|  |  |  |  |
| --- | --- | --- | --- |
| IST_logo | ***Industrial Processes Automation***  *MSc in Electrical and Computer Engineering*  *Scientific Area of Systems, Decision, and Control*  *Winter Semester 2016/2017* |  | *Group: A2*  *75167- Francisco Oliveira*  *75322-Catarina Gaspar*  *75637-Inês Lourenço*  *84993-Hugo Serieiro* |

***2nd Laboratory Assignment [[1]](#footnote-1)***

***Handling Faults in Keyboard Reading***

This laboratory assignment aims at studying Discrete Event Systems (DESs) in the aspects of modeling, analysis of properties and synthesis. Synthesis will be based on a recent methodology in the framework of supervised control. This assignment further develops the previous assignment in the keyboard reading component by introducing fault handling mechanisms.

The tools to be used in this work are MATLAB and a Petri Net editor. In the last part of the work the Schneider PLCs will be used once more to validate the proposed methodologies.

# Part B - Properties of the PN

**DES modeling:**

In the part A of this second laboratory assignment a Petri net was designed to model the process of reading one keyboard key as a DES. The aim of the current partial assignment, part B, is to analyze that Petri net, resorting to the properties studied in this course. Moreover, its simulation resorting to an available software package is required.

Note: see in the course webpage tools helping this part of the assignment, namely the graphical editor "PMEDIT" which allows creating models to import with the Matlab toolbox "tpn5". Both tools are compressed in a single ZIP file "PN\_editor\_MATLAB\_sim\_and\_Manual.zip".

