1 Structure d'un Système Automatisé de Production (SAP)

Une installation industrielle automatisée comporte :

- une **partie opérative** (chaîne de fabrication, de conditionnement, etc…)
- une **partie commande** qui fournit à la partie opérative des commandes lui permettant d'effectuer le travail de production pour laquelle elle a été conçue.
- Une partie **Interface Homme-Machine (IHM)** (angl. *HMI* pour *Human-Machine Interface*) pour assurer la mise en œuvre, la conduite de l'installation et sa maintenance.

La partie commande d'un automatisme peut être de type câblée ou programmée. Historiquement, les automatismes étaient réalisés à l'aide d'une logique câblée (pneumatique ou électrique). En cas d'erreurs, il fallait défaire le câblage et rétablir les connexions. Chaque extension ou modification de fonctions exigeait de modifier l'appareillage, d'intervenir sur le câblage et d'effectuer des travaux d'installation.

De nos jours, dès qu'une partie commande agit sur plus d'un actionneur (moteur, vérin,…) elle est de type programmée. Les fonctions d'automatismes sont programmées dans un **A.P.I.** (Automate Programmable Industriel) (angl. *PLC* pour *Programmable Logic Controller*). Le câblage en logique programmée est nettement réduit et il est facile de réaliser des extensions et de supprimer les erreurs.

2 Présentation des Automates Programmables Industriels

L'API traite le programme qui spécifie l'essentiel de la tâche d'automatisme. Son fonctionnement est comparable à celui d'un ordinateur dépourvu d'écran et de clavier : il n'est capable que d'exécuter des programmes. Il est programmé à partir d'un PC de développement (Engineering Station). Il n'est pas nécessaire que le poste de développement soit relié en permanence avec l'API pour que celui-ci assure la commande de la production.

Par rapport à un ordinateur, les API sont conçus pour fonctionner dans des ambiances industrielles difficiles.

2.1 Automates compacts/modulaires :

En plus des API compacts qui regroupent les principaux composants dans un espace réduit, il existe des automates modulaires qui peuvent être configurés individuellement selon les besoins. Ils se distinguent par leur flexibilité, leur puissance et leur maintenabilité.

Les différents constituants des API modulaires sont encliquetés sur un profilé support. L'unité centrale (CPU) est reliée aux autres modules par un bus interne.

a. L'alimentation :

Pour pouvoir fonctionner, l'API exige une tension d'alimentation de +24V CC. Le module d'alimentation convertit la tension alternative du réseau 230V en 24V continu. Ce module fournit l'énergie non seulement pour l'automate mais aussi pour les capteurs et actionneurs raccordés.

b. L'unité centrale :

L'unité centrale (*CPU* pour *Central Processing Unit*) est l'élément central d'un automate programmable. C'est sur elle qu'est mémorisé et traité le programme.

Un commutateur de mode permet de choisir manuellement l'état de fonctionnement de l'automate (marche(ou *RUN*) / arrêt (ou *STOP*)).

Les coupleurs sont utilisés sur les automates configurés sur plusieurs châssis. Ils assurent de façon autonome la communication entre les différents châssis.

c. Les modules de signaux :

La tension de fonctionnement interne des API est de 5V ou 3,3V. Il importe donc de convertir à ce niveau interne les signaux venant de l'extérieur et partant vers l'extérieur. Cette mission est assurée par les modules de signaux. Les modules de signaux forment l'interface entre la CPU et le process. On distingue 2 grandes familles de modules de signaux : les **modules TOR** (tout ou rien) et les **modules analogiques**.

Les modules analogiques assurent la conversion des signaux analogiques en signaux numériques de manière à permettre leur traitement par la CPU. C'est pourquoi ces modules se rencontrent surtout pour les tâches de régulation (de température, de pression…).

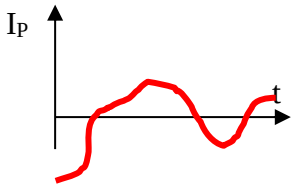
d. Les modules pour fonctions particulières :

Les modules de fonction élargissent la polyvalence de l'automate. Il existe des modules de fonction pour le comptage rapide, la régulation, le positionnement d'axe (par exemple commande de moteur pas à pas), etc…

Les automates sont mis en réseau pour leur permettre de communiquer. Les petites quantités d'informations peuvent être échangées directement entre CPU (pas de module de communication). Par contre, pour les gros volumes de données, on utilise des modules de communication qui possède un processeur de communication. Ces processeurs soulagent la CPU des tâches de communication et autorisent l'établissement de liaisons supplémentaires (par exemple avec des interfaces homme-machine).

