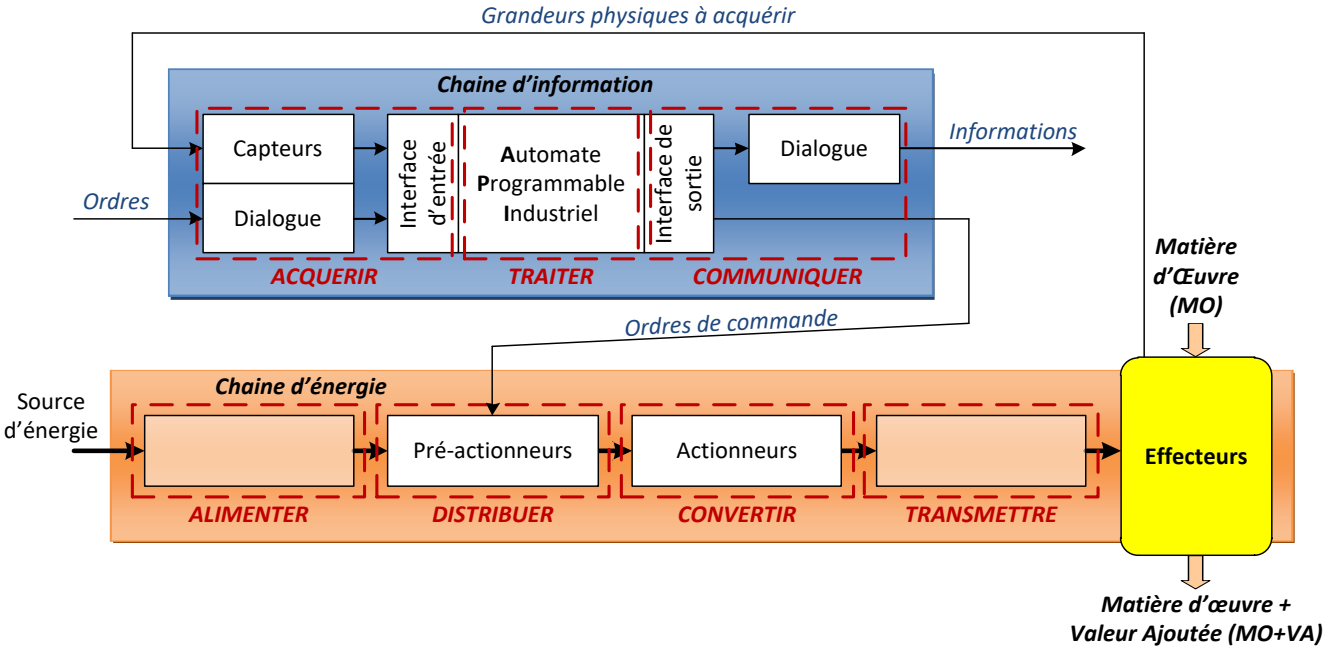
AUTOMATE PROGRAMMABLE INDUSTRIEL

1. Rappels : structure fonctionnelle des systèmes ou produits automatisés

Les principales catégories de constituants assurant les différentes fonctions d'un système automatisé sont les suivantes :



Fonction acquérir :

- Commandes et consignes : *Boutons poussoirs, claviers, interfaces tactiles, …*
- Informations : *Capteurs de position, détecteurs, codeurs numériques, …*

Fonction traiter :

- Logique câblée : *Câblages, raccordements, cartes électroniques, …*
- Logique programmée : *Automates programmables, microcontrôleur, programmes informatiques, …*

Fonction communiquer :

- Messages et visualisation : *Voyants, écrans, afficheurs, …*
- Ordres de commande : *Liaisons directes, liaisons informatiques, réseaux sans fils, …*

2. L'automate programmable industriel -API

2.1. Définition d'un automate programmable

Selon la norme NFC 63-850 : un automate programmable est un appareil électronique qui comporte une mémoire, programmable par un utilisateur automaticienà l'aide d'un langage adapté, pour le stockage interne des instructions composant les fonctions d'automatisme comme par exemple :

- Logique séquentielle et combinatoire ;
- Temporisation, comptage, décomptage, comparaison ;
- Calcul arithmétique ;
- Réglage, asservissement, régulation, etc,

L'automate sert à commander, mesurer et contrôler au moyen d'entrées et de sorties (logiques, numériques ou analogiques) différentes sortes de processus, systèmes ou produits…

