Support de Cours: Automatisme et Informatique Industrielle

Dr. Okba KRAA

2018/2019

Table des matières

Chapitre 1

Généralités sur les systèmes automatisés et l'informatique industrielle

Sommaire

1.1 Définition

Un automatisme est un sous-ensemble des machines, destinée à remplacer l'action de l'être humain dans des taches en générale simples et répétitives, réclamant précision et rigueur. On

passe d'un système dit manuel, à un système mécanisé, puis au système automatisé. Dans l'industrie, les automatismes sont devenus indispensables : ils permettent d'effectuer quotidiennement les taches les plus ingrates, répétitives et, dangereuses. Parfois, ces automatismes sont d'une telle rapidité et d'une telle précision, qu'ils réalisent des actions impossibles pour un être humain. L'automatisme est donc synonyme de productivité et de sécurité.

1.1.1 Notions d'automatisme

-**Système** : Ensemble de composants issus de différentes technologies (mécanique, électronique, informatique..etc.) qui assemblés d'une certaine façon sont destinés à remplir une fonction.
- **Automatique :** C'est l'ensemble des sciences et des techniques utilisées dans la conception et la réalisation des Systèmes Automatisés (SA).
- **Procédé :** une machine, un ensemble de machines ou plus généralement un équipement industriel.
- **Chaines opérationnelles :** ensemble de procédés fonctionnent séquentiellement pour faire une taché spécifiques ou produire un produit.
- **Automatisation d'un procédé** : consiste à en assurer la conduite par un dispositif technologique (appelé automatisme). L'ensemble procédé + automatisme est appelé système automatisé. L'automatique aux confluents de l'électronique, l'électrotechnique et l'informatique.
- **Opérateur :** est la personne qui va faire fonctionner le système. -**Actionneurs :** convertissent l'énergie d'entrée disponible sous une certaine forme (électrique, pneumatique, hydraulique) en une énergie utilisable sous une autre forme.
- **Préactionneurs :** Composant contrôlable utilisé pour distribuer l'énergie aux actionneurs.
- **Informatique Industrielle :**Une branche de l'informatique appliquée qui couvre l'ensemble des techniques de conception et de programmation de systèmes informatisés à vocation industrielle qui ne sont pas des ordinateurs.
- **Cahier de charge :**est le descriptif fourni par l'utilisateur au concepteur de l'automatisme pour lui indiquer les différents modes de marches et les sécurités que devra posséder l'automatisme. Ce cahier des charges décrit le comportement de la partie opérative par rapport à la partie commande.

1.1.2 Exemple d'un automatisation industriel

La palettisation fait partie des systèmes de manutention qui se sont le plus développés au cours des trois dernières décennies. Elle consiste à grouper un certain nombre de colis sur un support : la palette ; l'opération de groupage est faite par un palettiseur comme montre la figure suivante.

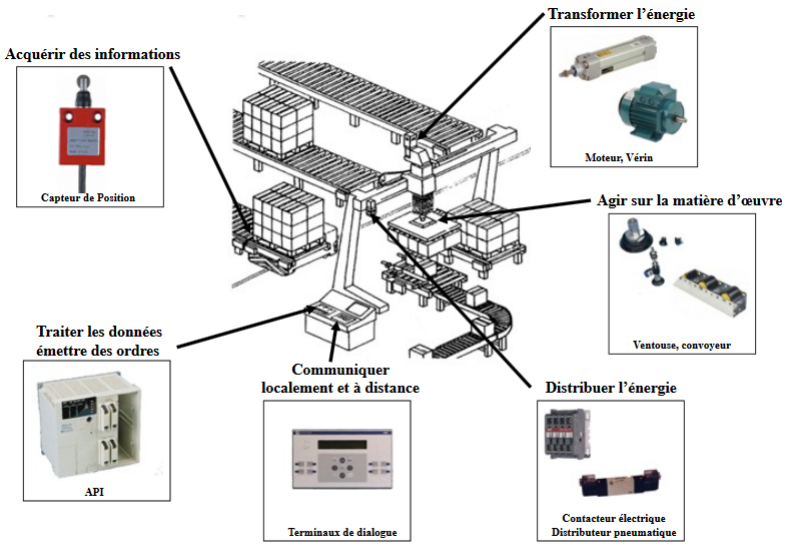


FIGURE 1.1 – Exemple d'une Chaîne de palettisation.

1.2 Objectif de l'automatisation

Le but principale est de remplacer l'intervention humaine dans la plus part des taches qui peut êtres rapides, très précises, dangereuses, répétitives et pénibles.

- L'automatisation aide à la compétitivité du produit.
- Améliorer la productivité en réduisant les coûts de production.
- Améliorer la qualité de production.
- Augmenter la production.
- Améliorer la flexibilité du système de production.
- Améliorer la qualité et la disponibilité des produits.

1.3 Structure des systèmes automatisés

Les systèmes automatisés, utilisés dans le secteur industriel, possèdent une structure de base identique. Ils sont constitués de plusieurs parties plus ou moins complexes reliées entre elles : la partie opérative (PO) ; la partie commande (PC) ou système de contrôle/commande.

1.3.1 Partie de commande (PC)

La PC gère, selon une suite logique, le déroulement ordonné des opérations à réaliser. Il reçoit des informations en provenance des capteurs de la Partie Opérative, et les restitue vers cette même Partie Opérative en direction des pré-actionneurs et actionneurs.

1.3.1.1 Réalisation matérielle de la PC

La réalisation matérielle de la partie commande peut être effectuée en **logique câblée (séquenceurs)**, en utilisant :

- technologie électronique ;
- technologie électrique ;
- technologie pneumatique ;

Basiquement, pour effectuer de petites et moyennes opérations de contrôle-commande,les relais et plus globalement la logique câblée peuvent suffir. Cependant lorsque les systèmes à commander deviennent de plus en plus complexes,on aura besoin d'autres organes de commande flexible comme la logique programmée utilisant :

- Automates Programmables Industriels (API) ;
- μ-ordinateurs industriels ;
- cartes électroniques à μp où à μ-contrôleurs ;

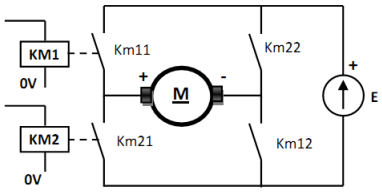


FIGURE 1.2 – Exemples de technologie de réalisation matérielle de la PC.

1.3.1.2 Types de commande

Il existe deux types de lois de commande :

- Commande séquentielle : variables booléennes

- Commande continue : asservissement et régulation de processus continu

1.3.1.3 Organisation de PC

On distingue deux types d'organisation de PC :

- Commande centralisée : Elle gère seule la totalité du SA à partir d'un unique système de traitement.

- Commande répartie : association de commandes autonomes.

1.3.2 Partie opérative (PO)

La PO est l'ensemble des moyens techniques qui effectuent directement le processus de traitement de la matière d'œuvre, à partir des ordres fournis par la PC. Elle comporte les éléments du procédé, c'est à dire : des pré-actionneurs des actionneurs, des capteurs. La figure suivante montre le principe et la structure technique de fonctionnement du système automatisé.

1.3.3 Principe de fonctionnement du système d'automatisation industriel

Comme montre la figure 1.3, l'opérateur va donner des consignes à la partie commande. Celle-ci va traduire ces consignés en Ordres qui vont être exécutes par la partie Opérative. Une fois les Ordres accomplis, la partie opérative va le signaler à la partie commande qui va à son our le signaler à l'opérateur. Ce dernier pourra donc dire que le travail à bien été réalisé.

1.4 Éléments d'un système automatisé

1.4.1 Pré-actionneur

Le Pré-actionneur est le constituant qui autorise le passage de l'énergie du milieu extérieur vers l'actionneur. Donc, Le Pré-actionneur distribue l'énergie nécessaire à l'actionneur en fonction des ordres reçus. Le pré-actionneur peut être :

- Tout ou Rien, il laisse passer ou non.

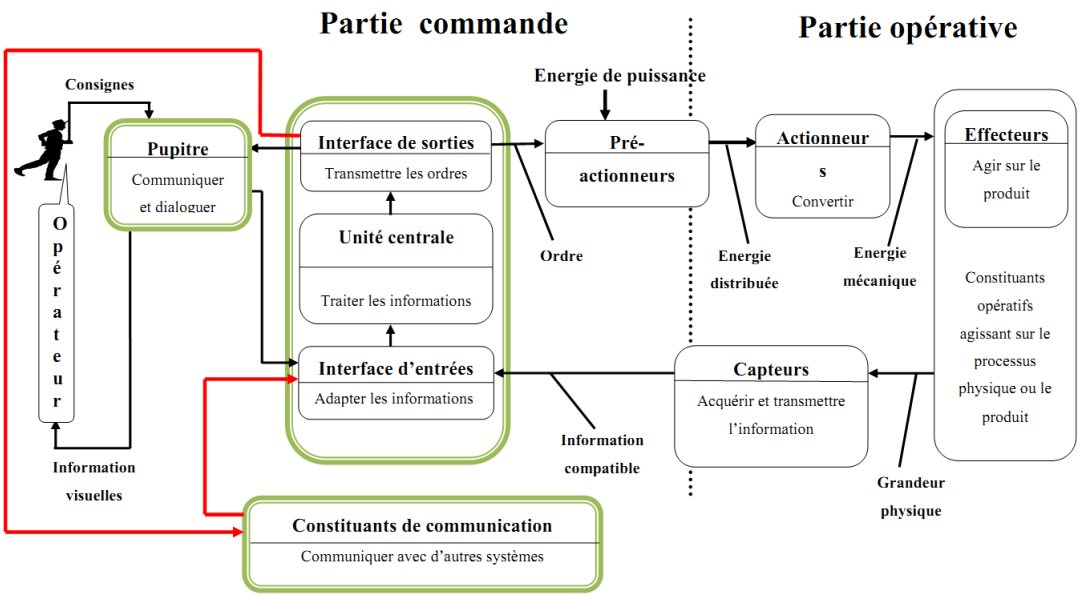


FIGURE 1.3 – Schéma de principe de fonctionnement des systèmes automatisés.

- Progressif, il ne laisse passer qu'une quantité d'énergie proportionnelle à la commande.

On distingue deux types de pré-actionneurs : pneumatiques et électriques. Ci-dessous quelques exemples des pré-actionneurs utilisés dans l'industrie.

FIGURE 1.4 – Contacteur de moteur électrique.

FIGURE 1.5 – Variateur de vitesse.

FIGURE 1.6 – Distributeur pneumatique.

FIGURE 1.7 – Distributeur électro-pneumatique.

1.4.2 Actionneurs

les Actionneurs permettent de transformer l'énergie reçue en un phénomène physique (déplacement, dégagement de chaleur, émission de lumière ...etc). Il existe différents types des actionneurs utilisé dans un système automatisés, la majorité des actionneurs sont classés selon la nature de leur énergie d'alimentation.

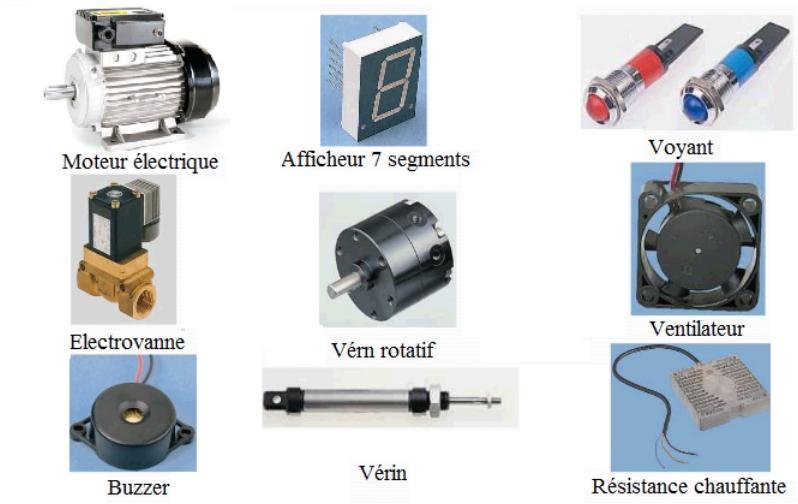


FIGURE 1.8 – Exemples d'actionneurs.

1.4.3 Capteurs

Les Capteurs permettent de prélever sur la partie opérative, l'état de la matière d'œuvre et son évolution ; il est capable de détecter un phénomène physique dans son environnement (déplacement, présence, chaleur, lumière, pression...) puis transforme l'information physique en une information codée compréhensible par la partie commande. Peuvent être électriques ou pneumatiques. Signaux du type TOR, Analogique ou Numérique.

FIGURE 1.9 – Différents types de capteurs.

1.4.4 Effecteur

Il est un organe utilise l'énergie convertie par les actionneurs pour produire un effet utile. Dans une chaine d'action, l'effecteur est l'organe terminal qui agit directement sur le produit traité par le système. Ci-dessous un exemple d'un effecteur.

1.4.5 Unité de dialogue

Elle permet à l'opérateur d'envoyer des consignes à l'unité de traitement et de recevoir de celle-ci des informations sur le déroulement du processus. En utilisant les pupitres de commande industriels qui offrent des services performants d'affichage, de saisie de paramétrage, de

FIGURE 1.10 – Organe effecteur d'un robot industriel.

commande pour la conduite de la machine, mais aussi de gestion de défauts, d'historiques sur les pièces et les défauts.

FIGURE 1.11 – Pupitre de commande industrielle.

Chapitre 2

Le Grafcet

Sommaire

2.1 Définition et notions de bases

Parmi des moyens ou des techniques de description (modélisation) du cahier des charges d'une chaine opérationnelle ou d'automatisme, le GRAFCET est le plus utilisé. **GraFCET** est l'abréviation **GRAphe Fonctionnel de Commande Etape/Transition**, il a été proposé par l'ADEPA (en 1977 et normalisé en 1982 par la NF C03-190).

C'est un outil graphique de description, permettant de représenter le cahier des charges d'un automatisme. Il est très utilisé pour la programmation des automates programmables industriels (API). Le GRAFCET est composé d'étapes, de transitions et de liaisons.

