Le Chapitre 3. Automates programmables industriels API

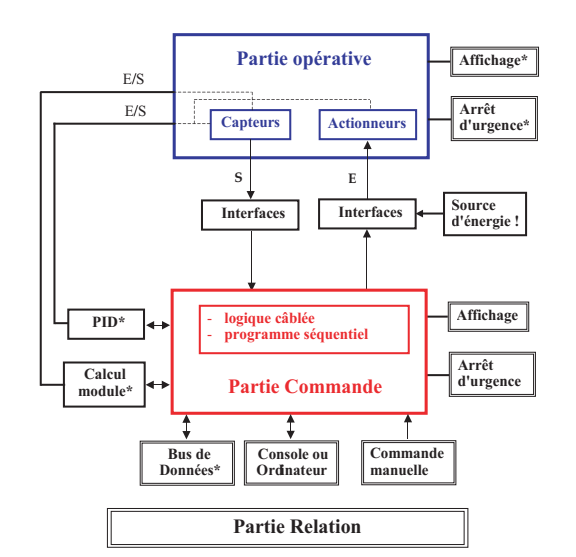
1. **Les Systèmes Automatisés de Production**

Un système de production est dit automatisé lorsqu’il peut gérer de manière autonome  
un cycle de travail préétabli qui se décompose en séquences et/ou en étapes. 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 (SCC) ;

– la partie relation (PR) de plus en plus intégrée dans la partie commande.



* **La partie opérative**

C’est la partie visible du système. Elle comporte les éléments du procédé, c’est a dire :  
– des pré-actionneurs (distributeurs, contacteurs) qui reçoivent des ordres de la partie commande ;

– des actionneurs (vérins, moteurs, vannes) qui ont pour rôle d’exécuter ces ordres. Ils transforment l’´énergie pneumatique (air comprimé), hydraulique (huile sous pression) ou électrique en énergie mécanique;

– des capteurs qui informent la partie commande de l´exécution du travail. Par exemple, on va trouver des capteurs mécaniques, pneumatiques, électriques ou magnétiques montés sur les vérins. Le rôle des capteurs (ou détecteurs) est donc de contrôler, mesurer, surveiller et  
informer la PC sur l’´évolution du système.

* **La partie commande**

Ce secteur de l’automatisme 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. L’outil de description de la partie commande s’appelle GRAFCET.

* **La partie relation**

Sa complexité dépend de l’importance du système. Elle regroupe les différentes commandes nécessaires au bon fonctionnement du procédé, c’est à dire marche/arrêt, arrêt d’urgence, marche automatique, etc ... L’outil de description s’appelle le Guide d’Etudes des Modes de Marches et d’Arrêts (GEMMA).

Les outils graphiques, que sont le GRAFCET et le GEMMA, sont utilisés par les automaticiens et les techniciens de maintenance.

1. **L’automate programmable industriel**

La structure simplifiée d’un ensemble automatisé peut se décomposer en trois parties essentielles :

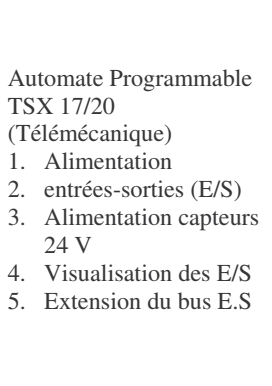
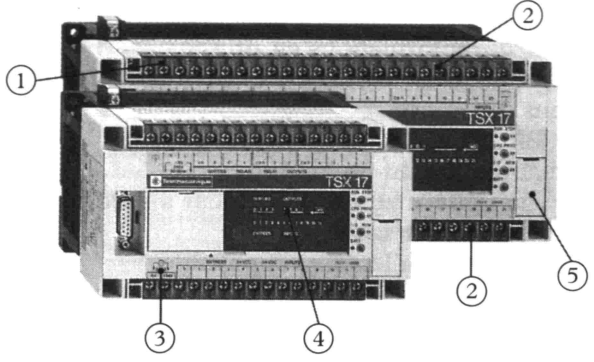
* les entrées (capteurs) Tout Ou Rien (TOR), parfois analogiques, destinées à fournir des informations sur l’état du processus : fin de course, détecteur de niveau, pressostat, thermostat, etc.
* l’automate qui traite les différentes informations d’entrée afin d’élaborer les ordres,
* les sorties transmettant les ordres élaborés par l’automate, aux différents actionneurs ou pré- actionneurs. : distributeurs de vérins, contacteurs de moteur,……

Un API est un 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’opérateur écrit, dans un langage d’application propre à l’automate, des directives concernant le déroulement du processus à automatiser.

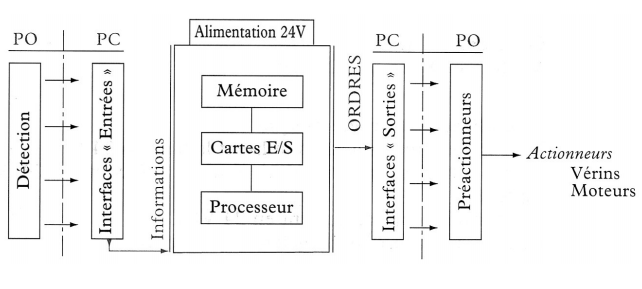
Son rôle consiste donc à fournir des ordres à la partie opérative en vue d'exécuter un travail précis comme par exemple la sortie ou la rentrée d'une tige de vérin. Celle-ci, en retour, lui donnera des informations relatives à l'exécution dudit travail.

