Chapitre I : Généralités sur les API

L'étude des API fait partie de la discipline « Informatique Industrielle » qui utilise les théories de l'automatique et les moyens de l'informatique dans le but de résoudre des problèmes de nature industriels.

I.1 Définition d'un API

Un Automate Programmable Industriel (API) est une machine électronique programmable par un personnel non informaticien et destiné à piloter en ambiance industrielle et en temps réel des procédés ou parties opératives. Un API est adaptable à un maximum d'application, d'un point de vue traitement, composants, langage.

Le développement de l'industrie à entraîner une augmentation constante des fonctions électroniques présentes dans un automatisme c'est pour ça que l'API s'est substitué aux armoires à relais en raison de sa souplesse dans la mise en œuvre, mais aussi parce que dans les coûts de câblage et de maintenance devenaient trop élevés.

Les API font partie d'un concept appelé « logique programmable » qui a remplacée la logique câblée et il rendu la tâche d'automatisation simple.

Les API sont apparus aux U.S.A vers 1969, où ils répondaient aux désires des industries de l'automobile afin de développer des chaînes de fabrications automatisées qui pourraient suivre l'évolution des techniques et des modèles fabriqués.

Les premiers produits des API sont de marque MODICON et ALLEN-BRADLEY fabriqués aux USA en 1969 et en France en 1971, par EDF, MERLIN-GERIN et ALSPA.

I.2 Caractéristique interne d'un API

Des API en boîtier étanche sont utilisés pour les ambiances difficiles (température, poussière, risque de projection ...) supportant ainsi une large gamme de température, humidité ... .

L'environnement industriel se présente sous trois formes :

• Environnement physique et mécanique (poussières, température, humidité, vibrations);
• Pollution chimique ;
• Perturbation électrique (parasites électromagnétiques).

Les caractéristiques principales d'un API sont :

•Coffret, rack, baie ou cartes ;
•Compact ou modulaire ;
•Tension d'alimentation ;
•Taille mémoire ;
•Temps de scrutation des entrées;
•Sauvegarde (EPROM, EEPROM, pile, RAM, …) ;
•Nombre et le type des entrées /sorties ;
•Modules complémentaires (analogique, communication,..) ;
•Langage de programmation.

I.3 Architecture Interne d'un API

La structure interne d'un API est assez voisine de celle d'un système informatique simple, l'unité centrale est le regroupement du processeur et de la mémoire centrale. Elle commande l'interprétation et l'exécution des instructions programme. Les instructions sont effectuées les unes après les autres, séquencées par une horloge.

D'une manière générale, un API est structuré autour de plusieurs éléments de base qui sont l'unité centrale de traitement (UCT), l'unité d'alimentation, les interfaces d'entrées-sorties, l'interface de communication et le périphérique de programmation.

