CHAPITRE IV : LES AUTOMATES PROGRAMMABLES INDUSTRIELS

I – INTRODUCTION

1- Définition et Historique des A.P.I.:

Un Automate Programmable Industriel est une machine électronique, programmable par un personnel non informaticien et destinée à piloter en ambiance industrielle et en temps réel des procédés automatiques.

Les automates programmables industriels ou **A.P.I.** comme on les appelle le plus souvent ou encore Programmable Logic Controller **(PLC** en anglais**)**, sont apparut aux Etats-Unis vers 1969 où ils répondaient aux désirs des industries de l'automobile de développer des chaînes de fabrication automatisées qui pourraient suivre l'évolution des technologies et des modèles fabriqués. L'A.P.I. s'est ainsi substitué aux armoires à relais en raison de sa souplesse, mais aussi parce que dans les automatismes de commande complexe, les coûts de câblage et de mise au point devenaient trop élevés.

Les premiers constructeurs américains étaient les entreprises ***Modicon*** et ***Allen-Bradley***.

Les A.P.I. offrent de nombreux avantages par rapport aux dispositifs de commande câblés, comme :

- La fiabilité.
- La simplicité de mise en œuvre *(pas de langage de programmation complexe)* .
- La souplesse d'adaptation *(système évolutif et modulaire)* .
- La maintenance et le dépannage possible par des techniciens de formation électromécanique.
- L'Intégration dans un système de production (*implantation aisée).*

Les A.P.I. ont subit des améliorations tous les 4 à 7 ans au fur et à mesure de l'apparition des composants électroniques tels que les microprocesseurs et les microcontrôleurs.

2- Domaines d'emploi des automates :

On utilise les API dans tous les secteurs industriels pour la *commande des machines* (convoyage, emballage...) ou des *chaînes de production* (automobile, agroalimentaire ...) ou il peut également assurer des fonctions de *régulation de processus* (métallurgie, chimie ...).

Il est de plus en plus utilisé dans le domaine du *bâtiment* (tertiaire et industriel) pour le contrôle du chauffage, de l'éclairage, de la sécurité ou des alarmes.

3- Nature des informations traitées par l'automate :

Les informations traitées par un API peuvent être de type :

- Tout ou rien (T.O.R.) ou logique : l'information ne peut prendre que deux états (0 ou 1 …). C'est le type d'information délivrée par un détecteur, un bouton poussoir …
- Analogique : l'information est continue et peut prendre une valeur comprise dans une plage bien déterminée. C'est le type d'information délivrée par un capteur (pression, température …)
- Numérique : l'information est contenue dans des mots codés sous forme binaire. C'est le type d'information délivrée par un ordinateur ou un module intelligent.

II – Architecture des A.P.I.

1- Aspect extérieur :

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

Les automates type *compact ou micro automates* intègrent le processeur, l'alimentation, les interfaces d'entrées / sorties. Selon les modèles et les fabricants, ils peuvent réaliser certaines fonctions supplémentaires (comptage rapide, E/S analogiques ...) et recevoir des extensions en nombre limité.

Exemples : LOGO de Siemens, ZELIO de Schneider, S7-200 de Siemens...

Ces automates sont de fonctionnement simple et sont généralement destinés à la commande de petits automatismes.

Pour les automates type *modulaire*, le processeur, l'alimentation et les interfaces d'entrées / sorties résident dans des unités séparées (**modules**) et sont fixées sur un ou plusieurs **racks** contenant le "fond de panier" (bus plus connecteurs).

Ces automates sont intégrés dans les automatismes complexes où puissance, capacité de traitement et flexibilité sont nécessaires.

Figure 1 : Automate modulaire (Modicon)

Figure 2 : Automate compact (LOGO)

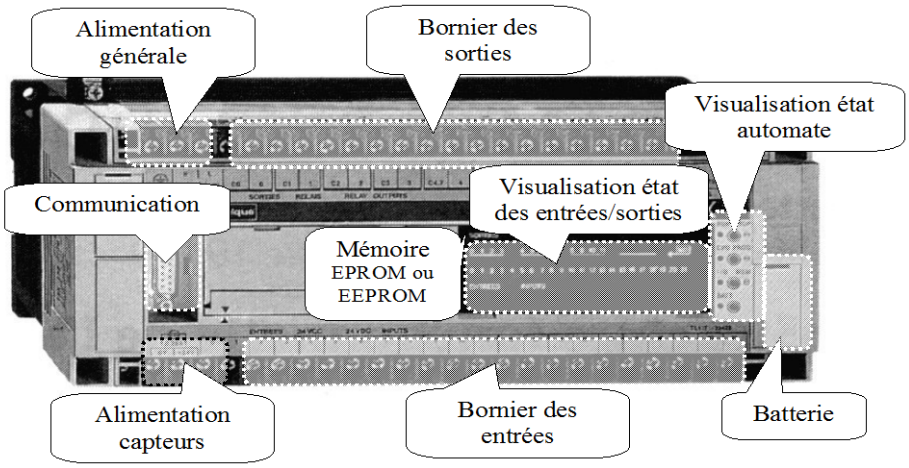


Figure 3 : Structure générale d'un A.P.I

2- Structure d'un système complet

La figure 4 représente un A.P.I. avec divers périphériques et auxiliaires qui représentent son environnement.

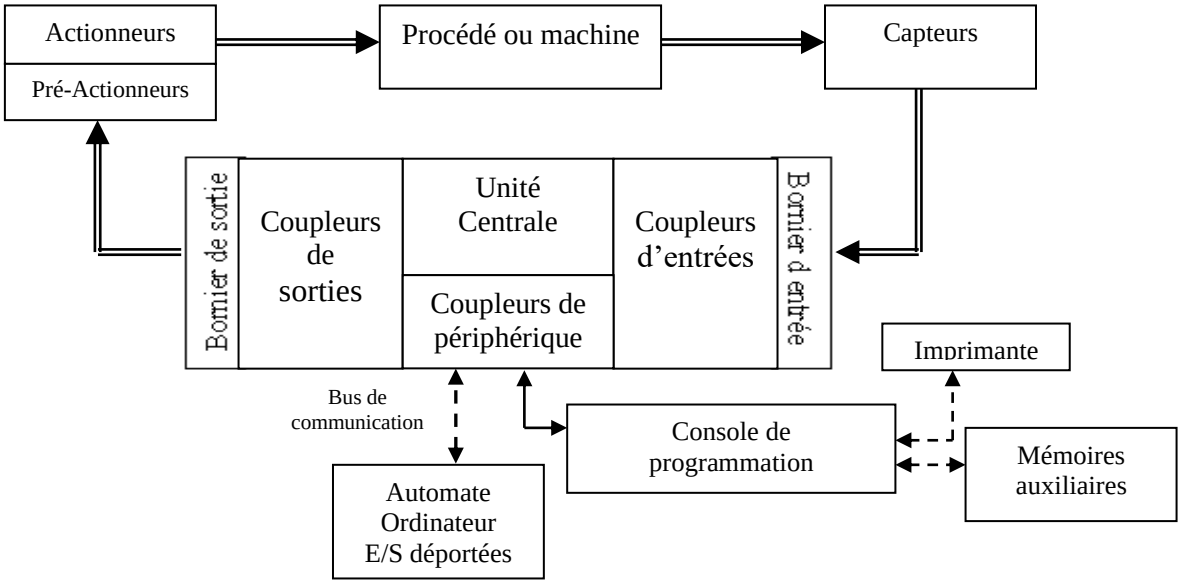


Figure 4 : L'A.P.I et son environnement(= liaisons permanentes ; - temporaire ; --- éventuelle)

3- Structure interne de l'automate

L'A.P.I. est constitué principalement de trois parties (voir figures 4 et 5) :

- ***Une unité centrale*** qui est le cerveau qui se trouve derrière toute prise de décision logique.