Exemple introductif On veux de contrôler une chaine de fonctionnement contient deux vérins pneumatiques (V1 et V2), dans l'étapes 1 si les vérins reculés avancer V1 a la position a1, dans l'étape 2 reculer V1 et avancer V2 jusqu'à la position b1.

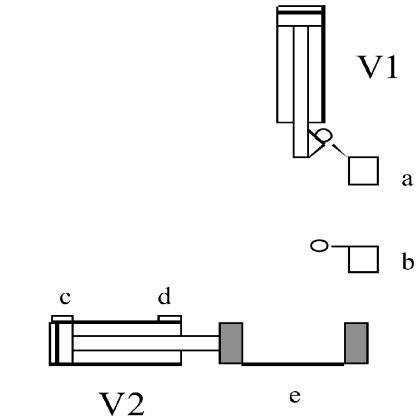
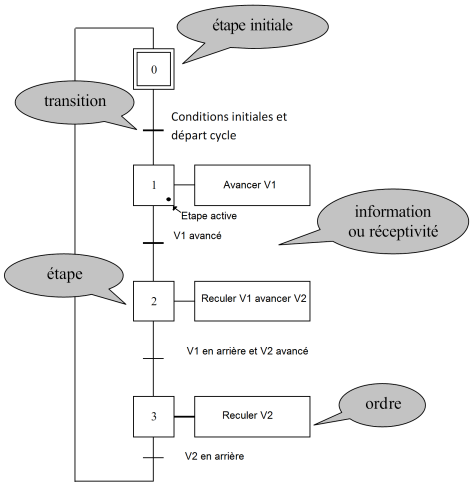


Illustration graphique de cycle de fonctionnement



Le GRAFCET de cycle de fonctionnemnt

FIGURE 2.1 – Exemple de GRAFCET simple.

2.1.1 Structure du GRAFCET

Liaison : Une liaison est un arc orienté (ne peut être parcouru que dans un sens). A une extrémité d'une liaison il y a une (et une seule) étape, à l'autre une transition

Étape : Une étape correspond à une phase durant laquelle on effectue une ou plusieurs actions pendant une certaine durée (même faible mais jamais nulle). L'action doit être stable, c'est à dire que l'on fait la même chose pendant toute la durée de l'étape, mais la notion d'action est assez large, en particulier composition de plusieurs actions, ou à l'opposé l'inaction (étape dite d'attente).

Transition : Elle indique la possibilité d'évolution entre deux étapes successives, elle n'est que logique (dans son sens Vrai ou Faux), sans notion de durée. A chaque transition est associée une condition booléenne (c.à.d avec des ET et des OU de l'état des capteurs) appelée réceptivité.

- **Réceptivité :** est une fonction combinatoire d'informations telles que :
  - États de capteurs ;
  - Action de boutons-poussoirs par l'opérateur ;
  - Action d'un temporisateur, d'un compteur ;
  - État actif ou inactif d'autres étapes ;
  - Comparaison d'une valeur analogique.

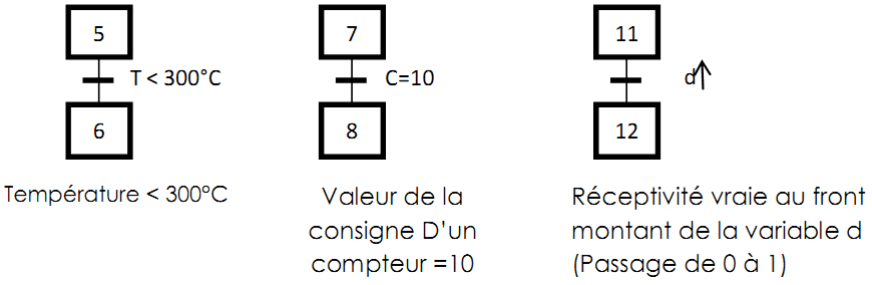


FIGURE 2.2 – Exemple de réceptivités.

2.1.2 Règles d'établissement du GRAFCET

- Sur le GRAFCET la liaison doit être représentée par un trait plein rectiligne, vertical ou horizontal. Une verticale est parcourue de haut en bas, sinon il faut le préciser par une flèche. Une horizontale est parcourue de gauche à droite, sinon le préciser par une flèche. Si plusieurs

liaisons arrivent sur une étape, pour plus de clarté on les fait arriver sur une barre horizontale, de même pour plusieurs liaisons partant de l'étape.

- Chaque étape doit être présentée par un carré, l'action est représentée dans un rectangle à gauche, l'entrée se fait par le haut et la sortie par le bas Chaque étape doit être numérotée par un entier positif, mais pas nécessairement croissant par pas de 1, il faut simplement que jamais deux étapes différentes n'aient le même numéro.

- La Transition est présenté par un petit trait horizontal sur une liaison verticale. On note à droite la réceptivité, on peut noter à gauche un numéro de transition (entier positif, indépendant des numéros d'étapes).

2.2 Règles d'évolution du GRAFCET

Selon la normalisation France (NF C03-190 juin 82) et international, On distingue cinq règles de base pour l'évolution du GRAFCET .

2.2.1 Règle 1 "Situation initiale"

L'initialisation caractérise le comportement initial de la partie commande vis-à-vis de la partie opérative et correspond aux étapes actives au début du fonctionnement. Elles sont activées inconditionnellement. A toute étape i, on peut associe une variable logique "Xi" telle que Xi=1 si l'étape est active et Xi=0 si l'étape est inactive comme il est montré sur la figure ci-dessous.

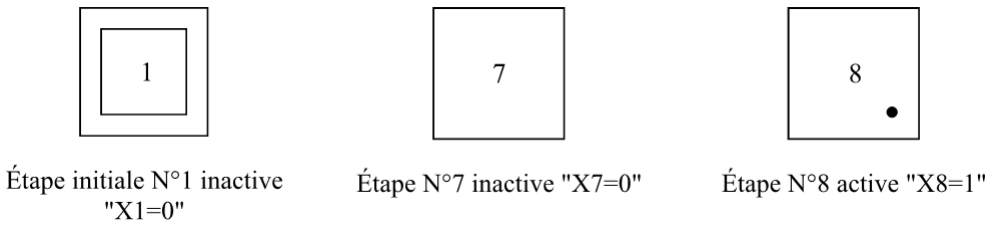


FIGURE 2.3 – Exemple des étapes actives et des étapes inactives.

2.2.2 Règle 2 "Franchissement d'une transition"

Une transition est soit validée soit non validée. Elle est validée lorsque toutes les étapes immédiatement précédentes sont activées et la réceptivité associée à la transition est vraie.

Lorsque ces deux conditions d'évolution sont réunies, la transition devient franchissable et alors obligatoirement franchie.

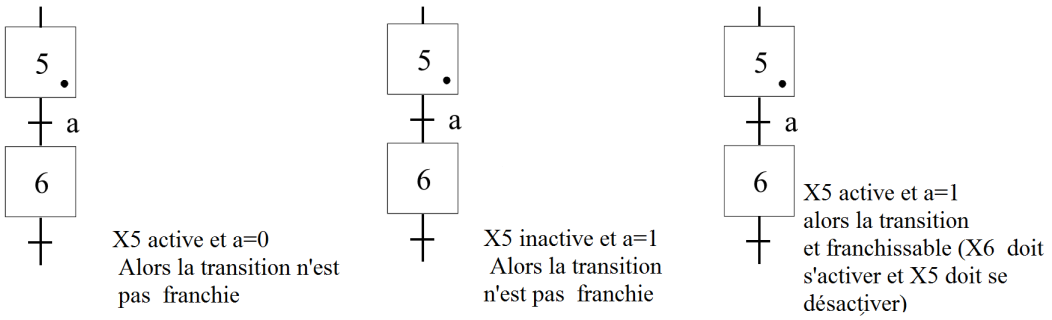


FIGURE 2.4 – Mécanisme de franchissement d'une transition.

2.2.3 Règle 3 "Évolution des étapes actives"

Le franchissement d'une transition entraîne l'activation de toutes les étapes qui suivent immédiatement cette transition et la désactivation de toutes les étapes qui précèdent immédiatement cette transition . Cette évolution du GRAFCET est donc synchrone.

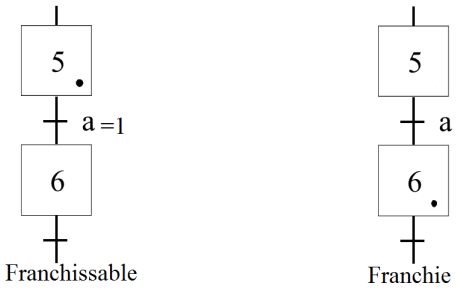


FIGURE 2.5 – Exemple de transitions simultanément franchissables.

2.2.4 Règle 4 "Évolution simultanée"

Plusieurs transitions simultanément franchissables sont simultanément franchies.

2.2.5 Activation et désactivation simultanée

Si au cours du fonctionnement, une même étape doit être désactivée et activée simultanément, elle reste activée. L'activation doit être prioritaire sur la désactivation au niveau d'une même étape.

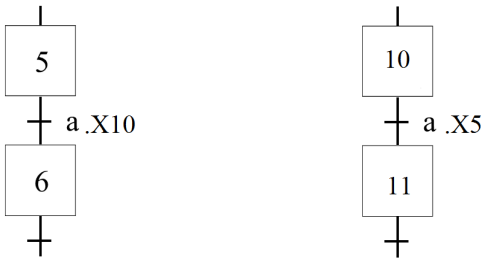


FIGURE 2.6 – Évolution des étapes durant le franchissement d'une transition.

2.3 Point de vue d'un grafcet

On distingue trois phases dans l'étude d'un système automatisée : le point de vue système, le point de vue partie opérative, le point de vue partie commande.

-**point de vue système**C'est un graphe qui décrit le fonctionnement global du système. Il traduit le cahier des charges sans préjuger de la technologie adoptée. Il permet de dialoguer avec des personnes non spécialistes (fournisseurs, décideurs ...) Son écriture, en langage clair, permet donc sa compréhension par tout le monde.

-**Le point de vue partie opérative :** Dans ce type de grafcet on spécifie la technologie de la partie opérative ainsi que le type de ses informations reçues (ordres) et envoyées (comptes-rendus). L'observateur de ce point de vue étant un spécialiste de la partie opérative, la partie commande ne l'intéresse que par ses effets.

-**le point de vue partie commande :** décrit le comportement de la partie commande par rapport à la partie opérative en tenant compte du choix de la technologie employée. Un schéma de câblage (électrique et pneumatique) décrit le raccordement des transmetteurs et des préactionneurs à la partie commande.

La figure 2.9 présente un exemple sur les différents points de vue de grafcet.

2.4 Classification des actions

2.4.1 Action continue

La durée de ce type d'action correspond à la durée d'activation de l'étape. Plusieurs actions continues peuvent être associées à une même étape. Une action maintenue dans plusieurs étapes successives s'appelle "action maintenue". Comme un exemple l'action A dans l'exemple ci-

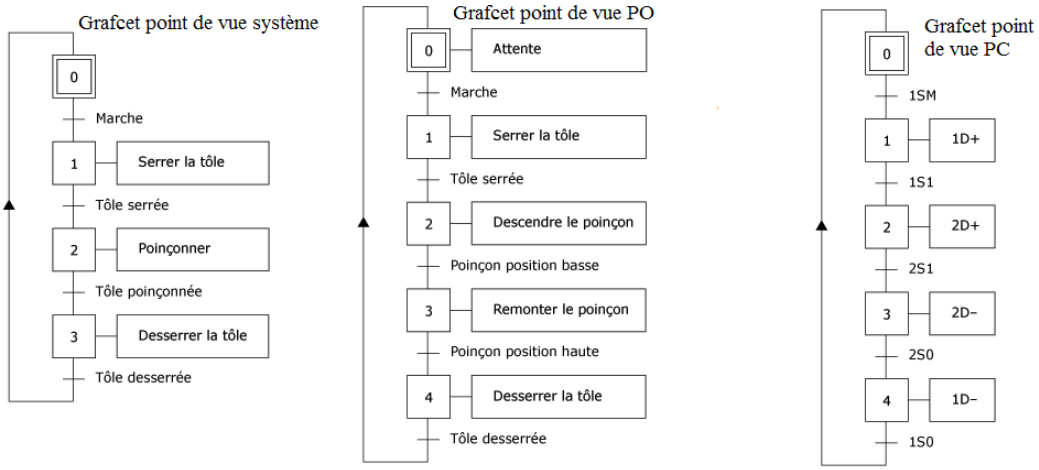


FIGURE 2.7 – Exemple sur les différents points de vue de grafcet.

dessous (2.8 est une action maintenue, l'action B est une action continue.

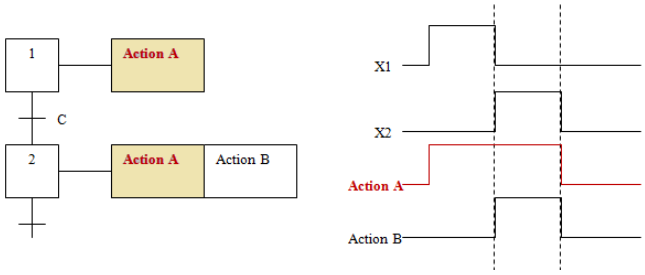


FIGURE 2.8 – Exemple d'actions continues.

2.4.2 Action conditionnelle

Une action Conditionnelle n'est vraie que si l'étape est active ET la condition est vraie.

2.4.3 Action temporisée

C'est une action conditionnelle dans laquelle le temps intervient comme condition logique. comme il est montré dans 2.10. Dans cet exemple l'action A est limitée 5 secondes et l'action B est retardée 10 de secondes à partir de l'activation de l'étape 1.

2.4.4 Action mémorisée

Une action mémorisée A comportant soit un S (pour set), soit un R pour reset. Le seul moyen d'arrêter une action positionnée par un Set est de faire un Reset.

