République Algérienne Démocratique Populaire
Ministère de l'Enseignement Supérieure et de la Recherche Scientifique
Université Abderrahmane Mira de Bejaia
Faculté de Technologie
Département De Génie Electrique

Mémoire de fin d'étude

En vue de l'obtention du diplôme de

Master II

Option : Automatisme

Thème

Automatisation des compresseurs d'aire de 40 bar avec STATEFLOW

Réalisé par :

Mr : BENHAMOUCHE Yacine

Encadré par :

Mr : AZZI Abdelmalek

2011-2012

Automatisation des compresseurs

D'aire de 40 bar avec STATEFLOW

Par : BENHAMOUCHE Yacine

Email : lefouyacine@live.fr

Université Abderrahmane Mira de Bejaia
Faculté de Technologie
Département de Génie Electrique

Remerciements

Je tiens à exprimer mes remerciements et ma gratitude à monsieur AZZI Abdelmalek, professeur à l'université A/Mira Bejaia, pour tous ses efforts, ses conseils et pour la modestie qui le caractérise malgré le grand savoir scientifique qu'il possède, choses qui m'ont permis de mener à bien cette étude.

J'adresse mes chaleureux remerciements aux membres du jury qui ont accepté d'évaluer ce travail.

Mes vifs remerciements vont également à l'ensemble des enseignants de département génie électrique de l'université A/Mira Bejaia pour la formation qu'ils nous ont assuré tout long de notre cursus universitaire et tous ceux à qui ont participé à notre formation.

Pour terminer je tiens à remercier tous mes collègues et mes amis qui m'ont aidé et qui m'ont apporté leur soutien moral ainsi que tous ceux qui ont participé de près ou de loin à l'élaboration de ce travail.

Dédicaces

Je dédie ce modeste travail à ma chère mère, à mon cher père, qui m'ont soutenu et encouragé durant toutes mes études et nulle chose ne récompensera leurs sacrifices. Que dieu les garde pour moi.

A tous les membres de ma familles qui m'ont aidé de plusieurs manières et pour leur soutien précieux durant les longues années de ma formation, ce qui leur fait valoir ma grande gratitude. Qu'ils me pardonnent mon manque de disponibilité et mes absences. Que ce travail soit une part de ma reconnaissance envers eux.

A la personne qui a partagé de plus près cette aventure avec moi, celle qui a été ma motivation permanente, celle avec qui j'ai vécu les meilleures années de ma vie, ma chère future femme *Sina* : Merci pour ton amour, ta patience, ta compréhension et ton soutien indéfectible.

A mes amis qui m'ont encouragé et suivi depuis longtemps : Foudil (*pifou*) et Amel, Sofiane (*sosso*) et Djedjiga (*djoudjou*), Nabil (*bilux*) et Mounira (*mouni*), Nourdine (*chvarvare*) et Nawal, Moussa (*ankawkaw*) et Meziane (*misldiricteur*). La réalisation de ce travail n'aurait pas été possible sans votre gentillesse et votre bonne humeur sans oublier bien sûr toutes les personnes que j'ai eu le plaisir de côtoyer toute au long de ces années passées à l'université A/Mira Bejaia : Saïd (elduk), Brahim (Brain), Djamal (HAPI), Nadir,..... . Cette liste n'est bien sûr pas exhaustive, et que les personnes non mentionnées veuillent bien m'excuser.

A tous ceux qui se donnent à fond à la recherche scientifique.

Table des matières

Chapitre II : Modélisation du processus

Chapitre III : Simulation du programme

Chapitre IV : Implémentation de la commande

Figures et Tableaux

Nomenclature

Ci (i=1, 2, 3)	compresseurs C1, C2, et C3
TOR	Tout ou Rien
EVjCi	Electrovannes de l'effet avant du compresseur Ci (i=1, 2, 3), EVj (j=1,2) sortie TOR
AMCi	Arrêt et Marche du compresseur Ci (i=1, 2, 3), sortie TOR
P	Pression captée par un capteur de pression
Pn	Pression nominale
Ii	Indicateur d'état pour les compresseurs Ci (i=1, 2, 3), entrée de type TOR
TEMPCi (i=1, 2, 3)	Le temps de fonctionnement du compresseur
TEMPC	Le temps de fonctionnement du compresseur qui est situé en première position
Vitesse(i)	indicateur d'état de fonctionnement
MAX	le compresseur Ci (i=1, 2, 3) fonctionne à l'état Maximum
MOY	le compresseur Ci (i=1, 2, 3) fonctionne à l'état Moyenne
VID	le compresseur Ci (i=1, 2, 3) fonctionne à Vide
AR	le compresseur Ci (i=1, 2, 3) à l'arrêt

Introduction Générale

Le but de ce travail est d'apporter des améliorations à la commande des différents compresseurs qui permettent de fournir de l'air comprimé, nécessaire au fonctionnement des souffleuses de bouteilles et cela quelques soient les besoins de la production en air comprimé.

Pour minimiser les pertes d'énergie, il faut organiser le fonctionnement des compresseurs d'une façon à ce que leurs activations s'effectuent en fonction de la demande d'air comprimé.

Pour des raisons de maintenance, la répartition équitable du temps de fonctionnement de chaque compresseur doit faciliter les taches des interventions programmées de service de maintenance réduisant ainsi leurs coûts.

Nous passerons par les différents chapitres pour expliquer la problématique et apporter une solution puis la mettre en œuvre par une programmation adéquate.

Dans le premier chapitre, on définit l'automatisme, sa structure de base et les différents types de commande ; ensuite, on passera aux automates programmables industriels utilisés actuellement. Enfin, on donnera une présentation du logiciel utilisé pour la programmation (Stateflow).

Dans le deuxième chapitre, on donnera quelques notions sur le compresseur volumétrique à piston et son fonctionnement et la présentation du processus à commander (les compresseurs) et quelques notions sur les types de commande existant dans nos jours. Enfin, une illustration de la problématique posée.

Le chapitre trois sera consacré à la programmation et à la simulation de la commande.

Le chapitre quatre sera consacré à l'implémentation de la commande, et par la suite, on donnera quelques solutions.

Chapitre I

L’automatisation et langage de programmation

I.1 Définition de l'automatisme

L'automatisme a comme objectif l'étude de la commande des systèmes industriels. Les techniques et les méthodes d'automatisation sont en continuelle évolution, elles font appel à des technologies tel que l'électromécanique, l'électronique, la pneumatique et l'hydraulique. Elles sont présentes dans tous les secteurs d'activités (menuiserie, textile, alimentaire, automobile…etc.).

Son premier objectif est l'amélioration des conditions de travail c'est-à-dire : remplacer l'énergie humaine fournie par une machine (Partie Opérative : P.O).

Les systèmes automatisés qui sont utilisés dans le secteur industriel. Ils possèdent une structure de base identique et ils sont constitués de plusieurs parties plus ou moins complexes reliées entre elles :

- ✓ la partie opérative (PO) ;
- ✓ la partie commande (PC) ;
- ✓ la partie relation (dialogue) (PR).

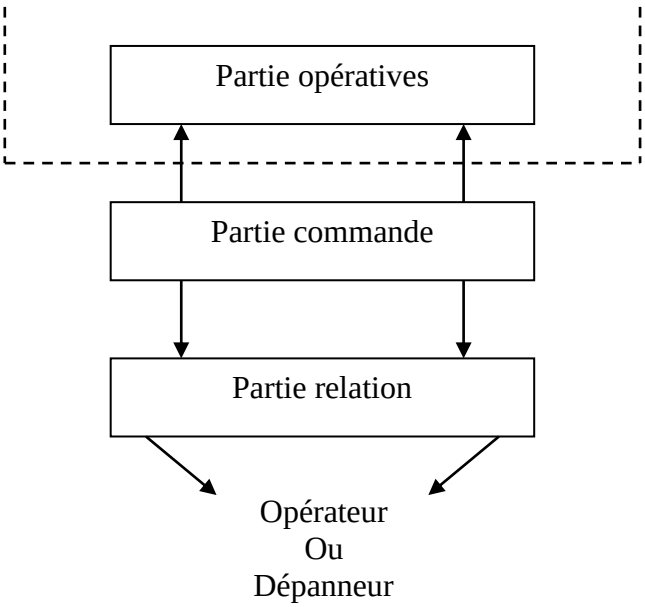


Figure 1 : *Structure générale d'un automatisme* **[1]**.

La partie commande transmet les ordres aux actionneurs à partir des informations fournies par les machines au moyen de capteurs tel que les capteurs de position, de vitesse et de lumière, …etc.

- ✓ La partie commande reçoit des informations transmises par un opérateur en fonctionnement normal, ou par le dépanneur en cas de disfonctionnement de la partie commande ou de la partie opérative.
- ✓ La partie relation se trouve entre la partie commande et l'homme, elle permet à ce dernier de transmettre des informations au moyen des dispositifs adoptés (boutons poussoir, commutateur,…etc.).
- ✓ De même, la partie commande retourne vers l'homme par des informations sous formes compréhensibles par exemple : des voyants, afficheurs ou signaux sonores, …etc. **[1]**.

I.1.1 Structure des automatismes

Selon les définitions de base énoncées au paragraphe précédent, nous allons maintenant détailler les éléments constitutifs d'un automatisme (**Figure 2**).

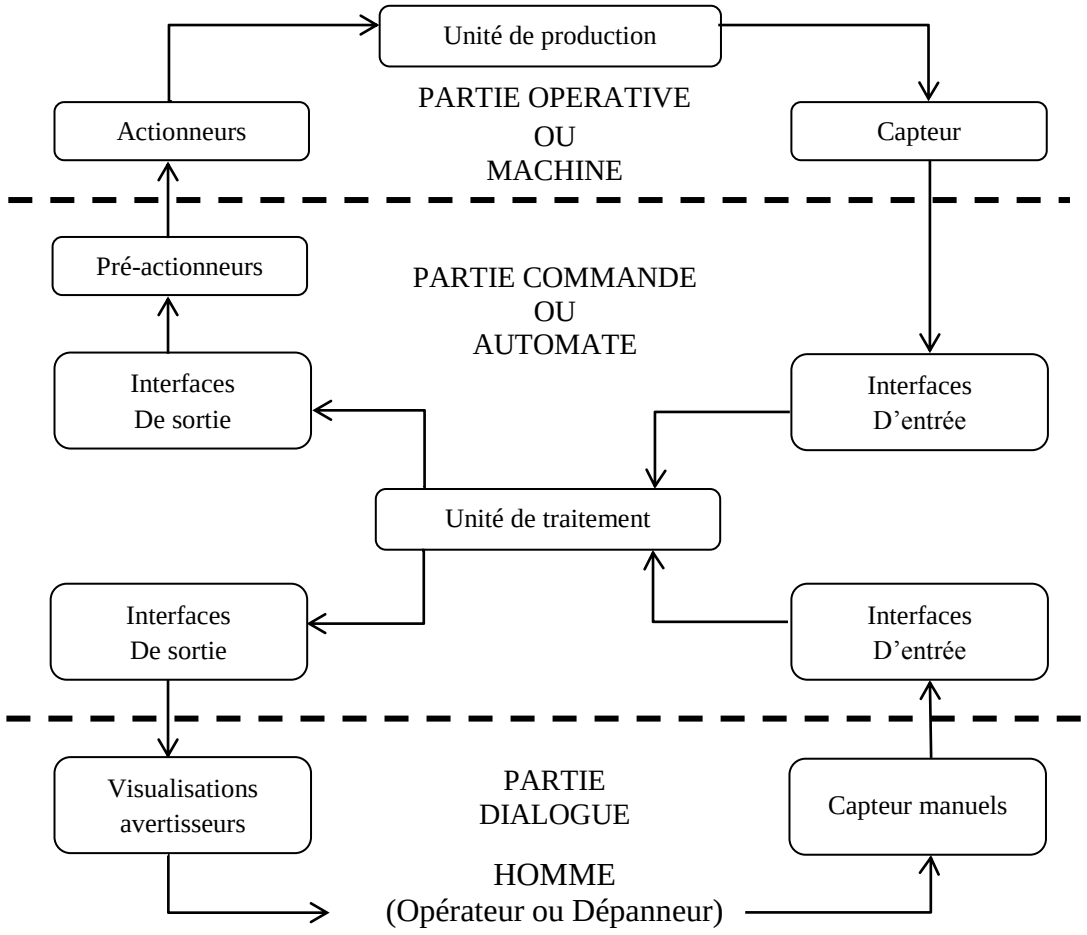


Figure 2: *Structure détaillée d'un automatisme* **[1]**.

I.1.1.1 Analyse de la partie opérative

La partie opérative se compose de trois ensembles :

- ✓ L'unité de production dont le fonctionnement est de réaliser la fabrication ou la transformation du produit.
- ✓ Les actionneurs qui apportent à l'unité de production l'énergie nécessaire à son fonctionnement à partir d'une source d'énergie externe (moteur).
- ✓ Les capteurs qui génèrent des informations utilisables par la partie commande **[1]**.

I.1.1.2 Analyse de la partie commande

La partie commande se compose de quatre ensembles :

- ✓ Les interfaces d'entrée qui transforment les informations issues des capteurs placés sur la partie opérative ou dans la partie de dialogue, en information de nature et d'amplitude compatible avec les caractéristiques de l'automate.
- ✓ Les interfaces de sortie qui transforment les informations issues de l'unité de traitement en information de nature et d'amplitude compatible avec les caractéristiques technologique de pré-actionneur d'une part, des visualisations et avertissement d'autre part.
- ✓ Les pré-actionneurs qui sont dépendant des actionneurs et qui sont nécessaire à leurs fonctionnement (démarreur pour un moteur, distributeur pour un vérin).
- ✓ L'unité de traitement qui élabore les ordres destinés aux actionneurs en fonction des informations reçues par des différents capteurs et du fonctionnement à réaliser **[1]**.