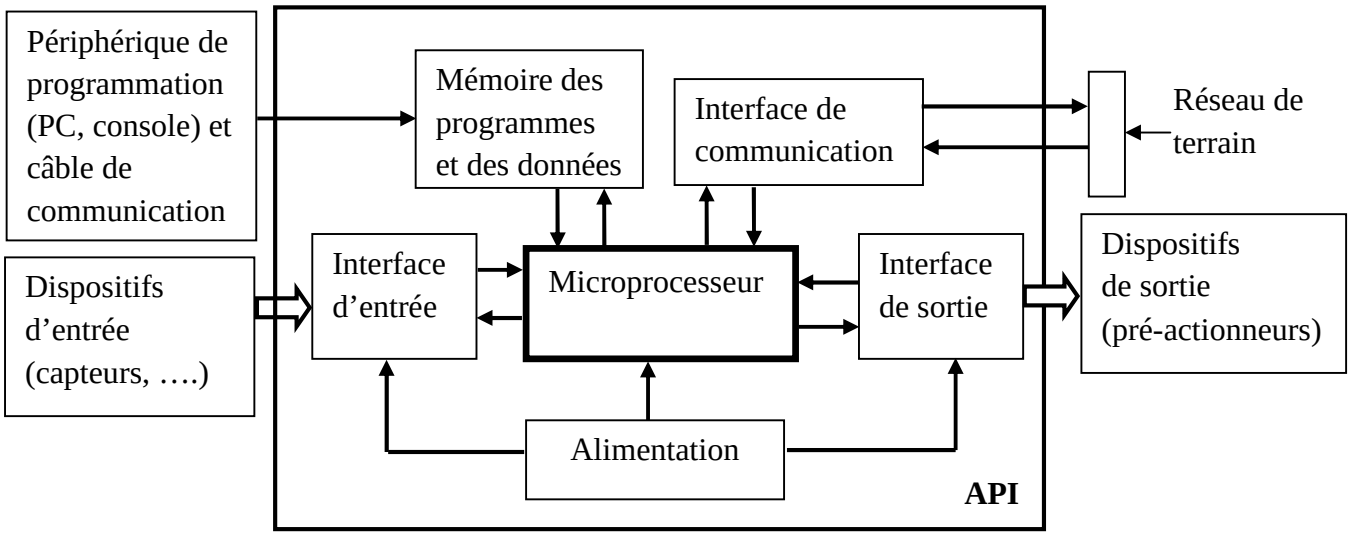


Figure I.1 : Structure interne d'un API.

-Microprocesseur (UCT, CPU) : C'est le cœur de l'API. Il interprète les signaux d'entrée et effectue les actions de commande conformément au programme stocké en mémoire, en communiquant aux sorties les décisions sous forme de signaux d'actions. L'UCT est composé d'une unité arithmétique et logique, d'un ensemble de registre et de l'unité de commande.

- **Unité d'alimentation :** Elle est indispensable puisqu'elle convertit une tension alternative en une basse tension continue (en général de 5 V) nécessaire au microprocesseur et aux modules d'entrées-sorties (12V, 24V, 48V, 110V, 240V) continu ou alternative.

-Périphérique de programmation : Il est composé d'une console ou d'un PC avec dans les deux cas d'un câble spécial et il est indispensable pour le transfert du programme dans API. La programmation des premier API utilise exclusivement une console, les API récents sont programmés en plus d'une console avec des PC. Une foie le programme est développé alors il est transféré dans la mémoire de l'API avec un câble spécial basé sur la transmission série.

-Mémoire : Elle contient le programme qui définit les actions de commande générées par l'UCT. Elle contient aussi les données délivrées par les entrées (mémoire image des entrées)

en vue de leur traitement, ainsi que celle délivrées aux sorties (mémoire image des sorties). Elle est partagée en plusieurs zones : mémoire image des entrées ; mémoire image des sorties ; zone des temporisations, zone des compteurs ; mémentos ; accumulateurs ; … . En général, chaque zone est identifiée par un identificateur de zone (lettre latine en majuscule).

Il existe dans les API deux types de mémoires qui remplissent des fonctions différentes :
- La mémoire Langage où est stocké le langage de programmation. Elle est en général figée, c'est à dire en lecture seulement (ROM : mémoire morte, l'EPROM ou EEPROM). Elle sauvegarde son contenu même après coupure de l'alimentation.
- La mémoire Travail utilisable en lecture-écriture pendant le fonctionnement c'est la RAM (mémoire vive). Elle s'efface automatiquement à l'arrêt de l'automate (nécessite une batterie de sauvegarde). Elle est partagée en deux zones :
-Zone mémoire des données : Elle contient les variables d'entrées, les variables de sorties, les variables internes (temporisations, compteurs, …).
- Zone mémoire programme : Elle contient le programme développé par l'utilisateur qui traite les données de la zone mémoire des données.

Le transfert de l'EPROM ou EEPROM vers la mémoire RAM de l'automate, s'effectue à chaque reprise secteur et si le contenu de celle-ci est différent.
- **Les interfaces entrées-sorties (E/S) :** Elles permettent au microprocesseur de recevoir et d'envoyer des informations aux dispositifs extérieurs. Les entrées peuvent être des interrupteurs, ou des capteurs, comme des cellules photoélectriques dans le cas de mécanisme de comptage, des sondes de température, des débitmètres, etc. Les sorties sont en général des pré-actionneurs associés aux bobines de moteur, à des électrovannes, … . Chaque point d'E/S dispose d'une adresse unique, que l'UCT peut utiliser. L'isolation (protection) électrique avec le monde extérieur est généralement réalisée par des photo-coupleurs (opto-coupleurs).