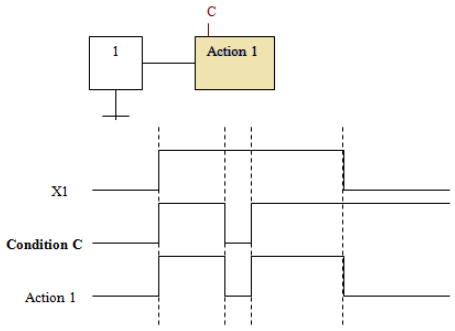


FIGURE 2.9 – Exemple d'action conditionnelle.

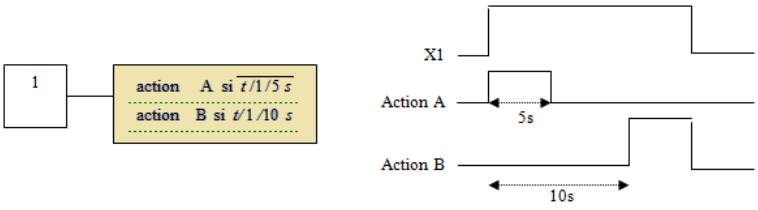


FIGURE 2.10 – Exemple des actions temporisées.

2.5 Structure de base de grafcet

2.5.1 Séquence linéaire

Une séquence linéaire est composée d'une suite d'étapes qui peuvent être activées les unes après les autres.

2.5.2 Sélection de séquence

Une sélection de séquence est un choix d'évolution entre plusieurs séquences à partir d'une ou plusieurs étapes. On dit qu'il y a Aiguillage ou divergence en OU lorsque le grafcet se décompose en deux ou plusieurs séquences selon un choix conditionnel. Comme la divergence en OU on rencontre aussi la convergence en OU. On dit qu'il y a convergence en OU, lorsque deux ou plusieurs séquences du grafcet converge vers une seule séquence. 2.14

L'exclusion entre les séquences n'est pas structurelle. Pour l'obtenir, il faut s'assurer soit de l'incompatibilité mécanique ou temporelle des réceptivités, soit de leur exclusion logique.

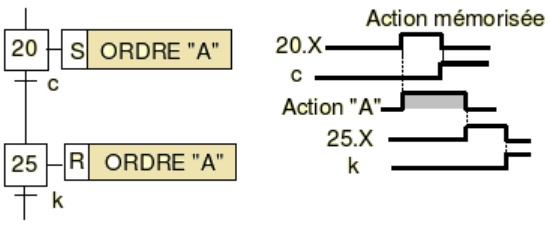


FIGURE 2.11 – Exemple d'une action mémorisée.

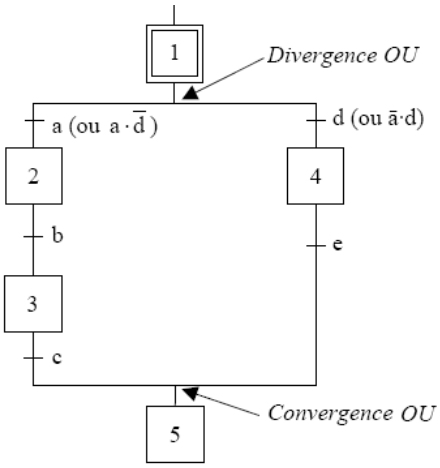


FIGURE 2.12 – Exemple de Aiguillage entre deux séquences.

2.5.3 Séquences simultanées

Si le franchissement d'une transition conduit à activer plusieurs étapes en même temps, ces étapes déclencheront des séquences dont les évolutions seront à la fois simultanées et indépendantes. La synchronisation permet d'attendre la fin de plusieurs activités se déroulant en parallèle, pour continuer par une seule

2.5.4 Saut d'étapes et reprise de séquence

Un saut d'étape permet de sauter une ou plusieurs étapes lorsque les actions associées à ces étapes deviennent inutiles. Un renvoi de séquence permet d'effectuer plusieurs fois une même séquence tant qu'une condition n'est pas réalisée (voir 2.15)

2.5.5 Exemple de perceuse

- Cahier des charges : Après l'ordre de départ cycle dcy, la perceuse effectue, selon l'épaisseur de la pièce un cycle avec ou sans débourrage.

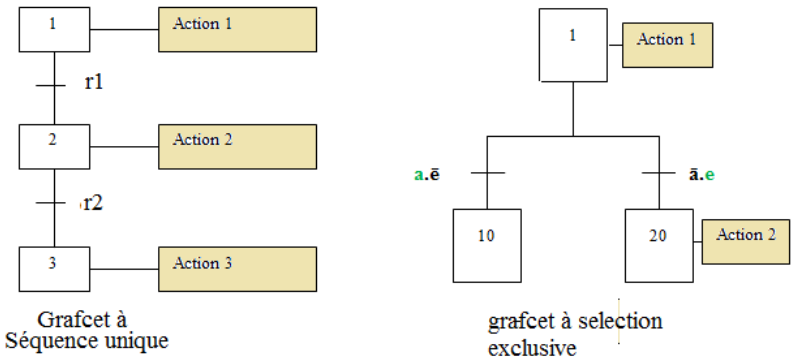


FIGURE 2.13 – Exemple de séquence unique et de sélection exclusive.

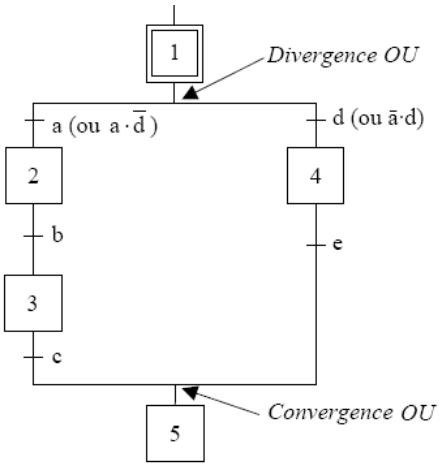


FIGURE 2.14 – Exemple de Aiguillage entre deux séquences.

- Capteurs :

- h, b1, b2, b3 : capteurs de position
- c : capteur de contact

- Actions :

- Descendre en grande vitesse
- Descendre en petite vitesse
- Remontée en grande vitesse

La 2.16 présente le grafcet de système de perçage

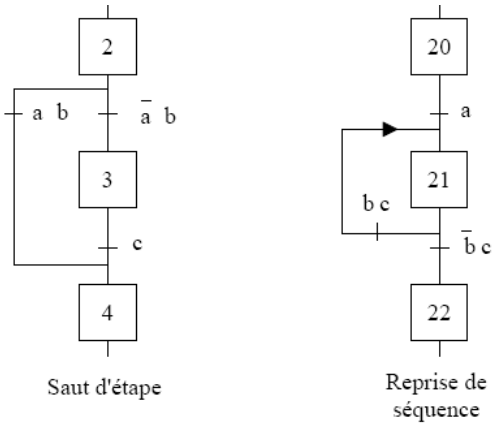


FIGURE 2.15 – Exemple de saut d'étapes et reprise de séquence.

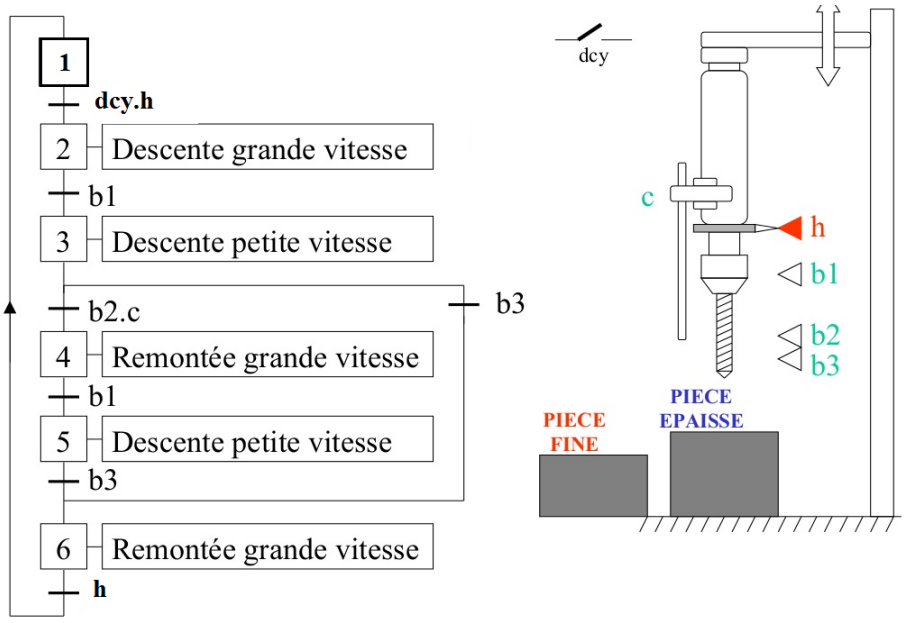


FIGURE 2.16 – Grafcet de perceuse.

2.6 Mise en equation du grafcet

L'objectif de la mise en equation du grafcet est décrire l'activité d'un étape en fonction de toutes les variables qui interviennent dans son activité et désactivité. Soit le grafcet simple en 2.17 :

A chaque étape i est associée une variable X_i :

– $X_i = 1$ si l'étape i est active

– $X_i = 0$ si l'étape i est inactive

Egalment la réceptivité :

– $R_i = 1$ si la réceptivité est fausse

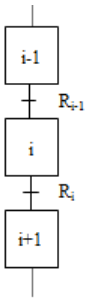


FIGURE 2.17 – exemple de grafcet simple.

– $R_i = 0$ si la réceptivité est vraie

D'après la règle 2 du grafcet, la Condition d'Activation de l'étape i donne :

"$CAX_i = X_i - 1R_i - 1$

D'après la règle 3 du grafcet, , la Condition de Désactivation de l'étape i donne :

$CDXi = X_iR_i = X_{i+1}$

Si la CA et la CD de l'étape i sont fausses, l'étape i reste dans son état (effet mémoire). L'état de X_i à l'instant $t + \delta t$ dépend de l'état précédent de X_i à l'instant t. on peut alors écrire :

$X_n = f(CAX_n, X_n, CDX_n)$

Donc c'est possible d'écrire le table de vérité de l'activité de l'étape Xn : Tableau de Karnaugh

Table 2.1 - Table de vérité de l'étape n.

X_n (t)	CAX_n	CDX_n	X_n (t +δt)
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

associé :

X_n \ CAX_n .CDX_n	00	01	11	10
0	0	0	1	1
1	1	0	1	1

Donc de l'état de l'tape Xn peut être par l'équation suivante : $X_n = CAX_n + CDX_n.X_n$

ou bien :

$X_n = X_{n-1}.t_{n-1} + X_{n+1}.X_n$

2.6.1 Gestion de mode Marche/Arrêt et Arrêt d'urgence

A l'état initiale du GRAFCET, les étapes initialles sont activées par contre les autres étapes sont désactivées.

On introduit une variable Init telle que :

- Init=0 : Mode MARCHE (déroulement du cycle)
- Init= 1 : Mode ARRÊT (initialisation du grafcet)

On introduit deux variables d'Arrêt d'urgence AUdur (Arrêt d'Urgence dur) et AUdoux (Arrêt d'Urgence doux) telles que :

- AUdur= 1 : Désactivation de toutes les étapes.
- Audoux =1 : Désactivation des actions, mais les étapes restent actives

2.6.1.1 Étape initiale

CAXi	CDXi	Equation de Xi
$X_{i-1}.t_{i-1} + Init$	$X_{i+1}.\overline{Init}$	$X_i = \left(CAX_i + \overline{CDX_i}.X_i + Init\right).\overline{AUdur}$

2.6.1.2 Étape non initiale

CAXi	CDXi	Equation de Xi
$X_{i-1}.t_{i-1}.\overline{Init}$	$X_{i+1}. + Init$	$X_i = \left(CAX_i + \overline{CDX_i}.X_i\right).\overline{Init}.\overline{AUdur}$

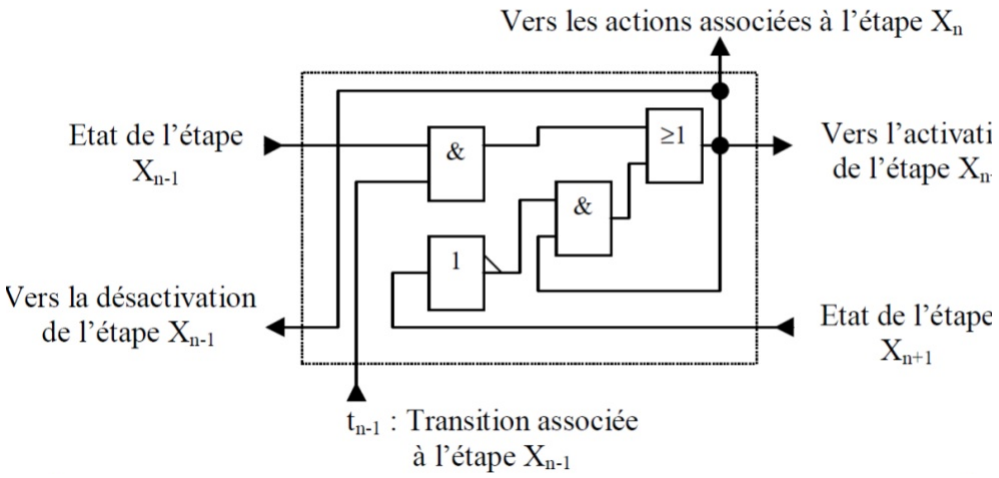
2.7 Matérialisation du grafcte

2.7.1 A l'aide des portes logiques

soit une étape décrit par :

$X_n = X_{n-1}.t_{n-1} + X_{n+1}.X_n$

Donc cette étape peut être réalisée à l'aide des portes logiques comme tout suite.