I.1.1.3 Analyse de la partie dialogue

La partie dialogue se compose de deux ensembles :

- ✓ Les visualisations et les avertissements qui transforment les informations fournies par l'automate en informations perceptibles par l'homme (informations optiques ou sonores).
- ✓ Les capteurs qui transforment les informations fournies par l'homme (action manuelle sur un bouton poussoir) à des informations exploitables par l'automate **[1]**.

I.2 Les différents types de commande

I.2.1 Le système automatisé combinatoire

Ces systèmes n'utilisent aucun mécanisme de mémorisation, pour chaque combinaison d'entré, correspond qu'une seule combinaison de sortie (0 ou 1), la logique associée est la logique combinatoire. Les outils utilisés pour les concevoir sont l'algèbre de Boole, les tables de vérité et les tableaux de Karnaugh **[2]**.

I.2.2 Le système automatisé séquentiel

Dans un circuit combinatoire, une seule sortie peut toujours être déterminée en fonction des entrées, mais un système séquentiel s'effectue étape par étape. Ces sorties ne se déterminent pas juste en fonction des entrées mais elles sont aussi dépendantes des états précédents du système. En effet avec les mêmes combinaisons d'entrée on peut avoir plusieurs combinaisons de sortie **[2]**.

I.2.3 La logique câblée

La logique câblée n'utilise que les fonctions logiques. Ces dernières sont raccordées entre elles par des connexions (fils électrique, circuit imprimé) d'où le nom de logique câblée.

Elle a un fonctionnement plus rapide, mais les connexions filaires entre les différents équipements limitent son utilisation pour des processus complexes **[1]**.

I.2.4 La logique programmée

La logique programmée nécessite tout un ensemble de fonction (ET, OU, Mémoire,… etc.), dont l'élément principal est un automate programmable industriel (API) ou un ordinateur. La fonction d'automatisme dépend du programme enregistré dans la mémoire.

Le programme peut être modifié sans intervention sur le câblage, d'où une souplesse au quelle la logique câblée ne peut prétendre **[1]**.

I.3 Définition des Automates Programmables industriels (API)

Ces dernières années, les avances technologiques ont conduit au développement des automates programmables industriels (API), ce qui est une révolution importante dans l'automatisme.

L'automate programmable industriel (API), ou en anglais « Programmable Logic Controller (PLC) », est une forme particulière de contrôle à microprocesseur, cette machine électronique programmable est destinée à piloter dans une ambiance industrielle et en temps réel. Elle est conçue pour être exploité par des ingénieurs, utilisé pour le contrôle, essentiellement pour la commande des procédées industriels. Son objectif principal est de rendre tout le mécanisme de type "laisser-faire-seul". Elle est constituée de deux dispositifs (entrée et sortie).

Les dispositifs d'entrée, c'est-à-dire : les capteurs comme les interrupteurs, et les dispositifs de sortie, c'est-à-dire : des pré-actionneurs comme démarreur pour un moteur un distributeur pour un vérin, … etc. (**Figure 3**) **[3]**, **[4]**.

Figure 3 : *Un automate programmable industriel*

I.3.1 Domaines d'emploi des automates

De nos jours les API ont une grande place dans tous les secteurs industriels et prennent une place importante dans les systèmes de commande automatique des machines. Ils remplacent avantageusement les systèmes de logique câblée, dans la plupart des applications industrielles. Les fonctions d'automatisme sont programmées, ce qui permet d'adapter facilement l'application chargée dans la mémoire de l'automate aux conditions de fonctionnement de la machine **[5]**, **[6]**.

I.3.2 Caractéristique d'un automate

Les points qui caractérisent un automate sont :

- ✓ Son application.
- ✓ Son nombre d'entrées / sorties Tout ou Rien (TOR) et les tensions admissibles.
- ✓ Le ou les langages de programmation possibles et leurs puissances de traitement.
- ✓ La taille mémoire nécessaire pour sauvegarder les programmes.
- ✓ Son boitier compact ou modulaire (monobloc).
- ✓ Le nombre d'entrées et sorties analogiques.

- ✓ La communication possible par réseau **[4]**.

I.3.3 Nature des informations traitées par un automate

Les informations peuvent être de type :

Tout ou Rien (T.O.R.) : l'information ne peut prendre que deux états (vrai/faux) .C'est le type d'information délivrée par un détecteur, (un bouton poussoir…etc.).

Analogique : l'information est continue et peut prendre une valeur comprise dans une plage bien déterminée. C'est le type d'information délivrée par un capteur (pression, température …etc.).

Numérique : l'information est contenue dans des mots codés sous forme binaire ou bien hexadécimale, c'est le type d'information délivrée par un ordinateur **[5]**.

I.3.4 Architecture extérieur des automates

Les automates sont devisés en deux grandes familles :

- Compact
- Modulaire

I.3.4.1 Type compact

Les API de type compact sont constituées d'un processeur, ils sont des appareils avec un nombre fixe d'entrées-sorties digitales et analogiques.

Ils sont cependant extensibles par blocs jusqu'à environ 250 entrées-sorties. Ils sont principalement exploités pour des applications de complexité moyenne avec, et un traitement limité des fonctions analogiques **[6]**.

I.3.4.1 Type modulaire

Les API de la gamme modulaire ou monobloc intègrent dans un seul module, la carte mère nommée Unité Central (UC ou CPU), l'alimentation et les interfaces d'entrées / sorties. Ces automates sont intégrés dans les automatismes complexes où ils nécessitent de la puissance, une capacité de traitement et de la flexibilité **[7]**.

I.3.5 Choix des langages de programmation des automates

Les langages de programmation des automates programmables sont partagés en trois groupes :

- ✓ Les langages textuels :
  - **Liste d'instruction «Instruction List »** (IL).
  - **Texte structuré « Structured Text »** (ST).
- ✓ Les langages graphiques :
  - **Plans de contacts « Ladder Diagram »** (LD)
  - **Diagramme de blocs fonctionnels** (FDB).

Il est toujours possible d'utiliser plusieurs langages au sein d'un même projet, fonctionnant sur un seul automate programmable. Le choix peut ainsi être guidé par la qualification des programmeurs qui interviennent aux différents stades du projet **[6]**.

I.4 Présentation de stateflow

Le produit de Stateflow est un outil de conception graphique et de développement pour la commande et la logique de surveillance, utilisé en même temps que le simulink. L'utilisation de Stateflow permet visuellement de modéliser et simuler les systèmes dynamiques complexes. Il est également facile de modifier la conception, évaluer les résultats et vérifier le comportement du système à n'importe quelle étape de la conception. Dans Stateflow, les modes de système sont représentés par des états. L'évolution des états actifs est régit par les transitions. Les évènements fournissent des motivations, que le modèle utilise pour la transition entre les états. Ces systèmes sont employés souvent pour modeler la logique où commander dynamiquement d'un dispositif physique tel qu'un ventilateur, un moteur ou une pompe. Des systèmes entraînés par les événements peuvent être modelés en tant que machines aux états finis.

Pour construire les machines aux états finis, le logiciel de Stateflow fournit les objets graphiques que l'on peut utiliser tout au long de la conception, pour cela, nous utilisons une palette d'objet dans le rédacteur de Stateflow **[8]**, **[9]**.

I.4.1 Hiérarchie des objets dans Stateflow

L'objet le plus élevé dans la hiérarchie est la machine de Stateflow. Elle contient tous les autres objets dans le modèle de Simulink.

Les machines de Stateflow arrangent des objets dans une hiérarchie basée sur la retenue. C'est-à-dire, un objet de Stateflow peut contenir d'autres objets (**Figure 4**).

De même, les diagrammes peuvent contenir l'état, la boîte (Box), la fonction, les données (data), l'événement, la transition et la jonction.

Continuant la hiérarchie de Stateflow, les états peuvent contenir aussi bien tous ces objets, y compris d'autres états. Vous pouvez représenter la hiérarchie d'état avec des Superstates et des sous-états.

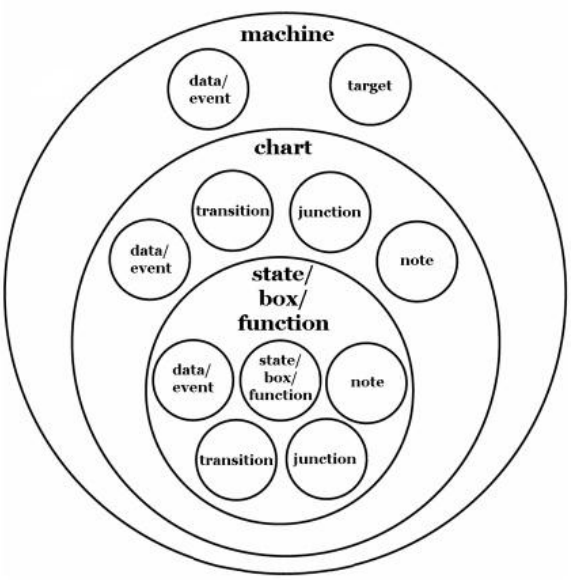


Figure 4: *La hiérarchie dans Stateflow* **[8]**.

I.4.2 La construction de la machine Stateflow

Pour construire une machine d'états finis, Stateflow propose un graphe ou chart qu'on peut le déplacer dans une fenêtre Simulink.

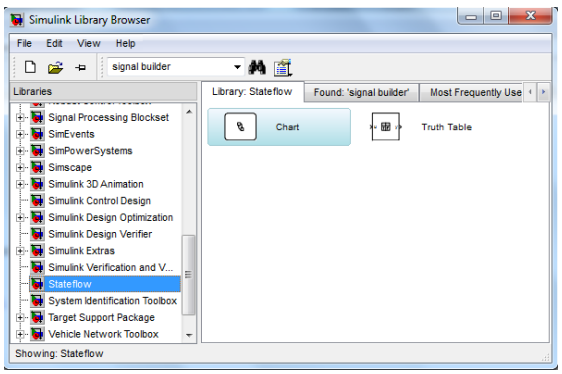


Figure 5 : *bibliothèque Simulink.*

Lors d'un double-clique sur le bloc **« Chart »** dans la fenêtre Matlab\Simulink, on aboutit à une fenêtre Stateflow dans laquelle se trouve une palette d'outils nécessaires pour construire une machine d'état fini.

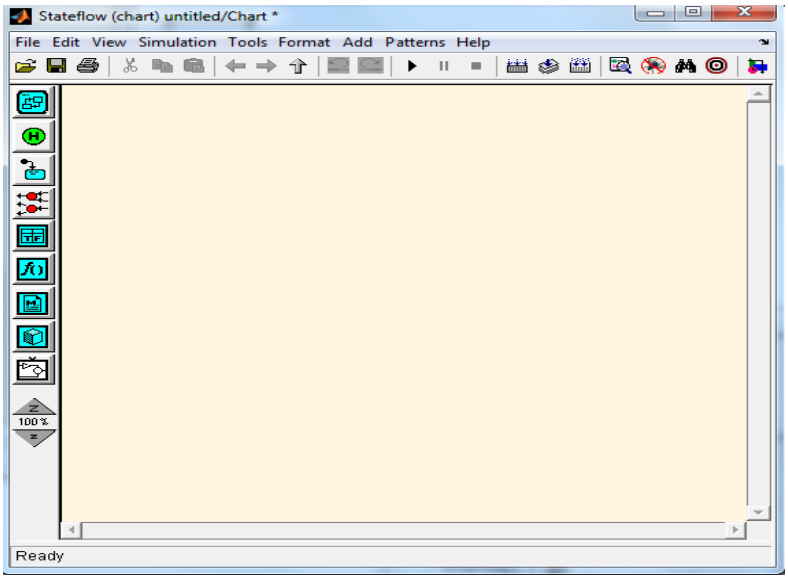
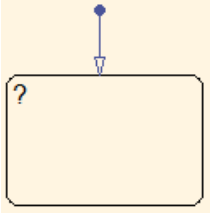
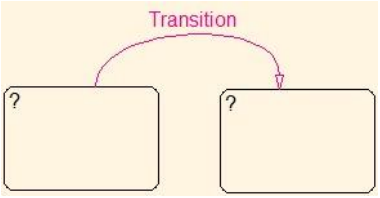
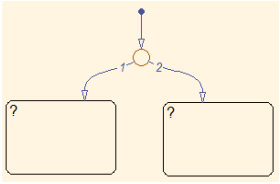
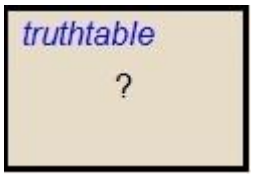
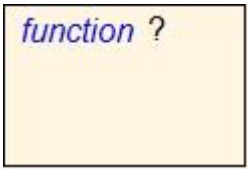
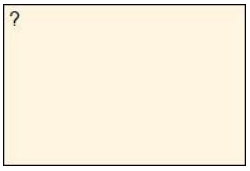


Figure 6 : *Environnement de Stateflow.*

Les différents outils de la *palette d'outils* sont représentés dans le tableau suivant :

Tableau 1: *Présentation des différents outils de Stateflow* **[10]**.

Nom	Notation et explication	Toolbar Icon
Etat	Rectangle qui définit l'état à qui l'on donne le nom a la position du signe (?). 	
Jonction de l'historique	Elle nous permet d'enregistrer l'activité d'un état.	
Transition de défaut	Transitions de défaut. Objets graphiques qui spécifient quel état exclusif qui est en activité quand il y a deux états exclusifs ou plus au même niveau dans la hiérarchie. 	

Transition	Transition d'un état à un autre sur certaine condition que l'on l'écrit à la place de (Transition). 	
Jonction connective	Jonction ou arrivent et partent plusieurs transition. Les transitions sont exécutées, soit par défaut selon leur création ou par la condition qui lui associe. 	
Fonction de table de vérité		
Fonction graphique	Une fonction graphique est une fonction que vous définissez graphiquement avec un graphique d'écoulement qui inclut la langue d'action de Stateflow 	
Fonction matlab		
Box	Le Box ou la boite est un objet graphique qui organise d'autres objets dans le diagramme, tel que des fonctions et des états 	
Fonction de simulink		

I.4.3 Les états

Dans les états, on décrit les actions à réaliser dans le cadre du graphe qui décrit la machine en états finis.

