Real - Time Implementation of Petri Nets into PLC

PETR PIVOŇKA, LUDĚK CHOMÁT
Department of Automatic Control and Instrumentation
Brno University of Technology
Kolejní 2906/4, 612 00 Brno
CZECH REPUBLIC
pivonka@feec.vutbr.cz, xchoma00@phd.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 controlling 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. This system allows design, simulation, visualization and immediate Petri net implementation into any platform. Our goal is to find out essential improvement in controlling algorithms.

Key-Words: - Petri net, PLC, Implementation, Hybrid Petri net, Real – Time, Sketcher, Control algorithm hybrid dynamic System.

1 Introduction

We will deal with immediate hybrid Petri net [2] implementation into programmable automat. This automat is manufactured by B&R, but Petri net can be imported into any existing platform. This implementation can be done with several ways, e.g.: Existing Petri net is in PC, and only requested inputs/outputs are synchronized with programmable automat. Part of Petri net is stored in computer, the other part is in PLC, and meanwhile PC and PLC cooperate.
Other possibility how to implement into PLC is comparative model and transfer model. Implementation method into programmable automat can be described in four steps. Next, simulation core for straight Petri net implementation into PLC will be described in detail. The Sketcher program, used to create Petri nets and implement it into programmable automat, is introduced furthermore. Following chapter deals with experimental verification of hybrid Petri net shown on an example. In the closing part we will summarize research results and outline further work orientation.

There are two possibilities to implement Petri net into programmable automat. First option is to transfer the model as depicted on figure 1. Second selected Petri net implementation is comparative model, shown on 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 automatically. Because the simulation consists in fact only of integer number vectors addition, it is easy to program it on programmable logic controller.

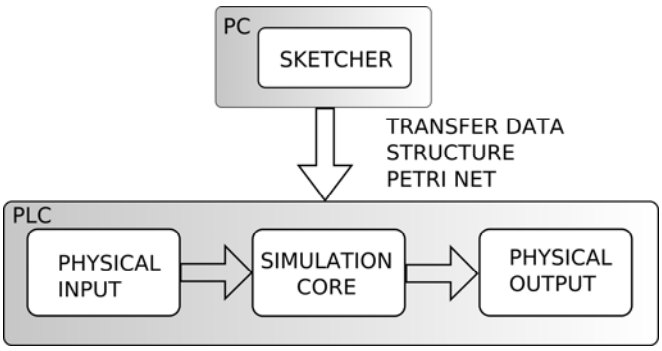

Figure 1: Hybrid system with Petri net verification

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 rise from a wrong controllers function

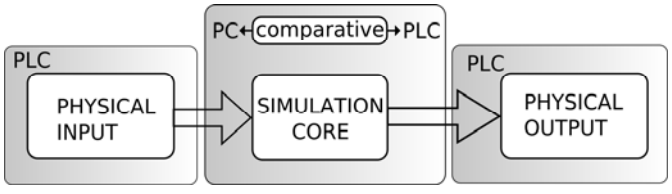

Figure 2: Hybrid system with Petri net verification

2 Direct Implementation into PLC

This chapter describes how to implement Petri nets into programmable automat. As mentioned before, the programmable automat is manufactured by B&R system 2005 Company, but in its principle Petri net can be implemented into any existing platform. As result, we are not connected to any company, not even on programmable automats. We can implement Petri net into some microchip or industrial computer etc. Before we can implement Petri net into PLC, it must be designed and tested. This prevents errors, which could later arise during real process administration. This process also increases the speed of fine-tuning in future hybrid Petri net control. We will use the Sketcher program, described in next chapter.

Next step is connecting PC with PLC through Ethernet or RS232 using PVI [7]. PVI is a modular environment for MS Windows supplied with B&R AUTOMATION NET framework. It provides communication interface between programmable automat B&R and user applications. It is possible to import data into other programs, like MS Excel, Word, even C++, Delphi etc. Nearly all external applications which use PVI, also communicate through basic library PVICOM (PVI Client/server). This interface is capable to react immediately and send many requests to manipulate into PVI Manager. PVI Manager is essential part of the system and is responsible for correct processing of all processed data and for correct task timing and their direction.

