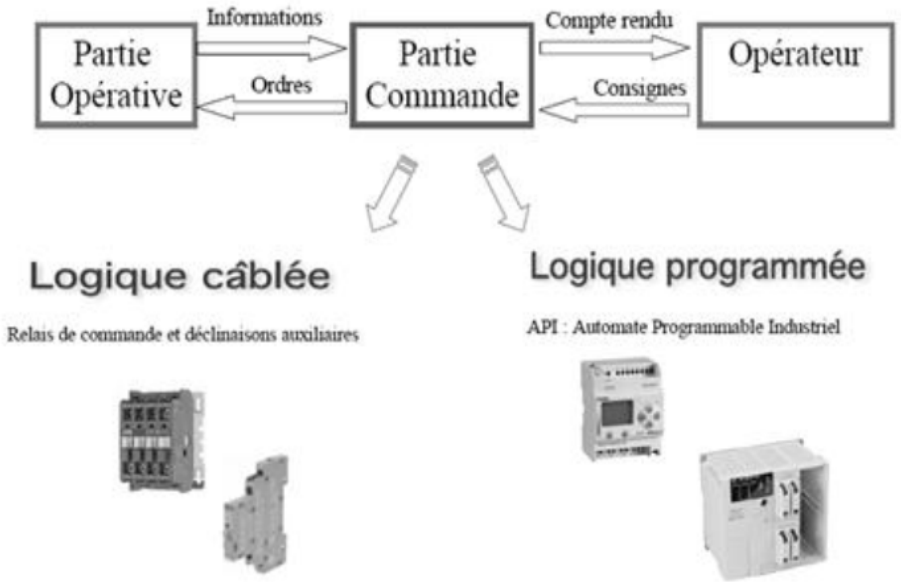
Chapitre III

Les Automates Programmables Industriels

III.1 Introduction

La partie commande d'un automatisme est le centre de décision. Il donne des ordres à la partie opérative et reçoit ses comptes rendus. La partie commande peut être mécanique (logique câblée), électronique (logique programmée) ou autre.

- **La logique câblée**: L'élément principal s'appelle module séquenceur et l'association de modules constitue un ensemble appelé séquenceur. Le pilotage des distributeurs se fait par une action de l'air comprime sur un piston qui fait déplacer le tiroir du distributeur à droite ou à gauche. L'ensemble, appelé tout pneumatique ou relais électromagnétiques, est homogène, fiable, non flexibile, et pas de communication possible.
- **La logique programmée**: L'élément principal de commande électrique s'appelle l'Automate Programmable Industriel ou l'API à base de microprocesseurs permettant une modification aisée des systèmes automatisés. La détection est électrique. Le pilotage des actionneurs se fait par l'intermédiaire de relais ou de distributeurs.



Logique câblée		*Logique programmée*
Avantages	**Inconvénients**	**Solutions**
utilisation de relais électromagnétiques et de systèmes pneumatiques pour la réalisation des parties commandes	• cher, • pas de flexibilité, • pas de communication possible	utilisation de systèmes à base de *microprocesseurs* permettant une modification aisée des systèmes automatisés

Fig. III.1. Logique câblée vers logique programmée

III.2 Définition

Un Automate Programmable Industriel (API) (en anglais *Programmable Logic Controller*, *PLC*) est une machine électronique programmable par un personnel non informaticien. L'API réalise des fonctions d'automatisme pour assurer la commande de pré-actionneurs et d'actionneurs à partir d'informations logique, analogique ou numérique.