**II.1 Structure d’un API**

**Structure externe d’un API**

****

**Structure interne d’un API**



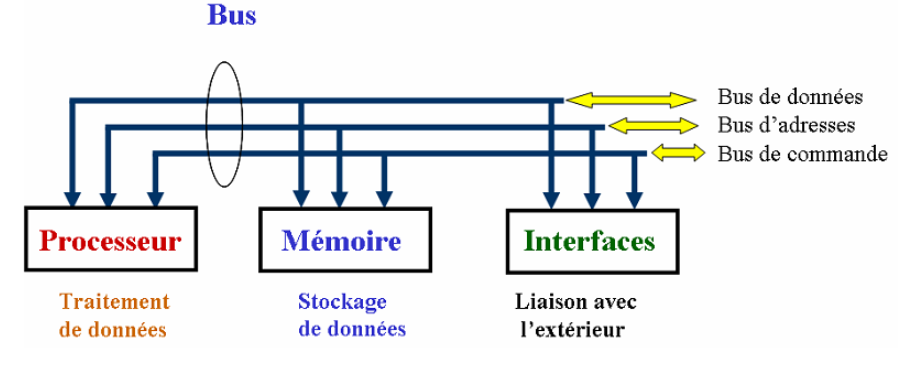
Les automates programmables comportent généralement quatre parties principales :  
- Une mémoire,

- Un processeur (ou microprocesseur),

- Des interfaces d’Entrées/Sorties,

- Une alimentation (220 V ---> 24 V).

Ces quatre parties sont reliées entre elles par des « bus» (ensemble de fils autorisant le passage des informations entre ces quatre secteurs de l'automate).



* **La mémoire**

Contient le programme qui définit les actions de commande effectuées par le microprocesseur. Elle est conçue donc pour recevoir, gérer, stocker des informations issues des différents secteurs du système qui sont :

- le terminal de programmation : introduction du programme ;

- le processeur qui 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.  
 **RAM** : elle est accessible en lecture et en écriture, elle s'efface automatiquement à l'arrêt de l'automate.

**ROM** : mémoire programmé par le fabriquant, elle est ineffaçables.

**PROM** : vendues vierges et programmables une seule fois par l’utilisateur

**REPROM** ou **EPROM** : utilisables plusieurs fois (écriture / effacement). Effacement à l’UV  
(Ultraviolet) pendant 10 à 30 minutes. Elles ne peuvent être reprogrammées qu’après un  
effacement total ;

**EEPROM** : effacement électrique et reprogrammation rapide sur place !

* ***Le processeur***ou ***unité centrale de traitement***(CPU, *Central Processing Unit*)

Contient le microprocesseur. Le CPU 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’action.

* **Les interfaces d’E/S**

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 et sorties 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.

* **L’alimentation électrique**

Permet de fournir à l’automate l’énergie nécessaire à son fonctionnement. Ils délivrent, à  
partir du 220 V alternatif, des sources de tension nécessaires à l’automate tels que : +5V,  
12V et 24V en continu.

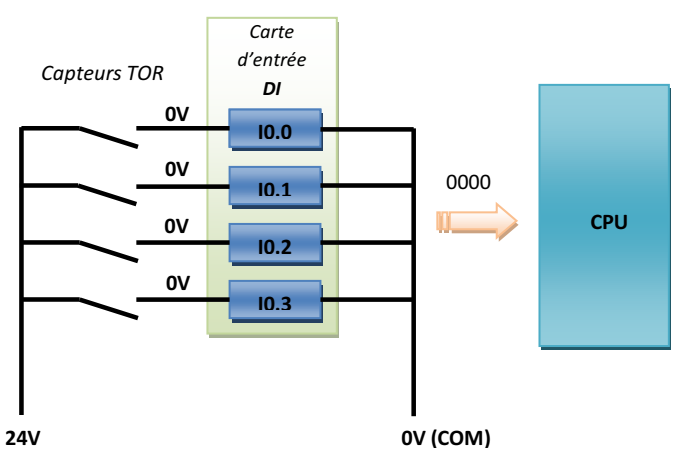
**II.2 Les carte d’Entrées/Sorties**

1. **Cartes d’entrées/sorties logiques (TOR)**

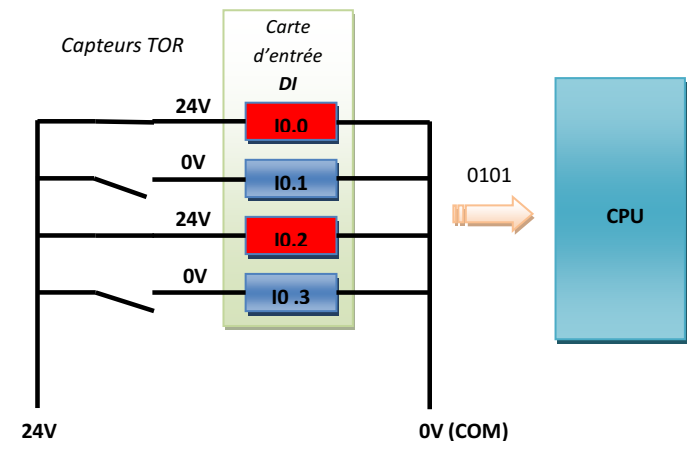
Les API offrent une grande variété d’E/S TOR (Tous Ou Rien) adaptées au milieu industriel et qui peuvent accepter suivant les cartes, des informations en courant ou en tension, alternatifs ou continus.

Chaque carte comporte généralement 8, 16, 32 entrées logiques

***Les entrées TOR*** *:* Elles permettent de raccorder à l’automate les différents capteurs logiques tels que : boutons poussoirs.