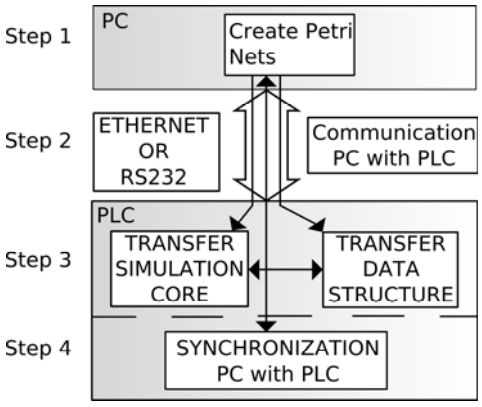

Figure 3: Hybrid implementation of Petri net

In third step we will send created hybrid Petri net with help of Sketcher program into programmable automat. In this programmable automat the Petri net is stored into memory and the simulation core reads this simple script and executes it. Simulation core description is the main topic of chapter 4. Now, the Petri net is activated and ready to manage any real process through the programmable automat inputs/outputs.

The last step was created to increase security and reliability. This means synchronization between PC and PLC because there is a reason to read physical input and output data. These values are sent in certain interval into PC, where we can handle them, for example to visualize controlled process or to compare measured data with estimated data etc.

3 Sketcher Development Environment Description

This chapter deals with the simulation program used to design Petri nets, its simulation and visualization. Program Sketcher is based on work [1]. Simulation program was created in MICROSOFT VISUAL C++ .NET 2003, where is also possible to use libraries MFC [3]. This library helped us to create better environment for hybrid Petri net creation. I'm sure this will be appreciated by every user of this system. Program Sketcher is a graphical tool on PC which is used to create Petri nets. The Sketcher program was basically produced to design simple Petri net. Next development was adding more options and applications, for example Petri net options broadening, communication between PC and PLC, PC and PP etc. Next chapters describe Sketcher in detail.

3.1 Program Description and Control

Main objects during the graphical user interface programming were transparency and controls simplicity. To control the program we can use main menu or to work faster there is also variety of icons in „Control panel". Keyboard shortcuts are also suitable. User interface is projected to be similar to any common program. Thanks to this, common users do not have problems with introduction with the program and are quickly able to create Petri net and implement it into programmable automat. Figure 4 describes Sketcher showing Petri net designed to control traffic lights.

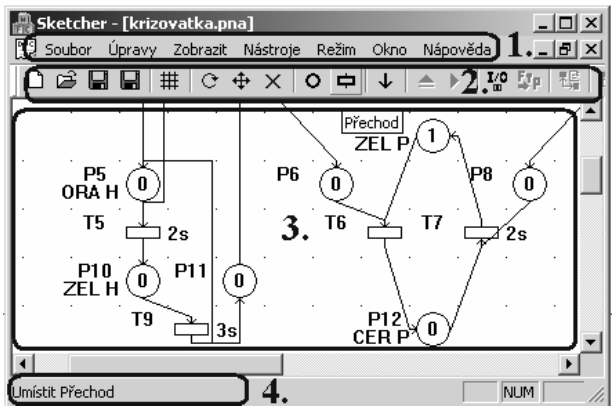

Figure 4: Draft, simulation and Petri net visualization program

3.2 Possible Petri Net Properties Implemented into PLC

Table 1 shows overview of all possible Petri nets implementations into programmable logic controller created in Sketcher simulation program. All properties are fully functional and simulation core is designed to easily extend Petri net options.