2.7.2 A l'aide des bascule RS

Rappel sur Bascule RS :

Un verrou RS possède deux entrées de contrôle : Set et Reset, et n'a pas d'entrée de donnée. Les deux signaux de sortie Q et est présent. Le fonctionnement de cette bascule est le suivant :

R	S	Q_{t+1}
0	0	Q_t
0	1	1
1	0	0
1	1	-

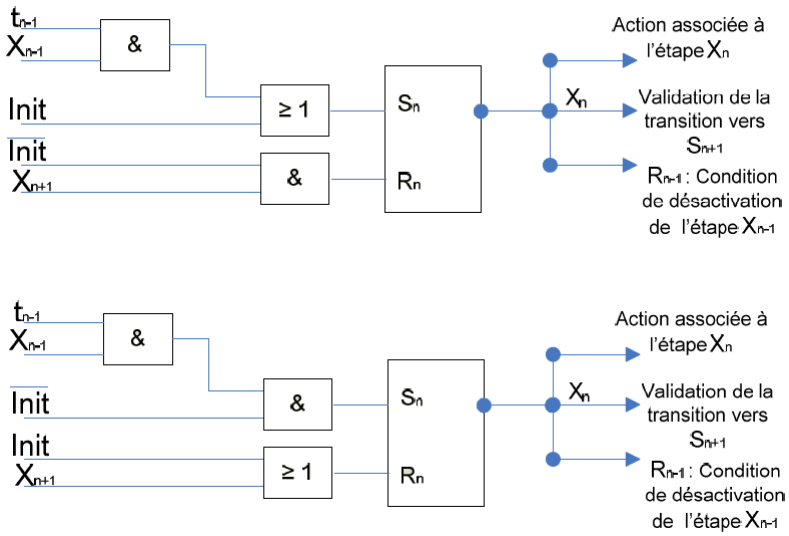
application au grafcet, en utilisant directement les conditions d'activation et conditions désactivation d'étape, où la condition d'activation de l'étape est câblée sur le SET de la bascule, et la condition de désactivation de l'étape est câblée sur le RESET de la bascule.

2.7.2.1 Câblage d'une étape initiale

$$CAXn = Xn-1.tn-1 + Init$$
$$CDXn = Xn+1.\bar{Init}$$

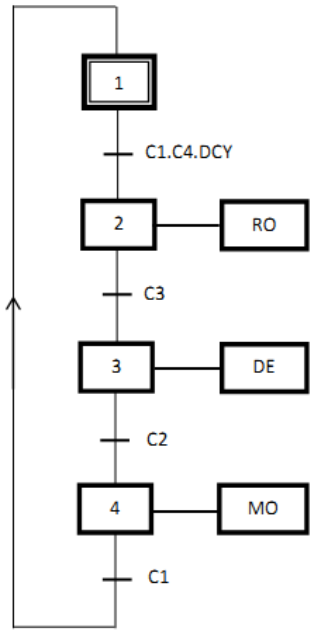
2.7.2.2 Câblage d'une étape non initiale

$$CAXn = Xn-1.tn-1.\bar{Init}$$
$$CDXn = Xn+1+Init$$



2.8 Exercice 1

soit le grafcet présenté en 2.8, - trouver les CA et CD de chaque étape puis décrire les equations des étapes. - Réaliser le câblage du grafcet à l'aide des porte logique puis à l'aide de bascule RS



Chapitre 3

Automates Programmables Industriels

Sommaire

3.1 Étude architecturale des microprocesseurs

3.1.1 Définition de microprocesseur

Un microprocesseur est un circuit intégré complexe caractérisé par une très grande intégration et doté des facultés d'interprétation et d'exécution des instructions d'un programme. Il est chargé d'organiser les tâches précisées par le programme et d'assurer leur exécution.il prend en compte les informations extérieures au système et assurer leur traitement. Il est considéré comme le cerveau du système.

3.1.2 Architecture de base de microprocesseur

Tous les micro-processeurs conçus jusqu'à aujourd'hui s'organisent globalement de même façon. Ils peuvent être décomposés en quatre composants décrits par le modèle de **von Neumann**. Ce dernier donne les quatre composants essentiels qui constituent un micro-processeur. Il décrit également les interactions entre ces différents composants.

Les quatre composants du modèle de von Neumann sont les suivants.

1. unité de contrôle
2. unité de traitement
3. mémoire
4. unité d'entrées/sorties

- Unité Arithmétique et Logique (UAL) : Elle dispose de circuits réalisant des opérations des fonctions logiques (ET, OU, comparaison , décalage,...) ou arithmétique (addition, soustraction,...). En entrée de l'UAL, on a des commandes permettant d'activer les opérations, venant de l ?unité de contrôle En sortie, on a les résultats des opérations et les conditions qui sont en fait les entrées de l'unité de contrôle.

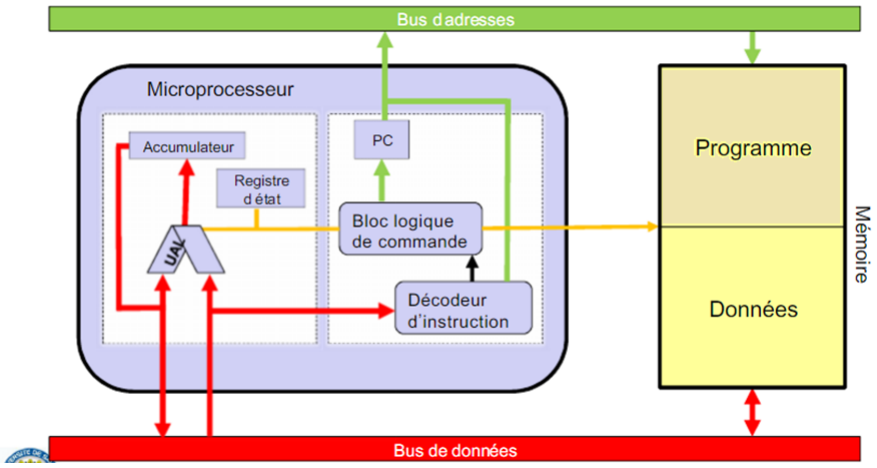


FIGURE 3.1 – Architecture de base de microprocesseur.

- Unité de contrôle ou séquenceur : L ?unité de contrôle est une circuit logique séquentiel chargée de séquencer l'algorithme et de générer les signaux de contrôle pour piloter les éléments du chemin de données. Elle envoie des commandes à l'unité de traitement qui va exécuter les traitements .

- Les bus : Un bus est un ensemble de lignes de communications groupés par fonction. Il permet de faire transiter (liaison série/parallèle) des informations codées en binaire entre deux points.

- Les registres : C'est un espace mémoire interne au processeur.

3.2 Étude architecturale de micro-contrôleurs

Le microcontrôleur est un dérivé du microprocesseur. Sa structure est celle des systèmes à base de microprocesseurs. Il est donc composé en plus de l'unité centrale de traitement, d'une mémoire (mémoire vive RAM et mémoire morte ROM), une (ou plusieurs) interface de communication avec l'extérieur matérialisé par les ports d'entrée/sortie.

Tous les micro-contrôleurs utilisent l'une des 2 architectures nommées Harvard et Von Neumann. Elles représentent les différentes manières d'échange de données entre le CPU (microprocesseur interne) et la mémoire.

Architecture Von Neumann : Toutes les données sont échangées sur ce bus qui, surchargé , rend la communication très lente. Communication En Série

Architecture Harvard : Le CPU peut lire une instruction (en ROM) et accéder à la mémoire de données (en RAM) en même temps. Communication En Parallèle

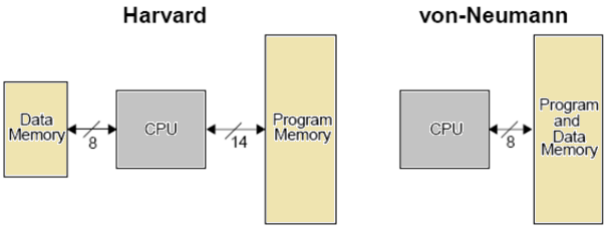


FIGURE 3.2 – Schéma de principe des architectures Harvard et Von Neumann.

3.3 Structure de micro-contrôleur

Les micro-contrôleurs améliorent l'intégration et le coût d'un système à base de microprocesseur en rassemblant ces éléments essentiels dans un seul circuit intégré avec des périphériques . Les périphériques sont des circuits électroniques intégrés au micro-contrôleur capables d'effectuer des tâches spécifiques. On peut mentionner entre autres :

- les convertisseurs Analogique/Numérique
- les convertisseurs Numérique/Analogique
- les générateurs de signaux à modulation par largeur d'impulsion (PWM, Pulse Width Modulation) ;
- les timers (compteurs de temps ou d'événements) ;
- les comparateurs (comparent deux tensions électriques) ;
- les contrôleurs de bus
- les oscillateurs (servent de base de temps aux timers)

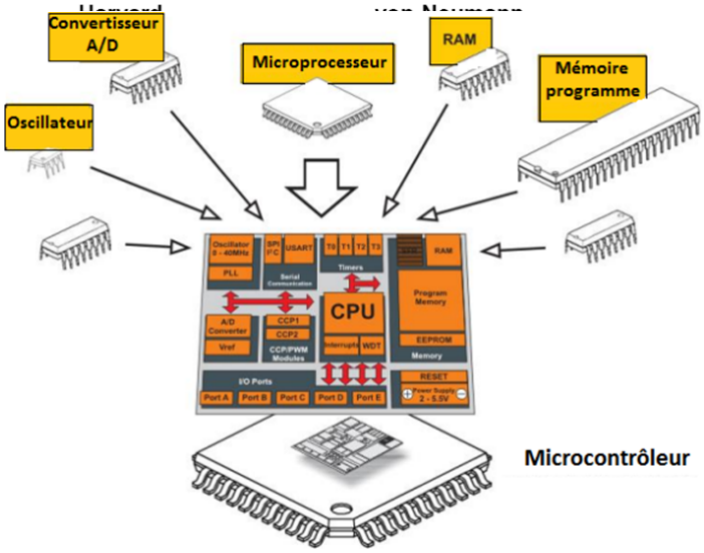


FIGURE 3.3 – Composants de la structure d'un micro-contrôleur.

3.4 Définition d'un automate

Un automate programmable est un appareil électronique qui fait un traitement des signaux a l'entrée et envois d'autre a la sortie, destiner au contrôle d'une machine ou d'un processus industriel, il comporte une mémoire programmable par un utilisateur non informaticien, à l'aide d'un langage adapté. En d'autres termes, un automate programmable est un calculateur logique, ou ordinateur, destiné à la conduite et la surveillance en temps réel de processus industriels. L'API est distinguer par Trois caractérises fondamentales :

- Il peut être directement connecté aux capteurs et pré-actionneurs grâce à ses entrées/sorties industrielles.
- Il est conçu pour fonctionner dans des ambiances industrielles sévères (température, vibrations, micro-coupures de la tension d'alimentation, parasites, etc.).
- Et enfin, sa programmation à partir de langages spécialement développés pour le traitement de fonctions d'automatisme fait en sorte que sa mise en ?uvre et son

exploitation ne nécessitent aucune connaissance en informatique

3.4.1 Structure générale des API

Les automates programmables industrielle sont construis avec des différentes structure, il existe : Les compacts, les racks tables et les modulaires Les caractéristiques principales d'un automate programmable industriel (API) sont : coffret, rack, baie ou cartes

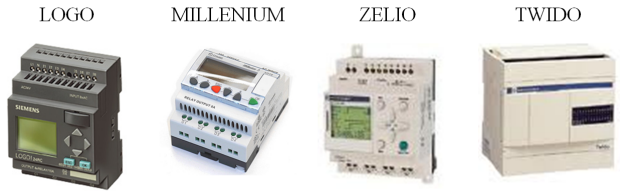


FIGURE 3.4 – Exemples des APIs compacts.

FIGURE 3.5 – Exemples des APIs modulaires.

3.5 Principe de fonctionnement

Le traitement à lieu en trois phases :

Phase 1 :Acquisition des entrées

Prise en compte des informations du module d'entrées et écriture de leur valeur dans RAM (zone DONNEE).

Phase 2 : Traitement des données

le processeur exécute les instructions de la mémoire programme en fonction des informations de la mémoire des données. Cette exécution se traduit par la modification de certaines variables et leur mise à jour dans la zone correspondante. .

Phase 3 : Emissions des ordres

les images des sorties dans la mémoire des données sont transférées dans le module de sortie pour être converti en signaux électriques pour la commande des préactionneurs et des dispositifs de visualisation. Ces valeurs sont verrouillées jusqu'au cycle prochain.

3.6 Critères de choix d'un automate

A part le critère de prix qui fait partie des critères les plus considérés lors du choix d'un automate,le choix définitif dépendra du cahier des charges et des spécificités techniques du projet. Mais généralement, On doit tenir compte les critères techniques suivants :

- Compact ou modulaire
- Tension d'alimentation
- Taille mémoire
- Temps de réponse (performance de processeur)
- Sauvegarde (EPROM, EEPROM, pile..)
- Nombre d'entrées / sorties
- Modules complémentaires (analogique, communication,..)
- Langage de programmation

3.7 Architecture interne d'un API

En général un automate programmable se constitue essentiellement d'une unité centrale, un module d'entrées/sorties, un module d'alimentation, un module de stockage et de liaisons et des

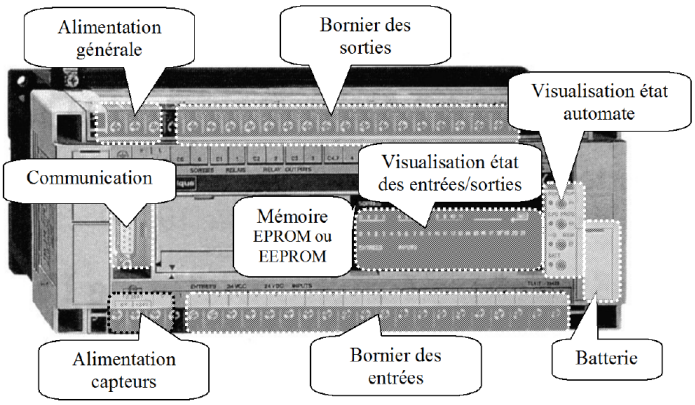


FIGURE 3.6 – Photo de l'API S7-200.

auxiliaires. Cette architecture est représentée par la figure suivante :

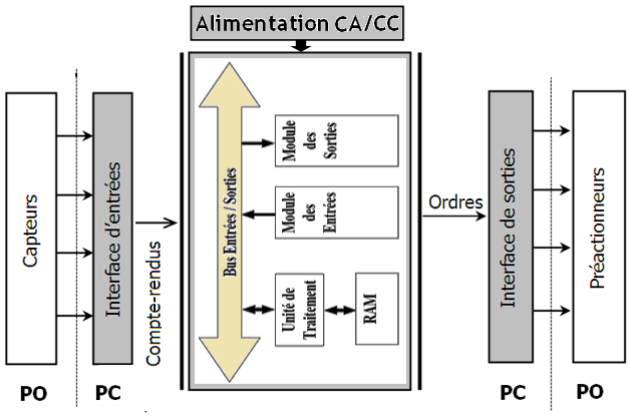


FIGURE 3.7 – Schéma d'architecture interne d'un API.