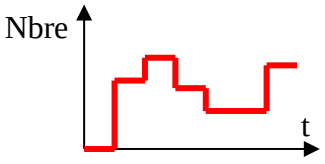
2.2 Signaux gérés par l'API

a. Signal analogique (*Analog Signal*)



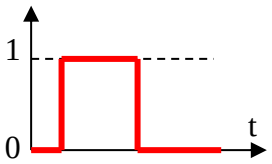
Un signal est dit **analogique** si l'amplitude de la grandeur physique le représentant peut prendre une infinité de valeurs.
Exemple le courant sortant d'un transmetteur de pression.
4 mA 5,123567 mA 10,214 mA 12 mA

b. Signal numérique (*Digital Signal*)



Un signal est dit **numérique** si l'amplitude de la grandeur physique le représentant ne peut prendre qu'un nombre fini de valeurs.
Exemple signal affiché sur un manomètre numérique 2000 points.
0112 1456 0006 1300

c. Signal logique Tout Ou Rien (T.O.R.) (*Digital Signal*)



Un cas particulier du signal numérique est le signal **logique ou Tout Ou Rien (T.O.R.)**. Ce signal ne peut prendre que deux valeurs : 0 ou 1, Vrai ou Faux, Ouvert ou Fermé.
Exemple le signal sortant d'un pressostat.
True False

3 Programmation des API

La norme CEI 61131-3 classe les langages de programmation des API en 3 catégories :

Langages littéraux	Langages graphiques	Structure de programme
Langage IL (liste d'instructions) **Langage ST** (littéral structuré)	**Langage LD** (à contacts) **Langage FBD** (à blocs fonctionnels)	**Diagramme SFC** (suite de séquences)

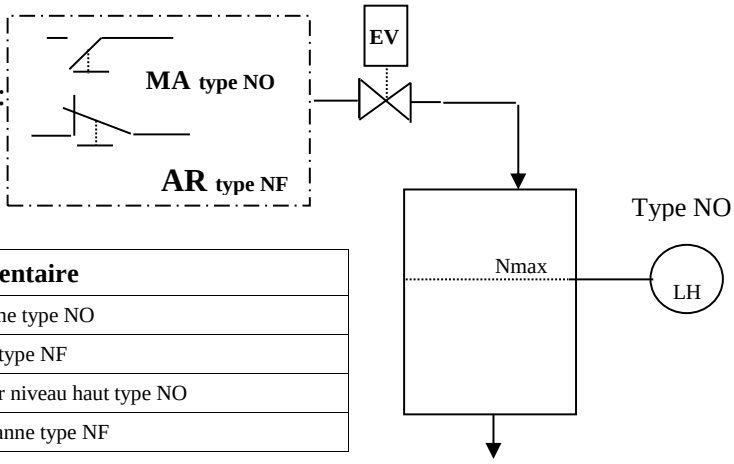
3.1 LANGAGES LITTERAUX

a. Langage IL (Instruction List ou langage à liste d'instructions)

Comme son nom l'indique, le programme est constitué d'une suite d'instructions respectant le format suivant :

Etiquette (non obligatoire)	Opérateur	Opérande(s)	Commentaire (non obligatoire)

Exemple : Soit à commander une électrovanne EV du schéma TI suivant :



Affectation des Entrées/Sorties :

Mnémonique	Adresse API	Commentaire
MA	I 0.0	BP marche type NO
AR	I 0.1	BP arrêt type NF
LH	I 0.2	Détecteur niveau haut type NO
EV	Q 4.0	Electrovanne type NF

Exemple de programmation en IL :

```
A(
O     "MA"          I0.0          -- BP marche type NO
O     "EV"          Q4.0          -- Electrovanne type NF
)
A     "AR"          I0.1          -- BP arrêt type NF
AN    "LH"          I0.2          -- Niveau haut type NO
=     "EV"          Q4.0          -- Electrovanne type NF
```

