TAMPEREEN TEKNILLINEN YLIOPISTO
TAMPERE UNIVERSITY OF TECHNOLOGY

MIGUEL ANGEL MONTAGUD CATALA

FORMALIZATION AND VISUALIZATION OF AN AUTOMATIC PRODUCTION LINE

Master of Science Thesis

Examiners: Andrei Lobov
prof. Jose L. Martinez Lastra
Examiner and topic approved on 31 May 2017

ABSTRACT

MIGUEL ÁNGEL MONTAGUD CATALÁ: Formalization and Visualization of an Automatic Production Line
Tampere University of Technology
Master of Science Thesis, 56 pages, 28 Appendix pages
August 2017
Examiners: Andrei Lobov, Professor José Luis Martínez Lastra

Keywords: formalization, visualization, automatic production line, verification and validation, model-checking, closed-loop system, DEDS, TNCES, FASTorium, MNG.

Verification and Validation (V&Y) of control software is nowadays assuming great significance in manufacturing systems, for it has been finally understood that a thorough study on this subject could mean a considerable improvement in the efficiency of production processes. For this reason V&V has become a necessity due to the pressures of market demand.

Manufacturing companies tend to solve these market pressures by the use of testing. But it is not quite correct, due to the fact that testing is a heuristic methodology and it has not a scientific foundation.

This thesis proposes another different methodology –a method consisting in the abstraction of the controlled object in a formal representation, –better known as "formalization". By means of formalization much more system information can be obtained and be used in the improvement of the efficiency of production processes.

In this thesis the benefits of formalization are proven by the application of the methodology in a real case. It means that the formalization of a case study will be developed obtaining significant results that will prove their own benefits. After the formalization the system can be subjected to model-checking –where a lot of information can be extracted from. One of the results of this thesis is the obtaining of the state space and the timing diagram of the system. Furthermore in this thesis it is highlighted and exposed one of the possible applications of formal methods –the system simulation in a visual representation.

PREFACE

I must confess I was afraid before my journey from Spain to Finland started. But after five months in Tampere University of Technology I can say I did not take a wrong decision regarding my exchange destination –it was an unforgettable experience!

At the present time I have just finished my master thesis and I cannot believe it yet. I am only a step away from accomplishing my dream, and that would not have been possible without the help of certain people to whom I dedicate this thesis.

First of all I would like to thank professor José Luis Martínez Lastra, who gave me the opportunity to make my master thesis under his supervision. He also offered me a place in FAST and provided with everything I needed –something I will always be grateful for. I would also like to thank him for his help through the whole semester as well as for the confidence he placed in me

In addition I would want to outline the work of my other supervisor, Andrei Lobov. He was very patient and attentive to me. Moreover, the knowledge I acquired from his wisdom is priceless.

Both of them made it possible for me to get to know another field in Industrial Automation I was not aware of yet and that proved to be very interesting for me.

Thanks to all my office mates, in special to Borja and Amalia –who made me feel at home. They were always willing to offer me the help I needed, and made my stay in FAST was more than pleasant as well.

I would also like to thank Anne and Matti for their friendliness and assistance. It was an honor for me to work next to them.

I am willing to see all these nice people again in some other moment of my live. I am sure everything will go well for each of them.

Finally I would like to dedicate this thesis to my parents, Miguel Ángel and Ana, because this fantastic adventure I lived never would have possible without their support, and I am aware my long staying abroad was not an easy thing for them.

Thank you very much!

CONTENTS

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ASRS	Automatic Storage and Retrieval System
AS-interface	Automatic Single interface
CNC	Computer Numerical Control
DC	Direct Current
DEDS	Discrete Event Dynamic Systems
HW	Hardware
IA	Industrial Automation
iMATCH	Integrated Model Assembler, Translator and CHecker
I/O(s)	Input(s) and Output(s)
LD	Ladder Diagram
MNG	MOVIDA NCES Generator
MPS	Modular Production Systems
OPC	Object Linking and Embedding for Process Control
PLC	Programmable Logic Controller
PN	Petri Nets
(T)NCES	(Timed) Net Condition/Event Systems
SFC	Sequential Function Chart
STL	Statement List
UML	Unified Modeling Language
V&V	Verification and Validation
XML	Extensible Markup Interchange

.

1. INTRODUCTION

1.1 Problem statement

Industrial Automation (IA) is the implementation of different technologies for the purpose of controlling and monitoring a process, dispositive or machine which, generally, executes functions or repetitive actions, so it can operate in an automatic way trying to avoid the human intervention.

Nowadays, (IA) is the fundamental pillar in the manufacturing systems. It is not only applied to machines, but also to the process management, services management and information management, improving any process in a more efficient way, since their installation, design, recruitment, maintenance, commercialization, etc. In addition, it is found in a lot of areas of economy, such as food manufacturing, automotive sector, pharmaceutical products, chemical products, plastic products, oil industry and telecommunications, among other industries which generate great benefits. It is a very competitive market and companies have to be equipped with the latest technology if they want to take part in it.

The main goal of IA is to produce as much quantity of product as possible within the shortest possible time, trying to reduce cost and with an acceptable quality. It is a very ambitious objective, but achievable. Each case should be thought out down to the smallest details in order to reach this objective in a successful way.

On the one hand, even though IA is a very advanced technology, it has still some imperfections, such as delay times, dead times, deadlocks, etc. There are small instants of time since a manipulated variable is changed until the system shows a response. Those failures, however small, could generate disruptions which in turn could result in the malfunction of the machine or even in its actual locking.

On the other hand, in most of productive companies, among the above-mentioned objectives of IA reducing time is the most desired one. Manufacturing companies want their machines to produce in the shortest possible time and in many cases they accomplish this goal by means of testing. In other words, they measure the times of the operation machines, change the automation control and measure the times again until they find the best control to achieve the best possible time without disturbing the main task of the machine or the process. This method is not based on theoretical justifications –it is a heuristic approximation. But there is other method which justifies in a theoretical way the results obtained and which is able to expose the controller to its verification,

validation and simulation. This method is the formalization of automated manufacturing systems.

This thesis is precisely about formalization –a method aimed to find the best functioning of a dispositive, machine or process, and unlike the testing, based on a mathematical justification that can be used to avoid changes produced by delay times and the possibility of deadlocks. In addition, the scope of this method is so wide that it can generate evolutions of the controlled object which can be observed in a visualization as a simulation.

In particular, this thesis concentrates on the formalization and visualization of a case study, the FASTorium, a small automatic production line.

1.2 Objectives

The main objective of this thesis is to formalize the automatic production line and simulate its evolution in a visualization. Formalization is an abstraction of the operation of the line in a mathematical way so it could be implemented in a computer and be subjected to model-checking. It is an alternative to testing; formalization is a little more laborious but it shows successful results and has a scientific basis proving them. The visualization is a digital representation of the line which evolves the same way as the system would operate.

In addition, the following secondary objectives should be noted:

- Validation of the final formalization of the line in order to be used later by other users. Once the formalization of the line is obtained, other users can check in the computer the well-functioning of their designed control software instead of by means of testing. Thus, control software are exposed to the Verification and Validation (V&V) in the model-checking, providing a lot of benefits.
- Assessment of the benefits of formalization in contrast to testing. The results obtained after the completion of the thesis will have to be able to provide more information and be more effectively justified than if obtained by means of testing.

The steps to achieve these objectives are: formalization of the plant, formalization of the control, creation of the closed-loop systems, realization of model-checking and visualization. The steps are reflected in a diagram in Figure 1. The design of the formal model plant in TNCES is extracted from the plant and the formal model controller in TNCES is extracted from the control software. Both models and the design of the visualization are manual and should be done by a person with knowledge of the modeled object. The closed-loop system results of the union of the model plant and model controller. It is when the system is ready to be subjected to model checking and to evolve the visualiza-

tion. The designs, the model checking and the evolution of the visualization have been developed by means of MOVIDA NCES Generator (MNG).

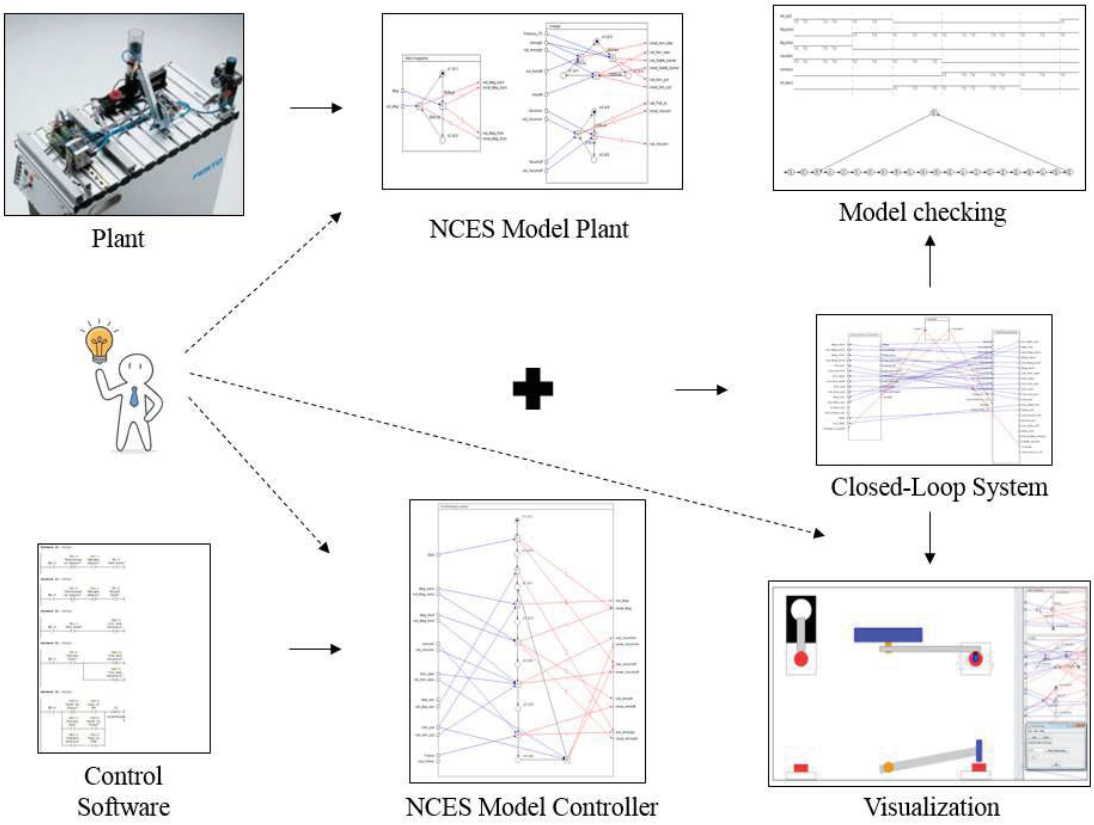


Figure 1. *Steps for the achievement of the aims of the thesis*

Hence, this thesis is within the framework of the studies achieved aiming to demonstrate the efficiency of formal methods in the verification and validation of the manufacturing systems. With this, it is studied an approach to this field and the proof of its utility against the trends which carried out in manufacturing factories nowadays.

1.3 Thesis Outline

This thesis is structured as follows:

Chapter 2 presents some previous researches done in the field of the importance of formal methods, and contains a review of the last studies in which formal methods were implemented. The first section of the chapter supports the reasons for using formal methods and sustains these reasons with some previous researches. The second section concentrates on why these methods did not have the expected success. Finally, the third section presents some similar cases taken as a reference in this thesis.

Chapter 3 provides the necessary theoretical background to understand the contents of this thesis. In this chapter, the main highlights are presented in the different sections. In

the first one, the concept of Discrete Event Dynamic Systems (DEDS) is introduced since the automatic production line of the case study of this thesis is considered one of these systems. DEDS are systems whose states take discrete values and could be represented as a succession which evolves according to an occurrence of events. This type of systems needs a special formalism to be represented, and so the second section concentrates on the modeling formalism used for the designing of the formal representations. This formalism is Timed Net Condition/Event Systems (TNCES) –an evolution of Petri Nets (PN) and Net Condition/Event Systems (NCES). Firstly a short introduction of PN and NCES explaining their main characteristics is exposed and secondly we focus on the TNCES formalism explaining the reason why it has been selected. The third section focuses on the closed-loop system modeling, the method that is going to be used to make the TNCES representations evolve. In this chapter, it is provided an explanation both of all the parts which compose this modeling and of what steps have been taken into account in the designing of the mentioned parts. These parts are: controller, plant and the interconnections between them (their inputs and outputs modules). Finally, the last section of the chapter is about MOVIDA TNCES Generator (MNG), the software used to formalize the production line and to visualize its evolution.

Chapter 4 contains the implementation of model-based verification of a real case, the FASTorium, a little line production composed of Modular Production Systems (MPS). In the first two sections of the chapter, both the whole production line and the operation of each module (station) that is a part of it are explained. The next three sections are based on the formalization of each station. The section devoted to controller modeling is a more generalized description, unlike the section dedicated to plant modeling more specific and detailed, interpreting each station separately due to the importance of the formalization of the plant. The last section of the chapter is about the closed-loop system modeling of the each station. Once this point is achieved, the system is ready to model-checking.

Chapter 5 focuses on the validation of the results obtained in chapter 4. The closed-loop system is subjected to model-checking. MNG provides automatically the state spaces and the timing diagrams of the each formal model. Once they are obtained, they can be validated and conclusions can be extracted. Likewise, a visualization design in the Simulator tool of MNG is created in order to make the simulation evolve with the formal models.

Chapter 6 concludes this thesis by analyzing the results of the work and drawing some conclusions from them. In this part, the aims reached are valued and analyzed. This chapter determines the next steps to take regarding the results obtained in this thesis and describes the possibility of future researches as well.

2. THE STATE OF THE ART

2.1 Why formal methods are unsuccessful

In recent years, there are very few studies on formal methods and its implementation in the Industrial Automation. Despite they have been widely developed along many years and their benefits in manufacturing systems have been proven, formal methods have not the expected acceptance among control engineers. It does not look an attractive theme and it is not yet too often recognized. This is so to such an extent that there are several surveys that identify why these type of methods are not being employed as often as they should. Examples of this surveys are [1], [2] and [18].

The first one, [1], explains that despite the activity within the academic community – and even the existence of some pioneering attempts of industrial test cases and commercial product development (for example ControlBuild Validation of TNI Software)- formal methods have not become a routine tool for control engineers yet. In [3], the same author tries to get a reason to answer why formal methods have are not really successful. According to this author the main reason is that -in order to be applied- it is required a profound knowledge of formal methods, and control engineers usually have little knowledge on how to formalize the development and validation processes. "Universities usually teach how to express a certain functionality in programming language without considering approaches to how to check if the program is written correctly. The most common approach at the moment is testing. Once a program has been written it is tested on the real controlled object. Studying and applying a formal method in validation and verification is relatively hard due to its complexity. As a result, formal methods of software validation and verification are beyond the scope of the control engineer" [3].

The second one, [2], also makes a wide research on this problem. This research bases their arguments on two hypothesis explaining why it is believed that formal methods did not succeed in their time. The first hypothesis is that industrial practitioners were reluctant to change their current methods and hence they overlooked the benefits that formal methods could provide. A single specification language could only describe a relatively small part of the system, and necessary tools were either not available, not compatible with other development tools, or too slow. The second hypothesis is that formal methods must overcome a number of relatively mundane but important practical hurdles before their benefits can be realized. These practical hurdles arise from the current state of software-development practice. While methods used in industry are rarely formally based, they are reasonably well-developed and understood. In order to be incorporated into industrial practice, formal methods must meet this current standard.

Furthermore, [18] identifies seven misconceptions why this field does not attract the attention of control engineers:

1. Formal methods ensure that the software is perfect. It creates unreal expectations and the idea that formal methods are closely an all-or-nothing approach.
2. Formal methods concentrate on proving correction. This has often the effect of formal methods seeming very difficult and not too much significant.
3. Formal methods are only useful for critic systems. This belief is based on the perception of the difficulty which involves the application of this kind of methods. Critic systems demand a more sophisticated use of formal methods, buy any systems can benefit from some techniques of the formal specification.
4. Formal methods require trained mathematicians and as long as they are based on mathematical notations this makes them seem difficult for the practice of software engineers.
5. Formal methods increase the development cost. It is supposed that the cost of using formal methods is very high, but actually it is convenient because it results in less maintenance cost of the software in the long term.
6. Formal methods are incomprehensible to the users. A formal specification is full of mathematical signs incomprehensible to any person who is not familiar to the notation.
7. Formal methods are not used in real big projects. They are often associated with academic departments and research organizations. It is supposed that only this organizations have the required capacity to implement formal methods and that these methods are only appropriate for the idealized applications developed by these groups.

In short, formal methods result a bit unattractive and undesirable for most of the control engineers for various reasons, being the most important that it is supposed that formal methods framework is a very complex field not within reach of everybody. But the truth is that compelling reasons exist in order to use these methods in the V&V field, and there are a lot of previous works which demonstrate it.

2.2 Reasons for use of formal methods

Despite the little success of formal methods, they provide many benefits in the automation field. There are some studies made with the purpose of explaining the reasons for using formal methods as well. For example, in [6] two main reasons are emphasized. The first one is the need for formal Verification and Validation, Simulation and Analysis of systems –due to the increasing importance of safety and quality because of the competitiveness between companies. The second one is the constant and unavoidable progress that is taking place in the production and its automation due to the fact that business requirements changes, that technological infrastructure is modernized, that governments change laws, etc. This reason involves an improvement of existing sys-

tems that has to be solved by means of the re-implementation, being transferred to new controller hardware (HW). In this survey, the author also makes clear what the aims of formalization are: Reverse-Engineering and Verification and Validation.

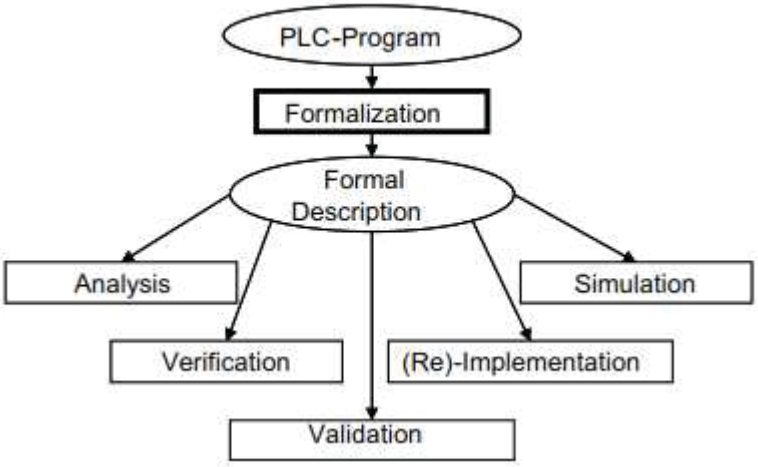


Figure 2. *Reasons for Formalization of PLC-Programs [6]*

Reverse Engineering is a process of evaluating something to understand how it works in order to duplicate or enhance it. The V&V of automation systems, according to the standard [7], is a process of determining whether the requirements for the system or the components are complete and correct, the products of each development phase fulfill the requirements or conditions imposed by the previous phase, and the final system or component complies with the specified requirements. A schedule of the reasons of the formalization of a Programmable Logic Controller (PLC) is represented in Figure 2.

According to [15], "In factory automation, formal system representations are used for two main purposes: to verify/validate/synthesize software control, and to coordinate manufacturing activities. […]. The utilization of formal methods for the synthesis and verification of process logic control has arisen as an alternative to the testing of direct implementations of control realizations against informal specifications. The formalized descriptions of the control objectives, the synthesized /reinterpreted control algorithm and (sometimes) the formal model of the uncontrolled plant are input to verification and validation procedures. […]. Coordination refers to obtaining a system-level functionality based on functionalities provided by each individual component of the system". In addition, it affirms that there are two main formal verification techniques: model checking and theorem proving. In model checking specifications of the system behavior are checked automatically on a finite model of the system. Theorem proving assumes that both the system and its expected properties are formalized in a mathematical logic. Inference rules are then applied to prove the properties from the axioms of the system description.

In spite of these researches remarking the importance of the benefits of formal methods, they have not reached the anticipated success, and the next section collects a set of studies which try to figure out the reasons why.

Formal methods contribute a lot of advantages in the field of V&V. It is the alternative to testing, much safer, more reliable and more effective than testing is. As a proof of it we can mention [1], [3], [4], [5] and [14], where some test cases were analyzed and the effectiveness of formal methods were verified in the V&V of automatic systems. Among these, researches there are a number whose objective –the application of formal methods in a case study- is very similar to the one pursued in this thesis. In the next section, these researches are commented.

2.3 Other similar case studies

The use of formal methods in the V&V of the automated manufacturing systems have achieved successful results and as evidence of it there are diverse studies made in this field in the recent years, such as [3], [4]and [14]. Although these researches are based on different case studies, their objectives are very similar to the one of this thesis, as mentioned before.

On one hand, [3] is one of the studies this thesis is based on. It is an approach to the formal verification of automated manufacturing systems with programmable control. The case study this approach is based on is the FlexLink Lifter, an industrial lifter designed to operate in pallet-based assembly systems with two-position levels. The objective of this approach is to discuss the application of validation and verification methods to the software developed for programmable control devices such as PLC or SoftPLC, where SoftPLC may be seen as a PC-Based control node. It is based on the results of the project 'Modeling and Formal Verification of Industrial Programming in Discrete Automation' (MOVIDA), funded by a consortium of industrial partners and the Finnish National Technology Agency (Tekes). The approach to the formal verification of control software found during MOVIDA project is introduced in this thesis.

TNCES formalism was chosen for modeling both the plant and the controller behavior. The author used MNG (MOVIDA NCES Generator) to design and interconnect the models; then he utilized the software application called iMATCH (integrated environment for Model Assembly, Translation and CHecking) along with SESA model-checker in order to analyze the models and the developed specification. iMATCH provides a timed state-values diagram showing the change of the states and variables values along with time evaluation.

Among the conclusions, we can find that the approach has several inconvenient aspects, mainly the development of a plant model and the exploration of the cause of behavior. They need of the Behavior Explorer (BE) –a set of methods encapsulated into the soft-

ware tools allowing the analyses of the causes for a particular system behavior. Despite this, the implementation of its case 0study provided the ready-to-use solution.

On the other hand, [4] and [14] are other short contributions which have the same objectives of this thesis. The peculiarity of these researches is that they are based on the application of formal methods in some of the modules of the same production line analyzed in this thesis.

The first study is based in four modules of the production line and concludes that building formal models could mean additional work expense that has to be justified; and for this, the practicability of verification approaches depends on user-friendly front ends and integrated software solutions that on the one hand prevent users from formalisms and dull theory and on the other hand adopt already-existing information and interfaces. Even so, it confirms that the correctness of control software is crucial not only for an accurate process control but also for a safe operation of technical processes.

The second research is based on one module of the production line and concludes that the results can: 1) contribute to solving the grand challenge by developing a way to encompass heterogeneous execution and interaction mechanisms for system components; 2) provide abstractions that isolate the design subproblems requiring human creativity from those that can be automated, which enables correct-by-construction models; 3) and eventually, ensures the robustness of the entire systems.

All these researches are intended to be applied to more complicated industrial systems in order to demonstrate its tangible benefits, and this thesis is yet another contribution to that same objective.

The next chapter introduces some theoretical foundations which are essential to understand the techniques used in this thesis before they are implemented in the case study.

3. THEORETICAL FOUNDATIONS

This chapter presents the theoretical and technical background information necessary to understand the procedure applied in the case study of this thesis. The next sections of the chapter introduce the most important topics related to the given problem research: the type of system the case study is, the formalism used in the formalization, the method to make evolve formal models and the software tool used to perform everything.

3.1 A preliminary matter: Discrete Event Dynamic Systems

Before starting with the implementation of the methods in the production line, it is necessary to analyze what type of system it is. Since the case study of this thesis is a Discrete Event Dynamic System (DEDS), it is required to focus this section on DEDS in order to understand how they evolve as well as their main characteristics. Anyway it is only a short introduction, -some good overviews of this type of systems can be found in [8] and [9].

As the name suggests, DEDS are systems which evolve, typically in an asynchronous way, according to the occurrence of discrete events in a specified amount of instants in time. Namely, they are:

- Dynamic: DEDS are characterized by the change of the variables as a function of time.
- Discrete: The state variables only change in a discrete set of points in time.
- Systems evolving by events: An event is an instantaneous happening that can change the state of the system. The occurrence of events makes the system change of state.

In this type of systems the initial and the final state of the studied variable are the only interesting and usable aspects: the passage from one state to another and the information between them are dismissed because they are irrelevant. The elapsed time between states is the only important issue about the change.

An illustrative example is given in Figure 3 in order to better understand the conception. It is a representation of a *x* variable course according to a succession of given events. As shown, time is not specified on the horizontal axis. Instead, there is a sequence of occurring events $\{e0 , e2 , e3 , e5 , e6 , e7\}$ causing changes in the variable values on the vertical axis $\{x2 , x3 , x6 , x5, x8 , x3\}$. Let us imagine for a moment that the *x* variable represents the different positions (states) an elevator may remain in, and that the

events are the calls or the pressings of the people using that elevator. In the initial state the elevator is on the second floor (*x2*) and then an event happens (*e2*) –the person using the elevator has pressed the button to the third floor. So, the *x* variable changes from *x2* to *x3* in a discrete evolution of time; the passage between both states is insignificant, only the elapsed time should be considered, but in this case it is not reflected. In addition, the *x* variable experiences the evolution reflected in the next graph due to the given occurrence of the events, but it should be noted that a different order of the events sequence can generate a different sequence of the values of the *x* variable.

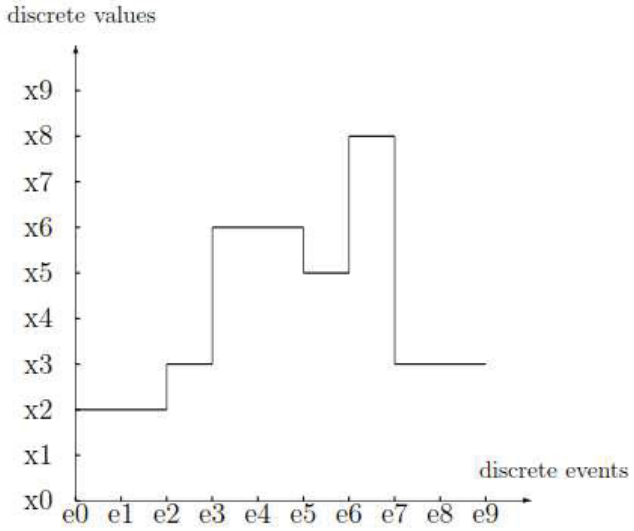


Figure 3. *The course of a variable of DEDS [9]*

There are a lot of existing dynamic systems with a DEDS structure, being among them manufacturing systems. In addition, more applications could enter into the DEDS framework in case state space approach in representing and analyzing was utilized.

