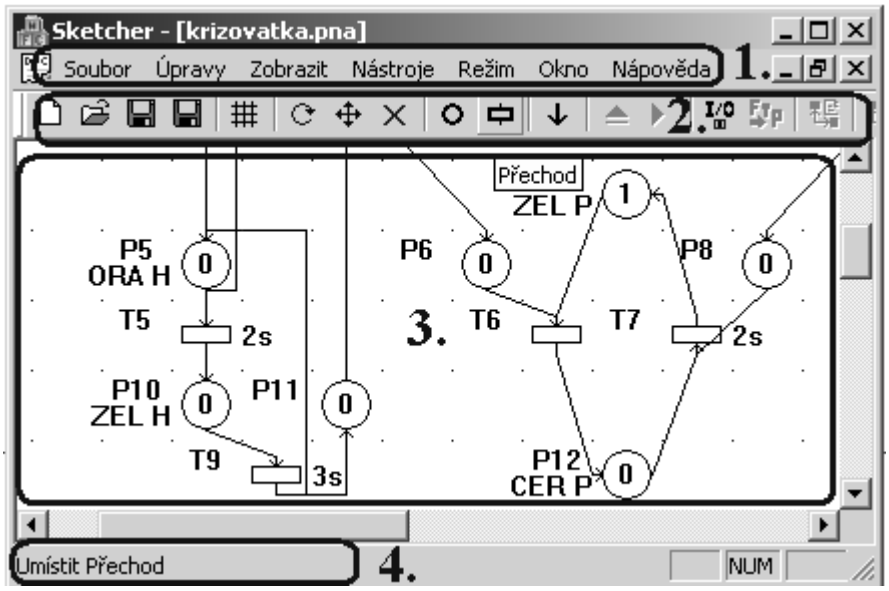
Main objects during the graphical user interface programming were transparency and controls simplicity. To control the programme we can use main menu or to work faster there is also variety of icons in „Control Panel“. Keyboard shortcuts are also suitable. *Figure 3* describes Sketcher showing Petri Net designed to control traffic lights.(*chapter 4*)

3.2. PETRI NET POSSIBLE PROPERTIES IMPLEMENTED INTO PLC

Table 2 shows overview of all possible Petri nets implementations into programmable logic controller. All nets are fully functional a simulation core is designed to be extended easily. Book [2] describes Petri net's all possible features.

Property	Autonomous	Non-Autonomous	Bounded	Non-Bounded	Safe
Realized	YES	YES	YES	YES	YES
Property	Non-Contact	Live	Inhibitor	Generalized	P-timed
Realized	YES	YES	NO	NO	YES
Property	T-timed	Effective conflict		Coloured PN	
Realized	YES	YES		YES	

Table 2: Property Petri net in simulation core



1 – Main menu

2 – Control panel

3 – Application workspace

4 – Status bar

Figure 3: Show programme for draft, simulation and visualization Petri net

3.3. POSSIBLE SKETCHER MODES

There are three modes of Petri net operations in the simulation programme. This is the Petri net design mode which enables the creation of any Petri net based on our specifications using the design-desktop. Other modes are the Petri net simulation mode where one can simulate the Petri net before it gets implemented into the target platform. Thanks to this

mode user can reveal errors before they appear in a real process. The last mode serves for Petri net visualization. The Sketcher programme is connected to the target platform and reads or writes the input/output values of the PLC in a certain interval. Read information is displayed on the desktop and the user can fine-tune the programme according to the specifications. (See *Figure 4* – denoted area and transition are active PLC inputs/outputs).

Figure 4: Visualization Petri net with s inputs/outpusts PLC

4. EXAMPLE

This chapter describes the usage of cross-roads feed device using our Petri net scripting language *figure 5*. *Figure 6* describes example of Petri net and also its data structure (*table 3*), which is implemented into programmable automat. The main function is to operate crossroad lights automatically.