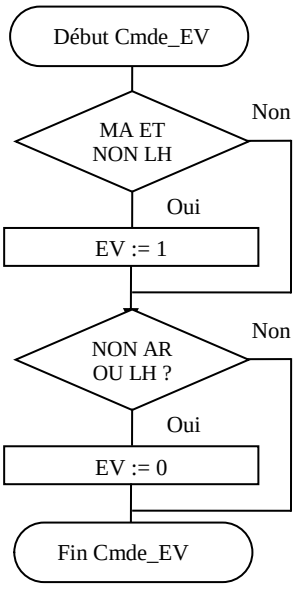
NB : Ce langage est proche du langage de programmation d'un microprocesseur : l'assembleur

b. Langage ST (Structured Text) ou langage littéral structuré)

Ce langage est composé d'expressions littérales constituées d'opérateurs et d'opérandes et d'énoncés.

Ce langage est proche d'un langage informatique comme le BASIC.

Un programme écrit en ST peut facilement être décrit à l'aide d'un organigramme, comme ci-contre.



Exemple de programmation en ST :

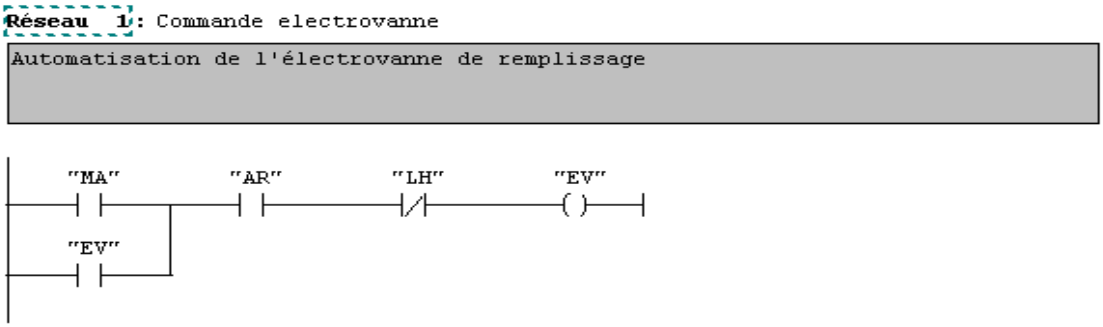
```
(* Commande de l'électrovanne EV*)
IF(Ma AND NOT Lh)THEN
     SET Ev;
END_IF;
IF(Lh OR NOT Ar)THEN
     RESET Ev;
END_IF;
```

3.2 LANGAGES GRAPHIQUES

a. Langage LD (Ladder Diagram) ou langage à contacts)

Ce langage est constitué de réseaux de contacts et de bobines entre deux barres d'alimentation. Ce langage est proche des schémas électriques.

En reprenant l'exemple de la page précédente et sur API Siemens en langage à contacts :



Informations mnémonique:

I0.0	MA	BP marche type NO
Q4.0	EV	Electrovanne type NF
I0.1	AR	BP arrêt type NF
I0.2	LH	Niveau haut type NO

b. Langage FBD (Function Block Diagram) ou langage en blocs fonctionnels

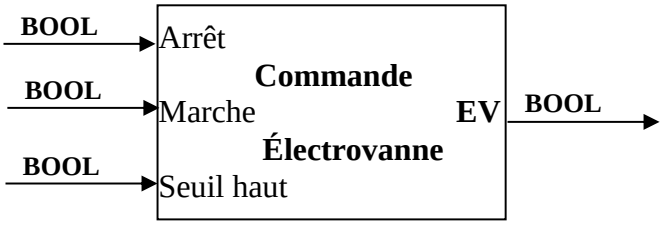
Ce langage se compose de réseaux de fonctions **préprogrammées ou non**, représentées par des rectangles. Ces blocs fonctionnels sont connectés entre eux par des lignes, le flux des signaux se faisant de la sortie (à droite) d'une fonction vers l'entrée à gauche de la fonction raccordée.