- ***Des coupleurs d'entrées/sorties*** qui assurent la liaison entre l'unité centrale et le monde extérieur (capteurs, préactionneurs, etc…)
- ***Des coupleurs de périphériques***.

Ces éléments communiquent par un bus appelé ***Bus d'entrées/sorties.***

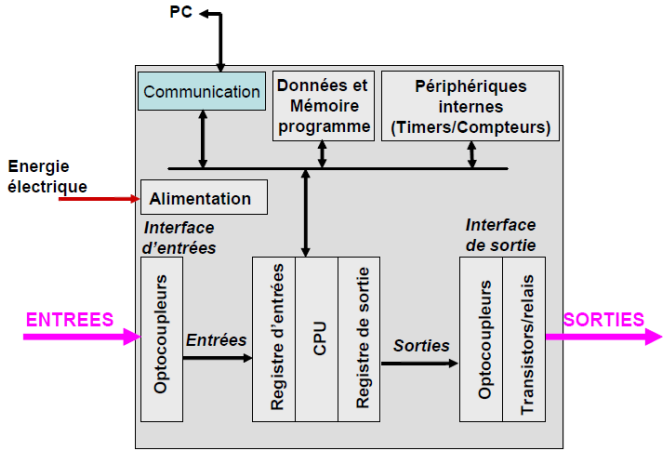


Figure 5 : Structure interne d'un API

3-1- <u>L'unité centrale ou *UC*</u>

L'UC est caractérisé par son processeur et sa mémoire centrale (voir figure 6).

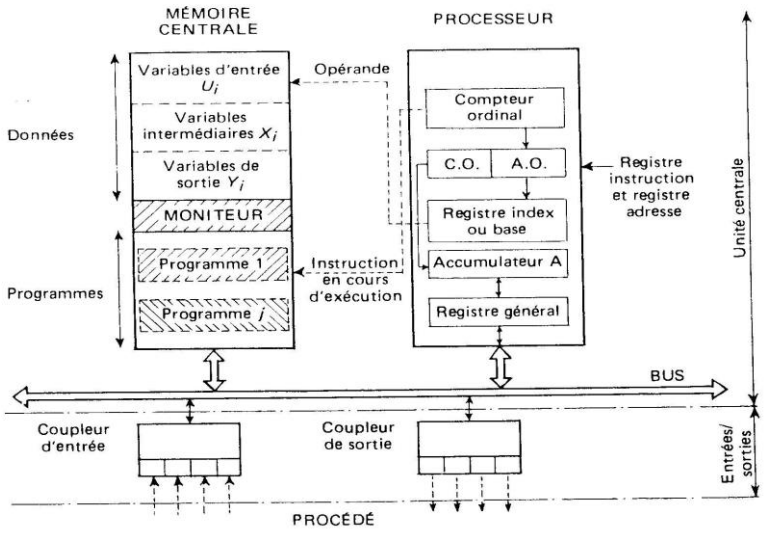


Figure 6 : Structure de l'UC d'un API

Le rôle de l'UC est d'exécuter les programmes qui se trouvent dans la mémoire.

a- <u>Le processeur</u> : Il gère le fonctionnement de l'automate programmable et exécute les instructions du programme au rythme de son horloge. Il réalise toutes les fonctions logiques, arithmétiques et de traitement numérique (transfert, comptage, temporisation ...).

Les processeurs sont actuellement réalisés par des microprocesseurs. Au début de l'apparition des API, les processeurs étaient des processeurs câblés (réalisés par des circuits logiques séparés) ou des processeurs microprogrammés (qui utilisait la logique programmable).

b- <u>la mémoire centrale</u>

La mémoire d'un automate comporte une zone programme (ou mémoire programmes ou utilisateurs), une zone données (ou mémoire données) et une zone réservée au moniteur ou système d'exploitation qui est le logiciel de gestion de l'A.P.I. (voir figure 6).

- la mémoire programme contient les programmes utilisateurs (à exécuter). Elle est en technologie RAM sauvegardée par pile ou batterie ou en EPROM ou actuellement en EEPROM.

- La mémoire données est organisée de façon spécialisée, elle est câblée directement sur les coupleurs d'entrées/sorties (elle contient les états des capteurs et pré-actionneurs). Elle est en technologie RAM obligatoirement et généralement sauvegardée par pile ou batterie.

3-2- Les coupleurs d'entrées/sorties

Les coupleurs d'entrées/sorties, appelés aussi interfaces d'entrées/sorties, sont des cartes électroniques qui assurent la liaison entre l'UC de l'automate programmable et la Partie Opérative (le processus via les capteurs et les pré-actionneurs).

Les coupleurs d'entrées reçoivent les signaux des capteurs et des commandes de l'opérateur (via le **pupitre de commande)** qu'ils traitent pour les rendre compatibles avec les caractéristiques internes de l'A.P.I.

Les coupleurs de sorties reçoivent les signaux de l'UC. Ils les amplifient et les rendent compatibles avec les pré-actionneurs commandés et les contrôles du **pupitre de commande**.

Pour la plupart des A.P.I., les coupleurs sont groupés avec l'UC, mais sont modulaires par carte ou par rack. Le coupleur accède d'une part au bus d'E/S, d'autre part au bornier qui se trouve le plus souvent sur la face avant de l'automate. Certains constructeurs proposent des E/S décentralisées (cartes d'entrées / sorties déportées).

Les A.P.I. offrent une grande variété d'E/S comme :

- E/S T.O.R. (Tout Ou Rien c.à.d binaires ou digital).
- E/S sur mot ou E/S numériques.
- E/S spéciales (modules intelligents) comme les E/S analogiques, les commandes d'axes, les cartes de comptage rapide et les cartes de régulation P.I.D…..

a- les interfaces d'entrées binaires (T.O.R)

Les informations logiques peuvent être de différente nature :

Alternatif ou Continu } en tension ou encourant → divers tensions de 12v à 220v ; divers courants de 1mA à 1A

Le coupleur est alors chargé de rendre compatible ces divers signaux, pour qu'ils soient calibrés en tension et en courant à la logique de l'automate (TTL, CMOS, …).

Le schéma de principe d'une entrée binaire est donné sur la figure 7.

