S9 – AUTOMATISME INFORMATIQUE INDUSTRIELLE

Fascicule 4

Les contraintes industrielles

S93

1) Introduction :

Nous avons étudié un outil de description du fonctionnement d'un système en logique séquentielle : le Grafcet. Cet outil permet une analyse du fonctionnement et place les éléments du système dans le déroulement chronologique du processus en signifiant les interactions et conséquences sur le fonctionnement.

La partie commande chargée de piloter la partie opérative conformément au Grafcet fait l'objet de ce chapitre.

2) Systèmes de commande : l'A.P.I.

2-1) Définition de la partie commande :

Elle est réalisée en logique câblée ou programmée :

- Logique câblée : Un circuit logique est réalisé avec des portes logiques et des bascules.
  - Inconvénient : Les fonctions obtenues restent figées
  - Avantage : l'exécution est très rapide

- Logique programmée : Système réalisé à base de microprocesseur
  - Inconvénient : Investissement initial élevé ; exécution moins rapide
  - Avantage : souplesse, système non figé. Adaptable sur différents systèmes (facilité de connexion)

Le principe repose sur une suite d'instructions ou de programmes chargés dans la mémoire de l'unité de traitement

La partie commande assure trois grandes fonctions :

- La lecture et l'adaptation des signaux d'entrées à la technologie choisie pour le traitement.
- Le traitement (combinatoire ou séquentiel).
- L'adaptation des signaux de sorties et l'envoi d'ordres aux pré-actionneurs.

2-2) L'Automate Programmable Industriel (A.P.I.) :

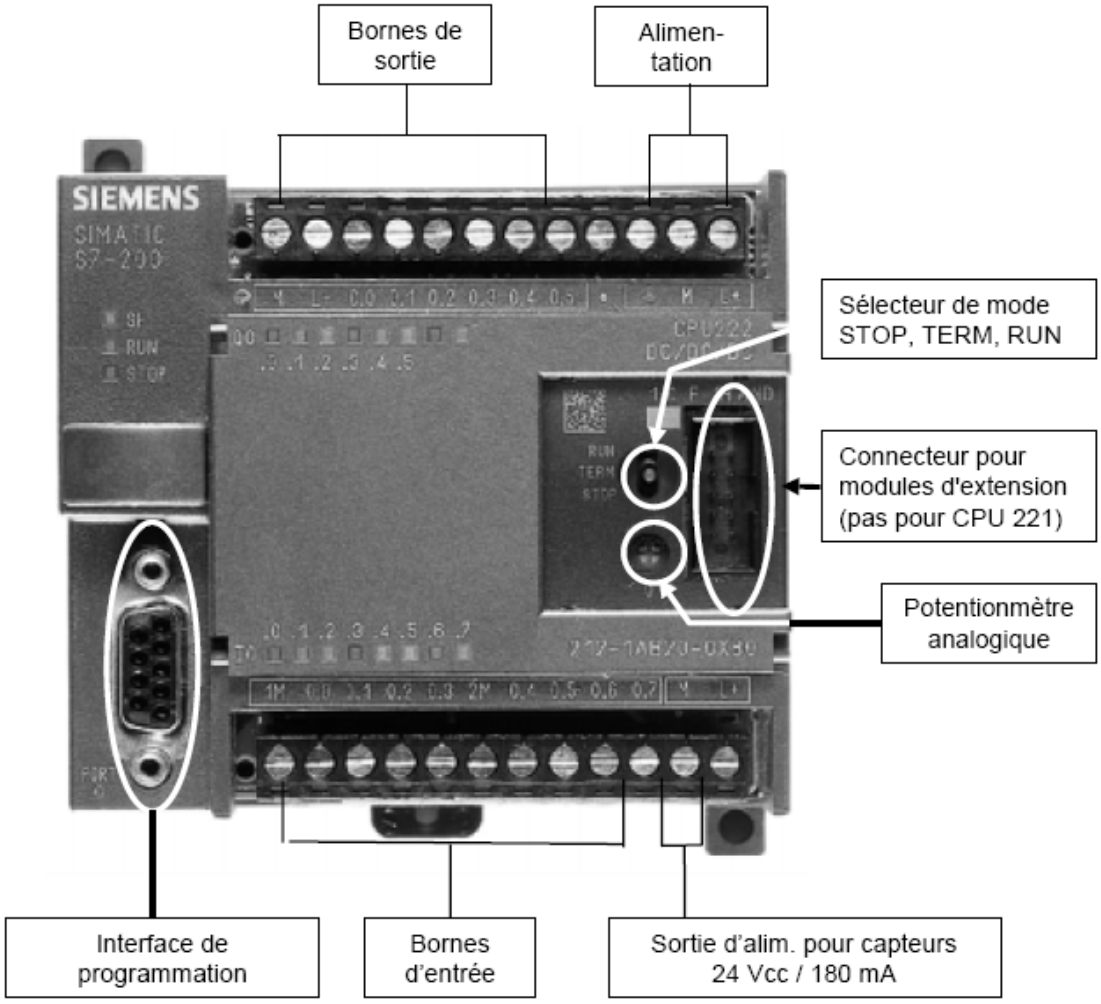
Les API sont apparus dans les années 70 pour remplacer les logiques câblées. Leur évolution rapide a permis de passer du simple fonctionnement en tout ou rien (type logique câblée) en un travail multitâche nécessaire aux systèmes de plus en plus complexes actuels.