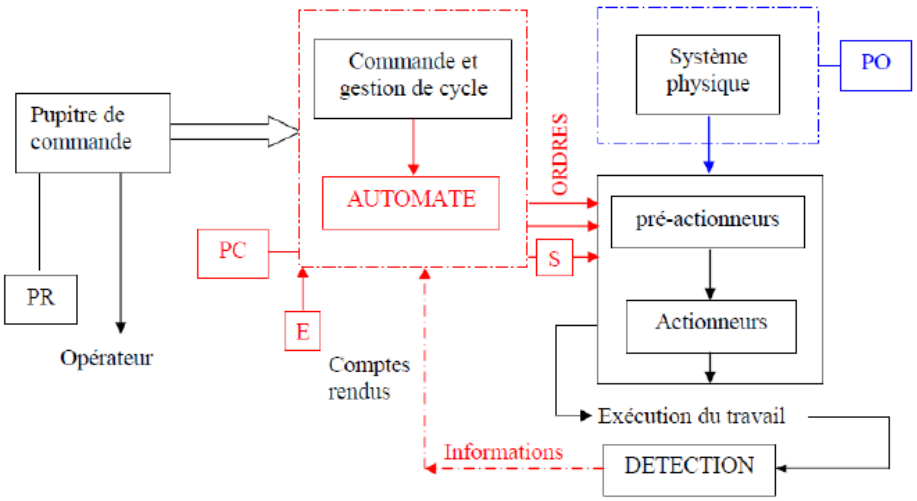


Fig. III.2. Situation de l'automate dans un système automatisé de production

On utilise les API dans tous les secteurs industriels pour la *commande des machines* (convoyage, emballage, ...) ou des *chaînes de production* (automobile, agroalimentaire, ...) ou il peut également assurer des fonctions de *régulation de processus* (métallurgie, chimie ...). Il est de plus en plus utilisé dans le domaine du *bâtiment* (tertiaire et industriel) pour le contrôle du chauffage, de l'éclairage, de la sécurité ou des alarmes.

III.3 Structure de l'API

Cet ensemble électronique gère et assure la commande d'un système automatisé. Il se compose de plusieurs parties et notamment d'une mémoire programmable dans laquelle l'operateur écrit, dans un langage propre à l'automate, des directives concernant le déroulement du processus à automatiser. Son rôle consiste donc a fournir des ordres à la partie opérative en vue d'exécuter un travail précis comme par exemple la sortie ou la rentrée d'une tige de vérin, l'ouverture ou la fermeture d'une vanne. La partie opérative lui donnera en retour des informations relatives à l'exécution du travail.

L'Automate Programmable Industriel (API) peut être de type *compact* ou *modulaire*.

- **Automate *compact* :** Il intègre le processeur, l'alimentation, les entrées et les sorties. Selon les modèles et les fabricants, il pourra réaliser certaines fonctions supplémentaires (comptage rapide, E/S analogiques ...) et recevoir des extensions en nombre limité.
- **Automate *modulaire***, le processeur, l'alimentation et les interfaces d'entrées/sorties résident dans des unités séparées (*modules*) et sont fixées sur un ou plusieurs *racks*

contenant le "fond de panier" (bus plus connecteurs). Ces automates sont intégrés dans les automatismes complexes où puissance, capacité de traitement et flexibilité sont nécessaires.