3.7.1 Module d'entrées d'un API

il transforme les signaux provenant des capteurs et des ordres de l'opérateur en signaux compréhensibles par l'automate. Le processeur stocke ensuite ces informations dans la mémoire de données image des entrées afin de les mémoriser. :

- Numérique : comptage rapide sur un codeur incrémental.
- Analogique : génératrice tachymétrique en entrée et variateur de vitesse en sortie
- Logique : entrées et sorties tout ou rien

3.7.1.1 Module d'entrée TOR

le mot TOR vient de "Tout Ou Rien" qui représente les signaux logiques. Les dispositifs d'entrée binaire réalisent, outre l'acquisition de l'information, les opérations suivantes :

– Mise en forme du signal (calibrage)
– Filtrage (élimination des parasites)
– Isolation (galvanique ou par lumière)

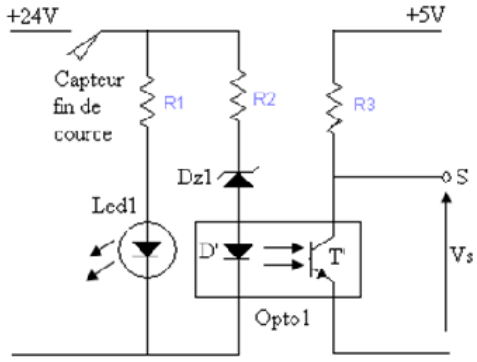


FIGURE 3.8 – Schéma du module d'entrée TOR d'un API.

3.7.1.2 Module d'entrée analogique

Les entrées analogiques transforment une grandeur analogique variant d'une façon continue en un code numérique, généralement sur 11 bits plus un bit de signe. Ces entrées disposent d'un seul convertisseur A/N (CAN), elles sont scrutées les unes à la suite des autres par un multiplexeur (MUX). Par contre, les sorties analogiques disposent d'un seul convertisseur par voie.

Les E/S analogiques sont caractérisées par l'amplitude du signal (VH et VL), par la vitesse de conversion et par la grandeur électrique (courant ou tension). Il existe, au niveau des entrées, trois types d'entrées analogiques :

– Haut niveau : 0-10V, 0-20mA, 4-20mA
– Pour thermocouple : 0-20mV, 0-50mV, 0-100mV
– Pour sondes Pt100 : 0-100mV, 0-250mV, 0-400mV

3.7.2 Module de sorties d'un API

il transmet aux préactionneurs et aux dispositifs de dialogue les ordres de commande et de signalisation résultants de l'exécution du programme. Le processeur vient chercher ses ordres dans la mémoire de données image des sorties, et les transfèrent en module de sortie qui seront transformés en signaux électriques par la suite. Cette architecture est représentée par la figure suivante :

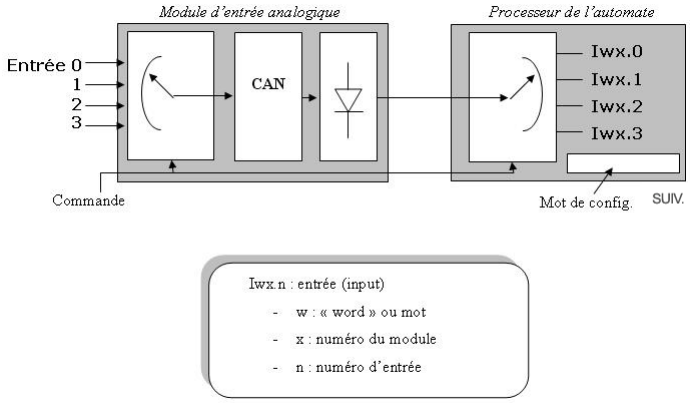


FIGURE 3.9 – Schéma du module d'entrée analogique d'un API.

3.7.2.1 Module de sortie TOR

Dès le forçage d'un bit de sortie, la carte fournit une tension, à 1 ou à 0, sur la sortie. Des composants comme des contacteurs, des électrovannes, des voyants, peuvent être commandés par ces sorties. Le temps de réponse d'une sortie (Ts), est le temps entre l'apparition d'un bit image de la sortie et la présence de tension sur cette sortie API.

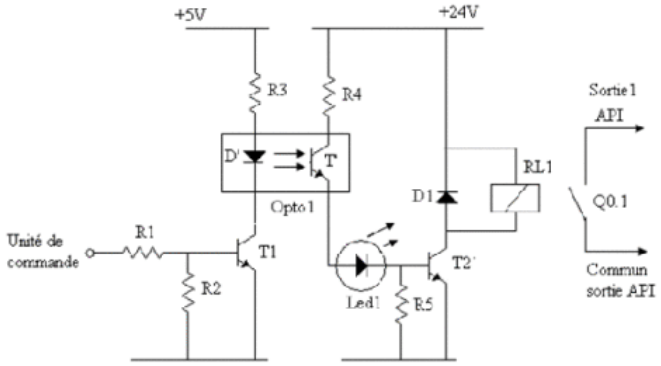


FIGURE 3.10 – Schéma d'architecture interne d'un API.

3.7.3 Module de communication

sert pour le dialogue entre l'automate et un autre équipement (automate, PC, ?etc.). L'exemple de : ModBus, ModBus Plus, Profibus, InterBus, DeviceNet, LonWorks, Ethernet, FIPIO, FIP-WAY, RS232, RS-485, AS-i, CANopen.

3.8 Langage de programmation des APIs

Chaque automate possède son propre langage. Mais par contre, les constructeurs proposent tous une interface logicielle répondant à la norme CEI1 1131-3. Cette norme définit cinq langages de programmation utilisables, qui sont :

langages graphiques :

- **1- Schéma a relais (LD) :** ce langage graphique est essentiellement dédie à la programmation d'équations booléennes (true/false).
- **3- Schéma par blocs (FBD) :** ce langage permet de programmer graphiquement à l'aide de blocs, représentant des variables, des opérateurs ou des fonctions. Il permet de manipuler tous les types de variables.

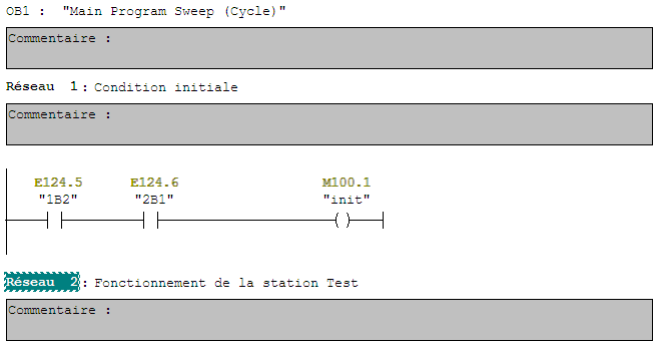


FIGURE 3.11 – Exemple d'un programme en langage LD.

Langages textuel :

- **2- Texte structure (ST) :** ce langage est un langage textuel de haut niveau. Il permet la programmation de tout type d'algorithme plus ou moins complexe.

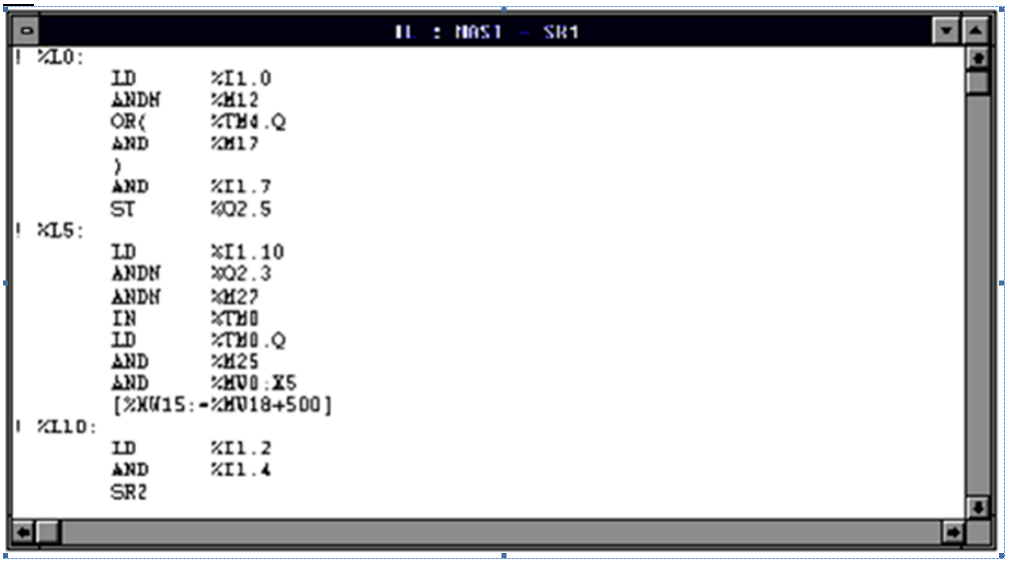


FIGURE 3.12 – Exemple d'un programme en langage ST.

– **4- Liste d'instructions (IL) :** ce langage textuel de bas niveau est un langage à une instruction par ligne. Il peut être compare au langage assembleur.

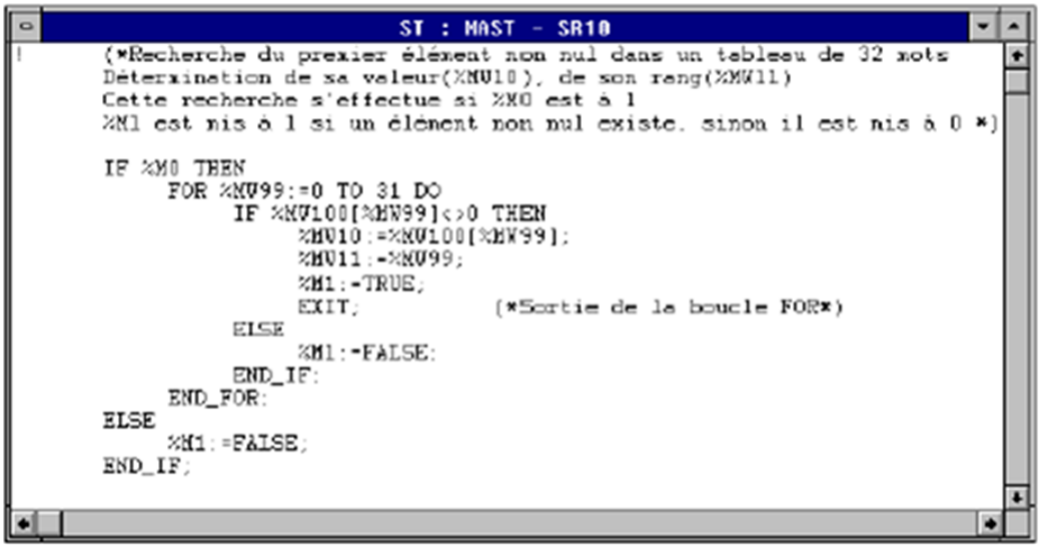


FIGURE 3.13 – Exemple d'un programme en langage IL.

5- Diagramme de fonctions séquentielles (Sequential Chart function, SFC) : Ce langage de programmation de haut niveau permet la programmation aisée de tous les procédés séquentiels.

Le tableau suivant présente l'avantage et l'inconvénient de chaque langage.

3.9 Programmation d'un API

Dans cette section on présente un exemple simple qui permet de suivre et comprendre les étapes principales de la programmation de l'API ; on adopte le langage de programmation Ladder comme une langage de programmation.

3.9.1 Programmation en langage Ladder

Le langage Ladder est composé d'une séquence de contacts (interrupteurs qui sont soit fermés,soit ouverts) et de bobines qui permettent de traduire les états logiques d'un système. Un réseau LD consiste en les éléments suivants :

– Connections
– Contacts et bobines.
– Éléments graphiques pour contrôler l'exécution de la séquence (sauts)
– Éléments graphiques pour appelés des fonctions blocs (FB)

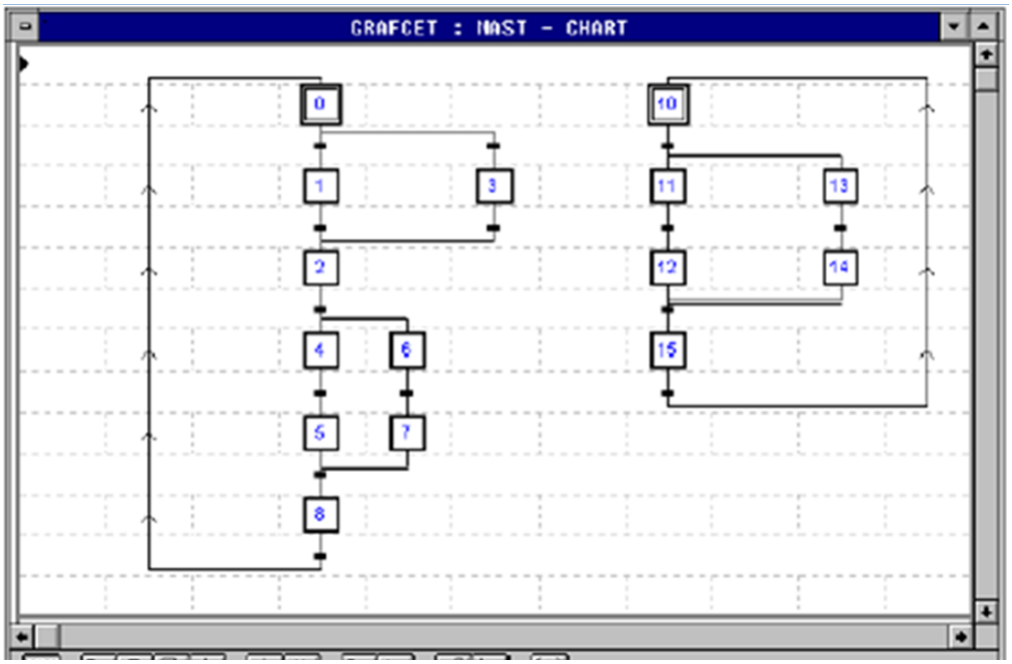


FIGURE 3.14 – Exemple d'un programme en langage SFC.