ZELIO de chez Schneider

EM4 de chez Crouzet

S71200 de chez Siemens

2.2. Avantages et inconvénients

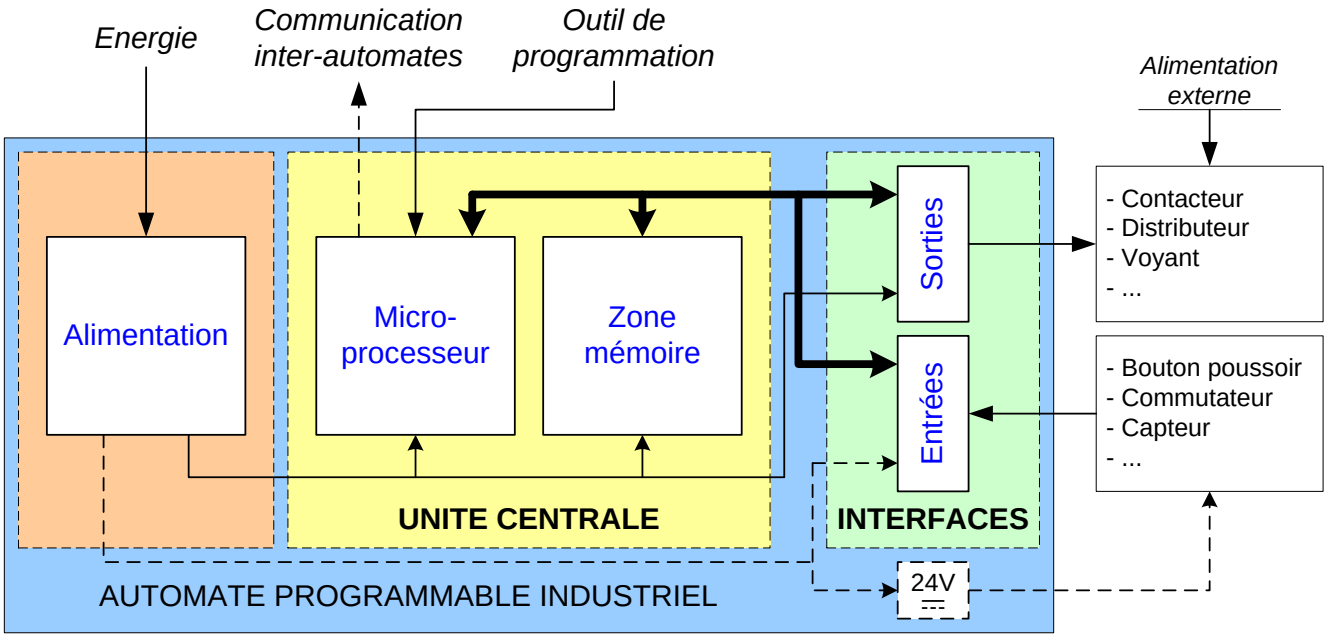
Les API présentent de nombreux intérêts :

- *Les éléments qui les composent sont particulièrement robustes leur permettant de fonctionner dans des environnements particulièrement hostiles* (poussière environnante, perturbations électromagnétiques, vibrations des supports, variations de température...)
- *Ils possèdent des circuits électroniques optimisés pour s'interfacer avec les entrées et les sorties physiques du système, les envois et réceptions de signaux se font très rapidement avec l'environnement.* Avec de plus une exécution séquentielle cyclique sans modification de mémoire, ils permettent d'assurer un temps d'exécution minimal, respectant un déterminisme temporel et logique, garantissant un temps réel.

Ils restent à l'heure actuelle les seules plateformes d'exécution considérées comme fiables en milieu industriel (avec les ordinateurs industriels). De plus ils nécessitent la maîtrise de langages spécifiques conformes à la norme CEI 61131-3 qui reprennent dans leur forme la logique d'exécution interne de l'automate. Ces langages apparaissent toutefois à beaucoup d'utilisateurs plus accessibles et plus visuels que les langages informatiques classiques.

2.3. Structure des automates programmables :

Un automate programmable reçoit les informations relatives à l'état du système et commande les pré-actionneurs suivant un ordre de séquencement défini dans une liste d'instructions qui lui sera communiquée.



