Partie 01 Automates Programmables Industriels

1. De la logique câblée aux microprocesseurs

En **logique câblée**, la loi de commande est réalisée en interconnectant judicieusement des opérateurs matériels réalisant des fonctions logiques de base [AND, OR, NOT], [porte NAND], [porte NOR], [relais normalement ouvert, relais normalement fermé].

Suivant la technologie adoptée, il peut s'agir d'opérateurs fluidiques (interconnectés par tuyauteries), de relais électromagnétiques ou de relais statiques (interconnectés par fil).

À la fin des années 60, Un fabricant américain de voitures décide de remplacer les systèmes de commande à base de logique câblée par une **logique programmée** (utilisation d'un automate programmable muni d'un microprocesseur).

Elle correspond à une démarche séquentielle, seule une opération élémentaire est exécutée à la fois, c'est un traitement série. Le schéma électrique est transcrit en une suite d'instruction qui constitue le programme. En cas de modification des équations avec les mêmes accessoires, l'installation ne comporte aucune modification de câblage seul le jeu d'instructions est modifié.

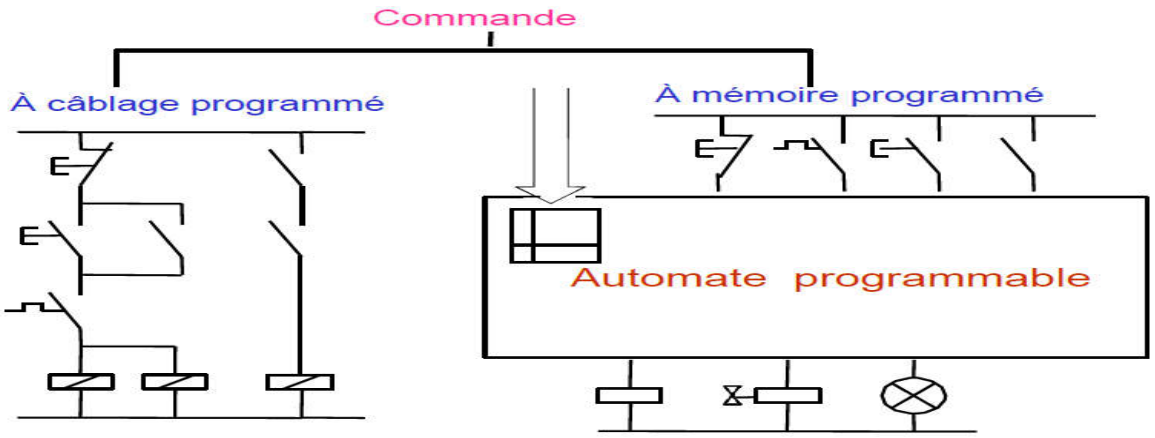

Figure 01. Schéma de commande (logique câblée vs logique programmée)

	Logique câblée	Logique programmée
Avantages	▪ Vitesse car fonctionnement simultané des opérateurs	▪ Banalisation du matériel ▪ Facilité de modification de la loi de contrôle ▪ Faible liaison entre le volume matériel et la complexité du problème. ▪ Possibilité de communication avec l'extérieur
Inconvénients	▪ Encombrement (poids et volume) ▪ Manque de souplesse ▪ Difficulté pour maîtriser les problèmes complexes ainsi	▪ Besoin de formation ▪ Plantage ▪ Vitesse inversement proportionnelle à la complexité du

que ceux liés au dépannage ▪ Pas de communication possible	problème.

Le microprocesseur à évolué dans trois directions :

- **Ordinateurs**: complexes et puissants, ils disposent de nombreux périphériques et d'une Interface Homme/Machine 'IHM' très évoluée.
- **Automates Programmables Industriels (API)** : ils sont orientés vers des applications industrielles et disposent d'interface donc de puissance permettant de commander différents actionneurs.
- **Microcontrôleurs** : Il s'agit ici de mettre dans le même boîtier un nombre maximum de circuits ce qui rassemble les éléments essentiels d'un ordinateur.

2. Automates programmables industriels

2.1 Définition

Un **automate programmable** est un appareil dédié au contrôle d'une machine ou d'un processus industriel, constitué de composants électroniques, comportant 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, au jeu d'instructions volontairement réduit, destiné à la conduite et la surveillance en temps réel de processus industriels.

Trois caractéristiques fondamentales distinguent totalement l'API des outils informatiques tels que les ordinateurs (PC industriel ou autres):

- 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, microcoupures 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 œuvre et son exploitation ne nécessitent aucune connaissance en informatique.

