**République Algérienne Démocratique et Populaire**

**Ministère de l'Enseignement Supérieur et de la Recherche Scientifique**
**Université Mouloud MAMMERI, Tizi-Ouzou**

Faculté de Génie Electrique et d'Informatique
Département d'Automatique

# MEMOIRE DE FIN D'ETUDES

En vue de l'obtention du diplôme

***DE MASTER PROFESSIONNEL EN AUTOMATIQUE***
***OPTION : AUTOMATIQUE ET INFORMATIQUE INDUSTRIELLE***

# *Thème*

***Matérialisation d'une commande pour un atelier à contraintes de temps de séjour.***


Proposé et dirigé par :

M. KARA Redouane

Présenté par :

M. OULD MOHAMMED Farid

M. OUADJER Omar


Soutenu le : 24 / 09 /2013

***Promotion 2013***

Ce travail a été préparé : au laboratoire de conception et commande des systèmes de production (L2CSP)

# *Remerciements*

*Nous commençons par remercier Dieu le tout puissant de nous avoir donné le courage et la volonté de mener ce travail.*

*Nos remerciements et reconnaissances à notre promoteur Mr R.KARA pour ses précieux conseils et orientations.*

*On tient aussi à remercier les membres de jury qui nos feront l'honneur d'évaluer ce travail.*

# *Dédicaces*

# *Dédicaces*

*Je dédie ce modeste travail :*

*A la mémoire de mon père.*

*A ma chère mère.*

*A mes frères et sœurs.*

*A toute ma famille.*

*A tous mes amis.*

*O. Farid*

# *Dédicaces*

*Je dédie ce modeste travail :*

*A la mémoire de mon père.*

*A toute ma famille.*

*A tous mes amis.*

*O. Omar*

# *Tables des matières*

**Chapitre II : Les réseaux de Petri et le temps**

**Chapitre IV : Commande d'un GDE p-temporel**

# *Table des illustrations*

**Liste des figures :**

# *Liste des symboles*

RdP : Réseau de Petri.

P : Place.

T : Transition.

Post : Matrice d'incidence arrière.

Pré : Matrice d'incidence avant.

M : Le marquage.

$M_i$ : Le nombre de marque dans la place $P_i$.

p-RdP : Réseau de Petri p-temporel.

t-RdP : réseau de Petri t-temporel.

GDE : Graphe d'événement.

# *Introduction générale*

# Introduction générale

L'automatique classique considère des systèmes dynamiques où le changement d'état dépend du temps, en continu. Le comportement des systèmes dynamiques en automatique classique est souvent modélisé par des équations différentielles ou aux différences. Par contre, les Systèmes dynamiques à Evénement Discret (SED) correspondent à des processus dont l'état change à l'occurrence d'un événement. Cela recouvre un grand nombre de processus comme les réseaux de transport, les systèmes informatiques, les systèmes multimédia.

Ce mémoire considère ainsi les systèmes à événements discrets qui font l'objet d'étude de nombreux chercheurs.

Les réseaux de Petri jouent un rôle important, ils présentent la caractéristique d'être un outil à la fois graphique et mathématique de modélisation. L'utilisation de ces graphes pour spécification et analyser de ces systèmes permet de décrire des phénomènes d'assemblage, de synchronisation, de partage de ressources.

Dans ce travail, notre choix s'est porté sur la modélisation graphique, plus précisément la modélisation par les réseaux de Petri p-temporels.

L'objectif de ce travail est d'utiliser les réseaux de Petri p-temporels pour la surveillance d'atelier à contraintes de temps de séjours et de commander l'atelier en temps réel. Nous présentons ainsi une méthode qui permet de matérialiser les places et les transitions d'un GDE p-temporels avec des bascules SR. Exploitant cette matérialisation, nous montrons que nous pouvons programmer les GDE p-temporels sous Step7 (programme LADDER).

Un autre objectif est de commander en boucle fermée un GDE p-temporel modélisé par les équations linéaires implicites dans le but d'atteindre un fonctionnement désiré.

Le présent mémoire est articulé autour de quatre chapitres qui sont :

- Le premier et le deuxième chapitre portent sur les réseaux de Petri en général. On étudie ainsi en détail les réseaux de Petri temporels.

Ces chapitres nous permettent d'aborder les méthodes d'analyse des réseaux de Petri en utilisant la méthode graphique ou la méthode algébrique.

Dans ce travail, notre choix s'est porté sur la modélisation graphique, plus précisément la modélisation par les graphes d'événements p-temporels, dont on a présenté la méthode pour la recherche du temps de cycle ainsi que les dates de franchissement des transitions de ces graphes, cette méthode est basée sur la programmation linéaire.

- Le troisième chapitre est consacré à l'utilisation des réseaux de Petri p-temporels (GDE p-temporels) pour la surveillance d'ateliers à contraintes de temps de séjour.

Nous présentons comment mettre en œuvre un réseau de Petri p-temporel (GDE p-temporel) sous step7 (programme LADDER). On a également expliqué comment matérialiser un GDE p-temporel (places, transitions) par des bascules SR.

- Dans le quatrième, on s'intéresse essentiellement aux GDE p-temporels modélisés par des équations d'état implicites et leur étude.

L'objectif de ce chapitre est de commander (contrôler) en boucle fermée un GDE p-temporel modélisé par des équations linéaires implicites dans le but d'atteindre un fonctionnement désiré tout en assurant l'existence de solution, le respect des contraintes imposées sur l'état de la commande et la convergence asymptotique de la trajectoire vers un comportement désiré.

A la fin de chapitre on a étudié un exemple d'application : étude d'une ligne de galvanoplastie (ligne de traitement de surface).

# Chapitre I :

# Les réseaux de Petri

## I.1 Introduction

Le modèle des réseaux de Petri (RdPs) a été crée en 1962 par Carl Adam Petri dans sa thèse intitulée « communication avec les automates », afin de modéliser les processus communicants **[1].**

Les réseaux de petri (RdPs) sont des outils graphiques et mathématiques permettant de modéliser le comportement dynamique des systèmes à événements discrets comme les systèmes manufacturiers, les systèmes de télécommunications, les réseaux de transports.

- La représentation graphique permet de visualiser la synchronisation, le partage des ressources, les choix et les conflits.
- La représentation mathématique permet d'établir les équations d'états à fin de déterminer les propriétés du modèle et de les comparer avec le comportement du système modélisé **[5].**

## I.2 Notions de bases

### I.2.1 Définition d'un réseau de Petri :

Un réseau de Petri est un graphe biparti, composé de deux sommets : les places et les transitions. Des arcs orientés relient les places aux transitions et les transitions aux places. (Les places sont représentées par des cercles et les transitions sont représentées par des barres ou des rectangles). Un arc ne relie jamais deux sommets de même nature.

Un RdP est représenté par un quadruplet R = (P, T, Pré, Post) où :

- P : est un ensemble fini et non vide de places, $P = \{P_1, P_2, \ldots, P_n\}$.
- T : est un ensemble fini et non vide de transitions, $T = \{T_1, T_2, \ldots, T_m\}$.
- Pré : $(P \times T) \rightarrow N$ (N ensemble des entiers naturels) est l'application d'incidence avant.

Pré $(P_i, T_j)$ est le poids de l'arc reliant la place $P_i$ à la transition $T_j$.

- Post : $(P \times T) \rightarrow N$ (N ensemble des entiers naturels) est l'application d'incidence arrière.

Post $(P_i, T_j)$ est le poids de l'arc reliant la transition $T_j$ à la place $P_i$ **[6].**

**Remarque I.1 :**

*Lorsque Pré $(P_i, T_j)$ et Post $(P_i, T_j)$ prennent leurs valeurs dans la paire {0, 1}, le réseau est dit ordinaire.*

**Notation**

$^{\circ}P_i$ : l'ensemble des transitions d'entrée de $P_i$.

$P_i^{\circ}$ : l'ensemble des transitions de sortie de $P_i$.

$^{\circ}T_j$ : l'ensemble des places d'entrée de $T_j$.

$T_j^{\circ}$ : l'ensemble des places de sortie de $T_j$.

### I.2.2 Notation matricielle d'un réseau de Petri

Pour un réseau de Petri avec *n* places et *m* transitions, la matrice d'incidence

$C = [C_{ij}]_{nxm}$ est une matrice de dimension (*n.m*), dont les composantes sont données par :

$$\mathbf{C_{ij} = Post\ (P_i, T_j) - Pré\ (P_i, T_j)}$$ Où :

- Post ($P_i$, $T_j$) est le poids de l'arc reliant la transition $T_j$ à sa place de sortie $P_i$
- Pré ($P_i$, $T_j$) est le poids de l'arc reliant $T_j$ à sa place d'entrée $P_i$ **[4].**

**Exemple :**

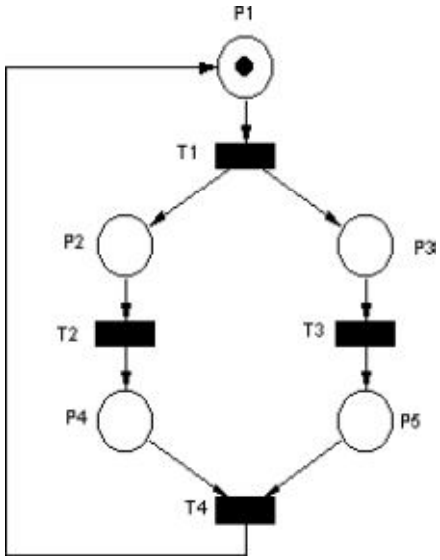



**Figure I.1** : Exemple d'un réseau de Petri

- Les matrices Pré, Post et la matrice d'incidence sont donnés par :

$$\text{Pré}(P_i, T_j) = \begin{array}{c} \begin{matrix} T_1 & T_2 & T_3 & T_4 \end{matrix} \\ \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 1 \end{pmatrix} \end{array} \begin{matrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{matrix} \quad ; \quad \text{Post}(P_i, T_j) = \begin{array}{c} \begin{matrix} T_1 & T_2 & T_3 & T_4 \end{matrix} \\ \begin{pmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{pmatrix} \end{array} \begin{matrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{matrix}$$

$$C = \text{Post}(P_i, T_j) - \text{Pré}(P_i, T_j) = \begin{array}{c} \begin{matrix} T_1 & T_2 & T_3 & T_4 \end{matrix} \\ \begin{pmatrix} -1 & 0 & 0 & 1 \\ 1 & -1 & 0 & 0 \\ 1 & 0 & -1 & 0 \\ 0 & 1 & 0 & -1 \\ 0 & 0 & 1 & -1 \end{pmatrix} \end{array} \begin{matrix} P_1 \\ P_2 \\ P_3 \\ P_4 \\ P_5 \end{matrix}$$

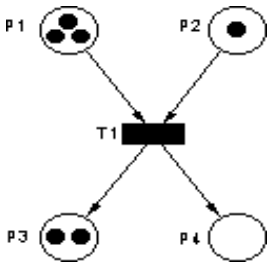
### I.2.3 Marquage

Chaque places contient un nombre entier positif ou nul de marques (jetons). Le marquage *M* définit l'état du système décrit par le réseau à un instant donné. C'est un vecteur colonne de dimension le nombre de places dans le réseau, la $j^{éme}$ composante de ce vecteur représente le nombre de marques dans la $j^{éme}$ place du réseau.

Un RdP marqué est un couple $M = (R, M_0)$ tel que :

- R est un réseau de Petri.
- $M_0 : P^n \rightarrow N^n$ est le marquage initial. N est l'ensemble des entiers naturels **[2].**

**Exemple :**




$$M_0 = \begin{pmatrix} 3 \\ 1 \\ 2 \\ 0 \end{pmatrix}$$

**Figure I.2** : Un réseau de Petri et son marquage initial.

### I.2.4 Franchissement d'une transition

Le franchissement d'une transition décrit le comportement dynamique du système modélisé. Un état est un marquage qui est modifié conformément aux règles de franchissement suivantes :

1) Une transition $T_j$ est dite validée si chaque place $P_i$ en amont (i.e. la place d'entée) contient un nombre de marques supérieur ou égal au poids de l'arc reliant ces places à $T_j$.

$$\mathbf{M(P_i) \geq Pré\ (P_i, T_j)}$$

2) Le franchissement d'une transition validée $T_j$ a pour conséquence :

(i) De retirer de chaque place d'entrée $P_i$ un nombre de marque égal au poids de l'arc $(P_i, T_j)$ reliant ces places d'entrée à $T_j$.

(ii) D'ajouter dans chaque place de sortie un nombre de marques égal au poids de l'arc $(P_i, T_j)$ reliant $T_j$ aux places de sortie.

Le franchissement d'une ou plusieurs transitions provoque alors le passage d'un marquage M à un autre marquage $M'$ tel que :

$$\mathbf{M'(P_i) = M(P_i) + Post\ (P_i, T_j) - Pré\ (P_i, T_j)}$$

On dit que le nouveau marquage $M'$ est accessible à partir de marquage initial $M_0$ **[4].**

**Remarque I.2**

- *Une transition franchissable n'est pas forcément immédiatement franchie.*
- *Une transition sans place d'entrée est toujours franchissable : c'est une* ***transition source****.*
- *Une transition sans place de sortie est* ***une transition puits****.*

## I.3 Les réseaux de Petri autonomes et non autonomes

### I.3.1 Réseau de Petri autonome

Un réseau de Petri autonome décrit le fonctionnement d'un système dont les instants de franchissement des transitions ne sont pas connus.

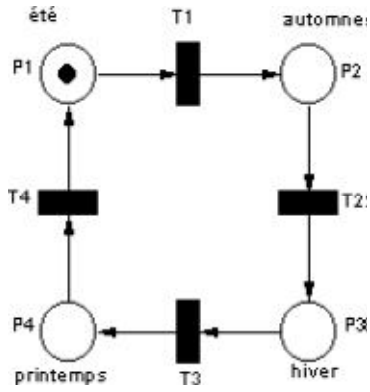



**Figure I.3** : Un réseau de Petri autonome.

La figure I.3 représente le cycle de saisons. A chaque saison est associée une place, et a chaque transition le passage d'une saison à une autre.

Le marquage correspond à l'été, mais on n'a aucune indication sur le moment où se produira le franchissement.

### I.3.2 Réseau de petri non autonome

Un réseau de Petri non autonome décrit le fonctionnement d'un système dont l'évolution est conditionné par des événements externes ou par le temps. Un réseau de Petri non autonome est synchronisé et/ou temporisé **[2].**

## I.4 Propriétés des réseaux de Petri

Les réseaux de Petri possèdent un certain nombre de propriétés (accessibilité, état d'accueil, la bornitude et la vivacité), dépondent du marquage initial du système et sont liées à l'évolution du réseau.

Elles permettent d'apporter des réponses aux questions, concernant l'accessibilité d'un marquage particulier, la bornitude, la vivacité **[5].**

### I.4.1 L'accessibilité

Le franchissement d'une transition validée dans un réseau de Petri apportera une modification au marquage initial. Un marquage M est dit accessible à partir de marquage initial $M_0$, s'il existe une séquence de franchissement $\sigma = T_1\ T_2\ \dots\ T_n$ qui transforme $M_0$ en M **[4].**

**Notation**

- $\mathbf{M_0\ [\sigma > M}$ ce qui signifie que M est accessible à partir de $M_0$ par $\sigma$.
- $\mathbf{R(M_0)}$ l'ensemble des marquages accessible à partir de $M_0$.
- $\mathbf{L(M_0)}$ l'ensemble de toutes les séquences de franchissement possible (réalisable) à partir de $M_0$.

### I.4.2 Etat d'accueil et réseau de Petri réinitialisable

Un réseau de Petri possède un état d'accueil $M_a$ pour un marquage initial $M_0$ si pour tout marquage accessible $M \in R(M)$, il existe une séquence de franchissement $\sigma$ tel que :

$$\mathbf{M\ [\sigma > M_a}$$

Si le marquage initial $M_0$ est un état d'accueil, alors le réseau de Petri est réinitialisable **[4].**

### I.4.3 Réseau de Petri borné (Bornitude)

Un réseau de Petri est borné pour un marquage initial $M_0$ si toutes ses places sont bornées pour $R(M_0)$ (le réseau de Petri est k-borné si toutes les transitions sont k-bornées).

Une place est dit bornée pour un marquage initial $M_0$ s'il existe un entier naturel k tel que, pour tout marquage accessible à partir de $M_0$, le nombre de marques de cette place est inférieur ou égal à k.

- Un réseau de Petri est dit **sauf** s'il est 1-borné **[6].**

### I.4.4 Vivacité

Un réseau de petri est dit vivant si et seulement si, pour tout marquage accessible

$M \in R(M_0)$, il est possible de franchir toute transition de réseau en progressant à travers une séquence de franchissement.

La vivacité d'un réseau garantit le franchissement de toute transition quel que soit le marquage atteint.

Un réseau de Petri vivant est un réseau de Petri sans blocage. Un blocage est un marquage tel qu'aucune transition n'est validée **[2].**

## I.5 Réseaux de Petri particuliers

### I.5.1 Graphe d'état

Un réseau de Pétri est un graphe d'état si et seulement si toute transition a exactement une seule place d'entrée et une seule place de sortie **[2].**

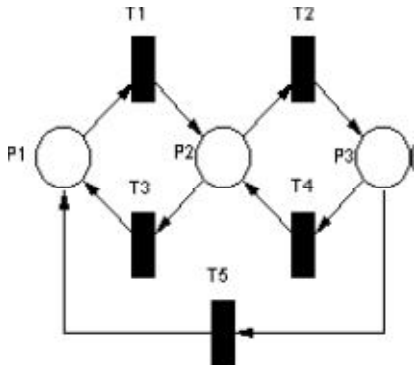



**Figure I.4** : Un graphe d'état.

### I.5.2 Graphe d'événements

Un Réseau de Petri est un graphe d'événement si et seulement si chaque place possède exactement une seule transition d'entrée et une seule transition de sortie **[2].**

### I.5.3 Réseau de Petri sans conflits

Un réseau de Petri sans conflit est un réseau dans lequel chaque place a au plus une transition de sortie.

Un réseau de Petri avec conflit est un réseau qui possède donc une place avec au moins deux transitions de sorties.

Un conflit est noté: $[P_i, \{T_1, T_2,\ldots, T_n\}]$ ; avec $T_1, T_2, \ldots, T_n$ étant les transitions de sorties de la place $P_i$ **[2].**

### I.5.4 Réseau de Petri à choix libre

Un réseau de Petri à choix libre est un réseau dans lequel pour tout conflit $[P_i, \{T_1, T_2,\ldots, T_n\}]$ aucune des transitions $T_1, T_2,\ldots, T_n$ ne possède aucune autre place d'entrée que $P_i$ **[2].**

### I.5.5 Réseau de Petri simple

Un réseau de Pétri simple est un réseau de petri dans lequel chaque transition ne peut être concernée que par un conflit au plus **[2].**

### I.5.6 Réseau de petri pur

Un réseau de Petri pur est un réseau dans lequel il n'existe pas de transition ayant une place d'entrée qui soit à la fois place de sortie de cette transition **[2].**

## I.6 Méthodes d'analyse

Après avoir modélisé un système physique par un réseau de Petri, une analyse du modèle peut être effectuée.

L'analyse du modèle peut se faire en utilisant :

- Le graphe des marquages accessibles ;
- L'algèbre linéaire ;

### I.6.1 Graphe des marquages accessibles et graphe de couverture

#### I.6.1.1 Graphe des marquages accessibles

Le graphe des marquages accessibles est composé de nœuds qui représentent les marquages accessibles et d'arcs correspondant aux franchissements des transitions permettant le passage d'un état à un autre **[5].**

**Exemple :**

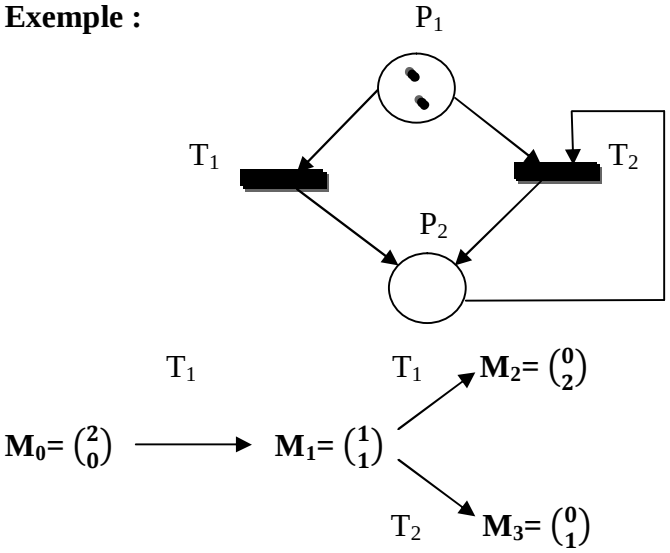



**Figure I.5** : Réseau de Petri avec son graphe de marquage.

Les propriétés déterminées à partir de ce graphe des marquages sont :

- deux blocages $M_2$ et $M_3$.
- 2-borné.
- non vivant.
- non réinitialisable.

### I.6.1.2 Graphe de couverture

Dans le cas où le réseau de Petri est non borné le graphe de marquage peut être infini. Pour pouvoir limiter la taille du graphe on introduit le symbole **'w',** qui peut être considéré comme un nombre entier, w > n tel que n ϵ N.

Pour tout entier n, w + n = w – n = w, ∀ n < w [4].

**Exemple :**

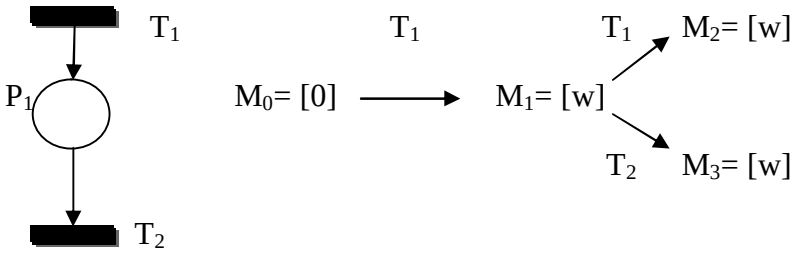



**Figure I.6** : Le graphe de couverture.

## I.7 L'algèbre linéaire

La deuxième méthode d'analyse est l'analyse par l'algèbre linéaire, cette dernière utilise la notion de la matrice d'incidence et des équations algébriques qui permettent la détermination des propriétés des réseaux de Petri **[2].**

**Notion d'équation d'état**

Soit $\sigma$ une séquence de franchissement réalisable à partir d'un marquage $M_i$, noté $M_i$ $[\sigma > M_k$.

Le vecteur caractéristique de la séquence $\sigma$, noté $\underline{\boldsymbol{\sigma}}$ est le vecteur de dimension m dont la $j^{éme}$ composante correspond au nombre de franchissement de la transition $T_j$.

Alors on peut écrire :

$$\mathbf{M_k = M_i + C\ \underline{\sigma}}$$

**Où** C désigne la matrice d'incidence du réseau de Petri **[2].**

**Exemple :**

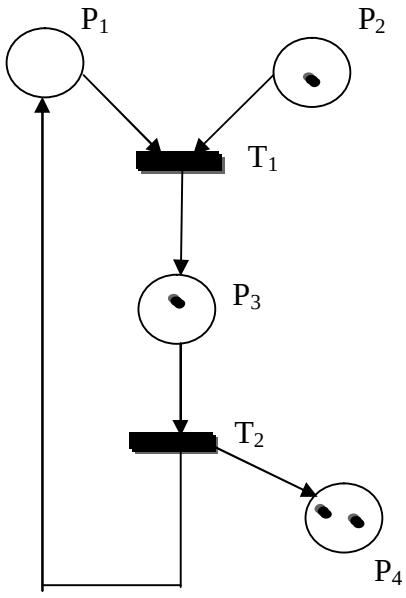



**Figure I.7** : Exemple.

$$M\,[T_2\,T_1 > M_0,\ \underline{\sigma} = \begin{pmatrix}1\\1\end{pmatrix},\ C = \begin{bmatrix}-1 & 1\\ -1 & 0\\ 1 & -1\\ 0 & 1\end{bmatrix},\ M_0 = \begin{bmatrix}0\\1\\1\\2\end{bmatrix}$$

$$\mathbf{M = M_0 + C.}\ \underline{\sigma} = \begin{bmatrix}0\\1\\1\\2\end{bmatrix} + \begin{bmatrix}-1 & 1\\ -1 & 0\\ 1 & -1\\ 0 & 1\end{bmatrix} \times \begin{pmatrix}1\\1\end{pmatrix} = \begin{bmatrix}0\\0\\1\\3\end{bmatrix}$$

L'équation d'état donne le marquage final connaissant le marquage initial et la séquence de franchissement de transition, mais elle ne permet pas de vérifier si le franchissement d'une séquence donnée est possible.

Si la séquence de franchissement $\sigma$ est réalisable (possible) alors l'équation d'état donne le marquage obtenu en franchissant cette séquence. L'inverse n'est pas vrai, c-à-d qu'un vecteur positif ou nul qui vérifie cette équation ne constitue pas forcément une séquence réalisable. Cette équation est une condition nécessaire d'accessibilité.