In DEDS, some of their events are controllable, meaning that those events can be enabled or disabled according to the a third person's decision. The aim of controlling DEDS is to conduct the system to a desirable state. From this point of view, it is easy to understand what the aim of the controller is –being the main driver of the behavior of the system. Furthermore, it could also happen that only a set of events were observable and not all of them, so the controller have to make decisions according solely to the observable events.

The case study of this thesis is a DEDS in which all the events are controllable and observable, so all the possible states and the way to achieve them are known. This facilitates both the controllers and the plants design. So, after the study of the system all its states can be identified. It just needs to find a formalism able to represent each state and that can be evolved by the occurrence of events.

The next section presents the selected formalism and all its functionalities.

3.2 Modeling formalism

In Control Theory continuous-time systems and discrete-time systems have successful modeling and control methods, but in the case of DEDS these methods usually do not prove useful for the same modeling and control purposes. They need and should be represented by another type of formalism. They require a formal method able to represent each state of the plant and to visualize the different ways the plant can take in order to achieve each of them –that is to say, an illustrative formalism that indicates in each moment the state the plant remains in.

There are different formalisms within formal methods used on modeling and control of DEDS, such as finite automata, Petri Nets (PN), General Transition Systems, Synchronous Languages or Higher Order Logic. But most of the models represented in this type of formalisms lack integrating capabilities: “while they may cope well with the modeling of a particular process, building the overall model of a system comprising several processes is difficult” [1].

The formalism trying to avoid this and habitually used in DEDS is Net Condition/Event Systems (NCES). This formalism has the representation of Petri Net but it is scripted by conditions and events as if it was the formalism of Function Block Diagrams.

Even so the elapsed time between states has to be measured, and for that purpose Timed Net Condition/Event Systems (TNCES) are needed. TNCES are the result of NCES representations adding the measured time between states.

The first section of this chapter contains a short approach to PN and NCES in order to understand the basics of the TNCES formalism. After that, the functionalities and capabilities of TNCES are exposed in the second section.

3.2.1 Petri Nets and Net Conditions/Event Systems

Before focusing on TNCES, it is necessary to know about its origins and the basic principles of the formalism in order to better understand its application in formalization of automated manufacturing systems. Hence a review of PN is needed, but it is only a short introduction explaining some important characteristics about the formalism. In order to increase the knowledge of this type of formalism, [10] and [11] are good examples to make an approach of the PN to the automation.

Petri Nets are a type of formalism represented by two main elements –places and transitions. Places are symbolized by circles and transitions by bars. Both elements are connected by arcs which in turn connect a place to a transition or a transition to a place.

Tokens are symbolized by small black circles. Each place can contain or not a finite number of tokens defining the state of the system described by the PN according to the places they are in.

The evolution of PN is effected by the firing of transitions. A transition can be fired if it is enabled, meaning that each of their input places contains at least a token. When a transition is fired a token is removed from each one of its input places and is moved to each of its output places. Figure 4 shows an example of a PN and how the firing of a transition is made.

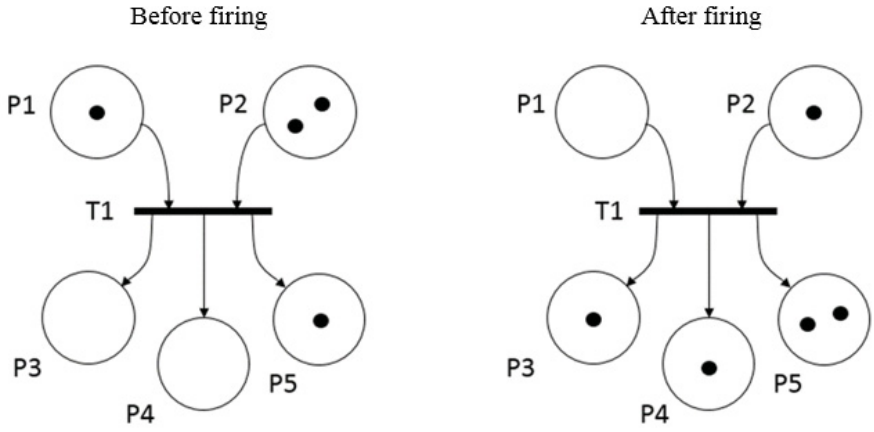


Figure 4. *Representation of firing a transition*

The marking of a PN (*m*) –or the state of a PN- at a given moment is a vector whose length represents the number of places and whose components represent the number of tokens remaining in each respective place. For example (see Figure 4), the marking of the PN before firing is: *m1*= (1, 2, 0, 0, 0); after firing it is *m2*= (0, 1, 1, 1, 2). The set of markings of a PN is used to represent the state space of a model. So the state space of a model may be considered as a representation of the set of all possible states of a PN. In addition, these diagrams indicate the paths each state of a PN can follow in order to change into another one.

Regarding with the application of PN in a functional description of behaviors in DEDS:

- A place represents a state achievable by a system. If a place contains a token it means that at that particular moment the system remains in the state the place is representing.
- Transitions are associated to events. Events can be logic functions of the input variables, the end of a counter or a timer. If a transition is enabled and the event associated to this transition happens, then the transition is fired.

Net Conditions/Event Systems go beyond. If 1) a transition is enabled, 2) the signal event associated to this transition happens and 3) certain signal conditions are fulfilled, then the transition is fired.

In 1995 Hanisch and Rausch introduced this formalism as input/output Petri Nets scripted using condition signals and event signals 56[12]. This methodology improves the expression capabilities of PN, contributing to the original PN both typed modularity and the new elements of PN notions –event arcs and condition arcs-, hence its importance. Condition arcs always go from a place to a transition whereas event arcs always go from one transition to another. A marked place could mean a condition for a transition, and the firing of a transition could mean an event for other transition. In this formalism a transition could have coupled a condition(s) or/and an event(s). It means that an enabled transition will be fired if the coupled condition(s) are fulfilled and the coupled event happens.

Furthermore, NCES are usually represented through interconnected modules. Their connections are performed by means of event arcs and condition arcs. The following example in Figure 5 tries to explain it:

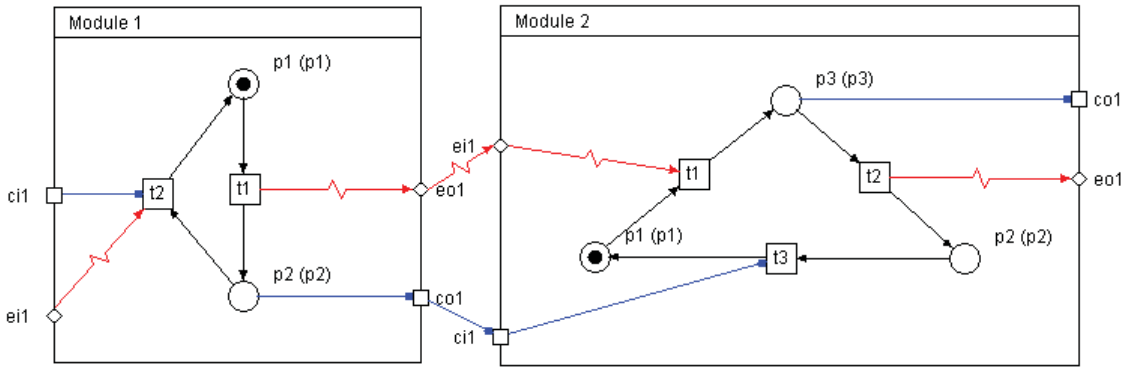


Figure 5. *TNCES modules representation*

Condition arcs are represented by blue lines whereas event arcs are represented by red lines. In the state of the current system, when transition t1 of Module 1 is fired, automatically transition t1 of Module 2 is also fired, because the event happens and the transition is enabled (its input places contains at least a token). In a different state of the current system, when place p2 of Module 2 is marked, transition t3 of Module 2 can only be fired if place 2 is also marked, because it is the condition for firing the transition. Besides, modules can also be influenced by external NCES modules –it is the case of transition t2 of Module 1: for the firing of t2 it would be necessary that the fulfillment of both ci1 condition and ei1 event coming from external modules happened at the same time. Moreover, a place or a transition can influence an external NCES module –it is the case of place p3 and transition t2 of Module 2.

Once the basics of this type of formalism is explained, in the next section TNCES are described in a more technical way and all the mathematical background of the selected formalism is detailed.

3.2.2 Timed Net Condition/Event Systems

In this section, Timed Net Condition/Event Systems (TNCES) are presented, a formalism derived from the NCES and the one used in the case study of this thesis.

TNCES are NCES extended with the consideration of the time [13]. TNCES can be defined as the following tuple:

$$TNCES = \langle P, T, F, m_0, \psi, CN, EN, DC \rangle \quad (1)$$

Where:

- $P = \{p_1, p_2, \dots, p_n\}$ is a finite set of places
- $T = \{t_1, t_2, \dots, t_m\}$ is a finite set of transitions
- $F \subseteq (P \times T) \cup (T \times P)$, is a finite set of flow arcs between places and transitions
- m_0 is the initial marking
- $CN \subseteq (P \times T)$ is a finite set of condition arcs
- $EN \subseteq (T \times T)$ is a finite set of event arcs
- ψ is input/output structure of TNCES module

$$\psi = \langle C^{in}, E^{in}, C^{out}, E^{out}, Bc, Be, Cs, Dt \rangle \quad (2)$$

  Where:
  - C^{in} is a finite set of TNCES module condition input signals
  - E^{in} is a finite set of TNCES module event input signals
  - C^{out} is a finite set of TNCES module condition output signals
  - E^{out} is a finite set of TNCES module event output signals
  - $Bc \subseteq C^{in} \times T$ is a set of TNCES module input condition arcs
  - $Be \subseteq E^{in} \times T$ is a set of TNCES module input event arcs
  - $Cs \subseteq C^{out} \times T$ is a set of TNCES module output condition arcs
  - $Dt \subseteq E^{out} \times T$ is a set of TNCES module output event arcs

Time plays an important role in TNCES, there are time constraints related to pre-transition flow arcs $F^- \subseteq P \times T$ [13]:

$$DC = \langle DR, DL, D_0 \rangle \quad (3)$$

Where:

- DR is the minimum time that the token should spend at a particular place before the transition(s) can be fired.
- DL is the maximum time the place may hold a token for (if all the other conditions for transition firing are met).
- D_0 is the initial set of clocks associated with the places.

DR has to be introduced in the input arc of the transitions, so the enabled transition has to wait this minimum time to be fired. That produces a better approximation to the real operation of the represented module.

Furthermore, TNCES can be used to represent hierarchical model by means of set of interconnected modules. The TNCES module is defined as [17]:

$$TNCES_{composite} = \langle M, Cm^{inp}, Cm^{out}, Em^{inp}, Em^{out}, Ic, Ie \rangle \quad (4)$$

Where:

- M is the set of modules that may be composed of $TNCES$ and $TNCES_{composite}$ modules
- Cm^{inp} is a set of composite TNCES module condition inputs
- Cm^{out} is a set of composite TNCES module condition outputs
- Em^{inp} is a set of composite TNCES module event inputs
- Em^{out} is a set of composite TNCES module event outputs
- $Ic \subseteq C_k^{out} \times C_l^{in} \cup Cm^{inp} \times C_i^{in} \cup C_i^{out} \times Cm^{out}$ is the set of condition arcs of composite TNCES
- $Ie \subseteq E_k^{out} \times E_l^{in} \cup Em^{inp} \times E_i^{in} \cup E_i^{out} \times Em^{out}$ is the set of event arcs of composite TNCES

The selected formalism is the most suitable formal method because of its illustrative and technical capabilities. These capabilities are used by software tools able to extract and manage all this information.

The next section explains what method is executed so that the TNCES representation can evolve as the desired operation, trying to resemble the real functioning of the system.

3.3 Closed-Loop System modeling

The industrial automation systems can be interpreted as the union of two main parts -the controller and the plant (controlled object). The controller is a HW device driven by software code that performs data processing, communication and decision making, whereas the plant contains the material-handling part of the equipment [5]. The communication between both of them is executed by means of the controller inputs/outputs and the sensors/actuators of the plant.

Regarding industrial automation systems, one part cannot work without the other since one is used to generate the inputs for the other and vice versa. As shown in Figure 6, the controller receives a number of inputs generated by the plant sensors and triggers the respective outputs resulting in the activation of the actuators. From the other point of view, the plant activates its actuators according to the controller outputs and then sends the signals coming from its sensors to the controller –which receives them as inputs. They do not share their complete state information or any common variables but they exchange digital or analog data.

.

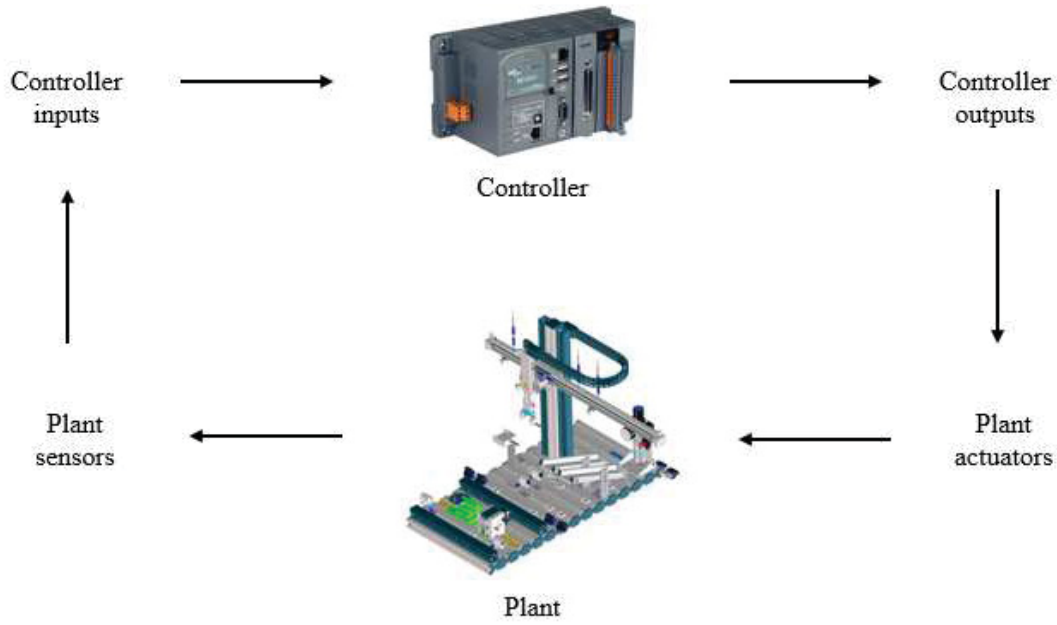


Figure 6. *Closed-loop system model*

Validation & Visualization of industrial automation systems focuses on the proper functioning of the controller. But controllers V&V would not be complete in case that the controller was considered on its own. It needs the plant model to get a complete conception about correctness of the control software –the closed-loop system of the controller and the plant have to be taken into account as well.

Formal methods are used to abstract the plant and the controller in order to validate and verify its functioning, and once more the necessity of formal methods is proven. The abstraction of both the controller and the plant results in a formal system model composed of the controller model and the plant ready for verification by means of model-checking. It is necessary that both the controller and the plant model are represented using the same formal language, so they can be interconnected after their modeling.

Figure 7 shows how the representation of the closed-loop system model is in a signal conception for the formal closed-loop system model. There is a remarkable comparison between this model and the real model depicted in Figure 6.

As shown in this Figure, the system model is composed of the formal plant model and the formal controller model. Both models have inputs and outputs (I/O): the controller outputs interconnect with the plant inputs, and the plant outputs do the same with the controller inputs, forming a closed-loop model. Interconnections between the two parts (controller and plant) are made by means of condition arcs between the outputs of one model and the inputs of the other. Inner interconnections between formal models and their respective I/Os are made by means of condition arcs and event arcs. A change in the state of the controller could influence the state of the plant just the way a change in the state of the plant could influence the state of the controller.

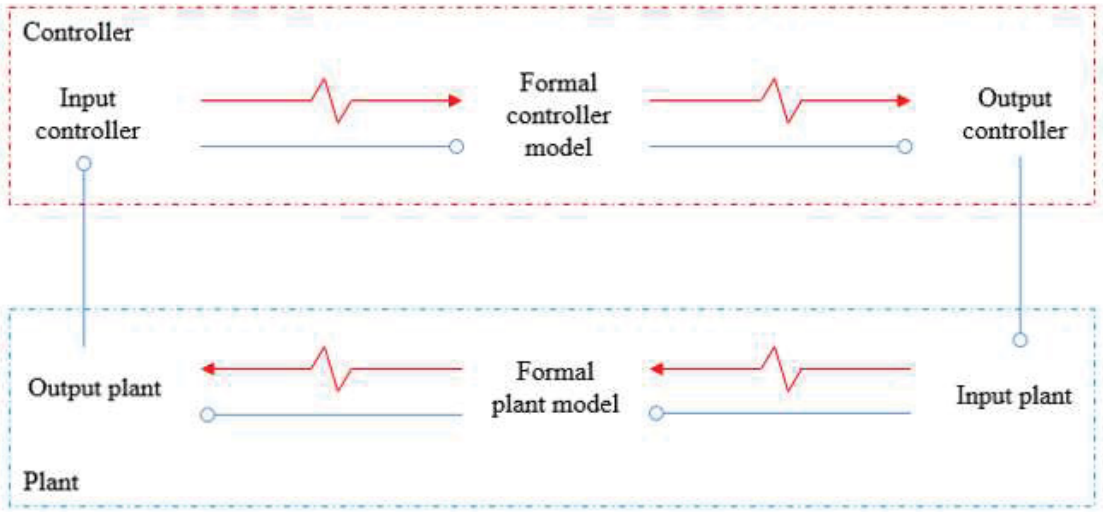


Figure 7. *Signal conception for the formal closed-loop system models*

The closed-loop system operates as follows: a change in the controller updates its outputs, altering in turn the plant inputs; because of this a change in the plant is produced and it is reflected in its outputs, that at the same time alter the controller inputs. The controller is now updated once more due to the alteration in its inputs, and the loop starts again.

The next sections concentrate on the different parts of the closed-loop system: controller model, plant model and their interconnections.

3.3.1 Controller modeling

This section focuses on the controller part of the framework. As previously commented, the use of formal methods in V&V is an alternative to the testing in order to verify, val-

idate and synthesize the software control. To achieve this model-checking it is necessary for the controller model to be represented with the same formal language used by the plant model. But control software is usually represented in other programming languages such as Ladder Diagram (LD), Sequential Function Chart (SFC) or Statement List (STL). Hence, ways need to be found to convert this programming language into the required formal method.

It is not complicated to generate the formal model of the controller -it can be performed automatically if the controller devices are properly specified and the information about the performance of each programming language instruction is correctly defined. As it will be commented later, the MNG has the possibility to automatically convert control software to NCES. The controller software in LD (files in .cxt) or in STL (files in .AWL) can be introduced in MNG and be automatically converted to NCES. In other words, if the control software in these programming languages is provided, it is very easy to obtain the NCES representation in a direct and fast way, saving a lot of time and work.

However, in order to get an in-depth analysis of the controller modeling, in this case the controllers design has been performed manually in TNCES and introduced in MNG. The reason for this procedure is that the aim of this thesis is to obtain a good abstraction of the plant using formal methods so that the formal model can be used in the future by other users in the V&V of their designed control software. The control software V&V is not being checked in this thesis, it is only used to build the closed-loop system model and to ensure that the plant evolves as the real operation of the system does. A control software already performed may contain some failures in its design and because of that the formalization of the plant may not have a proper functioning. The formal controller model is built only for this cause –to ensure that the formal plant model has the desired functioning. So it has to be performed manually in TNCES. Once validated the abstraction of the plant, any control software in LD or STL can be introduced in MNG, converted to TNCES automatically and ready to be subjected to V&V in the model checking.

Once clarified this remark this section focuses on the manual modeling of the controller in TNCES. For that, it is treated from a sequential point of view. In other words, the controller has to be designed as a sequence of actions activated or deactivated in a particular order, so the plant model can evolve in the most similar way to the plant operation evolution. It is the only necessary purpose followed by the controller modeling of this thesis –a design based both on sequencing the real plant operation and on ordering it in a way that it acts on the plant indicating which plant inputs are enabled and which are disenabled at each instant of time. What can be achieved with this? The fact that the plant changes its states in the same order that its real operation would do. What is achieved is the evolution of the plant in its actual behavior.

Hence, the result of the controller modeling is a TNCES representation whose places and transitions are forming a sequence. A transition is fired when it is enabled and the conditions connected to it are set. These transitions result in events that produce a change in the plant inputs. Subsequently, these changes produce in turn a number of variations in the controller inputs that change the input conditions which fire the next transition in the sequence of the formal controller model. It is a sequence of cause and effect or action and reaction.

In other words, it can be seen from another point of view as a succession of steps that may be the next:

1. The controller is in the initial state waiting for certain conditions.
2. When these conditions are fulfilled, the transition is fired producing a change in the plant.
3. The controller waits again until the plant produces the appropriate changes so that the conditions of one of the next controller transitions are fulfilled.
4. It follows the same procedure for the next firing transitions, receiving the changes of the plant and sending the opportune actions according to these changes.
5. In certain occasions, a place in the TNCES representation of the controller has more than one output transitions. The controller has to choose between different paths depending on the fulfilled conditions. It means that different sequences in the evolution of the plant may exist –such as accepting or rejecting a work piece depending on its height.
6. When the plant finishes its work cycle and starts again, the TNCES representation of the controller closes the loop. The last transition is joined to the respective place permitting to start again the work cycle of the plant. Then, the loop repeats constantly.

In short, the formal controller model is represented as a sequence resulting in the change of the plant from one state to another until the plant finishes its work cycle. The next section explains the most important part of this thesis, the plant modeling –how the plant is abstracted using formal methods.

3.3.2 Plant Modeling

In this section, the plant modeling in formal methods is discussed. The result of the plant modeling is the formal design of the plant that will be used by other users in the V&V of their designed control software. This is the reason why the application of this part in the case study is the most important section of this thesis.

The design of the controlled object (plant) does not have a common standard due to the different application domains, and that is why the plant modeling is performed manually. However, there exist some generic models for the most common elements in indus-

trial automatic systems, such as conveyors, buffers or industrial robot arms –elements whose application in industrial automatic systems is similar in any of them. The TNCES representation of one of these elements will be commented later as an example.

It is in the process of the plant modeling that the plant is really abstracted –when the real functioning of the machine, device or process is comprehended. The plant design has to be able to evolve the same way as the controlled object would really do. The evolution of the different states of the design in the formalism should be executed the same way as the controlled object would. To achieve that, a whole study of the controlled object has to be performed, identifying both the different states the plant can remain in and the different routes the system had to follow to achieve each of them. In other words, the viewing of each possible state of the plant has to be contemplated and the different ways to achieve these states must be analyzed.

It is not an easy task, since –as commented before- each controlled object is unique, has different work methodology and is manufactured for a unique purpose with different applications. Even so, some elements in the industrial automated systems can be treated as generic applications. For example, the conveyors may have distinct composition in two different automatic systems, but their application is similar for any of them –to take an object from one place to another. The formal model of a conveyor with two places of capacity would be as represented in Figure 8.

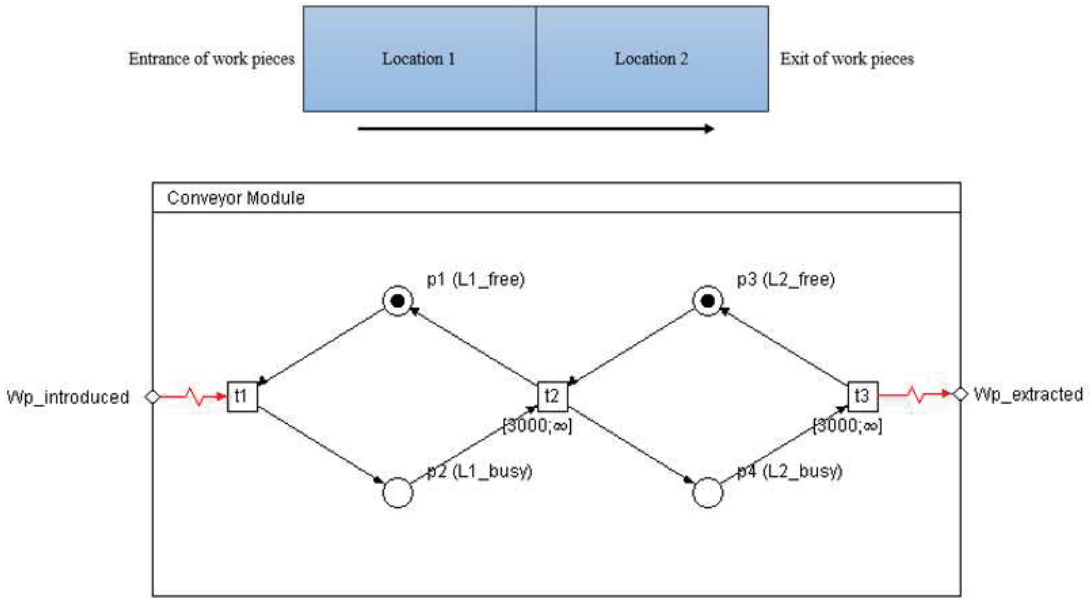


Figure 8. *Plant TNCES representation of a conveyor with two capacities*

As shown, p1 and p3 represent respectively that Location 1 and Location 2 in the conveyor are free, and p2 and p4 represent that the same locations are busy. It should be noted that t1 will be fired if an extern event happens, and t2 and t3 will happen three seconds after they will be enabled. The design operation goes as follows: When t1 (the

only one enabled) is fired, it means that a work piece has been introduced in Location 1, so Location 1 stops being free and becomes busy. In other words, the token of p1 transfers to p2. Then, three seconds later t2 will be fired and it means that Location 1 will be free again and Location 2 will become busy (p1 and p4 will contain a token). Two things can happen in this situation: either a new work piece can be introduced in Location 1 or the work piece of Location 2 is extracted. If the first situation happens, t1 will be fired and the token in p1 will transfer to p2 –where the token will remain until t3 is fired- and t2 will be enabled. If the second one happens, then Location 2 stops being busy and becomes free, and so the design is once again as it was in the beginning.