Alimentation :

- Elle fournit à partir des tensions usuelles (230 V, 24 V = ou ~) les tensions continues nécessaires au fonctionnements des circuits électroniques.
- Elle fournit parfois aussi l'alimentation nécessaire au fonctionnement des entrées.

Unité centrale :

- Le **microprocesseur** (unité de traitement) réalise toutes les fonctions logiques, les fonctions de temporisation, de comptage, de calcul … à partir d'un programme contenu dans sa mémoire.

  Il possède des voies de communication avec l'extérieur :
  - dialogue avec l'outil de programmation,
  - raccordement sur un réseau de communication inter-automates.
- La **zone mémoire** contient, la liste d'instructions qui constituent le programme de fonctionnement du système, ainsi que toutes les données utiles au fonctionnement interne de l'automate.

Interfaces :

- Les **entrées** reçoivent des informations en provenance des éléments de détection et du pupitre opérateur.
- Les **sorties** transmettent des informations aux pré-actionneurs et aux éléments de signalisation du pupitre.

3. Langages de programmation pour API: Norme IEC 1131-3

La norme définit cinq langages qui peuvent être utilisés pour la programmation des automates industriels. Ces cinq langages sont :

- LD (« Ladder Diagram », ou schéma à relais): ce langage graphique est essentiellement dédié à la programmation d'équations booléennes (vraie/faux).
- IL (« Instruction List », ou liste d'instructions): ce langage textuel de bas niveau est un langage à une instruction par ligne. Il peut être comparé au langage assembleur.
- FBD (« Function Block Diagram », ou schéma par blocs): ce langage permet de programmer graphiquement à l'aide de blocs, représentant des variables, des opérateurs ou des fonctions logiques. Il permet de manipuler tous les types de variables.
- SFC (« Sequential Function Char »): ce langage, de haut niveau, permet la programmation aisée de tous les procédés séquentiels.Il se rapproche du GRAFCET (Graphe Fonctionnel de Commande des Étapes et Transitions) , mode de représentation et d'analyse d'un automatisme.
- ST («Structured Text » ou texte structuré): ce langage est un langage textuel de haut niveau. Il permet la programmation de tout type d'algorithme plus ou moins complexe.

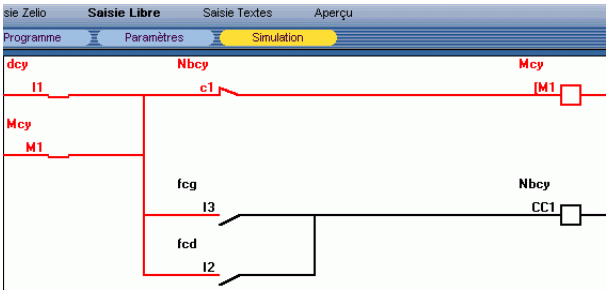
3.1. Objets communs

Toute expression, constante ou variable, utilisée dans un programme doit être caractérisée par un type :

- BOOL (Booléen) : ce type donne la valeur TOR qui sont équivalent à « 1 » ou « 0 ».
- DINT (Entier) : c'est un nombre entre -2147483647 et +2147483647. Il est exprimé dans l'une des bases suivantes : décimale, hexadécimale, octale ou binaire.
- REAL (Réel), en base 10. Il peut prendre 1 bit de signe.
- WORD (Mot) , souvent codé en hexadécimal
- TIME (Temporisation) : c'est une valeur strictement positive et commence par T#
- STRING (Chaîne) : elle doit être précédée et suivie par une apostrophe, et ne doit jamais excéder 255 caractères).