I.4 Fonctionnement d'un API

Un API est en permanence en activité dès sa mise sous tension. en mode « STOP », l 'API reste actif, seule l'exécution du programme est suspendue. Les fonctions réseau restent actives (échange avec les autres automates, surveillances des E/S ...). Pendant l'exécution du programme la mémoire image des entrées restent figée ainsi que les sorties physiques

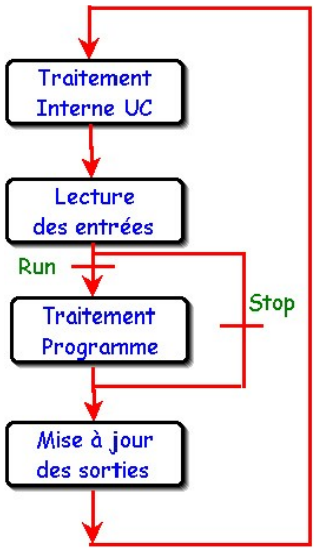


Figure I. 2 : Cycle d'un API.

Le cycle d'un API est composé de quatre phases :

•**Phase 1 : Gestion du système :** Autocontrôle de l'automate ;

•**Phase 2** : **Acquisition des entrées :** Prise en compte des informations du module d'entrées et écriture de leur valeur dans la mémoire image des entrées (zone des données dans la RAM).

•**Phase 3** : **Traitement des données :** Lecture du programme (située dans la RAM zone programme) par l'unité de traitement, lecture des variables (entrées, mémentos, variables, temporisations, …) dans la zone des données de la RAM, traitement et écriture des variables dans la mémoire image des sorties dans la même zone que les variables.

•**Phase 4** : **Emissions des ordres :** Lecture des variables de sorties dans la mémoire image des sorties zone des données de la RAM par les sorties physiques de l'API.

I.5 Les entrées/sorties (E/S) d'un API

Les entrées ainsi que les sorties peuvent êtres de trois types : discrètes tout ou rien (TOR), numériques (TOR) ou analogiques. Les E/S TOR peuvent êtres de type DC (à transistors) si l'alimentation de l'API ainsi que celles des capteurs, des pré-actionneurs est de type DC faible (12V ou 24V) et ils sont à relais si l'alimentation de l'API ainsi que celles des capteurs, des pré-actionneurs est de type AC ou DC forte.

I.5.1 Les E/S discrètes (TOR)

Ce sont des E/S à deux états, présence ou absence d'énergie (tension). Elles sont représentées par un seul bit. Les entrées de ce type sont bouton poussoir, interrupteur, fin de cours, capteur de présence ou d'absence d'objet, … et les sorties de cette nature sont les relais électrique, distributeur, … .

I.5.2 Les E/S numériques (TOR)

Elles peuvent êtres vues comme une suite d'E/S (signaux) discrètes (TOR) qui produisent des E/S (signaux) numériques. Elles sont représentées par au moins deux bits. L'encodeur (capteur d'angle de déplacement d'un moteur) est une entrée numérique et la commande d'un moteur à au moins deux vitesses avec l'arrêt est une sortie numérique.

I.5.3 Les entrées analogiques

Les entrées analogiques transmis à l'API sont l'équivalent numérique des signaux continus des grandeurs surveillées par des capteurs. Ces variables physiques peuvent êtres une température, une pression, un niveau, une tension, un courant, une vitesse, … . Elles sont équipées de convertisseur analogique-numérique (CAN) qui permet de convertir les signaux analogiques en signaux numériques sur plusieurs bits que la CPU de l'API peut traiter.

- Fonctionnement d'un CAN

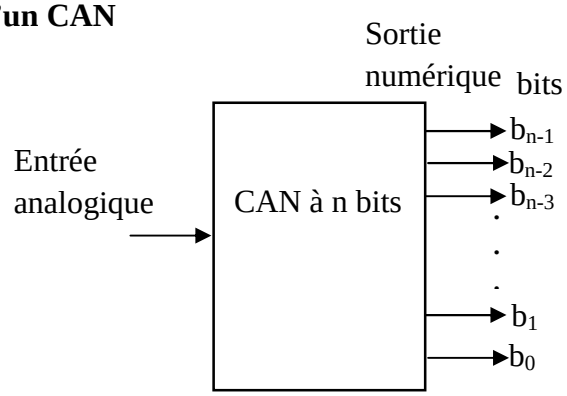


Figure I. 3 : Convertisseur analogique-numérique à n bits.