So the formal conveyor model with two capacities may remain in four different states (markings): *m1*= (1, 0, 1, 0), *m2*= (0, 1, 1, 0), *m3*= (1, 0, 0, 1) and *m4*= (0, 1, 0, 1). They would compose the state spaces in Figure 9:

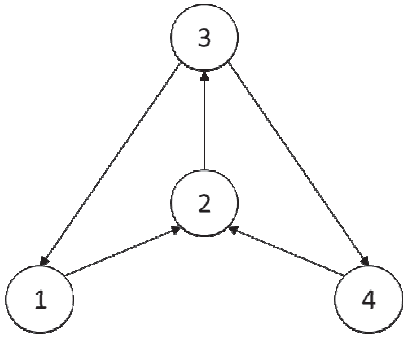


Figure 9. *State space of a conveyor of two capacities*

The formal model extracts the conveyor operation and its evolution is similar to the real functioning. This is the main idea of the use of formal methods: representing the operation of the controlled object in a mathematical logic that provides us with all the necessary information in the model-checking.

Actually, a conveyor may be a module of a global system –it is only an element of the plant. For the plant modeling, the different modules of the plant have to be identified, and then the operation of each module has to be studied separately.

Since the case study is a DEDS, it is easy to determine the different states the system can remain in. When these states are known, the interconnections between them have to be established. Designers have to bear in mind that in TNCES formalism each state is represented by a place, and the event producing the changes of state in the plant is represented by a transition. What is required to happen in order to achieve each state? What states may give rise to a succession of other states? Which states can each state be achieved from? These are questions the designer should answer in order to obtain a good representation of the formal model.

When the modeling plant is finished the result is a set of TNCES modules representing the different elements of the plant. For example, if a station is composed of a robot and

an elevator, then the plant model is composed of two modules –the robot module and the elevator module, each of them with its different representation in formal methods.

In addition, it is the same case as in the controller modeling –each module has its input conditions, connected by means of condition arcs, and has its output events, connected by means of event arcs. The evolution of the plant modules is similar to the evolution of the controller. The plant remains in an initial state and it will evolve to one state or another according to the fulfilled conditions. This evolution is executed by means of the firing of the transitions which create events that update the controller inputs and start the closed-loop again.

Finally, the time operation has to be considered in the plant modeling, meaning that the measuring of the time of each operation of the actuators has to be taken. The time of the conversion from one state of the plant to another has to be measured and introduced in TNCES representation. The above-mentioned time is the time for a transition to be fired after being enabled. As it was explained in the section devoted to TNCES time is established in the input arc of the transition. This provides a better approximation to the functioning of the controlled object.

The next section focuses on the modules interconnecting both models (controller and plant) –the inputs and outputs modules.

3.3.3 Data Representation– Input, output and update modules

In the previous sections the plant model and the controller model have been commented. This section concentrates on the interconnection between them. This interconnection is performed by means of the I/O modules.

Both models have respectively their I/O modules, but it is necessary to know that the controller outputs are inputs for the plant and that the plant outputs are inputs for the controller. For this reason the outputs modules of the controller have to be connected with the input modules of the plant, and vice versa, the outputs modules of the plant have to be connected with the input modules of the controller. In this way, the closed-loop system is performed.

Actually, data represented by I/O modules are of a Boolean type. They only accept two possible answers. In a representative point of view, on the one hand the controller inputs indicate whether one sensor is detecting or not, whereas the outputs indicate the activation or deactivation of the plant actuators. On the other hand, the plant inputs indicate whether one actuator is operating or not, whereas the outputs indicate the activation or deactivation of one sensor.

Regarding the TNCES representation of these modules, the input and output modules have the same TNCES representation in both models (controller and plant) and the output modules as well. Figure 10 shows the representation of these modules in TNCES:

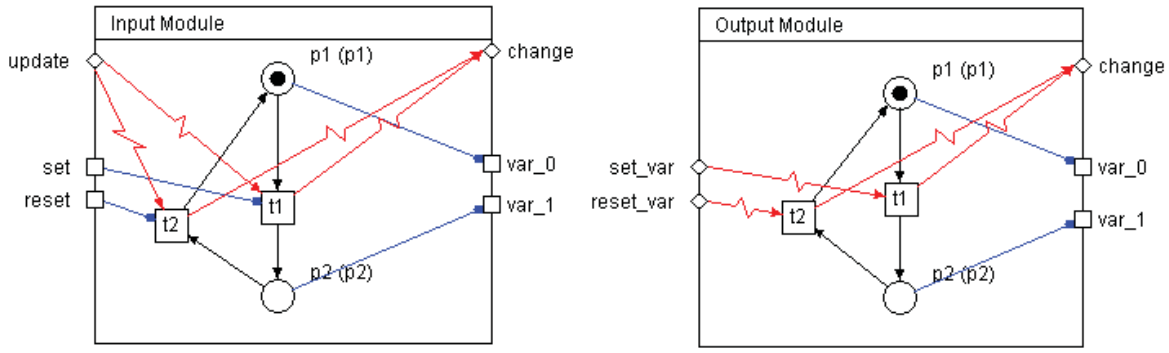


Figure 10. *Input and output modules in TNCES representation*

As may be seen, the input module and the output module do not differ so much. Both of them have two places and two transitions. Places are used as conditions in other external module. Transitions are also used as events in other external module but they can only be fired if another extern event happens. The only difference between them is that transitions in the input modules are conditioned by condition arcs (set/reset) from an extern module.

For the controller, inputs are considered as a change in the plant and outputs as an update of the variables of the plant. For the plant, inputs are considered as an update of its variables and outputs as a change itself.

Apart from the input and output modules, it has been designed a module called "Update" –sited between the controller and the plant in order to create a delay of 100 milliseconds. This is due to the fact that MNG needs a certain amount of time to update its variables in a correct way. In addition, this module is executed twice in the sequence of the closed-loop: once from the controller to the plant and once from the plant to the controller. This inconvenience of the tool can make the model slightly lose contact with reality, but the change is minimum. The update module is represented in Figure 11.

It is simply a module with two places and two transitions: t1 is fired when an external event happens and then, automatically, after 100 milliseconds t2 is fired producing a change in another module.

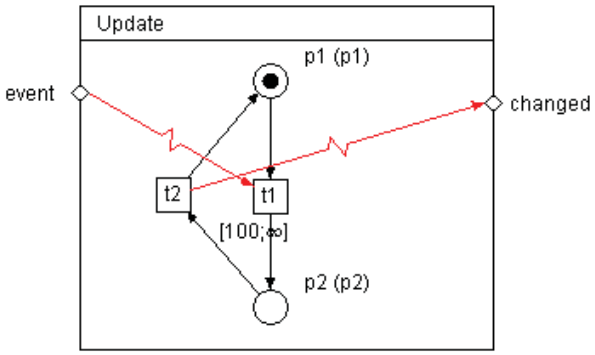


Figure 11. *Update module*

The next section presents the software tool used to design the formal models, to interconnect them in the closed-loop system model, to validate them and to make them evolve in a visualization.

3.4 MOVIDA NCES Generator

MOVIDA NCES Generator (MNG) is the software used to design the TNCES representations, to interconnect them, to generate the results after the model-checking and to make the models evolve.

This tool resulted from the project 'Modeling and Formal Verification of Industrial Programming in Discrete Automation (MOVIDA)' funded by a consortium of industrial partners and the Finnish National Technology Agency (Tekes). MNG is a PLC source code, an UML translator and a NCES editor. Having a model expressed in UML, it is possible to translate it into TNCES by means of MNG. The output of the MOVIDA NCES Generator can be introduced to other verification tools, but it also has a big set of functionalities that can be used on TNCES representations. Figure 12 represents the MOVIDA Tools Framework.

As the figure below demonstrates the plant modeling is done with UML with a tool that supports the Extensible Markup Language (XML) export of the diagrams, but in this case it is manually designed and directly introduced in the MNG. At the same time, the controller source code may be developed and translated into TNCES by MNG. MNG can input XML files holding a Unified Modeling Language (UML) model, and after that MNG can convert XML files into TNCES, but in this case it is also manually designed and directly introduced in MNG. Both models are designed by means of NCES Editor. Once obtained the controller model and the plant model, they can be interconnected in MNG and be subjected to model-checking in NCES Analyzer. The framework is composed of 9 basic tools: MOVIDA NCES Generator, NCES Editor, NCES Analyzer, Model Checker tool, Visualization tool, GUHA tool, Structural Reasoner, Training Utility and Hybrid Analyzer [16].

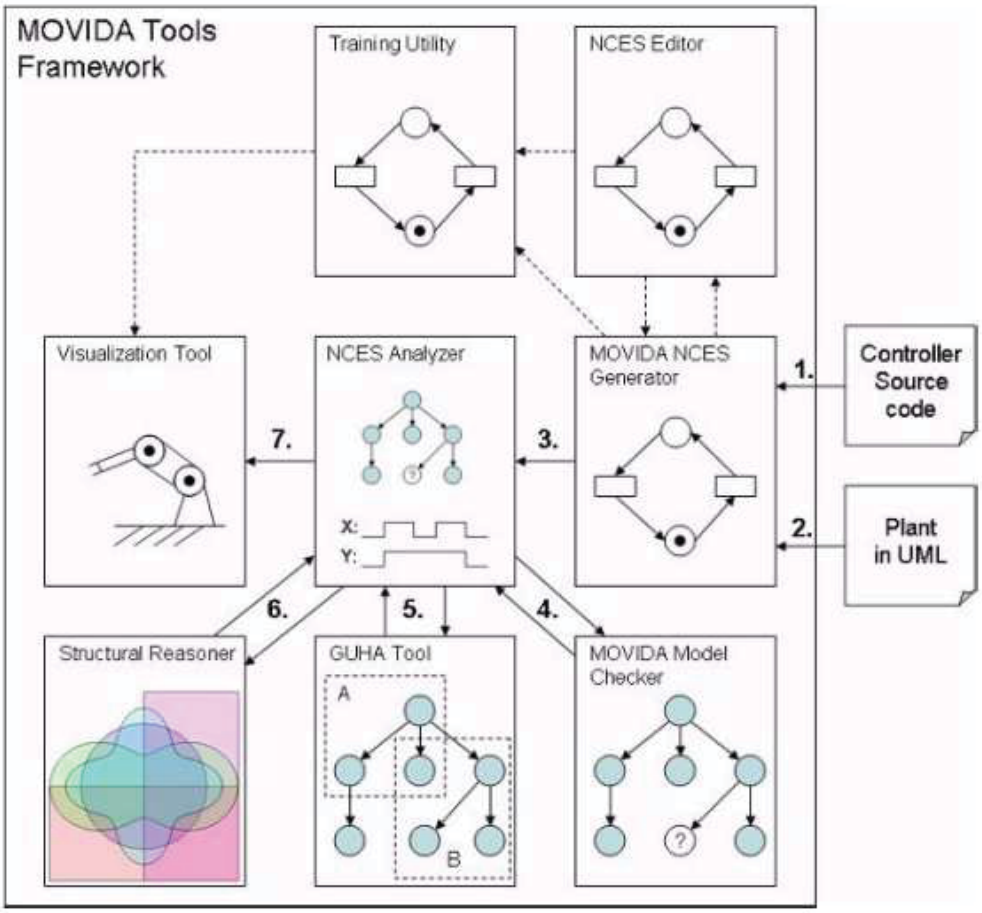


Figure 12. *MOVIDA Tools Framework [20]*

In this thesis, only five of the mentioned tools are used. They are:

- The NCES Editor, used to design both formal models (plant and controller) in TNCES representation. It is an essential tool in order to represent the places and the transitions, to interconnect them and to establish the measured times.
- The MOVIDA NCES Generator, used to join both models (plant and controller), to interconnect their I/O and to create the closed-loop systems.
- The Training Utility, used to make evolve the formal closed-loop systems in order to check any failure or the possibility of dead-locks. It has two options of evolution: step by step or running simultaneously in a preprogrammed period of time.
- The NCES Analyzer, used to do the model-checking of the formal models. This tool also generates the state spaces and the timing diagrams of the models.
- The Visualization Tool, used to design a visual representation of the system and to make it evolve with the simulated evolution in the Training Utility.

4. CASE STUDY

This chapter focuses on the implementation of the above-mentioned techniques in a real case study. The studied system is the FASTorium –an automatic line production in a miniature scale composed of different stations, as shown in Figure 13.

Hence, both models (controller and plant) of each station will be designed and interconnected resulting in the closed-loop system models. Then that interconnection will be subjected to model-checking, and the state space and timing diagrams of the models will be obtained. Lastly, the evolution of a designed visualization can be observed.

It is intended to achieve the main objectives of this thesis with this application –namely the result of the formal models ready-to-use and the proof of the benefits of formal methods. After the implementation the results will be analyzed and some conclusions will be drawn.

The following sections 1) describe the case study (FASTorium) and each of the stations it is composed of, 2) overview the controllers modeling, 3) study in-depth each of the plants modeling, and finally 4) obtain the closed-loop system.

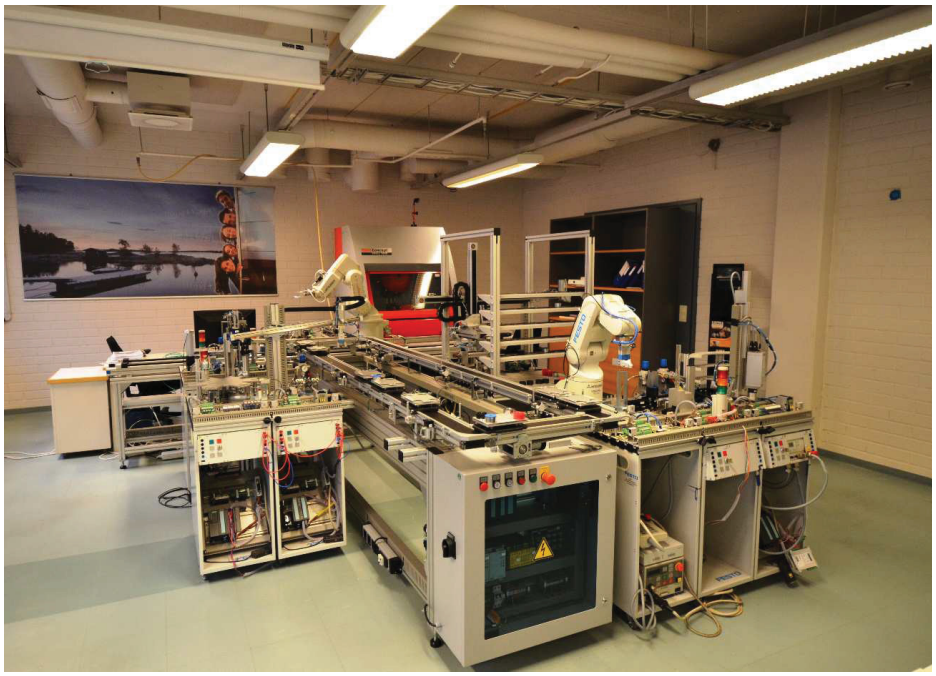


Figure 13. *The FASTorium line*

4.1 The FASTorium

The FASTorium is a set of Modular Production Systems (MPS) designed by FESTO. MPS are small modular stations interconnected to each other simulating the production line of determined work pieces . Its purpose is to be used in a learning environment. Due to its flexibility and modularity each station is removable and the global model can have different configurations depending on the order of the manufacturing processes. For this reason, in this thesis a particular order of the stations has been previously chosen. The layout configuration used in this thesis is represented in Figure 14. Furthermore, some stations are composed of a set of different modules that in turn can also have different configurations.

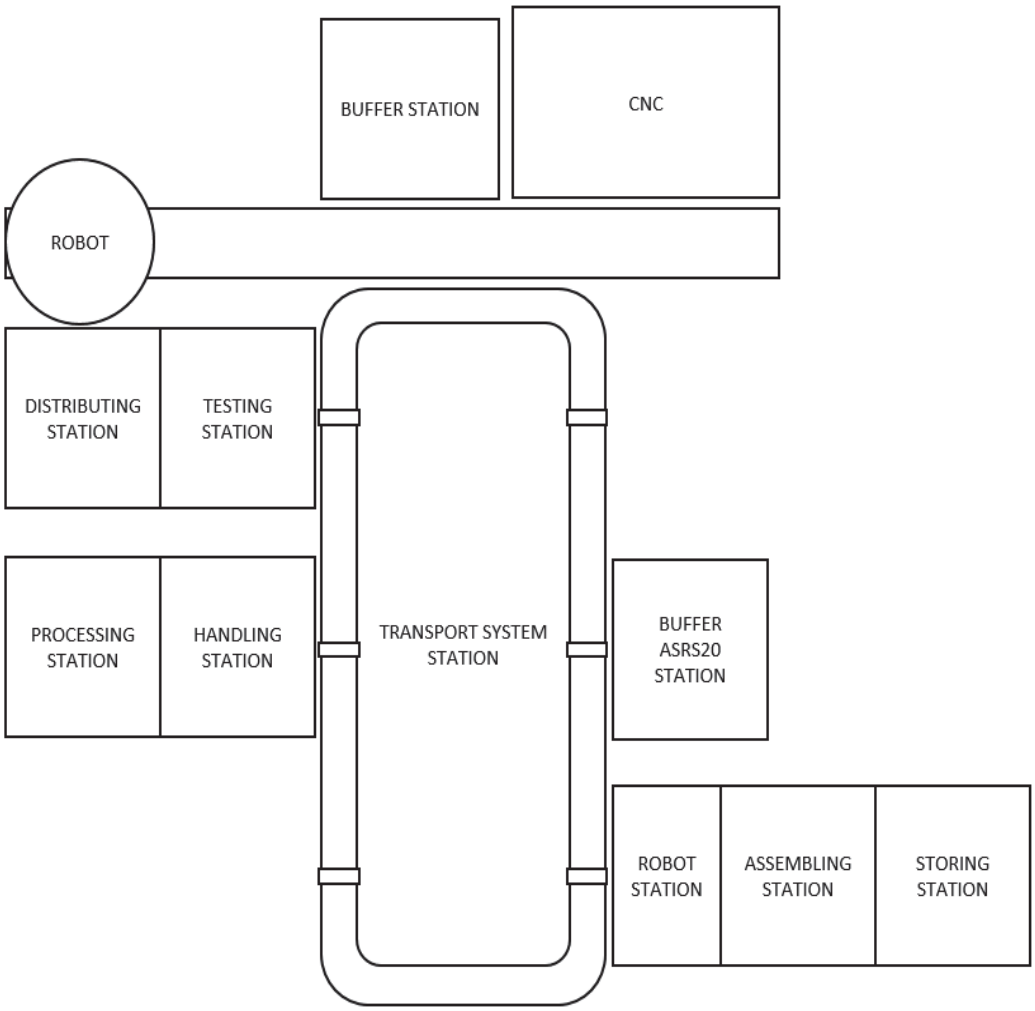


Figure 14. *Layout of FASTorium*

Each station executes diverse processing and control operations on work pieces. These pieces are cylindrical bases that can be differently colored (black, red or silver) and that are made of different materials (aluminum or plastic). The automatic production line simulates a manufacturing line whose stations execute specific activities such as supply, classify, transport, process, store, etc.

Certainly it consists of nine stations (Distributing, Testing, Handling, Processing, Robot and Assembling, Storing, ASRS20, Transport system and Buffer), an industrial robot (MITSUBISHI RV-3SDB-S15) and a CNC milling machine (EMCO CONCEPT MILL 105).

In addition, as previously commented, the model perfectly simulates a DEDS, since changes in the system variables are caused by the occurrence of events in the plant. It is a clear example for applying the formalization by means of TNCES and for being subjected to model-checking.

The next section explains in detail the functioning of each station and each element of the automatic production line.

4.2 Description of the stations

The Distributing station (Figure 15) retains the work pieces in a stack magazine module able to deliver them one by one by means of the pushing of a double-acting cylinder whose advancing and retracting speed is infinitely adjustable using one-way flow control valves. Whenever the work piece is delivered, a rotary drive transfers it to the next station using a suction cup. The swiveling range is adjustable between 0° and 180° by means of electrical limit switches (micro switches).

The Testing station (Figure 15) is in charge of both determining the material characteristics (recognition) and checking the height (measuring) of the work pieces. The station is provided with a lifting module with two-level positions. The recognition is executed in the bottom level, and the measuring in the top level. The lift is powered by means of a rodless lifting cylinder. The station works in this way: if the work piece meets the height requirement, then it will be pushed to the next station by an ejecting cylinder through an air cushioned slide module. Otherwise it will be rejected in a slide module in the bottom level. The moving compressed air tubing and the electrical cables are routed via the cable guide.

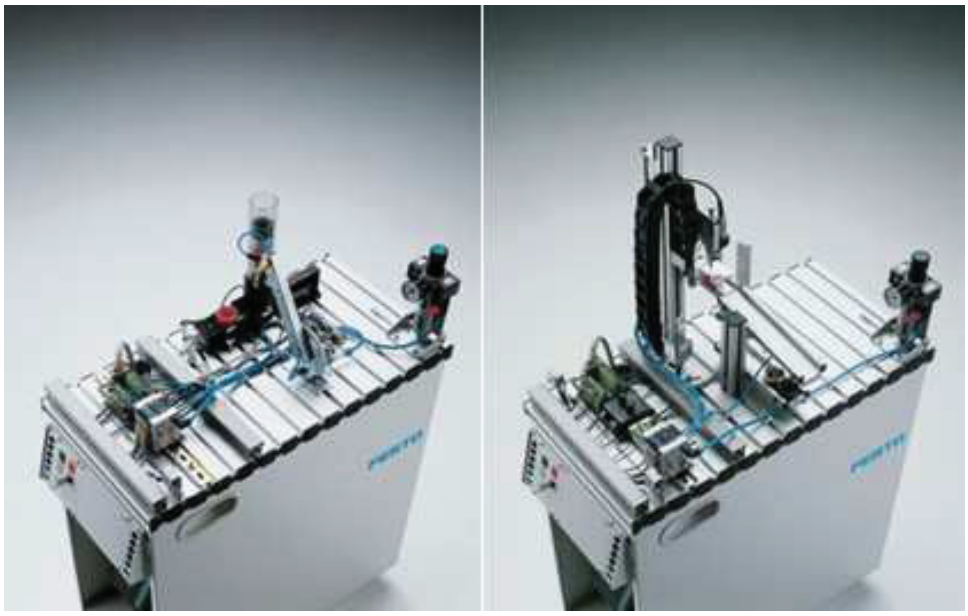


Figure 15. *Distributing station (left) and Testing station (right) [19]*

The Processing station (Figure 16) simulates a processing system. It is composed of a rotary indexing table module with six ratining positions and two processing (testing and drilling) modules. The work pieces enter in the rotary indexing table module; it turns around and displaces them to the different process phases. The rotary indexing table is operated by a direct current (DC) gear motor, and the six rotating plate positions are defined by the positioning screws on the rotary table and are sensed by means of an inductive sensor. Each of the semi-circular cylinder base retainers of the plate is provided with a hole in the center in order to facilitate sensing by means of a capacitive proximity sensor. In the fifth stop, there is a testing module that checks whether there is a hole in the work piece or not. In the sixth stop, a drilling module is used to simulate the drilling of the cylinder base. The motor of this module is operated via 24-volts DC and the speed is not adjustable. Finally, when the work piece arrives to the entrance position again it is removed.

The Handling station (Figure 16) is equipped with a PicAlfa module permiting horizontal and vertical displacement. The module can scroll horizontally to three different positions -“previous station” position, “next station” position and “rejection” position. The vertical displacement can be executed in each of these positions by means of a lifting cylinder fitted with a pneumatic gripper. This gripper takes the work pieces and classifies them into ‘black’ and ‘non-black’ using an optical sensor. The function of the station is to take the work pieces from the Transport system station to the Processing station, and then from the Processing station to the Transport system station or to “rejection position” according to the result of the Processing station test..

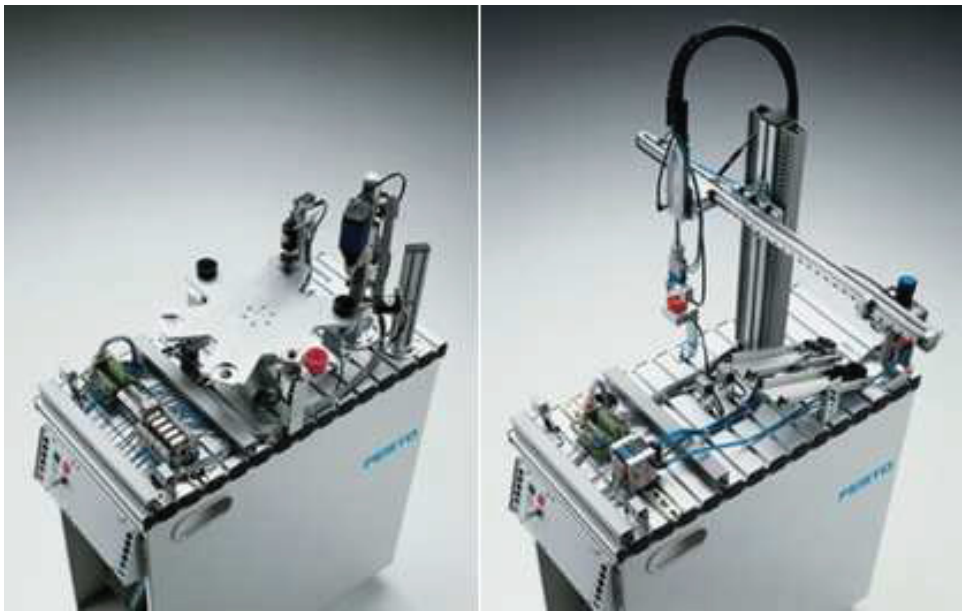


Figure 16. *Processing station (left) and Handling station (right)[19]*

The Robot station and the Assembling station (Figure 17) work with each other. The Assembling station is in charge of providing the pistons, the springs and the covers to the Robot station by means of stack magazines. The Robot station is equipped with a vertical articulated arm robot (RV-2AJ) that assembles/transports the cylinders and all the other elements (pistons, springs and covers). The black work pieces require of silver pistons and the non-black work pieces requiere of black pistons –the pistons are on a pallet. The robot is an industrial robot with 5 axes and is provided with a teach pendant and a controller. It is able to reach all the positions and is fitted with a gripper with three different grip tipologies, so permiting the robot to grab the distinct elements. Moreover, an optical diffuse sensor is integrated in the gripper for color identification.