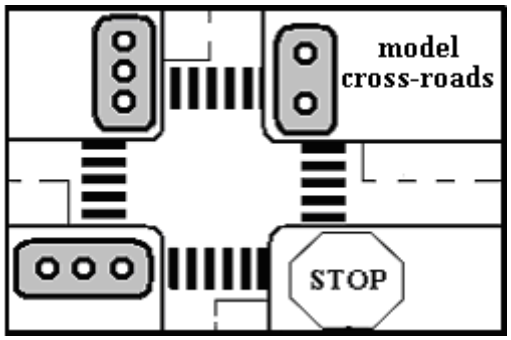


Figure 5: Model cross-roads

```
AddPlace 1 DO_01 -1 ;
AddPlace 2 NULL -1 ;
AddPlace 3 NULL -1 ;
AddPlace 4 &DO_04 -1 ;
AddPlace 5 &DO_02 -1 ;
AddPlace 6 NULL -1 ;
AddPlace 7 &DO_08 -1 ;
AddPlace 8 NULL -1 ;
AddPlace 9 &DO_05 -1 ;
AddPlace 10 &DO_03 -1 ;
AddPlace 11 NULL -1 ;
AddPlace 12 &DO_07 -1 ;
AddPlace 13 &DO_06 -1 ;
AddPlace 14 NULL -1 ;

AddToken 1 0 1 ;
AddToken 7 0 1 ;
AddToken 13 0 1 ;

AddTrans 1 NULL 0 ;
AddTrans 2 NULL 0 ;
AddTrans 3 NULL 0 ;
AddTrans 4 NULL 0;
AddTrans 5 NULL 0 ;
AddTrans 6 NULL 0 ;
AddTrans 7 NULL 0 ;
AddTrans 8 NULL 0 ;
AddTrans 9 NULL 0 ;
AddTrans 10 NULL 0;

AddFir 1 0 2 200 1 ;
AddFir 2 0 2 200 1 ;
AddFir 3 0 1 200 1 ;
AddFir 4 0 1 200 1 ;
AddFir 5 0 1 200 1 ;
AddFir 6 0 1 0 0;
AddFir 7 0 1 200 1 ;

AddFir 8 0 1 200 1 ;
AddFir 9 0 1 300 1 ;
AddFir 10 0 1 300 1;

AddAva 1 0 0 0 ;
AddAva 2 0 0 0 ;
AddAva 3 0 0 0 ;
AddAva 4 0 0 0 ;
AddAva 5 0 0 0 ;
AddAva 6 0 0 0 ;
AddAva 7 0 0 0 ;
AddAva 8 0 0 0 ;
AddAva 9 0 0 0 ;
AddAva 10 0 0 0 ;
AddAva 11 0 0 0 ;
AddAva 12 0 0 0 ;
AddAva 13 0 0 0 ;
AddAva 14 0 0 0 ;

Connect -1 1 1 0 ;
Connect 1 1 3 0 ;
Connect -1 5 3 0 ;
Connect 1 5 1 0 ;
Connect 1 5 5 0 ;
Connect -1 10 5 0 ;
Connect 1 10 9 0 ;
Connect -1 5 9 0 ;
Connect -1 11 9 0 ;
Connect 1 11 1 0 ;
Connect -1 2 1 0 ;
Connect -1 4 2 0 ;
Connect 1 4 4 0 ;
Connect -1 9 4 0 ;
Connect 1 9 8 0 ;
Connect -1 13 8 0 ;

Connect 1 13 10 0 ;
Connect -1 14 10 0
;
Connect 1 14 2 0 ;
Connect 1 9 2 0 ;
Connect -1 9 10 0 ;
Connect -1 3 2 0 ;
Connect 1 2 4 0 ;
Connect 1 3 3 0 ;
Connect -1 12 6 0 ;
Connect 1 12 7 0 ;
Connect -1 7 7 0 ;
Connect 1 7 6 0 ;
Connect -1 6 1 0 ;
Connect 1 6 7 0 ;
Connect -1 8 2 0 ;
Connect 1 8 6 0 ;
```