**Les invariants**

- **Les P-invariants**

**Définition :**

Un P-invariant est vecteur 'Y', à composantes entières positives de dimension m vérifiant :

$$\mathbf{Y^T.C = 0,}$$

Où C est la matrice d'incidence.

Soient Y un P-invariant et $M_0$ le marquage initial d'un réseau de Petri. Pour tout marquage M accessible à partir de $M_0$, on a :

$$\mathbf{Y^T.M_0 = Y^T.M}$$

Le support d'un P-invariant, noté $\|\boldsymbol{Y}\|$ est appelé la composante conservative d'un réseau de Petri.

$\|y\|$ L'ensemble des place qui correspondent aux composantes non nulles de Y **[2].**

**Exemple :**

$$Y = \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix} \quad ; \quad \|y\| = \{P_1, P_2\}.$$

- **Les T-invariants**

**Définition :**

Un T-invariant est vecteur 'X', à composantes entières positives de dimension n vérifiant :

$$\mathbf{X.C = 0,}$$

Où C est la matrice d'incidence.

Soit $\boldsymbol{\sigma}$ une séquence de franchissement réalisable et $\underline{\boldsymbol{\sigma}}$ son vecteur caractéristique. Si $\underline{\boldsymbol{\sigma}}$ = X est un T-invariant, le marquage obtenu à partir de marquage initial $M_0$ en franchissant la séquence $\boldsymbol{\sigma}$, on a

$$M = M_0 + C.X$$

C.X = 0, Car X est un T-invariant, par conséquent $M=M_0$.

Le support d'un T-invariant, noté ||x|| est appelé la composante répétitive stationnaire du réseau de Petri **[6].**

## I.8 Conclusion

Dans ce chapitre nous avons introduit les réseaux de Petri, comme outil de modélisation, leurs propriétés et certaines classes particulières qui vont nous servir dans la suite de ce mémoire.

# *Chapitre II :*

# *Les réseaux de Petri et le temps*

## I.1 Introduction

Dans le présent chapitre on a introduit les réseaux de Petri dépendant du temps pour tenir compte des contraintes temps dans la modélisation. Il existe principalement deux types d'extensions : les réseaux de Petri temporisés et les réseaux de Petri temporels.

Dans un réseau de Petri Temporisé une temporisation est associée à une place (on parle de réseau de Petri P-Temporisé) ou à une transition (on parle de réseau de Petri T-Temporisé).

Dans un réseau de Petri Temporel une temporisation est associée à une place (on parle de réseau de Petri P-Temporel) ou à une transition (on parle de réseau de Petri T-Temporel).

Dans notre étude on s'intéresse aux réseaux de Petri P-temporel.

## II.2 Les réseaux de Petri temporisés

Il s'agit de prendre en considération des contraintes temps dans la modélisation. L'introduction des temporisations dans un réseau de Petri (les réseaux de Petri temporisés) permettent une étude quantitative du modèle (évaluation des performances, dimensionnement).

Dans un réseau de Petri temporisé, les temporisations sont associées :

- Soit aux places (un temps qui correspond à la durée minimale de séjour d'un jeton dans une place) modèle **P-temporisé.**
- Soit aux transitions (un temps qui correspond à la durée de franchissement de cette transition) modèle **T-temporisé.**

### II.2.1 Les réseaux de Petri P-temporisés

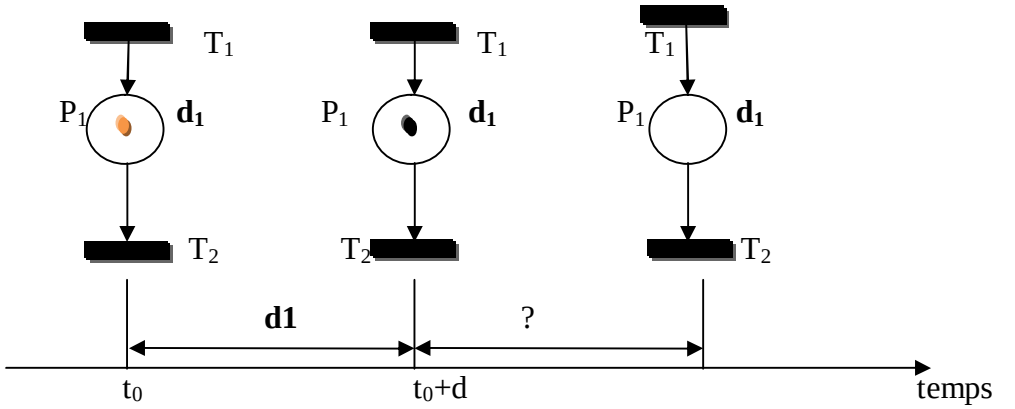
#### II.2.1.1 Définition d'un réseau de Petri P-temporisé

Un réseau de Petri P-temporisé est un doublet (R, Tempo) tel que :

- R est un réseau de Petri marqué.
- Tempo : P → N est la fonction de temporisation.

Tempo ($P_i$) = $d_i$, est la temporisation de la place $P_i$ **[1].**

### II.2.1.2 Règle de franchissement




**Le franchissement de $T_1$ :**
**La marque est déposée dans $P_1$, mais elle est indisponible, Donc $T_2$ n'est pas franchissable.**

**Au bout de $d_1$ unités de temps, la marque devient disponible la transition $T_2$ est maintenant Franchissable.**

**Le franchissement de $T_2$**

**Figure II.1** : Le fonctionnement d'un réseau de Petri P-temporisé.

La règle de franchissement doit tenir compte du temps. Soit $t_0$ l'instant où la marque est déposée dans la place $P_1$. La marque est indisponible pendant une durée égale à $d_1$, c'est-à-dire dans l'intervalle $[t_0, t_0+d_1]$.

Quand $d_1$ est écoule, la marque devient disponible comme le montre la figure II.1 **[1].**

### II.2.2 Les réseaux de Petri T-temporisés

### II.2.2.1 Définition d'un réseau de Petri T-temporisé

Un réseau de Petri T-temporisé est doublet (R, Tempo) tel que :

- R est un réseau de Petri marqué.
- Tempo : T → N est la fonction de temporisation.

Tempo ($T_j$) = $d_j$, est la temporisation de la transition $T_j$ **[1].**

### II.2.2.2 Règle de franchissement

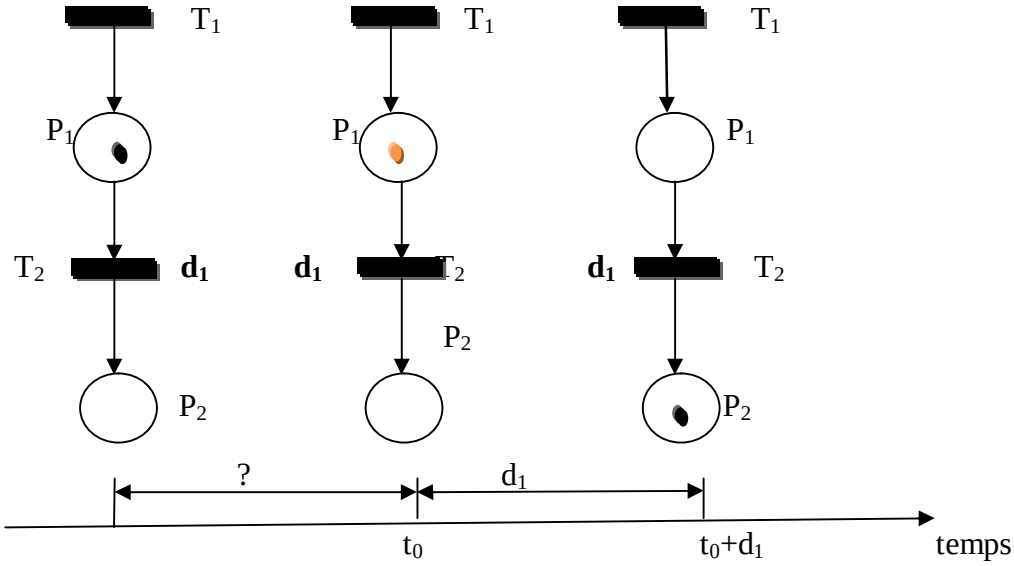



**Figure II.2** : le fonctionnement d'un réseau T-temporisé.

La transition $T_1$ est franchie, une marque est déposée dans $P_1$ ce qui entraine la validation de $T_2$ qui sera franchissable à n'importe quel moment.

Quand le franchissement est décidé, la marque nécessaire à ce franchissement est réservée. Lorsque la durée $d_1$ est écoulée depuis la décision de franchissement la transition est effectivement franchie.

### II.2.3 Les propriétés des réseaux de Petri temporisés

Les propriétés des réseaux de Petri temporisés qui peuvent être étudiées sont les mêmes que les réseaux de Petri et elles ont les même définitions.

- Les propriétés dynamiques des réseaux de Petri ne se conservent pas tout le temps lorsque l'on introduit une temporisation.
  - Un réseau de Petri temporisé peut être vivant alors que son réseau sous-jacent ne l'est pas.
  - Un réseau de Petri temporisé est nécessairement borné si le réseau sous-jacent est borné.

- En ce qui concerne les propriétés structurelles les temporisations associées aux réseaux de Petri temporisés imposent des contraintes supplémentaires sur le fonctionnement des réseaux sous-jacent.
    - L'ensemble des marquages accessibles d'un réseau de Petri temporisé est un sous ensemble des marquages des réseaux de Petri sous-jacent, et on trouve par conséquence :

Les T-invariants (respectivement P-invariants) des réseaux de Petri temporisé sont aussi les T-invariants (respectivement P-invariants) des réseaux de Petri sous jacent **[3].**

### II.3 Les réseaux de Petri temporels

Les modèles vus précédemment, P-temporisé (respectivement T-temporisé) les temporisations associées aux places (respectivement aux transitions) ne donnent pas les temps de séjour exacts des marques dans les places.

Il est impératif de prendre en considération des **fenêtres temporelles** pour pouvoir déterminer ces temps de séjour.

### II.3.1 Les réseaux de Petri t- temporels (t-RdPs) :

Les Réseaux de Petri t-temporels, appelé aussi le modèle de **Merlin.**

Dans ce modèle, un intervalle de temps $[a_j, b_j]$ est associé à chaque transition $T_j$ de réseau. Cet intervalle est relatif au moment où la transition devient validée.

Supposons que t est validée à l'instant $\theta$ , alors elle peut être franchie seulement entre $(a+\theta)$ et $(b+\theta)$ sauf si elle devient non validée à cause de franchissement d'une autre transition avec laquelle elle était en conflit **[6].**

#### II.3.1.1 Définition d'un réseau de petri t-temporel

Un réseau de Petri t-temporel (t-RdP) est un doublet (R, IS) où :

- R est un réseau de Petri marqué au sens autonome.
- IS est la fonction intervalle de tir statique :

$$IS: \begin{cases} T \rightarrow Q^+ \times Q^+ \cup \{\infty\} \\ T_j \rightarrow IS(T_j) = [a_j, b_j] \quad \text{avec } a_j \leq b_j \end{cases}$$

$IS(T_j)$ définit l'intervalle statique associé à la transition $T_j$, dans lequel elle peut être tirée.

- $a_j$ est la borne inférieure de $IS(T_j)$ = date statique de tir au plus tôt.
- $b_j$ est la borne supérieure de $IS(T_j)$ = date statique de tir au plus tard.

L'intervalle $[a_j, b_j]$ associé à la transition $T_j$ est relatif à l'instant de validation de la transition **[2].**

### II.3.1.2 Règle de franchissement :

Supposons que $T_j$ est validée à un instant $\theta$ :

Le franchissement de $T_j$ à une date $\theta$ si et seulement si :

- $\theta$ n'est pas inférieure à la date de tir au plus tôt de $T_j$ ($T_j$ ne peut être franchie avant $\theta + a_j$).
- $\theta$ n'est pas supérieur à la date de tir au plus tard de $T_j$ ($T_j$ doit être franchie à l'instant $\theta+b_j$) **[2].**

**Exemple :**

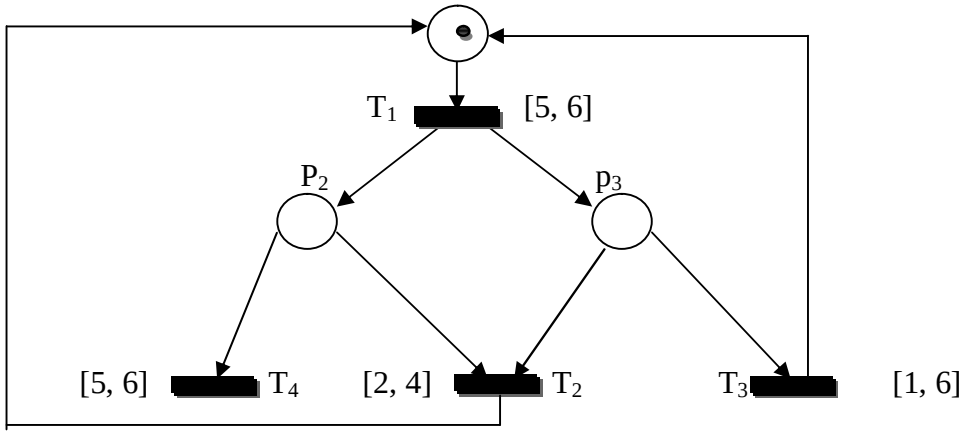



**Figure II.3** : Exemple d'un réseau de Petri t-temporel (t-RdP).

## II.4 Description du fonctionnement d'un réseau temporel

### II.4.1 Etat d'un réseau temporel et franchissement de transitions

Pour pouvoir caractériser le comportement d'un réseau de Petri t-temporel(t-RdP) la Notion d'état doit être défini dans un premier temps.

#### II.4.1.1 Définition d'un état d'un réseau de Petri t-temporel :

L'**état** d'un t-RdP est représenté par une paire E= (M, I) :

- M, est une application de marquage, affectant un nombre de marques à chaque place du réseau ($\forall p \in P, M(p) \geq 0$).
- I**,** une application intervalle de tir, associant à chaque transition du réseau l'intervalle de temps dans lequel doit être tirée. Les intervalles de tir de l'état E différent des intervalles attribués initialement aux transitions du réseau.

La définition de l'état nous donne la possibilité de déterminer les conditions de franchissement des transitions depuis un état.

Une transition T est franchissable à un instant θ depuis un état E = (M, I) si et seulement si les deux conditions suivantes sont satisfaites :

1) La transition T est validée par le marquage M au sens des RdPs autonomes

$$M(p) \geq Pré(P, T)$$

2) $\boldsymbol{\theta}$ est compris (bornes incluses) entre la date de tir au plus tôt de t et la plus petite des dates de tir au plus tard des autre transitions validées (dans l'état **E**) **[2].**

### II.4.1.2 Algorithme de calcul de l'état suivant :

Lorsque l'état E= (M, I) est atteint à l'instant $\theta$, le franchissement d'une transition T franchissable nous conduit à un nouvel état E'= (M', I'), déterminer par :

(1) le nouveau marquage M' :

$$\forall p \in P \quad M'(P)=M(P) - Pré(P_i, T_j) + Post(P_i, T_j)$$

Les nouveaux intervalles de tir I' pour les transitions :

(i) Pour toutes les transitions i non-validées par le marquage M', $I_i$' est vide (I=∅).

(ii) Pour toutes les transitions j validée par le marquage M et non en conflit avec la transition T :

$$I' = [\max(0, a_j - \theta), b_j - \theta] = [a_j', b_j'] ;$$

$[a_j', b_j']$ est l'intervalle dynamique associé à la transition j.

(iii) Les intervalles associés aux autres transitions sont les intervalles statiques **[6].**

**Exemple :**

Dans l'exemple de la figure (II.3), le marquage initial $M_0= (1\ 0\ 0)^T$, $T_1$ est sensibilisée (validée) à $\theta_0=0$, elle pourra être tirée (franchie) à $\theta_1 \in [5, 6]$, par exemple à $\theta_1= 5.5$. Le marquage atteint est alors $M_1= (0\ 1\ 1)^T$. Avec ce nouveau marquage, $T_2$, $T_3$ et T4 sont validées, avec les intervalles de tir statiques respectifs [2, 4], [1, 6] et [5, 6]. On trouve deux conflits structurels : le premier entre $T_2$ et $T_3$ et le second entre $T_2$ et $T_4$.

Le conflit entre $T_2$ et $T_4$ n'est pas effectif car les temporisations donnent la priorité à$T_2$. Par contre si le conflit entre $T_2$ et $T_3$ est résolu en faveur de $T_3$, $T_4$ pourra être franchie.

- $T_2$ est franchie à $\theta_2 \in [2, 4]$, $T_3$ et $T_4$ sont invalidée avant expiration de leur intervalles de tir statiques, le marquage atteint est $M_0$.

- $T_3$ est franchie à $\theta_3 \in [1, 4]$ ($T_3$ ne peut pas attendre au-delà de 4, à cause de la temporisation de $T_2$). $T_2$ est donc invalidée. Si $\theta_3$=2, par exemple, alors $T_4$ devra être tirée (à partir de $\theta_3$) entre les instants 3 et 4.

$$E_0=\begin{pmatrix}1\\0\\0\end{pmatrix};\begin{pmatrix}[5,6]\\\emptyset\\\emptyset\\\emptyset\end{pmatrix} \xrightarrow[\Theta_1=5.5]{T_1} E_1=\begin{pmatrix}0\\1\\1\end{pmatrix};\begin{pmatrix}\emptyset\\[2,4]\\[1,6]\\[5,6]\end{pmatrix} \xrightarrow[\Theta_2=1.5]{T_3} E_2=\begin{pmatrix}1\\1\\0\end{pmatrix};\begin{pmatrix}[5,6]\\\emptyset\\\emptyset\\[3.5,4.5]\end{pmatrix}$$

### II.4.2 Notion de classe d'état

Un nombre infini d'états peut être obtenus par écoulement de temps (un état peut avoir une infinité de successeurs). De ce fait la construction du graphe des marquages accessibles est impossible. On définit donc un graphe qui est proche du graphe des états, mais qui soit fini (classes d'états).

#### II.4.2.1 Définition d'une classe d'état

Une classe d'état regroupe tous les états accessibles partant de l'état initial, en franchissant une séquence de transitions $\sigma$.

Les états d'une même classe auront le même marquage et le domaine de tir, on note

C = (M, D) où :

- M est le marquage de la classe (même marquage pour tous les états de la classe).
- D est le domaine de tir de la classe, défini par l'union des intervalles de tir associés à chaque état de la classe **[3].**

**Exemple :** Reprenons le réseau de la figure (II.3).

On a $E_0$ : $M_0 = \begin{cases}1\\0\\0\end{cases}$ et $D_0 = 5 \leq \Theta 1 \leq 6$

Si on tir $T_1$ à un instant $\Theta_1$ on obtient : $E_1$ : $M_1 = \begin{cases}0\\1\\1\end{cases}$ et $D_1 = \begin{cases}2 \leq \Theta 2 \leq 4\\1 \leq \Theta 3 \leq 6\\5 \leq \Theta 4 \leq 6\end{cases}$

Si on tir $T_3$ à une date $\Theta_3$ tel que $\Theta_3 \in [1, 4]$ à cause de la condition de tir :

$$aj \leq \Theta i \leq \min_{I \neq \emptyset}\{bj\}$$

$$1 \leq \Theta 3 \leq min(4,6,6)$$

On aura l'état $E_2$ : $M_2 = \begin{cases} 1 \\ 1 \\ 0 \end{cases}$ et $D_2 = \begin{cases} 1 \leq \Theta 3 \leq 4 \\ 5 - \theta 3 \leq \Theta 4 \leq 6 - \Theta 4 \\ 5 \leq \Theta 1 \leq 6 \end{cases}$

Le domaine de tir d'une classe d'états peut être défini comme l'ensemble de solutions d'un système d'inégalités où les variables sont associées aux transitions validées par le marquage de classe considérée.

$$\mathbf{D = \{t/A.\ t \geq b\}} \qquad \text{(II.1)}$$

où :

- A, une matrice de coefficients
- b, un vecteur de constante
- t, un vecteur de variables, fonction des transitions validées par le marquage de la classe considérée **[6]**.

### II.4.2.2 Condition de franchissement d'une transition depuis une classe d'état

Une transition $T_j$ franchissable depuis la classe d'état précédente si et seulement si :

- $T_j$ validée par le marquage M au sens réseau de Petri autonome.
- $T_j$ peut être tirée avant toute autre transition $T_i$ aussi validée dans la classe C **[6].**

### II.4.2.3 Calcul de la classe suivante

Supposons la transition $t_i$ soit franchissable depuis la classe C= (M, D), avec

D = {t/A. t≥ b}.

La classe suivante C' = (M', D') est engendré par le franchissement de $t_i$.

Où :

- Le marquage M'est obtenu par :

$$\mathbf{M'(P_i) = M(P_i) + Post\ (P_i, T_j) - Pré\ (P_i, T_j)} \qquad \text{(II.2)}$$

- Le domaine D'obtenu comme suit :

i. On ajoute au système $A.t \geq b$ les conditions de tir de $t_i$, exprimant qu'elle est tirée la première parmi les transitions validées :

$$\mathbf{A.t \geq b,} \quad \mathbf{t_i \leq T_j,} \ \forall\, \boldsymbol{j \neq i} \qquad \text{(II.3)}$$

ii. Supprimer par substitution dans le nouveau système, les variables $t_j$ associées aux transitions en conflit avec $t_i$.

iii. Dans le système ainsi réduit, effectuer le changement de variable suivant :

$$\forall\, \boldsymbol{j \neq i,}\ \mathbf{t_j = t_i + t'_j}$$

Et éliminer par substitution toutes les occurrences de la variable $t_i$, pour ne garder que les nouvelles variables $t'_j$. On définit le domaine de tir pour les transitions qui sont restées validées pendant le tir de $t_i$.

iv. Compléter ce dernier système par une variable supplémentaire pour chaque transition nouvellement validée et associée à ces derniers leur intervalle de tir statique **[3]**.

**Exemple**

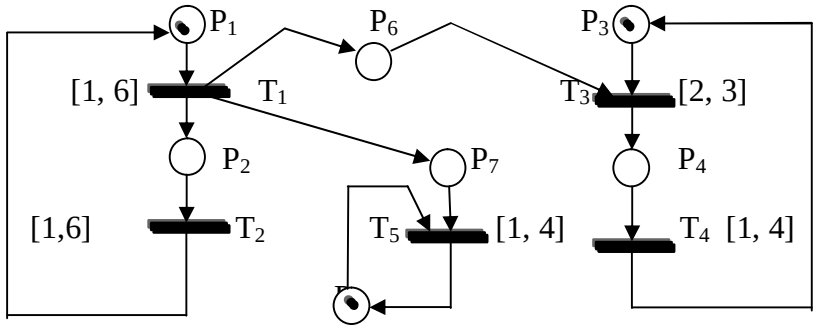



**Figure II.4** : Exemple explicatif.

$$C_0 : \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 0 \\ 0 \end{pmatrix}, \; 1 \leq \Theta 1 \leq 6 \xrightarrow{T_1} C_1 : \begin{pmatrix} 0 \\ 1 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \begin{cases} 1 \leq \Theta 2 \leq 6 \\ 2 \leq \Theta 3 \leq 3 \\ 1 \leq \Theta 5 \leq 4 \end{cases}$$

On peut franchir les transitions $T_2$, $T_3$ et $T_5$ respectivement dans les intervalles [1, 3], [2, 3] Et [1, 3]. A cause de la condition de tir suivante :

$$aj \leq \Theta i \leq \min_{I \neq \emptyset} \{bj\}$$

On peut par exemple franchir $T_2$ si et seulement si le système :

$$\begin{cases} 1 \leq \Theta 2 \leq 6 \\ 2 \leq \Theta 3 \leq 3 \\ 1 \leq \Theta 5 \leq 4 \\ \Theta 2 \leq \Theta 3 \\ \Theta 2 \leq \Theta 5 \end{cases}$$

Admet une solution (ce qui donne $1 \leq \Theta 2 \leq 3$).Puis après le tir de $T_2$ à l'instant $\Theta 2$, on effectuant le changement de variables suivant :

$$\Theta 3 = \Theta 2 + \Theta' 3$$

$$\Theta 5 = \Theta 2 + \Theta' 5$$

Nous obtenons :

$$\begin{cases} 1 \le \Theta 2 \le 6 \\ 2 \le \Theta 2 + \Theta' 3 \le 3 \\ 1 \le \Theta 2 + \Theta' 5 \le 4 \\ \Theta 2 \le \Theta 2 + \Theta' 3 \\ \Theta 2 \le \Theta 2 + \Theta' 5 \end{cases}$$

D'où, en éliminant $\Theta 2$ :

$$\begin{cases} 0 \le \Theta' 3 \\ 0 \le \Theta' 5 \\ -4 \le \Theta' 3 \le 2 \\ -5 \le \Theta' 5 \le 3 \\ -2 \le \Theta' 3 - \Theta' 5 \le 2 \end{cases}$$

La nouvelle classe est donc

$$C_2 : \begin{pmatrix} 1 \\ 0 \\ 1 \\ 0 \\ 1 \\ 1 \\ 1 \end{pmatrix}, \quad \begin{cases} 0 \le \Theta' 3 \le 2 \\ 0 \le \Theta' 5 \le 3 \\ -2 \le \Theta' 3 - \Theta' 5 \le 2 \end{cases}$$

## II.5 Réseau de Petri p-temporel (p-RdP)

Les Réseaux de Petri p-temporels, appelé aussi le modèle de **Khansa.**

Dans ce modèle, un intervalle de temps $[a_i, b_i]$ est associé à chaque places $P_i$ du réseau, ce dernier a été développé pour pouvoir modéliser et analyser les systèmes à contraintes de temps.