Les automates peuvent être de type **monobloc** '**compact**' ou **modulaire**.

- De type **monobloc 'compact'**, possède un nombre d'entrées et de sorties restreint et son jeu d'instructions ne peut être augmenté. Le type monobloc a pour fonction de résoudre des automatismes simples faisant appel à une logique séquentielle et utilisant des informations tout-ou-rien.

- De type **modulaire**, adaptable à toutes situations. Ces automates sont intégrés dans les automatismes complexes où puissance, capacité de traitement et flexibilité sont nécessaires.

Figure 02. Automate monobloc

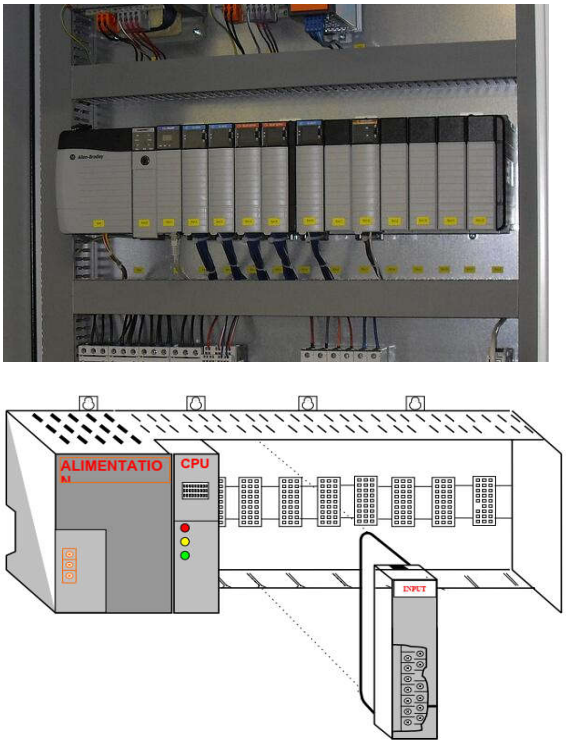

Figure 03. Automate modulaire

2.2 Principe de fonctionnement

Le moniteur d'exécution d'un API peut être composé de plusieurs sous-programmes appelés tâches. Une tâche est un ensemble d'opérations programmées pour s'exécuter successivement, puis s'arrêter jusqu'au prochain lancement. Dans API, une tâche est:

- **cyclique** : la tâche est immédiatement relancée après sa fin,

- **périodique** : la tâche est relancée toutes les T unités de temps,
- **événementielle** : la tâche est lancée à chaque fois qu'un événement prédéfini se produit.

Durant son fonctionnement, un API exécute le même cycle de fonctionnement qu'on appelle "cycle automate" ; la durée de ce cycle est typiquement de 1 à 50 ms:

- Avant chaque traitement, l'API lit les entrées et les mémorise durant le cycle automate ;
- Il calcule les équations de fonctionnement du système en fonction des entrées et d'autres variables et les mémorise ;
- Les résultats sont recopiés dans les sorties.

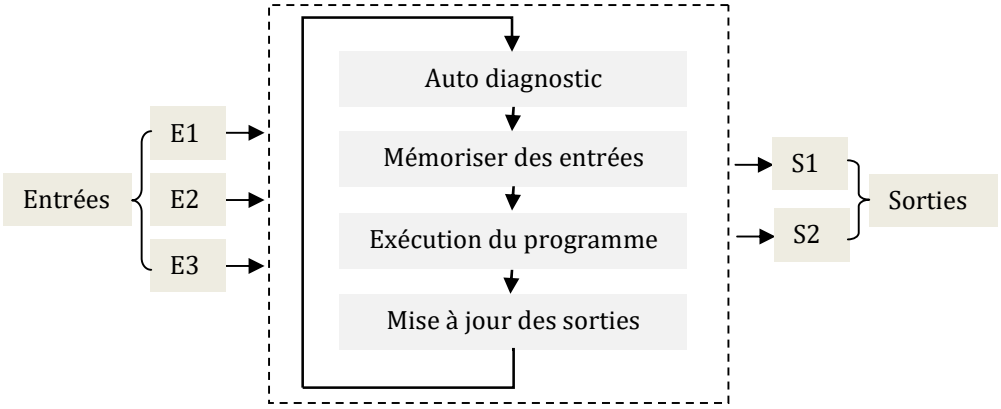

Figure 04. Mode opératoire 'Cycle d'un API'