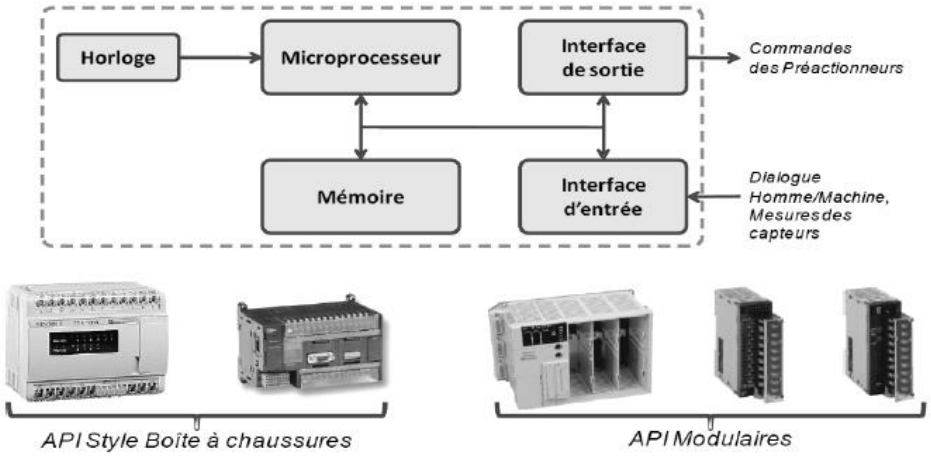
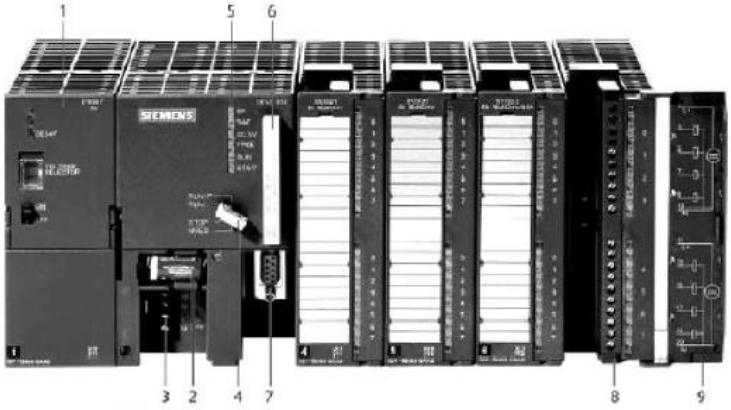


Fig. III.3. Architecture de l'API

Les APIs comportent quatre parties principales :

- Une mémoire ;
- Un processeur ;
- Des interfaces d'Entrées/Sorties ;
- Une alimentation (240 Vac ! 24 Vcc).



Automate modulaire (Siemens)

1 Module d'alimentation
2 Pile de sauvegarde
3 Connexion au 24V cc
4 Commutateur de mode (à clé)
5 LED de signalisation d'état et de défauts
6 Carte mémoire
7 Interface multipoint (MPI)
8 Connecteur frontal
9 Volet en face avant

Fig. III.4. Aspect extérieur de l'API

Ces quatre parties sont reliées entre elles par des bus (ensemble câble autorisant le passage de l'information entre ces 4 secteurs de l'API). Ces quatre parties réunies forment un ensemble compact appelé automate.

- **Le processeur :** Son rôle consiste d'une part à organiser les différentes relations entre la zone mémoire et les interfaces d'E/S et d'autre part à gérer les instructions du programme.
- **Les interfaces :**
  - L'interface d'Entrées comporte des adresses d'entrée, une pour chaque capteur relie.
  - l'interface de Sorties comporte des adresses de sorties, une pour chaque pré-actionneur. Le nombre d'E/S varie suivant le type d'automate. Les cartes d'E/S ont une modularité de 8, 16 ou 32 voies. Elles admettent ou délivrent des tensions continues 0 - 24 Vcc.
- **La mémoire :** Elle est conçue pour recevoir, gérer, stocker des informations issues des différents secteurs du système que sont le terminal de programmation (PC ou console) et le processeur, qui lui gère et exécute le programme. Elle reçoit également des informations en provenance des capteurs. Il existe dans les automates plusieurs types de mémoires qui remplissent des fonctions différentes : la conception et l'élaboration du programme font appel `a la RAM et l'EEPROM.
- **L'alimentation :** Tous les automates actuels utilisent un bloc d'alimentation alimente en 240Vac et délivrant une tension de 24 Vcc.