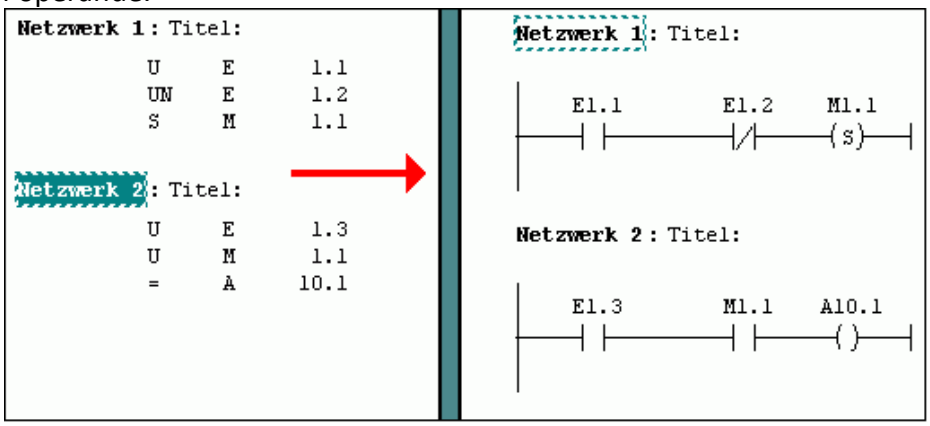
3.2. Langage LD

Le langage LD (ladder diagram) est une représentation graphique d'équations booléennes combinant des contacts (en entrée) et des relais (en sortie). Il permet la manipulation de données booléennes, à l'aide de symboles graphiques organisés dans un diagramme comme les éléments d'un schéma électrique à contacts. Les diagrammes LD sont limités à gauche et à droite par des barres d'alimentation.



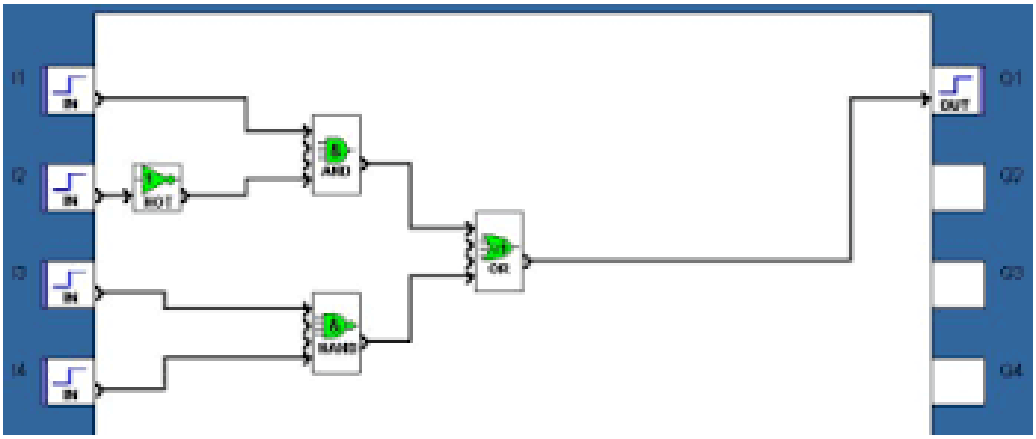
3.3. Langage IL

Le langage IL (instruction list), est un langage textuel de bas niveau. 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.