Un état peut être actif ou inactif selon des événements qui peuvent intervenir ou des conditions de validation des transitions.

Un état est appelé super-état, ou état parent vis à vis de ce qu'il contient et ceux-là deviennent des sous-états (ou enfants).

Les états « Eteint » et « Allume » sont des sous-états de l'état « Clignt », lui-même est un enfant de l'état racine « Compteur » (**Figure 7**).

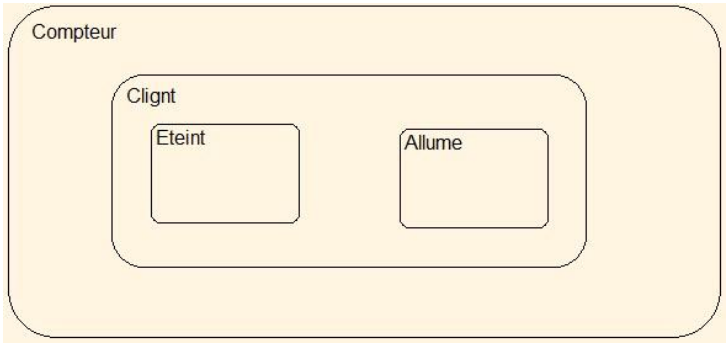


Figure 7: *La hiérarchie des états.*

Cette hiérarchie apparaît dans la fenêtre de gauche de l'explorateur (Tools … Explorer).
Notons que Stateflow est un enfant pour Simulink. Le bloc Stateflow (compteur) possède la même hiérarchie que les autres objets de Simulink (**Figure 8**).

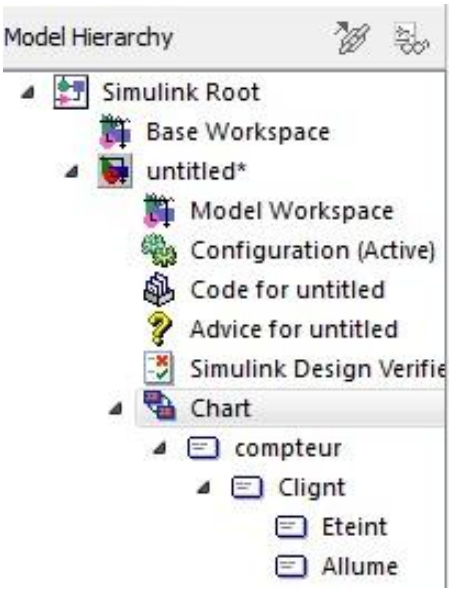


Figure 8: *La hiérarchie du modèle.*

Le cas de la **Figure 7** contient des états exclusifs, ils ne peuvent pas être actifs en même temps. On dit qu'ils sont en décomposition « OU », lorsque l'exclusivité est réalisée par les transitions qui contiennent des conditions qui valident le passage d'un état vers un autre.

Un graphe Stateflow peut posséder plusieurs états qui peuvent être actifs simultanément. Ces états sont dits parallèles en décomposition « ET ». Les états parallèles sont encadrés par des bords en pointillés.

Chaque état possède un nom (label) en haut à gauche du rectangle délimitant cet état.
En général, on trouve dans l'état :

nom/
entry : « actions à réaliser à l'entrée de cet état »
during : « actions à réaliser durant l'activité de cet état »
exit : « actions à réaliser dès qu'on sort de cet état »
on <nom_evt> : « actions à réaliser à l'occurrence de l'événement « nom_evt » »
bind : « variable » « actions »

Si on ne spécifie rien, c'est considéré comme entry donc les actions sont à exécuter dès l'entrée de l'état.

Le fait de mettre « bin : a » signifie que seul cet état ou des états enfants peuvent modifier la valeur de la variable « a », les autres états peuvent utiliser cette variable mais ne peuvent pas modifier sa valeur.

Le nom de chaque état doit être unique à l'intérieur de chaque super état pour qu'il n'y ait pas d'ambiguïté.
Les actions à réaliser dans le Stateflow sont très proches du langage « C » ou Matlab :

- Exemple :
- « a++ » : Incrémentation de la variable « a ».
- « abs(a) » : La valeur absolue de la variable « a ».
- « sqrt(a) » : La racine de la variable « a ».

Tout comme pour MATLAB, une ligne peut se terminer par le symbole… pour indiquer que la ligne continue jusqu'à la suivante.

- Exemple :

  « secondes = (secondes-60*minutes …

  - 3600*heures) ; »

On peut utiliser les fonctions MATLAB et les variables de l'espace de travail grâce à l'utilisation de l'opérateur « ml ».

I.4.4 Les transitions

Une transition est un objet graphique qui lie un état à un autre. Un label décrit les circonstances ou les conditions du passage entre ces états.

I.4.4.1 Les transitions par défaut

La transition par défaut (sans aucune condition, ni label) détermine l'état qui doit être actif lorsqu'il y a ambiguïté entre plusieurs états en décomposition OU, et qui ont le même niveau de hiérarchie.

Chaque état qui contient des sous-états doit avoir sur l'un d'eux une transition par défaut qui détermine cet état actif dès l'entrée de cet super état (ou état parent).

I.4.4.2 Labels des transitions

Une condition est une expression booléenne qui valide la transition du passage d'un état vers un autre. La condition est mise entre crochets.

Les conditions peuvent faire intervenir des variables locales ou des entrées SIMULINK. Dans une transition, on peut réaliser des actions comme celles qu'on fait à l'intérieur d'un état, et les actions réalisées dans une transition sont mises entre accolades.

Dans le cas suivant, la transition se fait lorsque la température dépasse le seuil «12^0 » et dans ce cas on arrête le moteur de recirculation de l'air en mettant la variable « recirc» à 0.

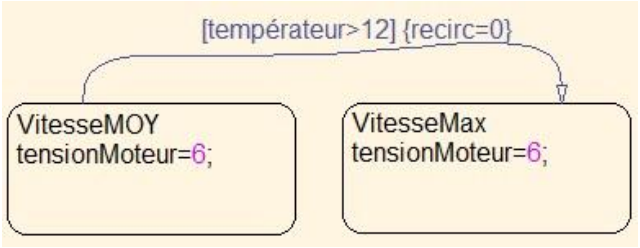


Figure 9: *Condition du passage d'un état à l'autre avec {condition-action}.*

Le label d'une transition est dans le cas général suivant :

« event[condition]{condition_action}/transition_action »

La transition sera validée lorsque :

- l'événement « event » a lieu.
- la condition est vraie.

I.4.5 Les événements

Les événements, objets non graphiques, agissent sur l'exécution du graphe Stateflow. Tous les événements doivent être définis par le menu « Add => Event » dans « Model Explorer ».

L'occurrence d'un événement peut servir à valider une transition du passage d'un état à un autre ou une action à s'exécuter.

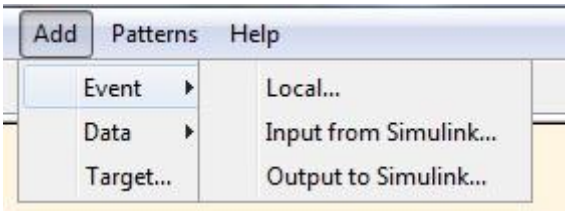


Figure 10 : *Comment ajouter les événements.*

I.4.6 Les objets Data

Les objets Data (données) peuvent être des variables locales, des entrées ou des sorties vers SIMULINK ou une constante… etc.

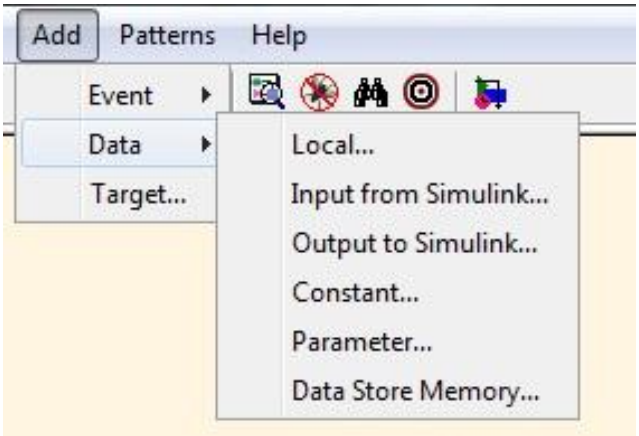


Figure 11 : *Comment ajouter les données (Data).*

I.4.7 Pourquoi employer une fonction graphique ?

Il est plus facile de créer, accéder et contrôler une fonction graphique qu'une fonction textuelle, telle qu'une fonction de langage C ou une fonction de MATLAB que vous devez définir extérieurement comme une fonction textuelle.

Une fonction graphique est un objet originaire de Stateflow. Vous employez le rédacteur (*patterns*) pour créer une fonction graphique qui réside dans le modèle avec les diagrammes qui appellent la fonction.

Chapitre II

Modélisation du processus

II.1 Introduction

Nous avons à commander un système de trois compresseurs identiques, qui sont des compresseurs alternatifs à piston de classe volumétrique. Ils peuvent fournir une pression nominale de 40 bar, et leurs sorties sont reliées à une seul conduite (système parallèle).

II.2 Compresseurs à piston

Les compresseurs à piston sont des compresseurs volumétriques comportant un ou plusieurs pistons qui se déplacent linéairement et alternativement dans leur cylindre et qui augmentent ainsi la pression de l'air en réduisant son volume. Les volumes d'air sont successivement admis dans le cylindre hermétiquement fermé. Ce type de compresseur est disponible en étage unique ou multi étages selon le niveau de pression requis. Les compresseurs à piston ont un rendement élevé **[11]**.

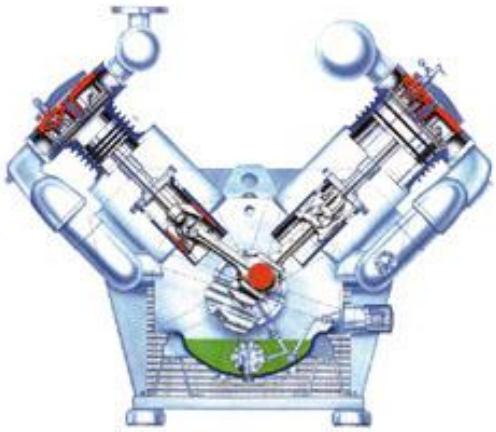


Figure 12: *Compresseurs à piston* **[11]**.

II.2.1 Le compresseur volumétrique à piston à un étage

La (**Figure 13**) montre les composantes d'un compresseur à un étage :

- ✓ Un cylindre.
- ✓ Un piston.
- ✓ Un clapet d'aspiration.
- ✓ Un clapet de refoulement.
- ✓ Une bielle.
- ✓ Un volant d'entrainement

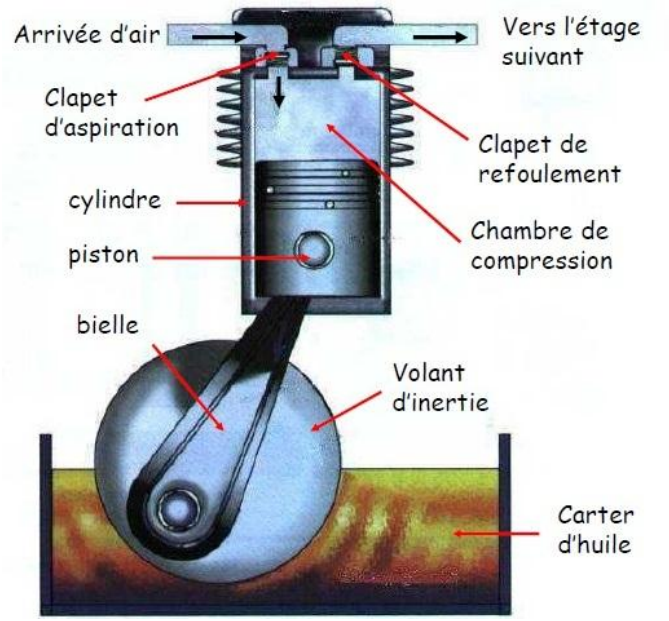


Figure 13 : *Les composantes de base d'un compresseur à un étage.*

Remarque :

- ✓ Le clapet est la partie mobile, le siège est la partie fixe. Et l'ensemble constitue une soupape.
- ✓ Les soupapes sont contenues dans la culasse qu'on appelle plus généralement boite à clapet.

II.2.2 Compresseurs alternatifs à piston (simple action)

Lorsque le volant tourne, il est entrainé par une machine électrique ou thermique, le piston est animé d'un mouvement alternatif quasi-sinusoïdal.

- ✓ Lorsqu'il descend, la pression dans le cylindre diminue. Dès qu'elle est inférieure à celle en amont du clapet d'aspiration, celui-ci s'ouvre, en laissant l'air rentrer à l'intérieur, cette phase est appelée (Aspiration).
- ✓ Lorsqu'il monte, la pression dans le cylindre augmente. Dès qu'elle dépasse la pression au-dessus du clapet de refoulement, celui-ci s'ouvre en laissant l'échappement de l'air vers la sortie. Cette phase est appelée (refoulement).
- ✓ L'actionnement des clapets se fait par la différence de pression entre l'intérieur de cylindre et en amont des clapets **[11]**, **[12]**.

II.2.3 Compresseurs alternatifs à piston (double action)

Le cycle de compression s'effectue en deux temps qui sont dus à l'architecture du compresseur, contenant deux clapets d'aspiration et deux clapets de refoulement (**Figure 14)**.