III.4 Traitement du programme automate

Tous les automates fonctionnent selon le même mode opératoire (fig.III.5) :

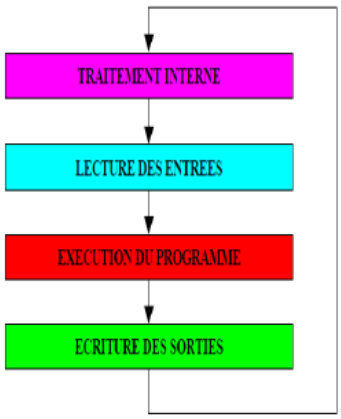


Fig. III.5. Traitement du programme automate

- **Traitement interne** : L'automate effectue des opérations de contrôle et met à jour certains paramètres systèmes (détection des passages en RUN / STOP, mises à jour des valeurs de l'horodateur, ...).
- **Lecture des entrées** : L'automate lit les entrées (de façon synchrone) et les recopie dans la mémoire image des entrées.
- **Exécution du programme** : L'automate exécute le programme instruction par instruction et écrit les sorties dans la mémoire image des sorties.

- **Ecriture des sorties** : L'automate bascule les différentes sorties (de façon synchrone) aux positions définies dans la mémoire image des sorties.

Ces quatre opérations sont effectuées continuellement par l'automate (fonctionnement cyclique).

On appelle *scrutation* l'ensemble des quatre opérations réalisées par l'automate et le *temps de scrutation* est le temps mis par l'automate pour traiter la même partie de programme. Ce temps est de l'ordre de la dizaine de millisecondes pour les applications standards. *Le temps de réponse total (TRT)* est le temps qui s'écoule entre le changement d'état d'une entrée et le changement d'état de la sortie correspondante : Le temps de réponse total est au plus égal à deux fois le temps de scrutation (sans traitement particulier). Le temps de scrutation est directement lié au programme implanté. Ce temps peut être fixé à une valeur précise (fonctionnement périodique), le système indiquera alors tout dépassement de période (fig. III.6).

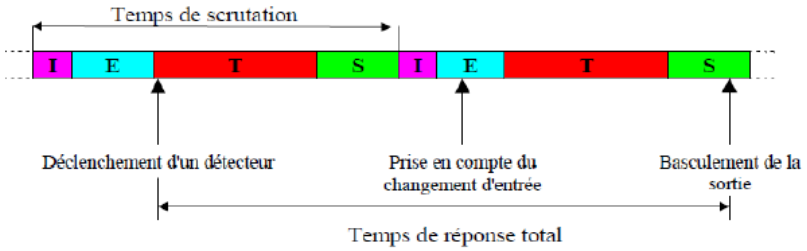


Fig. III.6. Temps caractéristiques d'un API

Dans certains cas, on ne peut admettre un temps de réponse aussi long pour certaines entrées : ces entrées pourront alors être traitées par l'automate comme des événements (traitement événementiel) et prises en compte *en priorité* (exemples : problème de sécurité, coupure d'alimentation ...). Certains automates sont également pourvues d'entrées rapides qui sont prises en compte avant le traitement séquentiel mais le traitement évènementiel reste prioritaire.

III.5 Programmation à l'aide du *GRAFCET*

Il existe 5 langages de programmation des automates (fig. III.7.) qui sont normalisés au plan mondial par la norme CEI 61131-3. Chaque automate se programmant via une console de programmation propriétaire ou par un ordinateur équipé du logiciel constructeur spécifique.

Langages textuels :	
- Liste d'instructions	*IL (Instruction List)*
- Texte structuré	*ST (Structured Text)*
Langages graphiques :	
- Boîtes fonctionnelles	*FBD (Function Block Diagram)*
- Schéma contact	*LD (Ladder Diagram)*
- Grafcet	SFC (Sequential Function Chart)

Fig. III.7. Langages de programmation des APIs.