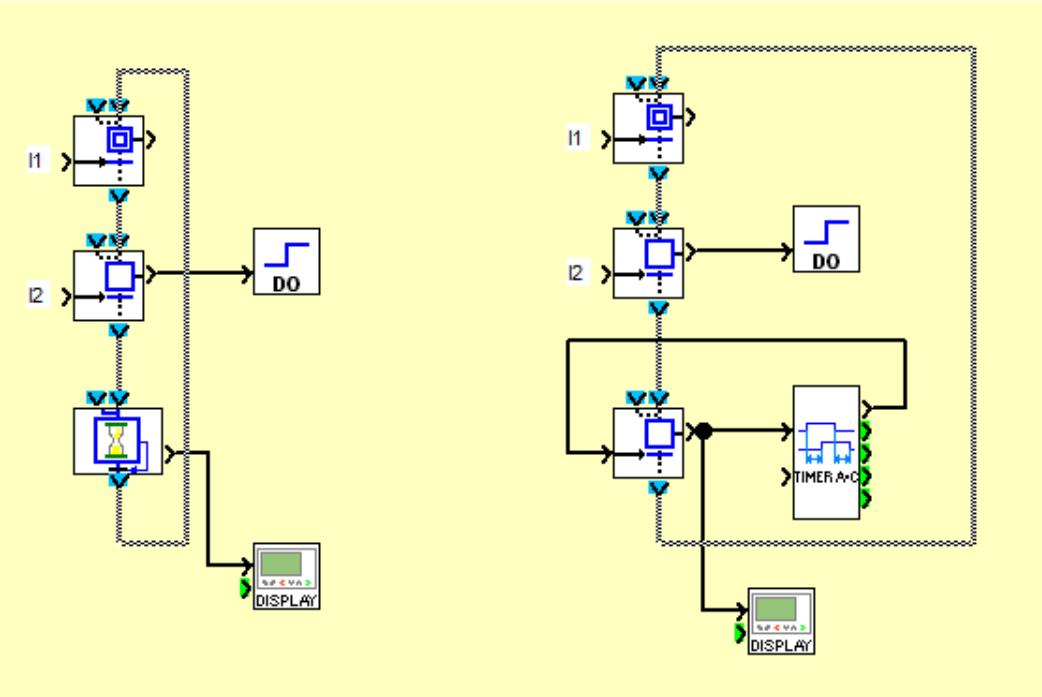
3.4. Langage FBD

Le langage FBD (function block diagram) est un langage graphique. Il permet la construction d'équations complexes à partir des opérateurs standard, de fonctions ou de blocs fonctionnels issus de la logique combinatoire.



3.5. Langage SFC

Le langage SFC (Sequential Function Chart), ou GRAFCET, est un langage graphique 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.



3.5.1. Les principales règles graphiques sont :

- un programme SFC doit contenir au moins une étape initiale.
- une étape ne peut pas être suivie d'une autre étape.
- une transition ne peut pas être suivi d'une autre transition.

3.5.2. Les composants de base (symboles graphiques) du graphique SFC sont :

- étapes et étapes initiales.
- Transitions.
- liaisons orientées.
- renvoi à une étape.

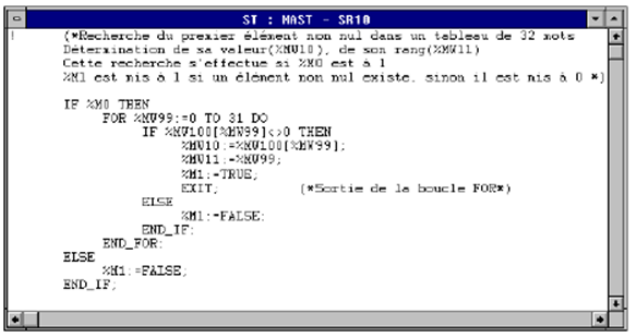
3.5.3. Les différents types d'action sont :

- action booléenne (Elle est forcée à chaque fois que le signal d'activité de l'étape change d'état.)
- action impulsionnelle programmée en ST, LD ou IL (c'est une liste d'instructions ST, IL ou LD, exécutée à chaque cycle pendant toute la durée d'activité de l'étape).
- action normale programmée en ST, LD ou IL ;
- action SFC (Une action SFC est une séquence fille SFC, lancée ou tuée selon les évolutions du signal d'activité de l'étape. Elle peut être décrite avec les qualificatifs d'action N (non mémorisée), S (set), ou R (reset).)

Plusieurs actions (de même type ou de types différents) peuvent être décrites dans la même étape. Un appel de fonctions ou de blocs fonctionnels permet d'intégrer des traitements décrits dans d'autres langages (FBD, LD, ST ou IL).

3.6. Langage ST

Le langage ST (Structured Text) 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 du langage SFC.



3.7. Adressage I.E.C.

Les variables sont déclarées en utilisant un nom symbolique et une adresse logique :

%	Attribut	Type
	I: Entrée Q : Sortie M : mémoire K : constante	x : Boolean, 1 bit B : Byte, Octet, 8 bits W : Word, Mot,16 bits D : Double Word, 32 bits F : Floating, Flottant

Ex : %I1.5 Entrée TOR n°5 de la carte n°1 du rack par défaut n°0
%Q2.4.F Sortie TOR voie F du module 4 du rack 2
%M128 bit interne n°128
%MW4 entier 16 bits n°4