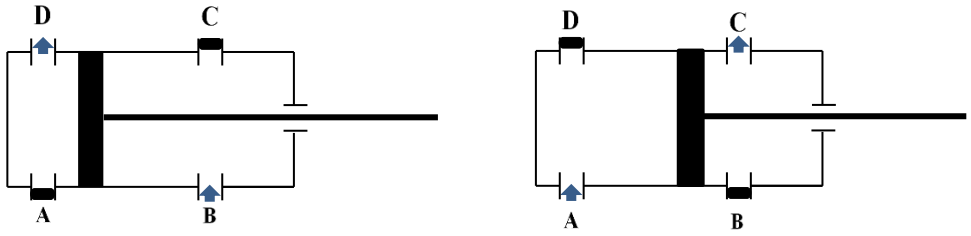
- **Temps** : dans un temps, il s'effectue une aspiration ou un refoulement dans un piston à simple action, et dans un autre double action l'aspiration d'un côté et le refoulement s s'effectue dans l'autre côté en même temps.

Premier temps :

- ✓ Aspiration par « B », et le clapet « A » est fermé.
- ✓ Refoulement par « D », et le clapet « C » est fermé.

Deuxième temps :

- ✓ Aspiration par « A », et le clapet « B » est fermé.
- ✓ Refoulement par « C », et le clapet « D » est fermé **[2]**.



Premier temps. Deuxième temps.

Figure 14 : *Les temps du cycle de compression.*

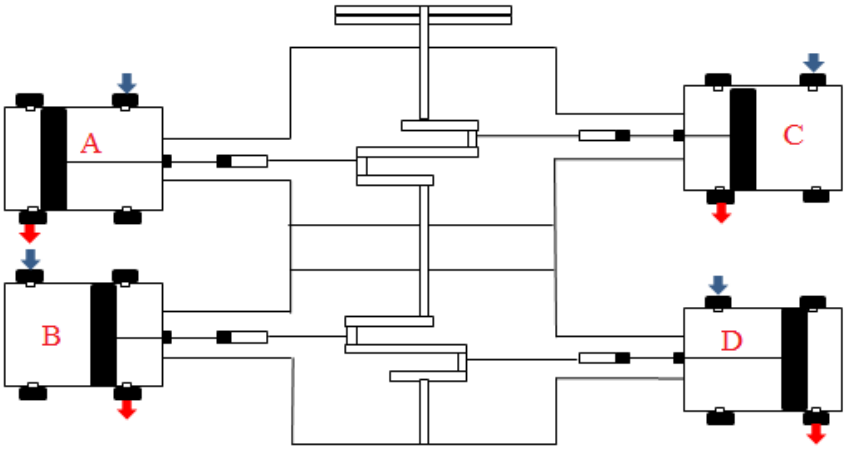
II.2.4 Le fonctionnement du compresseur à quatre étages

Les principes de fonctionnement des compresseurs à simple action, expliquent le fonctionnement du compresseur à quatre cylindres (**Figure 15**) utilisé dans cette étude :

- ✓ Deux cylindres (A, B) sont réservés au premier étage de compression, les deux aspirent de l'air atmosphérique.
- ✓ Un cylindre (C), pour le deuxième étage de compression, l'air aspiré est sous une pression d'environ 3 bar.
- ✓ Le dernier cylindre (D) pour le troisième étage de compression, l'air aspiré est d'environ 12 bar.

✓ La **Figure 15** donne le fonctionnement d'un cycle de compression qui s'effectue en deux temps aspiration et respiration **[13]**.

- **Premier temps :**



- **Deuxième temps**

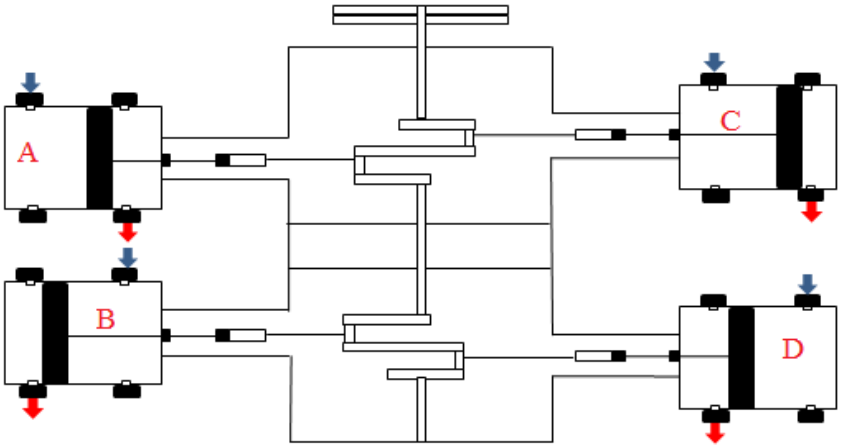


Figure 15 : *Compresseur à quatre cylindres* **[13]**.

II.2.5 Caractéristiques générales des compresseurs alternatifs à piston

Le compresseur à piston a besoin d'être lubrifié en permanence. La partie inférieure du carter forme une réserve d'huile.

Le compresseur à piston est très sensible à l'arrivé de fluide liquide : si quelques gouttes de liquide pénètrent au niveau des soupapes, elles provoquent une usure lente. Si le fluide liquide pénètre en grande quantité, la destruction des clapets est immédiate.

De là, des protections anti-coups de liquide ont été adoptées (ressort puissant sur le chapeau de cylindre, il est capable de se soulever en cas d'arrivée de liquide) **[14]**.

II.3 Distribution économique d'air comprimé

La distribution de l'air comprimé à l'application se fait grâce au réseau de tubes, tout type d'obstacle, de diminution de section et des irrégularités des proies dans le réseau de tubes engendreront une résistance aux flux d'air et génèreront une baisse de pression, résultante des pertes d'énergie.

Des pertes sont aussi générées par l'ensemble des équipements auxiliaires (mais le plus souvent indispensables) du compresseur, tels que les systèmes de filtration de particules de saleté ou d'eau, ainsi que les séparateurs air/lubrifiants, les refroidisseurs, les séparateurs d'humidité ou encore les sécheurs qui sont souvent intégrés au système.

Ces composants nécessitent une maintenance et un nettoyage réguliers afin de réduire les pertes en pression de sortie, le cheminement des conduites d'admission d'air doivent être les plus courts possibles avec le moins virages aigus possibles pour éviter les baisses de pression **[11]**.

II.3.1 Gestion de la variation de la demande

Les besoins en air comprimé d'une usine changent de façon permanente. Les systèmes de contrôle permettent d'adapter en temps réel la fourniture d'air comprimé avec la demande du système. Ces systèmes de contrôle constituent l'un des facteurs les plus déterminants dans le rendement énergétique du système **[11]**.

II.3.2 La régulation du compresseur

Pour la régulation des compresseurs, il existe plusieurs moyens avec leurs performances propres et leurs technologies d'application, et grâce à ces derniers, on pourra améliorer la performance de production d'air comprimé **[15]**.

On peut évoquer quelques méthodes :

II.3.2.1 Les variateurs de vitesse

La variation de vitesse, assurée par des variateurs (convertisseurs de fréquence AC) intégrés dans les moteurs d'entrainement qui sont en générale des moteurs asynchrones(les

variateurs de vitesse), et peuvent-être le moyen le plus efficace de contrôle de la capacité des compresseurs. La vitesse des compresseurs est régulée en fonction de la pression demandée. Grâce à ce type de contrôle, la pression de sortie peut être maintenue à un niveau très précis **[15]**.

II.3.2.2 Régulation Marche/Arrêt

Cette option n'est présentée que sur les petits compresseurs, avec une capacité. Le moteur s'arrête dès que la demande est satisfaite. Cette méthode est très efficace énergétiquement (consommation 0 % à l'arrêt et 100 % au fonctionnement).
Cependant, le redémarrage trop fréquent du moteur à charge partielle peut créer des problèmes **[15]**.

II.3.2.3 Régulation Tout ou Rien

Beaucoup de compresseurs utilisent ce type de régulation qui fonctionne souvent avec une vanne d'aspiration, celle-ci se ferme complètement lorsque la pression monte au-delà d'un certain niveau, alors que le compresseur continue de fonctionner à vide, ce qui diminue considérablement la puissance absorbée. Lorsque la pression redescend jusqu'à un certain seuil, la vanne s'ouvre à nouveau pleinement et le compresseur fonctionne alors à charge nominale.

Lorsque le fonctionnement à vide se prolonge au-delà d'un certain temps (15 à 20 minutes) le moteur peut être arrêté automatiquement **[15]**.

II.3.2.4 Les circuits de décharge

Les circuits de décharge d'un compresseur diminuent la puissance en réduisant la capacité volumétrique interne du compresseur. Ces circuits sont combinés au système de commande du moteur et assurent le réglage du volume tout en économisant de l'énergie. Comme la décharge s'effectue par les clapets d'aspiration, alors des griffes forcent l'ouverture de ces derniers lorsque la demande en air est réduite. Quand le piston est à double effet est donc doté de deux clapets d'aspiration, la décharge peut se faire progressivement pour assurer la régulation à trois étapes **[16]** :

- ✓ Pleine charge (marche à 100%) : le piston est en fonctionnement normal (aspiration et refoulement).

- ✓ Demi-charge ou charge partielle (marche à 50%) : le fonctionnement du piston est à simple effet qui est du à l'ouverture complète d'un seul clapet d'aspiration.
- ✓ Sans charge (mise à vide, à 0%) : les deux clapets d'aspiration sont ouverts complètement, donc il n'y a pas de production d'air comprimé **[3]**.

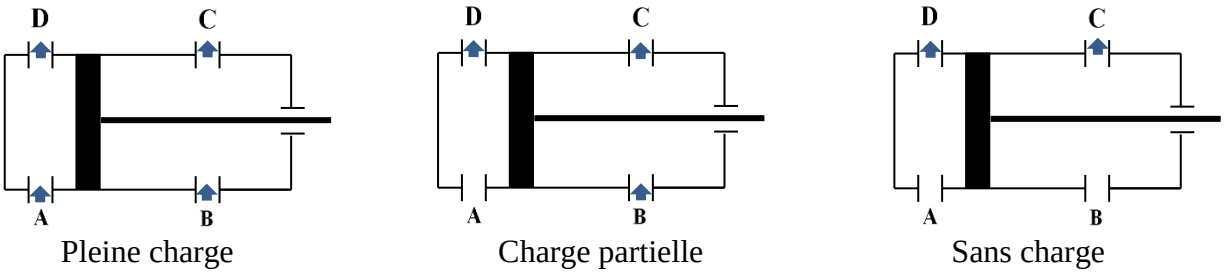


Figure 16 : *Diminution de la charge* **[16]**.

Ce réglage assure une consommation énergétique à peu-près proportionnelle à la charge réduite.

Le circuit de décharge agit sur les clapets d'aspiration pour obtenir le fonctionnement à 0%, 50% et 100%. L'action sur ces clapets nécessite un air comprimé à 7 bars (**Figure 17**).

Une partie de la pression de sortie du compresseur est utilisée pour aboutir à ces 7 bars par le détendeur, cette pression qui est utilisé pour actionner les clapets de respiration soit pour l'effet avant ou l'effet arrière pour les pistons à double effet.

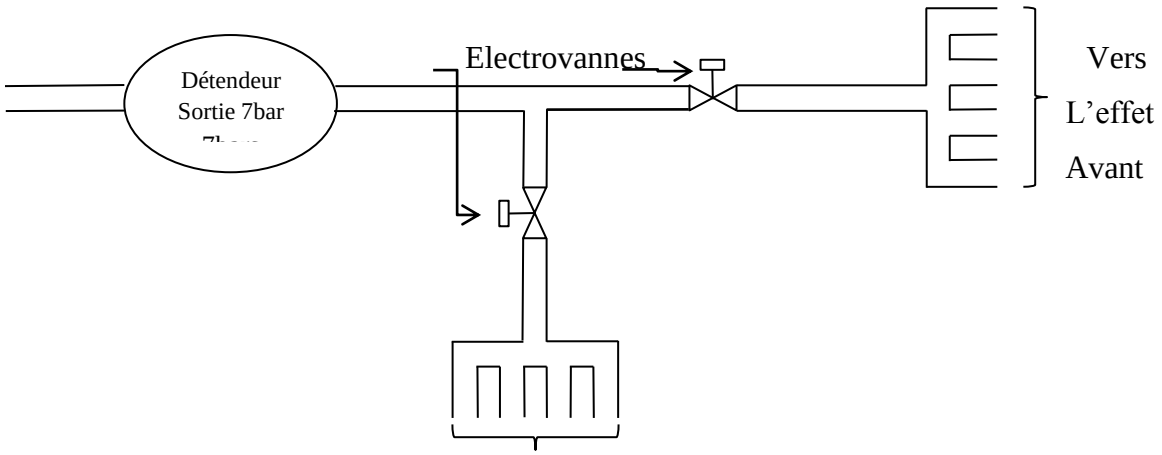


Figure 17 : *Éléments du circuit de décharge.*

Le compresseur est :

- A pleine charge, si aucune électrovanne n'est excitée.
- A demi-charge, si une seule électrovanne est excitée.
- Sans charge, si les deux électrovannes sont excitées.

II.4 Problématique

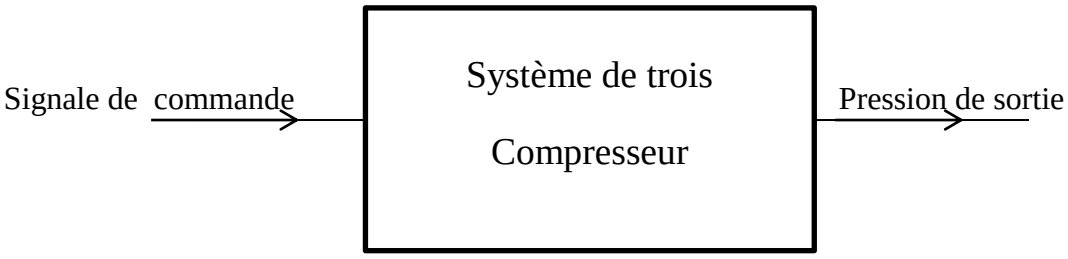


Figure 18 : *Illustration du système.*

La problématique peut être résumée comme suit :

- ✓ A partir des trois compresseurs mise en parallèle, nous devons obtenir une pression (P), vu que nos compresseurs ne fonctionnent pas tout le temps à pleine puissance. Nous devons mettre à vide, à demi-charge ou à pleine charge les compresseur pour différentes valeurs de pression dans le réseau d'air comprimé.
- ✓ Arrêter tout compresseur mis à vide après un certain temps.
- ✓ Répartir d'une manière équitable la durée de travail de chaque compresseur et l'afficher.