****

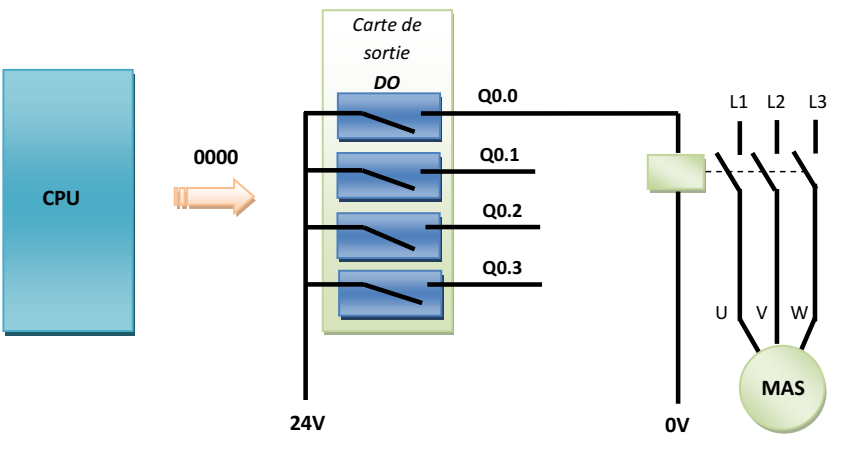
***Principe de connexion des entrées état au repos***

****

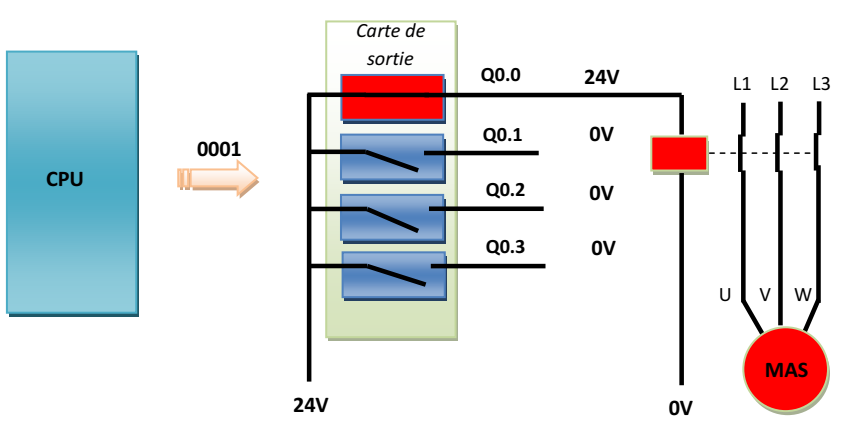
***Principe de connexion des entrées état actionnées***

Les entrées des capteurs TOR sont connectés ensemble par le pole plusse 24v fournie par l’automate et les sorties sont connectés aux entrées I0.0, I0.1, I0.2,….

***Les sorties TOR*** *:* Elles permettent de raccorder à l’automate les différents pré-actionneurs tels que : vannes,

****

***Principe de connexion des sorties état au repos***

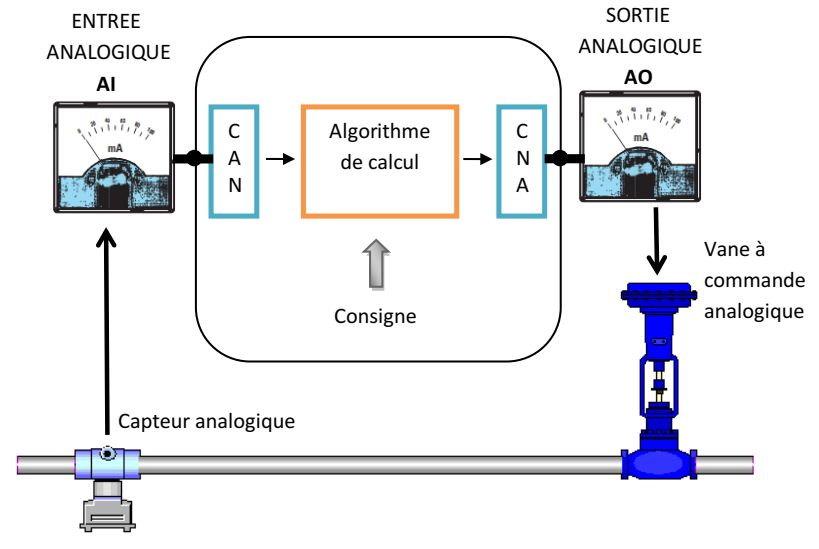
****

***Principe de commande des sorties état actionnée***

L’API commande le processus en connectant des **actionneurs** via les points de connexion appelés **sorties** à une tension de commande (24 V), ceci permet par exemple de démarrer ou d’arrêter un moteur, suivant un programme traité par l’automate.

1. **Cartes d’entrées/sorties analogiques**

A la différence des signaux binaires qui ne peuvent prendre que les deux états "Tension disponible +24V" et "Tension indisponible 0V", les signaux analogiques sont capables de prendre n'importe quelle valeur comprise entre 0V et 10V ou de 0 à 20mA, (Exemples: Température -50… +150°C, Débit 0… 200 l/min, Vitesse 0… 1500 tr/min)



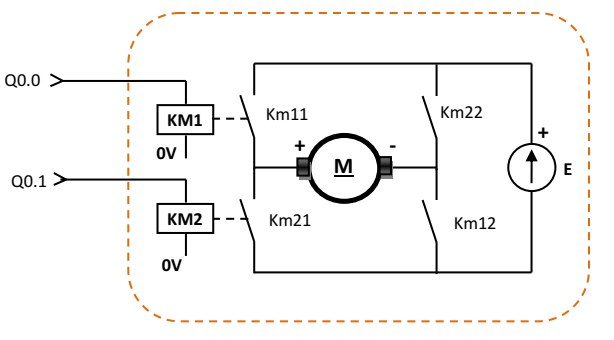
1. **Les actionneurs et les capteurs**

**III.1. Les actionneurs**

* **Commande d’un Moteur à courant continu à aimant permanant par API**

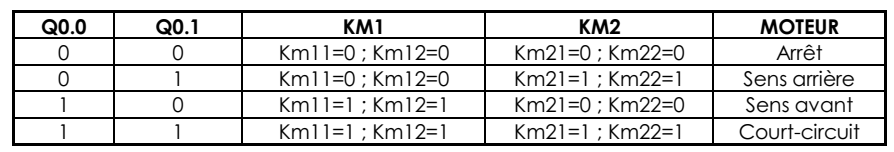
Pour inverser le sens de rotation d’un moteur à courant continu (MCC) et à aimant permanant, on permute les polarités de l’alimentation. Le circuit de puissance d’un MCC commandé par deux relais électromagnétiques.