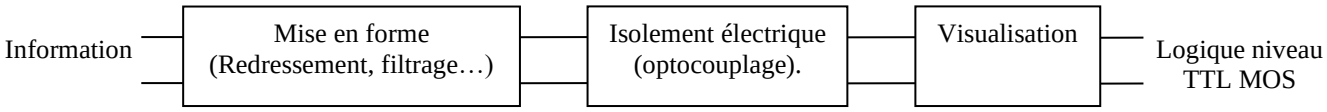


Figure 7 : schéma de principe d'une entrée binaire

La figure 8 donne un exemple du schéma d'une entrée binaire :

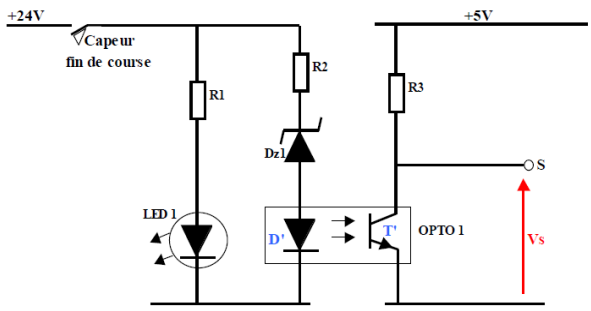


Figure 8 : Exemple du schéma d'une entrée binaire

b- les interfaces de sorties binaires

Le problème est le même que pour les interfaces d'entrées, mais dans le sens inverse. Le schéma de principe d'une sortie binaire est donné sur la figure 9.

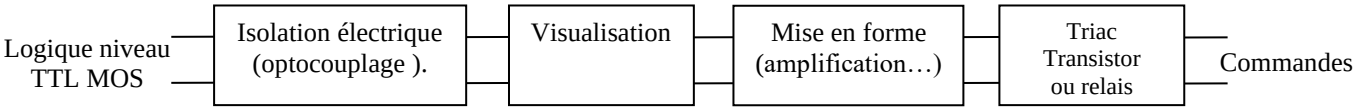


Figure 9 : schéma de principe d'une sortie binaire

La figure 10 donne un exemple du schéma d'une sortie binaire à relais:

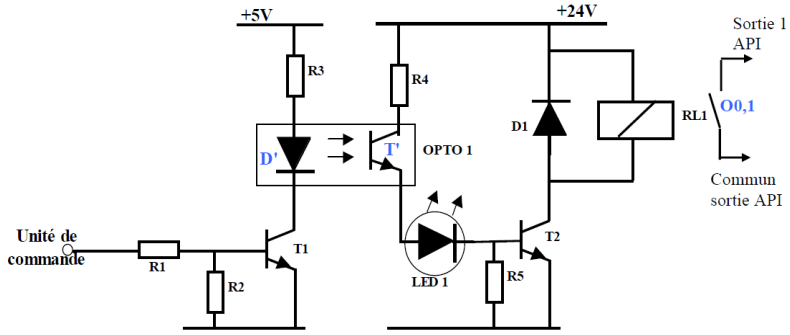


Figure 10 : Exemple du schéma d'une sortie binaire

Remarque : La capacité d'interfaçage est un critère de choix important d'un A.P.I., ainsi que son nombre d'E/S.

3-3- <u>Les périphériques et leurs coupleurs</u>

Ils sont divers. Les principaux sont les suivants :

☛ Mémoire de masse

☛ Imprimantes

Ces périphériques sont en général reliés sur la console de programmation et rarement directement sur l'automate. Leurs interfaces sont classiques en informatique, ils gèrent une transmission soit série soit parallèle de type :

- RS 232C
- USB

☛ Autres automates (superviseurs)

Leurs liaisons sont de deux types :

- Soit par des ports d'E/S de l'automate.
- Soit par une interface spécialisée appelée module ou cartes de communication (Ethernet ...) assurant une transmission normalisée à travers des ***bus de terrain***.

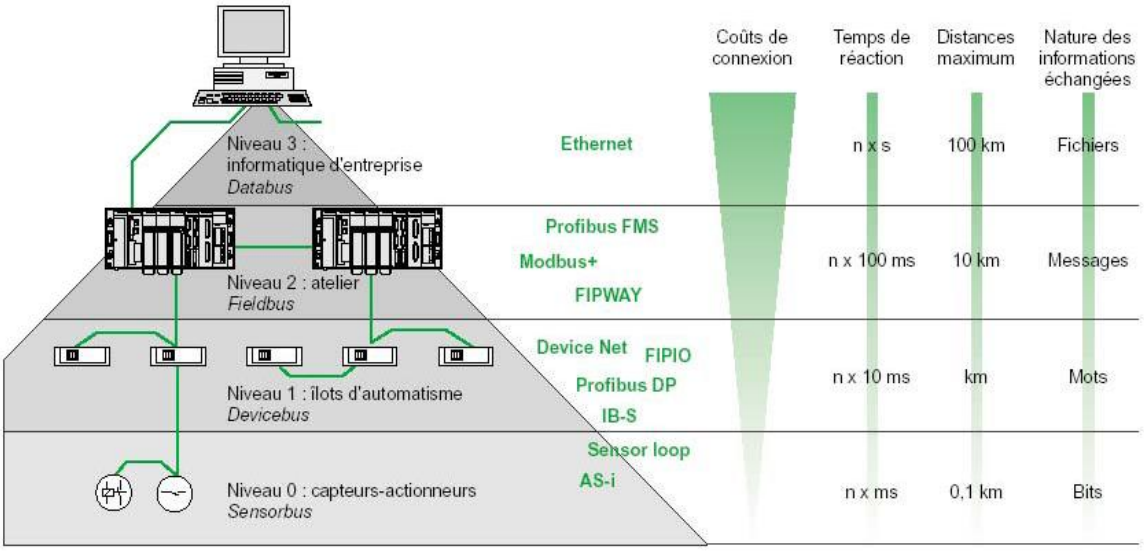


Figure 11: Pyramide CIM (*Computerized Integration Manufacturing)*

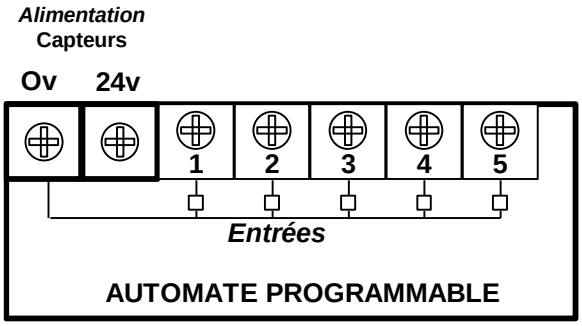
3-4- <u>Câblage des entrées / sorties d'un automate :</u>

- **Branchement des Entrées TOR**

L'alimentation électrique pour les capteurs /détecteurs peut être fournie par l'automate (en général 24V continu) ou par une source extérieure. L'automate dispose généralement d'un commun à toutes les entrées.

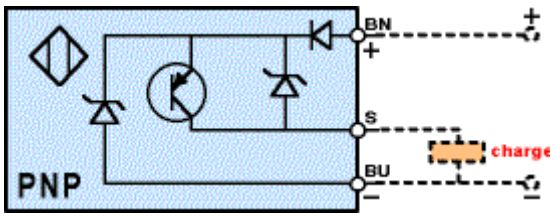
Il est possible de câbler à l'automate, des contacts classiques, mais aussi des capteurs électroniques deux ou trois fils.