La solution pour avoir notre pression tout en économisant de l'énergie peut être résumée dans le tableau suivant :

Tableau 2 : *Variation du fonctionnement des compresseurs en fonction de la pression***.**

Intervalle de pression	C3	C2	C1
0 < P < P1	100%	100%	100%
P1 <= P < P2	50%	100%	100%
P2 <= P < P3	0%	100%	100%
P3 <= P < P4	0%	50%	100%
P4 <= P < P5	0%	0%	100%

P5 <= P < P6	0%	0%	50%
P6 <= P < Pn	0%	0%	0%

Pour notre solution, chaque compresseur « i » (i = 1,2 ou 3), est commandé par deux électrovannes (EV1Ci et EV2Ci) qui nous permettent de réaliser la fonction montrée dans le **Tableau 2**.

Une commande de marche et d'arrêt (AMCi, i=1,2 ou 3) permet d'économiser de l'énergie en arrêtant automatiquement le compresseur s'il fonctionne à vide pendant une période de temps donnée (environ 15 minutes) en les gardant en réserve pour les redémarrer en cas de reprise de la demande. Nous avons aussi une sortie pour ce compresseur qui est la pression envoyée vers le réseau.

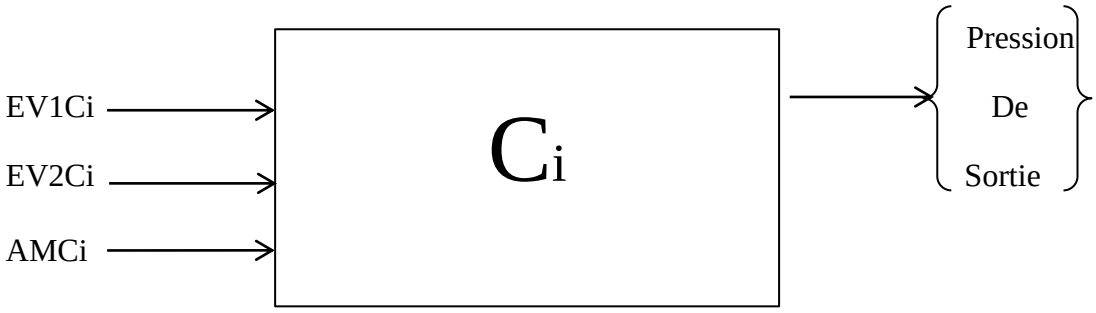


Figure 19 : *Entrées/ sortie d'un compresseur.*

Pour assurer un bon fonctionnement et un bon rendement, il faut des interventions sur chaque compresseur selon la fréquence de son fonctionnement, dans le **Tableau 3**, qi'il nous donne les possibilités d'intervention sur un compresseur :

Tableau 3 : *Entretien du système d'air comprimé* **[17]**.

Fréquence ou heures De fonctionnement	**Action**
Toutes les 3 000 heures	• Vérifier et remplacer les éléments filtrants • Vérifier/changer les éléments filtrants des reniflards de carter • Vérifier/nettoyer les robinets de purge des condensats

	• Contrôler l'état des accouplements d'arbres et de leurs éléments d'assemblage • Lubrifier les paliers des moteurs avec la quantité et le type spécifiés de graisse lubrifiante
Toutes les 15 000 heures	• Essayer tous les dispositifs de sécurité • Contrôler et nettoyer les échangeurs de chaleur • Vérifier et nettoyer les robinets d'extraction, clapets anti-retour, tuyauteries entre étages, supports antivibratoires • Contrôler et nettoyer les clapets anti-retour des carters d'huile et les filtres à tamis

D'après le **Tableau 2**, chaque nombre d'heures de fonctionnement nécessite des interventions spécifiques.

Dans notre solution, il faut prendre en compte les horaires de maintenance, et pour cela il faut afficher le nombre d'heures de fonctionnement de chaque compresseur pour créer une synchronisation avec le service de maintenance.

Chaque fois qu'un compresseur mis en première position, atteint son seuil de fonctionnement, il doit être remplacé par un autre qui le précède à son tour, remplacé par le compresseur restant et ainsi de suite.

Chapitre III

Simulation du programme

III.1 Introduction

Après avoir vu les composantes du processus à commander dans les chapitres précédents ; Il est possible de passer à l'étape final de l'automatisation, qui est la construction du modèle de Stateflow pour la commande et la simulation.

III.2 Le schéma du programme

Le modèle physique est représenté dans la **Figure 20** celui-ci a été réalisé selon un cahier de charge qui porte une solution à notre problématique.

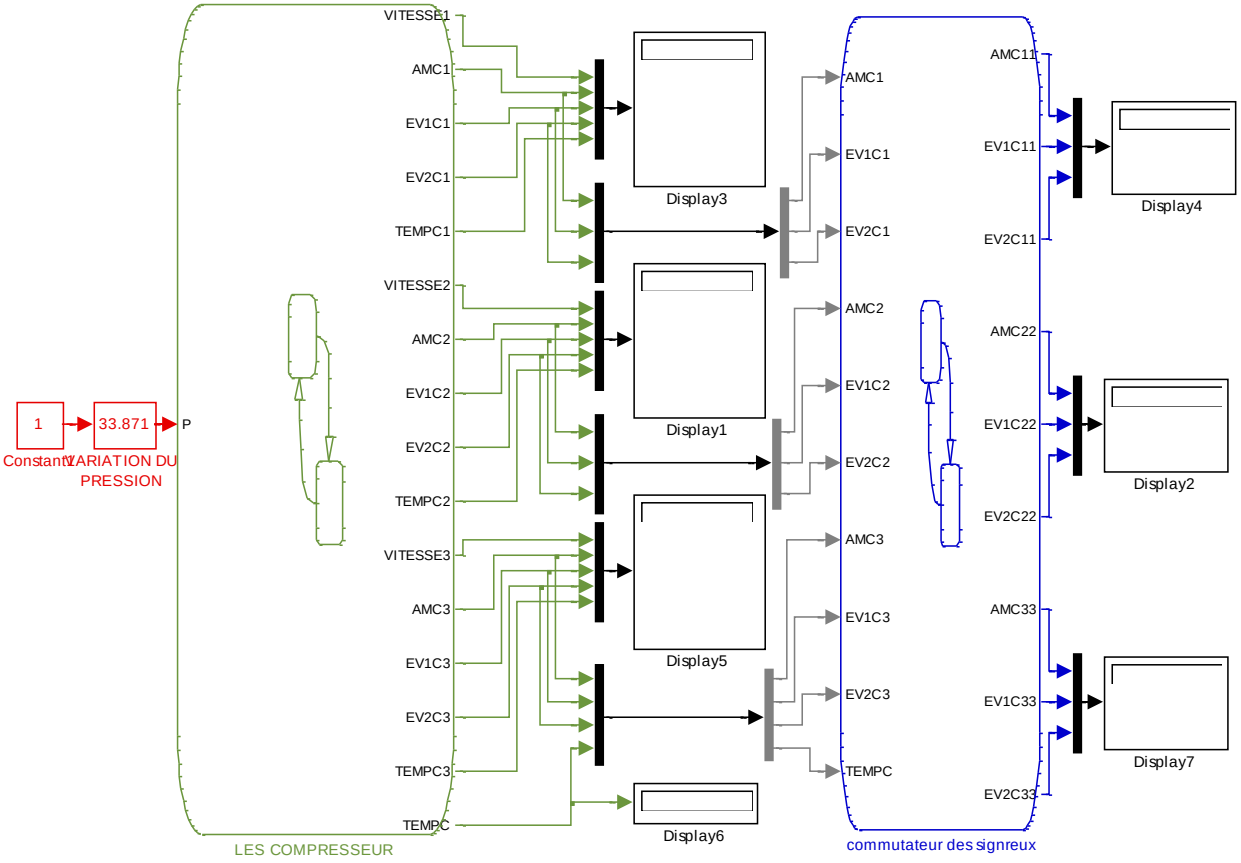


Figure 20 : *Schéma du programme de commande.*

Ce schéma est constitué de deux blocs en Stateflow, qui fonctionne en parallèle.

Le premier porte le nom : « les compresseurs » qui est destinés à délivrer des signaux de commande et le temps de fonctionnement des compresseurs. Le deuxième bloc porte le nom : « commutateur des signaux » celui-ci est destiné à la permutation des signaux de

commande de chaque compresseur d'une façon périodique, après 1500 heures de fonctionnement.

III.2.1 Le variateur de pression

Pour une variation de la pression de 0 à 50 bar, on aura une variation de tension délivré par le capteur sur une plage de 0 à 10 V.

Pour garder la pression dans la machine de Stateflow correspondante à la pression réelle, ça veut dire que les pressions associées aux transitions sont multipliées par un facteur de **5**, en utilisent un gain.

Pour la simulation, le capteur fourni les images de pression suivantes :

Tableau 4 : *Les pressions de travail.*

	Pression (bars)	Image de pression fournie par le capteur (Volt)
P1	34	6.8
P2	35	7
P3	36	7.2
P4	37	7.4
P5	38	7.6
P6	39	7.8

La pression varie entre [0 bar, 40 bar], dans cette étude les intervalles sont partagés en six. Pour chaque intervalle le compresseur à un état de fonctionnement bien spécifique (MAX, MOY, VID). Les résultats sont illustrés dans le **Tableau 5.**

Tableau 5 : F*onctionnement des compresseurs en fonction de la pression.*

Intervalle de pression	C3	C2	C1
0 < P < 34	100%	100%	100%
34 <= P < 35	50%	100%	100%
35 <= P < 36	0%	100%	100%
36 <= P < 37	0%	50%	100%
37 <= P < 38	0%	0%	100%
38 <= P < 39	0%	0%	50%
39 <= P < 40	0%	0%	0%

L'association d'une plage de pression bien spécifique à chaque transition est représentée comme un évènement qui fournit les motivations de transition entre les états ; cette pression est représentée comme une condition de changement d'état de fonctionnement. **Figure 21**.

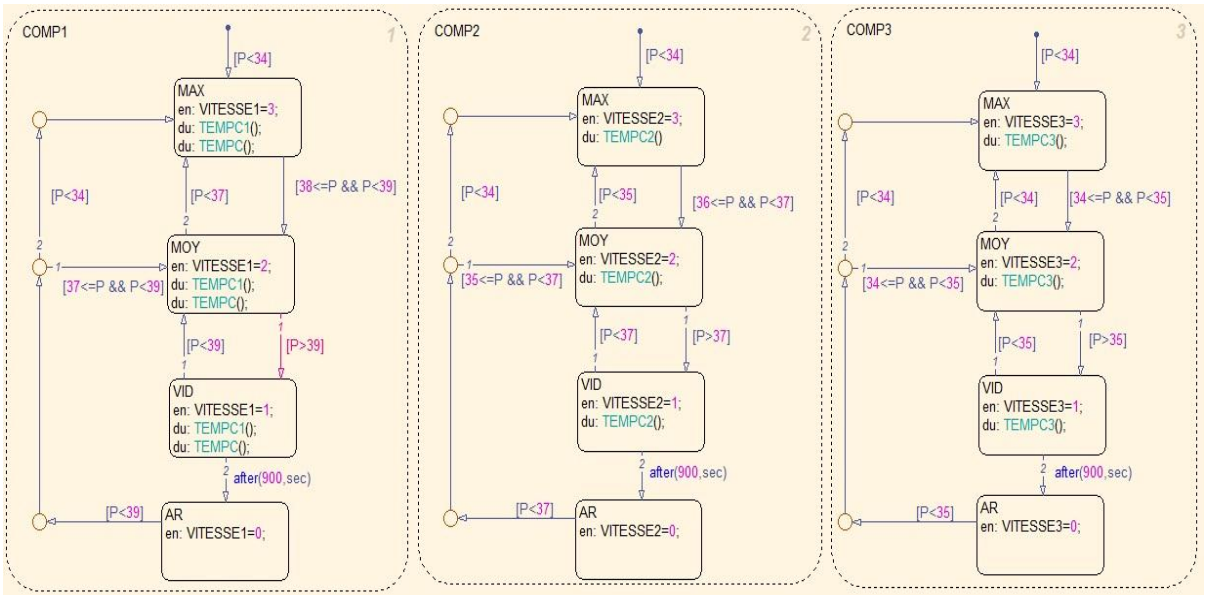


Figure 21: *Les évènements associent à la transition***.**

Le fonctionnement des compresseurs à plaine charge, à demi charge ou sans charge (MAX, MOY, VID) est assuré par l'activation ou la désactivation des électrovannes. **Tableau 6.**

Tableau 6 : *Mode de fonctionnement des électrovannes.*

Les 2 Electrovannes	Fonctionnement MAX	Fonctionnement MOY		Fonctionnement VID
EV1	0	1	0	1
EV2	0	0	1	1

Pendant que l'état « *MarcheAre*t » est activé (**Figure 22**), il délivre des signaux de commande qui s'occupe de la marche et de l'arrêt (AMCi, i=(1, 2, 3)) des compresseur et activation ou la désactivation des électrovannes (EV1Ci, EV2Ci/ , i=(1, 2, 3)).

L'exemple si-dessous illustre le fonctionement de la commande *in(COMP1.MAX)* .

Exmeple :

La fonction (*du : AMC1 = in(COMP1.MAX)*1 + in(COMP1.MOY)*1 + in(COMP1.VID)*1 + in(COMP1.AR)*0;*).

La sortie (*AMC1*=1) : Durant (*du)* l'état *COMP1* qui represente le compresseur placé en premiere position activé se trouve aux sous états (MAX, MOY, VID).