**Q1:** Draw the Petri net proposed in Part A using the Petri net editor available (pmedit.exe).

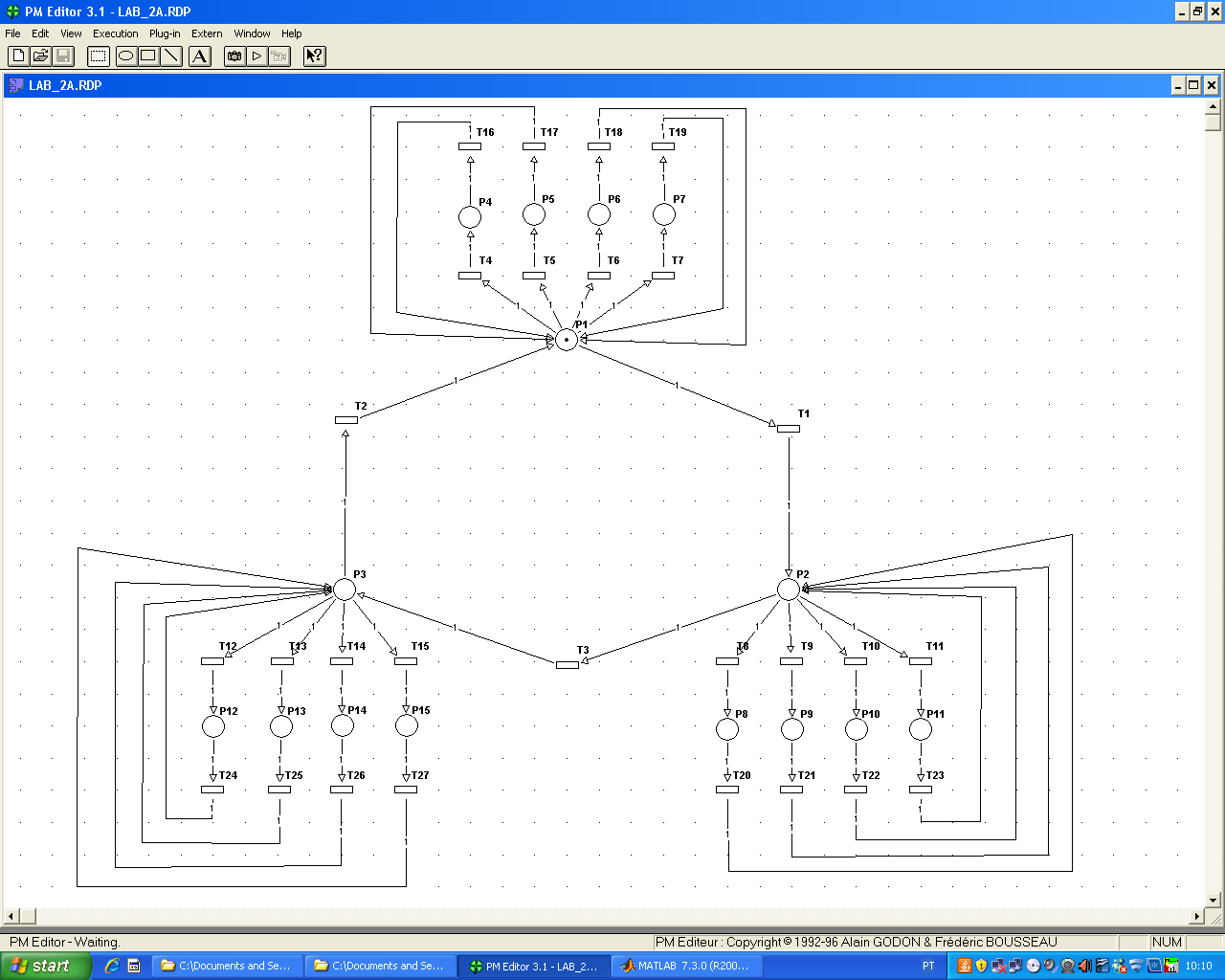


Figura 1 - Petri Net proposta

**Q2:** Simulate a sequence of events such that there is a sequence of different states and the final state coincides with the initial one. Provide an interpretation of the sequence of events given the original assignment.

A sequência de eventos escolhida foi:

tu= [...

0 mk\_keys(1) ; ...

1 mk\_keys([]) ; ...

2 mk\_keys(5) ; ...

3 mk\_keys(5) ; ...

4 mk\_keys([]) ; ...

5 mk\_keys([9 8]) ; ...

6 mk\_keys(8) ; ...

7 mk\_keys([]) ; ...

8 mk\_keys(7) ; ...

9 mk\_keys(10) ; ...

10 mk\_keys([]) ; ...

11 mk\_keys(1) ; ...

];

Com esta sequência podemos testar o funcionamento de sequências de inputs de diversas características, desde inputs em 2 colunas sequenciais, como na mesma linha, até um input prolongado com a duração de 2 instantes de tempo.

Ao mesmo tempo, pode-se comprovar como o programa ignora a sobreposição de múltiplas teclas pressionadas ao mesmo tempo.

O comportamento demonstrado nesta simulação representa a propriedade de invariância temporal da nossa Petri net, ou seja, a capacidade de, com uma determinada sequência de eventos, partir de um certo estado inicial, e no final do programa chegar a esse mesmo estado.