Le commun des entrées doit être relié au 0V

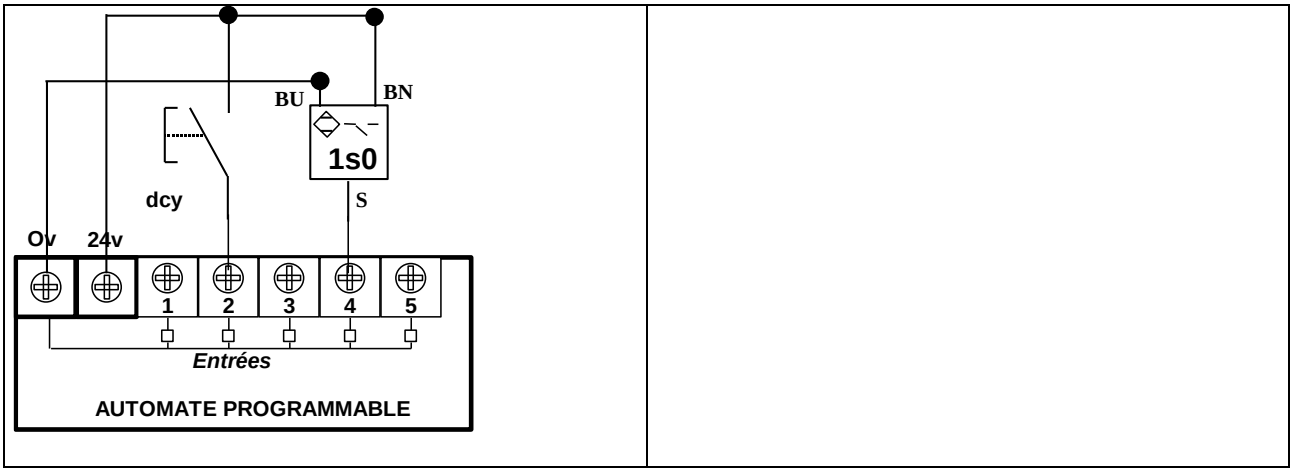


EX : l'API TWIDO fonctionne en logique positive donc pour mettre une entrée automate au 1 logique, il faut lui imposer un potentiel de +24 Volts.

Les détecteurs 3 fils ou détecteurs électronique de type PNP **:**



Lorsque qu'il y a détection, le transistor est passant (contact fermé). Il va donc imposer le potentiel + sur la sortie **S**. La charge est branchée entre la sortie **S** et le potentiel - . Ce type de détecteur doit être polarisé comme c'est montré sur la figure ci- contre.



➢ **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 (voir figure 12).

L'alimentation électrique est fournie par une source extérieure à l'automate programmable.
Les sorties comportent généralement un commun pour 3 ou 4 sorties. Cela permet d'alimenter des pré-actionneurs de tensions différentes, sans pour autant effectuer un relayage. Les sorties de l'automate peuvent être à Triac, à Transistor ou à relais.

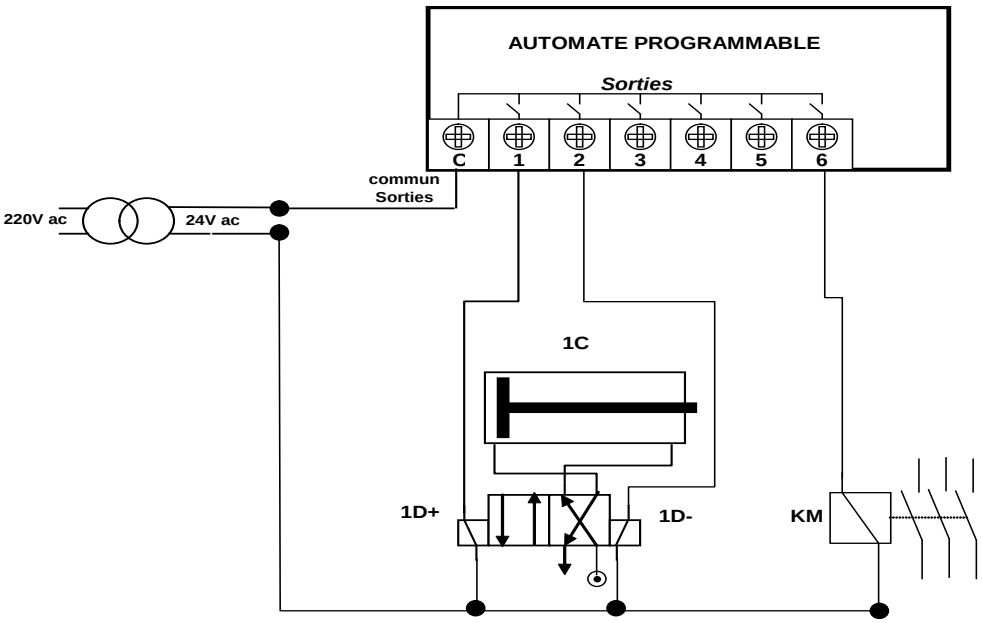


Figure 12: exemple de câblage des sorties sur un API

4 - <u>La console de programmation</u>

La console de programmation est l'outil privilégié de la communication ' Homme–Machine' pour le développement, la mise au point et, éventuellement, l'exploitation des applications.

Le premier rôle de la console est de transformer le langage de programmation en instructions exécutables par l'automate.

Elle permet lors de la mise au point du programme :

- La simulation pas à pas.
- La détection d'erreur de syntaxe.
- L'introduction, la correction et la modification des programmes (Editeur de textes).

Elle permet sur beaucoup de systèmes la programmation définitive sur EPROM.

Pour assurer ces différentes tâches, la console de programmation peut être connectée à l'A.P.I. (fonctionnement en ligne) ou non (fonctionnement autonome).

La console possède son propre processeur et sa propre mémoire, c'est l'équivalent d'un micro-ordinateur. La plupart des consoles actuelles sont d'ailleurs des micro-ordinateurs de commerce (PC équipés du logiciel constructeur spécifique).

III – Principe de fonctionnement d'un A.P.I :Le cycle d'un A.P.I.

Lorsque l'on a affaire à un ordinateur l'exécution du programme se fait en général ligne par ligne et d'une façon asynchrone.

Une des caractéristiques de l'automate est de fonctionner différemment c.à.d de façon cyclique. En effet avant d'exécuter quoi que ce soit, l'automate lit entièrement son programme ; et une fois l'exécution terminée recommence les mêmes opérations.

On définit alors la notion de cycle et de temps de cycle (entre 1ms et 30ms environ).