4. Choix d'un automate programmable industriel API

4.1. Règles générales

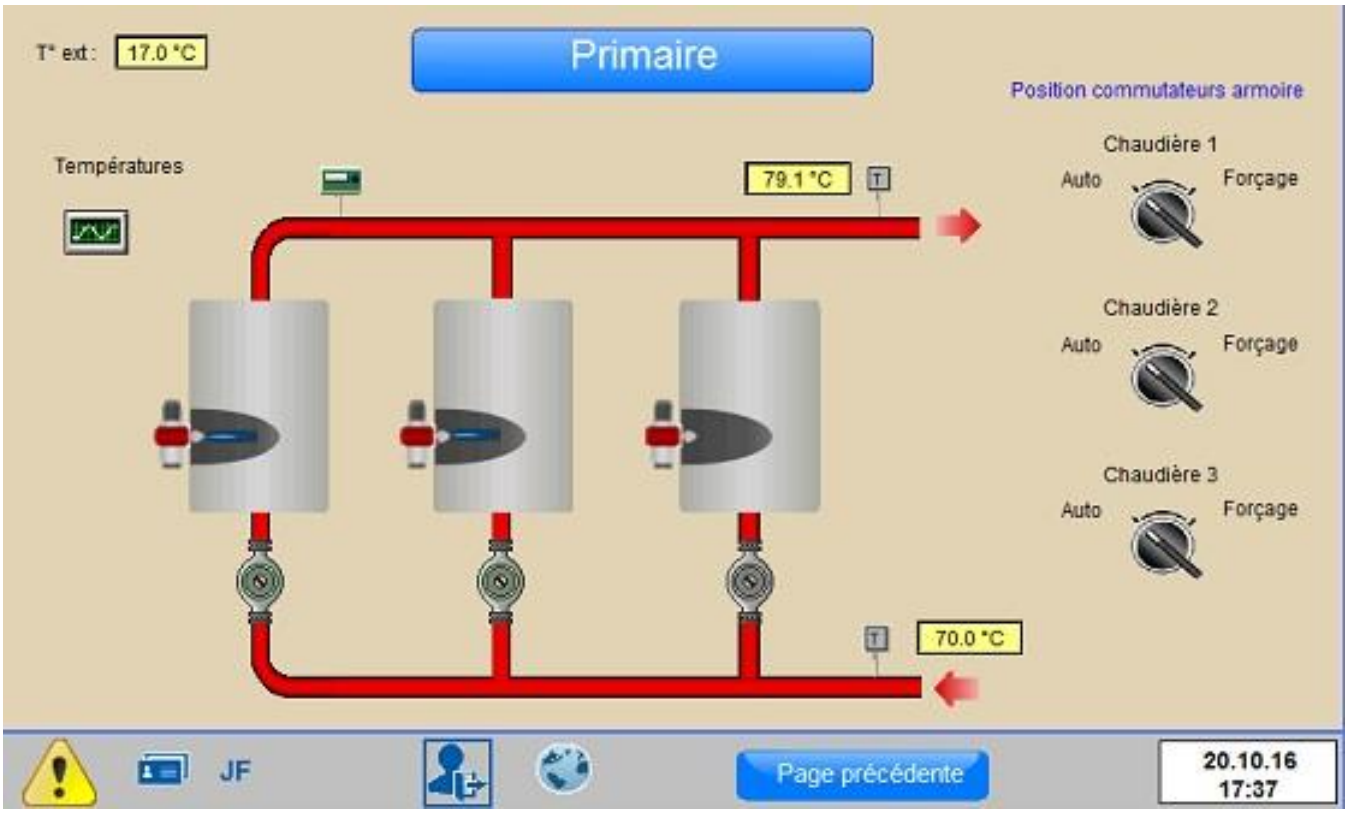
Le choix d'un automate programmable est en premier lieu le choix de l'équipe d'automaticiens. Le personnel de maintenance doit aussi être formé sur ces matériels. Un automate utilisant des langages de programmation de type SFC (GRAFCET) est également préférable pour assurer les mises au point et dépannages dans les meilleures conditions. La possession d'un logiciel de programmation est aussi sourced'économies (achat du logiciel et formation du personnel). Des outils permettant une simulation des programmes sont également souhaitables.

4.2. Principaux critères

Il faut ensuite quantifier les besoins :

- Type d'automate : *COMPACT ou MODULAIRE,*
- *Alimentation de l'automate et de ses entrées*
- *Nombre d'entrées / sorties :* le nombre de cartes peut avoir une incidence sur le nombre de racks dès que le nombre d'entrées / sorties nécessaires devient élevé.
- *Type de processeur :* la taille mémoire, la vitesse de traitement et les fonctions spéciales offertes par le processeur permettront le choix dans la gamme souvent très étendue.
- *Fonctions ou modules spéciaux :* certaines cartes (commande d'axe, pesage ...) permettront de "soulager" le processeur et devront offrir les caractéristiques souhaitées (résolution, ...).
- *Fonctions de communication :* l'automate doit pouvoir communiquer avec les autres systèmes de commande (API, supervision ...) et offrir des possibilités de communication avec des standards normalisés (Modbus, Ethernet TCP/IP, Profibus ...).