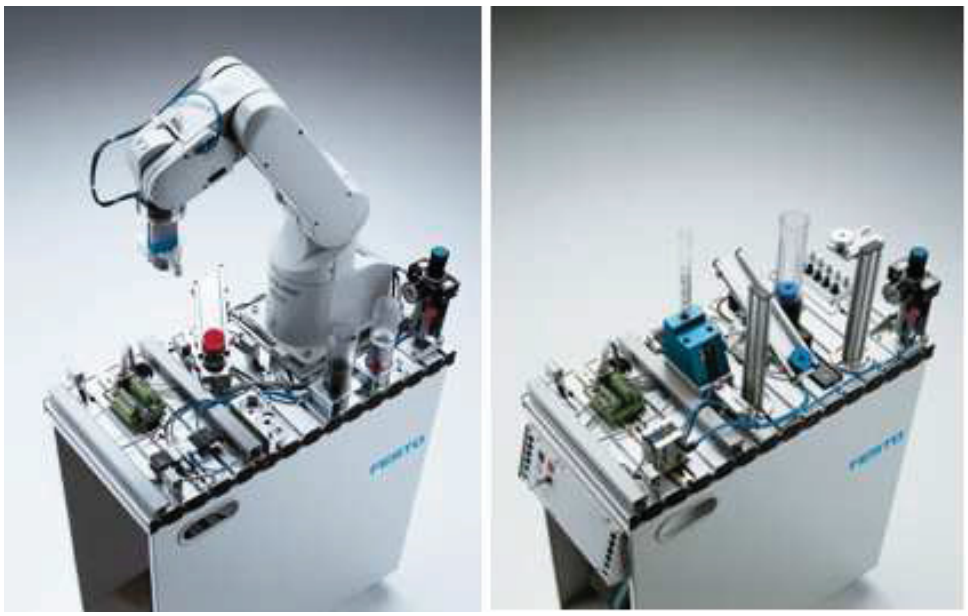


Figure 17. *Robot Station (left) and Assembling station (rigth) [19]*

.

The sequence of the system goes as follows: firstly, it takes the work piece to the pallet of the Transport system station, where it is able to give it the correct orientation so that the elements remain firmly attached; then the robot puts the piston, the spring and the cover on the work piece, in that precise order. Once the work piece is assembled, the robot can take it either to the previous station or to the next station according to the desired criteria –by color or by numbered entrance in the station.

The Storing station (Figure 18) is in charge of storing the work pieces in a rack module by means of a storage module fitted with a gripper able to detect the color characteristics of the work pieces. The module permits both a turning around and a vertical displacement. In addition, the gripper can be advanced or retracted. The rotary drive is executed by an electrical servo motor with an integrated controller. The linear drive is executed by an electrical linear axis with a separate controller. The rank has three levels and six storing places in each level. In this way the storage module picks up the work piece and stores it in a free place of the rack according to a particular order. Depending on the color base cylinder and the desired criteria the module leaves the work piece in a different level of the rack.

The ASRS20 station (Figure 18) is another storing place for the base cylinders. The Automatic Storage and Retrieval System (ASRS) has a shelf with 5 x 4 locations (20) to store work pieces. The station receives a work order via communication. The gripper permits the displacement in the three axis (X, Y, Z).

Figure 18. *Storing station (left) and ASRS20 station (right)* ***[19]***

The Transport system station (Figure 19) is sited in the middle of the whole plant. It is a transport conveyor of close circuit with a rectangular shape. It contains eight pallets turning around on the conveyor and able to transport the work pieces from one station to another. There are six stop places -two of them are intended only for waiting; in the other four, a station is sited. The pallets wait for the work pieces supply, and then transport them to the next station. Because of the waiting, there can be queues in any stop place. The material flow within the FMS-F lines is implemented to the pallet transport system. The belt segments are driven by 4 AC motors. Communication with the stop points is made via Automatic Single interface (AS-interface), and the pneumatic stoppers are controlled using industrial valve terminals.

The Mitsubishi RV-3SDB-S15 (Figure 19) is a 6-axis industrial robot in charge of transporting the raw work pieces from the Buffer station to the CNC milling machine, and then transporting them from CNC to the stack magazine of the Distributing station or to the third buffer of the Buffer station. The robot displaces horizontally by means of a conveyor that permits it to achieve all the positions.

Figure 19. *Transport System station (left) and Mitsubishi RV-3SDB-S15 (right)*

The Buffer station (Figure 20) is the starting point of the whole plant. It is in charge of delivering of raw work pieces by means of two input conveyor modules. These modules retain the pieces with a separator forming a queue and only supply one work piece each time. The separator is actuated via a short-stroke cylinder with a reversing mechanism. There is a third conveyor module –an output conveyor- that receives work pieces and stores them. Each conveyor belt of the input and output conveyor modules is driven by a DC gear motor.

The EMCO CONCEPT MILL 105 (Figure 20) is a CNC milling machine processing the raw work pieces provided by the Mitsubishi RV-3SDB-S15 in a desired criterion. The CNC opens and closes the door automatically.

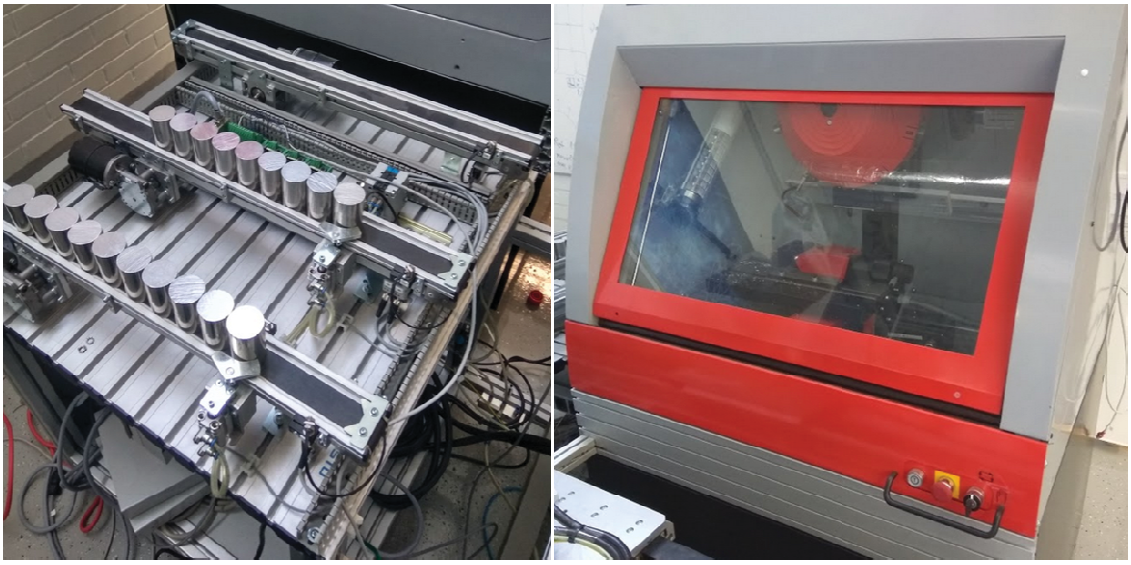


Figure 20. *Buffer station (left) and CNC machine (right)*

The next section concentrates on some considerations that should be taken into account about the I/Os of the stations.

4.3 Inputs and outputs

The first step to design the case study in formal methods is to know the I/Os in order to control each station of the systems. In this case they are provided by the designers of the MPS (FESTO).

Each I/O is represented in TNCES as it was shown in Figure 9 (section 3.3.3). For controller modeling, the I/Os are provided by FESTO, but for the plant modeling they have to be flipped. In other words, the controller inputs will be the plant outputs, and the controller outputs will be the plant inputs. This is the first point to be considered.

By way of example the next table represents the I/Os of the Distributing station controller. In the table, we can see: 1) the assigned address of the data of the controller indicating if it is an input or an output; 2) the symbol that has been assigned to the data for a better understanding; and 3) the description about the operations that the controller does when the input or the output is activated.

Table 1. *Input and outputs of the Distributing stations*

Address	**Symbol**	**Description**
I0.1	Mag_back	Magazine is in back position
I0.2	Mag_front	Magazine is in front position
I0.3	Vacuum	A cylinder base is sucked at the swivel drive
I0.4	Arm_take	Swivel drive is in magazine position
I0.5	Arm_put	Swivel drive is in position to the next station
I0.6	Mat_sen	ON=no cylinder bases in the feeder
I0.7	Follow	Light barrier to the following station
I1.0	START	Start button
I1.1	STOP	Stop button
I1.3	RESET	Reset button
Q0.0	Mag	Solenoid of the magazine cylinder
Q0.1	Vacumon	Solenoid switchs the vacuum on
Q0.2	Vacumoff	Solenoid switchs off the vacuum
Q0.3	Armleft	Solenoid moves swivel drive to the magazine
Q0.4	Armright	Solenoid moves swivel drive to the next station

The next step is to represent them in TNCES modules, but there are some considerations to be taken into account in the plant modeling. Sometimes controllers and plants of the same system do not share the same I/Os. It is clear that the controller model has its own I/Os, but the plant model may have not the same ones. It is due to the interconnections between systems. Whenever there are different plants controlled by different controllers and the plants are interconnected with each other –as it happens in the case study- there may be interconnections between the I/Os of the different stations as well. A controller of a station may activate an input of another plant different from that it is controlling. This situation creates a conflict between controllers and plants that has to be solved.

In this case study, three possibilities could happen regarding this conflict: 1) an output of a controller activates an input of a plant different from its own one; 2) the input of a

controller is activated and deactivated by a plant different from that it is controlling; 3) an input of a controller is activated by a plant and deactivated by another different plant.

In order to understand this, three examples of each possibility are explained. In the controller of the Testing station are, among others, the next I/Os:

- "Previous" is a controller output indicating the previous station that the Testing station is enabled. "Previous" has to be an input for the plant, but not for the Testing plant -it should be an indication for the previous station (Distributing station); that is to say, the controller of the Testing plant activates an input of the plant of Distributing station –an output of a controller activates an input of a plant different from its own one.
- "Safety_barrier" is a controller input representing a safety light-barrier sensor. It is detecting when the charger of the Distributing station is in the "next station" position, in order to indicate the lifting module of the Testing station not to lift because both it and the charger may collide. It is an output to be activated or deactivated according to the position of the charger in the Distributing station, an actuator of a different plant –the input of a controller is activated and deactivated by a plant different from that it is controlling.
- "Part_av" is a controller input indicating that a cylinder base is available by means of a sensor. It is an output that has to be activated from the plant that leaves the cylinder base in front of the sensor, and has to be deactivated from the plant that turns the cylinder base away from the sensor. In this case the Distributing station leaves the cylinder base and the Testing station takes it –an input of a controller is activated by a plant and deactivated by another plant.

In all these cases the respective I/Os have to be added to the plant modeling of the Distributing station -they are data from the controller of the Testing station that condition the plant of the Distributing station. In the whole system of this case study some similar circumstances happen in the same way.

One last appreciation is that data can be either monostable or bistable variables. The designer has to be able to identify these variables and to know how to manage each of them.

The I/Os of the remaining stations can be found in Appendix A.

4.4 Controller modeling

As stated above, MNG can convert directly to NCES the diagrams designed with LD or STL. In this way it is only necessary to introduce the controller of each station in the MNG in order to obtain the design in NCES formalism. But in this thesis all the models

of the controllers have been manually designed due to the above-mentioned reasons (section 3.3.1).

To do so, the NCES controllers are considered as a module representing a sequence of actions activated or deactivated in a particular order. Their design is based on numbering the steps of the real plant operation and on sequencing them so that the controller acts on the plant indicating what inputs are enabled and what are disenabled at each of these steps.

In order to be able to understand that, the example of the controller design of the Distributing station is featured below.

First of all, the real operation of the plant has to be understood and it has to be separated into steps and ordered in the particular sequence. As it was detailed in the previous point, the Distributing station has the function of displacing the work pieces from a stack magazine to the next station by means of a charger arm provided with a vacuum. The sequence of the plant operation is as follows:

1. If one work piece is identified in the stack magazine, the rotary drive swivels to the position “Next station”.
2. The stack magazine pushes a work piece.
3. The rotary drive swivels to the position “magazine”.
4. The vacuum is switched on.
5. The rotary drive swivels to the position “Next station”.
6. The vacuum is switched off.
7. The sequence repeats itself from step 2.

In addition, as the control software is controlling a plant representing a cycle of a manufacturing system that repeats each determined time, two things have to be taken into account:

- The firing of the first transition is executed to get the data of the First Cycle ready, because the plant could have another configuration due to the last time the plant was used.
- When the plant repeats the cycle and its variables are in their initial states, the loop in the controller has to be closed. Hence, the controller and the plant can evolve in a non-stop operation.

The only thing left then is to know the order it should follow -in other words, to know what conditions are necessary to activate or deactivate the appropriate variables in order to make the plant evolve the way the real operation does. For the Distributing station, the formal controller model in TNCES representation would be like those represented in Figure 21.

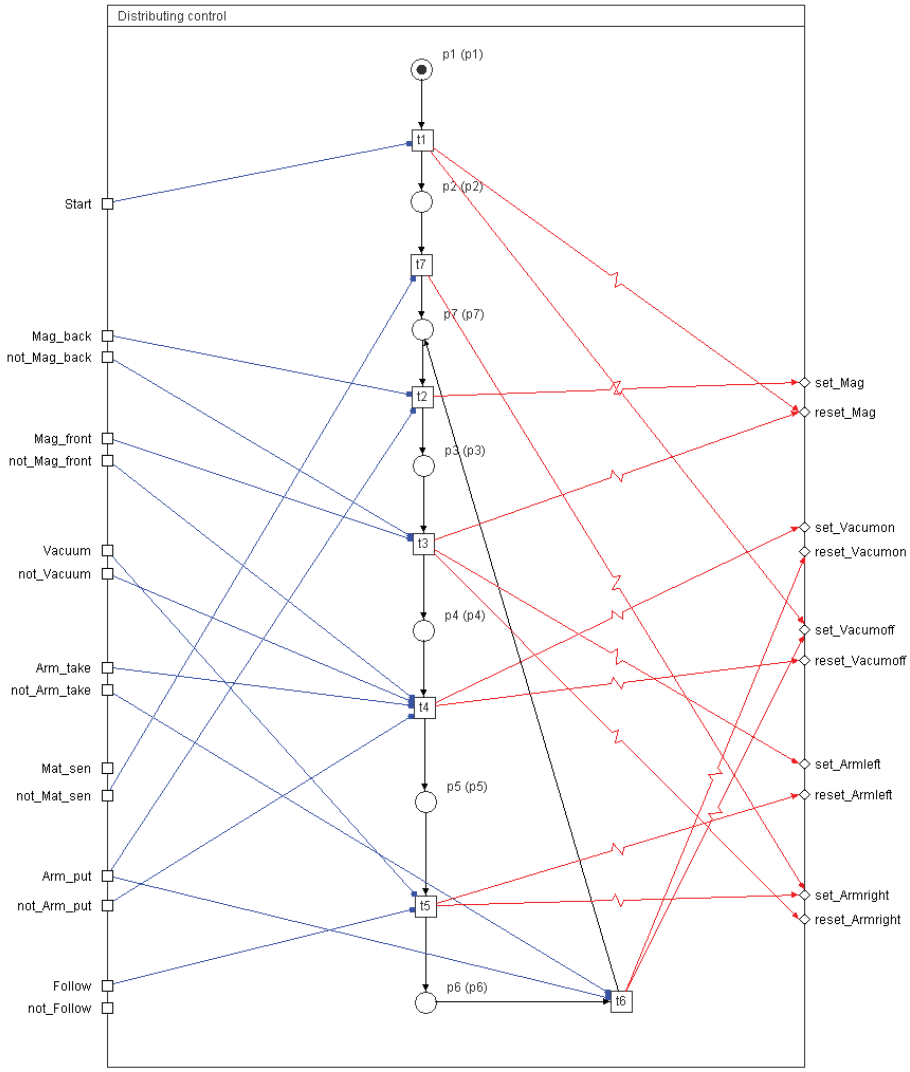


Figure 21. *TNCES representation of the Distributing station controller*

As shown in the Figure, when the Start is given the variables of the First Cycle are activated. If there is a work piece in the magazine (not_Mat_sen) then the solenoid moves the swivel drive to the next station (set_Armright). Whenever the arm is in the "next station" position (Arm_put) the controller resets the arm variable (reset_Armright) and sets the solenoid of the magazine cylinder (set_Mag). When the cylinder is advanced (Mag_front) the cylinder variable is reset (reset_Mag) and the solenoid moves the swivel drive to the magazine (set_Armleft). Whenever the arm is in "magazine position" (Arm_take) the controller resets the solenoid of the cylinder (reset_Armleft) and activates the vacuum (set_Vacumon and reset_Vacumoff). When the vacuum is activated (Vacuum) the solenoid moves the swivel drive to the next station (set_Armright), where the vacuum will later be deactivated (reset_Vacumon and set Vacum_on). Then the loop closes and it is ready to start again pushing a new work piece.

The fomal controller models of the rest of the stations are provided in Appendix B.

4.5 Plant modeling

The plant modeling is totally distinct –in this modeling the plant is really abstracted. It is a more intuitive and more logical design, as it was seen in section 3.3.2. To do that the main modules of the system must be firstly identified and then its behavior has to be interpreted in a separate way.

This design is more laborious and more interesting than the controller design. For this reason each plant modeling will be detailed in the following subsections.

4.5.1 Distributing station

In the Distributing station, two modules are identified -the stack magazine module and the charger module.

The stack magazine module has two states -the pushing cylinder could remain retracted (p1) or extracted (p2). The charger module is divided into two representations: the first one has three states representing its position (it could be in "Stack magazine" position (p1), in "Next station" position (p2) or between both positions (p3) –p3 has been contemplated because it is considered that the arm remains in the middle of both position in the first cycle); the second representation has two states representing whether the vacuum is switched off (p4) or switched on (p5).

In respect to the time measuring, the pushing cylinder is actioned in 900 milliseconds; the arm needs almost 2 seconds to move from one side to the other; and the vacuum is switched on/off in 100 milliseconds.

Figure 22 shows the modeling plant of the Distributing station in its initial state. As it can be observed the representations are inside different modules –Stack magazine module and Charger module. Each transition is conditioned by the plant inputs and fires different events that activate or deactivate the plant outputs. It also can be seen the measured timed in the input arcs of the transitions. The rest of the formal plants models can be found in Appendix C.

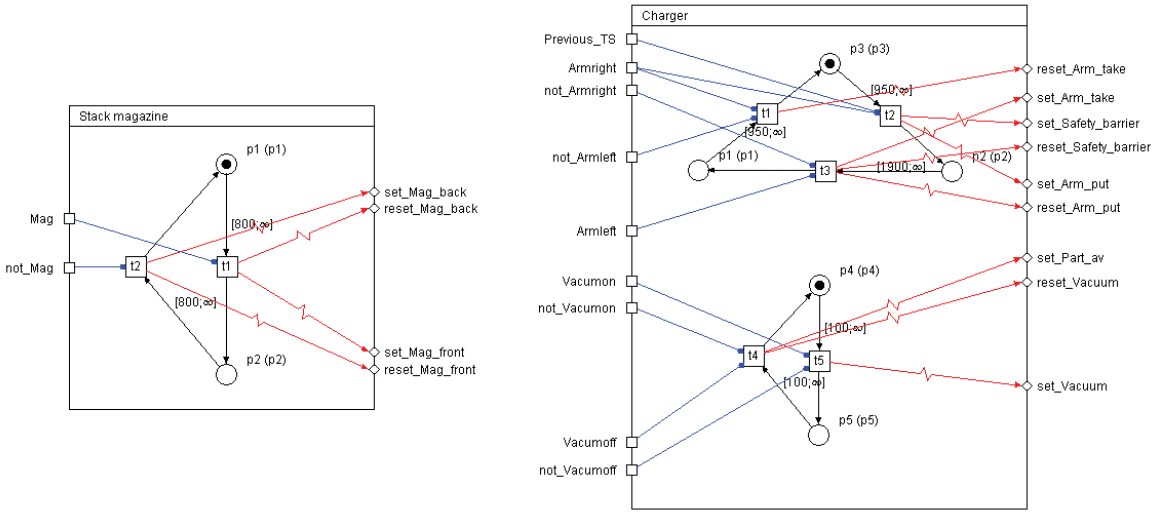


Figure 22. *TNCES representation of the Distributing station plant*

4.5.2 Testing station

In the Testing station, four modules are identified: recognition module, lifting module, measuring module and pushing module.

The recognition module has three states –there is not any work piece (p1), there is a black work piece (p2) or there is a non-black work piece (p3). The lifting module has two states –it is down (p1) or up (p2). The measuring module has three states –there is not any work piece (p1), there is a work piece and its height is correct (p2) or there is a work piece and its height is not correct (p3). The pushing module has two states –it is retracted (p1) or advanced (p2).

In respect to the measured time, measuring module and recognition module are instantaneous actions; the lifting module spends 100 milliseconds advancing and retracting; and the lifting needs 2 seconds to move from one position to the other.

4.5.3 Handling station

In the Handling station, two modules are identified: Colour_identification module and PicAlfa module.

The Colour_identification module has the same states as the recognition module of the Testing station. The PicAlfa module is divided into two representations: the first one has three states representing its position –PicAlfa is in “previous station” position (p1), in “following station” position (p2) or in “sorting” position (p3)-; and the second representation has two more states –the lifting cylinder with the gripper is up (p4) or down (p5).

In respect to the measured time, the Colour_identification module is instantaneous and the PicAlfa module needs 3 seconds to move from “previous station” position to “following station” position, and also needs 2 seconds from “following station” position to “sorting” position. The lifting module of PicAlfa takes 1 second from up to down.

4.5.4 Processing station

In the Processing station, three modules are identified: rotary indexing table module, testing module and rubbing module.

The rotary indexing table module has five states –there is not any work piece (p1), there is a work piece and it is rotating from “entry and exit” position to “testing” position (p2), there is a work piece in “testing position” (p4), the work piece is rotating from “testing” position to “drilling” position (p3) or the work piece is in “drilling” position (p5). The testing module has two states –it is testing (p2) or not (p1). The rubbing module has four states –there is not any work piece in “rubbing” position (p1), there is a work piece in “rubbing position” and the rubbing machine is down but not working (p2), there is a work piece in “rubbing position” and the rubbing machine is down working (p3) or there is a work piece in “rubbing” position and the rubbing machine is up (p4).

In respect to the measured time, the rotary table takes 700 milliseconds to turn 30 degrees and the actions of the other modules are virtually instantaneous.

4.5.5 Robot and Assembling stations

The Robot station and the Assembling station have been considered as one plant because they are much interconnected to each other.

In the Robot station, three modules are identified: the gripper module, the Colour_identification module and the robot module.

The gripper module has two states –it is open (p1) or closed (p2). The Colour_identification module has two states –there is a work piece (p2) or not (p1). The robot module has fifteen states, each of them representing the movement of the robot among the different positions it can achieve. Table 2 shows the respective places in TNCES representation of the plant corresponding to the operations of the robot (states) in the pertinent order.

In the Assembling station, two modules are identified: the spring magazine module and the cap magazine module.

Both modules have two states –the magazine is retracted (p1) or advanced (p2).

In respect of the measured time, in this case the velocity of the robot can be configured in the desired criteria of the control engineer. The stack magazine modules spend 900 milliseconds in providing their elements.

Table 2. *States of the plant of the Robot station*

Places	**Description**
p1	Robot waits to receive one work piece
p2	Robot moves to “pick up” position
p3	Robot moves to “change gripper” position
p4	Robot changes the gripper
p13	Robot turns the work piece to the correct orientation
p14	Work piece is in the correct orientation
p5	Robot moves to the “assembling” position
p6	Robot moves to the “piston pallet” position
p7	Robot moves to the “spring magazine” position
p8	Robot moves to the “cap magazine” position
p9	Robot changes the gripper
p10	Robot turns the cap to the correct orientation
p15	Cap is in the correct orientation
p11	Robot moves to the “next station” position
p12	Robot moves to the “pallet” position

4.5.6 Storing station

In the Storing station, three modules are identified: the mini_slide module, the holder module and the storing position module.

The mini_slide module has two states –it can be advanced (p1) or retracted (p2). The holder module has three states –there is not any work piece (p1), there is a work piece

and the color is identified (p2) and the gripper takes the work piece but the color identification remains until it is stored (p3). The storing module has seventeen states. It is because this station can move vertically in six different levels and can turn horizontally in six different positions. Each level in vertical is designated by three bits and so is each position in horizontal. The corresponding bits of each position are represented in Figure 23:

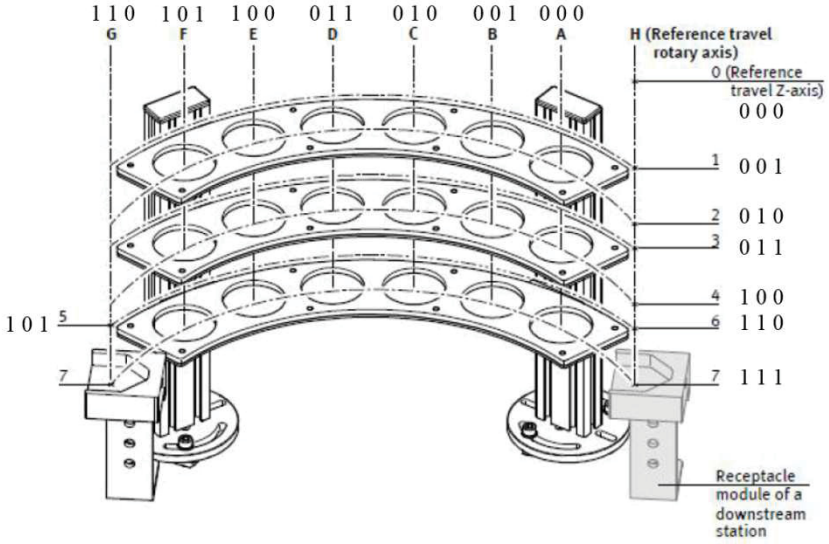


Figure 23. *Bit assignment of rows and columns of the Storing station*

These are the states of the storing position module: the storage module is waiting for a work piece in G5 (p11), the storage module moves to G7 to pick up the work piece (p12), the storage module is in G7 (p13), the storage module moves to G6 (p14), the storage module is in G6 (p1), the storage module moves to one of the three levels according to the color identification (p2, p3, p4), the storage module moves to the first free position in the shelf (p5, p6, p7, p8, p9, p10), the storage module is in the respective position (p18), the storage position leaves the work piece (p15), the storage module turns to column G (p16) and finally, the storage module moves vertically to its initial position (p17).

The measured time depends on the position the module is storing the current work piece in.

4.5.7 ASRS20 station

In the ASRS20 station, one module is identified: the ASRS20 module. This module is similar to the previous one –the storing position module of the Storing station- but it has been designed in an easier way in order to compare the possibilities of each one.

This module is divided into four representations, each one representing the position of the module and the state of its gripper. Hence, the first submodule represents the posi-

tion in the Y axis –the module is in back position (p1) or in front position (p2); the second submodule represents the position in the X axis –the module is on the right (p3) or on the left (p4); the third submodule represents the position in the Z axis –the module is in the down position (p5) or in the up position (p6); and finally the fourth submodule represents the state of the gripper –it can be open (p7) or closed (p8).

The measured time depends on the position the module is storing the current work piece in.

4.5.8 Buffer Station, Mitsubishi RV-3SDB-S15 and EMCO CONCEPT MILL 105

The Buffer station, the industrial robot and CNC machine have been considered in the same plant because they are highly interconnected to each other.