### II.5.1 Définition d'un réseau de Petri p-temporel

Un réseau de Petri p-temporel (p-RdP) est un doublet (R, IS) où :

- R est un réseau de Petri marqué au sens autonome.
- IS: $P \rightarrow Q^+ \times Q^+ \cup \{\infty\}$

  $P_i \rightarrow IS(P_i) = [a_i, b_i]$ avec $a_i \leq b_i$

$IS(P_i)$ définit l'intervalle statique de temps de séjour d'une marque (jeton) dans la place $P_i$. Cette marque ne participera pas à la validation des transitions dont $P_i$ est la place d'entrée que si elle a séjourné au moins la durée $a_i$ unité de temps dans cette place. La marque est considérée comme ''morte'' après $b_i$ unité de temps, et ne participera plus à la validation des transitions.

L'état de marque morte est spécifique au modèle des réseaux de Petri p-temporels, de même que la notion de séquence de tir marque-vivante **[6].**

A l'état initial, l'intervalle associé aux marques est $[0,\infty]$, et dès qu'une marque arrive dans une place par le franchissement d'une transition, elle prend l'intervalle associé à la place.

### II.5.2 Principe de fonctionnement

Le fonctionnement d'un réseau de Petri p-temporel est caractérisé par sa situation à un instant donné, la notion d'état nous donne la possibilité de caractériser cette situation.

A partir d'un état donné, à un instant donné, l'analyse et l'étude du comportement du réseau ce fait selon deux approches : « approche intervalle de temps » et approche « âge des marques ». La première approche est meilleure pour l'étude du comportement, la seconde est utilisée pour l'analyse des propriétés d'un réseau.

#### II.5.2.1 Définition fondée sur l'intervalle de temps

A un instant donné, l'état est défini par un doublet E= (M, I) avec :

- M : est l'application du marquage, affectant à chaque place un nombre de marques.

  $$(\forall p \in P, M(p) \geq 0) ;$$

- I : est une application intervalle dynamique de tir, noté $[a_i^k, b_i^k]$, associé à chaque marque k dans une place $P_i$. Cet intervalle dynamique est relatif à l'instant d'arrivée de la marque dans la place.

Supposons que la marque k arrive dans la place $P_i$ (son intervalle statique est $[a_i, b_i]$) à l'instant c.

A l'instant c+d ($a_i \leq d \leq b_i$) l'intervalle dynamique de k est **$[a_i^k, b_i^k]$= [max (ai - d, 0), bi – d].**

Le tir d'une transition dépend des intervalles dynamiques associés aux marques dans toutes ses places d'entrée **[6]**.

### II.5.2.2 Définition fondée sur l'âge des marques

A un instant donné, l'état est défini par un doublet E= (M, G), où :

- M : est l'application du marquage, affectant à chaque place un nombre de marques.

$$(\forall p \in P, M(p) \geq 0) ;$$

- G : est une application temps de séjour qui associe à chaque marque k dans la place Pi un nombre réel $g_i^k$ où $g_i^k$ est l'âge de cette marque (le temps écoulé depuis son arrivée dans la place $P_i$).

Soit $[a_i, b_i]$ l'intervalle statique associé à la place $P_i$. La marque k dans cette place peut participer à la validation des transitions de sortie si et seulement si :

(i) $g_i^k$ n'est pas inférieur à $a_i$, $g_i^k \geq a_i$.
(ii) $g_i^k$ n'est pas supérieur à $b_i$, $g_i^k \leq b_i$.

La marque k est morte quand son âge devient strictement supérieur à bi **[6]**.

Supposons que la marque k arrive dans la place $P_i$ à l'instant absolu t. L'âge de cette marque à cet instant est égal à zéro.
Son âge à l'instant absolu t'est $g_i^k = t - t'$. Elle ne participe à la validation des transitions de sortie de la place qui la contient, qu'à partir de l'instant $t' = t + ai$ et elle sera morte dès l'instant $t' > t + bi$.

Une marque sera morte si son âge devient strictement supérieur à la borne supérieure de l'intervalle statique associé à sa place d'accueil et si aucune des transitions de sortie de la place contenant cette marque n'est validée à cet instant.

**Exemple**

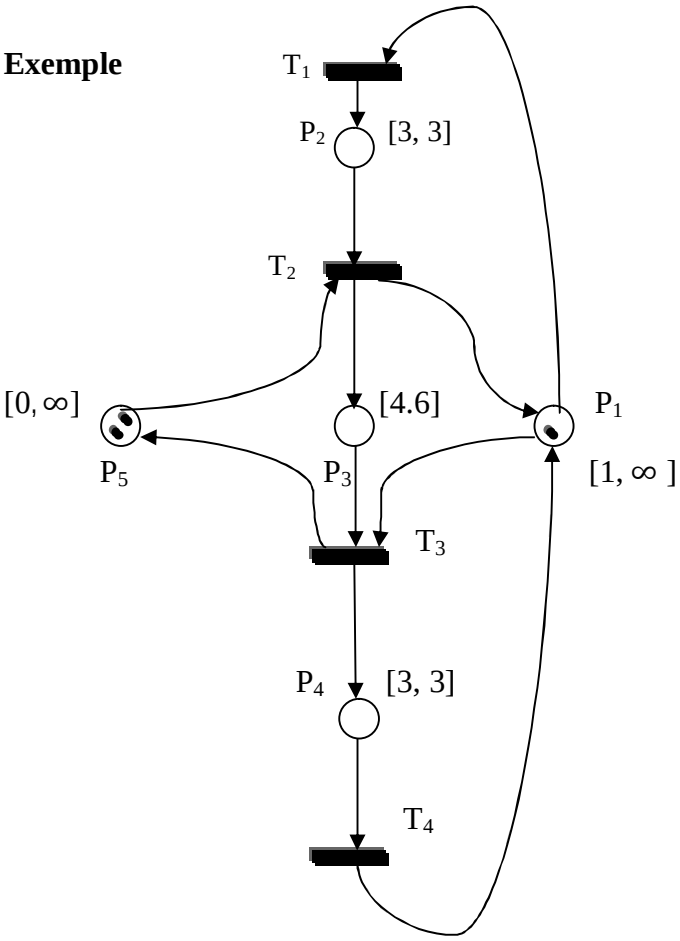



**Figure II.5** : Réseau de Petri p-temporel **[6].**

### II.5.3 Condition de franchissement d'une transition

Une transition $T_j$ est potentiellement franchissable à partir de l'état E= (M, G) si et seulement si :

- $T_j$ est validée au sens réseaux de Petri autonomes :

$$\forall p_i \in P : m(p_i) \geq Pré(P_i, T_i) ;$$

- $\forall P_i$ appartient à l'ensemble des places d'entrées de $T_j$, il existe au moins

Pré (Pi, $T_i$) marques dans cette place telle que :

$$\min(b_i, g_i^k) - \max(0, \max(a_i, g_i^k)) \neq 0;$$

**Où :**

k= 1, 2, 3, …, Pré ($P_i$, $T_i$).
[ai, bi] est l'intervalle statique associé à la place $P_i$.
$g_i^k$ est l'âge de la marque k dans la place $P_i$ **[6]**.

### II.5.3.1 Calcul de l'état suivant

Lorsque l'état E= (M, I) est atteint à instant donné, l'état E'= (M', I') est un état accessible. Le passage d'un état E= (M, I) à un état E'= (M', I') peut se faire de deux manières :

- Soit par l'écoulement de temps ;
- Soit par le franchissement d'une transition franchissable.

➢ **Définition 1 :**

L'état E'= (M', I') est un état accessible à partir de l'état E= (M, I) par l'écoulement de temps $\tau$ si et seulement si :

- M' = M.
- $\forall$ j une marque dans la place $P_i$, $a_i^{'j} = \max (a_i^j - \tau, 0)$ et $b_i^{'j} = b_i^j - \tau \geq 0$.

Où :

$a_i^j$ et $b_i^j$ (respectivement $a_i^{'j}$ et $b_i^{'j}$) représentent la borne inférieure et la borne supérieur de l'intervalle dynamique associé à la marque j dans la place $P_i$ depuis l'état E (respectivement E') **[6].**

➢ **Définition 2 :**

L'état E'= (M', I') est un état accessible à partir de l'état E= (M, I) par le franchissement d'une transition $T_j$ si et seulement si :

- La transition $T_j$ est franchissable à partir de l'état E
- $\forall$ p $\epsilon$ P M'(P)=M(P) - Pré ($P_i$, $T_j$) + Post ($P_i$, $T_j$)

Les marques qui ne bougent pas, gardent les mêmes intervalles dans E et E' (la durée de franchissement d'une transition considérée comme étant nulle).

Les marques nouvellement créées prennent les intervalles statiques associés à leurs nouvelles places d'accueil **[6]**.

## II.6 Les propriétés des réseaux de Petri p-temporels

### II.6.1 Bornitude et accessibilité

Un réseau de Petri p-temporel est borné pour un marquage initial $M_0$ si toutes les places sont bornées : il existe k $\in$n, tel que pour tout marquage accessible de $M_0$, le nombre de marques dans une place est inférieur ou égal à k.

Le réseau est k-borné si toutes ses places sont k-bornées.
La finitude du marquage et le problème d'accessibilité d'un marquage sont indécidables pour les réseaux de Petri p-temporel **[3].**

### II.6.2 Vivacité et blocage

Une transition $T_j$ d'un réseau de Petri p-temporel est vivante pour un marquage initial $M_0$ donné si : pour tout marquage $M_i$ accessible depuis $M_0$, il existe une séquence de franchissement (des transitions et des dates de franchissement associées) réalisable à partir de $M_i$ contenant $T_j$.

Un réseau de Petri p-temporel est vivant pour un marquage initial $M_0$ si toutes ses transitions sont vivantes pour $M_0$.

Un blocage est un marquage qui ne valide aucune transition.

L'une des principales particularités du modèle p-RdP réside dans le fait que certains jetons peuvent mourir au cours de l'évolution du réseau, c'est pourquoi de nouvelles propriétés ont été établies **[3]**.

### II.4.3 Vivacité des marques

Un état E = (M, I) d'un réseau de petri p-temporel est marques-vivantes si toutes les marques dans M sont vivantes.

Un réseau de Petri p-temporel est marques-vivantes pour un marquage initial $M_0$ (l'état initial $E_0$) si tous les marquages des états accessibles depuis $M_0$ sont des états marques-vivantes.
Si une marque d'un marquage d'un état accessible depuis E0 est morte, alors le réseau est marques-mortes **[3]**.

## II.7 Notion classes d'états

### II.7.1 Définition d'une classe d'état

Une classe d'état regroupe tous les états accessibles (partageant le même marquage) partant de l'état initial, en franchissant une séquence de transitions, vérifiant un ensemble de contraintes temporelles.

Les états d'une même classe auront le même marquage et le domaine de tir potentiel, on note

C = (M, D) où :

- M : est le marquage de la classe (le même marquage pour tous les états de la classe).
- D : est le domaine de tir potentiel (les intervalles de tir potentiels associés aux marques dans ce domaine) de la classe. Il est défini par l'union des intervalles associés à chaque état de la classe c-à-d. D =$\cup_{i=1}^{n} I'i$ . Où $I_i$ est le domaine de tir potentiel d'une marque i.

Le domaine D prendra la forme suivante :

$$\mathbf{D = \{\Theta / A.\Theta \geq b\}} \qquad \textbf{(}\text{II.4)}$$

Avec:

- A, une matrice de coefficients.
- b, un vecteur de constantes
- $\Theta$, un vecteur des variables dates de sortie des marques dans les places, $\Theta_i^j$ représentant la date de sortie de la marque j contenu dans la place $P_i$.

Chaque inégalité associée à une marque permet d'obtenir l'intervalle dynamique associé à cette dernière dans la classe considérée **[3]**.

### II.7.2 Condition de franchissement d'une transition depuis une classe d'état

Une transition $T_j$ est franchissable à partir de la classe C = (M, D) si et seulement si :

- $T_j$ est validée au sens des réseaux de Petri autonomes dans cette classe : $\forall$ $P_i$ appartient à l'ensemble des places d'entrées de $T_j$, il existe au moins Pré (Pi, $T_i$) inégalité dans le système des inégalités $\mathbf{A.\Theta \geq b}$.
- $T_j$ est potentiellement franchissable depuis la classe C : l'intervalle potentiel associé à la transition $T_j$ doit vérifier le système suivant :

$$\begin{cases} \mathbf{A.\theta \geq b} \\ \mathbf{a_i^k \leq \theta_{Tj} \leq b_i^k} \end{cases} \qquad \text{(II.5)}$$

Où $\theta_{Tj}$ est la date de franchissement de la transition $T_j$ et $[a_i^k, b_i^k]$ est l'intervalle dynamique associé à la marque k qui participe au franchissement de la transition $T_j$ dont sa place d'entrée est $P_i$. $a_i^k$ et $b_i^k$ sont calculés à partir des inégalités associées à la marque k dans le système (II.5).

De plus pour qu'aucune marque j dans les places d'entrées de la transition $T_j$ ne participe pas au franchissement de la transition $T_j$ ne meure pas, il faut rajouter au système (II.5) les inégalités : $\theta_{Tj} \leq \theta_i^j$. Le système devient alors :

$$\begin{cases} \mathbf{A}.\boldsymbol{\Theta} \geq \mathbf{b} \\ \mathbf{a_i^k} \leq \boldsymbol{\theta}_{\mathbf{Tj}} \leq \mathbf{b_i^k} \\ \boldsymbol{\theta}_{\mathbf{Tj}} \leq \boldsymbol{\theta}_{\mathbf{i}}^{\mathbf{j}} \end{cases} \qquad \text{(II.6)}$$

La transition $T_j$ doit être franchie dans l'intervalle d'intersection des intervalles dynamiques associés aux marques qui la valident. La dernière inégalité assure que le franchissement de la transition a lieu avant la mort des marques qui existent dans ses places d'entrée et qui ne participent pas au franchissement.

Si le système (II.6) n'a pas de solution alors la classe considérée est une classe marques mortes.

Enfin il faut vérifier que le franchissement de $T_j$ n'entraîne pas la mort des marques dans les places qui ne sont pas des places d'entrées pour $T_j$. Il faut alors ajouter au système (II.6), les inégalités suivantes : $\boldsymbol{\theta}_{\mathbf{Tj}} \leq \boldsymbol{\theta}_{\mathbf{1}}^{\mathbf{j}}$ où $P_1$ n'est pas une place d'entrée de la transition $T_j$.

Comme les relations peuvent exister entre les dates limites des jetons ne changeant pas de place lors de franchissement d'une transition du réseau, les inéquations de la forme $a_i^k \leq \theta_i^j \leq b_i^k$ ne sont pas suffisantes pour représenter le domaine de tir. Pour les couples de jetons vérifiant la caractéristique précédente, l'inégalité suivante doit aussi être considérée :

$$\boldsymbol{\theta}_{\mathbf{1}}^{\mathbf{p}} - \boldsymbol{\theta}_{\mathbf{k}}^{\mathbf{p}} \leq \boldsymbol{C_{pq}} \qquad \text{(II.7)}$$

Avec $\theta_k^p$ représente la date de sortie de la marque q contenue dans la place $P_k$ **[6]**.

### II.7.3 Algorithme de calcul de la classe suivante

Supposons que la transition $T_j$ soit franchissable à l'instant $\theta_{Tj}$ depuis une classe C = (M, D), avec D = {Θ/A.Θ ≥ b}.

La classe suivante C' = (M', D') est engendrée par le franchissement de $T_j$ à cet instant où :

- Le nouveau marquage M'est obtenu par :

$$\mathbf{M'(P_i) = M(P_i) + Post\ (P_i, T_j) - Pré\ (P_i, T_j)}$$

- Le nouveau domaine de tir D'est calculé à partir de D selon la procédure suivante :

(i) Ajouter au système D = {Θ/A.Θ ≥ b} les conditions de franchissement de $T_j$ : ajouter des inégalités qui attribuent à cette transition une variable $\theta_{Tj}$, instant de franchissement de cette transition. Ces inégalités auront la forme suivante :

$$\mathbf{a_i^k \leq \theta_{Tj} \leq b_i^k} \qquad \text{(II.8)}$$

$[a_i^k, b_i^k]$ est l'intervalle dynamique associé à la marque k qui participe au franchissement de la transition $T_j$. $a_i^k$ et $b_i^k$ sont calculés à partir des inégalités associées à la marque k dans le système D = {Θ/A.Θ ≥ b}.

Le système augmenté sera alors :

$$\begin{cases} \mathbf{A.\theta \geq b} \\ \mathbf{a_i^k \leq \theta_{Tj} \leq b_i^k} \end{cases} \qquad \text{(II.9)}$$

(ii) Exprimer que $\theta_{Tj}$ est inferieur ou égal à toutes les quantités $\theta_i^j$ associées aux autres marques (ne participant pas au franchissement de la transition) :

$$\mathbf{\theta_{Tj} \leq \theta_i^j} \qquad \text{(II.10)}$$

Pour chaque marque j qui ne participe pas au franchissement de $T_j$, il y a une inégalité. Le système d'inégalité deviendra :

$$\begin{cases} \mathbf{A.\theta \geq b} \\ \mathbf{a_i^k \leq \theta_{Tj} \leq b_i^k} \\ \mathbf{\theta_{Tj} \leq \theta_i^k} \end{cases} \qquad \text{(II.10)}$$

(iii) Effectuer pour toutes les marques qui ne participent pas au franchissement de $T_j$, le changement de variable suivant :

$$\mathbf{\theta_i^j = \theta_i'^j + \theta_{Tj}} \qquad \text{(II.11)}$$

(iv) Eliminer par substitution toutes les quantités $\theta_i^j$ associées aux marques participant au franchissement de la transition $T_j$ pour ne garder que les nouvelles quantités $\theta_i'^j$. On définit alors, les nouveaux domaines de tir potentiel des marques qui n'ont pas changé de places lors du franchissement de $T_j$.

(v) Compléter ce dernier système par une variable pour chaque marque nouvellement créée (c-à-d associé à ces marques les intervalles statiques de leurs places d'accueil **[6]**.

## I.8 Conclusion :

Dans ce chapitre nous sommes intéressés à la modélisation des systèmes dynamiques à contraintes de temps en utilisant les réseaux de Petri intégrant le temps.

Nous avons présenté brièvement les principaux modèles des réseaux de Petri qui permettent une intégration simple et efficace du temps.

# *Chapitre III :*

# *Utilisation des réseaux de Petri p-temporel pour la surveillance d'atelier*

# III.1 Introduction

**Motivation de l'utilisation des réseaux de Petri p-temporels**

Les réseaux de Petri temporisés ont été utilisés pour étudier les systèmes dont l'évolution dépend des contraintes de type temps de réponse, comme l'étude des protocoles de communication par exemple, d'après la règle de fonctionnement dans les réseaux de Petri temporisés on ne commence à compter le temps que si la transition est validée. Ainsi une marque peut rester une infinité de temps dans une place.

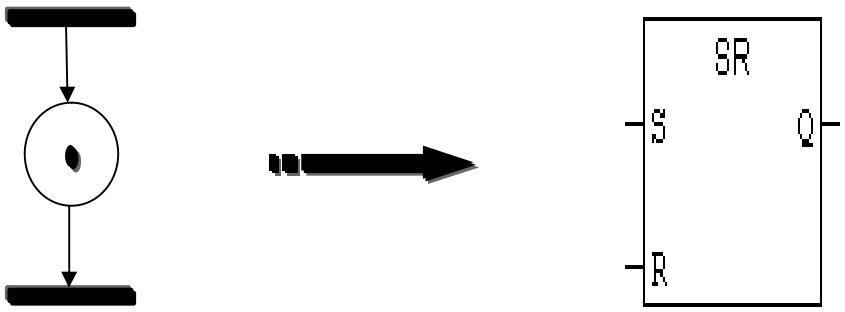
L'industrie fait toujours appel à des procédés où les temps de traitement des opérations doivent être compris entre deux durées, d'une part le temps minimum de traitement, d'autre part le temps maximum de traitement. Par exemple c'est le cas de l'industrie utilisant des réactions chimiques pour établir le traitement d'une pièce, cet exemple fait l'objet de notre application, il est claire que le non respect de ces contraintes peut avoir des conséquences graves sur la qualité de la production, c'est pourquoi nous utiliserons les réseaux de Petri p-temporels pour traduire l'obligation de respecter des temps de séjour.

# III.2 Mise en œuvre d'un réseaux de Petri p-temporel sous Step7

## III.2.1 Les places :

Des variables logiques sont associées aux places du réseau considéré. La signification de ces variables est le marquage ou non marquage de la place correspondante. Si la place est marquée la variable logique est égal à ''**un**'', sinon elle est égal à ''**zéro**'' dans le cas où la place est vide.

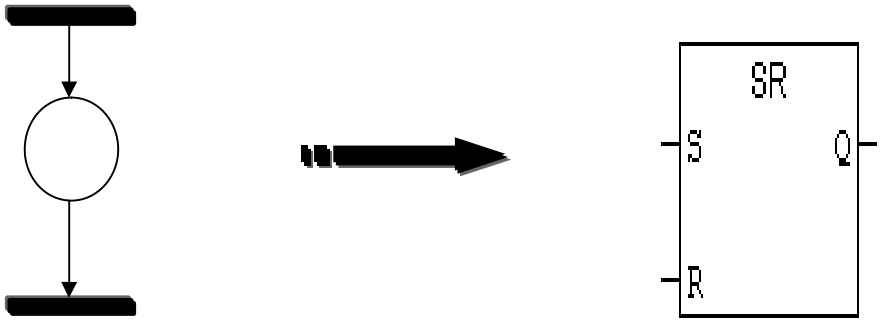
Le fait d'associer des variables logiques aux places on peut les représenter par des bascules **SR** (**S :** Set ''Mise à un'' et **R :** Reset ''Mise à zéro'' de la bascule). Chaque place est représentée par une bascule **SR** dont l'état logique représente le marquage ou non marquage de la place correspondante.
Cette variable d'état logique est la sortie d'une bascule SR(**S :** Set ''Mise à un'' et **R :** Reset ''Mise à zéro'' de la bascule) commandée par les entrées **S** et **R** et une horloge **H, S** et **R** vont véhiculer les conditions de franchissement d'une transition en amant de la place considéré.




Marquer une place

Mettre la bascule à un ''1''
« Q=1 »

**Figure III.1 :** Materialisation d'une place marquée**.**




Démarquer une place

Mettre la bascule à Zéro ''0''
« Q=0 »

**Figure III.2 :** Materialisation d'une place non marquée

**Remarque III.1 :**

Les figures III.1 et III.2 représentent la matérialisation des places mais sans temporisations.

### III.2.2 Materialisaton des places avec des temporisation (les intervalles de temps)

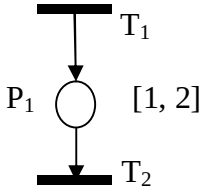
Pour inclure (ajouter) du temps dans la matéralisation précédente, il faut associer à chaque place un temporisateur ( dans notre application on a utiliser des conpteurs (temporisations ''**S_EVERZ''** sous forme de retard à la montée), qui fonctionnent comme suit :

1- Un compteur commance à compter dés que la place est marquée (l'arriver d'une marque dans une place).

2- Le compteur est remis à zero lorsque le jeton quitte la place.

3- Une variable logque x (la variable associer à la place) est vrai lorsque la temporisation est entre les deux bornes du l'intervalle (la borne inférieur et supérieur) de temps, elle est fausse en dehors.

**Exemple :**

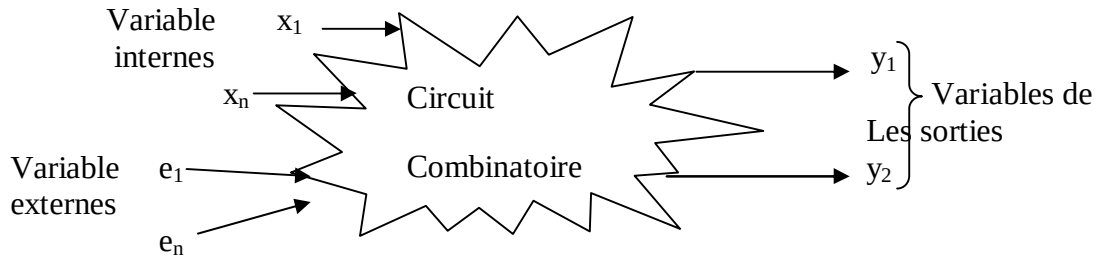



**X=1 si 1<= $Q_i$ <=2**

**X=0 si non**

### III.2.3 Les transitions :

Les transitions sont matérialisées par un circuit combinatoire comme suit :




Les transitions indiquent les possibilités d'évolutions du système, chaque transition est associée à une réceptivité.