Langage	Avantage	Inconvénient
LD	facile à lire et à comprendre par la majorité des électriciens langage de base de tout API	suppose une programmation bien structurée
FBS	Très visuel et facile à lire	peut devenir très lourd lorsque les équations se compliquent
ST	Langage de haut niveau (langage pascal) Pour faire de l'algorithmique	pas toujours disponible dans les ateliers logiciels
IL	langage de base de tout PLC type assembleur	très lourd et difficile à suivre si le programme est complexe Pas visuel.
SFC	Description du fonctionnement Peu flexible (séquentiel) de l'automatisme. Gestion des modes de marches pas toujours accepté dans l'industriel	Peu flexible

– Connecteurs.

On présente dans le tableau suivant donne les principaux éléments utilisés en Langage LD.

3.10 Mise en œuvre d'API

Lors de sa première mise en œuvre il faut réaliser la mise au point du système.

Prendre connaissance du système (dossier technique, des GRAFCETS et du GEMMA, affectation des entrées / sorties, Les schémas de commande et de puissance des entrées et des sorties). Lancer l'exécution du programme (RUN ou MARCHE) Visualiser l'état des GRAFCET, des variables...

Objet graphique	nom
-| |-	Contact normalement ouvert
-|/|-	Contact normalement fermé
-|P|-	Contact fermé au front montant
-|N|-	Contact fermé au front descendant
-( )-	Bobine normalement ouverte
-(/)-	Bobine normalement fermée
-(S)- (ou –(L)-	Bobine Latch (maintenu à 1 une fois actionné
-(R)- (ou –(U)-	Bobine Reset (remise à 0 de la bobine latch)
-(P)-	Bobine active au front montant de son entrée
-(N)-	Bobine active au front descendant de son entrée
|-<return>	Retour unconditionnel (vers le sous-programme appelant)
|-cond-<return>	Retour conditionnel
|->>Label	Saut inconditionnel
|-cond->>Label	Saut conditionnel

3.10.1 Vérification du fonctionnement

Il existe deux façons de vérifier le fonctionnement : En simulation (sans Partie Opératrive) puis en condition réelle (avec Partie Opérative).

3.10.1.1 Simulation sans Partie opérative

- Valider les entrées correspondant à l'état initial (position) de la Partie Opérative.
- Valider les entrées correspondant aux conditions démarche du cycle.
- Vérifier l'évolution des grafcets (étapes actives).
- Vérifier les ordres émis (sorties).
- Modifier l'état des entrées en fonction des ordres émis (état transitoire de la P.O.).
- Modifier l'état des entrées en fonction des ordres émis (état final de la P.O.).
- Toutes les évolutions du GEMMA et des grafcets doivent être vérifiées.

3.10.1.2 Simulation avec Partie opérative (Conditions réelles)

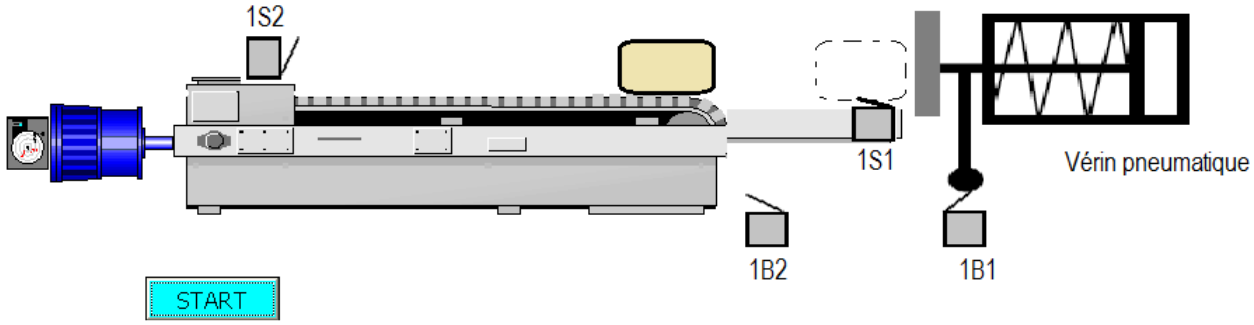
- Positionner la P.O. dans sa position initiale.
- Valider les conditions de marche du cycle.
- Vérifier l'évolution des grafcets et le comportement de la P.O.
- Toutes les évolutions du GEMMA et des grafcets doivent être vérifiées.

La programmation des API se décompose en trois étapes principale : la mise en grafcet de cahier de charge, la mise en equations de grafcet puis l'adressage et la programmation en un des langage de programmation des API. Pour bien comprendre un système de contrôle à base

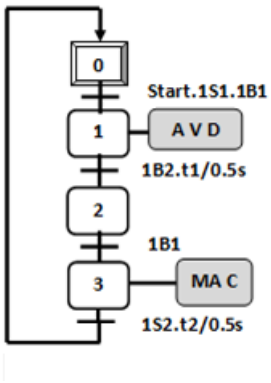
d'un API, on présente dans cette section un exemple d'application d'un API.

3.10.2 Exemple explicatif de la mise œuvre d'API

Soit le système présenté ci-dessous, ce système contient un vérin de distribution qui distribution de pièce du magasin vers le tapis ce dernier transporte la pièce jusqu'au bout du tapis devant capteur fin de course 1S2.



- Grafcet du système



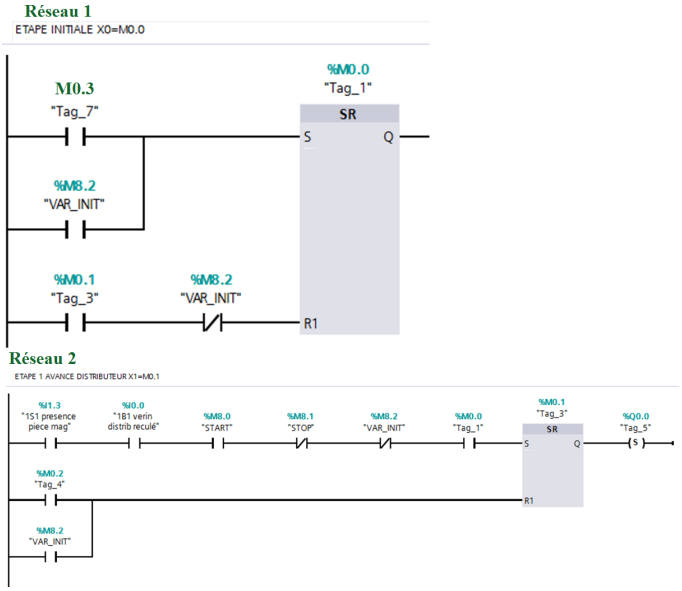
- Mise en equation du grafcet En utilisant les règles de franchissement et les equation des étapes à fin d'avoir le tableau des CA et CD et equation de chaque étape.

Xn	CAXn	CDXn	Xn=CA.Xn+$\overline{CDXn}$.Xn
X0	1S2.t2.X3+init	X1.$\overline{init}$	X0=1S2.X3+$\overline{X1}$.X0+init
X1	1S1.1B1.start.X0.$\overline{init}$	X2+init	X1= (1S1.1B1.Start.X0+$\overline{X2}$.X1).$\overline{init}$
X2	t1.1B2.X1.$\overline{init}$	X3+init	X2= (1B2.X1+$\overline{X3}$.X2).$\overline{init}$
X3	1B1.X2.$\overline{init}$	X0+init	X3= (1B1.X2+$\overline{X0}$.X3).$\overline{init}$

- Adressage des variables

- Programme en LD pour l'API S7-1200

Entrées d'API	Référence	adresse	Désignation
bouton Start	**Start**	**I8.0**	Bouton poussoir Start
bouton stop	**Stop**	**I8.1**	Bouton poussoir Stop
Vérin distributeur reculé	**1B1**	**I0.0**	Capteur ILS
Vérin distributeur avancé	**1B2**	**I0.1**	Capteur ILS
Présence pièce distributeur	**1S1**	**I1.3**	Capteur de proximité S1
Présence pièce convoyeur	**1S2**	**I1.4**	Capteur de proximité S2
Présence pièce distributeur	**1S1**	**I1.3**	Capteur de proximité S1
Présence pièce convoyeur	**1S2**	**I1.4**	Capteur de proximité S2
Sorties d'API	**Référence**	**adresse**	**Désignation**
Commande d'électrovanne de verin	**AVD**	**Q0.0**	Avance vérin distributeur
Commande de contacteur de convoyeur	**MAC**	**Q0.6**	Marche convoyeur



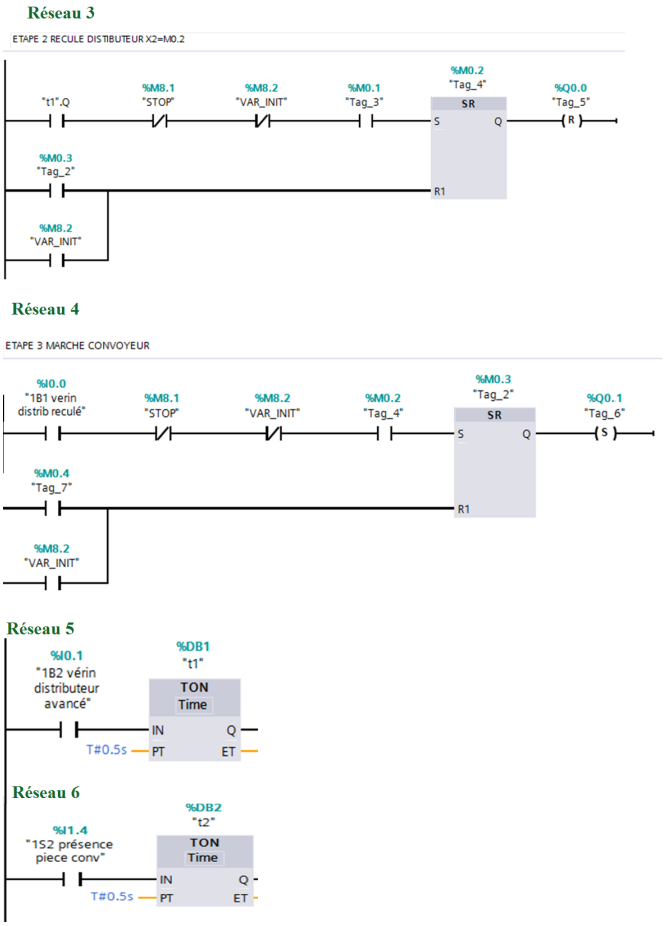
3.11 Réseaux d'automates

3.11.1 Interconnexion des APIs

L'interconnexion entre deux automates peut être réalisée très simplement en reliant une ou plusieurs sorties d'un automate à des entrées de l'autre et vice-versa.

3.11.2 réseau local industriel

Les réseaux locaux industriels sont des systèmes de communication entre des équipements qui interviennent directement dans le processus de fabrication des produits. Ces équipements sont des automates programmables, des calculateurs spécialisés dans le contrôle numérique, des



calculateurs de supervision, des équipements tels que des actionneurs électromécaniques et des capteurs intelligents.

Un réseau local industriel est utilisé dans une usine où tout système de production, pour connecter diverses machines afin d'assurer la commande, la surveillance, la supervision, la conduite, la maintenance, le suivi de production, la gestion, en un mot, l'exploitation de l'installation de production.

3.11.3 Réseau d'Atelier

Un réseau d'atelier est un système de communication permettant d'interconnecter des APIs, des terminaux d'atelier et des calculateurs. Il trouve ces principales applications dans les domaines de la :

- supervision industrielle,
- gestion de production,
- commande répartie de machines.

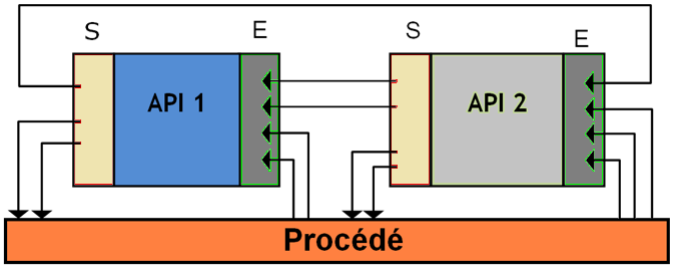


FIGURE 3.15 – Interconnexion simple (Entrées/Sorties) entre deux automates.

Les réseaux de terrain ont contribué à réaliser des gains de câblage importants, mais surtout ils ont rendu accessibles des services de diagnostic, de programmation, etc sur tout le sit.

3.11.3.1 Réseau d'Atelier PROFIBUS

Le Profibus fait partie des bus de terrain les plus utilisés dans le monde

3.11.4 Bus de terrain

Un bus de terrain est un système d'interconnexion d'appareils de mesure, de capteurs, actionneurs, etc. Le terme bus de terrain est utilisé par opposition au bus informatique. En effet, le bus de terrain est en général beaucoup plus simple, du fait des faibles ressources numériques embarquées dans les capteurs et actionneurs industriels. Les réseaux de terrain ou bus de terrain ou bus industriels permettent :

- la connexion entre plusieurs entités d'un même système sur un même support de communication et cela dans une zone géographique limitée (atelier, automobile, électronique embarquée, etc) ;
- le transport fiable de données sous une forme numérique de n'importe quel composant vers un autre ;
- l'ajout ou la suppression d'éléments au sein d'un même système (réduction ou extension du réseau) ;
- le travail en temps réel avec des protocoles de communication rapides.