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 ou KM2.



***Circuit de puissance pour commander un MCC en deux sens de rotation***

Le tableau suivant résume les ordres de commande en tenant compte des  
conditions de fonctionnement et de la protection de l’alimentation contre les courts  
circuits.

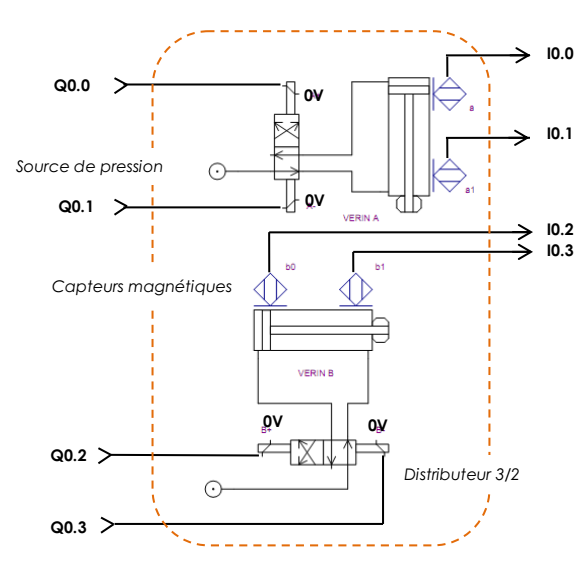


***Table d’ordre de commande et conditions de fonctionnements***

* **Commande des vérins à doubles effets**

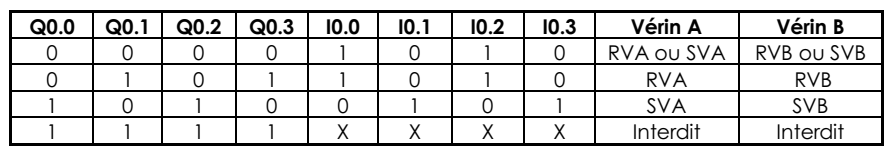
La commande des vérins pneumatiques nécessite un distributeur afin de distribuer la pression d’air pour faire sortir ou retirer la tige.

Un exemple utilisant deux distributeurs à commande électrique piloté par le module de sortie TOR (Q0.0 et Q0.1) d’un API.



***Circuit pneumatique pour commander deux vérins par un API***

Le tableau suivant résume les ordres de commande de l’API vers les solénoïdes des distributeurs et en tenant compte de la sécurité mécanique des vérins par une alimentation simultanée des deux solénoïdes.



RVA : recul vérin A ; RVB : recul vérin B

***Table d’ordre de commande et conditions de fonctionnements***

**III.2. Les capteurs**

Un capteur transforme une grandeur physique en une grandeur normée, généralement électrique, qui peut être interprétée par un API.

Mesurer une grandeur physique pour exploiter correctement un système automatisé il est nécessaire :

**De mesurer les variations de certaines grandeurs physiques (**La vitesse du vent, La pression, La température de l’eau …).

**De contrôler l’état physique de certains de ses constituants (**La position levée d’une barrière de parking, La présence d’une pièce, La présence de pression dans un circuit, La position d’un chariot…).

Suivant la nature du signal exploitable les capteurs se classent en trois catégories :

* CAPTEURS LOGIQUES du type tout ou rien (TOR).
* CAPTEURS ANALOGIQUES.
* CAPTEURS NUMERIQUES.

1. **Choix d’un automate programmable industriel**

Les automates doivent fonctionner sans danger pour les utilisateurs et sans risque  
d’interrompre la production et ceci malgré les contraintes très sévères qu’ils subissent en  
milieu industriel (Environnement physique et mécanique, Environnement chimique, Environnement électrique).

Après l’établissement du cahier des charges, il revient à l’utilisateur de regarder sur le marché l’automate le mieux adapté aux besoins, en considérant un certain nombre de critères importants :

- Le type et nombre des entrées/sorties nécessaire

- La nature du traitement (temporisation, comptage, ...) ;

- Les moyens de dialogue et le langage de programmation;

- La communication avec les autres systèmes ;

- Les moyens de sauvegarde du programme ;

- La fiabilité, robustesse, immunité aux parasites ;

- La documentation, le service après-vente, durée de la garantie, la formation

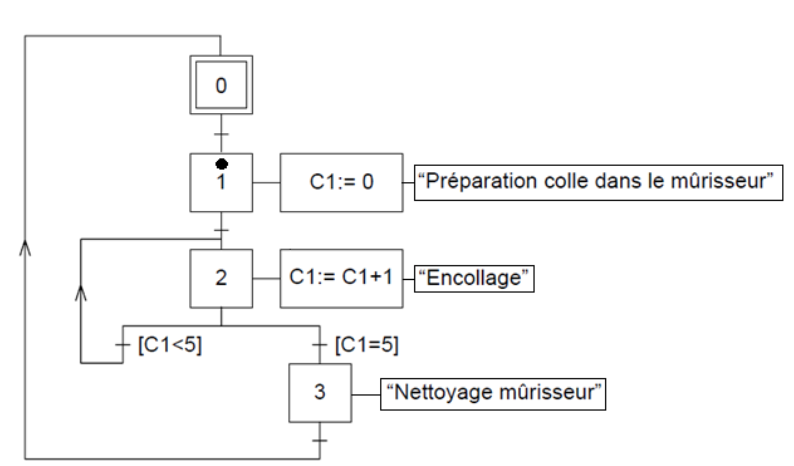
- Les capacités de traitement du processeur (vitesse, données, opérations, temps réel...)