Et la sortie (AMC1=0) : Durant (*du),* l'état *COMP1* activé et se trouve au sous état (AR).

```
MarcheAret
du : AMC1  = in(COMP1.MAX)*1 + in(COMP1.MOY)*1 + in(COMP1.VID)*1 + in(COMP1.AR)*0;
du : AMC2  = in(COMP2.MAX)*1 + in(COMP2.MOY)*1 + in(COMP2.VID)*1 + in(COMP2.AR)*0;
du : AMC3  = in(COMP3.MAX)*1 + in(COMP3.MOY)*1 + in(COMP3.VID)*1 + in(COMP3.AR)*0;
du : EV1C1 = in(COMP1.MAX)*0 + in(COMP1.MOY)*1 + in(COMP1.VID)*1 + in(COMP1.AR)*0;
du : EV2C1 = in(COMP1.MAX)*0 + in(COMP1.MOY)*0 + in(COMP1.VID)*1 + in(COMP1.AR)*0;
du : EV1C2 = in(COMP2.MAX)*0 + in(COMP2.MOY)*1 + in(COMP2.VID)*1 + in(COMP2.AR)*0;
du : EV2C2 = in(COMP2.MAX)*0 + in(COMP2.MOY)*0 + in(COMP2.VID)*1 + in(COMP2.AR)*0;
du : EV1C3 = in(COMP3.MAX)*0 + in(COMP3.MOY)*1 + in(COMP3.VID)*1 + in(COMP3.AR)*0;
du : EV2C3 = in(COMP3.MAX)*0 + in(COMP3.MOY)*0 + in(COMP3.VID)*1 + in(COMP3.AR)*0;
```

Figure 22 : *Les signaux de commande selon les états de fonctionnment des compresseurs.*

III.2.2 Calcule du temps de fonctionnement

Pour calculer le temps de fonctionnement de chaque compresseur, on utilise la fonction graphique qui porte le nom (TEMPCi \ i=(1, 2, 3)) (**Figure 23**).

La fonction « R=TMPC1 » est une fonction graphique de type « *If-Elesif* » qui s'occupe de générer des signeaux qui pilote les compresseurs. Elle sera exécutée pendant l'activation de l'état parent «COMP1» et le sous-état « MAX».

L'exemple si-dessous montre l'explication de l'exécution de la fonction Graphique.

« Si TEMPC1<1500

TEMPC1++

sinon TEMPC1=0 »

Un double clique sur la fonction « R=TEMPC1» ouvre la fenêtre de la fonction Graphique (**Figure 23**).

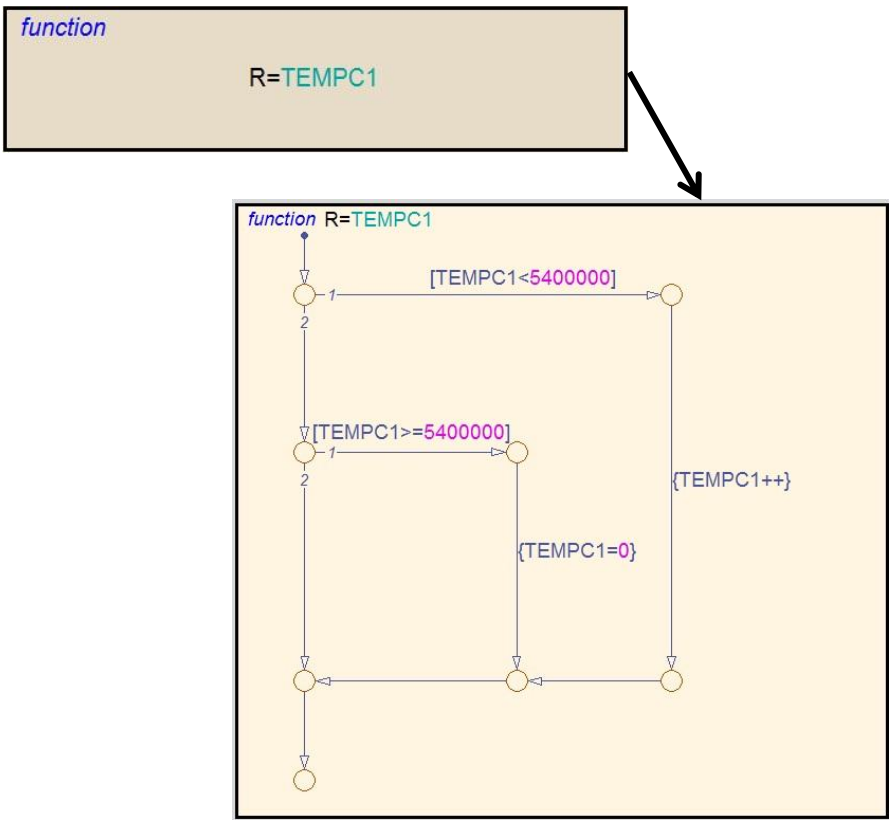


Figure 23 : *La fonction Graphique qui calcule le temps.*

III.2.3 La comparaison du temps de fonctionnement

Pour la comparaison du temps de fonctionnement des compresseurs, il suffit de se basé sur le compresseur placé en première position, car son fonctionnement à plein temps occupe un intervalle de pression plus élevé.

L'exécution de la fonction « TEMPC» est dû à l'activation de l'état « comparateur » (**Figure 24**).

Le bloc (commutateur des signaux) s'occupe de la permutation des signaux pour activer ou désactiver les compresseurs et les électrovannes, cette permutation est réalisée à l'aide d'une fonction Graphique « TEMPC » (*If-Elesif-Elseif-Else*).

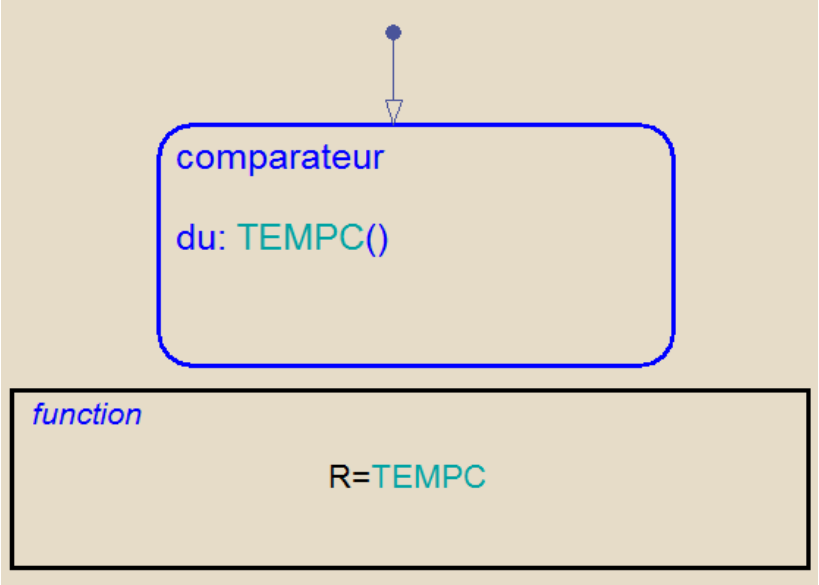


Figure 24 : *Le fonctionnement du modèle pour comparer le temps de fonctionnement.*

Une double clique sur la fonction (TEMPC) permet l'accès à la fonction Graphique (**Figure 25**)

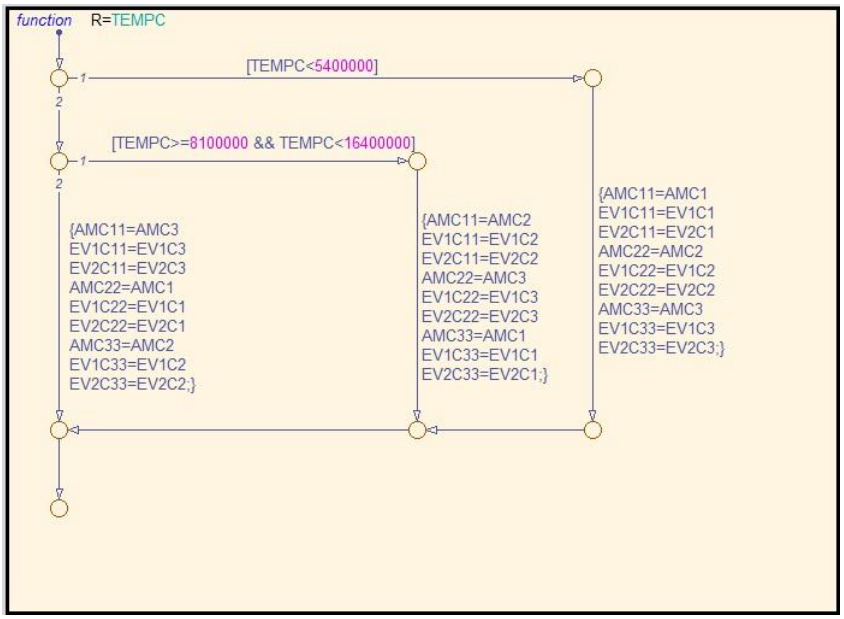


Figure 25 : *La permutation des signaux de commande.*

Et le **Tableau** 7 nous éclaire sur ces permutation :

Tableau 7 : *La permutation des signaux de commande.*

TEMPC< 1500h	[TEMPC>=1500 && TEMPC< = 3000h]	TEMPC> 3000h
AMC11=AMC1	AMC11=AMC2	AMC11=AMC3
EV1C11=EV1C1	EV1C11=EV1C2	EV1C11=EV1C3
EV2C11=EV2C1	EV2C11=EV2C2	EV2C11=EV2C3
AMC22=AMC2	AMC22=AMC3	AMC22=AMC1
EV1C22=EV1C2	EV1C22=EV1C3	EV1C22=EV1C1
EV2C22=EV2C2	EV2C22=EV2C3	EV2C22=EV2C1
AMC33=AMC3	AMC33=AMC1	AMC33=AMC2
EV1C33=EV1C3	EV1C33=EV1C1	EV1C33=EV1C2
EV2C33=EV2C3	EV2C33=EV2C1	EV2C33=EV2C2

III.2.4 Programmation de l'arrêt du compresseur

Lorsque un compresseur est sans charge, il reste en fonctionnement pendant (15 min ou 900sec) ensuite il s'arrête grâce au calcul effectué par la fonction (after (900,sec)), s'il n'est y a pas de reprise de la demande d'air comprimé (**Figure 26**).

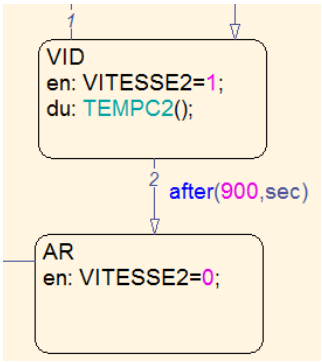


Figure 26 *: Arrêt programmé.*

Remarque :

Lorsque la sortie AMCi (i= 1, 2 ou 3) se met à zéro (0) le compresseur correspondant est à l'arrêt, lorsque la demande en air comprimé est augmentée, cette sortie se met à (1) et le compresseur sera actif de nouveau.

III.3 Simulation

La fin de construction du modèle est suivie par une simulation, en commençant par la configuration.

- ✓ Dans le menu « Simulation\configuration paramètre », on choisit « *inf* » pour « *stop time* ». Cela signifie que la simulation continu, tant qu'on ne décide pas du contraire.
- ✓ Dans le menu « Tools\Open simulation target », on vérifie que la case « *Enable debugging\animation* » est cochée.
- ✓ Dans le menu « Tools\Debug », section animation, on vérifie que la case « *enable* » est cochée et que le délai est fixé ; par exemple a (0.6), cela veut dire que le changement d'action se fait à chaque (0.6 sec).
- ✓ Vérifier que la pression est inférieur à (34 Bar), ensuite passer au menu « Simulation\Start », ceci lance la simulation du modèle mais pas à temps réel.
- ✓ Pour arrêter la simulation on clique « Simulation\Stop ».

III.3.1 Les résultats de la simulation

Faire varier la pression après l'exécution du la machine Stateflow, nous permet d'obtenir quatre différents cas :

- ➢ État MAX,
- ➢ État MOY,
- ➢ État VID,
- ➢ État AR.

La lecture des displays se fait ainsi (**Figure 27**):

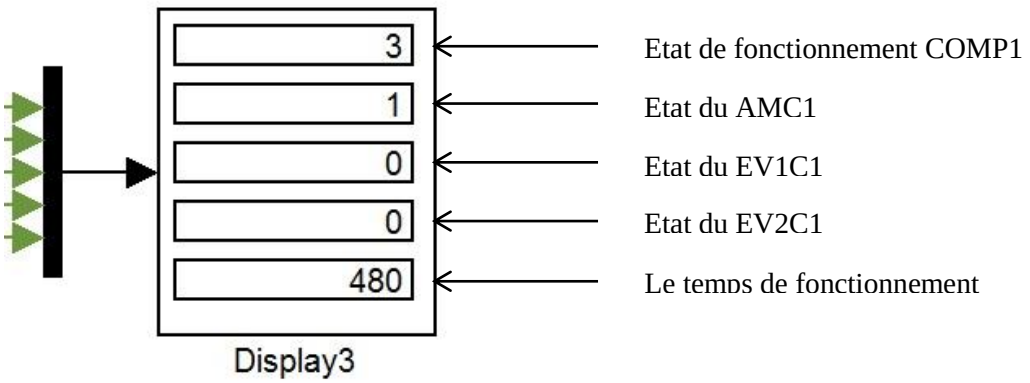


Figure 27 : *La lecture des Display.*

L'état de fonctionnement varie entre [0 et 3] et chaque nombre dans cet intervalle il correspond à un état bien spécifique, les différents états sont détaillés dans le tableau suivant :

Tableau 8 : *Etat de fonctionnement.*

	Etat de fonctionnement
3	MAX
2	MOY
1	VID
0	AR

III.3.2 La simulation a temps réel

Pour faire la simulation à temps réel il faut que « Real-Time Windows Target » soit installé, et la configuration des paramètres de simulation sera prise par défaut.