Avantages des bus de terrain :

- Réduction des coûts de câblage et possibilité de réutiliser le matériel existant ;
- Réduction des coûts de maintenance.

Inconvénients des bus de terrain :

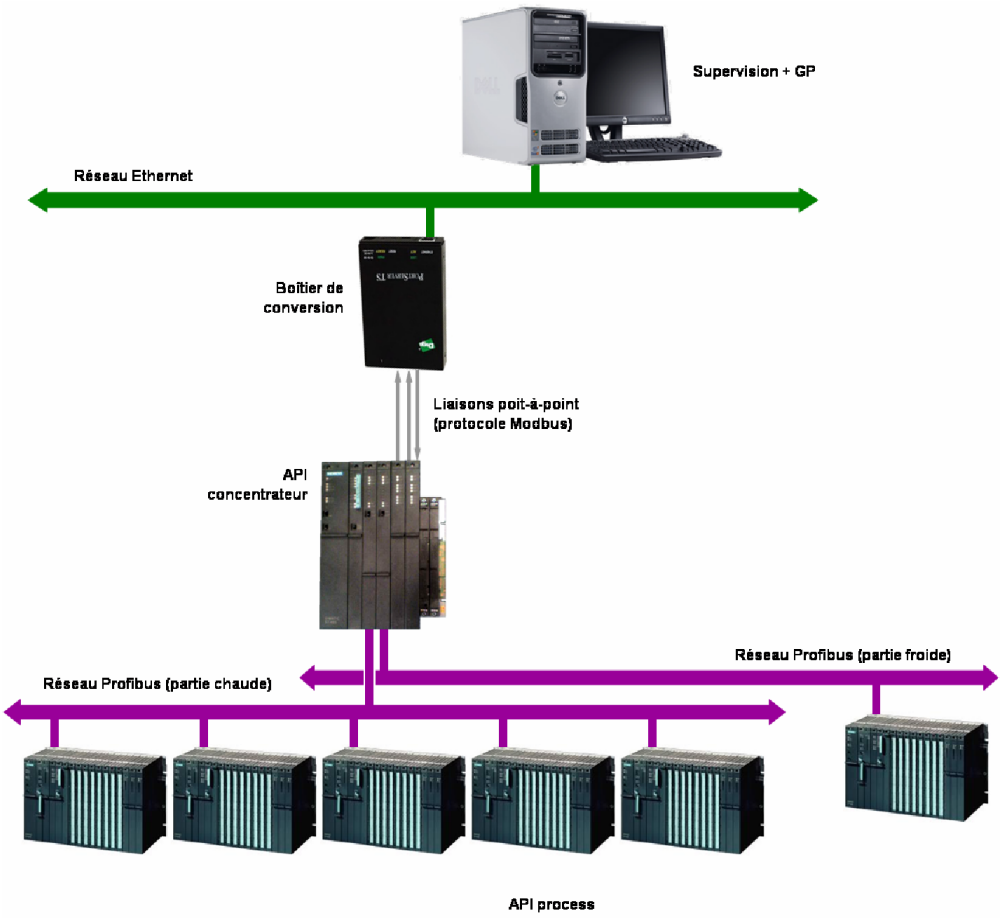


FIGURE 3.16 – Exemple du réseau des automates.

– Taille du réseau limitée
– Latence dans les applications à temps critique
– Coût global

3.11.5 Types de réseaux d'automates

Il excite différents types de topologies des réseaux industriels talque : Réseau en étoile, Réseau en anneau et Réseau hiérarchisé..etc.

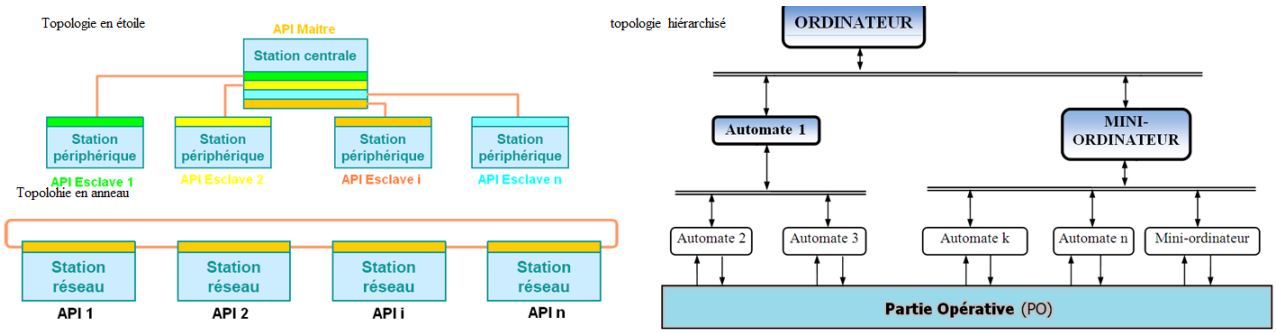


FIGURE 3.17 – différents types de topologies des réseaux industriels.

Chapitre 4

Applications en Électromécanique

Sommaire

4.1 Démarrage-Arrêt automatique des moteurs électrique

4.1.1 Démarrage-Arrêt automatique des moteurs CC

La figure 4.1 montre le circuit de puissance d'un MCC commandé par deux relais électromagnétiques. Avec cette structure (structure en pont), le moteur peut être alimenté par la source d'alimentation E ou E suivant l'état des relais KM1 (rotation droite)ou KM2 (rotation à gauche). Les demandes

de démarrage de moteur en sens droite ou à gauche peuvent être réalisées soit à l'aide de boutons poussoirs, respectivement S1 et S2. L'arrêt du moteur peut être réalisé à l'aide d'un bouton poussoir S3.

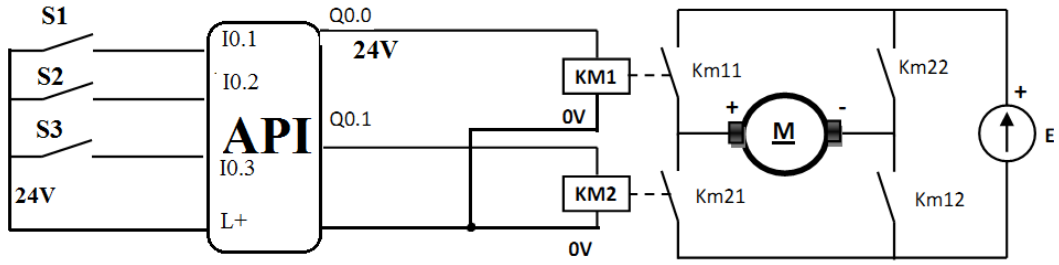


FIGURE 4.1 – Schéma de puissance pour commander un MCC en deux sens de rotation par API.

4.2 Démarrage-Arrêt automatique d'un moteur à courant alternatif

Le circuit de puissance des machines à courant alternatif (MCA), nécessite des appareillages de sectionnement, de protection et de commande, ce composant est un contacteur dont la tension de commande est 24V. Dont le cas où on utilise un API, le module de sortie TOR de ce dernier, fourni une tension de + 24V à son sorti Q0.0, la bobine sera alimentée par cette tension et ferme les trois contacts de puissance provoquant l'alimentation du moteur.

4.2.1 Commande d'un MCA en deux sens de rotation par API

Pour commander un moteur asynchrone triphasé en deux sens de rotations, le circuit de puissance nécessite deux contacteurs K1 et K2 (comme le montre la figure 4.3) nécessaires pour permuter les deux lignes de phases.

4.3 Commande de vérin pneumatiques par API

La commande des vérins pneumatiques nécessite un distributeur afin de distribuer la pression d'air pour faire sortir ou retirer la tige.

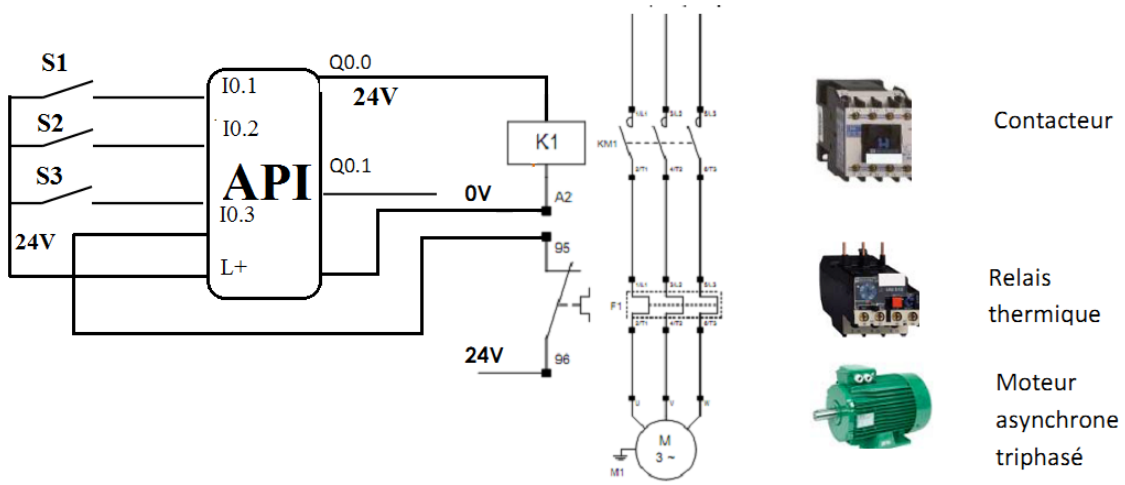


FIGURE 4.2 – Schéma de puissance pour commander un moteur asynchrone par API.

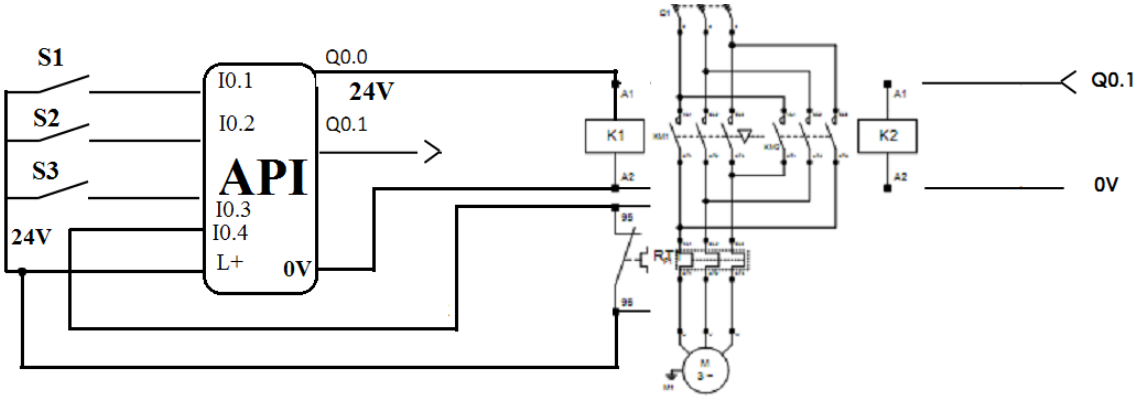


FIGURE 4.3 – Schéma de Commande d'un MCA en deux sens de rotation par API .

4.3.1 Commande de vérin simple effet

L'alimentation de la bobine du distributeur provoque la sortie de la tige du vérin. L'absence d'alimentation provoque le retour à la position de repos grâce au ressort de rappel.

4.3.2 Commande de vérin simple double effet

L'alimentation de la bobine A+ du distributeur provoque la sortie de la tige du vérin.En alimentant A-, la pression pneumatique fait rentrer la tige du vérin.

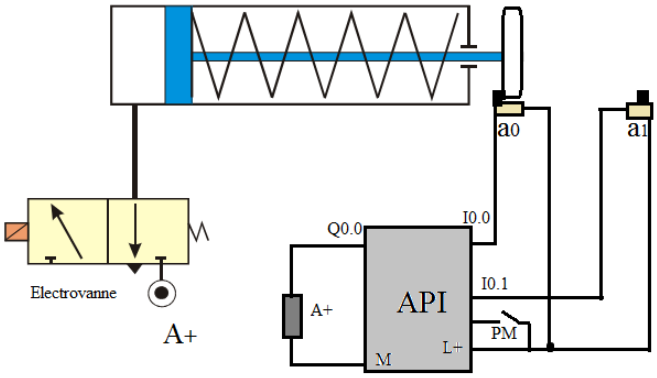


FIGURE 4.4 – Commande de vérin simple effet par un API.

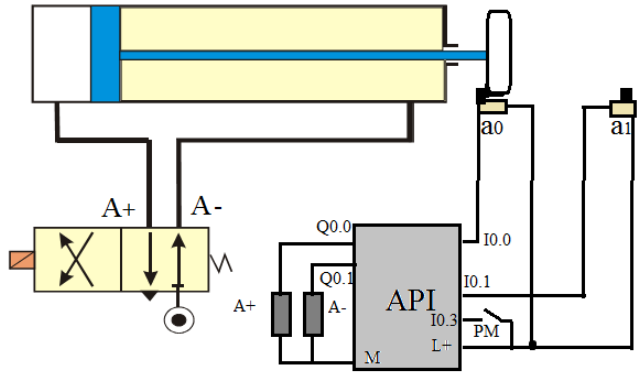


FIGURE 4.5 – Commande de vérin double effet par un API.

4.4 Automatisation d'un convoyeur

4.4.1 Définition du convoyeur

Le convoyeur est un système de manutention automatique qui permet de déplacer des produits finis ou bruts d'un poste à un autre par le mécanisme de transmission de puissance. Cette dernière est transmise d'un arbre moteur vers un ou plusieurs arbres récepteurs par l'intermédiaire de courroies ou de chaines.