- Les compétences/expériences de l’équipe d’automaticiens en mise en œuvre et en programmation de la gamme d’automate.

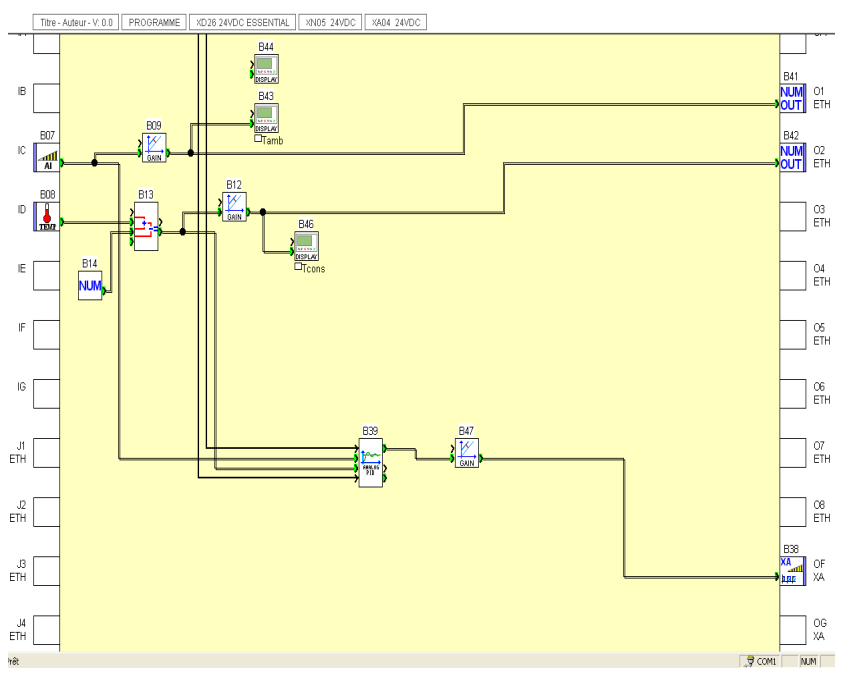
1. **Mise en œuvre et programmation d’un API**

Généralement, la mise en œuvre d’un API est la réalisation d’une partie commande en logique programmée nécessite la traduction du modèle concerné (GRAFCET, schémas, équations, ...) en programme exécutable par la machine. L’élaboration d’un tel programme vise donc à écrire les équations d’activation de sorties de l’API et les conditions associées.

Chaque automate possède son propre langage. Mais par contre, les constructeurs proposent tous une interface logicielle répondant à la norme CEI 11313 (*Commission Electrotechnique Internationale*). Cette norme définit **cinq langages** de programmation utilisables, qui sont :

– **GRAFCET ou SFC (*Sequential Function Charts*)** : ce langage de programmation de haut niveau permet la programmation aisée de tous les procédés séquentiels. 

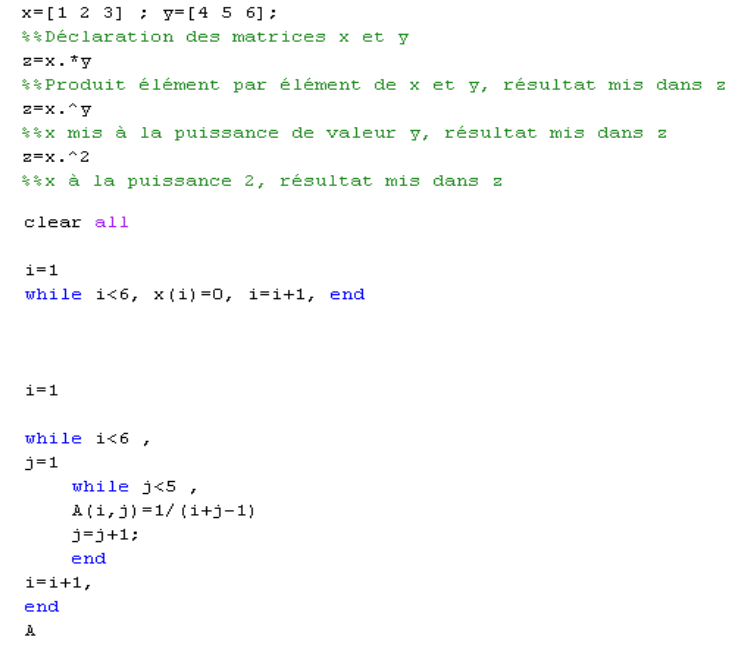
– **Schéma par blocs ou FBD (*Function Block Diagram*)** : ce langage permet de programmer graphiquement à l’aide de blocs, représentant des variables, des opérateurs ou des fonctions. Il permet de manipuler tous les types de variables.



– **Schéma à relais ou LD (*Ladder Diagram*)** : ce langage graphique est essentiellement dédié à la programmation d’équations booléennes (true/false).



– **Texte structuré ou ST (*Structured Text*)** : ce langage est un langage textuel de haut niveau. Il permet la programmation de tout type d’algorithme plus ou moins complexe (exemple MATLAB).