La réceptivité est une condition logique et/ou un événement.
Une condition logique est une fonction booléenne de variables externes et de variables internes.
Une variable interne est un bit contenant le résultat de la comparaison d'un compteur et d'une valeur. Une variable externe est l'information venant d'un capteur ou d'un système extérieur (la fin de la temporisation par exemple).

Pour pouvoir commander en tout temps le franchissement d'une transition on préfère utiliser des bascules pour matérialiser les transitions.

Pour respecter les règles d'évolution d'un réseau de Petri p-temporel, il faut utiliser une bascule **SR** pour chaque transition de réseau de Petri (de même pour les places). La bascule qui matérialise une transition **Tj** est activée quand la place précédente et la transition sont vraies, sa sortie active la place suivante et désactive la précédente.

- Quand doit-on mettre à zéro la bascule représentant la transition ?

Une bonne solution est de le faire quand le franchissement a été effectué, c'est-à-dire quant la bascule suivante est activée et que la précédente ne l'est pas.

L'état du réseau de Petri correspond à une évolution de marquage, les marques qui matérialisent l'état du réseau à un instant donné, peuvent passer d'une place à l'autre par le franchissement d'une transition.

L'évolution dynamique d'un réseau de Petri p-temporel dépend des marques et de leurs situations temporelles, c'est-à-dire, le marquage et le démarquage des places dépendent des intervalles de temps associés aux places. Une marque dans la place participe à la validation de ses transitions de sortie que si elle a séjourné au moins la durée minimale dans cette place. Elle doit quitter la place, donc franchir l'une des transitions de sorties au plus tard quand la durée de séjour devient maximale. Après la durée maximale, la marque sera ''morte'' et ne participera plus à la validation des transitons.

De même pour les bascules qui matérialisent ces places, l'évolution dépend de l'activation « mise à un » et la désactivation « mise à zéro » des bascules.
L'activation de la bascule (présence dune marque dans la place), pendant au moins la durée minimale (le temps de séjour minimum d'une marque dans la place).
Elle doit être désactivé, donc activer l'une des bascules qui matérialisent les transitions de sorties au plus tard quand la durée devient maximale (la durée de séjour de cette marque devient maximale)

**Exemple illustratif :**

Soit le réseau de Petri p-temporel suivant :

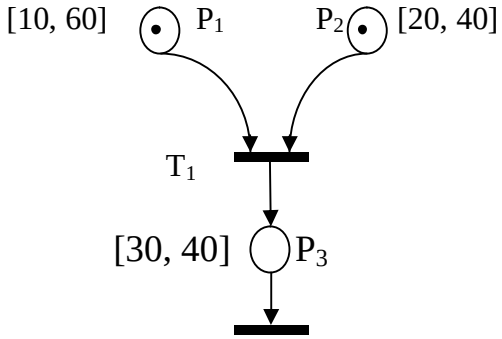
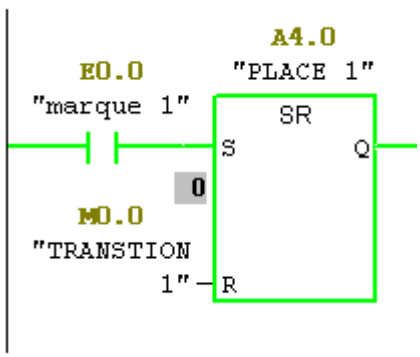
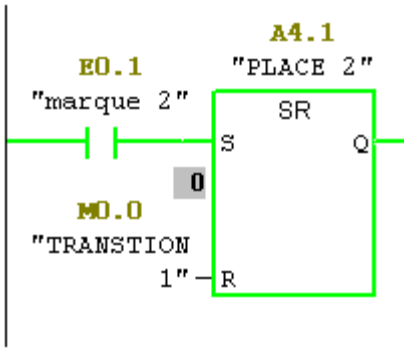
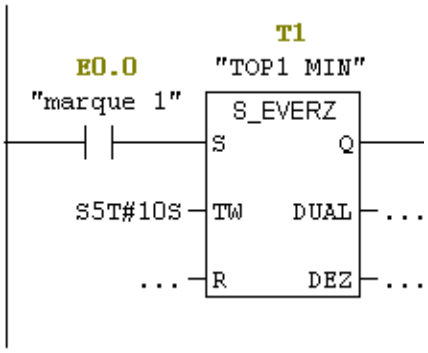


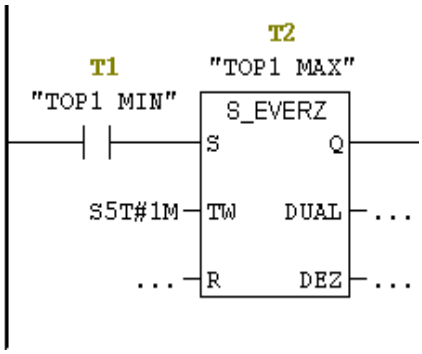

**Figure III.3 :** Exemple illustratif.

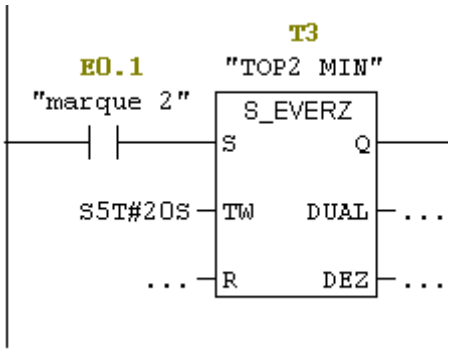
L'objet de cet exemple est de montrer comment mettre en œuvre un réseau de Petri p-temporel sous step7.
D'après les paragraphes précédents, on associé à chaque place et à chaque transition du réseau une bascule SR.
Les sorties des bascules qui matérialisent les places $P_1$ et $P_2$, initialement égal à un car ces dernières sont marquées. Elles ne participeront pas à l'activation de la transition qui matérialise la transition $T_1$ pendant au moins les durées minimales associées aux places (10 unités de temps pour $P_1$ et 20 pour $P_2$).

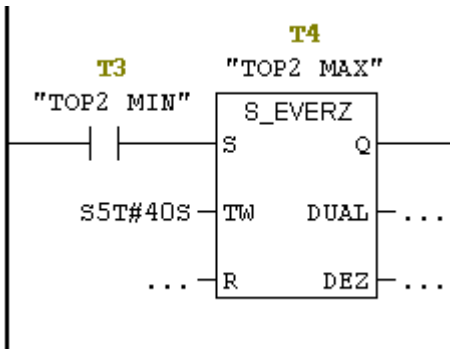


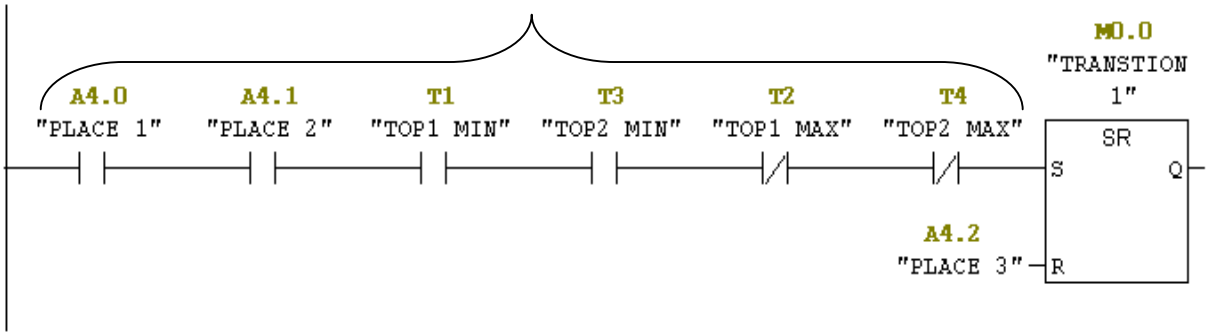


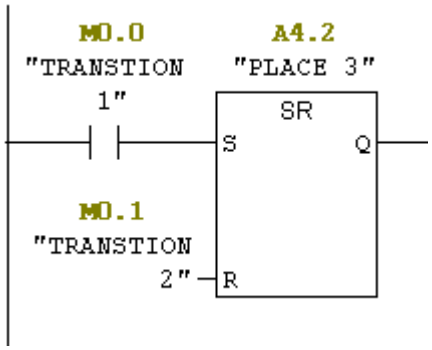



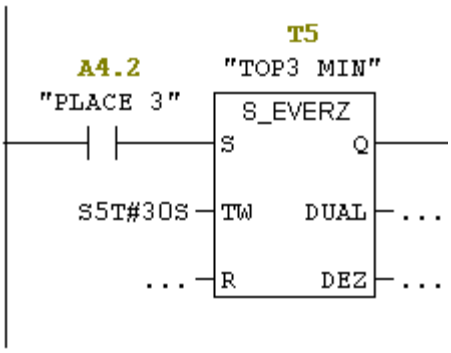



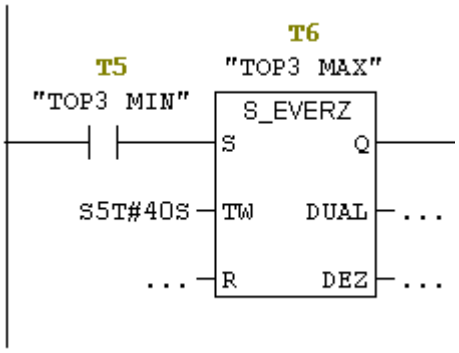




Réceptivité « 1 »

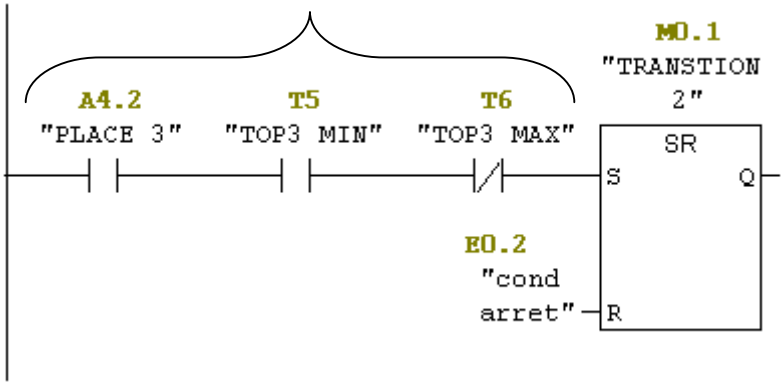





Réceptivité « 2 »




Les figures précédentes représentent le programme Step7 approprié (Programme LADDER) de l'exemple de la **Figure III.3**

## III.3 Application à un atelier de galvanoplastie

La galvanoplastie est une technique de traitement de surface qui consiste à déposer une fine couche de métal sur les pièces.

Nous illustrons nos développements à l'aide d'une cellule manufacturière dont la description fait l'objet du paragraphe suivant :

### III.3.1 Description :

La cellule considérée est une cellule de traitement assurant la coloration de pièces à l'aide de bains chimiques. Elle est constituée :

- De deux cuves identiques (de capacité unitaire) qui assurent le même traitement (coloration). Le temps de traitement (temps de trompe) d'une pièce doit durer entre 40 et 60 unités de temps.

- d'un robot (pont roulant) réalisant le chargement et le déchargement des deux cuves (bains chimiques). Après une opération de chargement ou de déchargement, le robot a besoin de 10 unités de temps pour être prêt pour l'opération suivante (chargement ou déchargement). Le temps de chargement est exactement 30 unités de temps, celui de déchargement est exactement 30 unités de temps, quelles que soient les cuves **[6].**

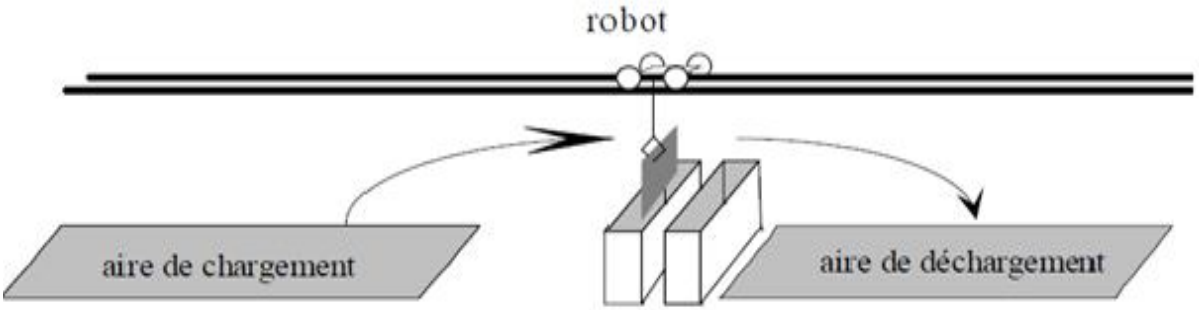



**Figure III.4 :** Une ligne de galvanoplastie (système : cellule de traitement).

### III.3.2 Modélisation :

Le modèle générale pour deux bacs, correspond à la figure III.5. Les places au milieu des axes correspondent aux trajets à vides du robot, l'axe à droite et à gauche dont les places en gris traduit les opérations de trempe (traitement) et les places en blanc traduisent le chargement et le déchargement.

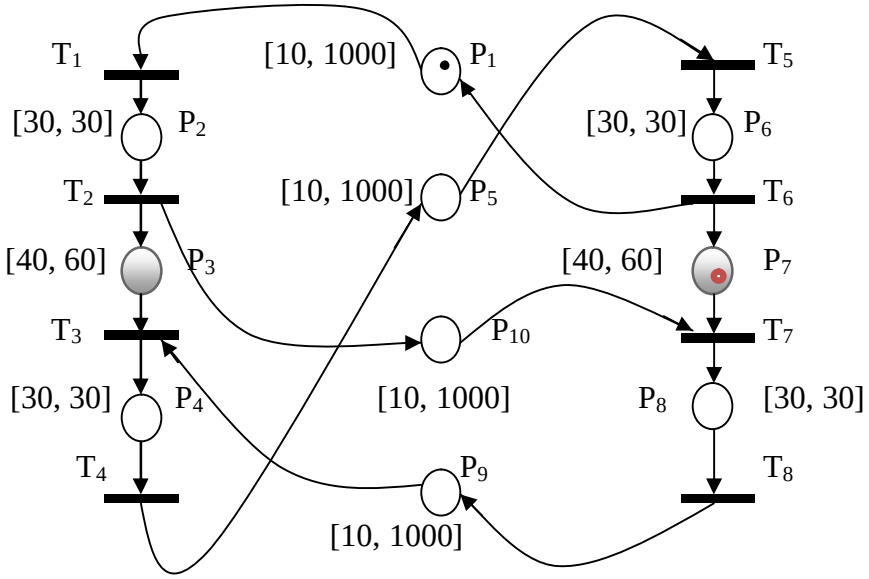



**Figure III.5 :** Le modèle réseau de Petri p-temporel de la cellule.

- $P_1$ Modélise le trajet à vide du robot (Cuve 2 vers l'aire de chargement) ;
- $P_2$ Modélise le trajet du robot en charge (Aire de chargement vers la cuve 1) ;
- $P_3$ Modélise le traitement de la pièce dans la cuve 1 ;
- $P_4$ Modélise le trajet du robot en charge (la cuve 1 vers l'aire de déchargement) ;
- $P_5$ Modélise le trajet à vide du robot (Aire de déchargement vers l'aire de chargement) ;
- $P_6$ Modélise le trajet du robot en charge (Aire de chargement vers la cuve 2) ;
- $P_7$ Modélise le traitement de la pièce dans la cuve 2 ;
- $P_8$ Modélise le trajet du robot en charge (Cuve 1 vers l'aire de déchargement) ;
- $P_9$ Modélise le trajet à vide du robot (Aire de déchargement vers la cuve 1) ;
- $P_{10}$ Modélise le trajet à vide du robot (Cuve 1 vers cuve 2, afin de récupérer la pièce traitée) ;

A partir des intervalles de temps associés aux places $P_1$, $P_5$, $P_9$ et $P_{10}$, nous savons que le robot a besoin de 10 unités de temps pour être prêt pour faire l'opération suivante (chargement ou déchargement des pièces). Par les intervalles associés aux places « $P_9$, $P_3$ »et « $P_{10}$, $P_7$ », nous modélisons le fait que le robot doit être libre au plus tard à la fin du temps de traitement (trempe) autorisé ([40, 60]) d'une pièce dans une cuve.

**Remarque III.2 :**

Afin d'avoir le bon fonctionnement (fonctionnement au plus tôt) dans notre système on a choisi la stratégie figure III.5 (le modèle de commande de la cellule).

Pour respecter les contraintes temporelles associées aux places, cette stratégie consiste à marquer initialement les places $P_1$ et $P_7$, ce qui veut dire qu'il y a une pièce en cours de traitement dans la cuve 2 (marque rouge) et que le robot prêt pour l'opération suivante (chargement de pièce dans la cuve 1).

## III.4 Le fonctionnement au plus tôt (fonctionnement au plus vite)

On appelle fonctionnement au plus tôt, le mode de fonctionnement suivant.

Toutes les transitions sont franchies dès que possible (c'est-à-dire dès qu'elles sont franchissables).

Le jeton qui arrive dans une place, peut contribuer au franchissement de la transition de sortie dès que le temps de séjour minimum se termine.

## III.5 Commande en boucle ouverte du l'atelier

Cette commande est décrite par la figure (III.5). On a régler les temps de traitements entre 40 et 60 unités de temps (place $P_3$, $P_7$), par contre, le déplacement du chariot (pont roulant) entre déférentes positions à besoin de 10 unités de temps a fin qu'il soit prêt pour faire l'opération suivante (chargement ou le déchargement des pièces).

A fin de respecter le cahier des charges de l'atelier c'est-à-dire que le robot exécute les opérations dans les délais imposés. Nous avons traduit ce modèle de commande figure (III.5) en programme Step7 (programmation utilisant des bascule et les temporisateurs), comme introduit précédemment (la mise en œuvre d'un réseau de Petri p-temporel sous Step7). Par ailleurs, la commande réalisée sous Step7 a été simulée sous SimFluid. Le logiciel SimFluid utilise des vérins et des distributeurs Pour simuler le fonctionnement d'un atelier.

On a représenté l'atelier à l'aide de vérins, par exemple un vérin à quatre positions pour représenter le pont roulant et deux vérins pour représenter les cuves.

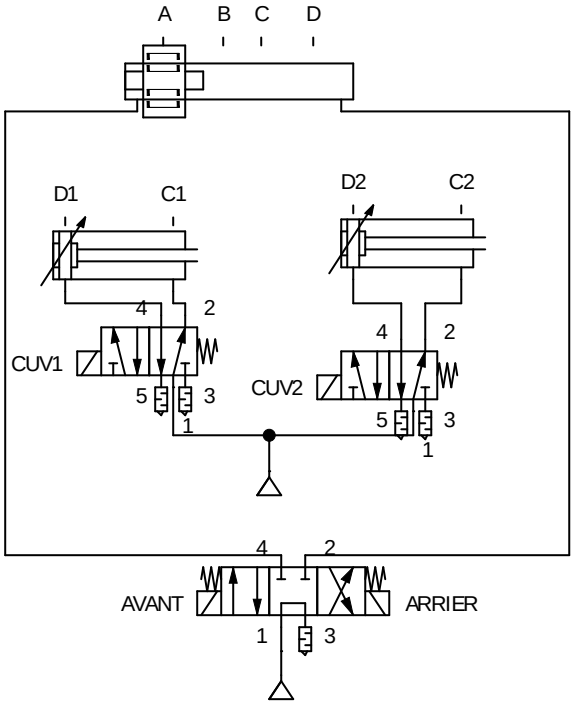



**Figure III.6** : Représentation de l'atelier sous **SimFuid.**

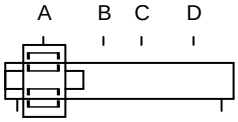



**Figure III.7** : Vérin à quatre positions qui représente le pont roulant.

On a représente le pont roulant avec un vérin à quatre positions :

- Position A : Lorsque le robot se positionne à la position A, cela veut dire que ce dernier est prêt à charge une pièce dans l'une des cuve, cette opération à besoin exactement 30 unités de temps.
- Position B : Quand le robot se trouve à la position B il sera prêt soit à mettre une pièce dans la cuve 1, ou récupère la pièce traitée (dans la cuve 1).

- Position C : A cette position le robot est soit prêt à mettre la pièce dans la cuve 2, ou récupère la pièce traitée (dans la cuve 2).
- Position D : Après avoir récupérer une pièce traité dans l'une des cuve, le robot se positionne à la position D pour le déchargement. Pour effectuer cette opération le robot à besoin exactement 30 unités de temps.

Le système contient deux cuves représentées par des vérins à deux positions Ci et Di.

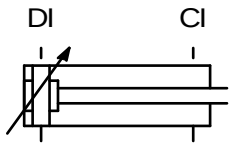



**Figure III.8 :** Vérin double effet à deux positions qui représente la cuve.

Quand le piston du vérin se trouve à la position Di, cela indique que la cuve est vide. Par contre s'il se trouve à la position Ci cela veut dire qu'il y a une pièce en cours de traitement dans la cuve.

La durée de traitement d'une pièce dans les cuves est entre 40 et 60 unités de temps (c'est-à-dire que la pièce est prête à être évacuer après 40 unités de temps mais elle sera défectueuse après 60 unités de temps).

Dans le cas ou il y a une pièce défectueuse on dira qu'il y a erreur de fonctionnement (par exemple le robot à une avance ou un retard). Pour éviter le dysfonctionnement du système on doit corriger cette erreur, et pour se faire on doit commander (contrôler) le système en boucle fermée (utilisation de correcteurs), en d'autre termes surveiller le système à chaque instant.

Pour cela on a utilisé la commande par des GDE p-temporels, la méthode consiste à utiliser les équations linéaires implicites afin de modéliser ces derniers ceci ferra l'objet de quatrième chapitre.

## III.6 Conclusion

Dans ce chapitre on a parlé sur la motivation de l'utilisation des réseaux de Petri p-temporels et leurs mises en œuvre sous Step7, dont on a expliqué comment matérialiser les places et les transitions par des bascules.

La fin de chapitre à été consacré à une application industrielle : un atelier de galvanoplastie. Dans un premier temps nous avons fait la description de l'atelier, sa modélisation (modèle réseau de Petri p-temporel figure (III.5) et comment on a programmé avec le Step7 (voir le programme annexe (I). Par la suite on a montré comment le modèle est simulé sous SimFluid.

# Chapitre IV :

# Commande d'un GDE P-temporel

## IV.1 Introduction

Le présent chapitre aborde la modélisation et la commande de GDEs P-temporels en utilisant des équations linéaires implicites.

## IV.2 Définition d'un graphe d'événement P-temporel

Un graphe d'événement P-temporel est défini par le doublet <R, IS> où :

R est un réseau de Petri marquée (place-transition)

IS: $P \rightarrow (R^+ \cup \{0\}) \times (R^+ \cup \{0, +\infty\})$

$P_i \rightarrow IS_i = [a_i, b_i]$ avec $0 \leq a_i \leq b_i$.

$[a_i, b_i]$ représente l'intervalle statique de temps de séjour d'une marque dans la place.

Le temps de séjour d'un jeton est compris dans un intervalle de temps. Le jeton qui arrive dans une place est indisponible pendant un certain temps. Il doit passer un temps de séjour minimum correspondant à la borne minimale de l'intervalle. Le jeton atteint un âge de maturité (qui lui permet de franchir une transition) après $a_i$ unités de temps de son arrivée dans la place. Il reste dans cet état de disponibilité durant $bi - a_i$ unités de temps.

Une des spécificités des réseaux de Petri p-temporels est la possibilité de la mort de jetons, le mode de fonctionnement est le suivant. Après un séjour de $b_i$ (borne max de l'intervalle) unités de temps dans la place, le jeton se trouve dans l'obligation de quitter cette dernière, sinon, il se retrouve dans un état de mort. Autrement dit, le jeton ne peut plus participer aux franchissements, et cela peut générer d'éventuels dysfonctionnements du système dans le futur (si par exemple, le jeton représente une ressource importante) **[7]**.

### IV.2.1 Définition : (Fonctionnement admissible)

Un fonctionnement est admissible pour un réseau de Petri P-temporel, lorsque sont évolution dynamique préserve la vivacité des jetons. En conséquence, il ne mène pas à un état de blocage **[7]**.

### IV.2.2 Définition d'un dateur :

Un dateur *x* est une application monotone (croissante) de $\mathbb{Z}$ dans $\mathbb{R} \cup \{-\infty, +\infty\}$.