In the Buffer station, two modules are identified: buffer 1 module and buffer 2 module. Both of them are identical and they represent the input conveyors of the station. This TNCES representation is a bit different to the rest because this one is the starting point of the whole system. It has to indicate how many raw work pieces are going to be processed. Hence, the representation has five places –one of them indicates the number of raw work pieces (p1), another represents that there is a work piece in the separator (p2), another indicates that there is not any raw work piece in the "pick up" position and that the separator is enabled (p5), another represents that the separator is open (p3) and the last one indicates that there is a raw work piece in the "pick up" position waiting for be taken by the robot (p4) and consequently the separator is closed and not enabled.

In the Mitsubishi RV-3SDB-S15, one module is identified –the robot module called "Mitsubishi". The operation of the robot is basically to transport the work piece from two possible input places (Buffer 1 or Buffer 2) to the CNC machine, and then transport them from CNC machine to two possible output places (Buffer 3 or Distributing station). Hence, the module has ten states: the robot is idle (p1), the robot moves to Buffer 1 or Buffer 2 (p2 and p3 respectively), the robot takes the work piece (p4), the robot moves to CNC machine (p5), the robot leaves the work piece and waits (p6), the robot takes the work piece from CNC machine (p7), the robot moves to Buffer 3 or Distributing station (p8 and p9 respectively) and the robot leaves the work piece (p10).

In the EMCO CONCEPT MILL 105, one module is identified: CNC machine module. This module is composed of three representations: 1) the door, with two states –it is open (p1) or closed (p2); 2) the clamping element also with two states –it is unclamped (p3) or clamped (p4); and 3) the sequence of the machine operation, with four states – the machine remains (p5), the automatic mode is switch on (p6), the axis are referenced (p7) and the program runs (p8).

In respect to the time, the buffer station takes 2 seconds to provide a raw work piece, whereas the CNC machine and the velocity of the robot can be configured.

4.5.9 Transport system station

In Transport system station, two modules are identified: the transport system module and the stoppers module. The transport system module is the module representing the conveyor system. This conveyor is very similar to that depicted in Figure 8 (section 5.2). In this case it has six operation positions, but it should be taken into account that in each position a congestion may be formed. In other words, queues may be formed if a previous station finishes its operation before the next one does. For this reason in the TNCES representation the conveyor model acts as a conveyor with twelve capacities - six places represent the "operation" positions (p1, p3, p5, p7, p9 and p11) and six places represent that a pallet is in the queue of each "operation" position (p2, p4, p6, p8, p10 and p12). In each "queue" position two pallets at most may be waiting, so the maximum capacity of the respective places is two tokens, unlike the "operation" positions –whose maximum capacity is one. Moreover, as it was explained in Figure 8, places representing whether a capacity is busy or idle are necessary, so there is a place for each "operation" position and for each "queue" position (p13, p14, p15, p16, p17, p18, p19, p20, p21, p22, p23 and p24). An initial state is required in the representation, hence in this case it is supposed that 1) there is one pallet in the "operation" positions 1, 5 and 6; 2) there are two pallets in the "queue" positions 1 and 6; and 3) there is one pallet in the "queue" position 5. The stoppers modules determine which stoppers are advanced and which are not. For each position the stopper can be retracted (p1, p3, p5, p7, p9 and p11) or advanced (p2, p4, p6, p8, p10 and p12).

In respect to measured time, the pallet needs 2 seconds to achieve the next "queue" position from an "operation position", and 1 second to achieve the next "operation position" from a "queue" position.

4.6 Closed-Loop systems modeling

Once both the controller design and the plant design are modeled, they have to be interconnected. To do so, both files are opened in MNG and each controller condition is connected to its respective plant condition, and vice versa.

As it was explained, another module called "update" is added between both modules. The output event of both models (controller and plant) is connected to the input event of this module (event), and the output event of this module (changed) is connected to the input event of the models.

Once interconnected, the validation of the designs has to be checked. First, the well-functioning of the designs is proven by means of the training utility of MNG. The step

duration is fixed and the simulation makes the design evolve step by step. In this case, it has been indicated the step duration as 1,000 milliseconds. With this simulation it is proven the non-existence of deadlocks and the good evolution of the TNCES.

The example of the closed-loop system of the Distributing station is represented in Figure 24. Once it is interconnected and the well-functioning is proven, it is ready to be applied to model-checking.

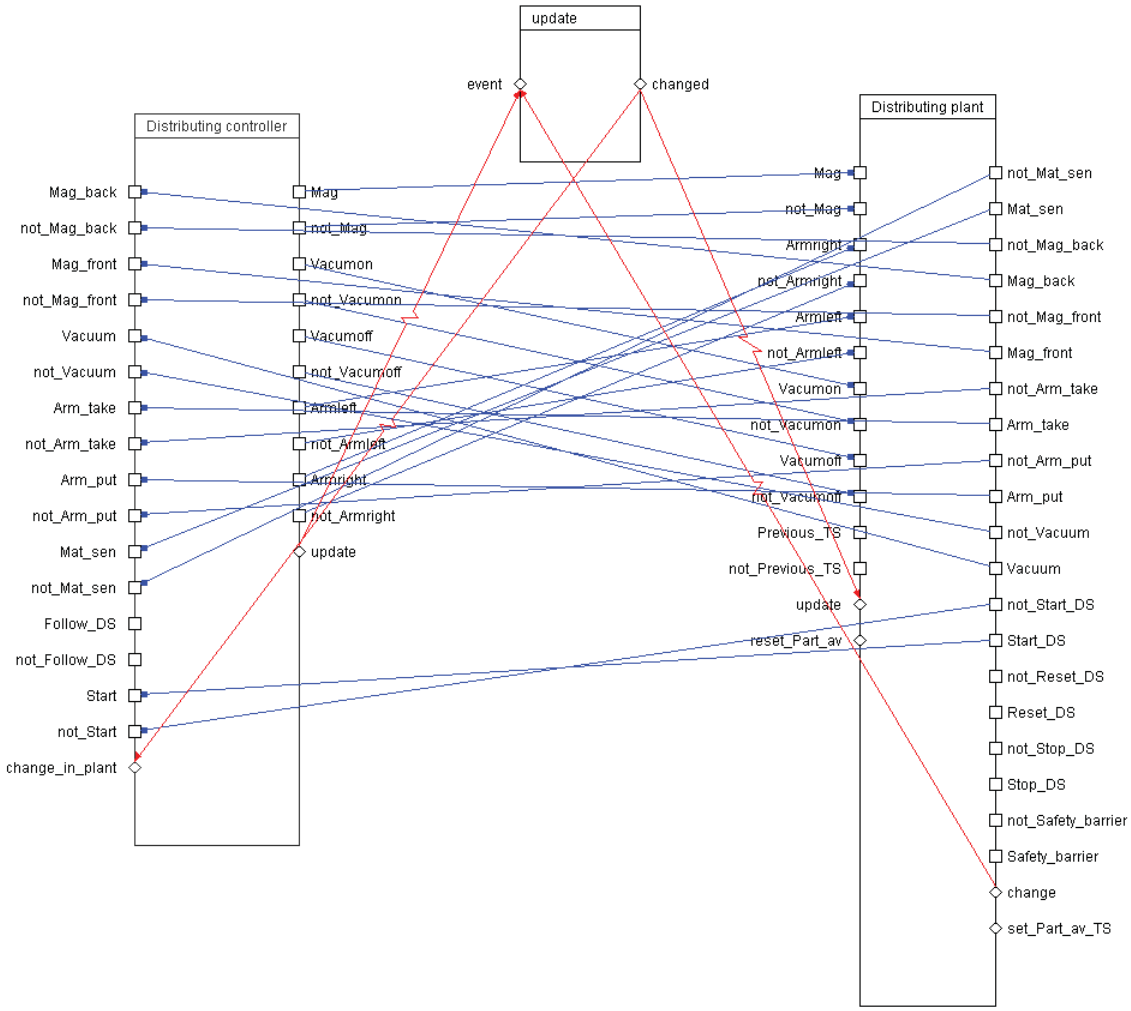


Figure 24. *Closep-loop system of the Distributing station*

The next chapter is about both the model-checking of the models and the results obtained from this model-checking.

5. VALIDATION OF RESULTS

This chapter is focused on the validation of the obtained results. To do so the results are subjected to analysis and correction. It is a very important part since it demonstrates that a correct formalization of the system has been done.

Once obtained the closed-loop system it is ready to be subjected to model-checking. The result of the model-checking will be the obtainment of both the state space of each model and the timing diagrams. The objective is to validate the designed models checking 1) the non-existence of deadlocks; 2) that the operation of the formal models is equal to the real operation of the production line; 3) that the measured times are correct; and 4) that the state spaces have the expected structure as well as that the timing diagrams have a good representation.

All of this is validated in the own MNG. Furthermore, a clear way to validate the formal methods is by means of a visualization. MNG offers the possibility of designing a visual representation that can evolve according to the evolution of a TNCES model.

The next sections explain the procedure of the model-checking, the obtained results as well as the design and the evolution of the visualization.

5.1 Model-checking and results analysis

The first step is to check that the TNCES representation evolve in a correct way, the non-existence of deadlocks and that the measured times adjusts to the real operation. To do this the formal model is introduced in the Training Utility of MNG. Then the simulation starts and it is checked that the evolution does not stop in any moment and that it has a good evolution. With that the non-existence of deadlocks is checked.

It is ready to the model-checking. so it is introduced in the NCES Analyzer of the MNG –the tool that analyzes the TNCES representations in a mathematic logic due to the properties of the axioms of the TNCES. This tool automatically results in the state space and the timing diagram.

The next subsections explain how these diagrams are obtained and what information they provide.

5.1.1 State spaces

The state space of a TNCES model is a representation of the set of states the TNCES model could remain in. In addition the sequence indicating what state results into other/s state/s is marked. So this representation is composed of the places representing a different state of the TNCES model and the arrows indicating the sequence. An example of can be found in Figure 9 (Section 3.3.2); it represents the state space of the conveyor studied in the same section.

It is a very interesting graph, many conclusions can be extracted of these type of representations. One of the most important requirements it must fulfill is that the representation has to constitute a closed-loop.

Figure 25 represents the state space of the formal model of the Distributing station.

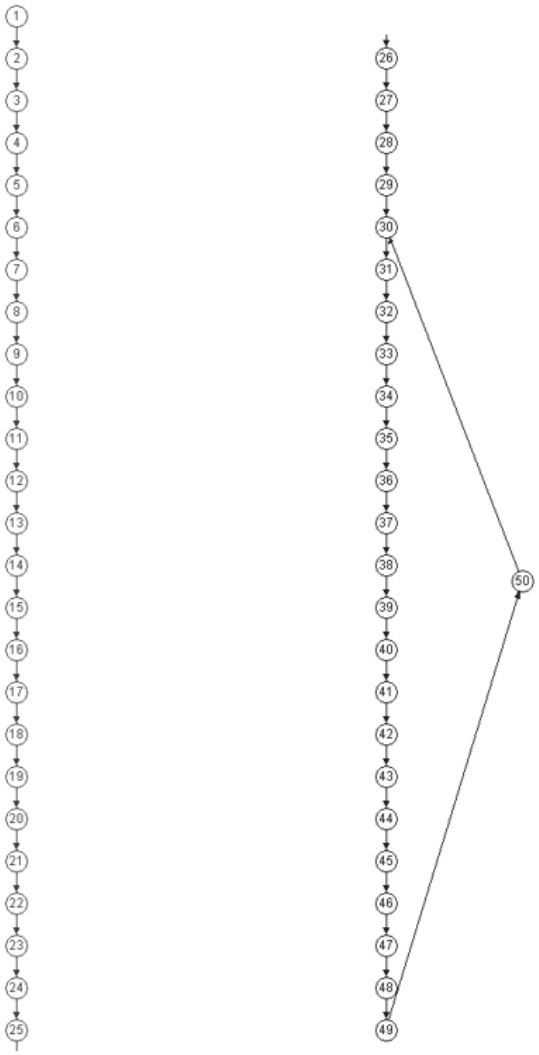


Figure 25. *State space of the Distributing station*

As shown, the representation has fifty places, meaning that the formal model of the Distributing station has fifty different states. It also can be observed that there is a closed-loop cycle from the p30 to p50, what represents the plant operation cycle. In other words, the Distributing station needs some conditions to be ready from the very beginning (p1 to p30) to the beginning of the closed-loop cycle. This cycle starts when there is a piece available and the charger is enabled. The places 30 and 50 are respectively the source state and destination state for representing the timing diagrams of the models.

These graphs permit to know whether the models have been correctly designed. Appendix D contains the state spaces of the Testing station, the Handling station, the Processing station and the ASRS20 station. They are commented below.

The state space of the Testing station has sixty one places and four closed-loop cycles are reflected. These different paths represent 1) the work piece is black and has the correct height; 2) the work piece is black but has not the correct height; 3) the work piece is red or silver and has the correct height; 4) the work piece is red or silver but has not the correct height. All these paths starts in p4.

The state space of the Handling station has seventy nine places and it is reflected two paths. That is because plant acts the in a different way depending on whether the work piece has passed the test of the Processing station or not. The cycles starts in p4.

The state space of the Processing station has thirty nine places and solely one path. The cycle starts in p6. The state space of the ASRS20 station has thirty six places and only one path. The cycle starts in p4.

The next section explains how to get the timing diagrams from the states spaces.

5.1.2 Timing diagrams

The timing diagrams are timed state-values representations showing the change of the states and variables values along with the time evaluation.

The NCES Analyzer of the MNG results automatically in the timing diagram. The period of time of the representation is determined by the source state and the destination state of the closed-loop cycle of the state space. The tool offers the possibility of showing all the variables, but for this case the most representative variables of each station are represented.

In the case of the Distributing station, these states are p30 and p50 respectively. Figure 26 shows the timing diagram of this station.

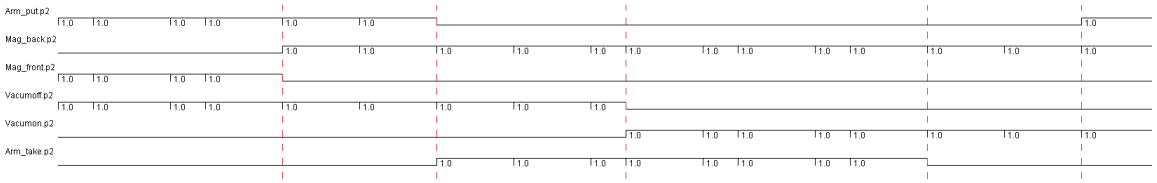


Figure 26. *Time diagram of the Distributing station*

The diagram represents the change of the most representative variables of the Distributing station along the work cycle.

In this case there is only a cycle because all the work pieces are managed in the same way. In Appendix E the timing diagrams of the Testing station, the Handling station, the Processing station and the ASRS20 station are presented. In some of them there is more than one time diagram due to the different paths that the state spaces have. It means that the variables change in a different order depending on the plant operation.

The next section concentrates on the design of a visual representation and its evolution. It is an illustrative way to validate the formal models.

5.2 Visualization

Even though the TNCES models have an intuitive representation, they are not sufficient illustrative. It would be much appreciated if a visual representation of the plant evolved the same way as the real operation of the system while the formal model is evolving.

MNG offers the possibility of designing a visual representation in the Visualization Tool and make it evolve. To do so, on the one hand a scenario in the Visualization screen is designed; and on the other hand the formal model in the Training tool of MNG is opened. After that, they are connected to each other.

In order to design the visualization all the characteristic elements of the plant have to be drawn in the same position they have in the initial state. Then the visualization and the formal model have to be connected to each other by means of the Animator tool. After this the mandatory rules have to be specified. These rules are the instructions followed by the visualization according to the changes in the formal model. The rules are composed by two elements –on the one hand the state model plant remains in, and on the other hand the changes that the visualization has to execute. These changes may be the displacement, the turn, the color changing or the deformation of the visualization elements. In this way the rules are determining the changes in the visualization according to the state of the formal model. This results in the evolution of the visualization according to the changes in the model plant simulation. Some examples of the mentioned rules are represented in Figure 27.

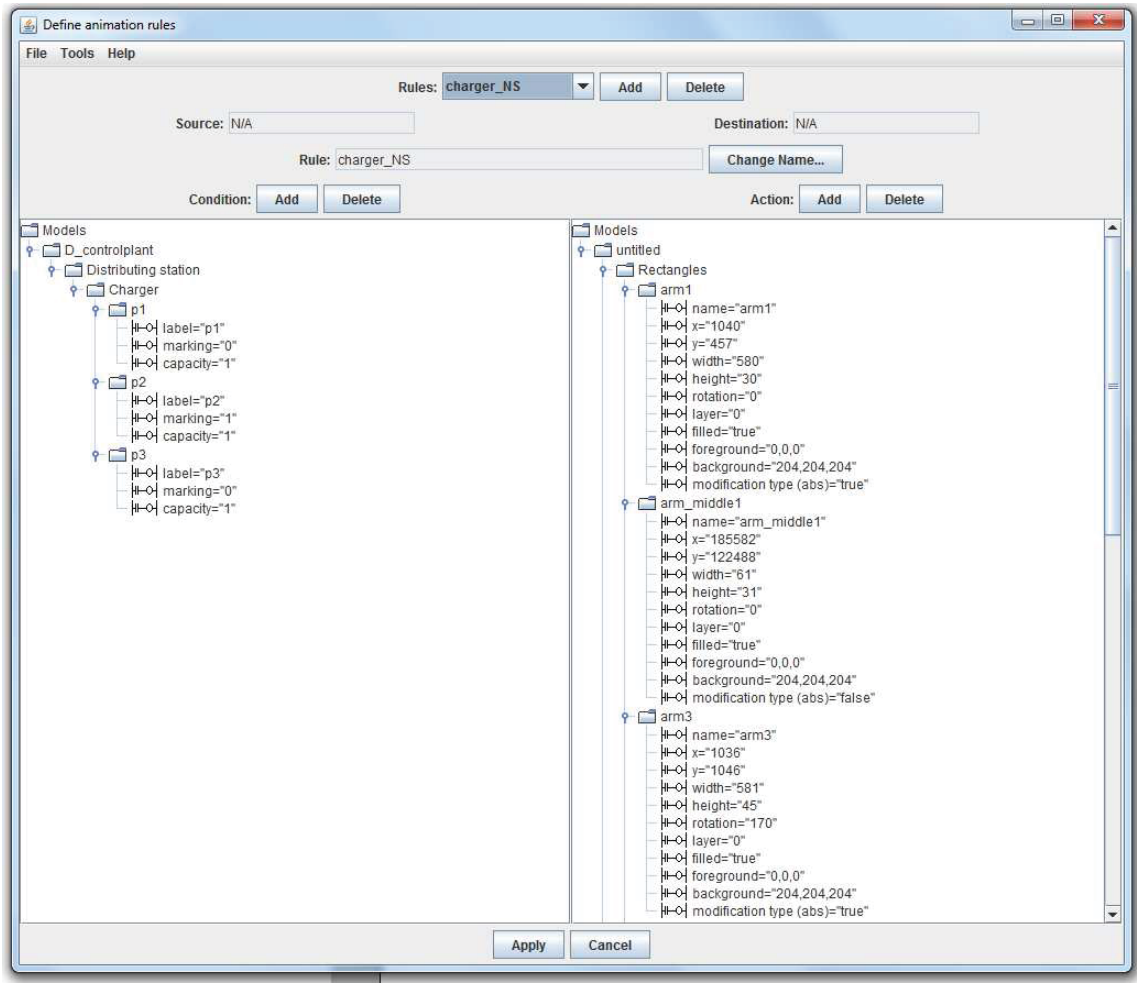


Figure 27. *Specified rules of a visualization*

In Figure 28 the evolution in the visualization of the Distributing station made in the Visualization tool of MNG is represented. It represents a plant view and a frontal view of the main elements of the station changing according to the evolution of the formal model.

To the right of each screen the formal plant model appears representing the state it remains in. The formal model is evolving in the Training Utility tool of MNG and causing changes in the visualization according to the designed mandatory rules.

As it is represented Figure 28 there are ten changes in the visualization, but when it arrives to the screen 10 it is inside the closed-loop cycle and it so it will repeat starting from screen 5. It is the same way as the real operation of the station would work and it is represented in an illustrative way.

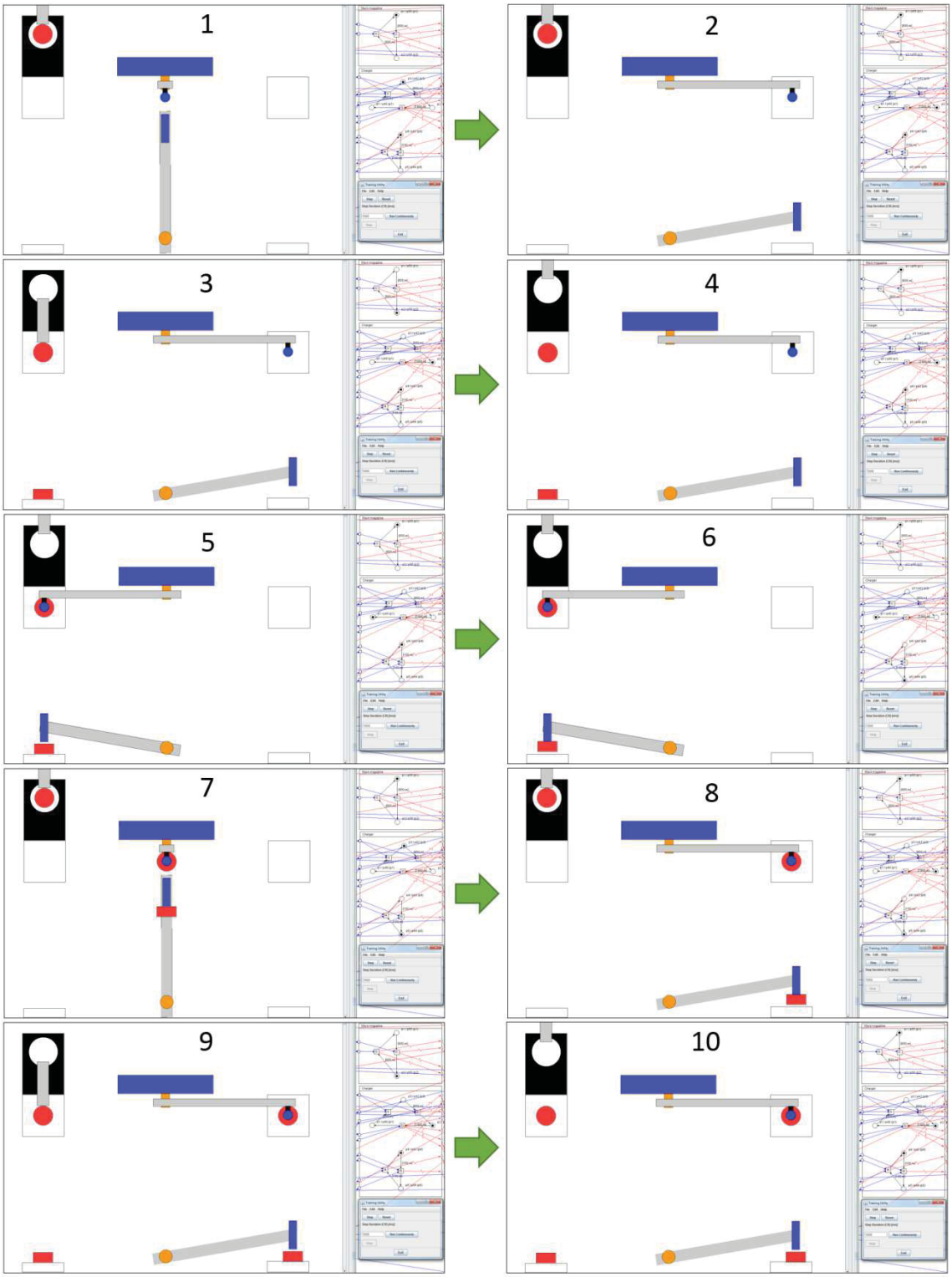


Figure 28. *Visualization of the Distributing station*

6. CONCLUSIONS

This chapter presents the conclusions of this work. After a short review explaining the followed steps an analysis of whether the aims of the thesis have been achieved or not is made. Finally, some possible future works –that would be good contributions to the scope of this thesis- are introduced.

6.1 System review, results and achievements

This thesis concentrates on the formalization of an automatic production and on the simulation of its evolution in a visual representation. The automatic production line is the FASTourim, composed of a set of stations. To achieve the main objective of the thesis the production line should be studied firstly in order to select the most appropriate modeling formalism. The case study is a DEDS –systems evolving by the occurrence of events and that can be represented as a succession of discrete states. Due to this, after a comparison of the most common formalisms the chosen one is the TNCES –a formalism both able to represent the system in a timed-illustrative way and supported by mathematical logic. The formal representation is evolved by means of the closed-loop system –a system composed of a controller model and a plant model that are interconnected creating a closed-loop cycle making each other evolve. Lastly, MNG is used to design the formal models, to interconnect and make them evolve, to extract and analyze results, and finally to design and simulate a visual representation.

The steps that have been taken to obtain the results are:

1. Formalization of the controller –obtaining the TNCES representation of the controllers of each station in the production line.
2. Formalization of the plant –obtaining the TNCES representation of the plants of each station in the production line.
3. Interconnection of the controller model and plant model –obtaining the closed-loop systems.
4. Validation of the closed-loop systems –obtaining the checking of non-existence of deadlocks in the formal models.
5. Model-checking of the closed-loop systems –obtaining the state spaces and the timing diagrams of the stations.
6. Design and simulation of the visualization –obtaining a visual representation that evolves as the represented station does.

The results of the thesis have been the formal models of the ready-to-use stations. They have been validated and users can do the verification and validation of their designed control software using them instead of using testing. Hence, the main objective of this thesis has been achieved.

In this thesis the benefits of formal methods are demonstrated as well. Once the formal models are designed and introduced in the MNG they may be subjected to model-checking. In addition, TNCES can be converted into other programming languages compliant with other most powerful software tools. Computers can extract much more information from formal designs than users in control tests do. Furthermore it is true that formalization takes time and it may be too complicated for big manufacturing systems; but despite that this strategy will ultimately benefit automation factories..

6.2 Future works

This thesis is within the framework of the studies aiming to demonstrate the efficiency of formal methods in the verification and validation of manufacturing systems. After the completion of the thesis, it allows to develop many studies in the field. Some of them are commented below.

One of the next steps may be to interconnect all the designed formal models in order to obtain the whole production line in the same system. It would be a model of all the set of plants and their respective controllers. The interconnection between controllers and other plants different from that they are controlling should be taken into account. It would be interesting to have a big model evolving completely from the supply of the work pieces to the storing.

Other possibility is to try and change the order of the production line. As it was stated before the stations are modular and removable. It would be interesting to change the established stations order configuring another layout. It could be used to validate the formal models designed in this thesis in a more efficient way.