– **Liste d’instructions ou IL (*Instruction List*)** : ce langage textuel de bas niveau est un langage à une instruction par ligne. Il peut être comparé au langage assembleur.



* **Exemple de programmation en langage LADDER**

Le langage LD est un langage à contacts permet la programmation à l’aide de symboles graphiques. Ces symboles sont organisés en suite réseaux reliés à gauche et à droite à des barres d’alimentation.

Les entrées sont représentées par des interrupteurs (contacts), et les sorties par des  
ou des bascules.

Les composants graphiques élémentaires d’un diagramme LD sont :

|  |  |  |
| --- | --- | --- |
| **N⁰** | **symbole** | **Désignation** |
| 1 |  | Contact d’entrée Normalement Ouvert |
| 2 |  | Contact d’entrée Normalement Fermé |
| 3 |  | Contact d’entrée détection front montant |
| 4 |  | Contact d’entrée détection front descendant |
| 5 |  | Contact inversé |
| 6 |  | Bobine de sortie |
| 7 |  | bobinage complémenté (inverse) |
| 8 |  | bobinage SET (verrouillage : mise à 1 du bit adressé) |
| 9 |  | bobinage RESET (déverrouillage : mise à 0 du bit adressé) |
| 10 |  | L’attente d’un Front Montant |
| 11 |  | L’attente d’un Front descendant |

Les sorties sont obligatoirement à droite du réseau

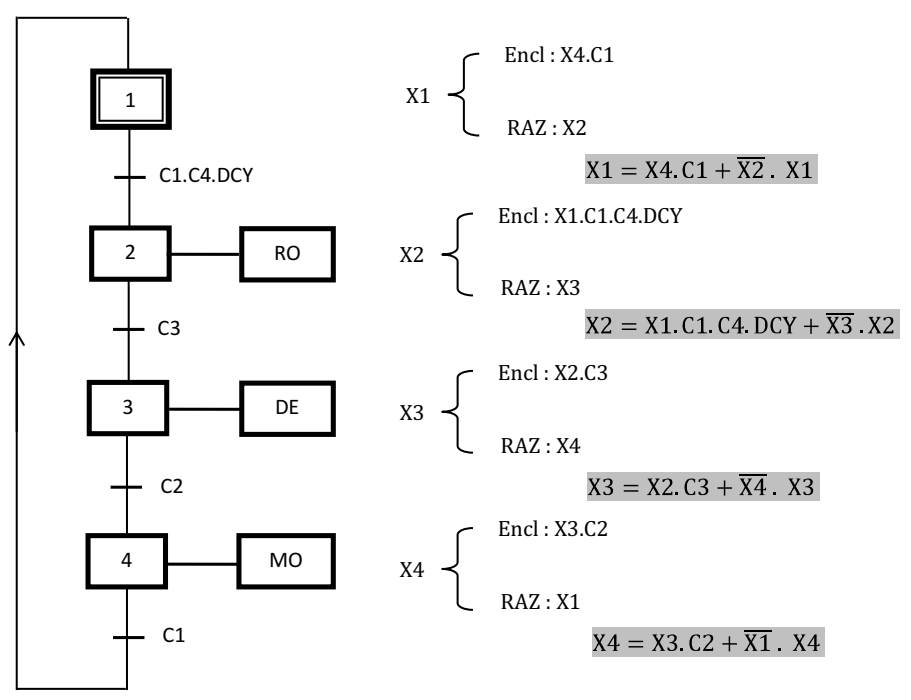
On doit évidemment identifier nos **E/S**, par leur code (**Ia.b** / **Qa.b**).

On relie les éléments en série pour la fonction **ET**, en parallèle pour le **OU**.

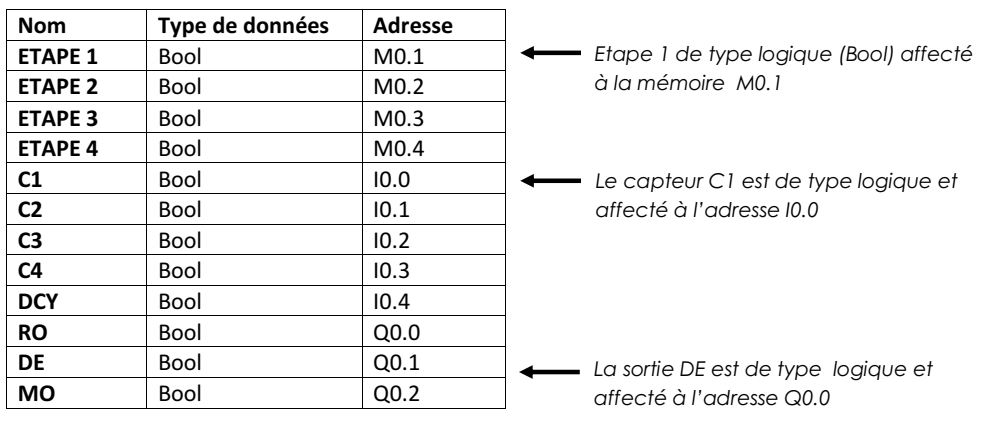
* **Adressage des entrées/sorties**

La déclaration d'une entrée ou sortie donnée à l'intérieur d'un programme s'appelle  
l'adressage. Les entrées et sorties des API sont la plupart du temps regroupées en groupes de huit sur des modules d'entrées ou de sorties numériques. Cette unité de huit est appelée **octet**. Chaque groupe reçoit un numéro que l'on appelle l'**adresse d'octet**.