Un dateur doit vérifier :

$$x(k+1) \geq x(k) \qquad \text{(IV.1)}$$

Nous associons à chaque transition $x_i$ un dateur $x_i(k)$, qui représente la date, à laquelle l'événement k a lieu **[7]**

## IV.3 Premières inégalités en dateur d'un GDE P-temporel

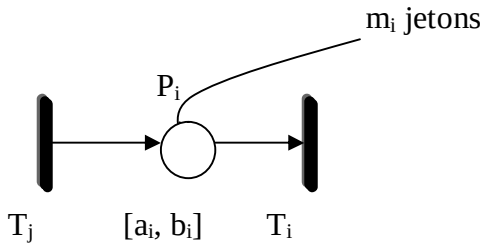



**Figure IV.1 :** Graphe d'événement P-temporel **[7]**.

Les inégalités suivantes représentent les relations entre les dates de franchissement.

- Pour la borne inférieur $a_i$, on a :

$$x_i(k) \geq x_j(k - m_i) + a_i \qquad \text{(IV.2)}$$

- Pour la borne supérieure $b_i$, nous avons :

$$x_i(k) \leq x_j(k - m_i) + b_i \qquad \text{(IV.3)}$$

## IV.4 Définition d'un système linéaire implicite

Si on associer à chaque transition du graphe une fonction dateur, l'évolution dynamique du système modélisé peut être représenté par une équation d'état implicite de la forme suivante :

$$\mathbf{E}\,\boldsymbol{x(k+1)} = \boldsymbol{A\,x(k)} + \boldsymbol{B\,u(k)} \qquad \text{(IV.4)}$$

Avec : $E \in Z^{qxn}$, $A \in Z^{qxn}$, $B \in Z^{qxn}$.

**Existence unicité de la solution**

Le système (IV.4) admet une solution pour tout $x_0$ et u si et seulement si :

$$\mathbf{rang(E, A, B) = rang\ (E)} \qquad \text{(IV.5)}$$

## IV.5 Obtention de la forme linéaire implicite à partir d'un GDE P-temporel

### IV.5.1 Motivation

**Exemple1 :** Soit le GDE P-temporel ci-dessous :

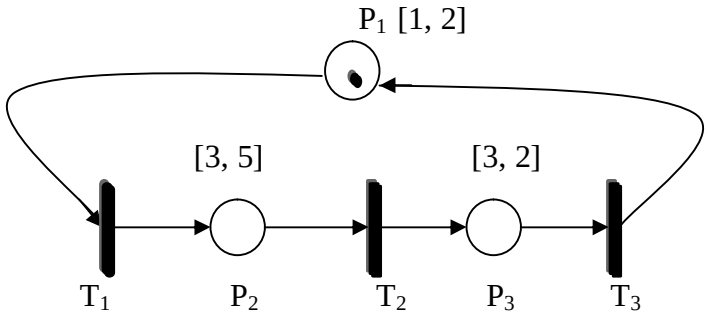



**Figure IV.2 :** Exemple d'un GDE P-temporel.

On associe à chaque transition $T_i$ un dateur $x_i$, avec i=1, …, 3, à chaque place $P_i$ est associée un temps de séjour de jetons noté $q_i$.

*$T_1 : x_1(k)$, $T_2 : x_2(k)$, $T_3 : x_3(k)$.*

*$P_1 : q_1(k)$, $P_2 : q_2(k)$, $P_3 : q_3(k)$, Avec* $a \leq qi(k) \leq b$.

Les équations aux dateurs sont obtenues comme suit :

$$x1(k) = x3(k-1) + q1(k)$$

$$x2(k) = x1(k) + q2(k)$$

$$x3(k) = x2(k) + q3(k)$$

Avec :

$$1 \leq q1(k) \leq 2$$

$$3 \leq q2(k) \leq 5$$

$$3 \leq q3(k) \leq 2$$

En mettant les équations précédentes sous forme matricielle nous obtenons :

$$\begin{bmatrix} 1 & 0 & 0 \\ -1 & 1 & 0 \\ 0 & -1 & 1 \end{bmatrix} \begin{bmatrix} x1(k) \\ x2(k) \\ x3(k) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x1(k-1) \\ x2(k-1) \\ x3(k-1) \end{bmatrix} + \begin{bmatrix} q1(k) \\ q2(k) \\ q3(k) \end{bmatrix}$$

**Exemple 2**

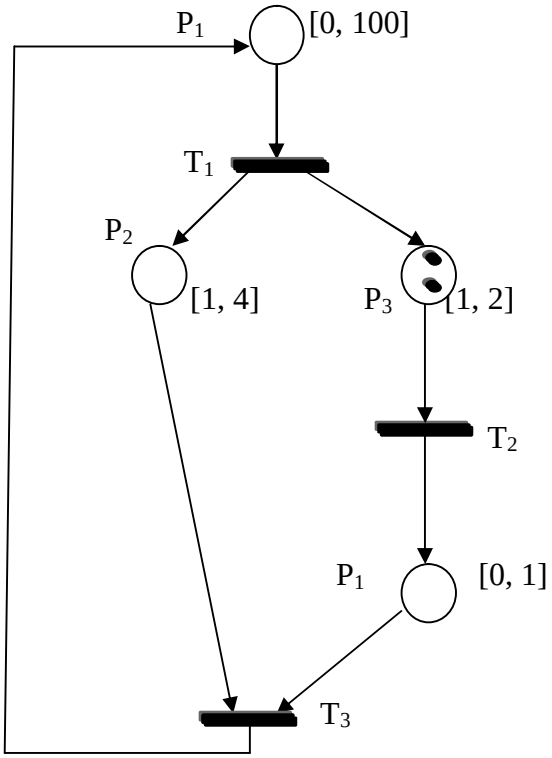



**Figure IV.3 :** Exemple d'un GDE P-temporel

Les équations aux dateurs sont obtenues comme suit :

$$x1(k) = x4(k-1) + q1(k)$$

$$x2(k) = x1(k-1) + q2(k)$$

$$x3(k) = x1(k-1) + q3(k)$$

$$x4(k) = x3(k) + q4(k)$$

$$x4(k) = x1(k) + q5(k)$$

Avec :

$$0 \leq q1(k) \leq 100$$

$$0 \leq q2(k) \leq 0$$

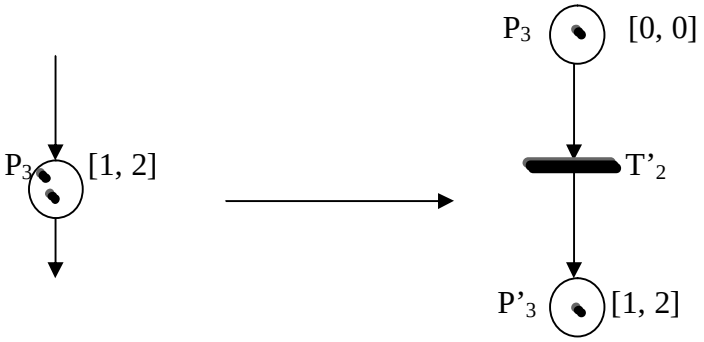
$$1 \leq q3(k) \leq 2$$

$$0 \leq q4(k) \leq 1$$

$$1 \leq q1(k) \leq 4$$

**Remarque IV.1 :**

Afin d'avoir une équation d'état (Exemple IV.3), on a modifié la place $P_3$ comme suit :




Les équations aux dateurs ci-dessus nous donnent la forme matricielle suivante :

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & -1 & 1 \\ -1 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x1(k) \\ x2(k) \\ x3(k) \\ x4(k) \end{bmatrix} = \begin{bmatrix} 0 & 0 & 0 & 1 \\ 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x1(k-1) \\ x2(k-1) \\ x3(k-1) \\ x4(k-1) \end{bmatrix} + \begin{bmatrix} q1(k) \\ q2(k) \\ q3(k) \\ q4(k) \end{bmatrix}$$

La forme obtenue est une équation linéaire implicite.

On note en générale la modélisation d'un GDE P-temporel par une équation linéaire implicite dont la représentation est donnée comme suit :

$$\begin{cases} \mathbf{E}\, \boldsymbol{x}(\boldsymbol{k}+\mathbf{1}) = \boldsymbol{A}\, \boldsymbol{x}(\boldsymbol{k}) + \boldsymbol{q}(\boldsymbol{k}+\mathbf{1}) \\ \mathbf{a} \leq \mathbf{q}(\mathbf{k}) \leq \mathbf{b} \end{cases} \qquad \text{(IV.6)}$$

Avec :

k : Le nombre de franchissement des transitions.

$x(k)$ **:** La date de k$^{éme}$ franchissement.

q(k) : Le temps de séjour d'un jeton dans une place pour le k$^{ème}$ franchissement.

a et b sont respectivement des vecteurs représentant les bornes min et max de temps de séjour.

## IV.6 commande en boucle fermée de GDE P-temporels

Dans cette section, nous allons nous intéresser à la commande des systèmes à contraintes de temps de séjour. De façon classique, la commande d'un système à contraintes de séjour modélisé par un GDE P-temporel est le pilotage des entrées en vue d'obtention un comportement désiré.

Le problème traité dans cette partie consiste à calculer à chaque étape (k) une commande q(k) en fonction de l'état des dateurs de telle sorte à converger asymptotiquement vers une trajectoire souhaitée (désirée) z(k), k $\epsilon$ **N**.

**Objectif :**

L'objectif du problème traité dans cette section consiste à calculer à chaque étape (k) une commande q(k) en fonction de l'état des dateurs de telle sorte à converger asymptotiquement vers une trajectoire désirée z(k). Pour se faire on doit chercher un retour d'état **F** tel que la solution x(k) de l'équation linéaire implicite (IV.6) tend asymptotiquement vers une trajectoire z(k) désirée, solution du système d'équations linéaires implicites suivantes :

$$\mathbf{E}\, z(k+1) = A\, z(k) + qz \qquad \text{(IV.7)}$$

La trajectoire désirée z(k) est donnée par un fonctionnement 1-périodique.

C'est-à-dire :

$$\mathbf{z}(k+1) = z(k) + \lambda\, u \qquad \text{(IV.8)}$$

Où :

- $\lambda$ est le temps de cycle désiré.
- u est un vecteur unitaire de dimension appropriée.

Le fonctionnement 1-périodique constitue une sous-classe importante des fonctionnements réalisables des GDE. Dans ce mode du fonctionnement, une période fixée sépare deux franchissements successifs des transitions.

Le système d'erreur induit par ces deux équation (IV.7) et (IV.8) est donné par :

$$\mathbf{E}\, \mathbf{z}(k+1) - E\, x(k+1) = E\, e(k+1) \qquad \text{(IV.9)}$$

$$A\,z(k) + qz - A\,x(k) + q(k+1) = E\,e(k+1)$$

$$\Rightarrow E\,e(k+1) = A\,(z(k) - x(k)) + qz + q(k+1)$$

On obtient :

$$\mathbf{E\,e(k+1) = A\,e(k) + qz + q(k+1)} \qquad \text{(IV.10)}$$

Avec :

$$\mathbf{e(k) = z(k) - x(k)} \qquad \text{(IV.11)}$$

Pour un tel système (IV.10) on peut définir des entrées q (k+1) données par un retour d'état statique de la forme :

$$\mathbf{q(k+1) = F\,e(k) + qz} \qquad \text{(IV.12)}$$

Si on remplace q (k+1) dans l'équation (IV.10) on obtient :

$$E\,e(k+1) = A\,e(k) + qz - F\,e(k) - qz$$

$$\mathbf{E\,e(k+1) = (A - F)\,e(k)} \qquad \text{(IV.13)}$$

Notre problème consiste à chercher un feed back F tel que :

1- Le système admet une solution.
2- Le système poursuivre la trajectoire désirée.

## IV.7 Application au système de galvanoplastie

L'objet de cette application est de trouver un moyen de commander (contrôler) un système dynamique dont les états changent en fonction de temps modélisé par des équations linéaires implicites, en utilisant les méthodes suivantes :

1- la poursuite de la trajectoire
2- l'étude de la stabilité.

On a proposé un programme Matlab afin d'optimiser le temps de cycle et les dates de franchissements pour un comportement désirée z(k) vérifiant un comportement 1-périodique.

$$\mathbf{z(k) = z(k-1) + \lambda max\ u}$$

### IV.7.1 Description du fonctionnement

Le système considérée est une cellule de traitement de surface, coloration de pièce à l'aide de bains chimique, la cellule est constituée de :

- De deux cuves identiques (de capacité unitaire) qui assurent le même traitement (coloration). Le temps de traitement (temps de trompe) d'une pièce doit durée entre 40 et 60 unités de temps.

- d'un robot (pont roulant) réalisant le chargement et le déchargement des deux cuves (bains chimiques). Après une opération de chargement ou de déchargement, le robot a besoin de 10 unités de temps pour être prêt pour l'opération suivante (chargement ou déchargement). Le temps de chargement est exactement 30 unités de temps, celui de déchargement est exactement 30 unités de temps, quelles que soient les cuves.

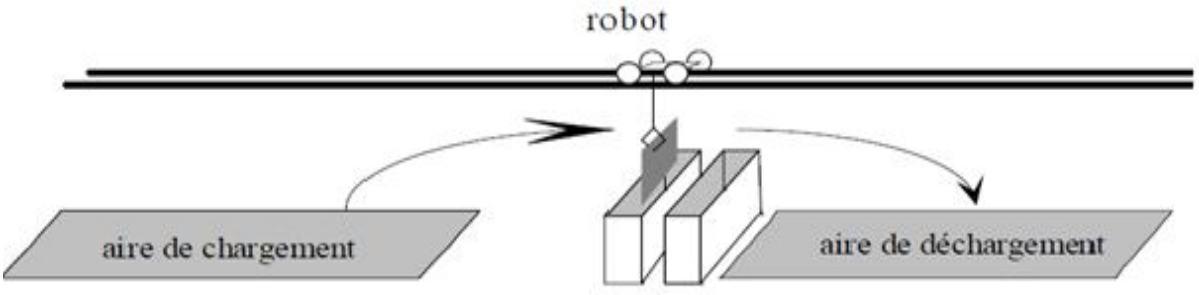



**Figure IV.4 :** Une ligne galvanoplastie (système : cellule de traitement)

### IV.7.2 Modélisation

La cellule est modélisée par le GDE P-temporel suivant :

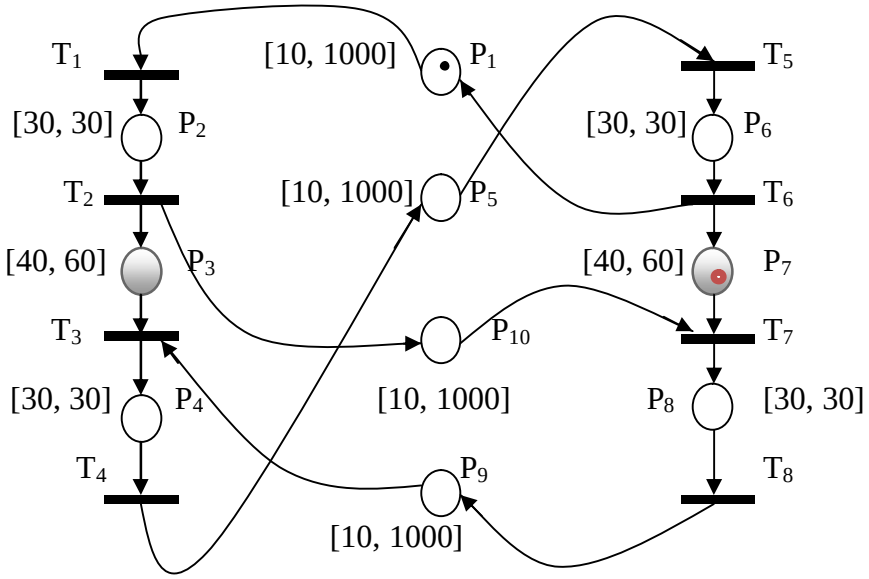



**Figure IV.5 :** La cellule modélisée par un GDE p-temporel.

Le modèle générale pour deux bacs, correspond à la figure (IV.5). Les places au milieu des axes correspondent aux trajets à vides du robot, l'axe à droite et à gauche dont les places en gris traduit les opérations de trempe (traitement) et les places en blanc traduisent le chargement et le déchargement.

- $P_1$ Modélise le trajet à vide du robot (Cuve 2 vers l'aire de chargement) ;
- $P_2$ Modélise le trajet du robot en charge (Aire de chargement vers la cuve 1) ;
- $P_3$ Modélise le traitement de la pièce dans la cuve 1 ;
- $P_4$ Modélise le trajet du robot en charge (la cuve 1 vers l'aire de déchargement) ;
- $P_5$ Modélise le trajet à vide du robot (Aire de déchargement vers l'aire de chargement) ;
- $P_6$ t Modélise le trajet du robot en charge (Aire de chargement vers la cuve 2) ;
- $P_7$ Modélise le traitement de la pièce dans la cuve 2 ;
- $P_8$ Modélise le trajet du robot en charge (Cuve 1 vers l'aire de déchargement) ;
- $P_9$ Modélise le trajet à vide du robot (Aire de déchargement vers la cuve 1) ;
- $P_{10}$ Modélise le trajet à vide du robot (Cuve 1 vers cuve 2, afin de récupérer la pièce traitée) ;

A partir des intervalles de temps associés aux places $P_1$, $P_5$, $P_9$ et $P_{10}$, nous savons que le robot a besoin de 10 unités de temps pour être prêt pour faire l'opération suivante (chargement ou déchargement des pièces). Par les intervalles associés aux places « $P_9$, $P_3$ »et « $P_{10}$, $P_7$ », nous modélisons le fait que le robot doit être libre au plus tard à la fin du temps de traitement (trompe) autorisé ([40, 60]) d'une pièce dans une cuve.

**Remarque IV.2 :**

Afin d'avoir un bon fonctionnement pour notre système on a choisi la stratégie de la figure (IV.5) (le modèle de commande de la cellule).

Pour respecter les contraintes temporelles associées aux places, cette stratégie consiste à marquer initialement les places $P_1$ et $P_7$, ce qui veut dire qu'il y a une pièce en cours de traitement dans la cuve 2 et que le robot soit prêt pour l'opération suivante (chargement de pièce dans la cuve 1).

### IV.7.3 Optimisation des dates de franchissement

Nous associons à chaque transition du GDE P-temporel $T_i$ un dateur $x_i(k)$, qui représente la date, à laquelle l'événement k a lieu (le moment exacte de franchissement de la transition).

L'évolution peut être représentée par les équations suivantes :

$$\begin{cases} x1(k) = x6(k-1) + q1(k) \\ x2(k) = x1(k) + q2(k) \\ x3(k) = x2(k) + q3(k) \\ x4(k) = x3(k) + q4(k) \\ x5(k) = x4(k) + q5(k) \\ x6(k) = x5(k) + q6(k) \\ x7(k) = x6(k-1) + q7(k) \\ x8(k) = x7(k) + q8(k) \\ x3(k) = x8(k) + q9(k) \\ x7(k) = x2(k) + q10(k) \end{cases} \Rightarrow \begin{cases} x1(k+1) = x6(k) + q1(k+1) \\ x2(k+1) - x1(k+1) = q2(k+1) \\ x3(k+1) - x2(k+1) = q3(k+1) \\ x4(k+1) - x3(k+1) = q4(k+1) \\ x5(k+1) - x4(k+1) = q5(k+1) \\ x6(k+1) - x5(k+1) = q6(k+1) \\ x7(k+1) = x6(k) + q7(k+1) \\ x8(k+1) - x7(k+1) = q8(k+1) \\ x3(k+1) - x8(k+1) = q9(k+1) \\ x7(k+1) - x2(k+1) = q10(k+1) \end{cases}$$

On remarque que le système d'équation est sous la forme du système linéaire implicite suivant :

$$\begin{cases} \mathbf{E}\,\boldsymbol{x}(\boldsymbol{k}+\mathbf{1}) = \boldsymbol{A}\,\boldsymbol{x}(\boldsymbol{k}) + \boldsymbol{q}(\boldsymbol{k}+\mathbf{1}) \\ \mathbf{a} \le \mathbf{q}(\mathbf{k}) \le \mathbf{b} \end{cases}$$

Tel que :

$$E = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ -1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & -1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & -1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & -1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & -1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & -1 & 1 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & -1 \\ 0 & -1 & 0 & 0 & 0 & 0 & 1 & 0 \end{pmatrix}$$

Et

$$A = \begin{pmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

Afin d'obtenir un comportement (comportement désiré), la méthode consiste à chercher une commande q(k) en fonction de l'état des dateurs d'une manière à converger asymptotiquement vers une trajectoire Z(k) désiré.

Solution de système linéaire implicite suivant :

$$\mathbf{E}\, \boldsymbol{z(k+1)} = \boldsymbol{A\, z(k)} + \boldsymbol{qz}$$

La trajectoire désirée Z(k) est donnée par un fonctionnement 1-périodique.

C'est-à-dire :

$$\mathbf{z(k+1)} = \boldsymbol{z(k)} + \boldsymbol{\lambda\, u}$$

Avec :

- λ est le temps de cycle désiré.
- u est le vecteur unitaire.

De l'équation (IV.8)

$$\boldsymbol{E}\, \mathbf{z(k+1)} = \boldsymbol{E\, z(k)} + \boldsymbol{E\, \lambda\, u} \qquad \text{(IV.8')}$$

De (IV.7) et (IV.8') on aura :

$$\boldsymbol{A\, z(k)} + \boldsymbol{qz} = \boldsymbol{E\, z(k)} + \boldsymbol{E\, \lambda\, u}$$

$$\boldsymbol{E\, z(k)} - \boldsymbol{A\, z(k)} = \boldsymbol{E\, \lambda\, u} - \boldsymbol{qz}$$

$$\begin{cases} (\boldsymbol{E}-\boldsymbol{A})\,\boldsymbol{z}(\boldsymbol{k}) + \boldsymbol{E}\,\boldsymbol{\lambda}\,\boldsymbol{u} - \boldsymbol{qz} = \mathbf{0} \\ \mathbf{a} \leq \mathbf{qz} \leq \mathbf{b} \\ \boldsymbol{\lambda} \geq \mathbf{0} \\ \boldsymbol{z} \geq \mathbf{0} \end{cases}$$

On utilisant la programmation linéaire on a utilisé Matlab pour optimiser (minimiser, maximiser) les dates de franchissement et le temps de cycle.

Pour ce programme on a choisit :

$\min f(x) = \lambda$. Comme fonction objectif.

Sous contraintes de type inégalité suivantes :

$qz \leq b$

$-qz \leq -a$

$-\lambda \leq 0$

$-z \leq 0$

Après l'exécution de programme Matlab, on a obtenu les résultats suivants :

- Le temps de cycle est $\lambda_{min}$ = 160 unités de temps.
- Les dates de franchissement :

z = (18.0823, 48.0823, 98.0823, 128.0823, 138.0823, 168.0823, 58.0823, 88.0823)$^T$.

### IV.7.4 Etude de la stabilité

Pour étudier la stabilité, on utilise le résultat de la proposition suivante :

Le système implicite : $\boldsymbol{E}\,\mathbf{x}(\boldsymbol{k}+\mathbf{1}) = \boldsymbol{A}\,\boldsymbol{x}(\boldsymbol{k})$ à une solution si et seulement si :

$\exists\, \boldsymbol{Q} \in \mathbf{R}^{\mathbf{nxn}}$ tel que $\mathbf{E\,Q = A}$

Pour l'équation $\mathbf{E\,e(k+1) = (A-F)\,e(k)}$ .

On cherche une matrice carrée Q tel que :

$$\mathbf{E\,Q = (A-F)} \qquad \text{(IV.14)}$$

La solution de système est donnée par :

$$\mathbf{e(k+1) = Q\ e(k)}$$

On dit que le système est stable si les valeurs absolues des valeurs propres de la matrice Q sont inférieures ou égales à 1.

Dans notre application, pour étudier la stabilité du système. On utilise le résultat de la proposition précédente.

C'est-à-dire :

On procède à la recherche des deux matrice Q et F tel que l'équation E Q = A- F soit satisfaite.

Dont, F est le retour d'état trouvé par un placement de pôle, rechercher pour que le système en boucle fermée soit asymptotiquement stable (le système est stable si les valeurs absolues des valeurs propres de la matrice Q sont inférieures ou égales à 1).