Furthermore, as it was above-mentioned, MNG can convert the TNCES in XML. This language permits the possibility of introducing the models in other more powerful software tools. In this way a deeper analysis of the representations may be performed. It would help the implementation of improvements in the automatic production line control.

Another future project would be to make the tridimensional virtual designs of the stations evolve by means of the formal models. This virtual designs are provided by FESTO. They are tridimensional representations of each station used to validate the designed controller of the users. The FASTorium is used in an academic environment and it is another functionality for the students –to control tridimensional virtual designs prior to the current production line. The designs are found in the software Cyros Mechatronics. It offers the possibility of connecting to them by means of OPC (Object Linking and Embedding for Process Control). But the OPC server used by software is a predetermined server and it has all their clients assigned. Despite that the work would consist in creating a new OPC client and incorporating it in the server. In this way we could observe the evolution of virtual designs scripted by formal model systems.

REFERENCES

[1] Hanisch, H-M., Lobov, A., Lastra, J.L.M.,Tuokko, R., and Vyatkin, V. (2006) "Formal Validation of Intelligent Automated Production Systems: Towards Industrial Applications", Int. J. Manufacturing Technology and Management, Vol.8, Nos. 1/2/3, pp.75-106.

[2] Knight J.C., DeJong C.L., Gibble M.S., and Nakano L.G. (October 1997) "Why are Formal Methods Not Used More Widely?".

[3] Lobov A. (2004) "An approach to the formal verification of automated manufacturing systems with programmable control", Master of Science Thesis.

[4] Preuße S., Lapp H-C., and Hanisch H-M. "Closed-loop System Modeling, Validation, and Verification".

[5] Vyatkin V. (2007-2012) "Modelling and Verification of Discrete Control Systems with Net Condition/Event Systems and Visual Verification Framework".

[6] Younis B., Frey G., "Formalization of Existing PLC Programs: A Survey", Proceedings of CESA 2003, Lille, France, July 2003.

[7] IEEE Standard Glossary of Software Engineering Terminology, IEEE Std., New York, 1990.

[8] Tarek M. S., "Discrete Event Dynamic Systems: An Overview", May 1991.

[9] Capkovic F., "Modelling and Control of Discrete Event Dynamic Systems", BRIC, Department of Computer Science University of Aarhus. October 2000.

[10] David R., Alla H. (2005) "Discrete, Continuous, and Hybrid Petri Nets" Berlin: Springer-Verlag Berlin Heidelberg.

[11] Zhou M. (1995) "Petri Nets in flexible and agile automation", New Jersey Institute of Technology

[12] Rausch M., Hanisch H-M., “Net condition/event systems with multiple condition output”, Proceedings of the Symposium on Emerging Technologies and Factory Automation, vol.1, pp.592-600, Oct. 1995.

[13] Hanisch H-M., Thieme J., Luder A., Wienhold A., “Modeling of PLC Behavior by Means of Timed Net Condition/Event Systems”, Proceedings of 6th International Conference on Emerging Technologies and Factory Automation, pp 391-396, Sept. 1997.

[14] Vyatkin V., Hanisch H-M., Pang C., Yang C-H., ”Closed-Loop Modeling in Future Automation System Engineering and Validation”, IEEE Transactions on systems, man, and cybernetics – Part C: Applications and reviews, Vol. 39, No. 1, January 2009.

[15] Popescu C., “An Approach to Incremental Modelling of web Services Orchestration: An application to Deadlock-free Scheduling in Automated Systems”. Thesis for the degree of Doctor of Technology .Tampere University of Technology, October 2009.

[16] Lobov A., “Formal Validation of Discrete Automation Systems Applying Structural Reasoning and General Unary Hypothesis automaton Methods”, Thesis for the degree of Doctor of Technology. Tampere University of Technology, December 2009.

[17] Lobov A., Lastra J., Tuokko R., Vyatkin V., Modelling and Verification of PLC-based Systems Programmed with Ladder Diagrams, To appear in proceedings of 11th IFAC Symposium on Information Control Problems in Manufacturing, INCOM 2004, Bahia, Brasil, April 2004

[18] Quispe-Otazu, R. What are the formal methods? Bloc of Rodolfo Quispe-Otazu [Internet]. February 2009. Enabled in: http://www.rodolfoquispe.org/blog/que-son-los-metodos-formales.php

[19] FESTO Didactic Homepage: http://www.festo-didactic.com/int-en/learning-systems/mps-the-modular-production-system/stations/the-stations-in-the-modular-production-system-at-a-glance.htm

[20] FAST Laboratory TUT, http://www.tut .fi/en/fast/research/open-outcomes/index.htm

APPENDIX A: INPUTS AND OUTPUTS

Testing station

Address	Symbol	Description
I0.0	Part_av	Cylinder base is available
I0.1	Sen_mat	Check if cylinder base is black – 0 = black
I0.2	Safety_barrier	Safety light barrier
I0.3	Work_ok	Check the height of the cylinder base – 1= ok
I0.4	Lift_up	Lifting cylinder is up
I0.5	Lift_dow	Lifting cylinder is down
I0.6	Push_ba	Pushing cylinder is in back position
I0.7	Follow	Light barrier to the following station
I1.0	START	Start button
I1.1	STOP	Stop button
I1.3	RESET	Reset button
Q0.0	Liftdown	Solenoid of the lifting cylinder down
Q0.1	Liftup	Solenoid of the lifting cylinder up
Q0.2	Matpush	Solenoid of the pushing cylinder
Q0.3	Slide	Solenoid of the air slide
Q0.7	Previous	Light barrier to the previous station

Handling station

Address	Symbol	Description
I0.0	Part_av	Cylinder base is available
I0.1	Han_prev	Handling at previous station
I0.2	Han_foll	Handling at following station
I0.3	Han_sort	Handling at sorting position
I0.4	Lift_dow	Lifting cylinder is down
I0.5	Lift_up	Lifting cylinder is up
I0.6	Sen_mat	Material detection signal = 0 = back
I0.7	Follow	Light barrier to the following station
I1.0	START	Start button
I1.1	STOP	Stop button
I1.3	RESET	Reset button
Q0.0	Handpre	Solenoid of handling cylinder to previous station
Q0.1	Handfoll	Solenoid of handling cylinder to following station
Q0.2	Lifting	Solenoid of the lifting cylinder with gripper
Q0.3	Gripper	Solenoid of the gripper = 1 = open
Q0.7	Previous	Light barrier to the previous station

Processing station

Address	Symbol	Description
I0.0	Part_av	Cylinder base is available
I0.1	Wor_rub	Cylinder base is at rubbing position
I0.2	Wor_test	Cylinder base is at testing position
I0.3	Rub_up	Rubbing machine is up
I0.4	Rub_down	Rubbing machine is down
I0.5	Rot_pos	Rotary table is in position
I0.6	Wor_ok	Cylinder base hole is correct
I0.7	Follow	Light barrier to the following station
I1.0	START	Start button
I1.1	STOP	Stop button
I1.3	RESET	Reset button
Q0.0	Rubmach	Rubbing machine
Q0.1	Rotary	Rotary table
Q0.2	Rubdown	Rubbing machine move down
Q0.3	Rubup	Rubbing machine move up
Q0.4	Clamping	Cylinder base clamping for rubbing
Q0.5	Testing	Testing the hole
Q0.6	Reject	Reject the Cylinder base to the following station
Q0.7	Previous	Light barrier to the previous station

Assembling station

Address	Symbol	Description
M_IN(1)	B1	Optical sensor red/alu=1, black/no work piece = 0
M_IN(2)	B2	Optical sensor for measuring orientation
M_IN(6)	PART_AV	Cylinder base is available
M_IN(8)	1B1	Feeder of springs in back position
M_IN(9)	1B2	Feeder of springs in front position
M_IN(10)	2B1	Feeder of cover in front position
M_IN(11)	2B2	Feeder of cover in back position
M_IN(12)	B3	Spring is available
M_IN(13)	STATION_B1	No cover at gripping position after feeding
M_IN(14)	STATION_B2	Feeder for covers is empty
M_IN(15)	IP_FI	IP following station available
M_IN(3)	START	Start button
M_IN(4)	STOP	Stop button
M_IN(5)	RESET	Reset button
M_OUT(8)	STATION_1Y1	Feeding of a spring
M_OUT(9)	STATION_2Y1	Feeding of a cover
M_OUT(10)	HCLOSE1	Close the gripper
M_OUT(11)	HOPEN1	Open the gripper

Robot station

Symbol	Description
Part_av_RS (INPUT)	Cylinder base available
Pinit (INPUT)	Robot is initial position
P1 (INPUT)	Robot is in “pick up” position
P2 (INPUT)	Robot is in “change gripper” position
P3 (INPUT)	Robot has changed to orientation gripper.
P4 (INPUT)	Robot is in “orientation” position
P5 (INPUT)	Robot is in “assembling” position
P6 (INPUT)	Robot is in “pick up piston” position
P7 (INPUT)	Robot is in “pick up spring” position
P8 (INPUT)	Robot is in “pick up cap” position
P9 (INPUT)	Robot has changed to cap gripper
P10 (INPUT)	Robot is in “orientation cap” position
P11 (INPUT)	Robot is in “next station” position
P12 (INPUT)	Robot is in “previous station” position
Gripper_Robot (INPUT)	The gripper is open=1
turn (OUTPUT)	Robot turns the orientation
Go_to_Pinit (OUTPUT)	Robot move to Pinit
Go_to_P1 (OUTPUT)	Robot move to P1
Go_to_P2 (OUTPUT)	Robot move to P2
Go_to_P3 (OUTPUT)	Robot move to P3.
Go_to_P4 (OUTPUT)	Robot move to P4
Go_to_P5 (OUTPUT)	Robot move to P5
Go_to_P6 (OUTPUT)	Robot move to P6

Go_to_P7 (OUTPUT)	Robot move to P7
Go_to_P8 (OUTPUT)	Robot move to P8
Go_to_P9 (OUTPUT)	Robot move to P9
Go_to_P10 (OUTPUT)	Robot move to P10.
Go_to_P11 (OUTPUT)	Robot move to P11
Go_to_P12 (OUTPUT)	Robot move to P12

Transport system station

For each operation position, there are the next I/Os:

Address	**Symbol**	**Description**
IN1	01_Palav	Pallet at stopper
IN2	01_Cong	Congestion sensor inductive
IN3	01_Wpav	Work piece on pallet sensor optical
IN4	01_Strel	Stopper is released
OUT1	01Stopre	Release stopper

Storing station

Address	Symbol	Description
I0.0	Sl_Adv	Mini slide retracted
I0.1	Sl_Ret	Mini slide extended
I0.2	Sens_1	Color sensor black assembled cylinder
I0.3	Sens_2	Color sensor red assembled cylinder
I0.4	Sens_3	Color sensor silver assembled cylinder
I0.5	La_Ok	Motion complete linear axis
I0.6	Ra_Ok	Motion complete rotary axis
I1.0	START	Start button
I1.1	STOP	Stop button
I1.3	RESET	Reset button
Q0.0	Sld_Ext	Extend mini slide
Q0.1	Close_G	Close gripper
Q0.2	Pos_0	Position controller bit 0
Q0.3	Pos_1	Position controller bit 1
Q0.4	Pos_2	Positon controller bit 2
Q0.5	Start_La	Starting signal linear axis
Q0.6	Start_Ra	Starting signal rotary axis

ASRS20 station

Address	Symbol	Description
I124.0	Zax_up	Z-axis is in up position
I124.1	Zax_down	Z-axis is in down position
I124.2	Xax_left	X-axis is in left position
I124.3	Xax_righ	X-axis is in right position
I124.4	Yax_back	Y-axis with gripper is in back position
I124.5	Yax_frnt	Y-axes with gripper is in front position
I124.6	Gri_clos	Gripper is closed
I124.7	Gri_open	Gripper is open
I125.7	START	Start button
I125.1	STOP	Stop button
I125.3	RESET	Reset button
I125.4	PenOrder	Pending Order
Q124.0	Xleft	X-axes move to the left
Q124.1	Xright	X-axes move to the right
Q124.2	Zup	Z-axes move up
Q124.3	Zdown	Z-axes move down
Q124.4	Yaxes	Solenoid of the Y-axes=0=back position
Q124.5	Gripper	Solenoid of the gripper=0=closed
Q124.6	Xfast	X-axes fast motion
Q124.7	Zfast	Z-axes fast motion
Q125.7	Statread	Station is ready

CNC machine

Address	Symbol
IN34	CNC door opened
IN35	CNC door closed
IN36	CNC clamping device unclamped
IN37	CNC clamping device clamped
IN38	CNC reference mode
IN39	CNC in reference
IN40	CNC Emco ready
OUT32	CNC run program
OUT33	CNC automatic mode
OUT34	CNC reference axis
OUT35	CNC AUX on
OUT36	CNC open door
OUT37	CNC close door
OUT38	CNC open clamping
OUT39	CNC close clamping

Buffer station

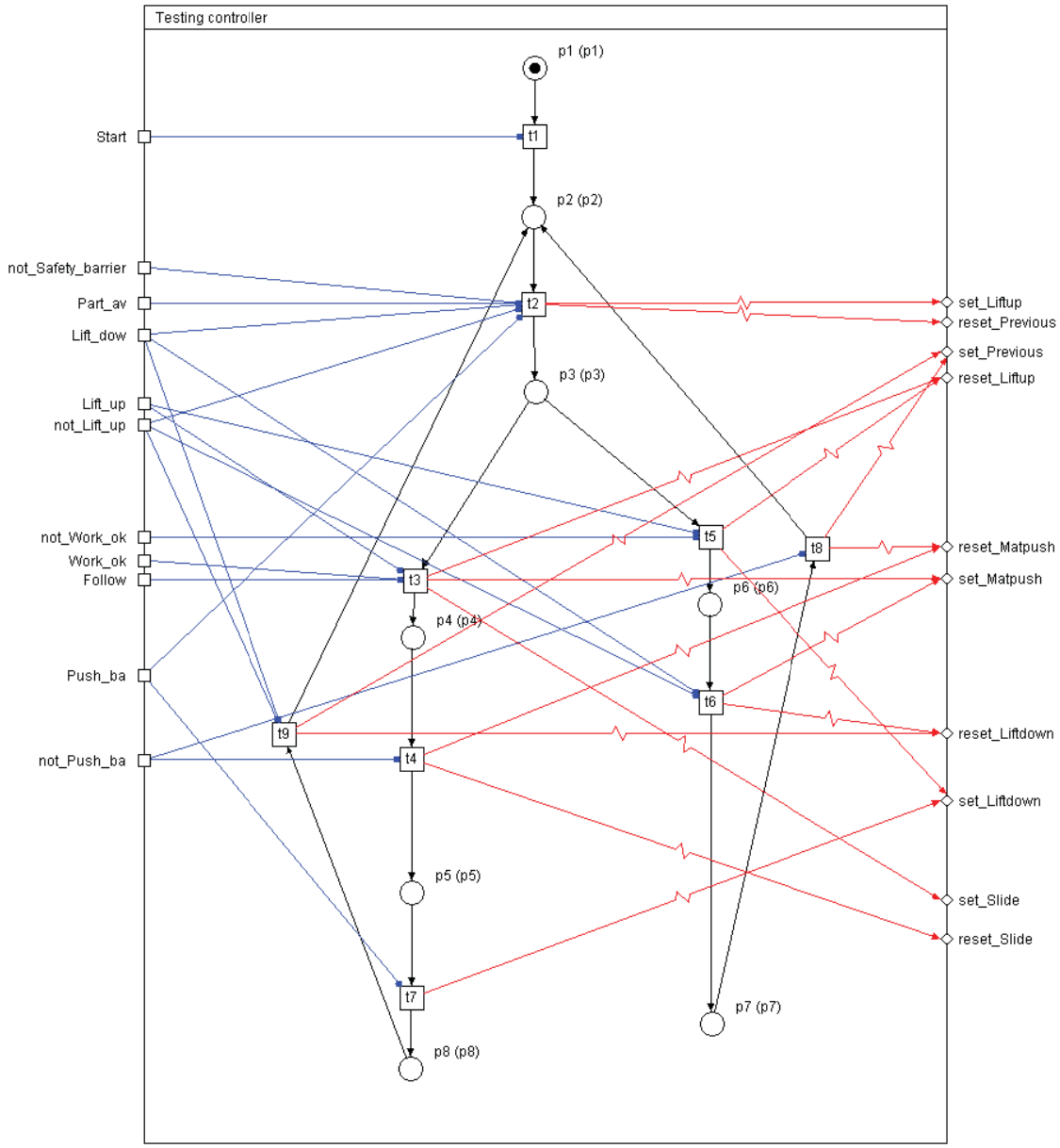
Address	**Symbol**
IN1	START button
IN2	STOP button
IN4	RESET button
IN8	Processed parts conveyor 3 full
IN9	Raw work piece 1 available
IN10	Raw work piece 2 available
IN11	Raw work piece 1 in separator
IN12	Raw work piece 2 in separator
IN13	Separator 1 opened
IN14	Separator 2 opened
OUT8	Conveyor 1 motor on
OUT9	Conveyor 2 motor on
OUT10	Conveyor 3 motor on
OUT11	Open separator 1
OUT12	Open separator 2

Mitsubishi RV-3SDB-S15

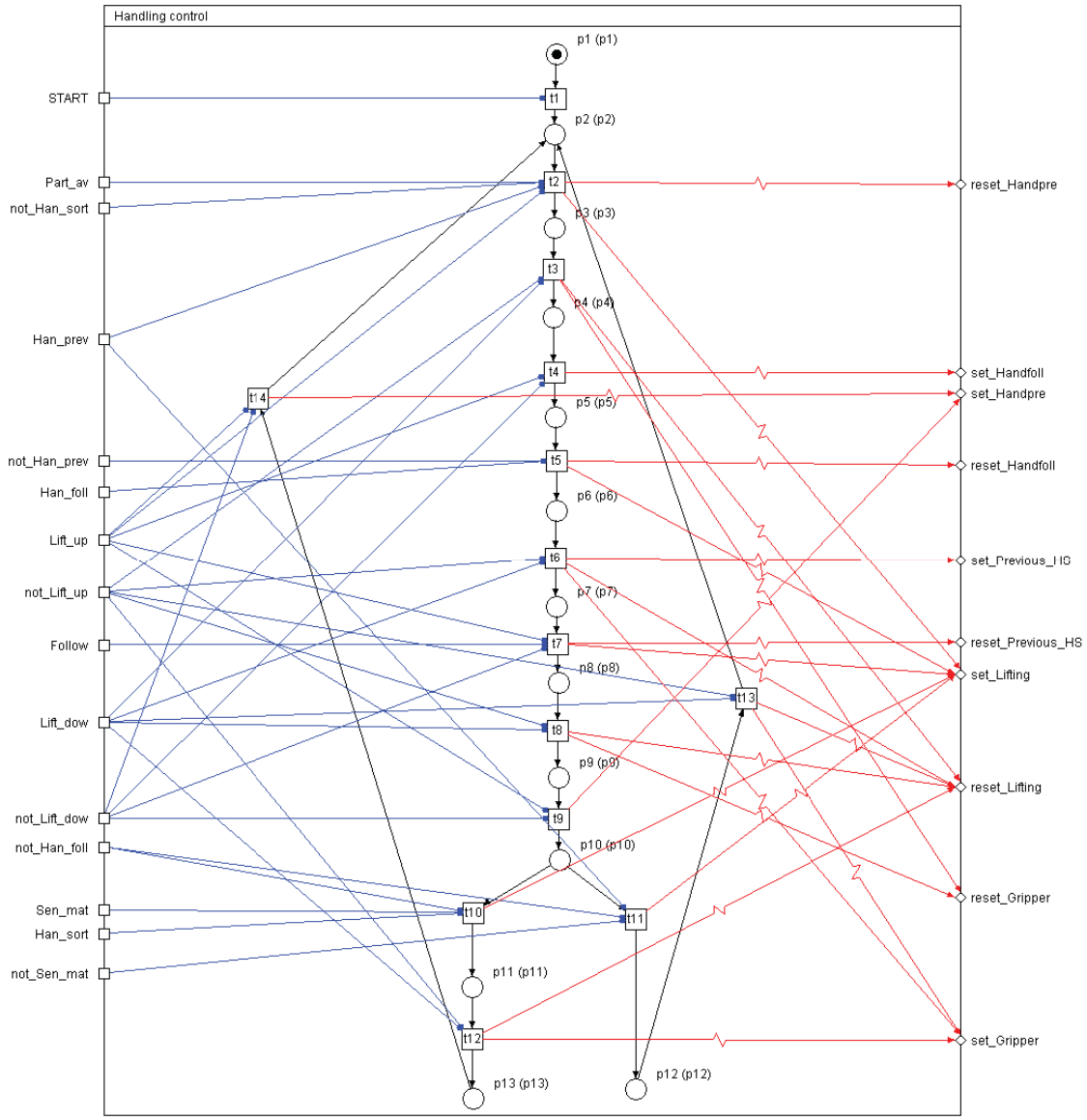
Symbol	Description
Robot_in_P1 (INPUT)	Robot is in Buffer 1 position
Robot_in_P2 (INPUT)	Robot is in Buffer 2 position
Robot_in_P3 (INPUT)	Robot is in Buffer 3 position
Robot_in_P4 (INPUT)	Robot is in CNC position
Robot_in_P5 (INPUT)	Robot is in Distribution station position
Workpiece_taken (INPUT)	The work piece is taken
Robot_takeB1 (INPUT)	Robot take the cylinder base in B1
Robot_takeB2 (INPUT)	Robot take the cylinder base in B2
Robot_Ready (INPUT)	Robot is idle
BC_in_B1 (OUTPUT)	Available cylinder base in B1
BC_in_B2 (OUTPUT)	Available cylinder base in B2
Go_to_P1 (OUTPUT)	Robot move to B1 position
Go_to_P2 (OUTPUT)	Robot move to B2 position
Go_to_P3 (OUTPUT)	Robot move to B3 position
Go_to_P4 (OUTPUT)	Robot move to CNC position
Go_to_P5 (OUTPUT)	Robot move to Distributing station position
Take_workpiece (OUTPUT)	Robot take a work piece
Leave_Workpiece (OUTPUT)	Robot leave a work piece