Afin de permettre l'adressage d'une entrée ou sortie à l'intérieur d'un octet, chaque octet est divisé en huit **bits**. Ces derniers sont numérotés de 0 à 7. On obtient ainsi l'**adresse du bit…** L'API représenté ici a les octets d'entrée 0 et 1 ainsi que les octets de sortie 0 et 1.

* **Exemple:**

**Adressage des entrées/sorties**



*Tableau: table de variables mnémoniques*

Par exemple pour adresser la 5ème entrée du **DCY** en partant de la gauche, on définit l’adresse suivante :

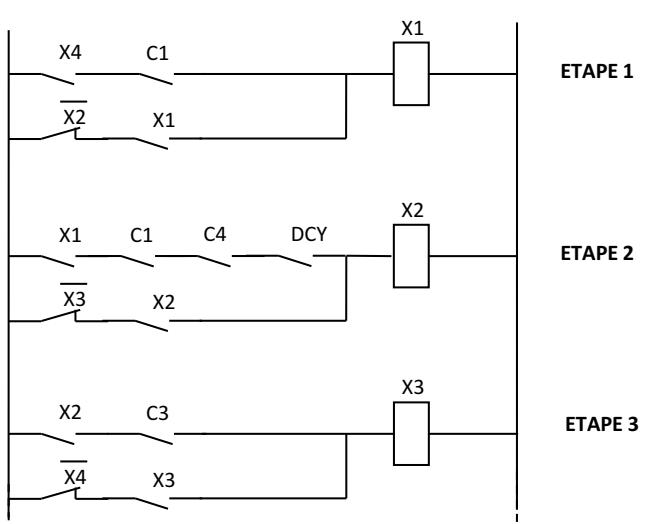
**I0.4: I** indique une adresse de type entrée, **0** l’adresse d’octet et **4** l’adresse de bit. Les adresses d’octet et de bit sont toujours séparées par un point.

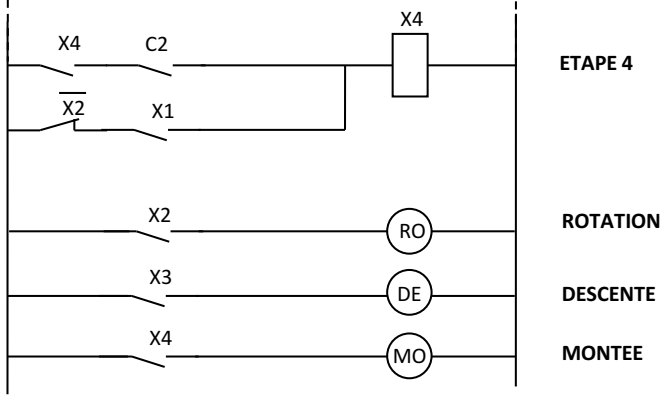
Pour adresser la 3ème sortie, par exemple, on définit l’adresse suivante :

**Q0.2 Q** indique une adresse de type Sortie, **0** l’adresse d’octet et **2** l’adresse de bit. Les adresses d’octet et de bit sont toujours séparées par un point.

***Remarque :*** *la numérotation des adresses commence à zéro (Le bit de la première sortie est un 0).*

Les équations des mémoires étape déterminée précédemment nous donnent le schéma de câblage électrique suivant :

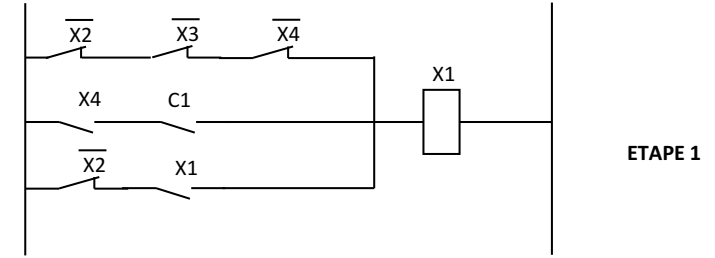




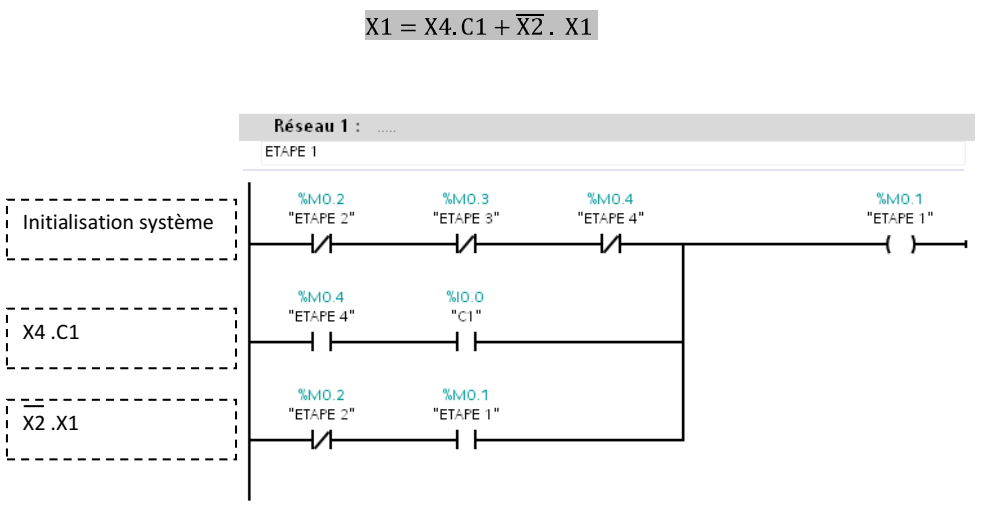
Pour établir la commande de chaque sortie, il suffit de considérer la ou les étapes durant lesquelles la sortie doit être enclenchée. Ainsi:  
La sortie RO a lieu durant l’ETAPE 2 d’où RO = X2  
La sortie DE a lieu durant l’ETAPE 3 d’où DE = X3  
La sortie MO a lieu durant l’ETAPE 4 d’où MO = X4