Exemple de supervision



5. Raccordement d'un Automate Programmable Industriel

5.1. Généralités

Conformément à la structure de la chaine d'information, on raccorde à l'API :

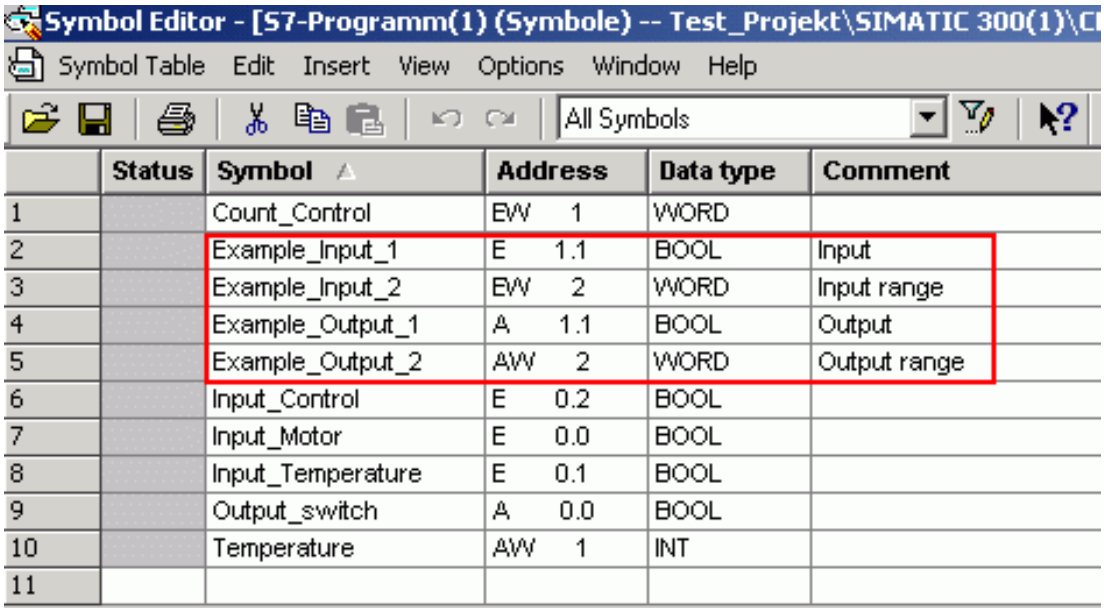
Les entrées (input) : Tout élément transmettant :

- des ordres en provenance de l'opérateur (boutons poussoirs, commutateurs, coups de poings d'arrêt d'urgence…),
- des informations en provenance du système lui-même (capteurs, relais thermiques…).

Les sorties (output) : Tout élément transmettant :

- soit des informations en direction de l'opérateur (voyants, afficheurs…),
- soit des ordres en direction du système lui-même (contacteurs, distributeurs…).