APPENDIX B: CONTROLLERS

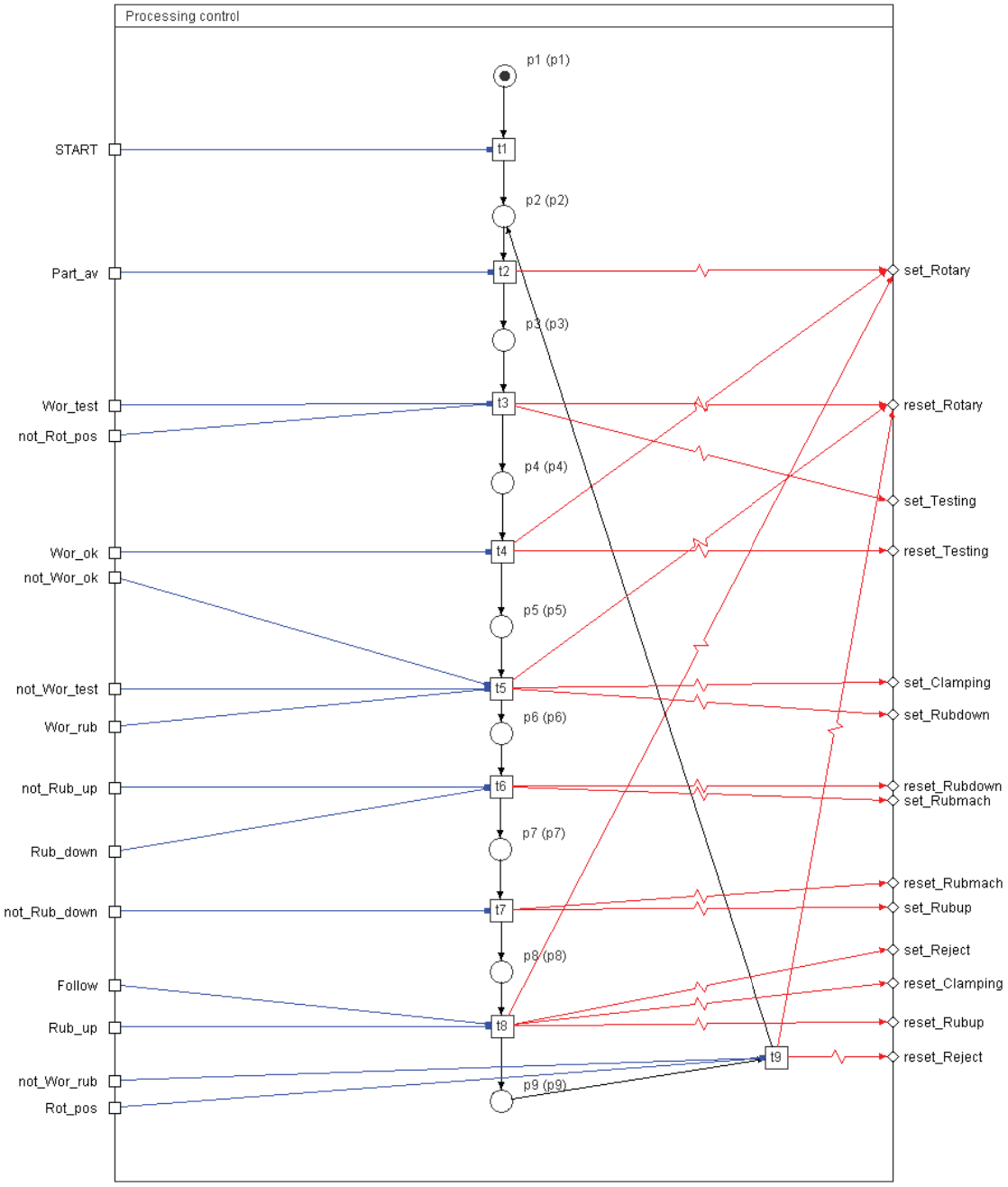
Testing station



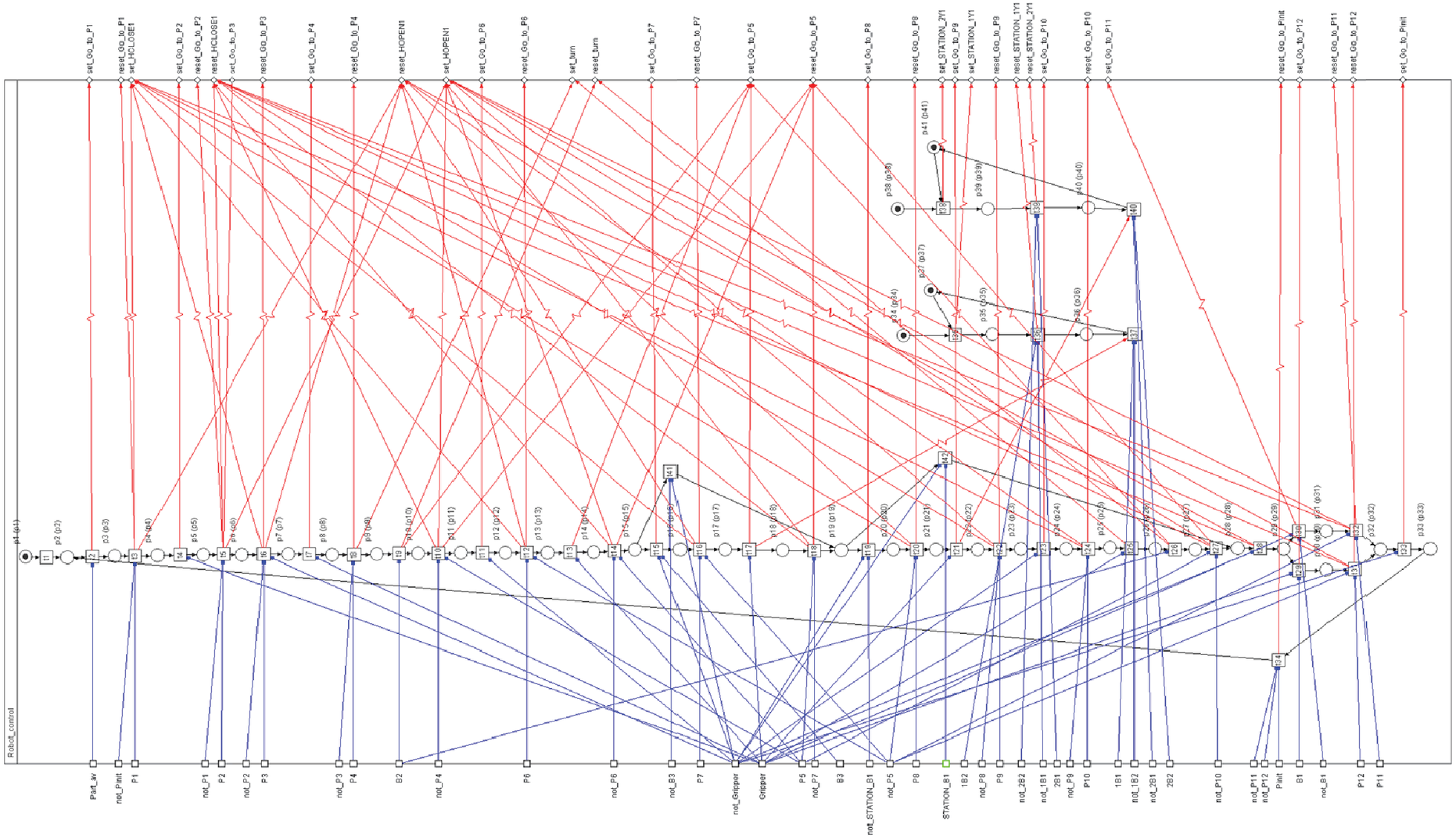
Handling station



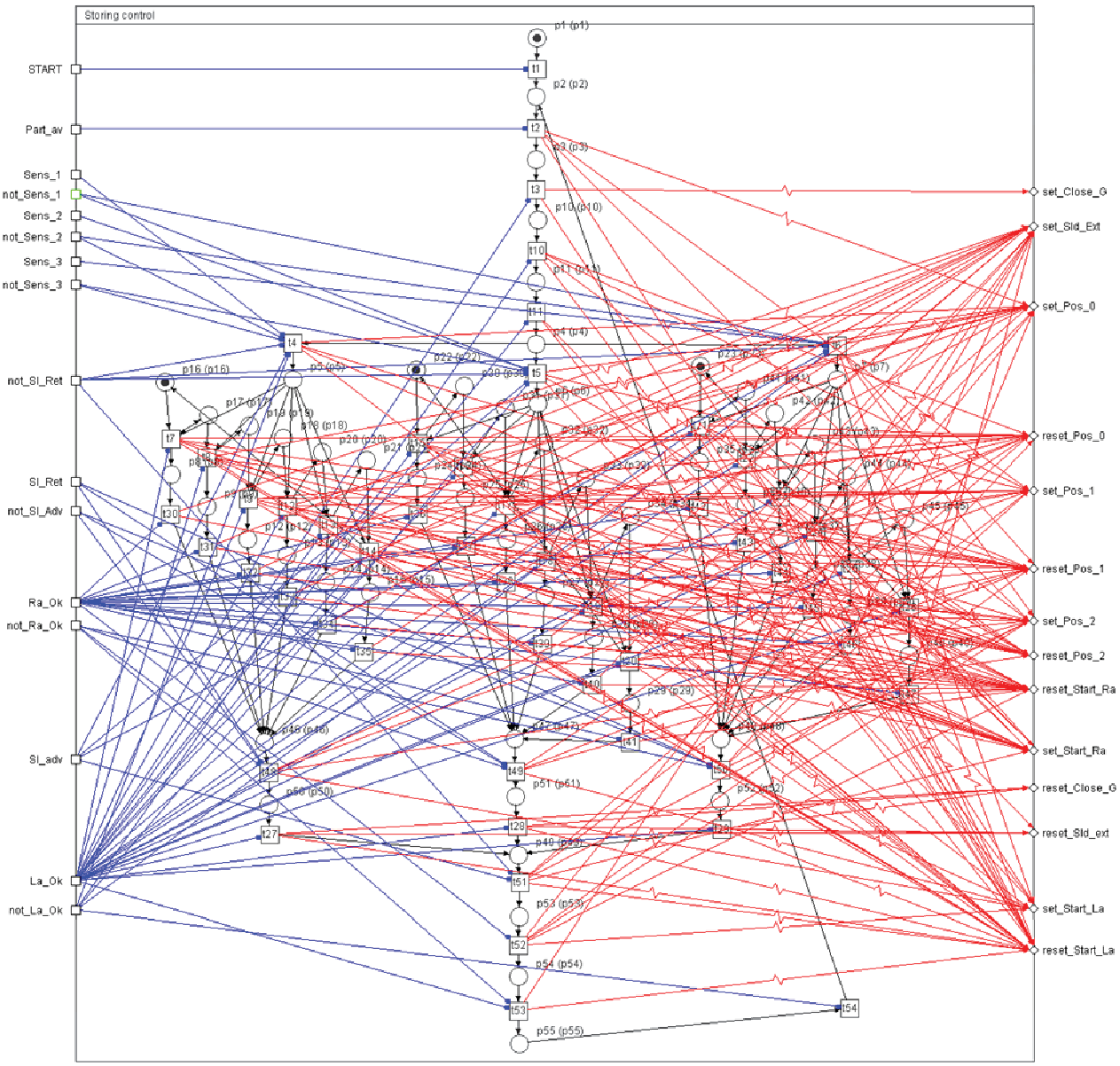
Processing station



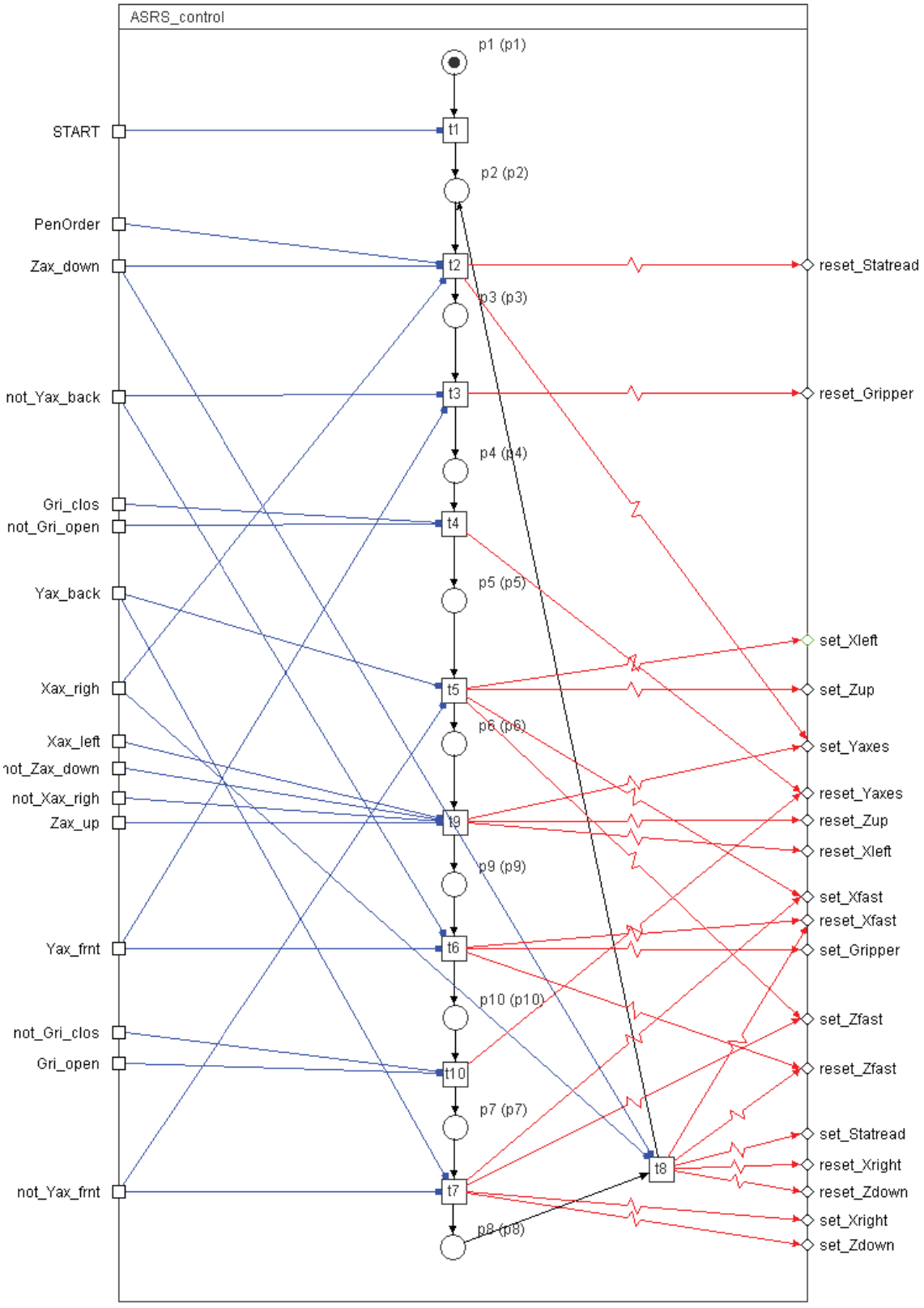
Robot and Assembling station



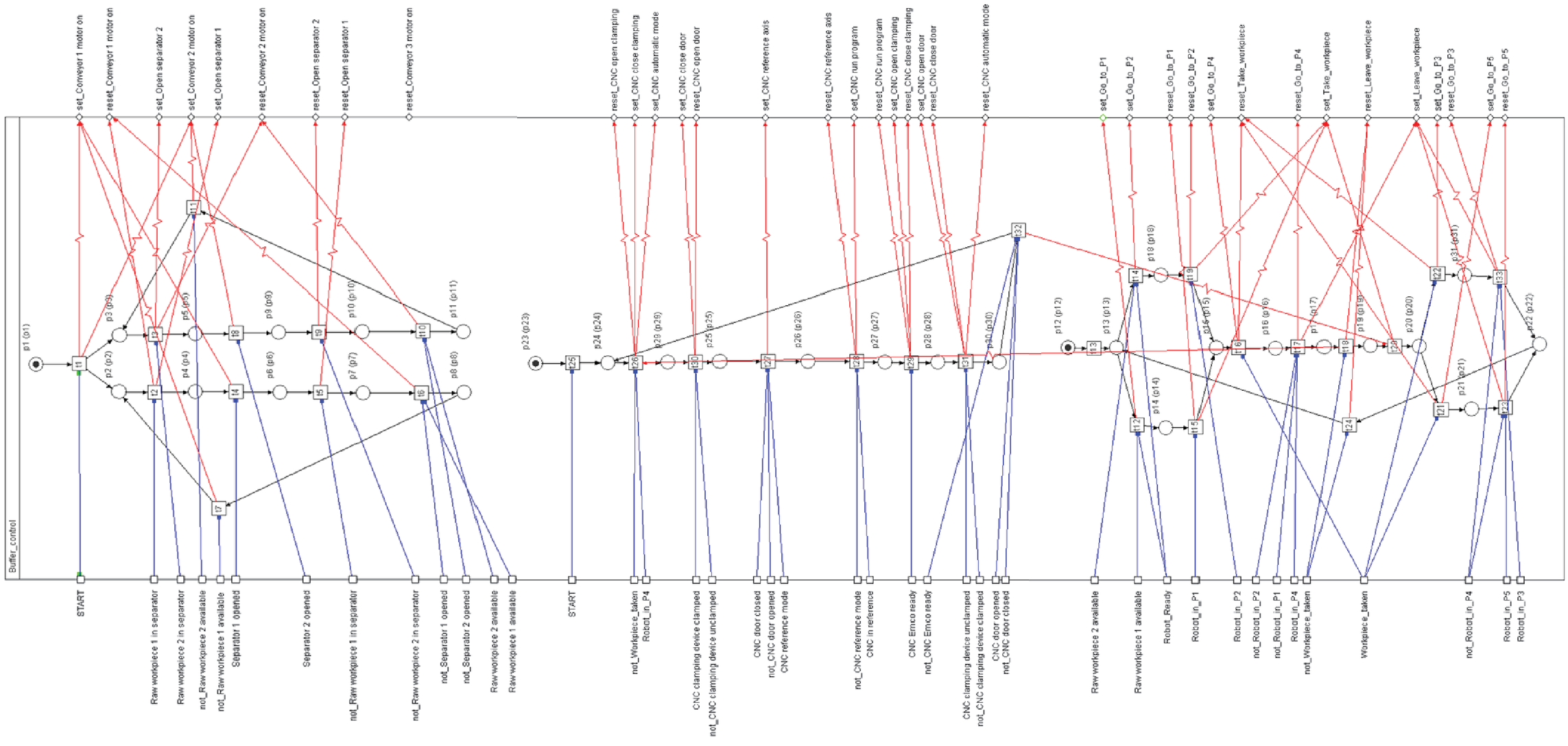
Storing station



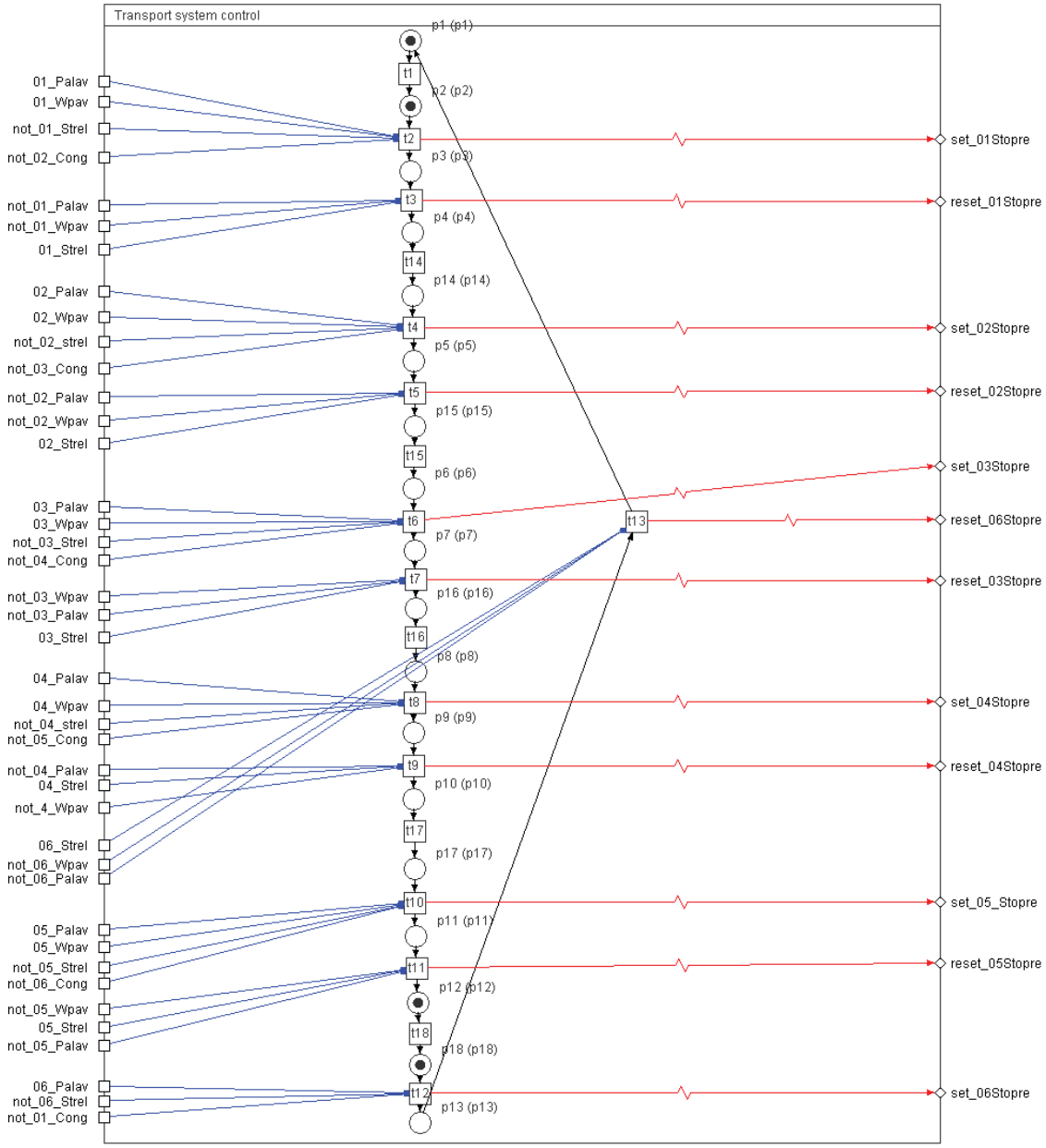
ASRS20 station



Buffer station, Mitsubishi RV-3SDB-S15 and CNC machine

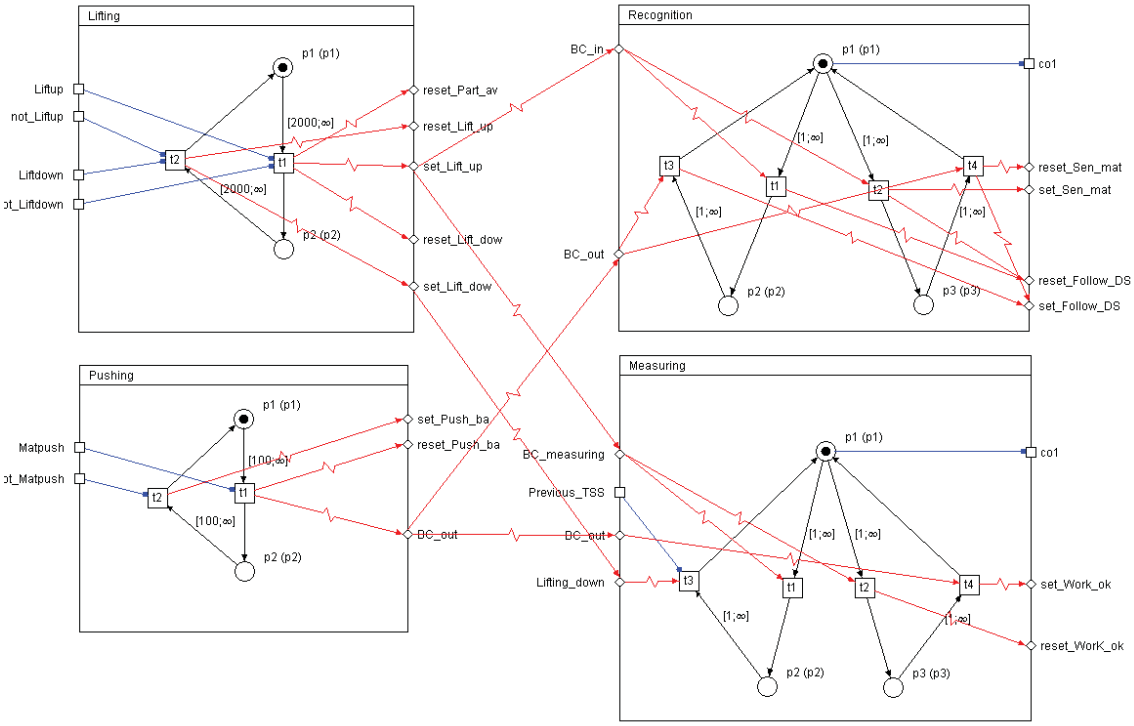


Transport system station

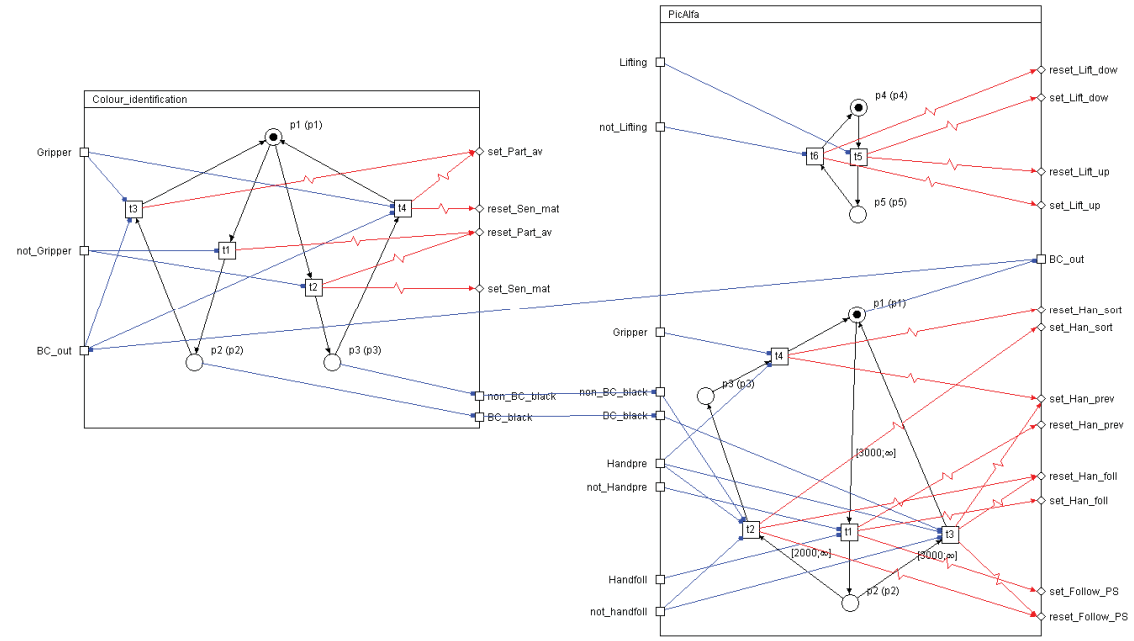


APPENDIX C: PLANTS

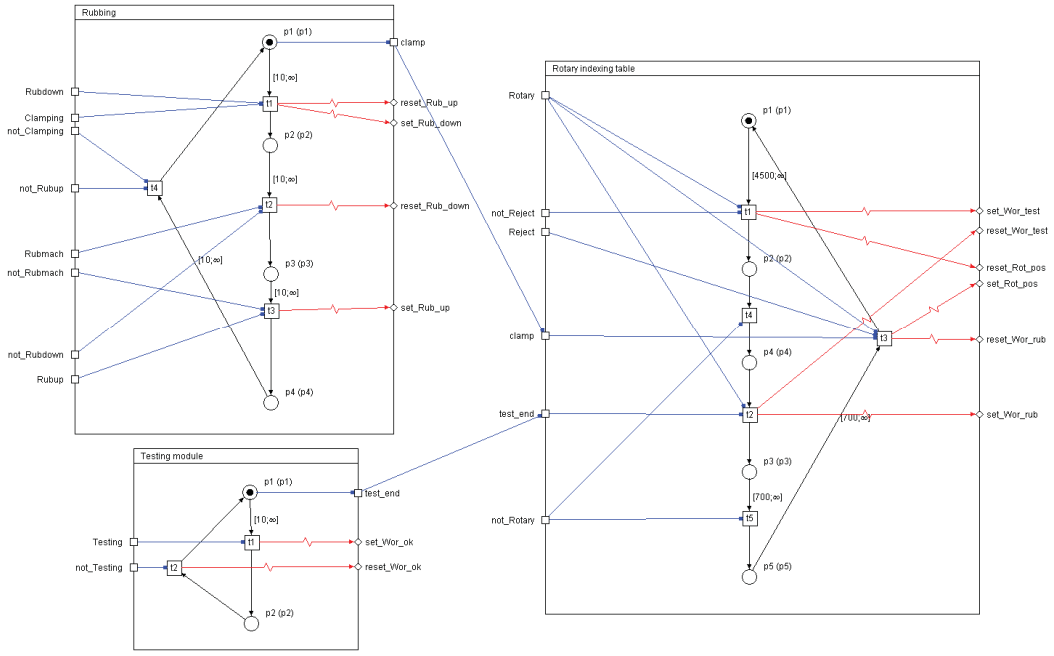
Testing station



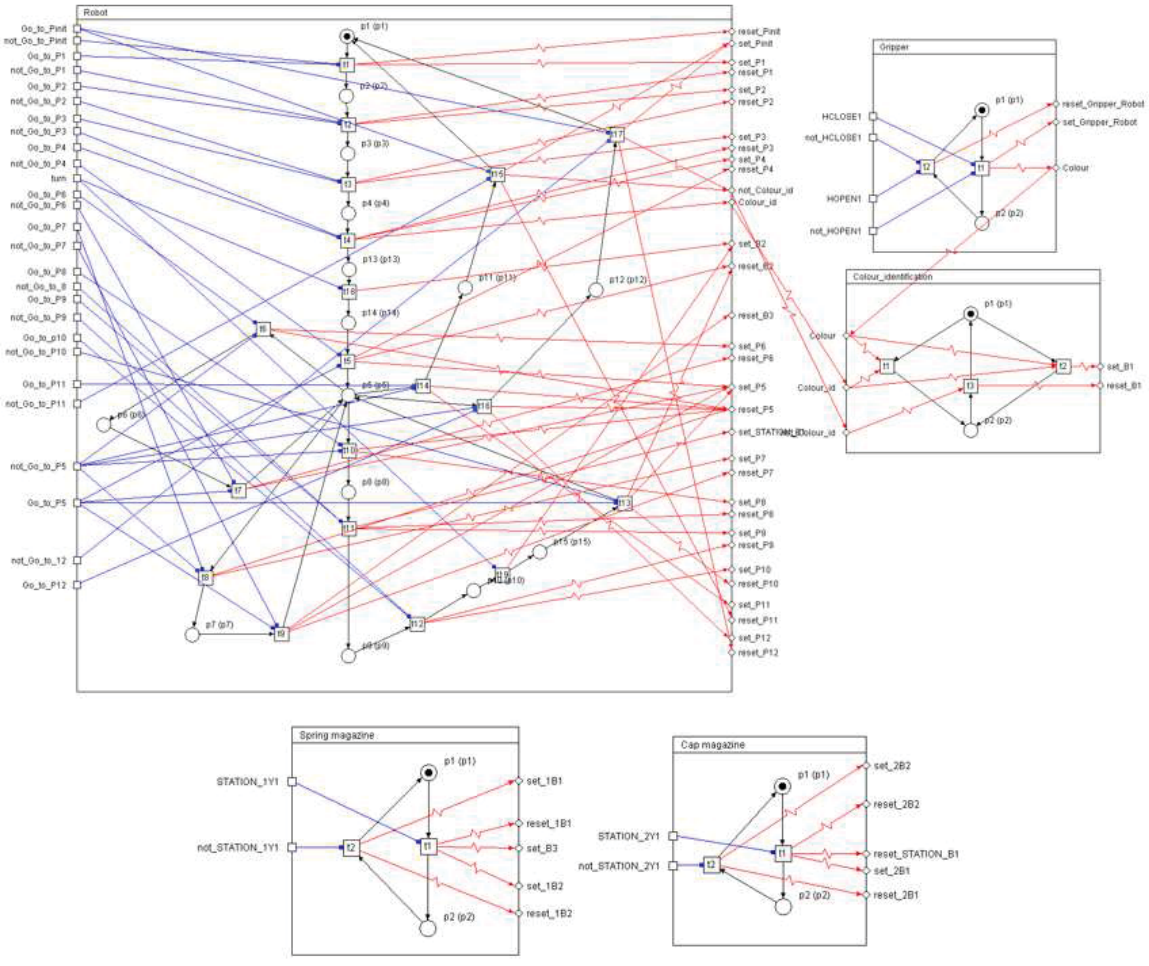
Handling station



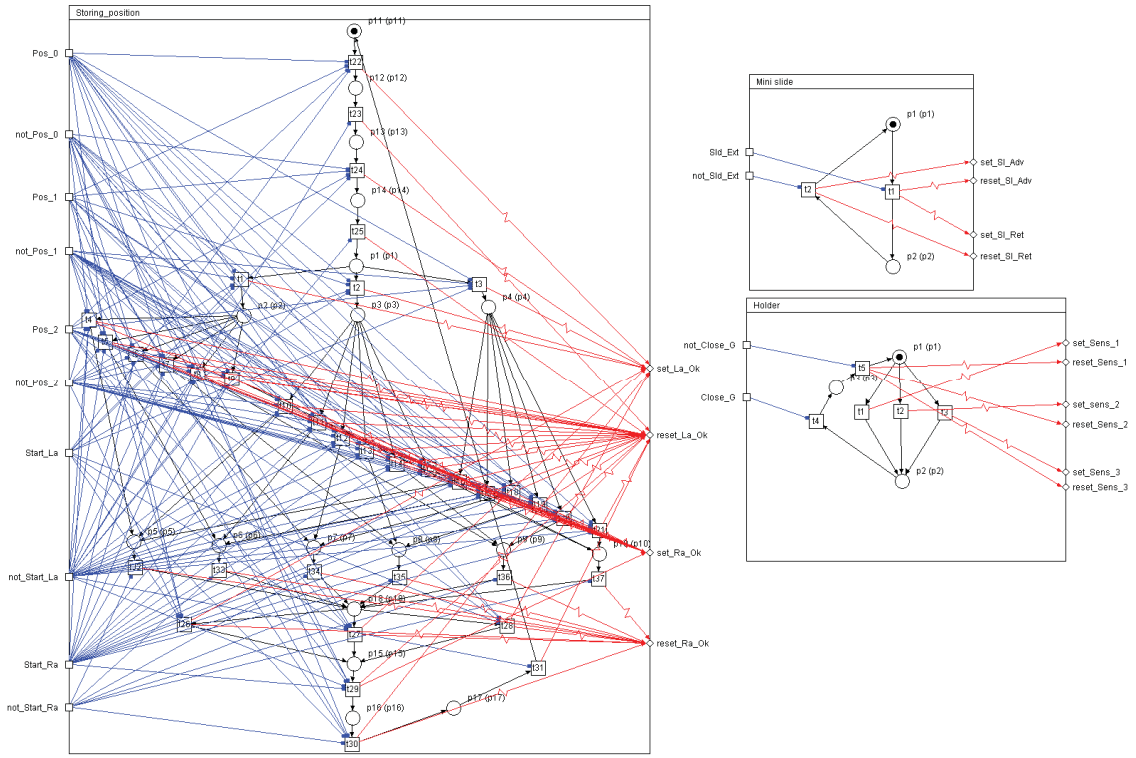
Processing station



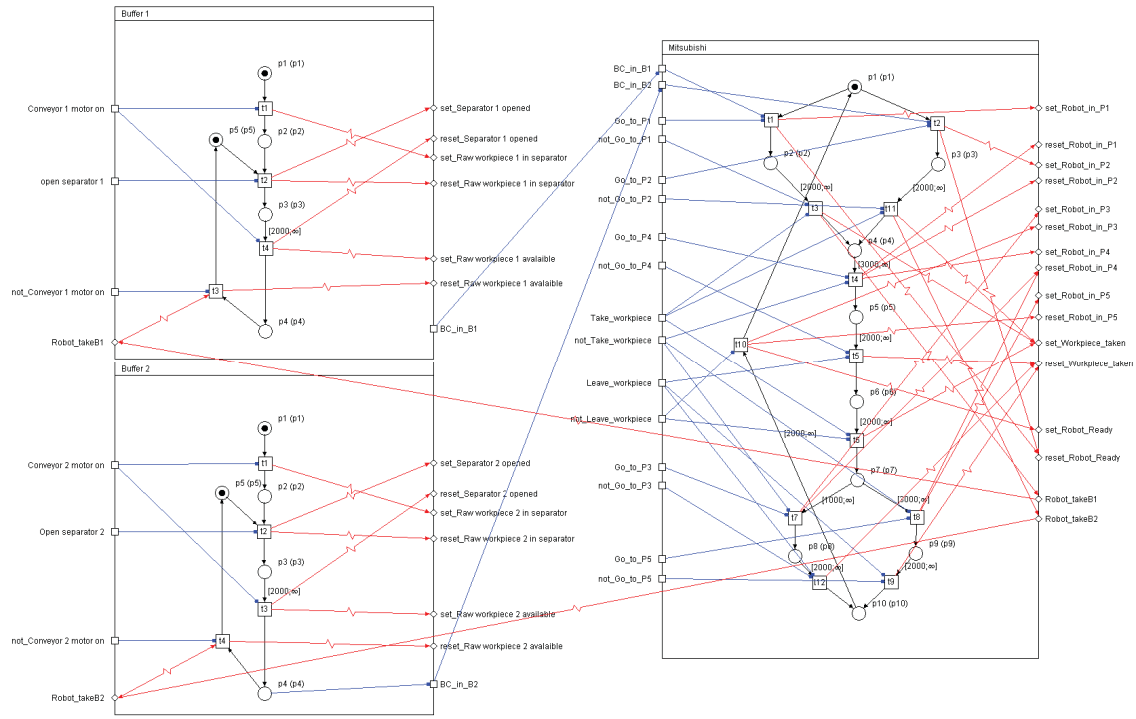
Robot station and Assembling station



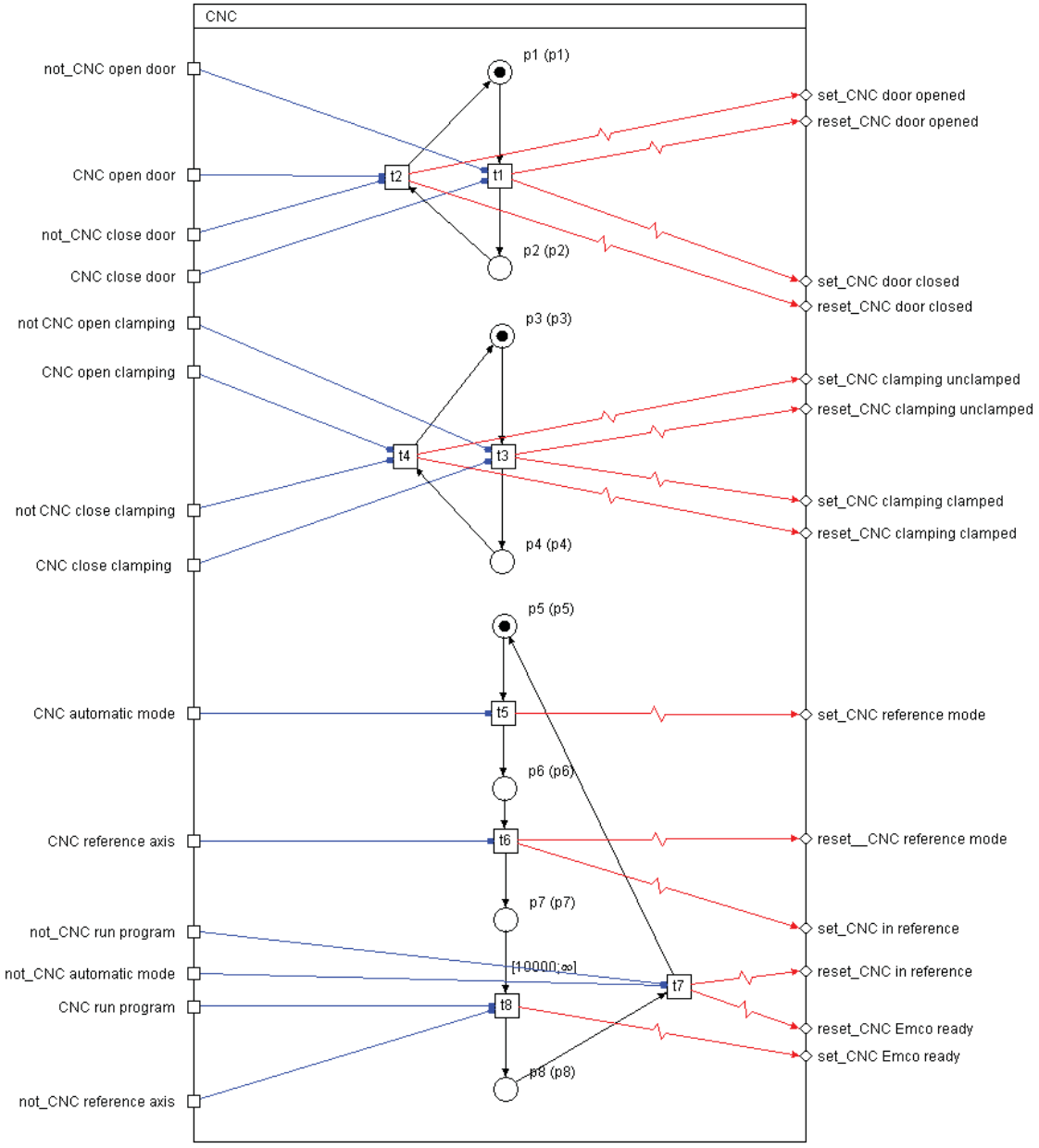
Storing station



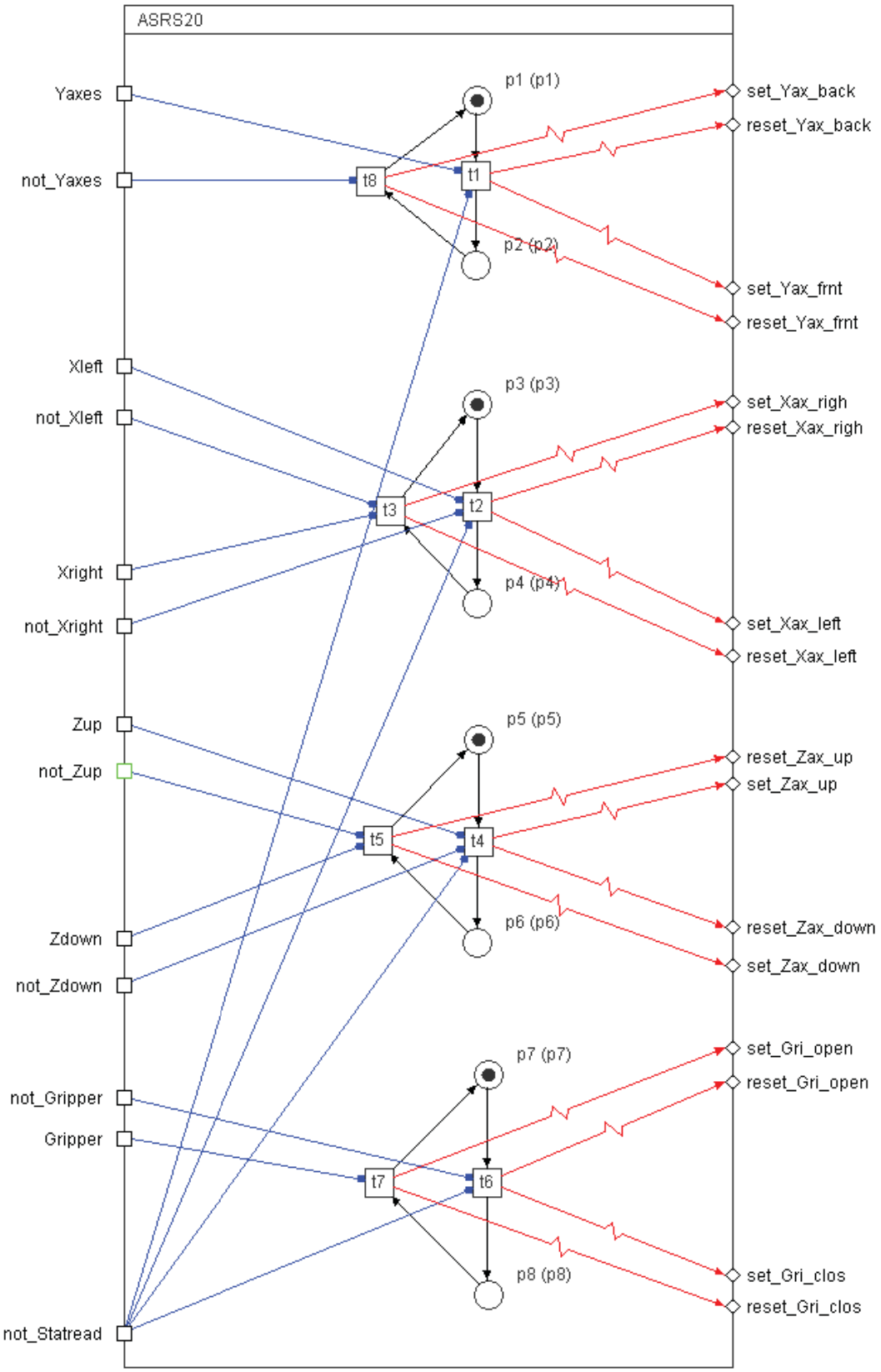
Buffer station and Mitsubishi RV-3SDB-S15



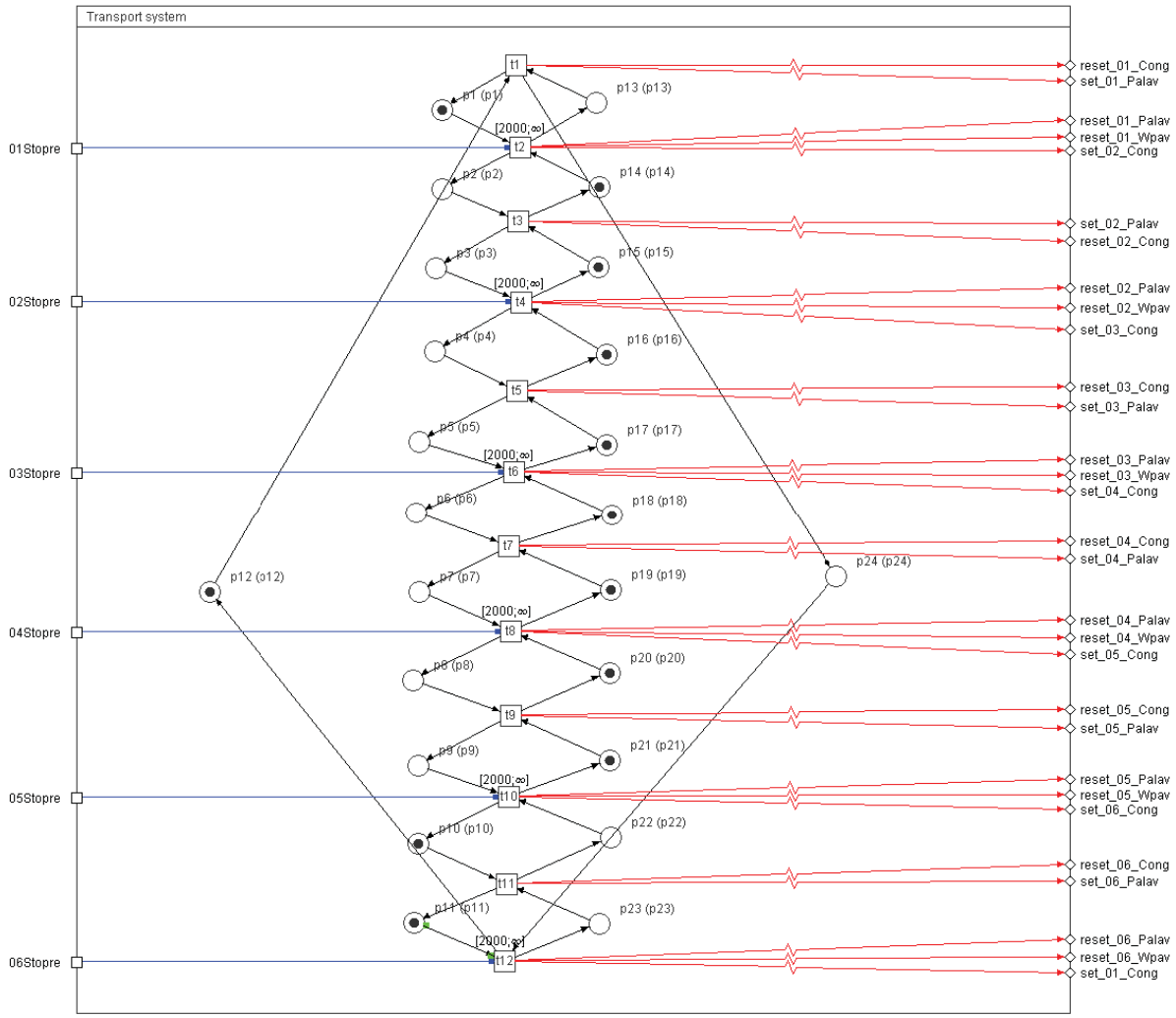
CNC machine


CNC
not_CNC open door
CNC open door
not_CNC close door
CNC close door
not CNC open clamping
CNC open clamping
not CNC close clamping
CNC close clamping
CNC automatic mode
CNC reference axis
not_CNC run program
not_CNC automatic mode
CNC run program