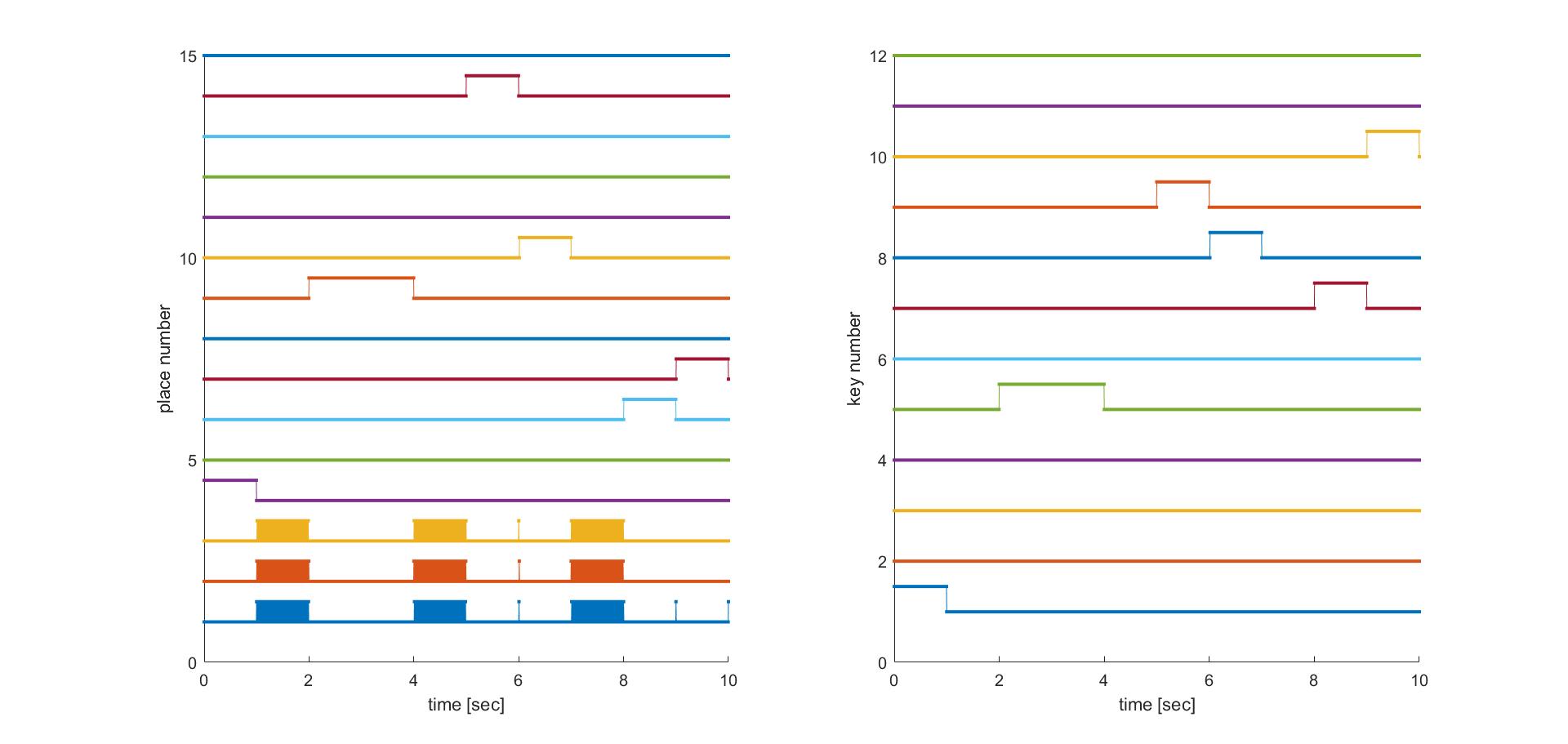


Figura 2 - Representação do funcionamento da Petri net

No exemplo dado, o *token* desta Petri net começa no estado P4, correspondente à tecla “1”, e após a execução da sequência de ações, o token chega ao mesmo estado de onde inicialmente tinha partido.

**Q3:** 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 | Temporal Invariance |
| 2 | Conservation |
| 3 | Reachability |
| 4 | 1-Bounded |
| 5 | Safeness |
| 6 | Liveness |
| 7 | Coverability |

**Q4:** Study property #1, resorting to the methods studied in the course.

**Temporal Invariance**

A partir de um estado inicial há pelo menos uma sequência de acções que permite que o token volte ao estado inicial. Isto é fundamental em qualquer Petri net como a nossa, em que o programa deve ser cíclico tendo sempre o mesmo comportamento consoante a acção que é feita, pelo que foi a propriedade considerada mais importante.

Matematicamente isto pode ser visto vendo que transições ‘q’ levam a que o Petri net volte ao estado inicial, com . Para isto foi utilizada a função null(D, ‘r’) no matlab onde se obteve 13 soluções possíveis, ou seja há 13 sequências de transições que, dado um estado incial com um token no P1, fazem voltar ao estado com um token no P1. Uma das soluções apresentadas é a seguinte,

Em que é representada simplesmente a transição da coluna 1 para a 2, a coluna 2 para a 3 e finalmente a coluna 3 para a 1.