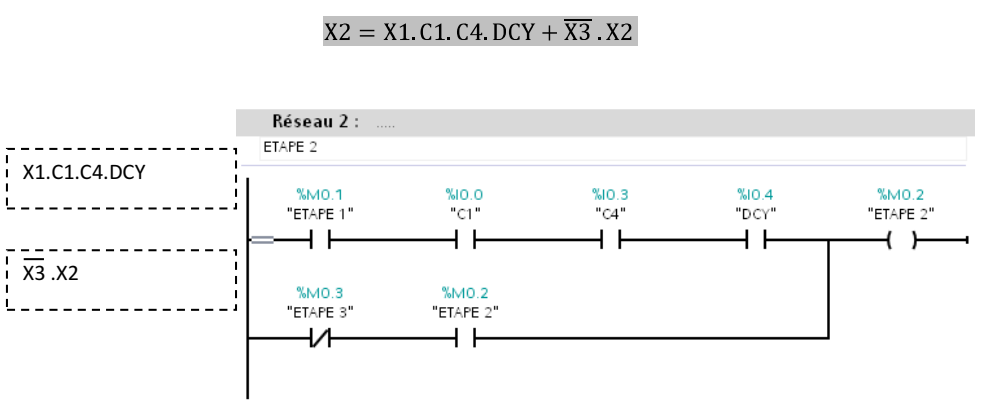
**Initialisation de la séquence :** Nous remarquons sur le schéma précédent qu’à la mise sous tension, toutes les mémoires se trouvant ici à l’état repos, aucune évolution n’est possible.  
Il est donc impératif d’initialiser la séquence en venant enclencher la mémoire X1 matérialisant l’étape initiale de notre GRAFCET.

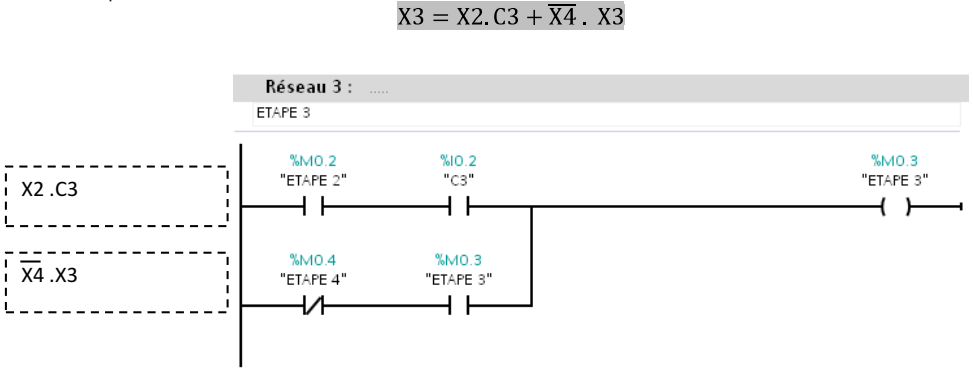
En testant l’état repos de toutes les mémoires d’étape suivantes, pour venir alors systématiquement enclencher la mémoire X1, comme le montre le schéma suivant :

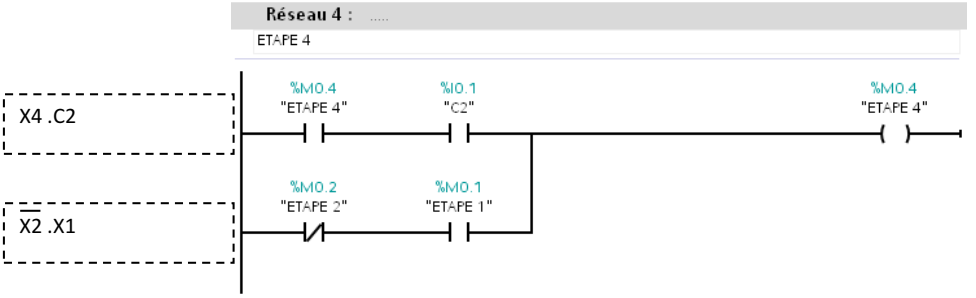


**Programmation des entrées**





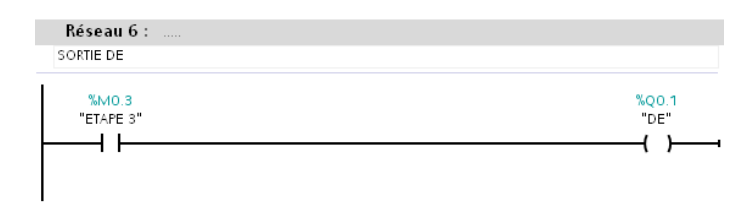




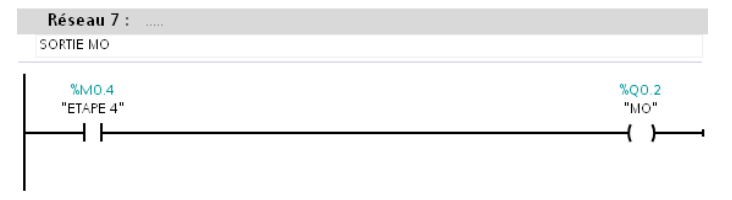
**Programmation des sorties  
R0** : est actionné uniquement à l’étape 2



**DE** : est actionné uniquement à l’étape 3



**MO** : est actionné uniquement à l’étape 4



Le programme peut être simplifié si en utilisant les bobines **Set/ Reset** ou les bascules **SR** ou **RS** et en tenant compte des cinq règles du GRAFCET.