Le signal numérique de sortie de n bits correspond au niveau du signal d'entrée analogique. On a 2^n valeurs possibles du signal.
La *résolution* du CAN est $V_{max}/(2^n-1)$ avec V_{max} la valeur maximal du signal (tension) analogique et elle désigne le plus petit changement du signal analogique qui donne lieu à une modification d'un bit dans la sortie numérique. Les valeurs du signal analogique qui produisent chaque sortie numérique sont appelées *niveaux de quantification*.

I.5.3 Les sorties analogiques

La CPU d'API délivre des signaux TOR ou numérique. Si l'entrée du pré-actionneur nécessite (fonctionne) avec un signal analogique (continu) alors il est impératif de convertir les signaux numériques en signaux analogiques. Cette tache est réalisée par des sorties analogiques équipées de convertisseurs numérique-analogique (CNA).

- Fonctionnement d'un CNA

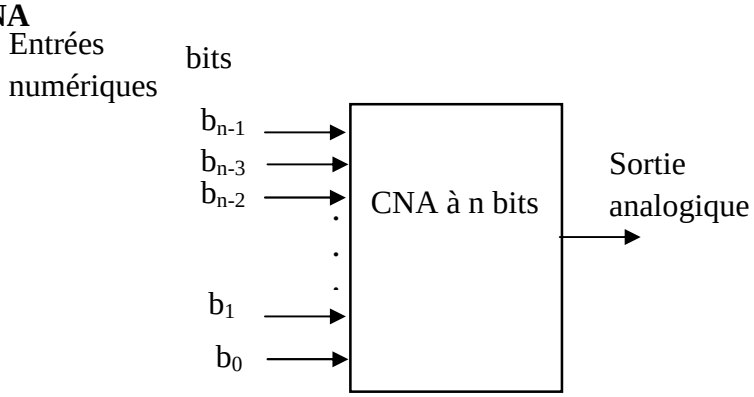


Figure I.4 : Convertisseur numérique-analogique à n bits.

Le CNA est caractérisé par son nombre n de bits. Le changement d'un bit entre deux poids consécutifs produit une impulsion de sortie dont l'amplitude correspond à la variation minimale de la sortie analogique. Cette variation représente la précision ou la résolution de CNA calculée par la relation $V_{max}/(2^n-1)$ avec V_{max} la valeur maximal du signal (tension) analogique que peut délivré le CNA.
Remarque : La résolution, donc la *précision* du CAN et du CNA augmente avec l'augmentation de leurs nombres de bits.

I.6 Les interfaces d'entrées/sorties

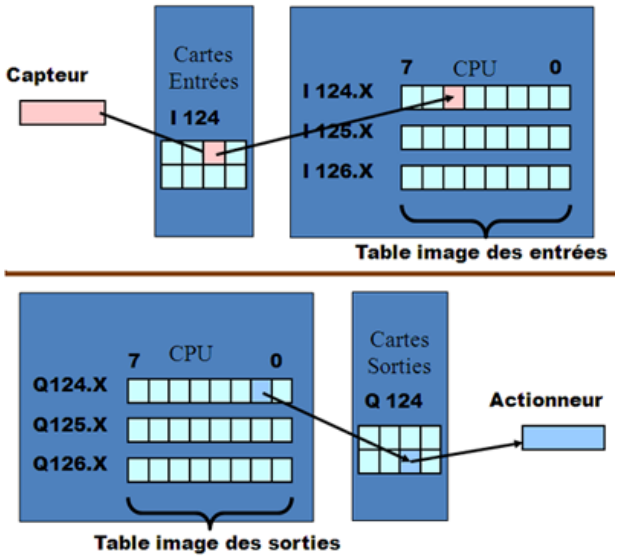


Figure I.5 : Les interfaces d'entrées/sorties.

L'interface d'entrée comporte des adresses d'entrée. Chaque capteur est relié à une de ces adresses. L'interface de sortie comporte de la même façon des adresses de sortie. Chaque pré-actionneur est relié à une de ces adresses. Le nombre de ces entrées est sorties varie suivant le type d'API. Les cartes d'E/S ont une modularité de 8, 16 ou 32 voies. Les tensions disponibles sont normalisées (24, 48, 110 ou 240V continu ou alternatif ...).

I.7 Les cartes d'entrées

Elles sont destinées à recevoir l'information en provenance des capteurs et adapter le signal en le mettant en forme, en éliminant les parasites et en isolant électriquement l'unité de commande (CPU) de la partie opérative (capteur). La tension continue de 24V est celle appliquée à l'entrée des capteurs (à gauche sur le circuit) et la tension continue de 5V est celle appliquée à l'entrée de la CPU.