III.5.1 Le GRAFCET (GRAphe Fonctionnel de Commande Etape/Transition)

Le GRAFCET (GRAphe Fonctionnel de Commande Etape/Transition) est un outil graphique de description des comportements d'un système logique [6]. Il est très utilisé pour la programmation des automates programmables industriels (API). Il est composé d'*étapes*, de *transitions* et de *liaisons* :

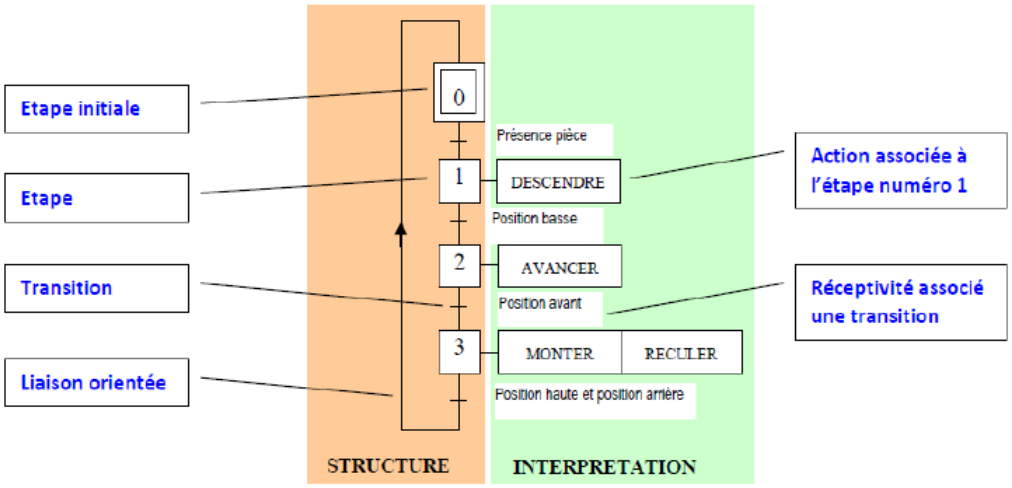


Fig. III.8. Structure d'un GRAFCET.

Étape: correspond à une phase durant laquelle on effectue une ACTION pendant une certaine durée. On numérote chaque étape 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. Une étape est dite active lorsqu'elle correspond à une phase "en fonctionnement", c-à-d qu'elle effectue l'action qui lui est associé. On représente quelquefois une étape active à un instant donné en dessinant un point à l'intérieur.

Étape initiale: c'est une étape active au début du fonctionnement. Elle se représente par un double carré.

Liaison: est un arc orienté (ne peut pas être parcouru que dans un sens). Elles relient les étapes aux transitions et les transitions aux étapes. Le sens général d'évolution est du haut vers le bas. Dans le cas contraire, des flèches doivent être employées

Transitions **:** une transition indique une possibilité d'évolution d'activité entre deux ou plusieurs étapes. Cette évolution s'accomplit par le franchissement de la transition. C'est une condition de passage d'une étape à une autre.

Réceptivité : La réceptivité associée à une transition est une fonction logique, qui peut être vraie ou fausse est associée à chaque transition.

Action*:* L'action indique, dans un rectangle, comment agir sur la variable de sortie, soit par assignation (action continue), soit par affectation (action mémorisée)

III.5.2 Règles d'évolution

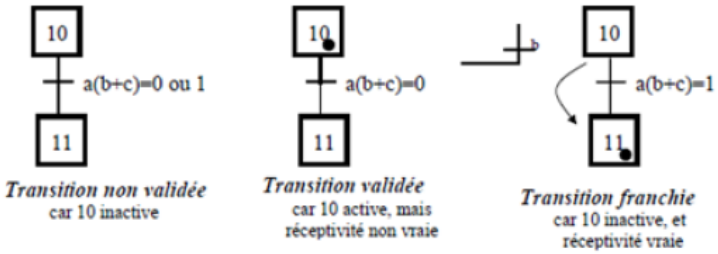
Règle 1 : ***Situation initiale***

L'étape initiale caractérise le comportement de la partie commande d'un système en début de cycle. Elle correspond généralement à une positon d'attente. L'étape initiale est activée sans condition en début de cycle. Il peut y avoir plusieurs étapes initiales dans un même grafcet.