Property	Autonomous	Non – Autonomous
Realized	YES	YES
Property	Bounded	Non – Bounded
Realized	YES	YES
Property	Safe	Live
Realized	YES	YES
Property	Generalized	Inhibitor
Realized	YES	NO
Property	P – timed	T – timed
Realized	YES	YES
Property	Effective conflict	Discrete PN
Realized	YES	YES
Property	Continuous PN	Hybrid PN
Realized	YES	YES

Table 1: Petri net simulation core properties

3.3 Sketcher Modes

There are three available modes in the simulation program which allow working with Petri net. First is the Petri net design mode which allows creation of any Petri net based on our specifications. Other mode is the Petri net simulation mode where we 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 is used for Petri net visualization. The Sketcher program is connected to the target platform and is synchronized. In certain interval it reads the input and output values of the PLC. Read information's are displayed on the desktop and the user can watch or fine-tune the program according to his requests. (*Figure 5* – bold area and transition are active physical PLC inputs/outputs).

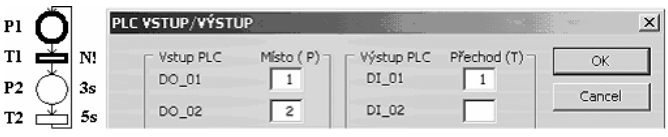

Figure 5: Petri net visualization with PLC inputs/outputs

4 Simulation Core Description

As mentioned, for implementation into PLC we picked the transfer model variant and comparative model. We will devote to the first variant (transfer model), because comparative model is only extension to the first variant. Petri net created in Sketcher is saved into *.pnd file. Created file carries all information's about properties and Petri net connections [4]. This data structure can be called as simple script language. This script can be taken as special program language for Petri net description. An example of the created file's content is shown in chapter 5. Very important part of Sketcher is its simulation core, which processes created Petri net script. Simulation core is designed to be imported into any platform. Before we copy the created file into PLC's memory, we must compile the simulation core into the target platform, which will run the Petri net. Simulation core is copied into the target platform only once and PLC is ready to read from file script for Petri net creation. If we want to change the Petri net, we have to copy file with new data into PLC. Picture 6 shows simulation core developmental diagram. Simulation core performs all necessary calculations for transition firing, placing tokens into places, starting P – timed, T – timed and firing transitions.

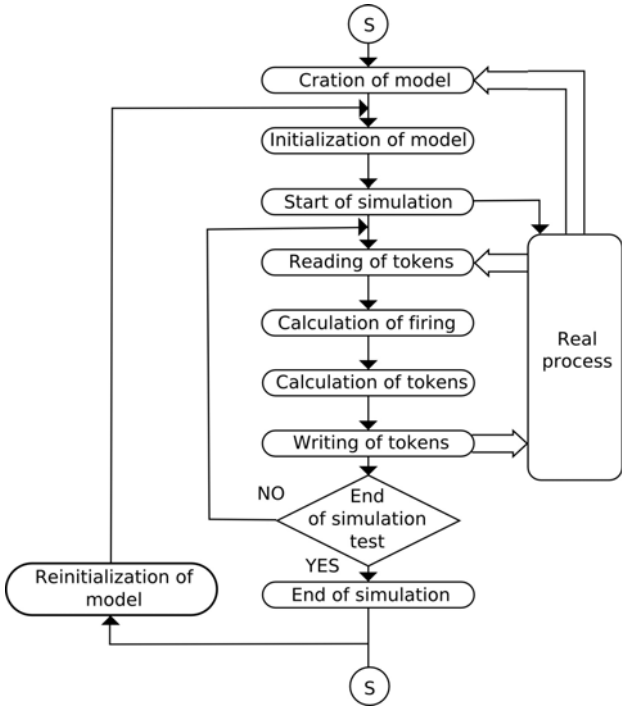

Figure 6: Simulation core flow diagram

Table 2 describes all functions necessary to create continuous, discrete or hybrid Petri net. Left column shows the name of the function, in the right column we can see detailed function description. Function and every element in the function are described under the table.

Near the function we can see an example of record in created script. Complete script for created hybrid Petri net is shown in created example in chapter 5.