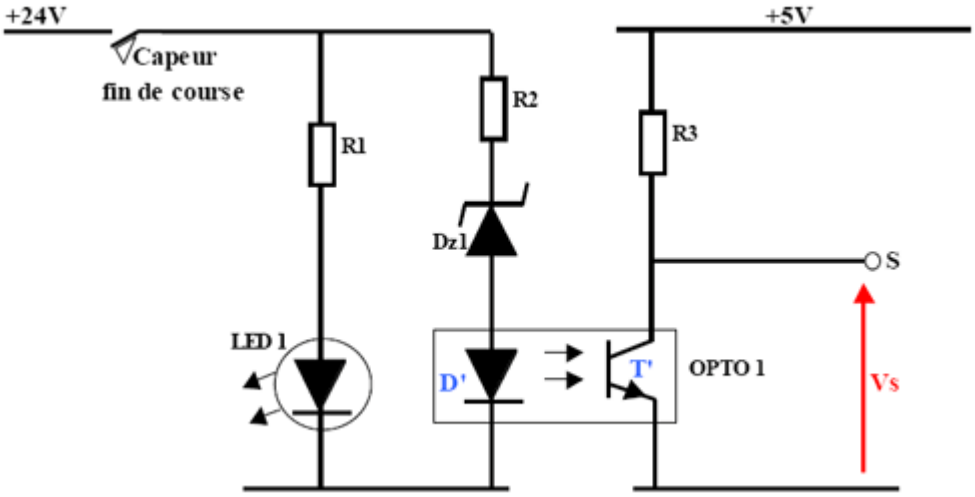


Figure I.6 : Carte d'entrée d'un API.

Remarque

Les appareils (capteurs, …etc) à courant continu peuvent être connectés à l'unité d'entrée (UE) de l'API de deux manières différentes soit en fourniture ou en absorption de courant de la part de l'UE de l'API.

Dans le cas de la fourniture de courant (UE de l'API est une source de courant), le capteur est branché à la borne négative de l'UE de l'API.

Dans le cas de l'absorption de courant (le capteur fournit du courant et l'UE de l'API reçoit ou absorbe du courant), le capteur est branché à la borne positive de l'UE de l'API.

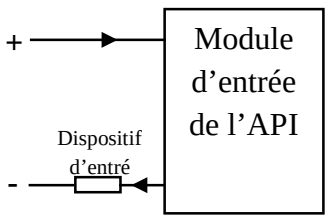


Figure I.7 : L'UE de l'API à fourniture de courant.

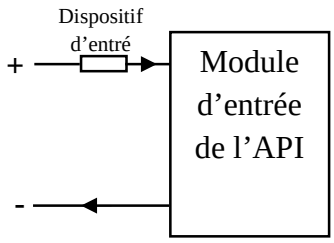


Figure I.8 : L'UE de l'API à absorption de courant.

I.8 Cartes de sorties

Elles sont destinées à commander les pré-actionneurs et éléments des signalisations du système et adapter les niveaux de tensions de l'unité de commande (sortie) à celle de la partie opérative du système en garantissant une isolation galvanique entre ces dernières. La tension continue de 5V est celle délivrée par la CPU, c'est-à-dire à sa sortie et la tension continue de 24V est celle appliquée au pré-actionneur (la tension de sortie peut être alternative).