Il existe plusieurs types de cycle mais le plus répondu est le celui représenté sur la figure 13.
Ce cycle comprend 5 phases :

- **Phase 1** : Lecture ou Acquisition des entrées: Prise en compte des informations des modules d'entrées et écriture de leur valeur dans RAM (zone DONNEE).
- **Phase 2** : Exécution de le programme ou Traitement des données **:** Lecture du programme (située dans la RAM programme) par l'unité de traitement, lecture des variables (RAM données), traitement et écriture des variables (internes, sorties …) dans la RAM données.
- **Phase 3 :** Traitement de toute demande de communication
- **Phase 4 :** Exécution du test d'auto--diagnostic (Gestion du système Autocontrôle de l'automate)
- **Phase 5** : Ecriture des sorties : Lecture des variables de sorties dans la RAM données et transfert vers le module de sorties.

Le temps de scrutation de chaque cycle est vérifié par un temporisateur appelé ***Watchdog*** *(chien de garde)* qui enclenche une procédure d'alarme en cas de dépassement de celui-ci (réglé par l'utilisateur).

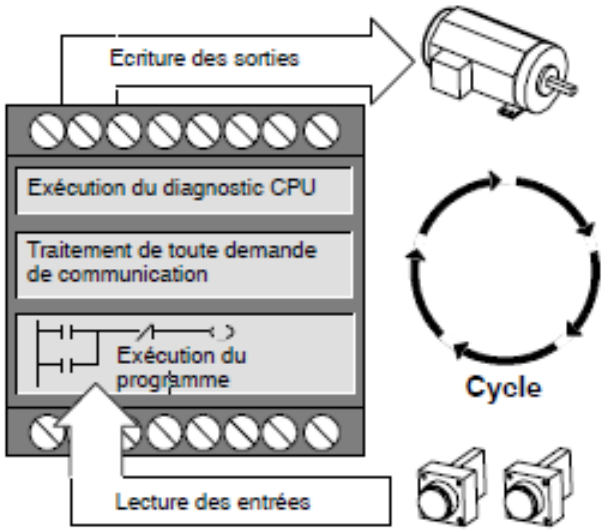


Figure 13 : Cycle typique d'exécution des programmes d'un A.P.I.

IV– Le langage des A.P.I.

1- Introduction

Les langages des A.P.I. sont des langages intermédiaires entre le langage évolué et le langage machine. Ils ont l'avantage d'avoir un jeu d'instructions incluant uniquement les fonctions logiques, cela a comme

conséquences, une meilleure compréhension par les automaticiens et une simplification du compilateur de la console de programmation et du logiciel constructeur.

2- Les divers types de langages

Malheureusement il n'y a pas eu d'unicité chez les constructeurs quant au langage de programmation. Néanmoins 4 langages sont parmi les plus utilisés (norme CEI 61131-3):

- Le langage LADDER (*LD : Ladder diagram*)
- Le langage booléen (*FBD : Function Bloc Diagram*)
- Le langage GRAFCET (*SFC : Sequential Function Chart*)
- Le langage mnémonique (*IL : Instruction list*)

2-1- variables traitées par un automate : Réferences

Les APIs traitent plusieurs types de variables et utilisent des adresses spécifiques ou références pour chacune d'elles :

- variables bit ou (T.O.R.) sous forme d'entrées, sorties ou bit internes appelé mémento par certain APIs.
- variables analogique : sous forme d'entrées, sorties ou internes
- variables Numérique: sous forme d'octet, de mots, double mots ou mots flottants.

Exemples d'adressage:

- Références de l'automate LOGO! : LOGO! met à la disposition du programmeur (version maxi):

• 24 entrées TOR référencées de **I_1** à **I_{24}** • 16 sorties TOR référencées de **Q_1** à **Q_{16}** • 24 mémentos numériques référencés de **M_1** à **M_{24}** avec le **M_8** est le mémento de démarrage • 8 bits de registre à décalage référencés de **S_1** à **S_8**	• 8 entrées analogiques référencées de **AI_1** à **AI_8** • 2 sorties analogiques référencées de **AQ1** à **AQ2** • 6 mémentos analogiques référencés de **AM_1** à **AM_6**

- ***Pour L'API Twido*** :

Liste des bits opérandes : Le tableau suivant décrit quelques objets bits qui sont utilisés comme opérandes dans des instructions booléennes.

Type	Description	Repère ou valeur	Nombre maximal	Accès en écriture (1)
Valeurs immédiates	0 ou 1 (False ou True)	0 ou 1	-	-
Entrées Sorties	Ces bits sont les "images logiques" des états électriques des E/S. Ils sont stockés dans la mémoire de données et sont mis à jour à chaque scrutation de la logique du programme.	%Ix.y.z (2) %Qx.y.z (2)	Remarque (4)	Non Oui
AS-Interface Entrées Sorties	Ces bits sont les "images logiques" des états électriques des E/S. Ils sont stockés dans la mémoire de données et sont mis à jour à chaque scrutation de la logique du programme.	%IAx.y.z %QAx.y.z	Remarque (5)	Non Oui
Interne (mémoire)	Les bits internes sont des zones de mémoire internes utilisées pour stocker des valeurs intermédiaires lorsqu'un programme est en cours d'exécution. **Remarque :** Les bits d'E/S non utilisés ne peuvent pas être employés comme des bits internes.	%Mi	128 TWDLC•A10DRF, TWDLC•A16DRF 256 Tous les autres automates	Oui
Système	Les bits système %S0 à %S127 surveillent le bon fonctionnement de l'automate ainsi que la bonne exécution du programme de l'application.	%Si	128	Selon i

Type	Description	Repère ou valeur	Nombre maximal	Accès en écriture (1)
Blocs fonction	Les bits des blocs fonction correspondent aux sorties des blocs fonction. Ces sorties peuvent être directement câblées ou exploitées en tant qu'objet.	%TMi.Q, %Ci.P, etc.	Remarque (4)	Non (3)
Blocs fonction réversibles	Blocs fonction programmés à l'aide d'instructions de programmation réversible BLK, OUT_BLK et END_BLK.	E, D, F, Q, TH0, TH1	Remarque (4)	Non
Extraits de mots	Pour certains mots, un des 16 bits est extrait en tant que bit opérande.	Variable	Variable	Variable
Etapes Grafcet	Les bits %X1 à %Xi sont associés aux étapes Grafcet. Le bit étape Xi est à l'état 1 lorsque l'étape correspondante est active et à l'état 0 lorsqu'elle est désactivée.	%X21	62 TWDLC•A10DRF, TWDLC•A16 DRF 96 TWDLC•A24DRF, TWDLCA•40DRF et automates modulaires	Oui

- l'adressage des entrées/sorties se conformes à la syntaxe suivante :

%	I, Q	x	.	y	.	z
Symbole	Type d'objet	Position de l'automate	point	Type d'E/S	point	Numéro de voie

Groupe	Elément	Valeur	Description
Symbole	%	-	Un repère interne doit toujours débuter par un symbole de pourcentage (%).
Type d'objet	I	-	Entrée. « Image logique » de l'état électrique de l'entrée d'un automate ou d'un module d'E/S d'expansion.
	Q	-	Sortie. « Image logique » de l'état électrique de la sortie d'un automate ou d'un module d'E/S d'expansion.
Position de l'automate	x	0 1 - 7	Automate maître (maître de liaison distante). Automate distant (esclave de liaison distante).
Type d'E/S	y	0 1 - 7	E/S de base (E/S locale sur un automate). Modules d'E/S d'expansion.
Numéro de voie	z	0 - 31	Numéro de la voie d'E/S sur l'automate ou le module d'E/S d'expansion. Le nombre de points d'E/S disponibles dépend du modèle de l'automate ou du type du module d'E/S d'expansion.