Après l'exécution du programme Matlab la matrice F obtenue est la suivante :

$$F = \begin{pmatrix} -0.0400 & 0.0000 & 0 & -0.0000 & -0.0000 & 1.0000 & -0.0000 & -0.0000 \\ 0.0400 & -0.0100 & 0.0000 & -0.0000 & 0.0000 & 0.0000 & -0.0000 & -0.0000 \\ 0.0000 & 0.0100 & -0.0800 & 0.0000 & -0.0000 & -0.0000 & 0.0000 & -0.0000 \\ -0.0000 & 0.0000 & 0.0800 & -0.0500 & 0.0000 & 0.0000 & -0.0000 & 0.0000 \\ 0.0000 & -0.0000 & 0.0000 & 0.0500 & -0.0250 & -0.0000 & 0.0000 & -0.0000 \\ -0.0000 & -0.0000 & -0.0000 & 0.0000 & 0.0250 & -0.0110 & -0.0000 & 0.0000 \\ 0.0000 & -0.0000 & 0.0000 & 0.0000 & 0.0000 & 1.0000 & -0.0090 & -0.0000 \\ 0.0000 & -0.0000 & -0.0000 & 0.0000 & -0.0000 & -0.0000 & 0.0090 & -0.0020 \\ 0 & -0.0000 & -0.0800 & 0.0000 & 0.0000 & -0.0000 & 0.0000 & 0.0020 \\ -0.0000 & 0.0100 & -0.0000 & 0.0000 & 0.0000 & -0.0000 & -0.0090 & 0 \end{pmatrix}$$

Et l'erreur est calculée comme suit :

$$E\,e(k+1) = (A-F)\,e(k)$$

Après l'exécution de programme Matlab pour 10 itérations

$$e = \begin{pmatrix} 10.0000 & 0.4000 & 0.0160 & 0.0006 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0 \\ 40.000 & 0.4000 & 0.0040 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0 \\ 90.000 & 7.2000 & 0.5760 & 0.0461 & 0.0037 & 0.0003 & 0.0000 & 0.0000 & 0.0000 & 0.0 \\ 120.00 & 6.0000 & 0.3000 & 0.0150 & 0.0008 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0 \\ 130.00 & 3.2500 & 0.0813 & 0.0020 & 0.0001 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0 \\ 160.00 & 1.7600 & 0.0194 & 0.0002 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0 \\ 50.000 & 0.4500 & 0.0041 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0 \\ 80.000 & 0.1600 & 0.0003 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0000 & 0.0 \end{pmatrix}$$

On remarque que l'erreur tend vers zéro après la $6^{eme}$ itération.

Calcul de la matrice carrée Q :

On a :

$E\,Q = A\text{-}F \rightarrow Q = (E^{-1} * A) - (E^{-1} * F)$.

Avec : $E^{-1} = (E^{T} * E)^{-1} * E^{T}$.

$E^{G}$ est une inverse à gauche de E.

D'où :

$$Q = \begin{pmatrix} 0.0400 & 0.0000 & -0.0000 & 0.0000 & 0.0000 & -0.0000 & -0.0000 & -0.0000 \\ 0.0000 & 0.0100 & -0.0000 & 0.0000 & -0.0000 & -0.0000 & -0.0000 & 0.0000 \\ -0.0000 & 0.0000 & 0.0800 & -0.0000 & -0.0000 & 0.0000 & -0.0000 & 0.0000 \\ 0.0000 & -0.0000 & -0.0000 & 0.0500 & -0.0000 & 0.0000 & -0.0000 & -0.0000 \\ 0.0000 & -0.0000 & -0.0000 & 0.0000 & 0.0250 & 0.0000 & -0.0000 & 0.0000 \\ 0.0000 & 0.0000 & -0.0000 & -0.0000 & -0.0000 & 0.0110 & -0.0000 & 0.0000 \\ 0.0000 & -0.0000 & -0.0000 & 0.0000 & -0.0000 & 0.0000 & 0.0090 & 0.0000 \\ -0.0000 & -0.0000 & -0.0000 & -0.0000 & -0.0000 & 0.0000 & -0.0000 & 0.0020 \end{pmatrix}$$

Après l'exécution du programme Matlab on a abouti à des résultats souhaités :

- le retour d'état calculé est un retour d'état stabilisant.
- L'erreur après la 6$^{émé}$ itération tend vers zéro.

La figure ci-dessous représente les erreurs du système. On remarque que cette erreur tend vers zéro après la 6$^{éme}$ itérations.

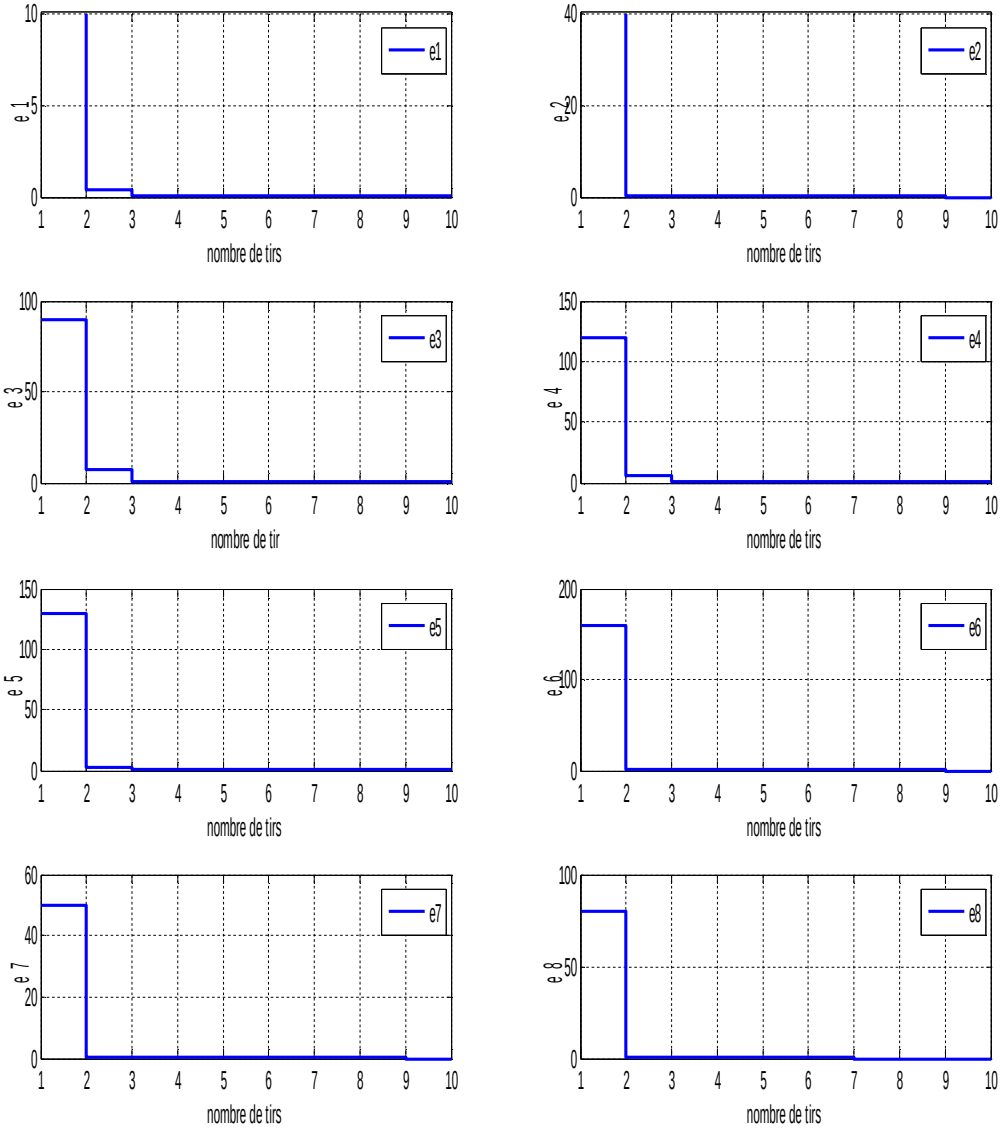



**Figure IV.6 :** Contrôle des erreurs.

La figure ci-dessous représente le temps de séjour des jetons dans les places. On remarque bien que cette valeur devient stable après un moment donné.

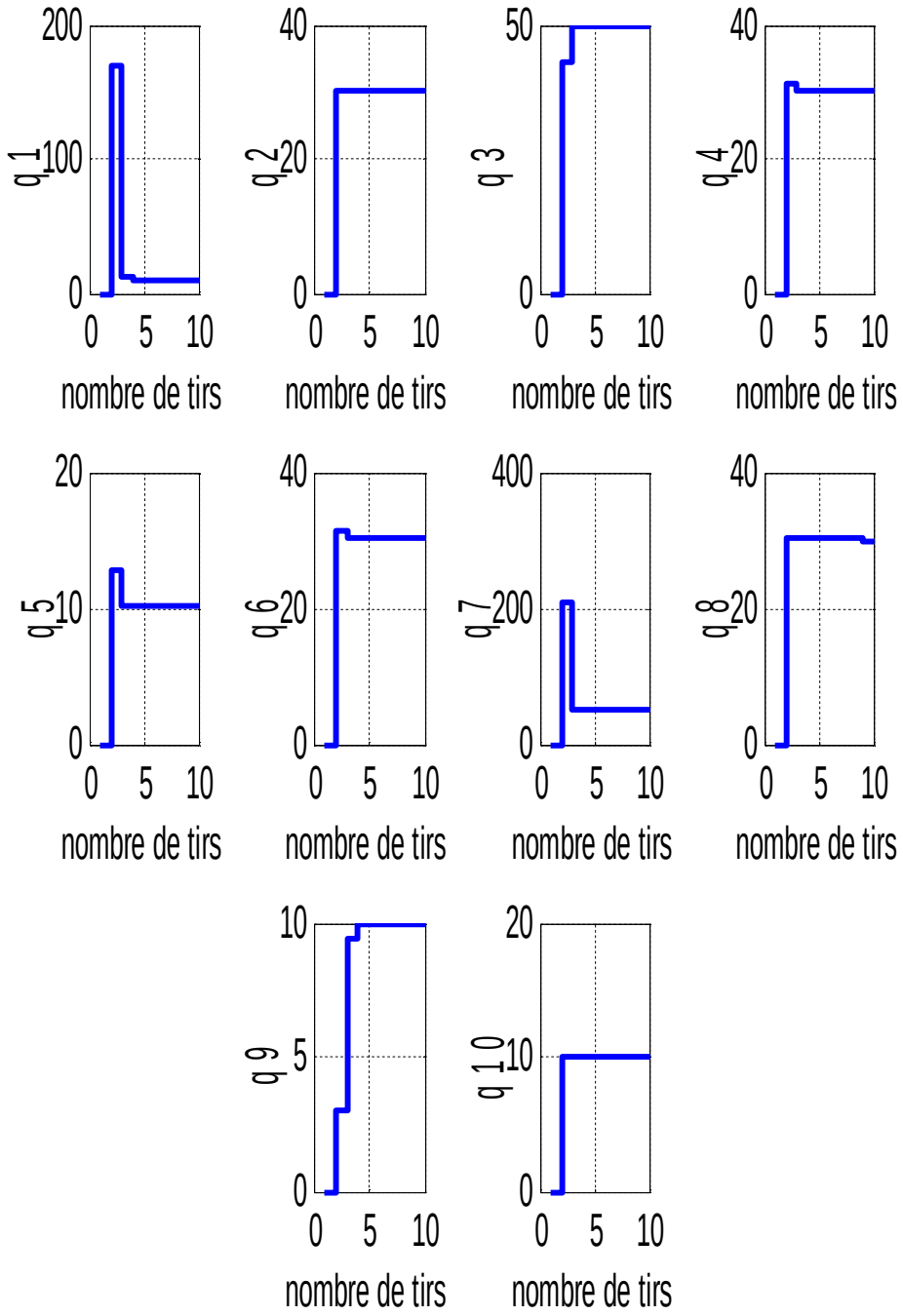



**Figure IV.7 :** Evolution temps de séjour en fonction du nombre de tirs

## IV.8 Commande en temps réel de l'atelier

Après avoir obtenu des résultats intéressants avec Matlab, la commande en temps réel de l'atelier consiste à implanter ce dernier sous Step7. Pour se faire on doit réaliser la liaison entre Matlab et Step7. En d'autre terme, les résultats de calcul sous Matlab doivent être injectée dans un bloc (correcteur) dans le Step7.

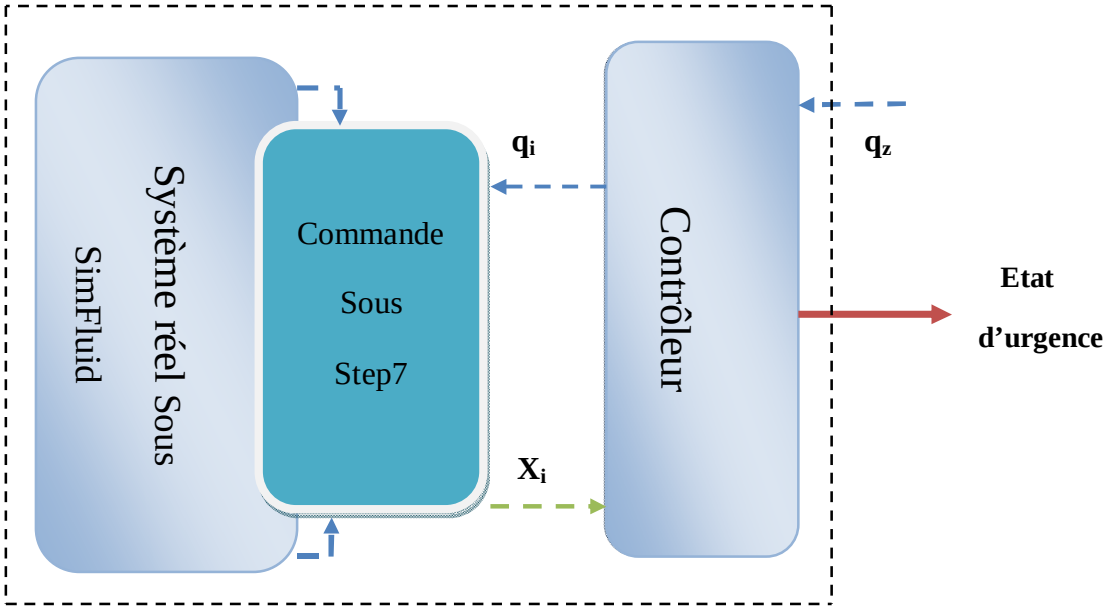



**Figure IV.8 :** Schéma de commande.

### IV.8.1 Objectif de l'application

Matlab/Simulink est le logiciel utilisé pour modéliser et simuler les systèmes étudiés. Simulink fournit un rédacteur graphique qui permet à l'utilisateur de créer des processus réel et développer des solutions. Les blocs réalisés dans Simulink peuvent être compilé en code C/C++. WinAC ODK (Open Development Kit) permet d'exécuter le code C/C++ dans l'environnement de Windows en temps réel de WinAC RTX. WinAC Target permet l'intégration facile des modèles Simulink dans le projet Step7. L'objectif de l'application est l'intégration du programme de correction (correcteur) réalisé avec Matlab dans le Projet Step7 **[8]**.

### IV.8.2 Contenu principal de cette application

- Création d'un bloc dans Simulink (contrôleur).
- Compilation du bloc en code C/C++ utilisant WinAC.
- Intégration du bloc dans le projet Step7.

Matlab/Simulink fournit l'option pour compiler les modèles de Simulink en code C/C++. Ce code peut être intégrer dans un projet de WinAC ODK, appeler et exécuter dans le projet Step7 par l'intermédiaire des bibliothèques DLL/RTDLL crées par le logiciel IntervalZero **[8]**.

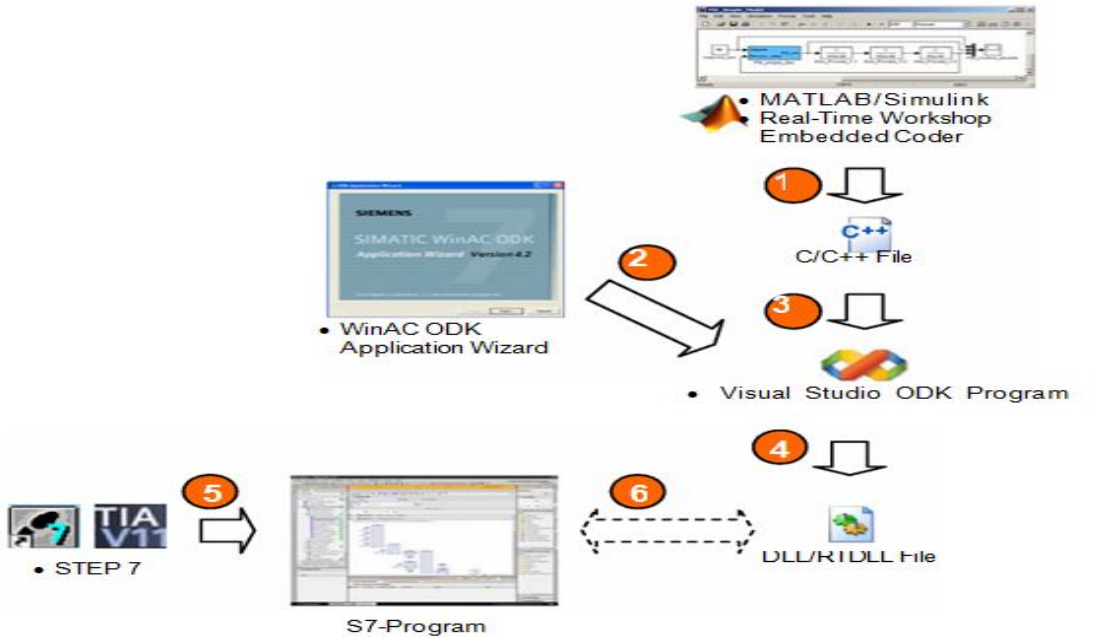



**Figure IV.9 :** Les étapes à suivre pour réaliser la commande **[8]**.

**Tableau IV.1.**

| Etapes | Instructions |
|---|---|
| 1 | Création d'un bloc (contrôleur) utilisant Matlab/Simulink, compiler le bloc en Code C/C++ utilisant Real-Time Workshop Embedded Coder. |
| 2 | WinAC ODK doit être utilisé pour créer le projet ODK. |
| 3 | L'intégration manuelle de Projet Matlab/Simulink dans le projet ODK |
| 4 | Création des bibliothèques DLL/RTDLL utilisant IntervalZero. |
| 5 | Création de Projet Step7. |
| 6 | L'appel des bibliothèques DLL/RTDLL doit être programmé dans S7 programme. |

### IV.8.3 Logiciels utilisés dans cette application

- **Matlab V8 (R2012 b) :** Matlab est utilisé pour résoudre des problèmes mathématiques et pour montrer graphiquement les résultats dans Matlab, la programmation est faite dans un langage de programmation. Il offre une grande portée des fonctions mathématiques.

- **Simulink V8.2 :** Simulink est utilisé pour la programmation graphique des modèles.

- **WinAC Target :** WinAC Target c'est un logiciel qu'on ajoute pour Simulink pour compiler les modèles pour WinAC RTX.

- **Microsoft Visual Studio 2008/2010 :** Visual studio est environnement de développement avec des langages de haut niveau tels que C, C++…

- **Step7 V5.5 :** Step7 est l'environnement de développement pour programmer les contrôleurs programmables logiques.

- **SIMATIC WinAC RTX et WinAC ODK :** WinAC RTX est une version logicielle d'un automate S7 pour lequel une commande en temps réel est mise à la disposition du système d'exploitation Windows par un système en temps réel. Il exécute des programmes utilisateur STEP 7 comme le font d'autres automates S7 et permet une intégration aisée dans STEP 7 et des applications Windows standard. WinAC RTX s'exécute dans deux environnements distincts : des processus s'exécutant dans le système en temps réel et des processus s'exécutant dans l'environnement Windows.

  - Les processus qui s'exécutent dans le système en temps réel exécutent le programme utilisateur STEP 7 pour WinAC RTX, donnant à la commande du processus la priorité la plus élevée.
  - Les processus qui s'exécutent dans l'environnement Windows traitent toutes les autres opérations, telles que la communication et les interfaces avec les systèmes et applications Windows.

- **IntervalZero SDK V9.1.2** : IntervalZero offre un environnement en temps réel pour les systèmes de PC. Il crée des bibliothèques RTDLL en utilisant le Visual studio.

## IV.9 Conclusion

Dans ce chapitre on a vu la modélisation des systèmes dynamiques par des GDE P-temporels, pour cela on a introduit les équations linéaires implicites afin de modéliser ces derniers.

La modélisation des GDE p-temporels par les équations linéaires implicites nous a permis d'arriver à une méthode pour commander (contrôler) les systèmes à contraintes de temps de séjour.

Vue les problèmes qu'on a rencontré dans notre réalisation (commande en temps réel) et l'indisponibilité du logiciel (IntervalZero SDK V9.1.2) permettant d'effectuer la liaison entre Matlab et Step7, on n'a pas peut réaliser la liaison entre Step7 et Matlab pour notre application.

# *Conclusion générale*

## Conclusion générale

L'objectif de ce travail à été de répondre à la problématique suivante : Modélisation, Commande et Surveillance d'atelier à contraintes de temps de séjours (Atelier de galvanoplastie). (La problématique consiste à la surveillance d'une ligne de traitement de surface).

Pour des raisons de simulation, on a été amené à modéliser l'atelier (la ligne de traitement) afin de procéder aux différents testes possibles.

Les réseaux de Petri plus précisément les réseaux de Petri p-temporels étant l'outil de cette modélisation qui permet de décrire fidèlement le comportement dynamique du système étudié. Par la suite, on a procédé à la matérialisation de ce modèle par des bascules (SR) pour approcher la solution de ce système décrit par des GDE p-temporels à la programmation logique (Programme LADDER).

Après avoir procédé à la modélisation et la matérialisation, nous avons simulé le modèle en boucle ouverte avec les logiciels Step7 et SimFluid, l'utilisation de ce dernier nous a permis de représenter l'atelier à l'aide des vérins.

Le premier test consiste à programmer sous Matlab, le problème consiste à optimiser les dates de franchissements et le temps de cycle d'un GDE p-temporel et commander ce dernier en utilisant les méthodes classiques suivantes :

- L'étude de la stabilité.
- La poursuite de la trajectoire.

Et pour se faire on a cherché un retour d'état qui nous permet de répondre à de telles exigences. Les résultats de cette simulation montrent que le comportement désiré est atteint et que les erreurs du système tendent vers zéro après un certain temps et le système lui même devient stable.

Le deuxième test consiste à commander la ligne de traitement (système étudié) en boucle fermée et en temps réel, pour réaliser cette commande nous faisant appel à une liaison entre Matlab et Step7. En d'autre terme, les résultats de calcul sous Matlab (Feedback) doivent être injectés dans le programme fait sous Step7 sous forme d'un bloc (correcteur).

Vue les problèmes qu'on a rencontré dans cette réalisation (commande en temps réel) et l'indisponibilité du logiciel (IntervalZero SDK V9.1.2) permettant de créer les bibliothèques DLL/RTDLL à fin d'effectuer la liaison Step7 et Matlab.

# Bibliographie

**[1]** **Annie Choquet-Geniet. « LES RSEAUX DE PETRI,** *Un outil de modélisation* **»,** Paris, 2006.

**[2]** **R. KARA.** Cours de productique ; Département d'automatique, UMMTO, 2013.

**[3]** **Patrice BONHOMME,** « *RESEAUX DE PETRI P-TEMPORELS : CONTRIBUTION A LA COMMANDE ROBUSTE* », Thèse préparée au sein du Laboratoire d'Automatique et de Micro-informatique Industrielle (LAMII/CESALP) de l'Ecole Supérieure d'Ingénieurs d'Annecy 2001.

**[4]** **Redouane KARA, «** *Contribution à l'Analyse Quantitative et à la Commande des Réseaux de petri continus à Arcs Valués. Application aux Systèmes de Production Manufacturière* », Thèse de doctorat, UMMTO 2009.

**[5]** **François DEFOSSEZ,** « *Modélisation discrète et formelle des exigences temporelles pour la validation et l'évaluation de la sécurité ferroviaire* », Thèse de doctorat, École Centrale de Lille 2010.

**[6]** **Wael KHANSA,** « *RESEAUX DE PETRI P-TEMPORELS CONTRIBUTION A L'ETUDE DES SYSTEMES A EVENEMENTS DISCRETS* », Thèse de doctorat, Ecole Supérieure d'Ingénieurs d'Annecy 1997.

**[7]** **Abdelhak GUEZZI, «** *Modélisation, analyse de performances et commande des systèmes à événement discrets »*, Thèse de doctorat, Ecole doctorale STIM
Sciences et technologies de l'information et de mathématiques 2010.

**[8]** WinAC Target for MATLAB/Simulink: Integrating and calling Simulink models using STEP 7 and WinAC ODK using PID control as an example WinAC RTX, STEP 7, WinAC ODK
Application  April 2013

# *Annexes*

Dans cette annexe, une présentation des différents outils utilisés pour la programmation et la simulation de notre application, pour la programmation on a choisi le langage Step7. Ce dernier a l'avantage d'exploiter d'autre logiciels et outils de simulation comme SimFluid.

## I.1 Automate programmable Industriel (API) :

### I.1.1 Définition :

Un automate programmable industriel (API) est un système électronique destiné à commander une installation industrielle en utilisant des fonctions logiques séquentielles ou numériques.