2.3 Structure interne d'un automate programmable industriel

La structure matérielle interne d'un API obéit au schéma donné sur la figure suivante :

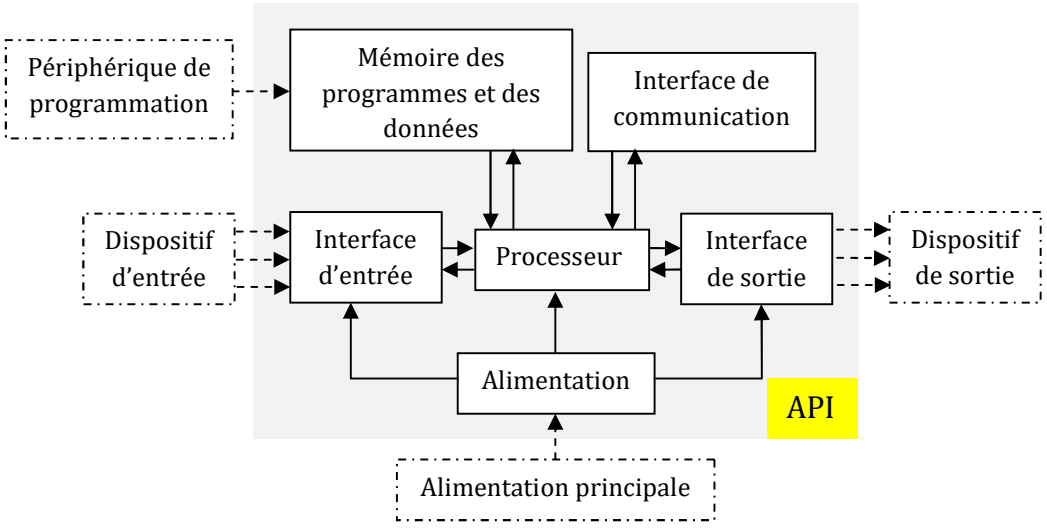

Figure 05. Structure d'un API

2.3.1 Processeur ou unité centrale de traitement (CPU, Central Processing Unit)

Le CPU interprète les signaux d'entrée et effectue les actions de commandes conformément au programme stocké en mémoire, en communiquant aux sorties les décisions sous forme de signaux d'action.

Les opérations effectuées par cette unité peuvent être réparties sur deux catégories:

- **Logiques:** And, Or, Xor, Not, Shift et Rotate (Et, Ou, Ou Exclusif, Non, Decalage et Rotation).
- **Arithmétiques:** l'addition, la soustraction, la multiplication, la division.

Figure 06. Unité centrale de traitement 'CPU'

2.3.2 Module d'alimentation

Il assure la distribution d'énergie électrique aux différents modules. Donc, il est nécessaire pour convertir la tension d'entrée alternative (220V) du secteur en une tension continue (24V, 48V...) nécessaire au processeur et aux circuits des modules d'interface d'entrée et de sortie.

2.3.3 Mémoires

La mémoire contient le programme qui définit les actions de commande effectuées par le microprocesseur. Elle contient également les données qui provient des entrées en vue de leur traitement, ainsi que celle des sorties.

Deux familles de mémoires sont utilisées dans les API :

*** Les **mémoires vives**, ou **mémoires à accès aléatoire** «**Random Access Memory (RAM)**». Le contenu de ces mémoires peut être lu et modifié à volonté, mais il est perdu en cas de manque de tension (mémoire volatiles). Elles nécessitent par conséquent une sauvegarde par batterie. Les mémoires vives sont utilisées pour l'écriture et la mise au point du programme, et pour le stockage des données.

*** Elles sont à lecture seule (**mémoires mortes**), les informations ne sont pas perdues lors de la coupure de l'alimentation des circuits. On peut citer les types suivants :

- **ROM «Read Only Memory»** : Elle est programmée par le constructeur et son programme ne peut être modifié.
- **PROM «Programmable ROM»** : Elle est livrée non enregistrée par le fabricant. Lorsque celle-ci est programmée, on ne peut pas l'effacer.
- **EPROM «Erasable PROM»** : C'est une mémoire PROM effaçable par un rayonnement ultraviolet intense.
- **Mémoire Flash** : C'est une mémoire EEPROM rapide en programmation. L'utilisateur peut effacer un bloc de cases ou toute la mémoire.

On peut les classer par leur **utilisation** ou leur **contenu (classement logiciel des mémoires):**

***** Mémoire programme :** contient la liste ordonnée des instructions à traiter par le processeur.

***** Mémoire de données :** sauvegarde les entrées en provenance des périphériques d'entrées (capteurs, Boutons de pupitre,…), les résultats intermédiaires de calcul du processeur, les données de sorties (pré-actionneurs, afficheur,…).