Exemples de blocs fonctionnels standards (fourni par le constructeur de logiciel):

- Bloc fonctionnel compteurs
- Bloc fonctionnel temporisateurs
- Bloc fonctionnel PID…

Exemple de bloc fonctionnel utilisateur (développés par le programmateur et réutilisables)

- **Exemple 1** de la page précédente :



3.3 STRUCTURATION D'UN PROGRAMME:

Le diagramme SFC (Sequentiel **F**unction **C**hart**)** ou langage fonctionnel de séquences

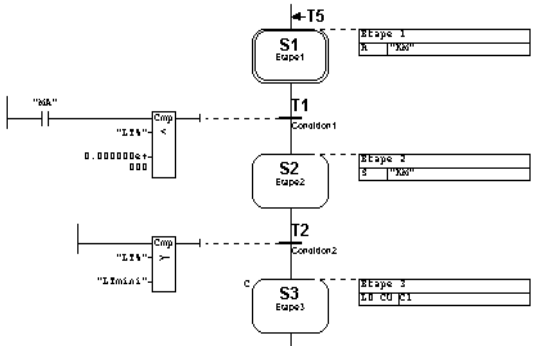
Le diagramme fonctionnel de séquence SFC à ne pas confondre avec la description du comportement d'un système (connu sous le nom de GRAFCET en France, voir norme **CEI 60848**).

Ce langage est destiné à la description de fonctions de commande séquentielles. Le programme correspondant est constitué d'un ensemble d'**étapes** et de **transitions** reliés entre elles par des **liaisons dirigées**.

Chaque étape est associée à un ensemble d'**actions**.

Chaque transition est associée à une **condition de transition**.

Exemple de programmation SFC dans Step7 (Siemens) :



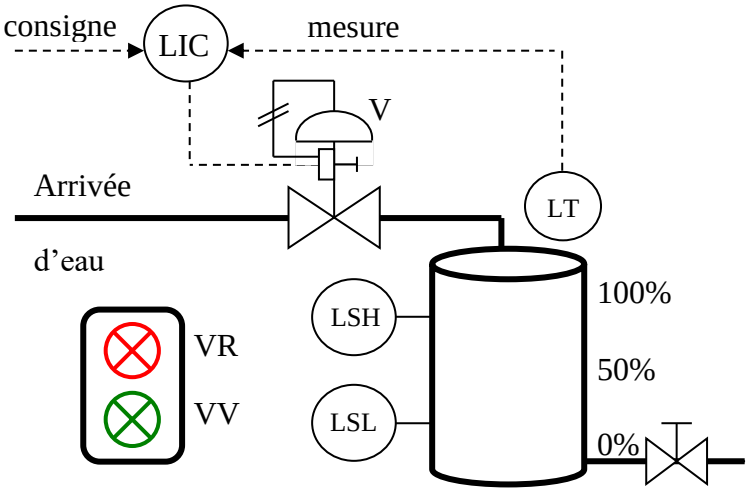
4 Application : régulation de niveau

On souhaite réaliser :

- Une régulation de niveau.
- Un affichage de sécurité sur les niveaux.

- Entourer sur le schéma les éléments intervenant dans la régulation analogique de niveau.

- Entourer sur le schéma les éléments intervenant dans l'affichage de sécurité : **c'est l'automatisme logique T.O.R.**



Le détecteur de niveau haut LSH est de technologie NF et le détecteur de niveau bas LSL est NO.

- Que veut dire NO (*NO*) et NF (*NC*) ? Normalement ouvert (*open*) et normalement fermé (closed)
- Représenter le schéma de principe de câblage de ces entrées automate (24V).

	LSL (NO)			LSH (NF)		
Niveau	Etat physique capteur	Tension entrée API	Etat logique variable API	Etat physique capteur	Tension entrée API	Etat logique variable API
0%	Repos	0 Volt	***Niv_LSL*** = 0	Repos	24 Volt	***Niv_LSH*** = **1**
50%	Actionné	24 Volt	***Niv_LSL*** = 1	Repos	24 Volt	***Niv_LSH*** = **1**
100%	Actionné	24 Volt	***Niv_LSL*** = 1	Actionné	0 Volt	***Niv_LSH*** = **0**

- Réaliser un programme en LADDER (schéma à contacts) pour faire allumer le voyant rouge VR si le niveau dépasse le capteur LSH.
- Réaliser un programme en LADDER (schéma à contacts) pour faire allumer le voyant vert VV si le niveau est en dessous du capteur LSL.