Règle 2 : ***Franchissement d'une transition***

Une transition est validée lorsque toutes les étapes, immédiatement précédentes reliées à cette transition, sont actives. Le franchissement d'une transition se produit :

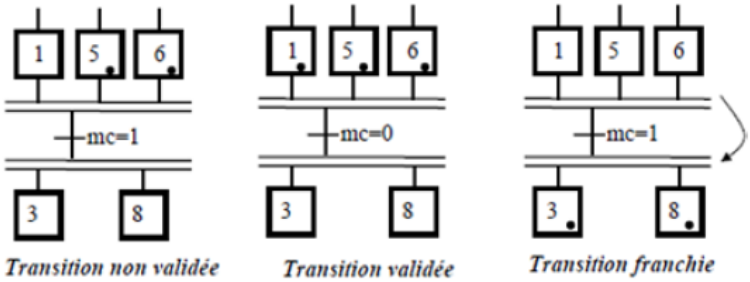
- lorsque la transition est VALIDÉE ;
- ET QUE la réceptivité associée à cette transition est VRAIE.



Règle 3 : ***Évolution des étapes actives***

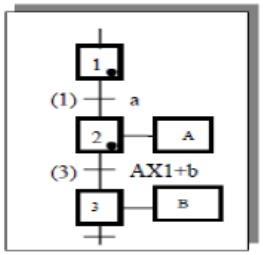
Le franchissement d'une transition provoque simultanément :

- L'activation de toutes les étapes immédiatement suivantes.
- La désactivation de toutes les étapes immédiatement précédentes.



Règle 4 : ***Évolutions simultanées***

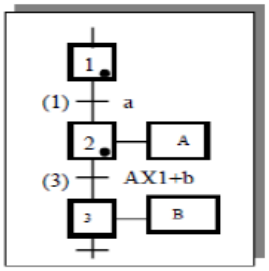
Plusieurs transitions simultanément franchissables sont simultanément franchies.



Les étapes 1 et 2 sont actives (X1=X2=1). les transitions (1) et (2) sont alors validées, lorsque a=1 (2) est aussi vraie **alors (2) et (3) simultanément franchissables** sont **simultanément franchis.**

Règle 5 : ***Activation et désactivation simultanée d'une même étape***

Si au cours d'une évolution, une même étape se trouve être à la fois activée et désactivée, elle reste active.



L'étape 3 simultanément activée et désactivée **reste active**

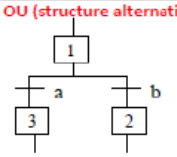
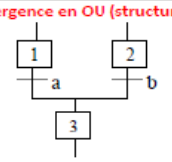
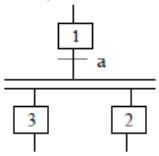
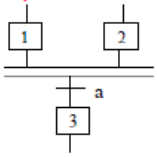
III.5.3 Les structures de base

a. Séquence unique

C'est une suite d'étapes pouvant être activées les unes après les autres.

b. Séquences simultanées et alternatives

Plusieurs séquences sont actives en même temps, après le franchissement d'une transition.

Divergence en OU (structure alternative) :  Si 1 active et si a seul, alors désactivation de 1 et activation de 3, 2 inchangé. Si a et b puis 1 active alors désactivation 1, activation 2 et 3 quel que soit leur état précédent. (règle 4)	**Convergence en OU (structure alternative) :**  Si 1 active et a sans b, alors activation de 3 et désactivation de 1, 2 reste inchangé Si 1 et 2 et a et b alors 3 seule active
Divergence en ET (structure simultanée) :  SI 1 active et SI a, alors désactivation de 1 et activation de 2 ET 3.	**Convergence en ET (structure simultanée) :**  SI 1 active seule et a, alors aucun changement. Si 1 ET 2 et a, alors activation de 3 et désactivation de 1 et 2.