### I.1.2 Structure de l'automate programmable :

Les sous ensembles fondamentaux composant un automate programmable sont :

- ✓ L'unité centrale : qui traite les variables en fonction du traitement logique programmé en mémoire et élabore les ordres de commande ;
  La tête de bac qui assure la liaison entre l'unité centrale et les interfaces.
- ✓ Les interfaces de l'entrée : qui reçoivent les données machines provenant des capteurs.
- ✓ Les interfaces de sortie : qui appliquent les processus de commande.

## I.2 Langage de Programmation (Step7)

Le Step7 est un logiciel destiné à la programmation des automates Siemens. Il permet de concevoir, configurer, programmer, tester, mettre en service et maintenir les systèmes d'automatisation SIMATIC.

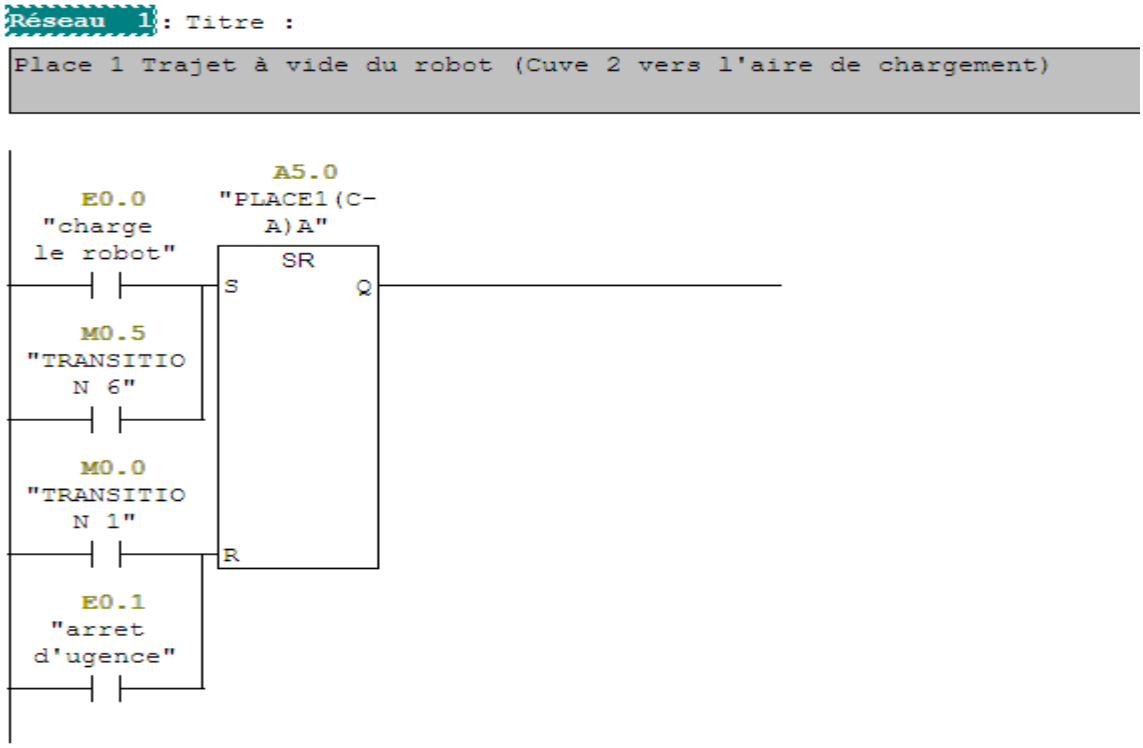
- ➢ STEP 7 intègre en particulier les outils suivants :

  - Tous les langages de programmation pour automates programmables définis dans le standard CEI 61131-3: schémas contact, logigrammes, listes d'instructions, graphes séquentiels (S7-GRAPH) et langages structurés (S7-SCL).
  - Le logiciel de simulation automate S7-PLCSIM pour la mise au point de programmes sans disposer des automates cible
  - Outil de configuration graphique des composants matériels et des réseaux de Communication.

➢ **S7-PLCSIM**

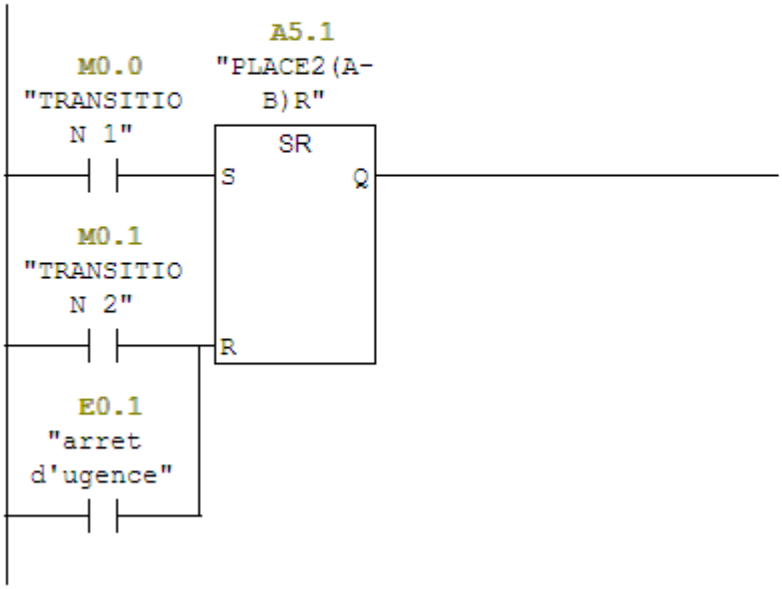
- Dans S7-PLCSIM, le programme utilisateur STEP 7 est exécuté et sera essayé dans un automate programmable simulé. La simulation étant réalisée entièrement dans le logiciel STEP7, l'utilisateur n'a pas besoin de matériel S7.
- Avec *S7-PLCSIM*, la simulation des programmes utilisateur *STEP7* qui ont été développés pour les automates S7-300 et S7-400.
  - *S7-PLCSIM* offre une interface simple au programme utilisateur *STEP7* servant à visualiser et à modifier différents objets tels que les variables d'entrée et de sortie.
  - *S7-PLCSIM* offre une interface utilisateur graphique permettant de visualiser et de modifier les variables de ces programmes, d'exécuter en mode Cycle unique ou Cycle continu le programme du système cible simulé ou de modifier l'état de fonctionnement de l'automate simulé.
  - *S7-PLCSIM* comprend également un objet COM appelé *S7ProSim* pour accéder par programme à un système cible simulé.

## I.3 Le Programme développé pour notre application



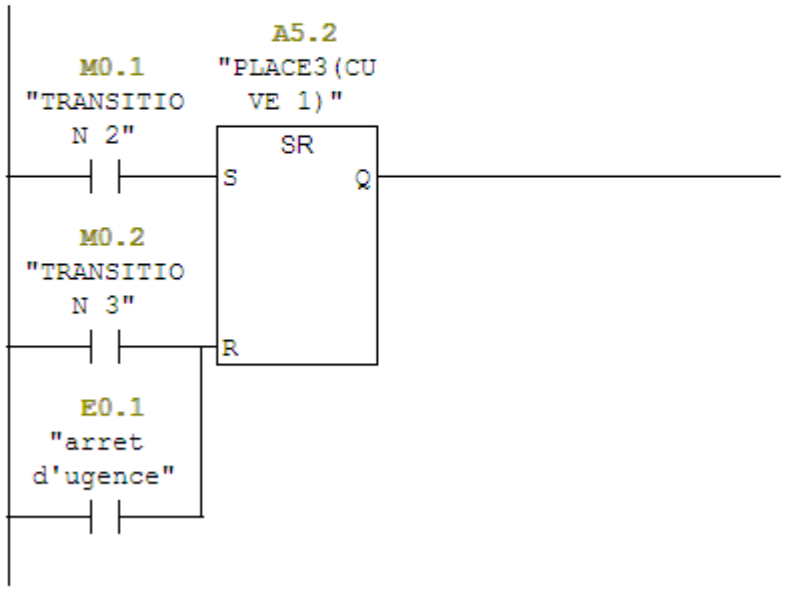
Réseau 2 : Titre :

Place 2 Trajet du robot en charge (Aire de chargement vers la cuve 1)




Réseau 3 : Titre :

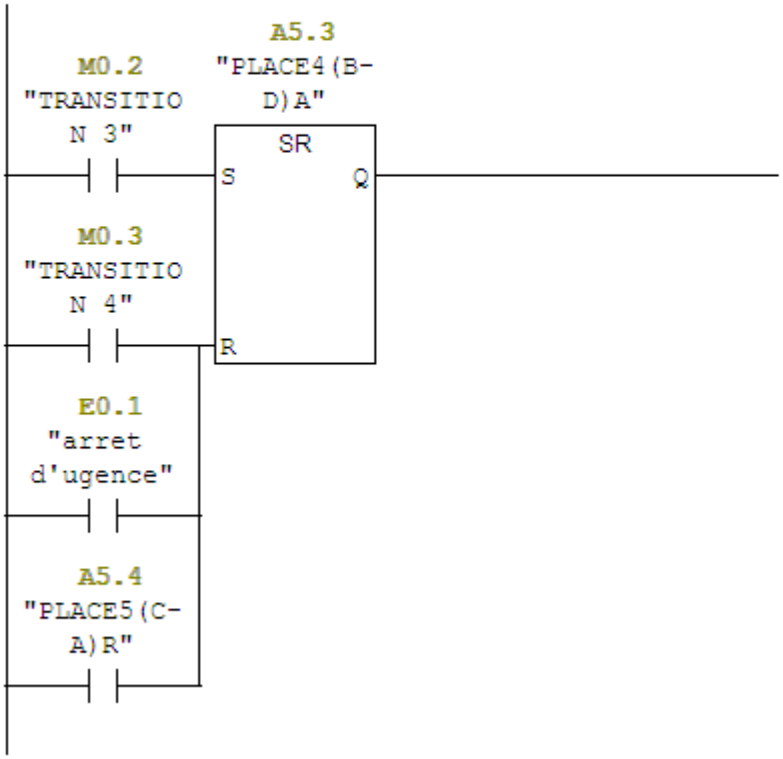
Place 3 Traitement de la pièce dans la cuve 1



# Annexe (I)

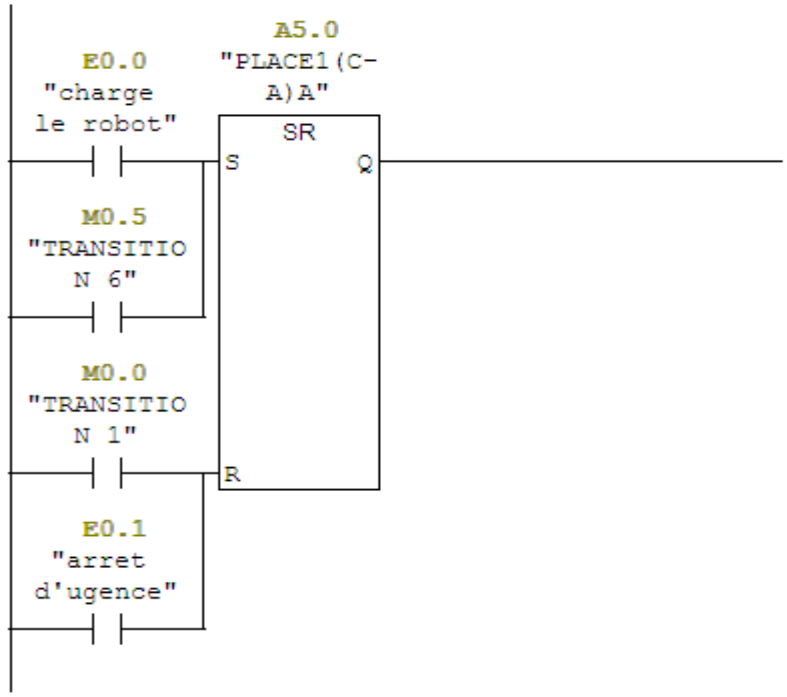
Réseau 4 : Titre :

Place 4 Trajet du robot en charge (la cuve 1 vers l'aire de déchargement)



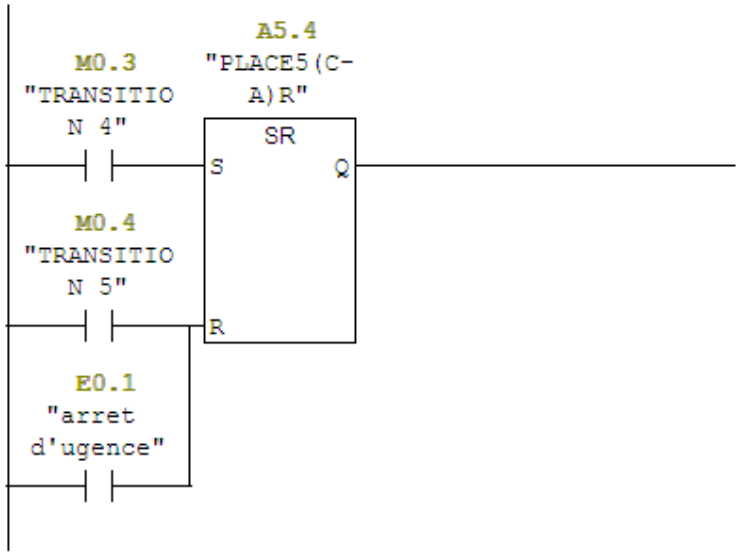

Réseau 1 : Titre :

Place 1 Trajet à vide du robot (Cuve 2 vers l'aire de chargement)



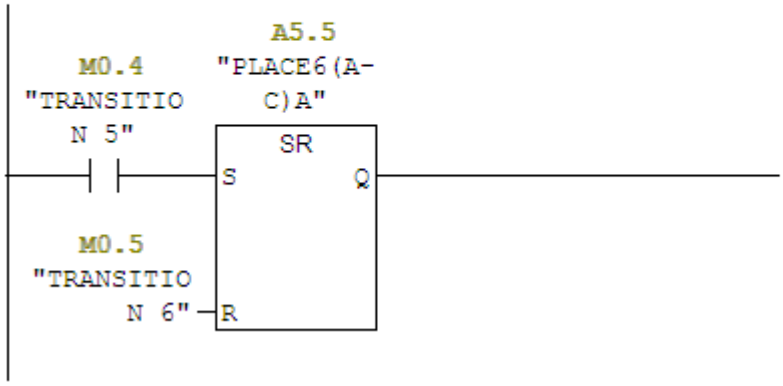
Réseau 5 : Titre :

Place 5 Trajet à vide du robot (Aire de déchargement vers l'aire de chargement)



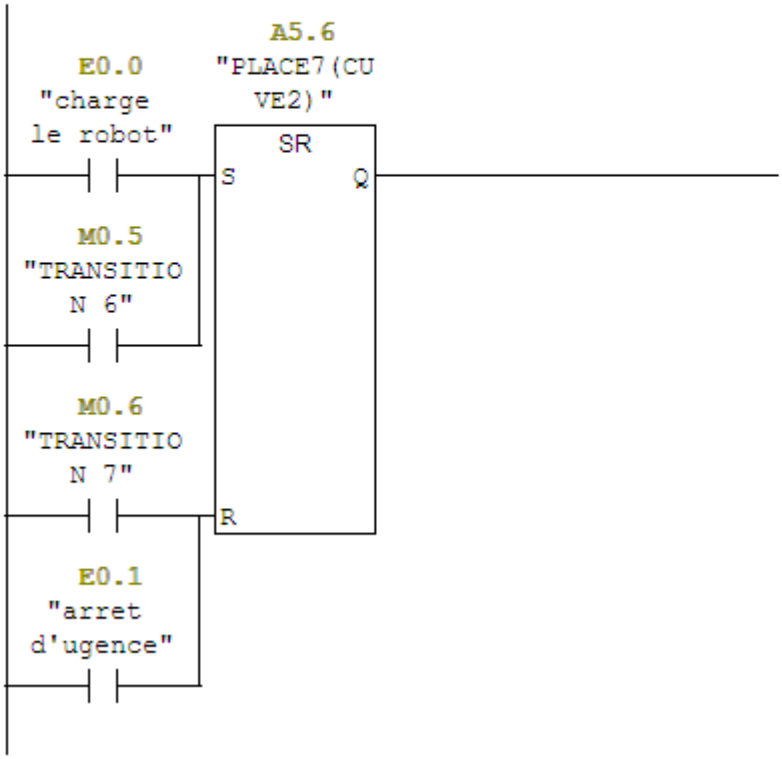

Réseau 6 : Titre :

Place 6 Trajet du robot en charge (Aire de chargement vers la cuve 2)



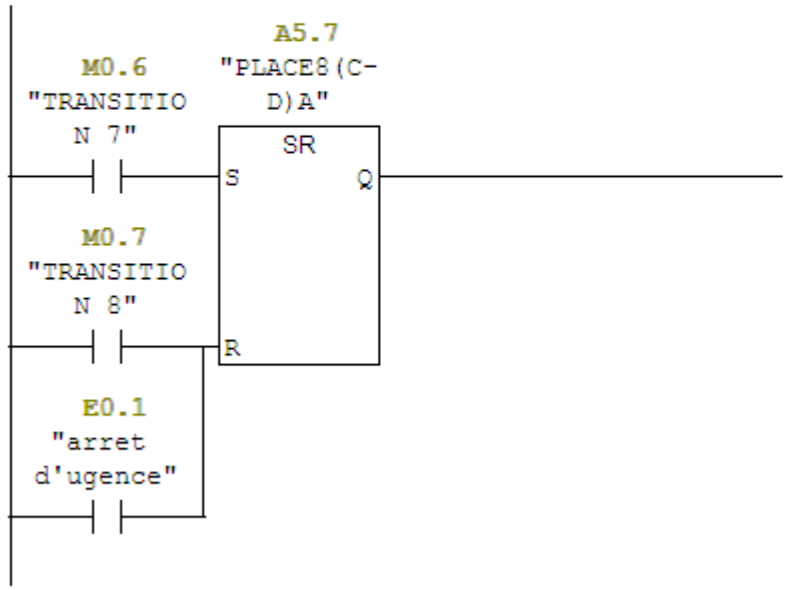
Réseau 7 : Titre :

Place 7 Traitement de la pièce dans la cuve 2



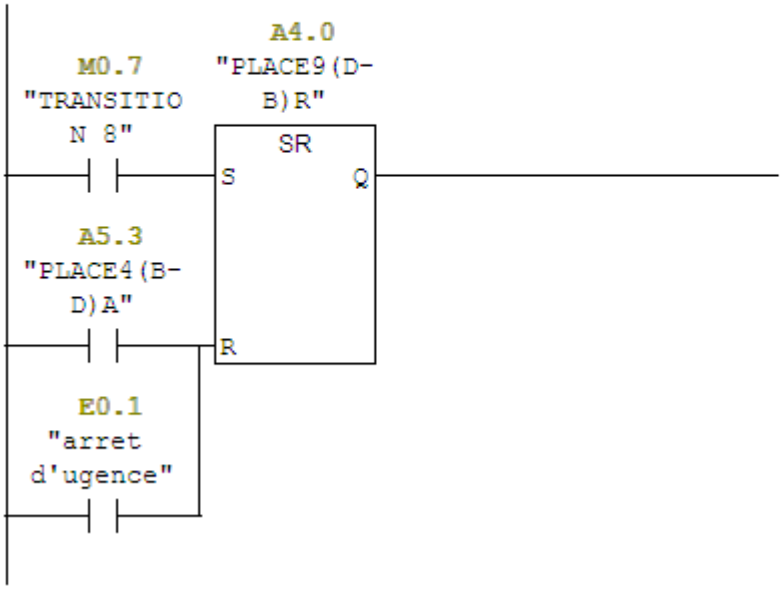

Réseau 8 : Titre :

Place 8 Trajet du robot en charge (Cuve 1 vers l'aire de déchargement)



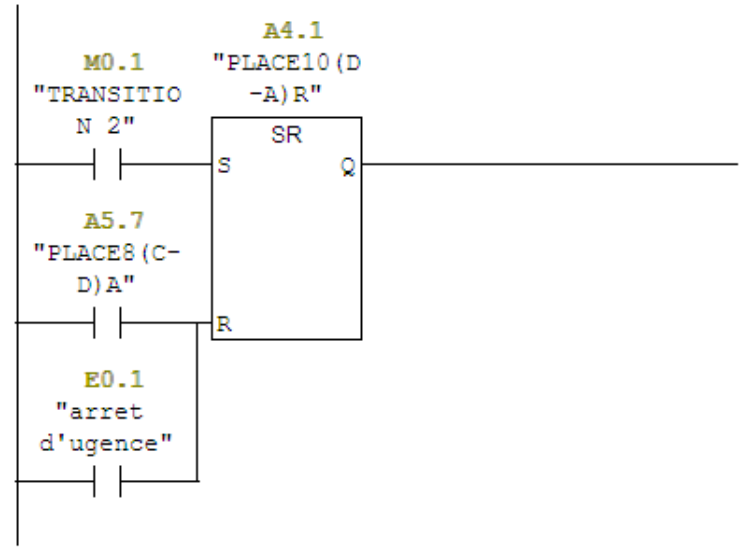
Réseau 9 : Titre :

Place 9 Trajet à vide du robot (Aire de déchargement vers la cuve 1)

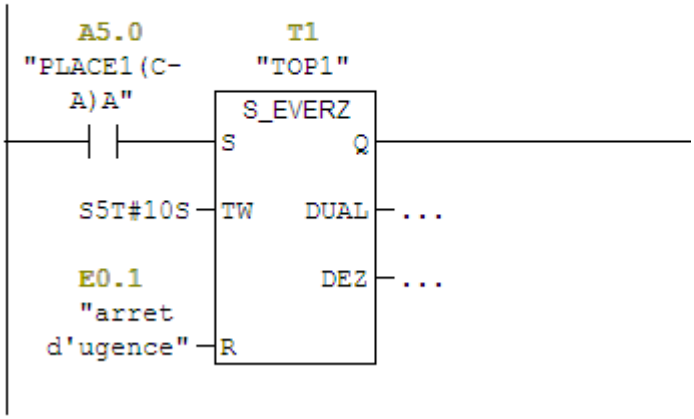


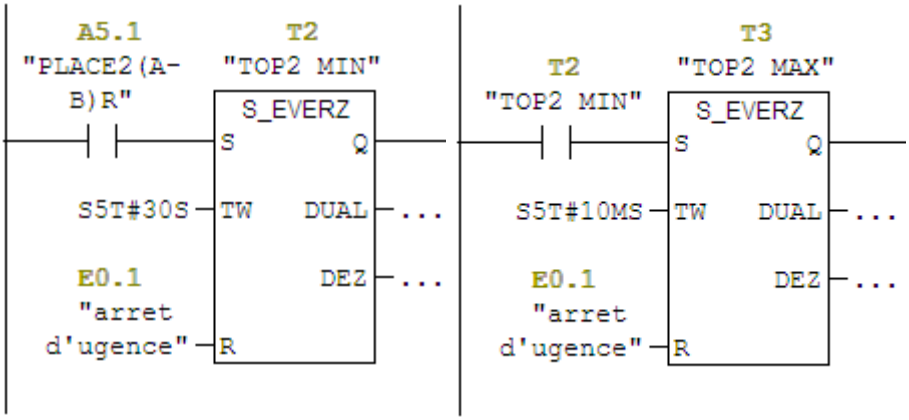

Réseau 10 : Titre :

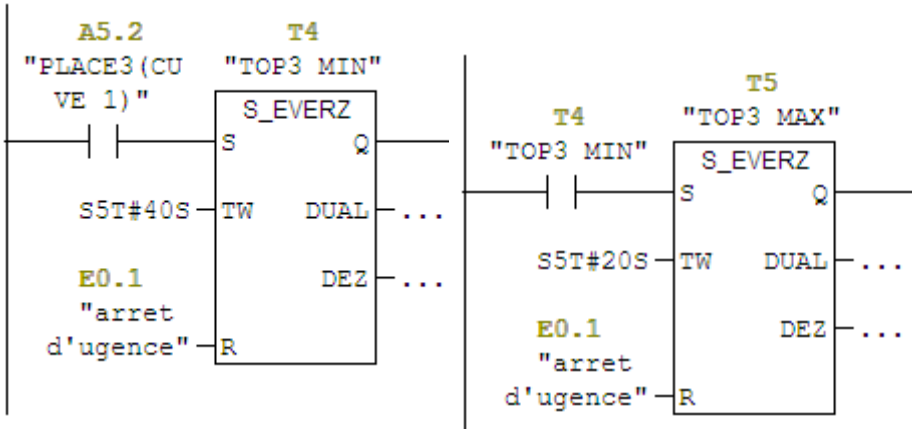
Place 10 Trajet à vide du robot (Cuve 1 vers cuve 2, afin de récupérer la pièce traitée)