2.3.4 Modules d'entrées/sorties 'I/O modules'

Les modules d'entrées/sorties (I/O) dans un API sont des interfaces entre le système (CPU) et le monde extérieur (processus).

Un **module d'entrée** est un circuit électronique qui permet la conversion d'une valeur analogique en une valeur numérique (CAN) exploitable par le processeur de la carte CPU. De même, un **module de sortie** est un circuit électronique qui permet la conversion des données numérique, venus depuis la carte CPU, en données analogiques exploitables par le processus. Cependant, le processeur de la carte CPU doit avoir des entrées et des sorties numériques avec un niveau de tension entre 0 et 5V.

A. Interface d'entrée

A.1 Interface Tout on Rien (TOR): À partir d'un signal quelconque en entrée, les interfaces fournissent en sortie deux tension 0V ou MAX V (MAX= 5 /10 /24 V). Ces interfaces sont de type à contacte. Ce type d'entrée n'ayant que deux états (ON/OFF, OUVERT/FERMÉ, VRAI/FAUX, etc.).

Entrées TOR
→ Disjoncteur
→ Fin de course
→ Boutons poussoir
→ Contacts de relais
→ Contacts du démarreur
→ Détecteur de niveau
→ Etc.

A.2 Interface d'entrée analogique: l'information traitée est continue et prend une valeur qui évolue dans une plage bien déterminée. La variation de la grandeur d'entrée est convertie en une variation :

- **En tension :** de 0V, à10V
- **En intensité :** de 0 Ma à 20mA, ou de 4 mA à 20mA

Entrées analogiques
→ Capteur de débit → Capteur de pression → Capteur de température → Capteur de vitesse → Potentiomètres → Positionneur de vanne → Etc.

Le principe est résumé par le schéma fonctionnel suivant :

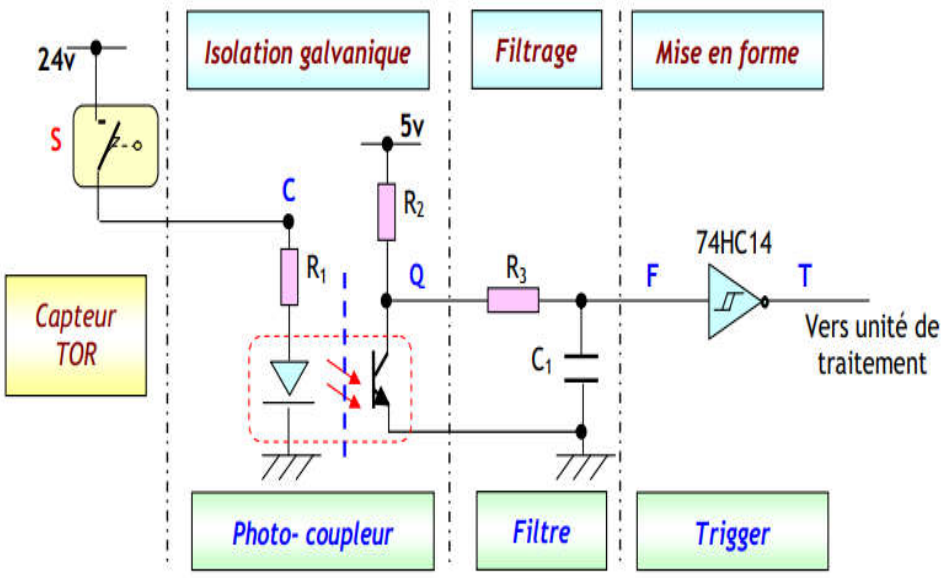

Figure 07. Circuit d'entrée

B. Interface de sortie

B.1 Interface Tout on Rien (TOR): Comme les modules d'entrées TOR, les modules de sorties TOR connectent des dispositifs de sorties (actionneurs ou prés actionneurs) de champ à l'automate programmable. Ce type de sortie n'ayant que deux états (On/Off, Ouvrir/Fermer, Vrai/Faux, etc.).

B.2 Interface de sortie analogique : Les cartes de sorties analogiques sont utilisées dans les applications nécessitant le contrôle des actionneurs répondant à des niveaux de tension ou de courant d'une façon continu.