Sinon dans « Workspace » on exécute la commande suivante « rtwintgt –install » pour l'installation de « Real-Time Windows Target » et pour la configuration par défaut on exécute la commande suivante « rtwinconfigset('nom de fichier ') ».

Ensuite :

- Aller dans le menu « Tools », puis « Code Generation » on clique sur « Build Model » pour générer le code « C ».
- Aller dans le menu « simulation », puis on clique sur « Connect To Target ».
- Aller dans le menu « simulation », puis « Start real-time Code ».

Ensuite la simulation à temps réel sera lancée, et on peut suivre l'évolution de notre commande a temps réel.

Chapitre IV

Implémentation de la commande

IV.1 Présentation du processus

Dans le Chapitre précédant, on a vu comment crée le modèle Stateflow et sa simulation sous l'environnement Simulink. Dans ce chapitre on va voir comment faire une implémentation de cette commande (**ANNEXE 1**).

IV.2 Implémentation de la commande

Pour l'implémentation de notre commande il faut disposer du matériel suivant :

- Un Pc qui contient les différents logiciels: Matlab v7, Simulink, ControlDesk.
- Une carte *dSPACE "RTI1104"*.
- Un moteur à courant continu.
- Une source de tension variable.

IV.2.1 Code Generation « Real Time Workshop »

C'est un outil de Matlab qui permet de générer automatiquement un code C à partir du diagramme schéma blocs de Simulink. De plus, il permet de produire un code optimisé, portable et personnalisable d'après les modèles générés par Simulink.

Il supporte tout type de systèmes, c'est-à-dire continus, discrets ou hybrides. Il permet la simulation du système étudié en mode externe. Il permet de connecter le modèle Simulink aux matériels physiques. De nombreuses cibles sont possibles. **[18]**, **[19]** :

- XPC target, création d'un exécutable temps réel sur PC cible.
- *Les cartes dSPACE.*
- Les carte d'acquisition « ADVANTECH ».

IV.2.2 Présentation de la carte *dSPACE "RTI1104"*

La carte DSP utilisée est la *dSPACE "RTI1104"*. Le processeur principal est un MPC8240, avec un cœur Power PC 603e et une horloge interne à 250 MHz. Il a une capacité mémoire de 8 Mo en Flash et de 32 Mo en SDRAM **(Annexe 2)**.

Il dispose de 8 convertisseurs analogiques numériques (4 en 16 bits, 4 en 12 bits), de 8 convertisseurs numériques analogiques (CNA) de 16 bits qui peuvent délivrer une tension analogique comprise entre -10V et +10V, d'une liaison série, de 2 codeurs incrémentaux, de

20 entrées-sorties numériques, d'une DSP esclave (TMS320F240) et de 3 timers (32 bits) qui peuvent fonctionner d'une manière indépendante.

Pour programmer le DSP, il faut réaliser tout d'abord un schéma dans l'environnement Simulink de Matlab. Il est préférable de se placer dans son répertoire de travail pour élaborer le fichier Simulink car la compilation génère de nombreux fichiers contenus dans un nouveau répertoire généré à chaque étape de compilation **[20]**.

IV.2.3 L'implémentation de la commande et la compilation

Après l'acquisition du matériel nécessaire pour l'implémentation, il nous reste à faire quelque changement sur le programme.

Dans "Simulink Library Browser" on trouve une librairie nommée "dSPACE RTI1104" dans laquelle on peut choisir les composants utilisés pour communique avec la carte de DSP. Un exemple de blocs utilisés est présenté sur la **Figure 28**. En faisant un double clic sur chaque bloc, on peut choisir le champ dans lequel on souhaite écrire ou lire les données.

Une fois le schéma Simulink terminé et sauvegardé, l'étape suivante consiste à générer le code associé au schéma Simulink et à le transférer dans la DSP. Si la compilation a réussi, le chargement du programme et son exécution dans la DSP se font automatiquement.

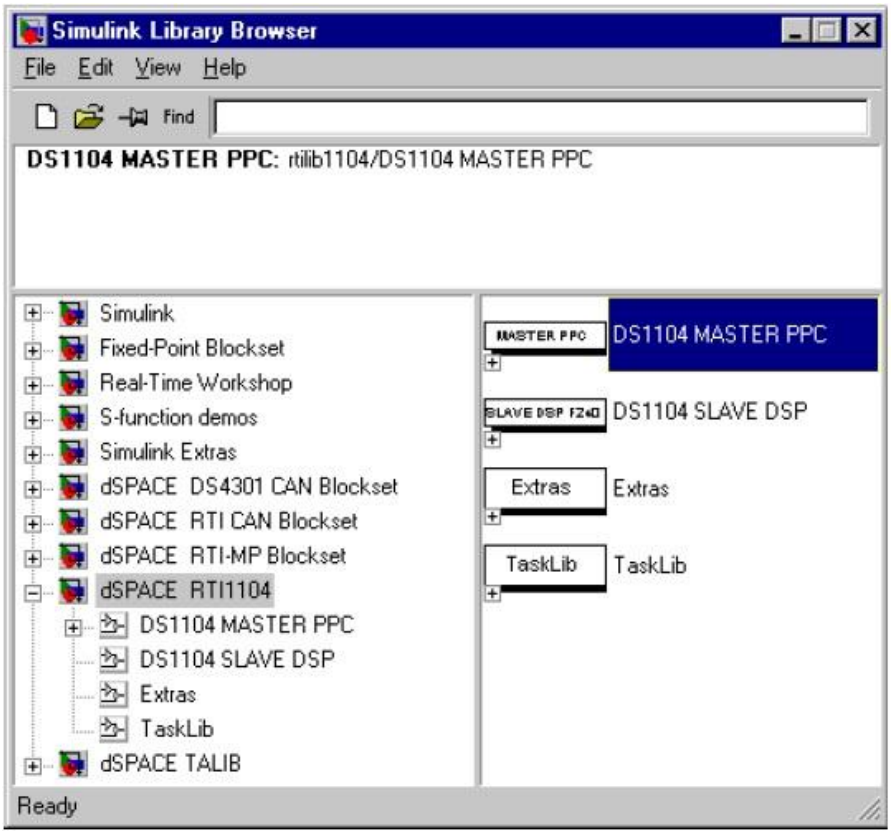
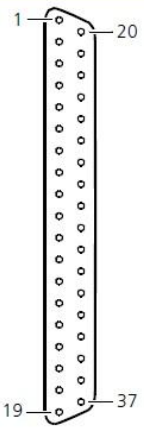


Figure 28 : *Simulink Library Browser.*

Et comme nos signaux sont de nature numérique, donc on utilise le port des entrées-sorties numériques « I/O » (**Figure 29**).



Connector (CP18)	Pin	Signal	Pin	Signal
	1	GND		
	2	SCAP1	20	GND
	3	SCAP3	21	SCAP2
	4	GND	22	SCAP4
	5	ST2PWM	23	ST1PWM
	6	GND	24	ST3PWM
	7	SPWM1	25	GND
	8	SPWM3	26	SPWM2
	9	SPWM5	27	SPWM4
	10	SPWM7	28	SPWM6
	11	SPWM9	29	SPWM8
	12	GND	30	GND
	13	GND	31	GND
	14	GND	32	GND
	15	GND	33	GND
	16	SSIMO	34	SSOMI
	17	SCLK	35	SSTE
	18	VCC (+5 V)	36	GND
	19	VCC (+5 V)	37	GND

Figure 29: *Port des entrées-sorties numériques « I/O ».*

Pour compiler un schéma Simulink :

- Aller dans le menu « simulation », puis « simulation parameters ».
- Dans l'onglet « Solver », fixer « Type » à « fixed-step » puis dans « fixed step » à « Te » (période d'échantillonnage choisie). Mettre « stop time » à « inf ».
- Dans l'onglet « Code Generation » puis « System target file » charge le fichier « rti1104.tlc ».
- Aller dans le menu « simulation », puis « Star ».

La compilation génère de nombreux fichiers dont deux sont particulièrement importants :

- ***.sdf** listant tous les paramètres du schéma bloc, utilisés sous Control Desk pour faire le lien entre le firmware et l'IHM (Interface Homme Machine).
- ***.ppc** qui est le firmware chargé dans la carte DSP.

IV.2.4 Présentation de Control Desk

Control Desk est une interface qui permet de visualiser en temps réel les différentes variables du fichier développé sous Simulink et de modifier également les paramètres définissant le mode de fonctionnement des blocs constituant le schéma Simulink. La visualisation de variables ou de signaux et la modification de paramètres sont possibles par l'intermédiaire d'instruments graphiques que l'on sélectionne. Un exemple de construction d'un écran graphique est présenté ci-après (ou IHM : Interface Homme Machine).

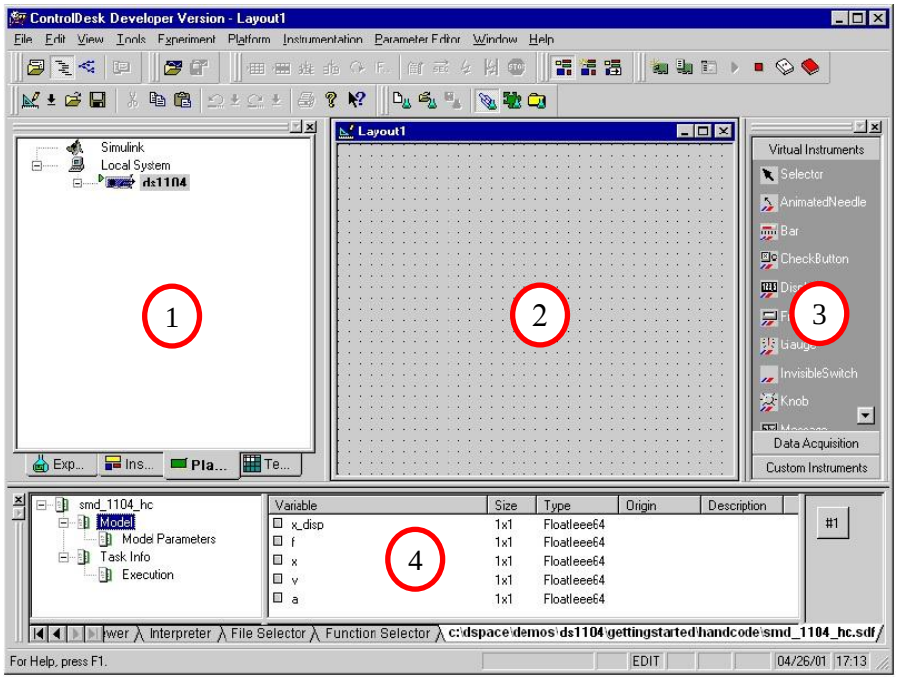


Figure 30 : *L'interface graphique de Control Desk.*

Les différentes fenêtres rencontrées sur Control Desk sont :

1 – la fenêtre de navigation (Navigator).
2 – l'aire de travail contenant la layout (visualisation graphique).
3 – le choix d'instrument (instrument selector).
4 – la fenêtre d'outils (tool window).

- ***Remarque*** : un "layout" est une interface graphique à laquelle on peut ajouter divers instruments dans le but de visualiser ou de modifier en temps réel les différentes variables du projet **[22]**.

IV.2.4.1 Démarrer un nouveau projet sous Control Desk

Sous Control Desk, un projet s'appelle un « experiment ».

Pour créer un nouveau projet en allant dans « fichier » puis « new experiment ». On insérera ensuite la « layout » à notre projet (« fichier » => « new » => « layout »). On ajoutera enfin tous les instruments de visualisation et de modification des variables. Il faut pour cela d'abord, récupérer le fichier ayant pour extension « *.sdf » qui est créé par Simulink au moment du « build », puis on clique sur « model root » et on fait par un click gauche sur la variable, glisser le « out » (lecture de la variable) ou le « value » (modification de la variable) vers l'instrument visuel qui doit y être relié.

Pour lancer la simulation, cliquée sur l'icône « Animation Mode  ». On est alors en mode simulation (ou animation) **[22]**.

IV.4 Les solutions envisagées

De nous jour, l'automaticien cherche des solutions plus durable et moins couteuse pour commander les processus industriel.

Dans ce chapitre on donne quelque solution pour résoudre ce problème :

IV.4.1 Real - Time Windows Target

Dans le cas d'un environnement WINDOWS, ce petit noyau est essentiel, son rôle est primordial, il assure l'exécution de notre commande en temps réel. Cet environnement permet de compiler le schéma Simulink utilisé en temps-réel et de piloter la carte E/S.

Cette architecture permet de faire les simulations et les expérimentations sur le même PC et de manière très rapide, le tout pour un coût relativement limité. L'inconvénient est que l'on peut difficilement faire des expériences qui ont un temps d'échantillonnage très petit, cela mène le système d'exploitation « Windows » à une exécution très lente et parfois des arrêts ; ce qui pose des difficultés pour gérer des processus qui ont besoin d'une continuité de la commande **[21]**.

A cause des certaine inconvénients de logiciel « Real - Time Windows Target » on propose une autre solution qui est :

IV.4.2 Xpc Target

Xpc target est une solution hôte-cible pour développer, tester et analyser des systèmes temps réel sur PC standard. C'est un environnement entièrement intégré à Matlab qui utilise un PC cible, connecté au PC hôte pour lancer des applications temps réel. Les connexions utilisées peuvent être deux types : soit par port série RS 232, soit par connexion TCP/IP.

Dans cet environnement, on utilise un PC hôte, muni de Matlab et Simulink, pour créer des modèles de diagrammes. Après la création de ces modèles, l'application peut être lancée.