4.4.2 Automatisation du convoyeur à bande

La figure suivante représente un convoyeur entraîné par un moteur électrique. Deux capteurs, "CAP 1" et "CAP 2", déterminent les positions extrêmes de la boîte. Le poste de commande comprend des boutons-poussoirs "Pm :marche" et "Ar :arrêt", ainsi qu'une lampe témoin

indiquant que le convoyeur est en marche

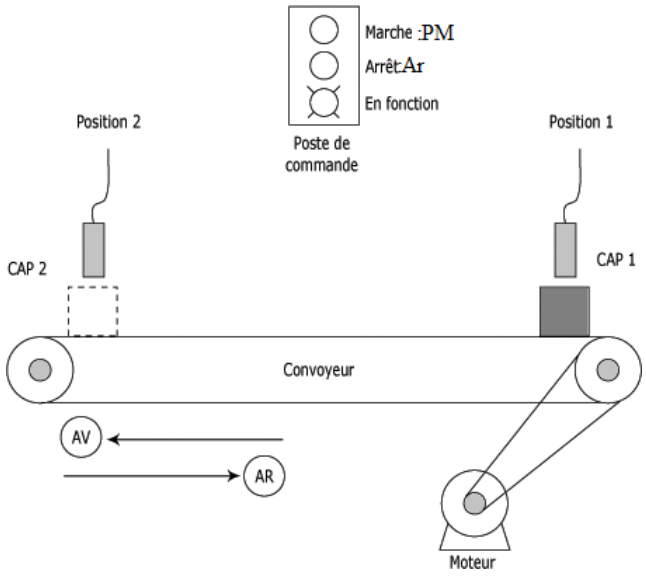


FIGURE 4.6 – Exemple du Convoyeur à bande.

4.4.3 Automatisation d'un élévateur de monte charge

Un monte charge, programmé pour desservir régulièrement les trois niveaux d'une société, se trouve à la mise sous tension au niveau 1, les portes ouvertes. L'opérateur lance le cycle en appuyant sur un bouton de départ cycle Dcy. Il y a alors, au bout d'un temps T0 de 5s, la fermeture des portes, la montée de la cabine jusqu'au niveau 2 puis l'ouverture des portes. Il y séjourne pendant un temps T1 de 5mn. Enfin il monte au niveau 3, y reste pendant un temps T2 de 5mn avant de redescendre au niveau 1 en position initiale .

Désignation des préactionneurs :

- OU : ouverture des portes
- FER : fermeture des portes
- KMH : contacteur moteur déplacement vers le haut
- KMB : contacteur moteur déplacement vers le bas

Capteur

- P1 : niveau 1
- P2 : niveau 2
- P3 : niveau 3
- Dcy : départ du cycle
- PO : portes ouvertes

– PF : portes fermées

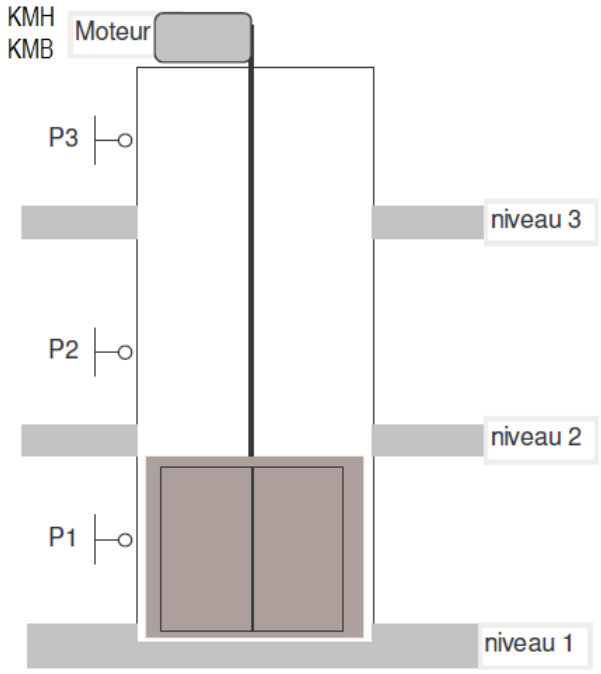


FIGURE 4.7 – Exemple d'un élévateur de monte charge .

4.4.4 Automatisation d'ascenseurs à traction électrique

Un ascenseur est un transport vertical assurant le déplacement en hauteur. Les ascenseurs à traction à câbles sont les types d'ascenseurs que l'on rencontre le plus, notamment dans les bâtiments tertiaires. Ils se différencient entre eux selon le type de motorisation :

– à moteur-treuil à vis sans fin,

– à moteur-treuil planétaire,

– à moteur à attaque directe (couramment appelé "Gearless" ou sans treuil),

Quel que soit le type, les ascenseurs à traction à câbles comprennent généralement (4.9) :

– une cabine,

– un contre-poids,

– des câbles reliant la cabine au contre-poids,

– des guides,

– un système de traction au-dessus de la cage de l'ascenseur,

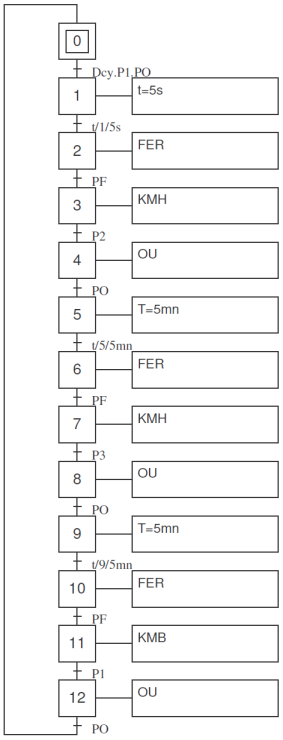


FIGURE 4.8 – Grafcet du élévateur de monte charge .

4.4.4.1 Ascenseur à 3 étages

Dans cet exemple de montré sur la figure 4.9, Cette cabine est entraînée par un moteur électrique à deux sens de marche KM1 et KM2 (montée et descente). La présence de la cabine à un étage est détectée par un capteur à chaque niveau (P1, P2 et P3). Elle doit s'arrêter lorsqu'elle rencontre le contact de l'étage qui a été demandé. Les commandes du moteur sont KM1 pour la montée, KM2 pour la descente et aucune action pour l'arrêt. A chaque arrêt de l'ascenseur à un étage, nous attendons l'ouverture des portes et nous déclenchons une temporisation de 3 secondes. Au bout de ce temps, si les portes sont refermées, nous relançons l'ascenseur pour servir le prochain appel.

Ordres

- **KM1** :Montée cabine
- **KM2** :Descente cabine
- **OP** :Ouverture porte
- **FP** :Fermeture porte
- **E1,E2,E3** : Bouton poussoir appel 1^{er},2^{eme} et 3^{eme} étage, successivement
- **Capteurs**

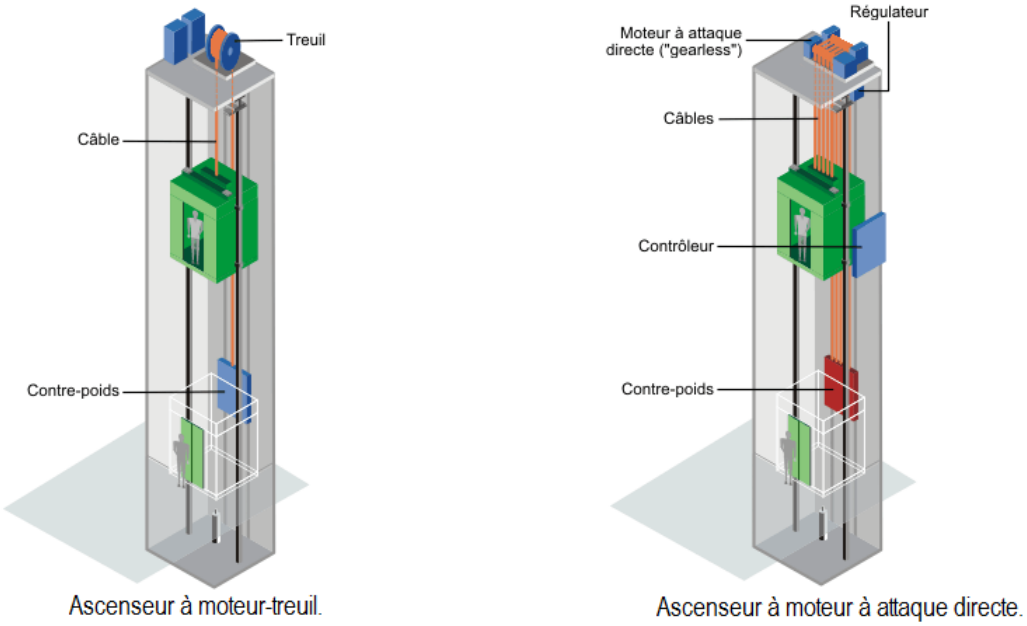


FIGURE 4.9 – Exemple de types des ascenseurs à câble.

– **a** :porte ouverte
– **b** :porte fermée
– **P1,P2,P3** :position de la cabine

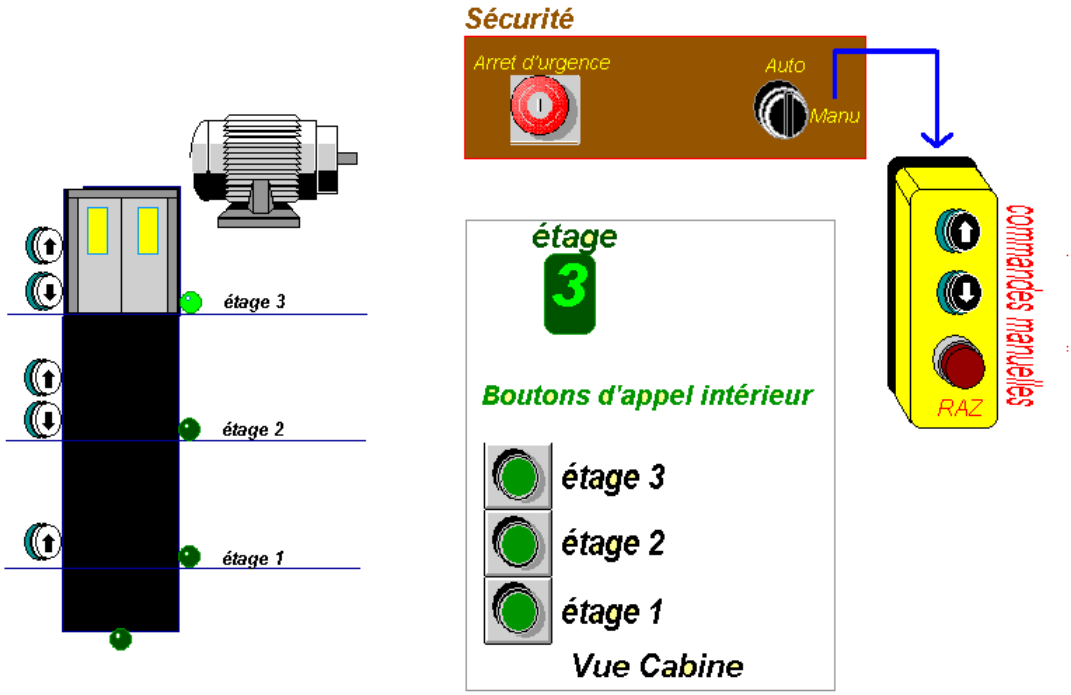


FIGURE 4.10 – Exemple du ascenseur à 3 étages.

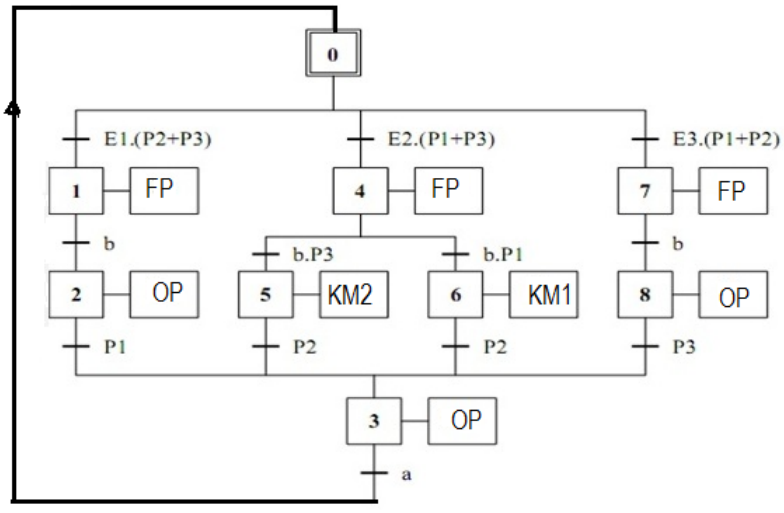


FIGURE 4.11 – Grafcet d'un ascenseur à 3 étages.

Références

- Gérard Boujat et Patrick Anaya. Automatique industrielle en 20 fiches. Dunod. 2013.
- Jean-Yves Fabert. Automatismes et Automatique : Cours et Exercices Corrigés. Edition Ellipses, 2003.
- Frederic P.Miller, Agnes F.Vandome, John McBrewster.Automates Programmables Industriels : Programmation informatique. Edition Alphascript Publishing 2010.
- G. Michel. Les API : Architecture et applications des automates programmables industriels. Edition Dunod 1988.
- Dr. SAADOUN ACHOUR. Automatisme industriel. support de cours. Département de génie électrique,univeristé de Biskra ; 2013.
- William Bolton, « Automates Programmables Industriels », DUNOD, Paris, 2015.
- C.VRIGNON et M.THENAISIE, « l'automatisation », ISTIA, 17 octobre 2005.
- L. BERGOUGNOUX, « Automates Programmables Industriels », POLYTECH Marseille,Département de Mécanique Energétique, 2005.
- PHILIPPE HOARAU, « L'Automate Programmable Industriel », TS MAI, http ://bannaladi.fr/cours/Traitement/API.pdf.