**Q5:** Study property #2, resorting to the methods studied in the course.

**Conservation**

Uma Petri net é strictly conservative se na soma de todos os lugares em todos os estados o número de tokens for constante. No caso do nosso programa isto é verdade, pois qualquer que seja o estado há sempre só um token a transitar. Isto pode ser provado matematicamente através da seguinte propriedade: . Isto foi feito no matlab utilizando a função null(DT, ‘r’), onde se obteve o seguinte resultado,

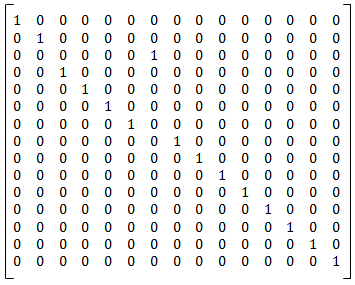
A condição para que fosse conservative seria todas as entradas serem maiores que 0. O resultado obtido mostra que para além de ser conservative, como todas as entradas são 1 é também strictly conservative.

**Q6:** Study property #3, resorting to the methods studied in the course.

**Reachability**

Esta propriedade define se é possível garantir que, a partir de um certo lugar, se consegue aceder a qualquer outro existente na Petri net. Na Petri net proposta, de modo a garantir o correcto funcionamento do keyboard, é fundamental garantir que todos os lugares são acessíveis a partir de outro, não havendo nenhum dead lock. Esta Petri net tem um set de estados acessíveis finito (número de diferentes possíveis estados), uma vez que o número de tokens se mantém constante e o número de lugares é finito. O set tem então exactamente 15 estados acessíveis, que é o número de lugares existentes.

Através das funções “graphnet” e “disp\_gr”, obteve-se a matriz RM indicada seguidamente, que é a matriz dos estados acessíveis, e criou-se a respectiva interpretação gráfica, que comprova que, como foi explicado, todos os estados são acessíveis.