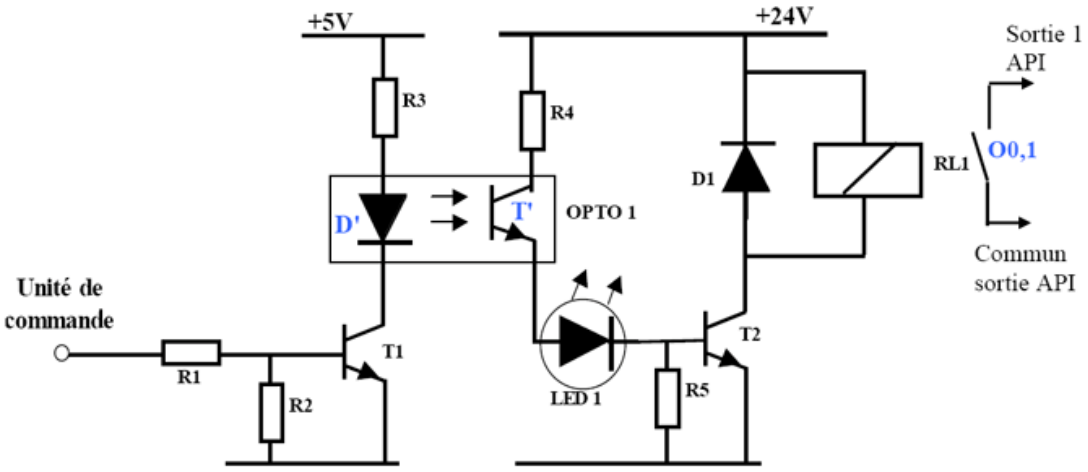


Figure I.9 : Carte de sorties d'un API.

Remarque

Les appareils (pré-actionneurs, …etc) à courant continu peuvent être connectés à l'unité de sortie (US) de l'API de deux manières différentes soit en fourniture ou en absorption de courant de la part de l'US de l'API.

Dans le cas de la fourniture de courant (US de l'API est une source de courant), le pré-actionneur est branché à la borne négative de l'US de l'API.

Dans le cas de l'absorption de courant (le pré-actionneur fournit du courant et l'US de l'API reçoit ou absorbe du courant), le pré-actionneur est branché à la borne positive de l'UE de l'API.

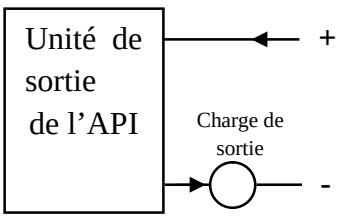


Figure I.10 : L'US de l'API à fourniture de courant.

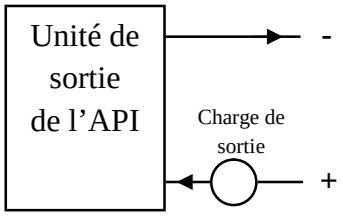


Figure I.11 : L'US de l'API à absorption de courant.

I.9 Exemple de cartes d'E/S d'un API

- ***Cartes de comptage rapide*** : Elles permettent d'acquérir des informations de fréquences élevées incompatibles avec le temps de traitement de l'automate (signal issu d'un codeur de position).

- ***Cartes de commande d'axe*** : Elles permettent d'assurer le positionnement avec précision d'élément mécanique selon un ou plusieurs axes. La carte permet par exemple de piloter un servomoteur et de recevoir les informations de positionnement par un codeur. L'asservissement de position pouvant être réalisé en boucle fermée.
- ***Cartes d'entrées/sorties analogiques*** : Elles permettent de réaliser l'acquisition d'un signal analogique et sa conversion numérique (CAN) indispensable pour assurer un traitement par le microprocesseur. La fonction inverse (sortie analogique) est également réalisée. Les grandeurs analogiques sont normalisées : 0-10V ou 4-20mA.
- *Cartes de régulation PID ;* - *Cartes de pesage ;*
- *Cartes de communication* (RS485, Ethernet ...) ; - *Cartes d'entrées/sorties déportées.*

I.10 Branchement des Entrées TOR

Le principe de raccordement consiste à envoyer un signal électrique vers l'entrée choisie sur l'automate dés que l'information est présente.
L'alimentation électrique peut être fournie par l'automate (en général 24V continu) ou par une source extérieure. Un API peut être à **logique positive** ou **négative.**
- **Logique positive :** Le commun interne des entrées est relié au 0V

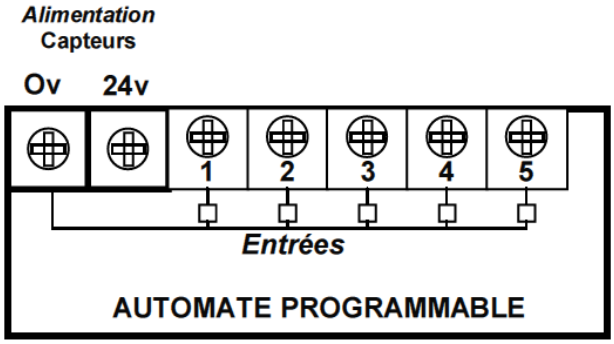


Figure I.12 : Branchement des entrées TOR d'un API à logique positive.