Table 3: Script for cross-roads

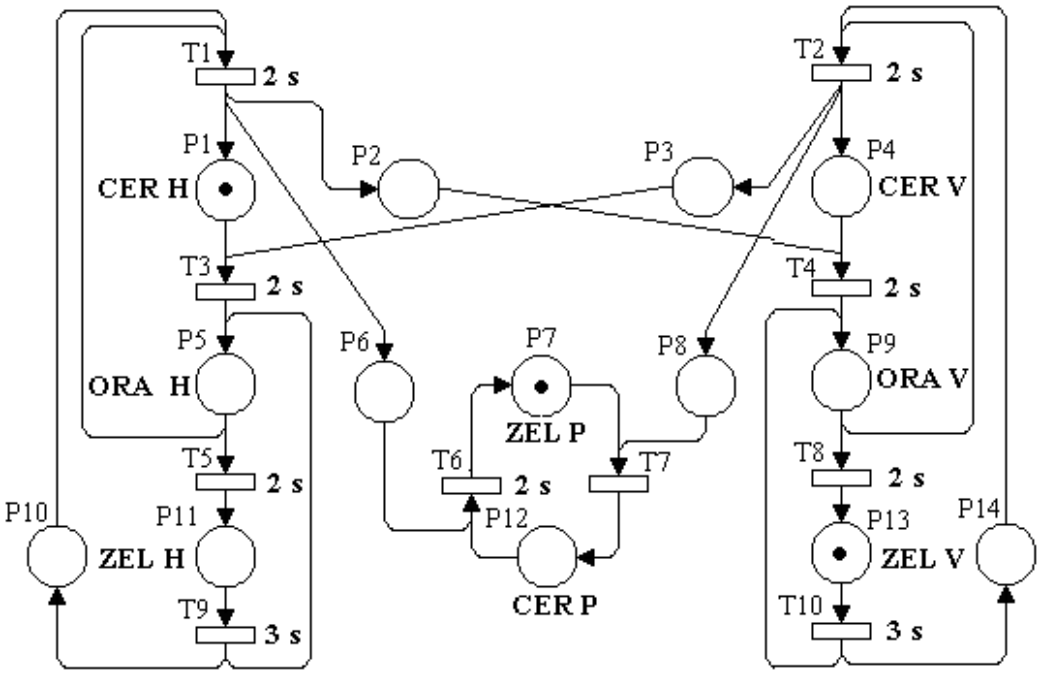


Figure 6: Petri net for cross-roads

5. CONCLUSION

In this study we described how to implement Petri net into programmable logic controller. One of the possible implementations is to create Petri net script on PC and import it into PLC. The advantage of this possibility is that we can test and verify Petri net on PC before its implementation on target platform. The next possible option is comparative model. Thanks to this method we can monitor the Petri net and compare results. It is appropriate for safety and reliability of created Petri net. Next step is introduction of a special program Sketcher. Our goal is to find innovation property control algorithm.

ACKNOWLEDGEMENT:

The paper has been prepared as a part of the solution of Czech Science Foundation GAČR project No. 102/06/1132 Soft Computing in Control and by the Czech Ministry of Education in the frame of MSM MSM0021630529 Intelligent Systems in Automation.

REFERENCES

[1] CHOMÁT, L.: Petri nets in environment of programmable logic controller B&R, Diploma thesis, VUT Brno, 2006

[2] ALLA, H.: Discrete, Continuous and Hybrid Petri Nets, France, 2005

[3] Automation Studio™ Programming. B&R Automation, 2001

[4] IEC (International Electrotechnical Commission). IEC Standard 61131-3: Programmable controllers – Part 3, 1993.

[5] PROSISE, J.: Programming Windows with MFC, Computer Press Praha, 2000, http://www.cpress.cz