RM =

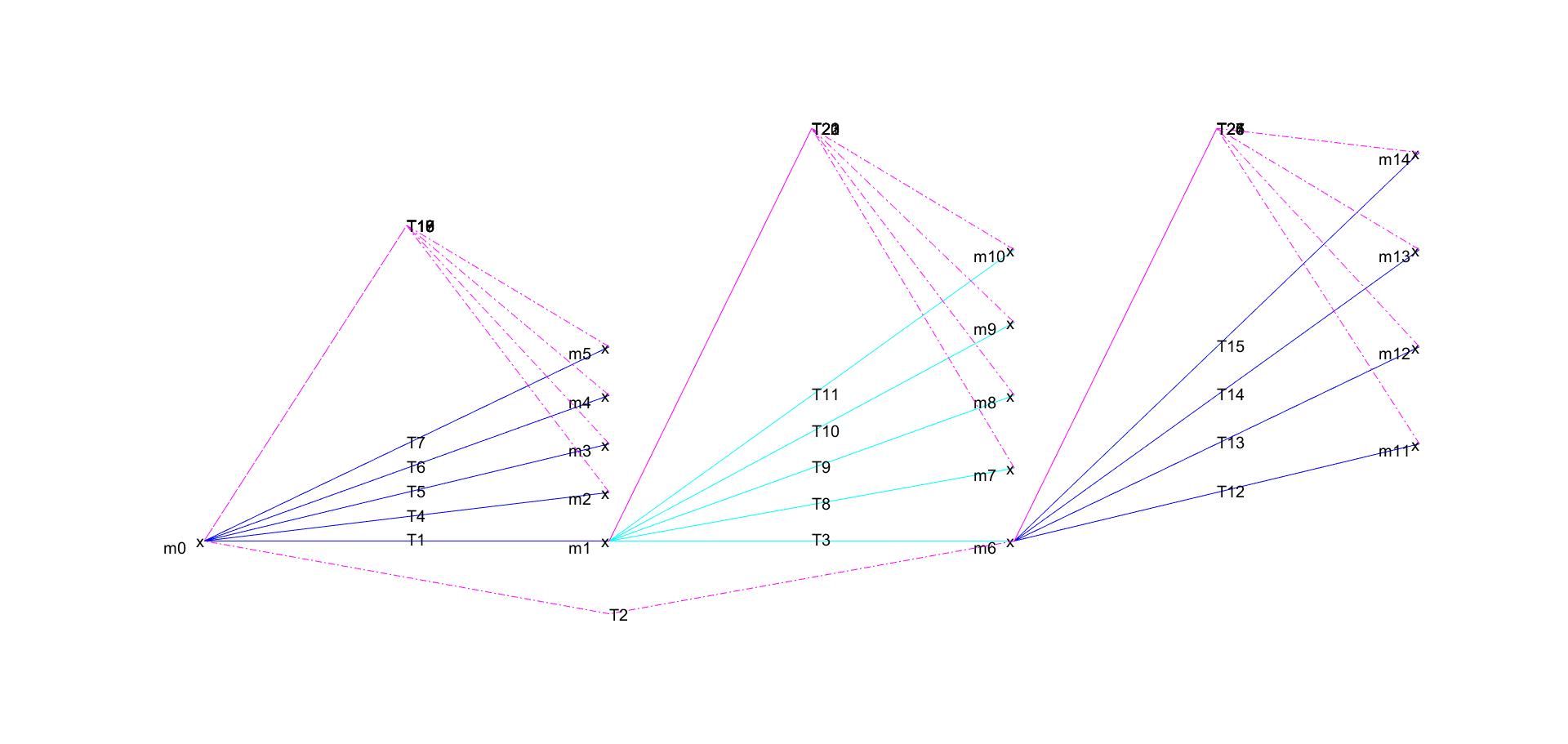


Figura 3 - Representação dos estados acessíveis

**Q7:** Study property #4, resorting to the methods studied in the course.

**-Bounded**

Uma Petri Net diz-se k-bounded quando cada lugar existente tem, no máximo, k tokens em todos os estados possíveis. De forma a garantir que uma Petri net seja bounded, é preciso que não haja em sítio nenhum a geração de token para garantir que o número tokens não aumente infinitamente. No caso da Petri net proposta neste laboratório não há geração de tokens, e não há nenhum peso associado a qualquer transição. Dito isto, pode-se concluir que é 1-bounded pois o estado inicial tem somente 1 token.

**Q8:** Study property #5, resorting to the methods studied in the course.

**Safeness**

Esta propriedade é garantida sempre que a Petri net é 1-bounded uma vez que estas são equivalentes. Ou seja, em todos os estados há sempre no máximo 1 token em cada lugar.

**Q9:** Study property #6, resorting to the methods studied in the course.

**Liveness**

A Liveness de uma transição tem como objectivo classificar o número de vezes que uma transição pode ser acionada. No caso da Petri Net proposta neste laboratório, todas as transições podiam ser acionadas infinitas vezes. Para além disto, estando em qualquer estado, havia sempre uma sequência de eventos que levava a uma qualquer transição específica a ser acionada. Dado isto, pode-se concluir que todas as transições da Petri Net proposta são de nível 4.

**Q10:** Study property #7, resorting to the methods studied in the course.

**Coverability**

Esta propriedade verifica-se analisando o número de tokens em cada lugar para dois estados consecutivos. Se, em qualquer lugar, o número de tokens no estado anterior for inferior ao número de tokens no estado seguinte, então o estado actual não é coverable por outros lugares alcancáveis. Na Petri net proposta neste laboratório, só há um token a transitar entre lugares, pelo que todos os estados têm o token num lugar diferente. Desta forma, nenhum estado é coverable por algum lugar alcancável dado que o número no lugar actual vai sempre passar de 1 para 0 no estado seguinte. Por esta razão, esta propriedade não foi tida em conta, pelo que a sua importância para o funcionamento do keyboard é reduzida e estando assim colocada em último lugar.