Entrées : % Ix.y.z
Sorties : % Ox.y.z
Exemple : % I0.0.5 : % I0.5
% O0.3.4 : % O3.4

2-2- **Le langage LADDER** (*LD : Ladder diagram*)

Appelé aussi langage à contact, langage à relais ou réseau en échelle, il a été développé par les américains en pensant qu'il semblerait plus familier aux automaticiens.

Ce langage utilise les symboles graphiques tels que : contacts, relais, bobine et blocs fonctionnels et s'organise en réseaux (labels). C'est le plus utilisé.

Nom	Elément graphique	Fonction
Contact à ouverture	⊣ ⊢	Contact passant lorsque l'objet bit de contrôle se trouve à l'état 1.
Contact à fermeture	⊣/⊢	Contact passant lorsque l'objet bit de contrôle se trouve à l'état 0.
Contact de détection d'un front montant	⊣P⊢	Front montant : détecte le passage de 0 à 1 de l'objet bit de contrôle.
Contact de détection d'un front descendant	⊣N⊢	Front descendant : détecte le passage de 1 à 0 de l'objet bit de contrôle.

Nom	Elément graphique	Fonction
Bobine directe	⊣( )⊢	L'objet bit associé prend la valeur du résultat de la zone de test.
Bobine inverse	⊣(/)⊢	L'objet bit associé prend la valeur du résultat inverse de la zone de test.
Bobine d'enclenchement	⊣(S)⊢	L'objet bit associé est réglé sur 1 lorsque le résultat de la zone de test est 1.
Bobine de déclenchement	⊣(R)⊢	L'objet bit associé est réglé sur 0 lorsque le résultat de la zone de test est 1.

Nom	Elément graphique	Fonction
Connexion horizontale	———	Relie en série les éléments graphiques de test et d'action entre les deux barres verticales.
Connexion verticale	|	Relie les éléments graphiques de test et d'action en parallèle.

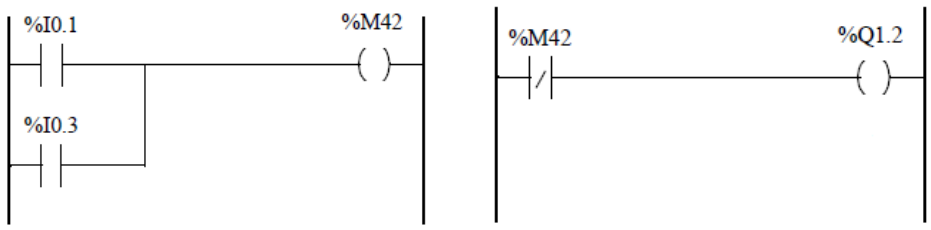
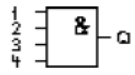
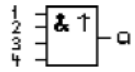
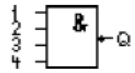
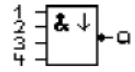
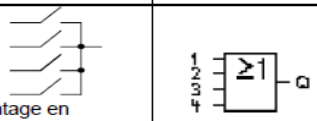
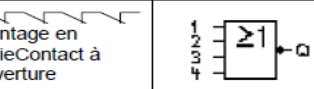
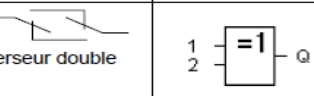
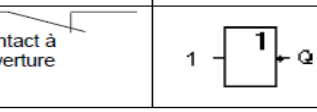


Figure 14 : Exemples de programmation LADDER

2-3- **Le langage Booléen** (*FBD : Function Bloc Diagram*)

Ce langage utilise les symboles du logigramme. Il peut être facilement traduit en langage machine.

Représentation dans le schéma des connexions	Représentation dans LOGO!	Désignation de la fonction de base
Montage en série Contact à fermeture		AND (ET)
		AND avec évaluation de front
Montage en parallèle Contact à ouverture		NAND (non ET)
		NAND avec évaluation de front

Représentation dans le schéma des connexions	Représentation dans LOGO!	Désignation de la fonction de base
Montage en parallèle Contact à fermeture		OR (OU)
Montage en sérieContact à ouverture		NOR (non OU)
Inverseur double		XOR (OU exclusif)
Contact à ouverture		NOT (négation, inverseur)

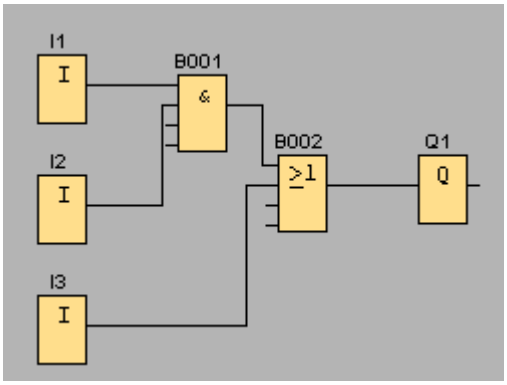


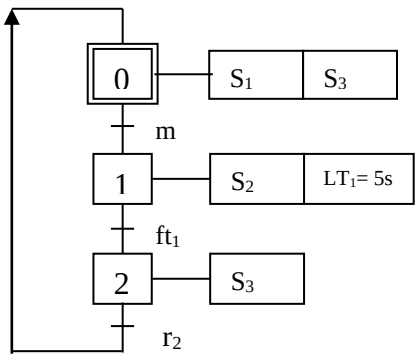
Figure 15: Exemple de programmation Booléenne : symboles des logigrammes relatifs à l'API **LOGO** de Siemens

2-4- <u>Le langage GRAFCET</u> (*SFC : Sequential Function Chart*)

C'est un langage graphique qui permet de tracer directement le schéma Grafcet de l'automatisme considéré. Dans ce cas, pour plus de facilité, on construit un Grafcet niveau 3 qui est le même que le niveau 2, mais les variables d'E/S du niveau 2 sont remplacés par les adresses de l'automate (appelés références).