- **Logique négative :** Le commun interne des entrées est relié au 24V

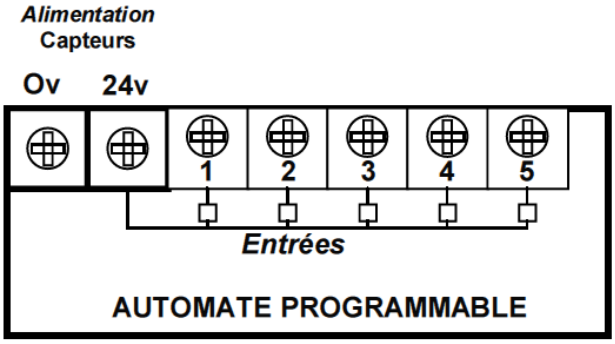


Figure I.13 : Branchement des entrées TOR d'un API à logique négative.

I.11 Branchement des sorties

Le principe de raccordement consiste à envoyer un signal électrique vers le pré-actionneur connecté à la sortie choisie de l'automate dés que l'ordre est émis.
L'alimentation électrique est fournie par une source extérieure à l'API.

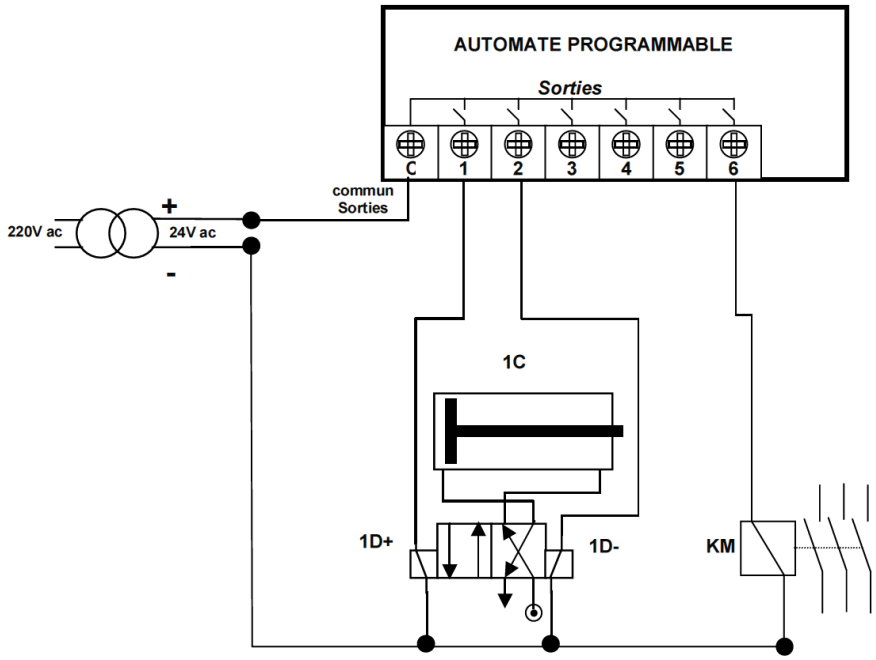


Figure I.14 : Branchement des sorties TOR d'un API.
Les sorties 1 et 2 sont reliées au distributeur bistable et la sortie 6 à un contacteur.

I.12 Les langages de programmation d'un API

Aux USA, les problèmes de logique (en industrie) ont été résolus par des armoires à relais, puis par des mini-ordinateurs et en fin par des API. En Europe, on a utilisé les armoires à relais, puis la logique statique (portes logique) et en fin la logique programmable (API). Chaque marque API a un logiciel spécifique, malgré les ressemblances qui existent.

Les API peuvent être programmés par cinq langages qui sont classés en deux types :

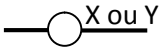
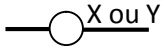
I.12.1 Les langages graphiques

Ils ont été les premiers utilisés sur les API et ils sont de trois types :

A) Langage à contact, schéma à relais, diagramme en échelle *(Ladder diagram LAD)*

Il a été créé par des américains à partir des schémas (électriques) à relais. L'intérêt principal est le passage presque direct de la logique câblée à celle programmée. Il est utilisé par la plupart des marques d'API. Les constituants de base du schéma à relais sont aux nombre de cinq et ils sont récapitulés dans le tableau ci-dessous, actuellement d'autres constituants sont aussi utilisés:

Types de constituants	Convention américaine	Convention européenne	Signification
Constituants Logiques	a	a	Relais normalement ouvert.
	$\bar{a}$	$\bar{a}$	Relais normalement fermé.
			Ouverture de branche parallèle.
			Fermeture de branche parallèle.