**Q11:** *[Macro-place-expand]* Assume that one wants to replace (i) a single-input and single-output macro-place, i.e. one place, its input arc and its output arc by (ii) a sequence of places and in between transitions/arcs, where the complete set has just one input arc and one output arc. How would you change the incidence matrix of a Petri Net to include new places and transitions? Suggest a Matlab function prototype that allows doing the referred replacement of a place by a set of places and transitions.

A nova matriz incidência de modo a incluir os novos places e transições será do tipo:

D =

De modo a incluir novos places e transições, à matriz de incidêcia original elimina-se a linha correspondente ao place onde se quer inserir a nova Petri Net. Assim sendo, obtém-se uma nova matriz, *a*.

A matriz *b* será uma matriz de zeros com o número de colunas igual ao número de transições novas a incluir, e com o número de linhas igual ao número de places menos 1 da Petri Net original.

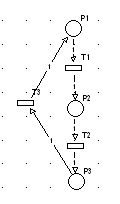
A matriz *c* será uma matriz com o número de colunas igual ao número de transições da Petri Net original, e com o número de linhas igual ao número de places novos a inserir. A primeira linha de *c*  corresponderá ao place onde ser quer inserir a nova Petri Net, pelo que esta linha terá apenas um 1 na posição correspondente à transição que antecede a Petri Net a inserir. A última linha desta matriz corresponde ao último place a inserir, e esta terá apenas um -1 na posição correspondente à transição que precede a Petri Net a inserir. As restantes linhas desta matriz serão nulas.

A matriz *d* corresponde à matriz de incidência da Petri Net que se quer inserir na Petri Net original.

Foi desenvolvida uma função que recebe as matrizes Pre e Post da Petri Net original, bem como o lugar onde ser quer inserir a nova Petri Net, e ainda as matrizes Pre e Post que definem a Petri Net a inserir. Esta função matlab encontra-se na pasta em conjunto com este relatório, e tem como nome *Insert\_macro\_place*.

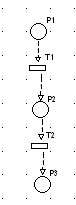
Para demonstração da função acima descrita, é apresentado um exemplo abaixo:

A Petri Net original está representada seguidamente, onde P2 é o macro-place.



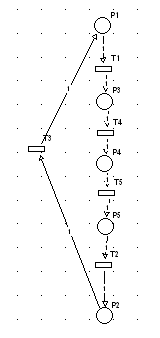
As matrizes Pre e Post desta Petri Net são:

No lugar de P2 pertende-se substituir por uma outra Petri Net:



As matrizes Pre e Post da Petri Net a inserir são:

Esta inclução resulta na seguinte Petri Net:



Da qual resulta a seguinte matriz de incidência:

**Q12:** *[PLC test]* Run the demonstration of the Petri nets to PLC code converter, named tst1\_blink\_on\_off.m, provided in the course webpage (see Annex 1). By analyzing the demonstration, describe what are the major programmed components (definitions) necessary, in addition to the Petri net, to build the PLC program.

Seguindo os procedimentos indicados, o ficheiro Matlab “tst1\_blink\_on\_off.m” chama o compilador, dando-lhe alguns dados importantes, como por exemplo:

PN= define\_petri\_net;

mu0= [1 0 0 0]';

pre= [1 0 0 0; 0 1 0 0; 0 0 1 0; 1 0 0 0; 0 0 0 1]';

pos= [0 1 0 0; 0 0 1 0; 1 0 0 0; 0 0 0 1; 1 0 0 0]';

Que é a representação da definição da Petri net, com 4 lugares e 5 transições, dada pela matriz D seguinte e o respectivo diagrama:

Matriz D:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| -1 | 1 | 1 | -1 | 1 |
| 1 | -1 | 0 | 0 | 0 |
| 0 | 1 | -1 | 0 | 0 |
| 0 | 0 | 0 | 1 | -1 |

# https://scontent.flis6-1.fna.fbcdn.net/v/t34.0-12/15319459_1274963345908041_1432658791_n.png?oh=348d05ce3da30679f298e979044074b2&oe=5841FE5D

P1 – Led vermelho

P2 – Led laranja

P3 – Led verde

P4 – OFF mode

T4 – Switch no modo OFF

T5 – Switch no modo ON

Figura 4 - Petri net implementada no PLC

Dá também a informação acerca das transições prioritárias, neste caso, a T4:

tprio= [4];

E ainda informação acerca do timer de controlo entre certos lugares e transições, neste caso 0.5 segundos será o tempo que cada lugar P1 P2 e P3 ficará ativado, correspondendo ao respetivo led estar aceso.

Para além da Petri Net, este ficheiro tem ainda que mapear os inputs físicos correspondentes a transições da Petri net. Neste caso:

input\_map = define\_input\_mapping;

O input do programa é a posição do switch SENSOR. Caso este seja posto no modo ON, correspondente ao input i0.2.2, a transição T5 é acionada e o programa fica num ciclo dos estados P1, P2 e P3, até que este seja desligado, isto é, até que o SENSOR seja colocado a OFF. Nesse caso é acionada a transição T4, que por ser prioritária garante que a Petri net deixa de transitar nos estados anteriores e passa para o P4.

O mesmo acontece para os outputs:

output\_map = define\_output\_mapping;

Estes são acionados quando o programa está a correr em ciclo em cada um dos estados P1, P2 e P3, indo corresponder respetivamente aos outputs %q0.4.1 (led vermelho), %q0.4.2 (led laranja) e %q0.4.3 (led verde).

Um dos scripts do compilador tem a função de converter os índices em endereços físicos do PLC, sendo as transições e lugares guardadas em palavras de memória. Para além disso, outro script é responsável pelos temporizadores dos estados referidos, e outro codifica a lista dos lugares em pequenos conjuntos de bits.

Com toda esta informação o compilador escreve num ficheiro o programa, em structured text, para dar ao PLC.

# Annex 1 - Convert a Petri net to a PLC program

In this annex is detailed the download and test of a converter of a Petri net to a PLC program. A demonstration example is included. The converter is written and runs in Matlab.

Please find in the webpage of the course the "Petri net compiler & demo" as a zip file. Decompress the zip file, change to folder ./tst1\_blink\_turn\_on\_off/ and run tst1\_blink\_on\_off.m . Note that the compiler is in folder ./pn\_to\_plc\_compiler/ which is going to be asked you the first time you run de demonstration.

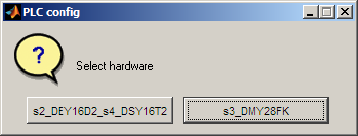


Figure 1: Selection of the hardware configuration.

In the first run of tst1\_blink\_on\_off.m it is asked the hardware configuration. Two configurations are considered, namely the "s2\_DEY16D2\_s4\_DSY16T2" and the "s3\_DMY28FK" (see figure 1). Configuration "s2\_DEY16D2\_s4\_DSY16T2" means the PLC has two modules, at slots two and four, and the modules are named "DEY16D2" and "DSY16T2". Configuration "s3\_DMY28FK" means the PLC has only the module "DMY28FK" mounted at slot three. In case you want to revisit this configuration you can either restart Matlab or run the command clear global.

After running the demonstration, you obtain file tst1\_mk\_program\_res.txt containing structured text code that you can copy into a Unity project. Within the Unity project you need to do declare some additional variables like the timer names, timing values and flags. Please see more details in the text file ./tst1\_blink\_turn\_on\_off/\_readme.txt.

1. Original guide by Prof. Paulo J. Oliveira. Revised by Prof. José Gaspar (2016). [↑](#footnote-ref-1)