|  |  |  |  |
| --- | --- | --- | --- |
|  | ***Modeling and Automation of Industrial Processes****MEEC/MEGE - 2021 / 2022* |  | *Group: \_\_\_\_**\_\_-\_\_\_\_\_**\_\_-\_\_\_\_\_**\_\_-\_\_\_\_\_**\_\_-\_\_\_\_\_* |

***2nd Lab. – Alarm System as a Discrete Event System [[1]](#footnote-1)***

***Part A - Petri Net***

This laboratory assignment aims at studying Discrete Event Systems (DESs) in the aspects of modeling and analysis of properties. This assignment develops the previous assignment by describing its high-level system as a DES and studying its operation.

Consider an intrusion alarm system which can be working at each time in one of the following conditions:

* OFF
* Ready to detect presences (PD ready)
* Announce a presence (PD sound)
* Alarm ready to detect intrusions (AL ready)
* Alarm sounding because of an intrusion (AL sound)

The currently working condition is changed by events, created with switches SW:

* Presence switch is active (SW0 true)
* Presence switch is inactive (SW0 false)
* Alarm mode switch is active (SW1 true)
* Alarm mode switch is inactive (SW1 false)
* Window switch is active (SW2 true)
* Window switch is inactive (SW2 false)

Schneider's Unity Pro will be used to validate the production of a PLC program from a Petri net. The tools mostly used in this work run in MATLAB / JAVA. Using the login information indicated in the course webpage, see in the course SVN:

- The graphical freeware editor "PIPE2" which allows creating Petri Net models to import with the MATLAB toolbox "TPN5", function rdp (use the version available in the SVN)

- MATLAB functions simulating Petri nets, in particular the *"5 Philosophers"* demo.

**Preliminary experiments:**

**A1:** *[PN to PLC]* See in the course SVN the demonstration of the Petri nets to PLC code converter, named tst3\_blink\_on\_off.m . **(i)** Draw the graph of the Petri net considered in the demonstration. **(ii)** Run the demo on Unity Pro (simulator) by inserting the structured text output of tst3\_blink\_on\_off.m into a Unity Pro project (see Annex A1). Note that some testing can be done with myterminal5. **(iii)** Show a data log of the inputs and/or outputs.

**A2:** *[PN simulation]* See in the course SVN the *"5 philosophers' dinner"* simulation named pdinner\_tst.m (see also Annex A2). **(i)** Show that in the simulation one philosopher has a request to eat *not served*. (ii) Propose one modification to pdinner\_IO.m so that another philosopher gets a request to eat *not served*.

**DES modeling:**

Using as a base guideline the work done in the first laboratory assignment, it is now desired to model the high-level system as a DES. More in detail, is required to develop a Petri Net that describes the events and the state evolution of the intrusion alarm system. In addition to the formal definition of the Petri Net, it is desired also to obtain the corresponding incidence matrix.

Note: Please consider making a **Petri Net as simple as possible**, so that parts A and B can be handled by handwriting, i.e. not needing computer-based studying of the DES properties.

**A3:** Write the table of pre and post conditions for each of the events. *(Fill the next table, adding as many lines as needed).*

|  |  |  |
| --- | --- | --- |
| **Event** | **Pre-Conditions** | **Post-Conditions** |
|  |  |  |

**A4:** *[PN graphical display]* Draw the graph of your Petri Net. Note: use the Petri net editor available (PIPE2) to create a Petri net and save it as an XML file. Call PIPE2 from MATLAB with the command pn\_editor use PIPE2 file open user interface to work on the right file. Include in the report a PNG image, exported by PIPE2, showing the graph of the Petri net.

**A5:** *[PN incidence matrix]* Obtain the incidence matrix, D=D+-D-, describing your Petri Net.

**A6:** *[PN simulation]* Simulate the Petri net you proposed in the previous questions. Comment whether your simulation software implies, or does not imply, privileged places or transitions while receiving random inputs. As helping references see question A2.

**A7:** *[PN properties]* Discuss the properties that the proposed Petri net should verify and order the importance of those properties in the table below.

|  |  |
| --- | --- |
| **Order of importance** | **Property** |
| 1 |  |
| 2 |  |
| 3 |  |
| 4 |  |
| 5 |  |
| 6 |  |
| 7 |  |

**A8:** Study property #1, resorting to the methods studied in the course.

**A9:** Study property #2, resorting to the methods studied in the course.

**A10:** Study property #3, resorting to the methods studied in the course.

**A11:** Study property #4, resorting to the methods studied in the course.

**A12:** Study property #5, resorting to the methods studied in the course.

**A13:** Study property #6, resorting to the methods studied in the course.

**A14:** Study property #7, resorting to the methods studied in the course.

# *Annex A1 - Convert a Petri net to a PLC program*

In this annex is detailed the testing of a converter of a Petri net to a PLC program. The converter is written and runs in MATLAB. The converter and the demonstration example are included in the SVN of the course.

Please find in your local installation the folder ./lab2\_intro/ and run tst3\_blink\_on\_off.m . Note that the compiler is in the course SVN, folder ../pn\_to\_plc\_compiler/ which is accessible when you do login\_api.

The command plc\_z\_code\_helper('config', ...) included in tst3\_blink\_on\_off.m, already defines the PLC modules / memory configuration. In case you remove that command, the configuration needs to be defined once per MATLAB session. Three configurations are possible (see figure A1.1):

- s2\_DEY16D2\_s4\_DSY16T2 means the PLC has two modules, at slots two and four, and the modules are named DEY16D2 and DSY16T2.

- s3\_DMY28FK means the PLC has only the module DMY28FK mounted at slot three.

- m0-9\_m10-19 means the input/output is based on memories %m0 till %m19.



Figure A1.1: Selection of the hardware configuration, input/output modules or memory interface.

After running the demonstration, you obtain file tst3\_mk\_program\_res.txt containing structured text code that you can copy into a Unity project. Within the Unity project you need to declare some additional variables like the timer names, timing values and flags. Please see more details in the text file \_readme.txt .

To have more information on the Petri net to PLC code converter see:

 <http://users.isr.ist.utl.pt/~jag/course_utils/pn_to_plc/pn_to_plc.html>

# *Annex A2 - Philosophers Dinner Petri Net Simulation*

Please find in the SVN of the course the "5 philosophers dinner" simulation folder. Locate and run in MATLAB the file pdinner\_tst.m. Note that you need also the rdp.m function found in the tpn5 toolbox which is also in the SVN - see the SVN root file \_readme.txt for details.

The pdinner\_tst.m demonstration script loads a Petri net model shown in figure A2.1 and built using PMEDIT. The Petri net model is read using the referred rdp.m function.



Figure A2.1: Petri net model representing the 5 philosophers dinner problem.

The default simulation has the transitions scheduled, driven by a timetable. To run a simulation where the transitions are driven randomly, you can edit the file PN\_tfire.m and replace the line:

 qk= pdinner\_IO(act, t);

by the line:

 qk= round(rand(1,5)); qk= [qk not(qk)];

Note that only transitions 1 till 5 are computed randomly. Transitions 6 till 10 are defined simply as the negation of the transitions 1 till 5.

To have more information on the Petri net simulator and the specific simulation of the "5 philosophers dinner" see:

 <http://users.isr.ist.utl.pt/~jag/course_utils/pn_sim/PN_sim.html>

#

1. Revised by Prof. José Gaspar (2022). [↑](#footnote-ref-1)