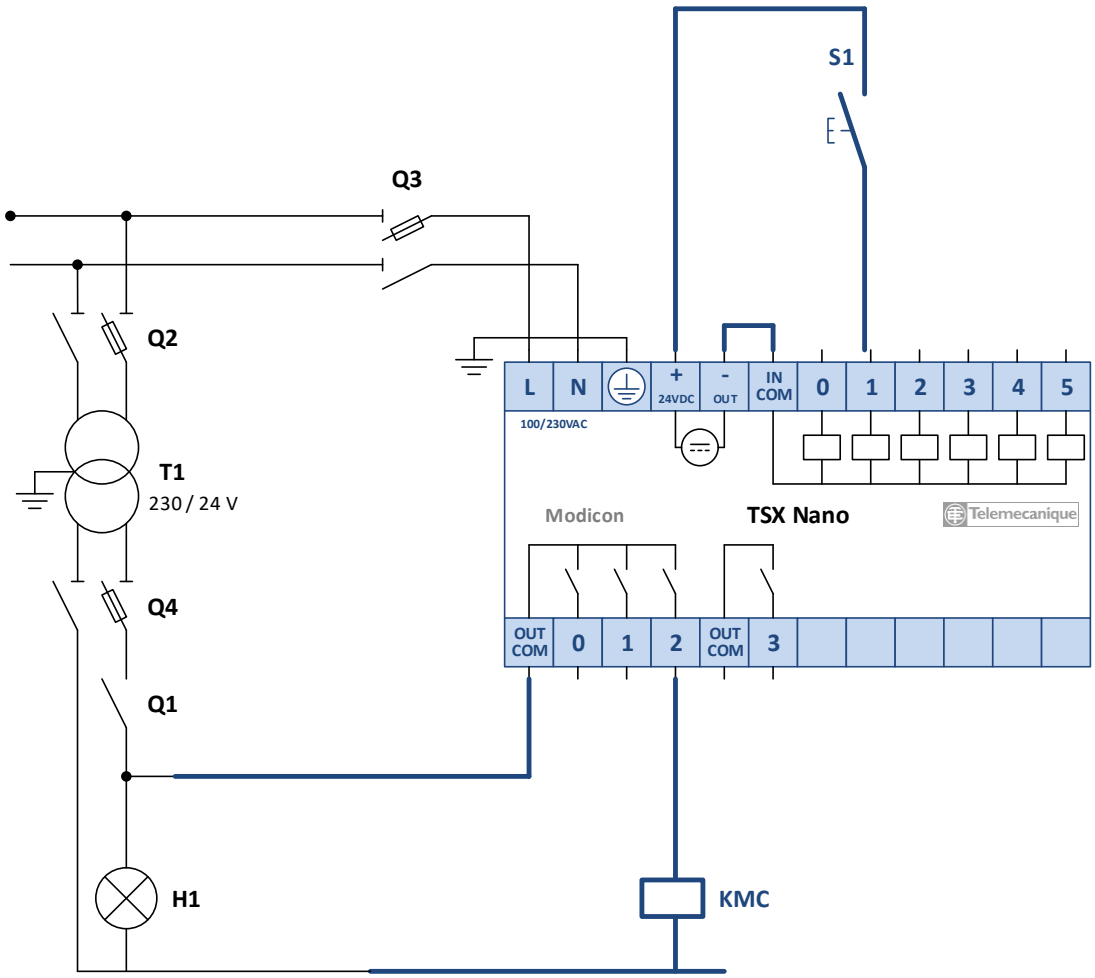
On créer souvent alors la table des mnémoniques utilisés qui est la correspondance entre les éléments physique du schéma et les adresses de l'automate :



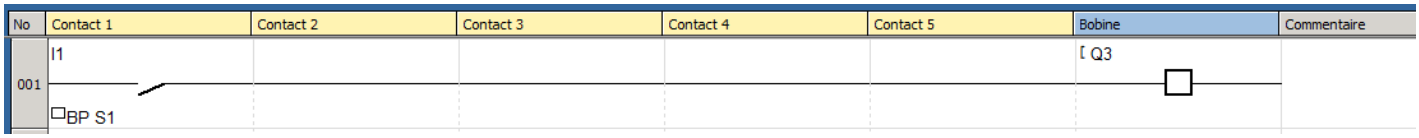
	Status	Symbol	Address	Data type	Comment
1		Count_Control	EW 1	WORD	
2		Example_Input_1	E 1.1	BOOL	Input
3		Example_Input_2	EW 2	WORD	Input range
4		Example_Output_1	A 1.1	BOOL	Output
5		Example_Output_2	AW 2	WORD	Output range
6		Input_Control	E 0.2	BOOL	
7		Input_Motor	E 0.0	BOOL	
8		Input_Temperature	E 0.1	BOOL	
9		Output_switch	A 0.0	BOOL	
10		Temperature	AW 1	INT	
11					

5.2. Raccordement des entrées/sortiesTOR d'un API

Exemple de raccordement d'un bouton poussoir S1 sur l'entrée 1 (I1) en 24 VDC et d'un contacteur KMC en 24VAC sur la sortie 2 (O2), en utilisant un automate alimenté en 230V AC et disposant de sa propre alimentation interne.



Exemple de programmation en LD :



Exemple de tableau mnémonique :

ENTREES		
Adresse	Mnémonique	Commentaire
I1	S1	Bouton poussoir Marche
SORTIES		
Adresse	Mnémonique	Commentaire
O2	KMC	Contacteur moteur convoyeur

5.3. Raccordement des entrées/sorties ANALOGIQUES d'un API

Exemple de raccordement d'un capteur de température 0-10V sur l'entrée analogique 5 (I5) en 24 VDC et d'un variateur de lumière en 24VDC sur la sortie 7 (O7).