Exemple: Programmation Grafcet sur l' API Twido de Schneider Electric: (voir fiche Twido)

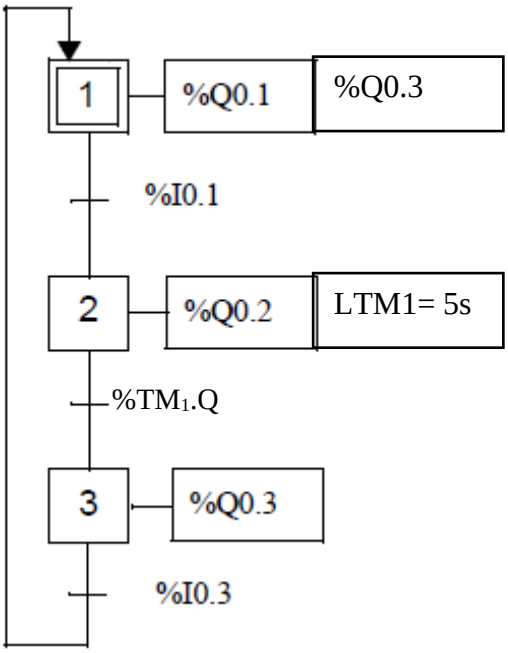
<u>Grafcet niveau2</u> :



<u>Tableau des références</u>

variables	Références
m	%I0.1
r_2	%I0.3
S_1	%Q0.1
S_2	%Q0.2
S_3	%Q0.3
T_1	$\%TM_1$

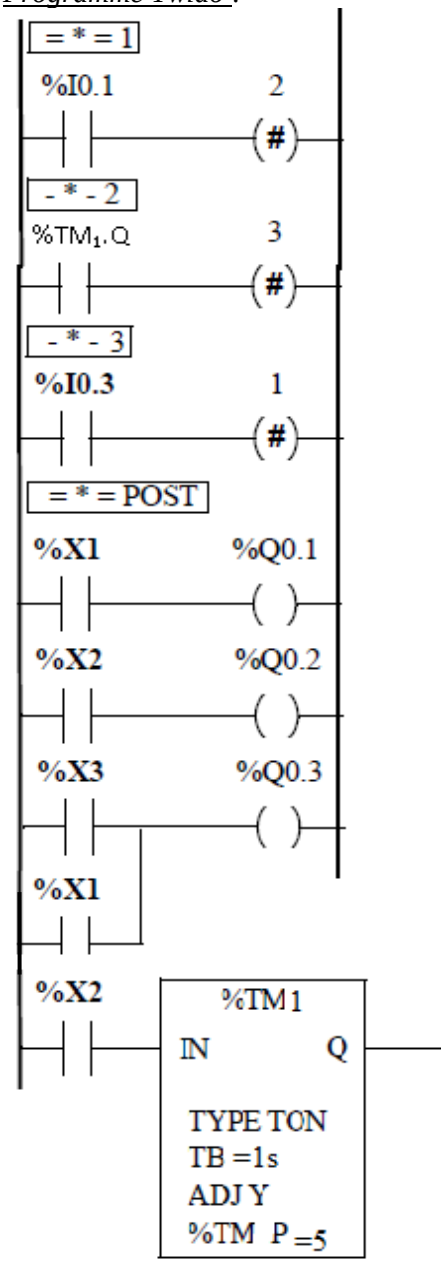
Grafcet niveau3 :



Instructions Grafcet Le tableau suivant répertorie toutes les instructions et les objets requis pour la programmation d'un graphe Grafcet.

Représentation graphique (1)	Transcription en langage TwidoSoft	Fonction
Illustration : Etape initiale Transition Etape	=*= i	Lance l'étape initiale (2).
	# i	Active l'étape i après avoir désactivé l'étape courante.
	-*- i	Lance l'étape i et valide la transition associée (2).
	#	Désactive l'étape courante sans activer d'autre étape.
	#Di	Désactive l'étape i et l'étape courante.
	=*= POST	Lance le traitement postérieur et termine le traitement séquentiel.
	%Xi	Bit associé à l'étape i. Peut être testé et écrit (le nombre maximum d'étapes dépend de l'automate).

Programme Twido :



2-4- <u>Le langage mnémonique</u>(*IL : Instruction list*)

C'est un langage littéral qui utilise le langage d'assemblage, largement utilisé dans le domaine informatique. Très peu utilisé par les automaticiens.

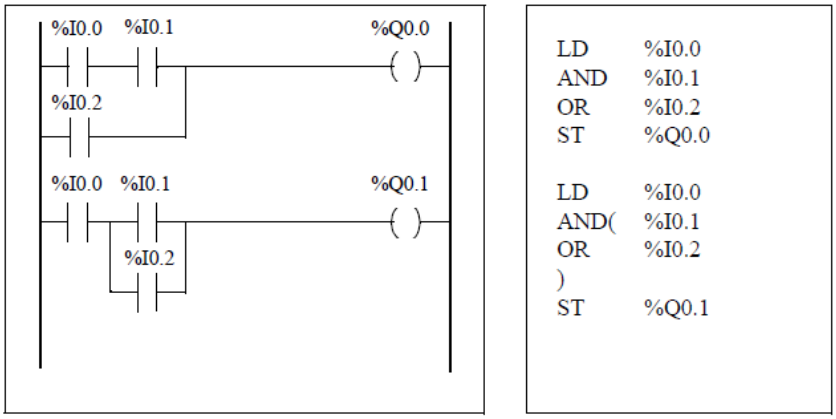


Figure 14 : Exemple de programmation Twido en mnémonique

2-5- Blocs fonction-Blocs opération

En complément aux possibilités de représentation, les langages de programmation permettent l'utilisation de blocs fonctions et de blocs opérations. Ces blocs sont des fonctions préprogrammées, paramétrables, utilisables directement par le programmeur.

Exemple de blocs fonctions : Temporisateur, Compteur.