Symbole d'affectation			Affectation du résultat à une variable interne (X) ou de sortie (Y).

B) Les graphes de fonction séquentielle (*SFC Sequentiel Function Charts*), GRAFCET

Il est utilisé pour décrire les opérations séquentielles. Le procédé est représenté comme une suite connue d'étapes (états stables), reliées entre elles par des transitions. Une condition booléenne est attachée à chaque transition. Les actions dans les étapes sont décrites avec les langages ST, IL, LD ou FBD.

C) Les diagrammes de schémas fonctionnels (FBD *Function Blocs Diagram*), Logigramme

Il a été développé en Europe. Il permet de programmer sur un API les équations (logiques) d'un automatisme sous forme d'un schéma logique (portes logiques et bascules).

I.12.2 Les langages textuels

Ils sont de deux types :

A) Les listes des instructions *(IL, Instruction List)*

C'est un langage textuel de bas niveau de type assembleur. Il est particulièrement adapté aux applications de petite taille. Les instructions opèrent toujours sur un résultat courant (ou registre IL). L'opérateur indique le type d'opération à effectuer entre le résultat courant et l'opérande. Le résultat de l'opération est stocké à son tour dans le résultat courant. Un programme IL est une liste d'instructions. Chaque instruction doit commencer par une nouvelle ligne, et doit contenir un opérateur, complété éventuellement par des modificateurs et, si c'est nécessaire pour l'opération, un ou plusieurs opérandes, séparés par des virgules (','). Une étiquette suivie de deux points (':') peut précéder l'instruction. Si un commentaire est attaché à l'instruction, il doit être le dernier élément de la ligne. Des lignes vides peuvent être insérées entre des instructions. Un commentaire peut être posé sur une ligne sans instruction.

Le tableau suivant résume les opérations courantes du langage IL :

Opérateur	**Modificateurs**	**Opérande**	**Description**
LD	N	variable, constante	charge l'opérande
ST		variable BOOL	stocke le résultat courant
S		variable BOOL	forçage à TRUE (à un)
R		variable BOOL	forçage à FALSE (à zéro)
AND	N (	variable BOOL	ET logique
&	N (	variable BOOL	ET logique
OR	N (	variable BOOL	OU logique
XOR	N (	variable BOOL	OU exclusif
ADD	(	variable, constante	Addition
SUB	(	variable, constante	Soustraction
MUL	(	variable, constante	Multiplication
DIV	(	variable, constante	division
GT	(	variable, constante	Test >
GE	(	variable, constante	Test >=
EQ	(	variable, constante	Test >=
LE	(	variable, constante	Test <=
LT	(	variable, constante	Test <
NE	(	variable, constante	Test < >

CAL JMP RET	C N C N C N	instance de bloc fonctionnel étiquette	Appel d'un bloc fonctionnel Saut à une étiquette Retour de fonction
)			Exécute l'instruction différée

B) Le texte structuré (*ST, Structured Text*)

Le langage ST est un langage textuel de haut niveau dédié aux applications d'automatisation. Ce langage est principalement utilisé pour décrire les procédures complexes, difficilement modélisables avec les langages graphiques. C'est le langage par défaut pour la programmation des actions dans les étapes et des conditions associées aux transitions (réceptivités) du langage SFC. Il est remplacé par le langage C dans les API les plus puissants.

I.13) Normalisation de l'industrie des API

La difficulté posée par les API est la multitude des versions des langages de programmation de chaque marque. Afin de limiter cette diversion, une norme internationale a été établie dans un premier temps pour le langage le plus adopté par les fabriquant à savoir le langage LAD. Cette norme est publiée en 1993 par la Commission Electrotechnique Internationale, est désignée sous la référence CEI 61131 (pour les langages CEI 61131-3). La dernière version, qui date de 2013, est une extension qui reste compatible avec la version antérieure. Cette norme couvre l'intégralité du cycle de vie des API. Elle est composée de huit parties. Elle comprend une bibliothèque de fonctions préprogrammées. Elle donne une définition formelle de chaque paramètre d'entrée et de sortie afin que des blocs fonctionnels conçus par différents programmeurs (de différentes marques) puissent être facilement interconnectés. N'importe qu'elle marque d'API conforme à la norme CEI 61131 prend en charge ces fonctions sous forme d'une bibliothèque, le code étant écrit dans la partie PROM d'une mémoire flash de l'API.