Comme les entrées analogiques, les cartes de sorties analogiques sont généralement connectées à des dispositifs de contrôle via des transducteurs. Ces transducteurs amplifient, réduisent ou modifient le signal de tension en un signal analogique qui, à son tour de contrôler l'actionneur de sortie.

Le principe est résumé par le schéma fonctionnel suivant :

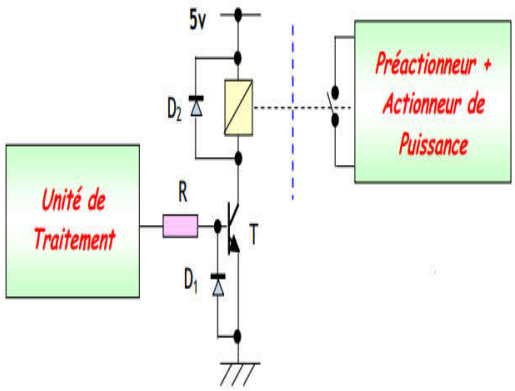

Figure 08. Circuit de sortie

2.4 Programmation des API

La programmation d'un API consiste à traduire dans le langage spécifique de l'automate, les équations de fonctionnement du système à automatiser.

La norme industrielle IEC 61131-3 définit cinq langages de programmation des automates programmables, celles-ci consistent en deux langages textuels, (IL: liste d'instructions) et (ST: texte structuré), et trois langages graphiques, (LD : schéma à contacts), (FBD: schéma fonctionnel) et (SFC: Sequential Function Chart).

2.4.1 Langages de programmation textuels

A. Liste d'instructions IL (Instruction List)

Les listes d'instructions est très proche du langage informatique dit assembleur. Ce langage prend en charge la programmation basée sur un accumulateur. Les opérateurs IEC 61131-3 sont pris en charge, ainsi que plusieurs entrées/plusieurs sorties, les négations, les commentaires, la définition/redéfinition des sorties et les sauts non conditionnels/conditionnels.

Exemple: Démarrer un moteur si le bouton «START» est pressé et si on n'est pas dans une condition d'alarme.

```
VAR
        start :      BOOL  AT \%IX0.1 ;
        alarm :      BOOL  AT \%MX0.1 ;
        power_on : BOOL  AT \%OX0.1 ;
END_VAR
        LD          start
        ANDN        alarm
        ST          power_on
```

B. Texte structuré ST (Structured Text)

Ce langage textuel de haut niveau est un langage évolué. Il permet la programmation de tout type d'algorithme plus ou moins complexe.

```
if (start and (not alarm))
        then power_on
end if
```

2.4.2 Langages de programmation graphiques

A. Boîtes fonctionnelles : FBD (Function Block Diagram)

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.

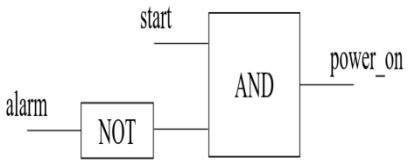

Figure 09. Exemple de programme en FBD

B. Langage Ladder : LD (Ladder Diagram)

Le langage Ladder est une succession de "réseaux de contacts" véhiculant des informations logiques depuis les entrées vers les sorties. C'est une simple traduction des circuits de commande électriques.

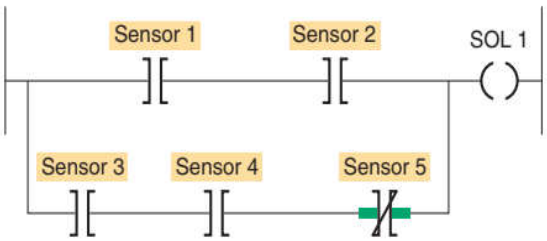

Figure 10. Exemple de programme Ladder

C. Sequential function chart 'SFC'

Langage de programmation inspiré du standard français GRAFCET (GRAphe Fonctionnel de Commande Etape/Transition). C'est un Langage graphique de haut niveau qui a une structure ressemblant à la structure d'un réseau de Petri.

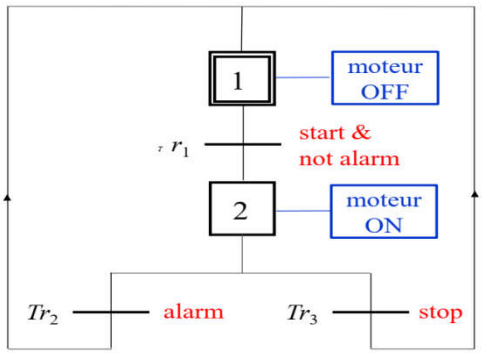

Figure 11. Exemple de programme SFC