not_CNC reference axis
p1 (p1)
p2 (p2)
p3 (p3)
p4 (p4)
p5 (p5)
p6 (p6)
p7 (p7)
p8 (p8)
t1
t2
t3
t4
t5
t6
t7
t8
[10000;∞]
set_CNC door opened
reset_CNC door opened
set_CNC door closed
reset_CNC door closed
set_CNC clamping unclamped
reset_CNC clamping unclamped
set_CNC clamping clamped
reset_CNC clamping clamped
set_CNC reference mode
reset__CNC reference mode
set_CNC in reference
reset_CNC in reference
reset_CNC Emco ready
set_CNC Emco ready

ASRS20 station

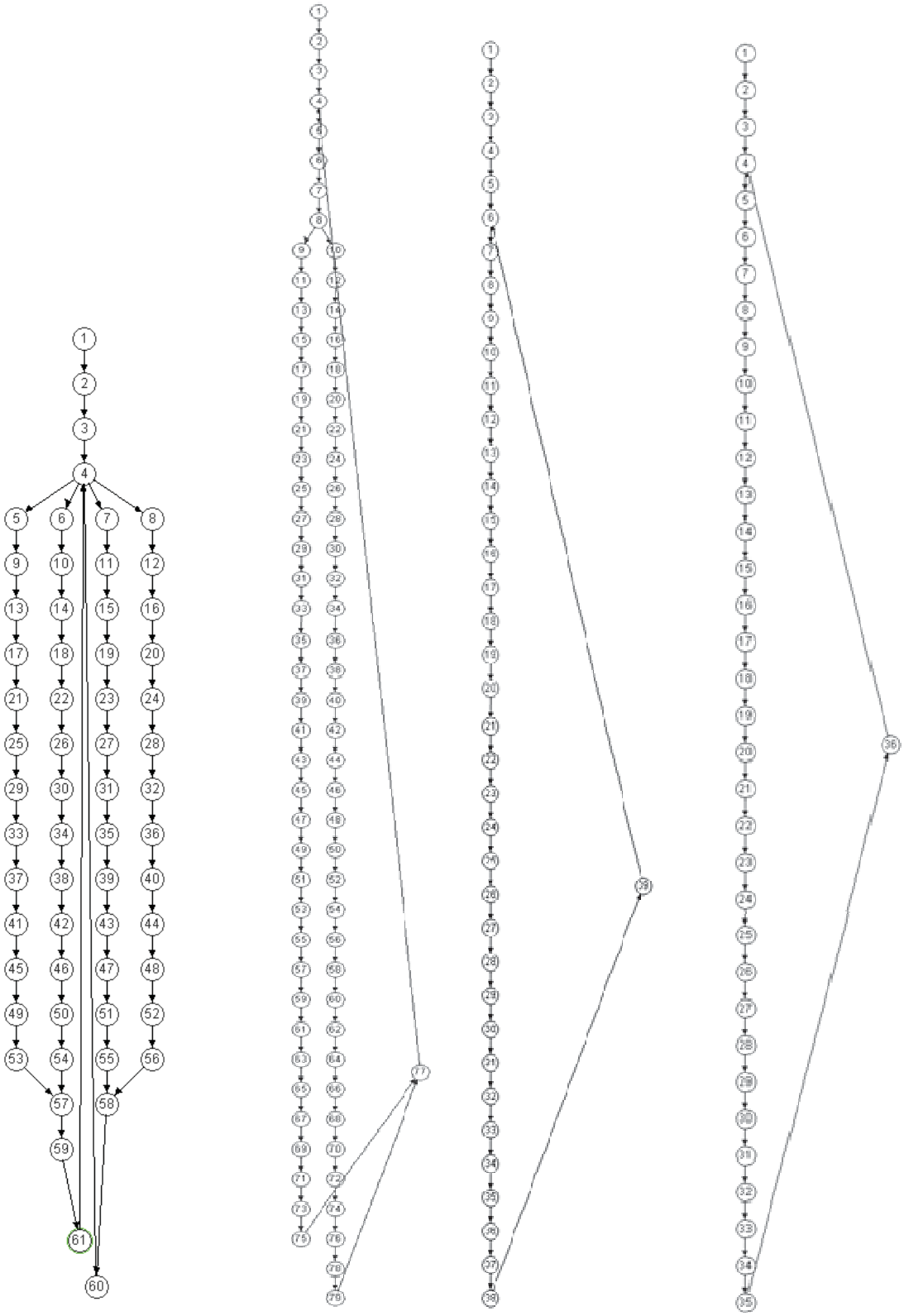


Transport System station



APPENDIX D: STATE SPACES

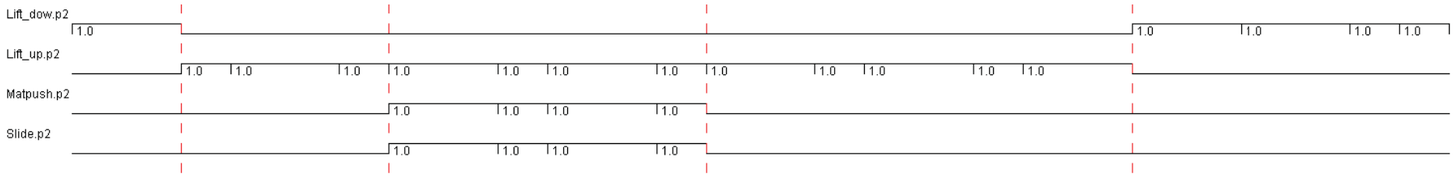
Testing **Handling** **Processing** **ASRS20**



APPENDIX E: STATE SPACES AND TIMING DIAGRAMS

Testing station

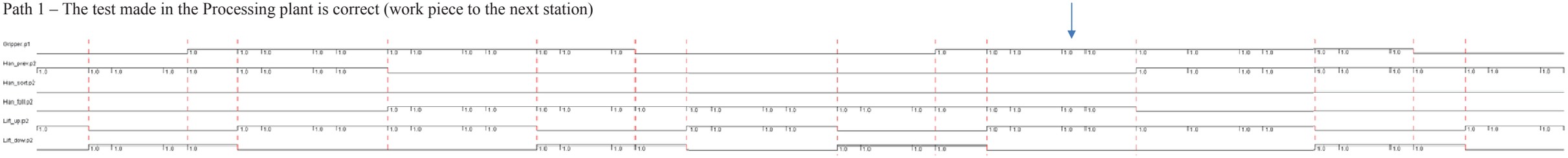
Path 1 – The height of the work piece is correct (work piece to the next station)



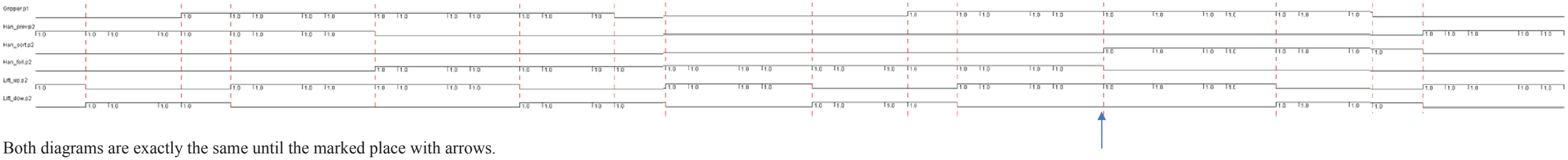
Path 2 – The height of the work piece is not correct (work piece to the sorting slide)

Handling station

Path 1 – The test made in the Processing plant is correct (work piece to the next station)



Path 2 – The test made in the Processing plant is not correct (work piece to the sorting position)



Both diagrams are exactly the same until the marked place with arrows.

Processing station

ASRS20 station