Function	Function description
AddPlace	Create place in net
AddTrans	Create transition in net
AddToken	Create token in place
AddFP	Set for place
AddFT	Set under which condition can be transition fired
Connect	Connect places and transitions

Table 2: Petri net scripting language functions list

Description of Petri net scripting language functions:

*AddPlace(EN, *Plc, Capacity, PN),* where:
EN is number of place (P1, P2 . . . , P99)
**Plc* connection on real output from PLC (DO_01, DO_02 ... , DO_20)
Capacity – maximum number of tokens in place (-1, 0, 1 . . . 99)
PN – discrete or continuous Petri net (D, C)
AddPlace 5 &DO_04 -1 C – Created place P5 is connected with digital output 4 can have random number of tokens and it is continuous.

*AddTrans(EN, *Fev, Nega, PN)*, where:
EN is number of transition (T1, T2 . . . , T99)
**Fev* connection on real input of PLC (DI_01, DI_02 . . . DI_20)
Nega is negation of real input (0, 1)
PN – discrete or continuous Petri net (D, C)
AddTrans 3 N 0 D– Created transition T3 is not connected with any input and is discrete.

AddToken(P, C, N), where:
P presents number of transition (1 – 99)
C is color of net (0 – 3)
N number of tokens (0 – 99)
AddToken 5 0 2 – Two tokens are placed into place P5.

AddFP(P, C, PConst), where :
P presents number of transition (1 – 99)
C is color of net (0 – 3)
PConst is time constant
AddFP 8 0 520 – Place P8 is P-timed on 520 ms.

AddFT(T, C, P, TConst), where:
T presents number of transition (1 – 99)
C is color of net (0 – 3)
P is priority transition (0 – 2
TConst is time constant
AddFT 2 0 2 200 – Transition T2 has the biggest priority and is P-timed on 200 ms.

Connect (Dir, P, T, C, G), where:
Dir connection of place with transition or vice versa (-1, 1)
P is number of place (1 – 99)
T is number of transition (1 – 99)
C is color of net (0 – 3)
G is generalized Petri net (0 – 99)
Connect 1 9 2 0 3 – Place P9 is connected with transition T2 a edge has weight 3.

5 Example

This chapter shows the usage of all previously mentioned information's. For simple understanding we chose easy example, which is completely analyzed. Our goal is to create hybrid Petri net able to manipulate bottle filling. We have a 30 l tank, and we want to fill 0.25 l or 0.5 l bottles. In the beginning of every cycle we choose which content we want to fill. The bottles are filled in one working cycle after one chosen selection. Figure 7 describes simplified bottle filling model. Table 3 shows complete hybrid Petri net data structure. Figure 8 shows this created hybrid Petri net.

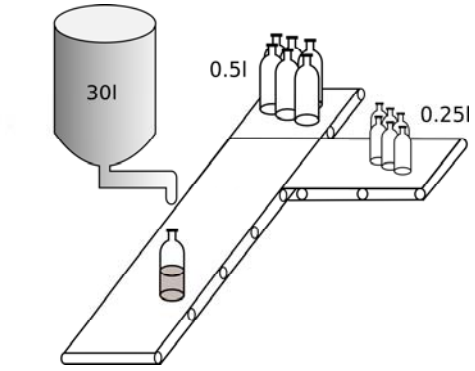

Figure 7: Selection based bottle filling model

Now we will explain bottle filling control. Thanks to transition T1 and T2 we can choose which bottle content we will fill. Let's say we activate input &DI_01 which is connected with transition T1. Token gets from P1 to P2. After transition P2 is activated necessary traffic stripes are launched to move the smaller bottles. After 4 s transition T3 is launched, because every needed condition for transition are complete. That means P4 shows the number of available bottles, P6 means the production line is ready to fill the bottles, and P9 that there is desired content of liquid for the bottle. After 10 s the bottle is filled and the token moves to place P7, where it signalizes number of full bottles. One working cycle finished and everything can start again.