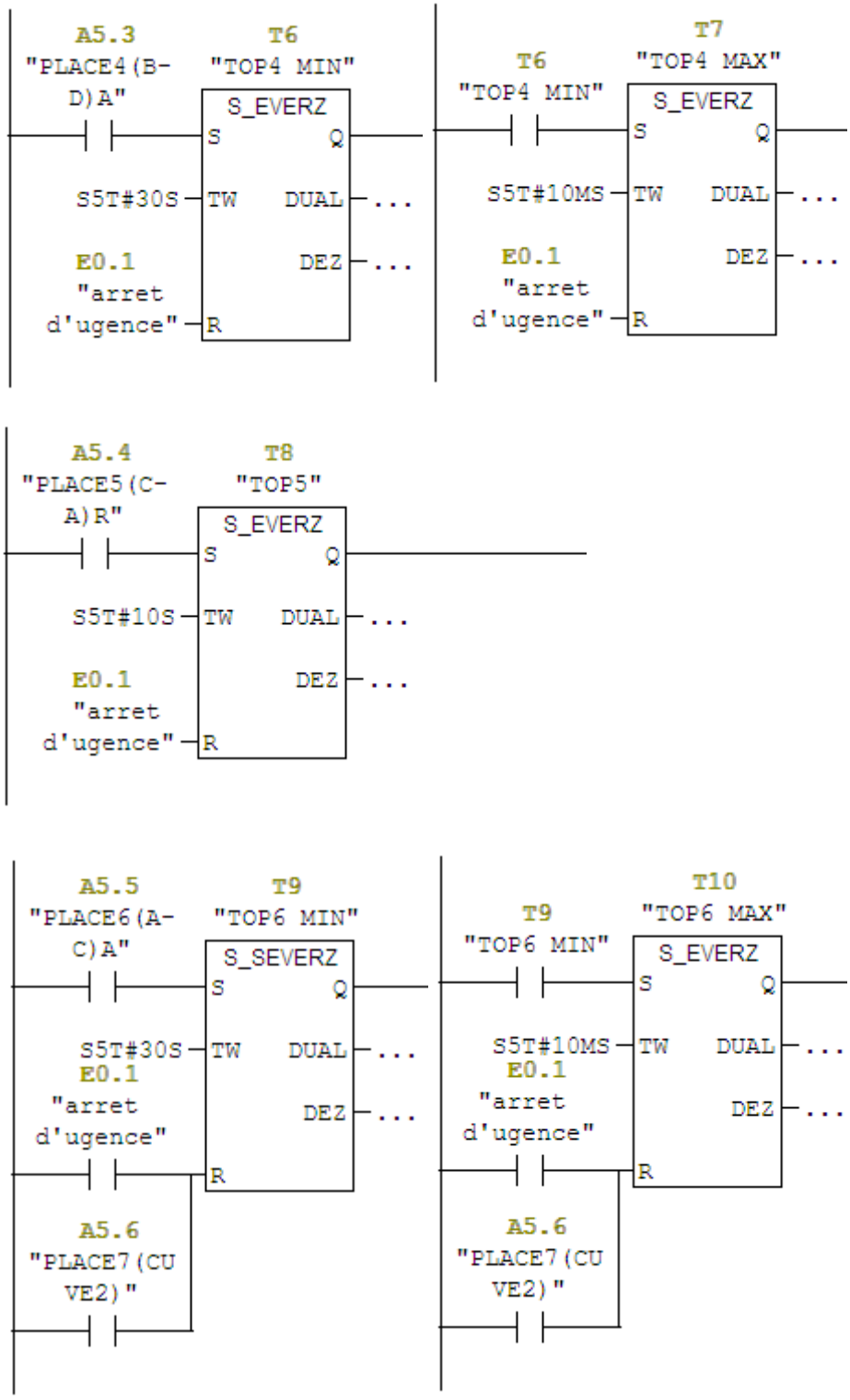


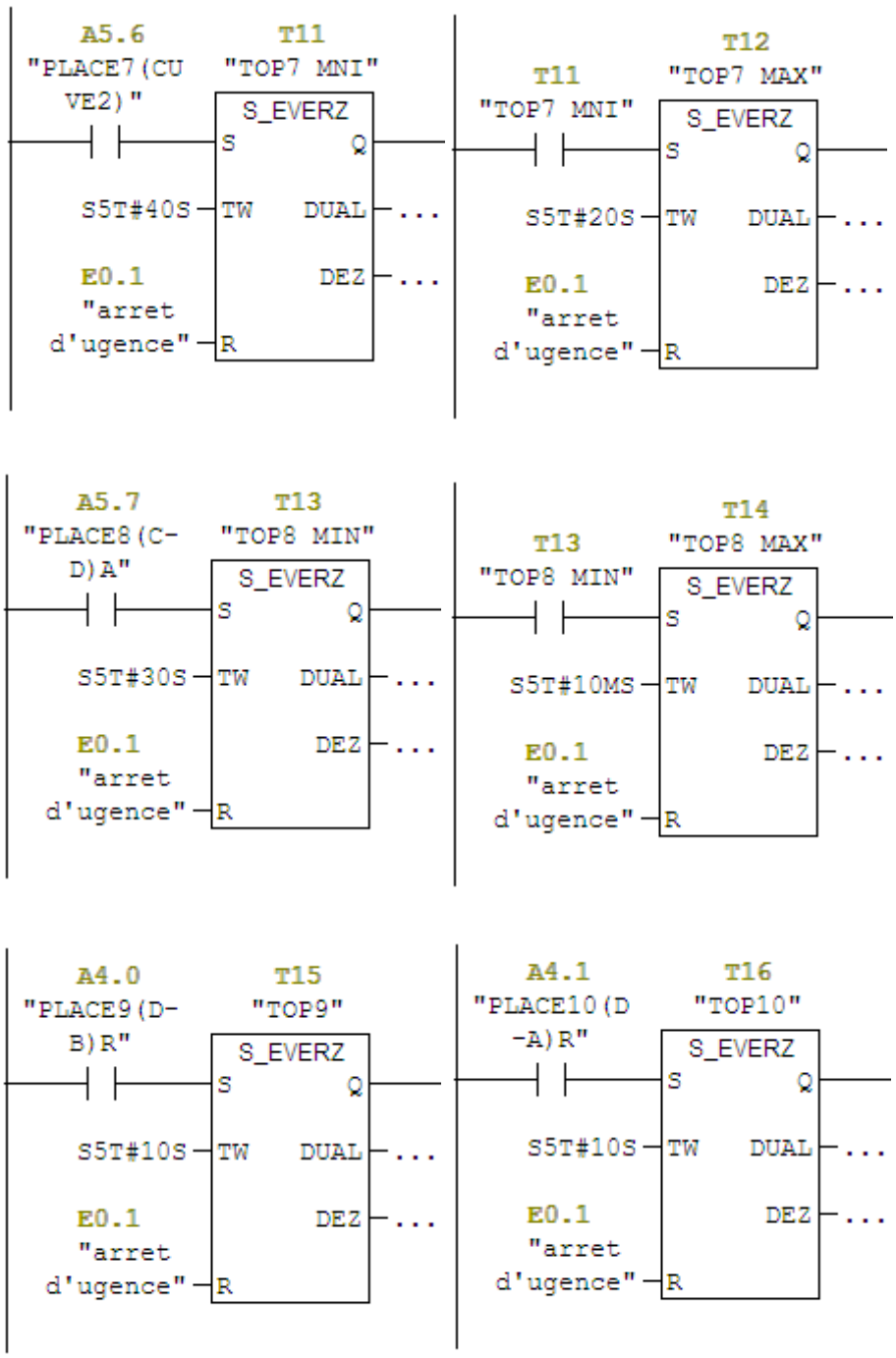
les temporisations associée ou places



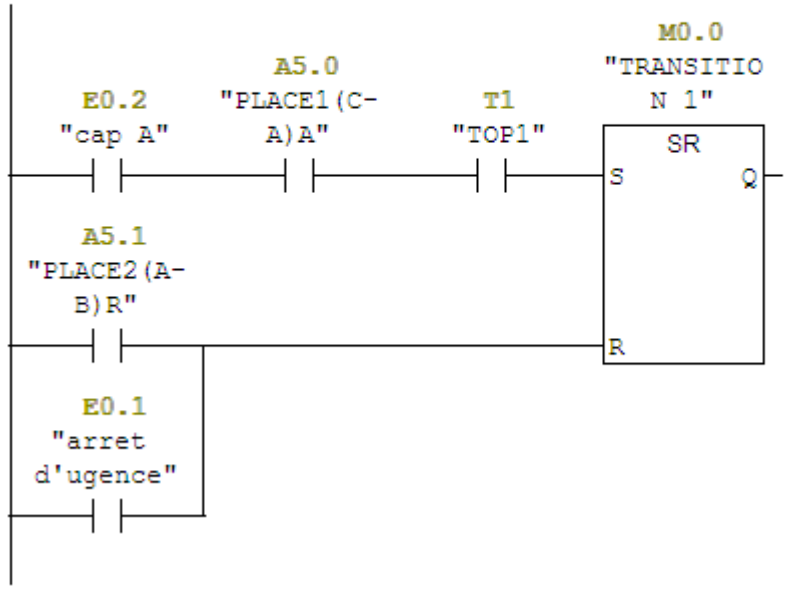




A5.3
"PLACE4(B-D)A"
T6
"TOP4 MIN"
S_EVERZ
S
Q
S5T#30S
TW
DUAL
...
E0.1
"arret d'ugence"
R
DEZ
...
T6
"TOP4 MIN"
T7
"TOP4 MAX"
S_EVERZ
S5T#10MS
A5.4
"PLACE5(C-A)R"
T8
"TOP5"
S5T#10S
A5.5
"PLACE6(A-C)A"
T9
"TOP6 MIN"
S_SEVERZ
A5.6
"PLACE7(CU VE2)"
T9
"TOP6 MIN"
T10
"TOP6 MAX"


A5.6
"PLACE7(CU VE2)"
T11
"TOP7 MNI"
S_EVERZ
S
Q
S5T#40S
TW
DUAL
...
E0.1
"arret d'ugence"
DEZ
...
R
T11
"TOP7 MNI"
T12
"TOP7 MAX"
S_EVERZ
S
Q
S5T#20S
TW
DUAL
...
E0.1
"arret d'ugence"
DEZ
...
R
A5.7
"PLACE8(C-D)A"
T13
"TOP8 MIN"
S_EVERZ
S
Q
S5T#30S
TW
DUAL
...
E0.1
"arret d'ugence"
DEZ
...
R
T13
"TOP8 MIN"
T14
"TOP8 MAX"
S_EVERZ
S
Q
S5T#10MS
TW
DUAL
...
E0.1
"arret d'ugence"
DEZ
...
R
A4.0
"PLACE9(D-B)R"
T15
"TOP9"
S_EVERZ
S
Q
S5T#10S
TW
DUAL
...
E0.1
"arret d'ugence"
DEZ
...
R
A4.1
"PLACE10(D-A)R"
T16
"TOP10"
S_EVERZ
S
Q
S5T#10S
TW
DUAL
...
E0.1
"arret d'ugence"
DEZ
...
R

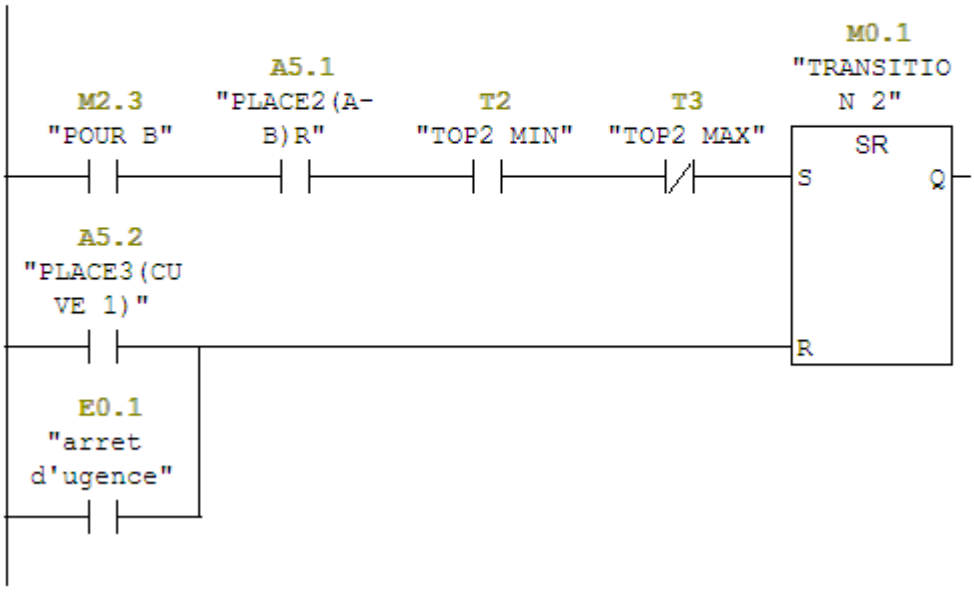
Réseau 27 : Titre :

Matérialisation de la transition 1



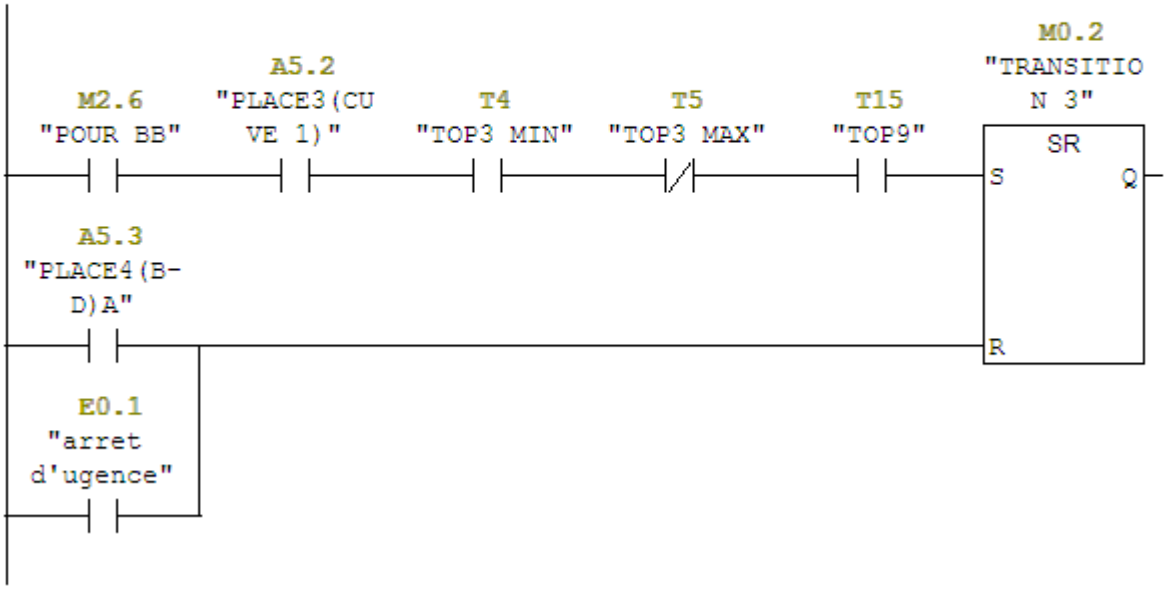

Réseau 29 : Titre :

Matérialisation de la transition 2



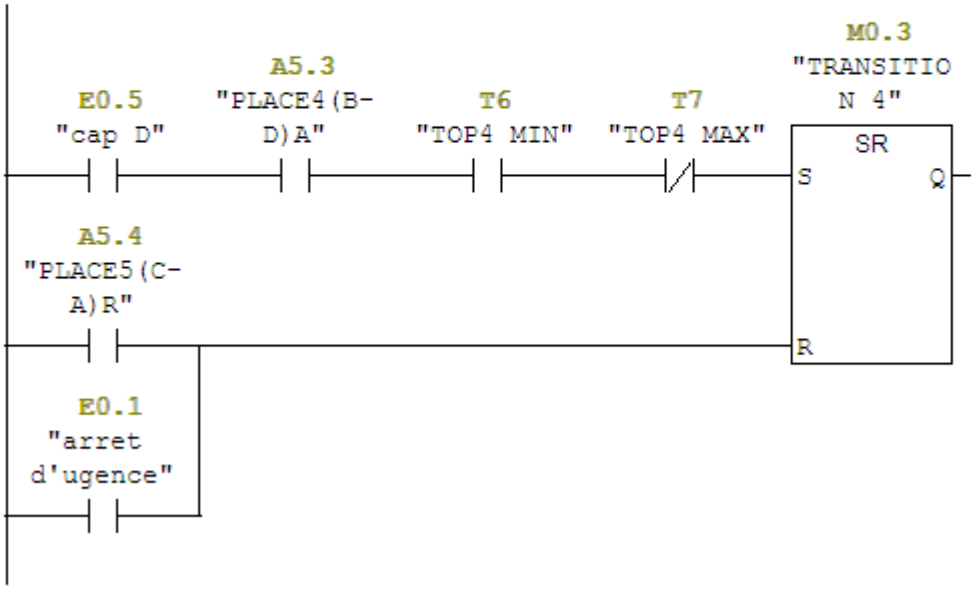
Réseau 31 : Titre :

Matérialisation de la transition 3



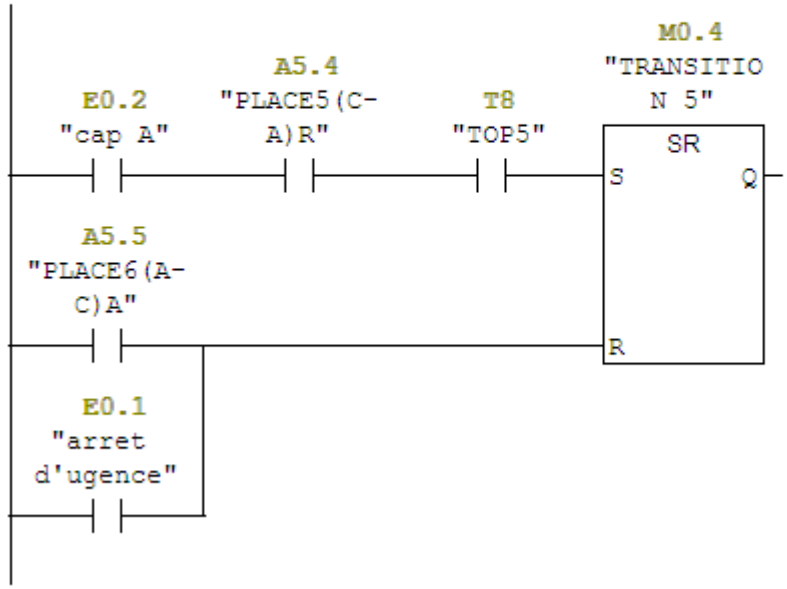

Réseau 32 : Titre :

Matérialisation de la transition 4



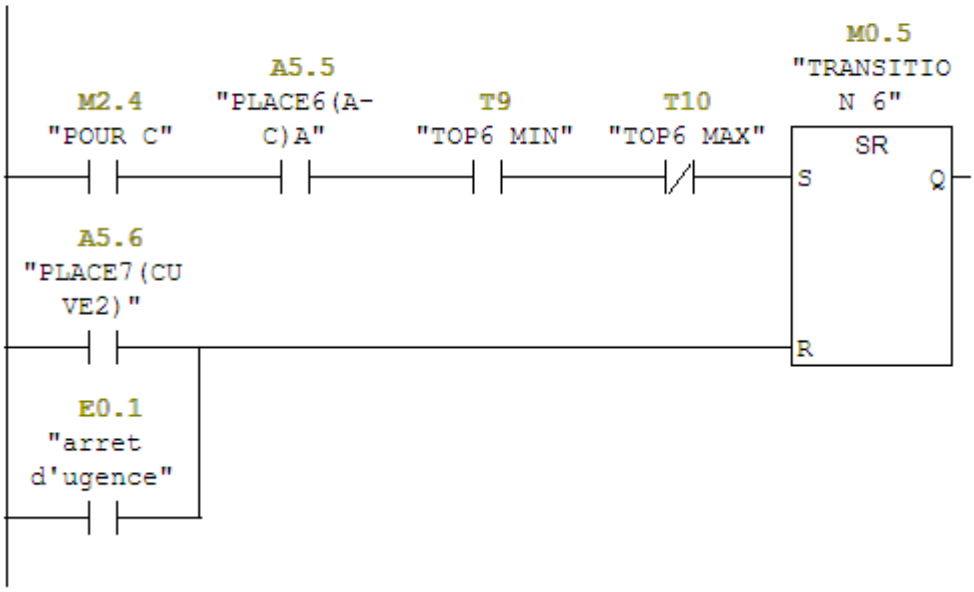
Réseau 33 : Titre :

Matérialisation de la transition 5



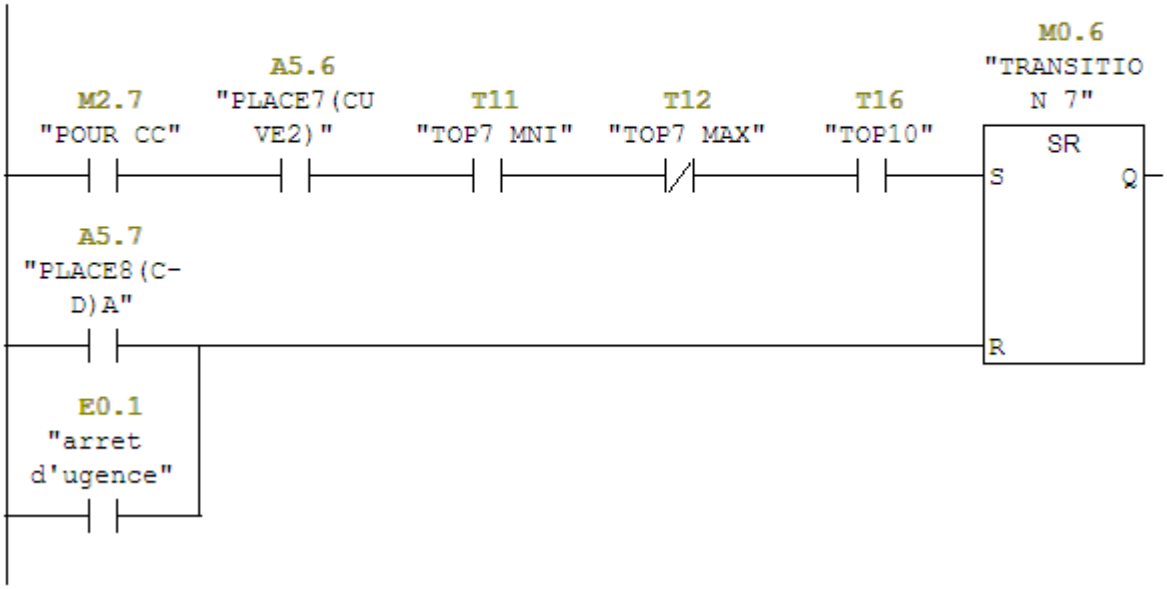

Réseau 35 : Titre :

Matérialisation de la transition 6



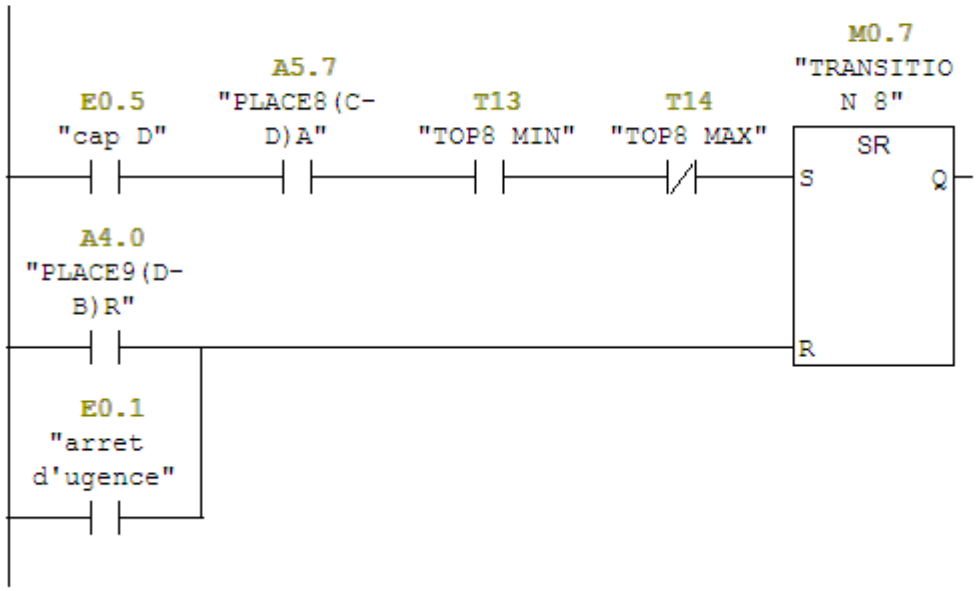
Réseau 37 : Titre :

Matérialisation de la transition 7




Réseau 38 : Titre :

Matérialisation de la transition 8



## I La Liaison entre Step7 et SimFluid

Pour réaliser cette liaison on fait appelle ou programme fait sous Step7 puis on lance le simulateur S7-PLCSIM et l'atelier représente sous SimFluid et la liaison se fait automatiquement grâce ou logiciel **Festo Didactic EzOPC.**