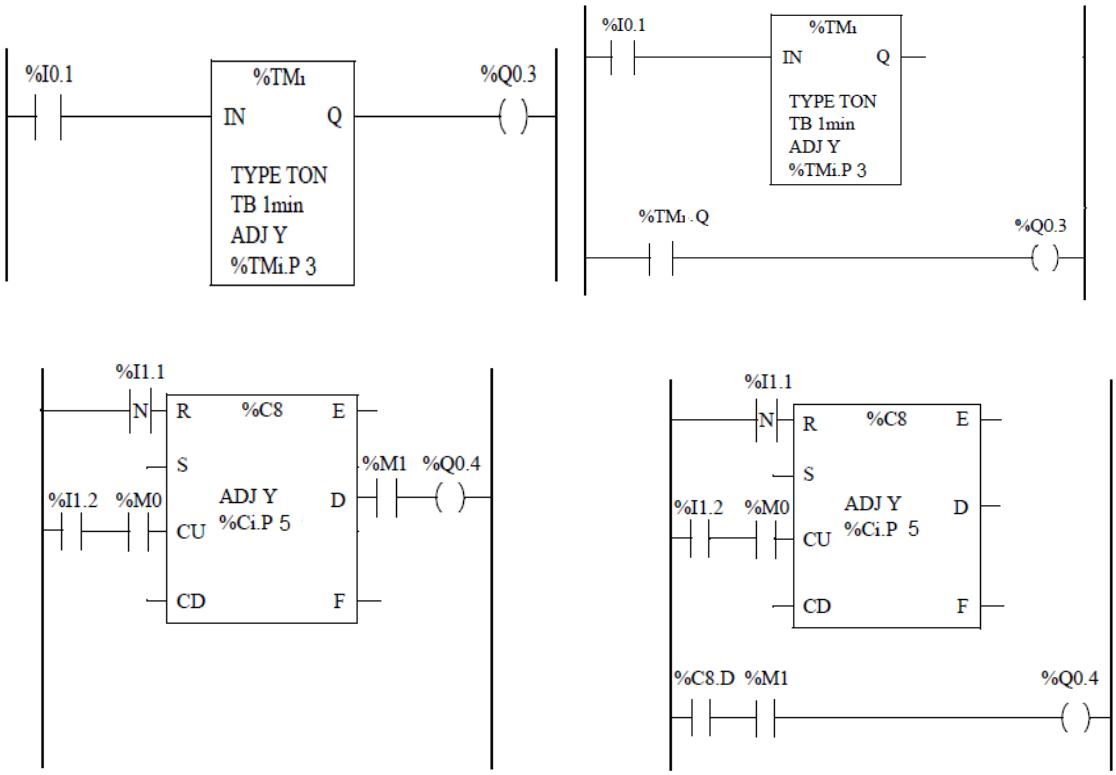
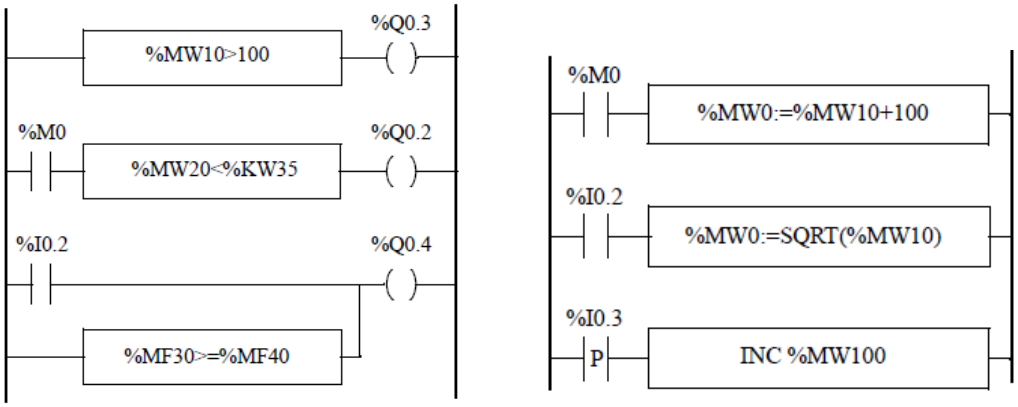


Figure 15 : Exemple de programmation de blocs fonctions

Exemple de blocs opérations : Comparateur, Additionneur.



3- Exemples de programmation :

Programmation de l'A.P.I LOGO ! DE SIEMENS

EXEMPLE DE LA PAGE 47

V – CHOIX D'UN A.P.I.

1- Gamme d'un automate

La multiplication des produits et de leurs caractéristiques nécessite l'introduction d'une classification pour tenter de rapprocher les automates disposant de caractéristiques similaires.

On définit la gamme d'un automate suivant deux notions : quantitative et qualitative.

☛ Classification quantitative (suivant le nombre d'E/S), on a alors :

- Bas de gamme $20 \leq E+S < 100$
- Milieu de gamme $100 \leq E+S < 500$
- Haut de gamme $500 \leq E+S$

20 étant la limite de rentabilité d'un A.P.I.

☛ Classification qualitative (relatif aux instructions de calcul ou de traitement numérique)

- Classe 1 pas de traitement numérique
- Classe 2 traitement numérique simple
- Classe 3 traitement numérique évolué

On définit ainsi deux coefficients :

☛ I_v relative à la gamme

$$I_v = U_b + Y_b + \alpha (U_n + Y_n) + T + C$$

Avec U_b : nombre d'entrées binaires
Y_b : nombre de sorties binaires
U_n : nombre d'entrées numériques
Y_n : nombre de sorties numériques
α : taille du mot de données numériques (en bits)
T : nombre de temporisateurs
C : nombre de compteur

☛ I_c relative à la classe

$$I_c = \sum \text{ des coûts}$$

Avec

Nature du traitement	+ ou -	Comparaison sur mot	Comptage temporisation	* ou /	Transcodage	Moyenne	$\sqrt{}$	Régulation
Coût	1	1	1	2	3	3	4	5

D'où on a :

Bas de gamme	$I_v \leq 300$
Milieu de gamme	$300 < I_v \leq 1000$
Haut de gamme	$I_v > 1000$
Classe 1	$I_c \leq 1$
Classe 2	$1 < I_c \leq 8$
Classe 3	$I_c > 8$

La sélection d'un A.P.I. apte à satisfaire un besoin, dépend de la taille de l'application envisagée (indice I_v) et la complexité des traitements numériques (indice I_c)

2- <u>Critères technologiques</u>

En plus des critères de gamme et du prix, s'ajoute le critère technologique relatif à :

- La technologie de l'unité centrale : la vitesse de traitement et les fonctions spéciales offertes par le processeur permettront le choix dans la gamme souvent très étendue.
- Le langage de programmation : Un automate utilisant des langages de programmation de type GRAFCET est préférable pour assurer les mises au point et dépannages dans les meilleures conditions.
- La nature et la taille de la mémoire
- Caractéristiques des Entrées-Sorties

- 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 (Profibus ...).

VI- Sécurité :

Les systèmes automatisés sont, par nature, source de nombreux dangers (tensions utilisées, déplacements mécaniques, jets de matière sous pression ...).

Placé au cœur du système automatisé, l'automate se doit d'être un élément fiable car :
- un dysfonctionnement de celui-ci pourrait avoir de graves répercussions sur la sécurité des personnes,
- les coûts de réparation de l'outil de production sont généralement très élevés,
- un arrêt de la production peut avoir de lourdes conséquences sur le plan financier.

Aussi, l'automate fait l'objet de nombreuses dispositions pour assurer la sécurité :

- Contraintes extérieures : l'automate est conçu pour supporter les différentes contraintes du monde industriel et à fait l'objet de nombreux tests normalisés (tenue aux vibrations, CEM ...)
- Coupures d'alimentation : l'automate est conçu pour supporter les coupures d'alimentation et permet, par programme, d'assurer un fonctionnement correct lors de la réalimentation (reprises à froid ou à chaud)
- Mode RUN/STOP : Seul un technicien peut mettre en marche ou arrêter un automate et la remise en marche se fait par une procédure d'initialisation (programmée)
- Contrôles cycliques : Procédures d'autocontrôle des mémoires, de l'horloge, de la batterie, de la tension d'alimentation et des entrées / sorties
- Vérification du temps de scrutation à chaque cycle appelée ***Watchdog*** *(chien de garde),* et enclenchement d'une procédure d'alarme en cas de dépassement de celui-ci (réglé par l'utilisateur)
- Visualisation : Les automates offrent un écran de visualisation où l'on peut voir l'évolution des entrées / sorties

La défaillance d'un automate programmable pouvant avoir de graves répercussions en matière de sécurité, les normes interdisent la gestion des arrêts d'urgence par l'automate ; celle-ci doit être réalisée en technologie câblée.

On peut également ajouter des modules de sécurité à l'automate (sécurité des machines).

Il existe enfin des automates dits de sécurité (APIdS) qui intègrent des fonctions de surveillance et de redondance accrues et garantissent la sécurité des matériels.