AddPlace 1 N 1 D	AddFP 1 0 0
AddPlace 2 DO_01 -1 D	AddFP 2 0 0
AddPlace 3 DO_02 -1 D	AddFP 3 0 0
AddPlace 4 N -1 D	AddFP 4 0 0
AddPlace 5 N -1 D	AddFP 5 0 0
AddPlace 6 N -1 D	AddFP 6 0 800
AddPlace 7 N -1 D	AddFP 7 0 1000
AddPlace 8 N -1 D	AddFP 8 0 1000
AddPlace 9 N -1 S	AddFP 9 0 0
AddTrans 1 DI_01 0 D	Connect 1 1 1 0 1
AddTrans 2 DI_02 0 D	Connect 1 1 2 0 1
AddTrans 3 N 0 D	Connect 1 2 3 0 1
AddTrans 4 N 0 D	Connect 1 3 4 0 1
	Connect 1 4 3 0 1
AddToken 1 0 1	Connect 1 5 4 0 1
AddToken 4 0 3	Connect 1 6 3 0 1
AddToken 5 0 4	Connect 1 6 4 0 1
AddToken 8 0 2	Connect 1 9 3 0 025
AddToken 9 0 30	Connect 1 9 4 0 05
	Connect -1 2 1 0 1
AddFT 1 0 1 0	Connect -1 3 2 0 1
AddFT 2 0 1 0	Connect -1 1 3 0 1
AddFT 3 0 1 400	Connect -1 7 3 0 1
AddFT 4 0 1 800	Connect -1 1 4 0 1
	Connect -1 8 4 0 1

Table 3: Data structure for implementation into PLC

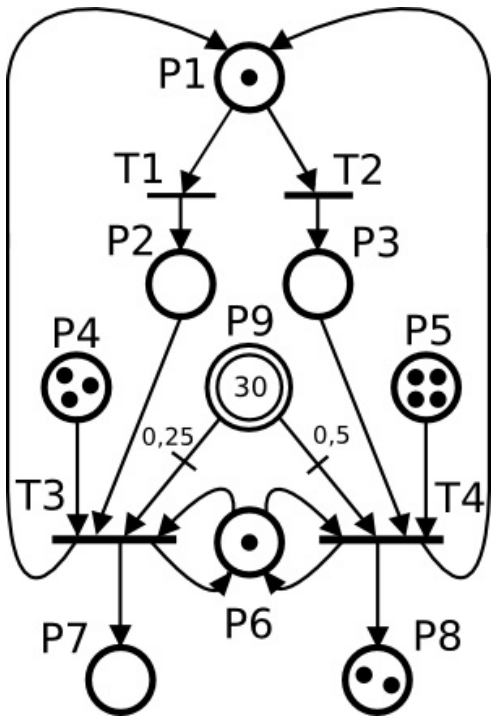

Figure 8: Hybrid Petri net used to control bottle filling

6 Conclusion

In this article we described how to implement Petri net into programmable logic controller. We chose two variants for direct implementation. First variant was selected because its simple and reliable Petri net implementation from PC into PLC. Second variant was chosen that we can control the Petri net during regulation in real process. By putting together these two variants we guaranteed safe and reliable Petri net performance, which nowadays is the most important priority in real process administration. We also described graphical program used to design Petri nets and also depicted its simulation core. Simulation core manages complete Petri net process no t only in PC, but also in target platform, where the Petri net is being implemented. In the end we described simple example just to understand the whole course of study. Of course, in real the Petri net would be much more comprehensive, but this was not the goal of this work. We wanted to point on created hybrid Petri net and equal data structure. As we can see in Table 3, this data structure is quite simple and well arranged. The object of this and following work is to find out substantial improvement in controlling algorithms options usable in real process

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. In Czech, VUT BRNO, 2006
[2] ALLA, H., *Discrete, Continuous and Hybrid Petri Nets*, France, 2005
[3] POSISE, J., *Programming Windows with MFC*, Computer Press Praha, 2000
[4] *Automation Studio Programming*, B&R Automation, 2001
[5] IEC (International Electrotechnical Commission), *IEC Standard 61131-3,* Programmable Controllers – Part 3, 1993
[6] PURI, A., *Theory of hybrid systems and event systems*, Phd thesis, Department of electrical engineering, Berkeley, 1995
[7] B&R Documentation. Automation Net/PVI Online Help, *AS240Ee*, 1997-2007