Xpc target permet d'ajouter des blocs d'entrée / sortie au modèle construit, et d'utiliser le PC hôte, muni des outils Code Generation « Real Time Worshop » (RTW) et d'un compilateur C, pour créer le code C exécutable. Le code C est ensuite téléchargé du PC hôte vers le PC cible, puis lançant l'application temps réel. Après que ce code exécutable a été téléchargé, l'application cible peut être lancée et testée en temps réel **[18].**

IV.4.2.1 Utilisation de Xpc Target

Xpc target est une boîte à outils qui permet de développer une application externe sur un PC cible dépourvu de système d'exploitation. Un CD/DVD de boot est générée par Xpc target. Ce CD/DVD contient les éléments nécessaires pour la communication avec le PC hôte, ainsi que l'application qui sera exécutée sur ce PC. On pourra lancer l'exécution et modifier les paramètres en ligne à partir du PC hôte.

Xpc target offre d'autres solutions comme la communication à partir d'une page web, mais pour cela il on doit une liaison TCP/IP.

Le mode externe de « Code Generation » permet à deux PC cible et l'hôte de communiquer. Le PC hôte est celui sur lequel Matlab et Simulink sont exécutés. Le PC cible est celui sur lequel l'exécutable créé par « Code Generation » est exécuté **[18]**.

IV.4.3 Carte d'acquisition: ADVANTECH PCI 1711

Dans cette étude on a utilisé la carte *dSPACE "RTI1104"* qu'est très utile mais son prix dépasse les 5000 Euro, on propose une carte qui ne coute que dix fois moine cher qui peut résoudre le problème de communication avec les compresseurs ou le processus à commander.

C'est une carte d'acquisition universelle qui est installée sur le port PCI du PC de commande et dispose de connecteurs extérieurs pour des entrées/sorties analogiques et digitales, et son environnement de développement est Matlab.

Figure 31: *Carte d'acquisition ADVANTECH PCI 1711*.

Ces principales caractéristiques sont :

- La fonction Plug & play ;
- 16 entrées analogiques configurables simples (single-ended) ou source flottante ;
- Convertisseurs A/D industriels normalisés à approximations successives ;
- 12 bits utilisés pour la conversion des entrées analogiques ;
- La fréquence maximale d'échantillonnage est de 100KHz ;
- Gamme des entées analogiques est programmables et contrôlable par software ;
- Chaque canal à sa gamme individuelle stockée dans la RAM de la carte ;

- 2 sorties analogiques (convertisseur D/A) ;
- 16 canaux d'entées digitales ;
- 16 canaux de sorties digitales ;
- Un compteur/timer programmable ;
- Scanne automatique des gains/canaux.

Son principal rôle dans notre application, nous permettra la commande d'un système continu à travers un PC, en convertissant les signaux analogiques eux signaux et vice-versa **[24], [23]**.

Dans notre cas, on utilise :

- 09 sorties numériques (convertisseur D/A) pour délivrer la tension de commande.
- Une entrée analogique (convertisseur A/D) pour récupérer l'image de la pression.

Conclusion Générale

Pour l'automatisation du fonctionnement des compresseurs, la logique câblée pourra résoudre le problème, cependant elle ne permet aucune modification du programme, ce qui va à l'encontre de l'entreprise qui est en permanente évolution. Il est impératif d'avoir des systèmes de commandes flexibles. La logique programmée permet des modifications sur le programme de commande en fonction des besoins de l'entreprise,

Les cartes Analogique\Digitale (ADC) pourront résoudre le problème de l'automatisation du côté de flexibilité et de côté de faible coût.

La gestion des différents compresseurs a pour objectif d'assurer l'approvisionnement en air les différentes souffleuses, tout en garantissant une consommation optimale d'énergie et une meilleur synchronisation avec le service de maintenance, tout en programmant des maintenances périodiques et ainsi éviter l'interruption de la production. Le programme qui est conçu pour les trois compresseurs permet l'extension du système de production de l'air comprimé, il suffit seulement de former des groupes en trois compresseurs.

La solution apportée, pourrait être une réponse acceptable à la problématique, alors espérant qu'elle sera prise en compte par l'entreprise Cevital et que cette étude sera une référence.

Bibliographie

[1] J-C.Bossy, D.Merat, « *Automatisme appliqué* », Edité par : EL éducalivre.

[2] P.CRARE, I.KACEM, « *Ce qu'il faut savoir sur les automatismes* », Editeur : Ellipses, PP 9-65, ISBN : 978-2-7298-3659-7.

[3] H.AZAIEZ, « *Introduction aux A.P.I* », 2007.

[4] W.Bolton, « *Les Automates Programmables Industriels* », Editeur : DUNOD, PP 1-17, 2009, ISBN 978-2-10-054705-0.

[5] «*Les Automates Programmables Industriels* », Edite par l'entreprise : la Briquerie Niedercorn LT « la Briquerie » 57100 THIONVILLE.

[6] B.SCHNEIDER et A.BEURET « *Automatisation Industrielle* », Edité par : institut d'Automatisation Industriel, 2006.

[7] P.CRARE, I.KACEM, « *Ce qu'il faut savoir sur les automatismes* », Editeur : Ellipses, PP 9-65, ISBN : 978-2-7298-3659-7.

[8] Help « Stateflow ».

[9] K.BENMANSOUR, « *Réalisation d'un banc d'essai pour la Commande et l'Observation des Convertisseurs Multicellulaires Série : Approche Hybride* », Thèse Doctorat, Université de Cergy Pontoise, 2009.

[10] N.Martaj, « *Matlab R2009, Simulink et Stateflow pour Ingénieurs, Chercheurs et Etudiants* », Editeur : Springer, 2010, ISBN : 978-3-642-11763-3.

[11] Document ADEME/DABEE/Département Industrie et Agriculture « *Compresseurs D'air* », 2006.

[12] T. DESTOOP, « *Compresseurs volumétriques* », Technique de l'ingénieur, traite de génie mécanique, pp. B 4 220 1 - B 4 220 29.

[13] « documentation technique Compresseur atlas Copco Crépelle, cde N : 2603- Cevital-Algérie-40P36-63LH 1O53W 7593, Classeur N 2 ».

[14] J.BOUTELOUP, « *Production de chaud et de froid* », *(tome 2)*, Collection Climatisation Conditionnement d'Air, Les Editions Parisiennes, 1997.

[15] Document ADEME/DABEE/Département Industrie et Agriculture « *Régulation De L'air Comprimé*», 2006.

[16] « *Compresseurs et Turbines* », Edité par : Energie, Mines et Ressources canada.

[17] N.LAHOUAZI, O.CHENITI, « *Etude et dimensionnement d'un compresseur pour la nouvelle raffinerie de sucre CEVITAL* », Mémoire de fin d'étude, 2007, Université Abderrahmane Mira – Bejaia.

[18] S.CLAEYS, O.RENIER, « *Développement De Commande à Temps Réel* », Rapport de projet de fin d'étude, Université Des Sciences Et Technologiques De Lile, 2005.

[19] M.BENDJEDIA, « *Synthèse d'algorithmes de commande sans capteurs de moteurs pas à pas et implantation sur architecture* programmable », Thèse de Doctorat, Université de France-Comte, 2007.

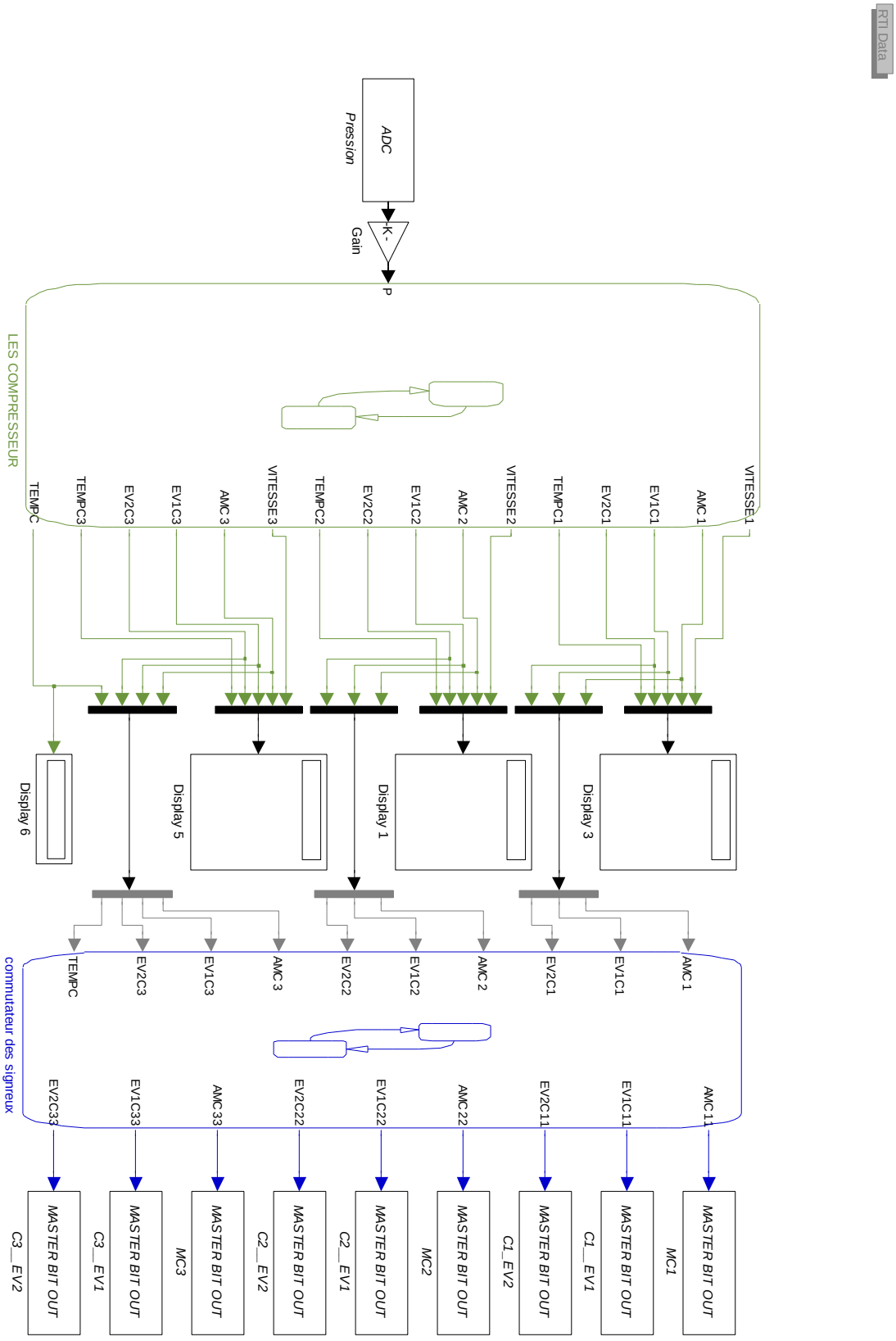
[20] M-M.ABDUSALAM, « *Structures et stratégies de commande des filtres actifs parallèle et hybride avec validations expérimentales* », Thèse doctorat, l'Université Henri Poincaré, Nancy-I, 2008.

[21] A.Hably, J.Dumon, « *Observation et commande par retour d'état d'un procédé de bacs Communicants* », Article, 2010.

[22] G.MOUSSOUAMI, P.AUPOIX, « *Etude du pendule inversé* », Rapport de Projet, Université de Caen Basse-Normandie, 2007.

[23] F.LAHOUAZI, « *Mise en œuvre d'une stratégie de commande neuro floue : Application à un pendule inversé* », Mémoire de Magister, Université Mouloud Mammeri, Tizi-Ouzou, 2011.

Annexe1


RTI Data
ADC
Pression
K -
Gain
P
LES COMPRESSEUR
TEMPC
TEMPC3
EV2C3
EV1C3
AMC 3
VITESSE3
TEMPC2
EV2C2
EV1C2
AMC 2
VITESSE2
TEMPC1
EV2C1
EV1C1
AMC 1
VITESSE1
Display 6
Display 5
Display 1
Display 3
commutateur des signreux
TEMPC
EV2C3
EV1C3
AMC 3
EV2C2
EV1C2
AMC 2
EV2C1
EV1C1
AMC 1
EV2C33
EV1C33
AMC33
EV2C22
EV1C22
AMC22
EV2C11
EV1C11
AMC11
MASTER BIT OUT
C3_EV2
MASTER BIT OUT
C3_EV1
MASTER BIT OUT
MC3
MASTER BIT OUT
C2_EV2
MASTER BIT OUT
C2_EV1
MASTER BIT OUT
MC2
MASTER BIT OUT
C1_EV2
MASTER BIT OUT
C1_EV1
MASTER BIT OUT
MC1

Annexe 2


ADCH1
ADCH2
ADCH3
ADCH4
ADCH5
ADCH6
ADCH7
ADCH8
DACH1
DACH2
DACH3
DACH4
DACH5
DACH6
DACH7
DACH8
Digital I/O
UART RS232
UART RS422 RS485
dSPACE
CLP1104
IO0
IO1
IO2
IO3
IO4
IO5
IO6
IO7
IO8
IO9
IO10
IO11
IO12
IO13
IO14
IO15
IO16 / INT1
IO17 / INT2
IO18 / INT3
IO19 / INT4
SSTE
SCLK
SSIMO
SSOMI
TXD
RXD
DCD (CTS)
RTS (-RTS)
DTR (TXD)
CTS (-CTS)
DSR (RXD)
VCC
SPWM1
SPWM2
SPWM3
SPWM4
SPWM5
SPWM6
SPWM7
SPWM8
ST1PWM
ST2PWM
ST3PWM
SCAP1
SCAP2
SCAP3
SCAP4
Pin0 1
Pin90 1
Index 1
Pin0 2
Pin90 2
Index 2

Résumé

Nous avons à commander un système de trois compresseurs identiques, qui sont des compresseurs alternatifs à piston de classe volumétrique, qui peuvent fournir une pression nominale de 40 bar, et leurs sorties sont reliées à une seul conduite.

Pour la régulation de la pression on à utiliser la méthode des circuits de décharge d'un compresseur pour diminuent la puissance en réduisant la capacité volumétrique interne du compresseur.

Pour l'utile de commande on a utilisé le l'environnement de Stateflow de Matlab.