Destinés à l'industrie, ils fonctionnent directement sur les sites de production et supportent les conditions sévères de fonctionnement (parasites électromagnétiques, microcoupures, environnement …)

a) Structure générale des A.P.I. :

Les caractéristiques principales d'un automate programmable industriel (**API**) sont :

- Compact ou modulaire
- Tension d'alimentation
- Taille mémoire
- Sauvegarde (EPROM, EEPROM, pile, …)
- Nombre d'entrées / sorties
- Modules complémentaires (analogique, communication,..)
- Langage de programmation



Aspect extérieur d'un automate S7-200 CPU222

Des API en boîtier étanches sont utilisés pour les ambiances difficiles généralement rencontrées en industrie :

- environnement physique et mécanique (poussières, température, humidité, vibrations);
- pollution chimique ;
- perturbation électrique. (parasites électromagnétiques)

b) Architecture de base d'un A.P.I. :

Les API comportent quatre principales parties :

- Une unité de traitement (un processeur CPU);
- Une mémoire ;
- Des modules d'entrées-sorties ;
- Des interfaces d'entrées-sorties ;
- Une alimentation 230 V, 50/60 Hz (AC) - 24 V (DC).

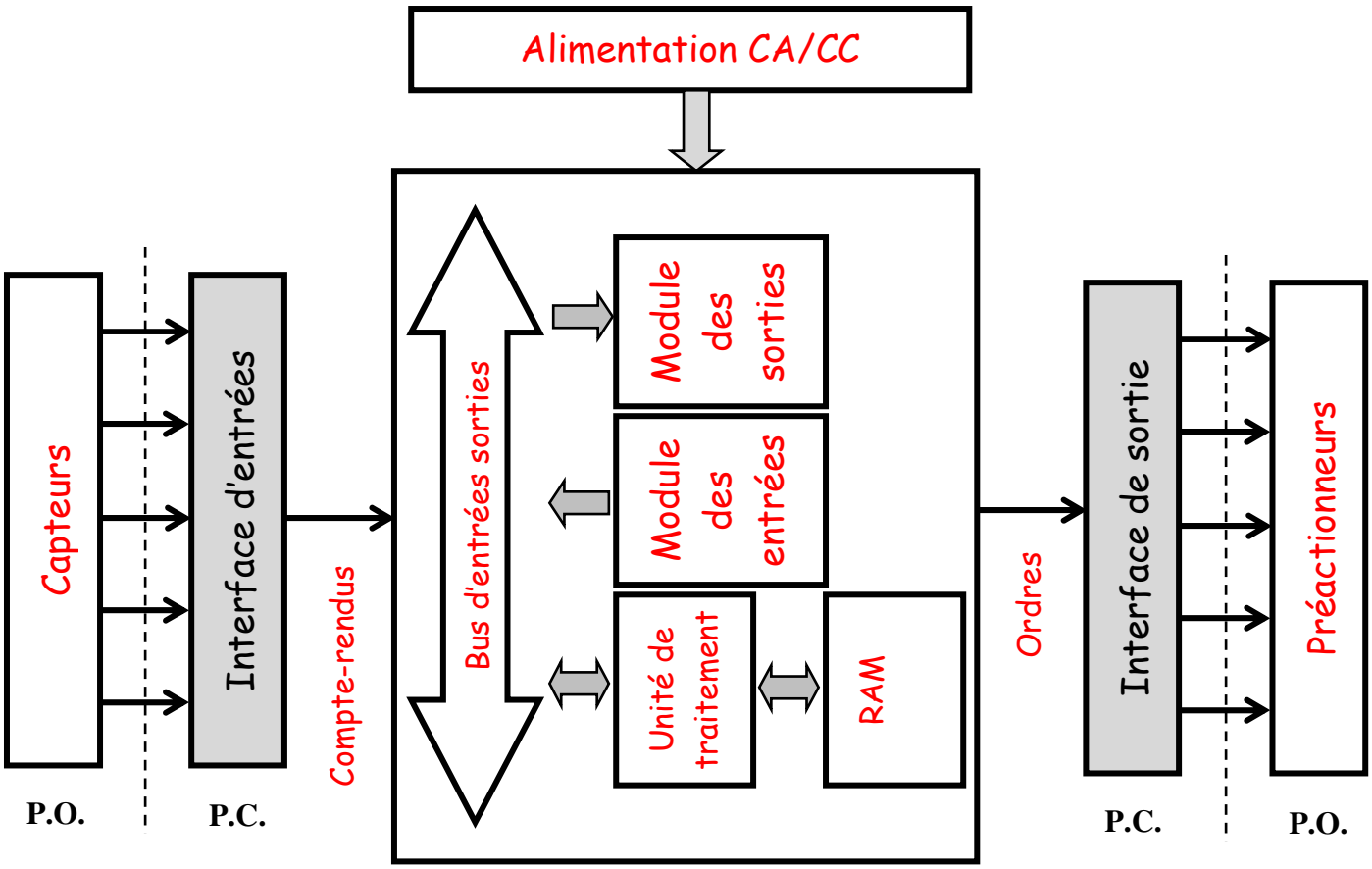
La structure interne d'un automate programmable industriel (API) est assez voisine de celle d'un système informatique simple ; l'unité centrale est le regroupement du processeur et de la mémoire centrale.

Elle commande l'interprétation et l'exécution des instructions programme. Les instructions sont effectuées les unes après les autres, séquencées par une horloge.

Deux types de mémoire cohabitent :

- La mémoire Programme de type ROM (mémoire morte) où est stocké le langage de programmation
- La mémoire de données utilisables en lecture-écriture pendant le fonctionnement de type RAM (mémoire vive). Elle fait partie du système entrées-sorties. Elle fige les valeurs (0 ou 1) présentes sur les lignes d'entrées, à chaque prise en compte cyclique de celle-ci, elle mémorise les valeurs calculées à placer sur les sorties.

L'architecture générale est présentée ci-dessous :

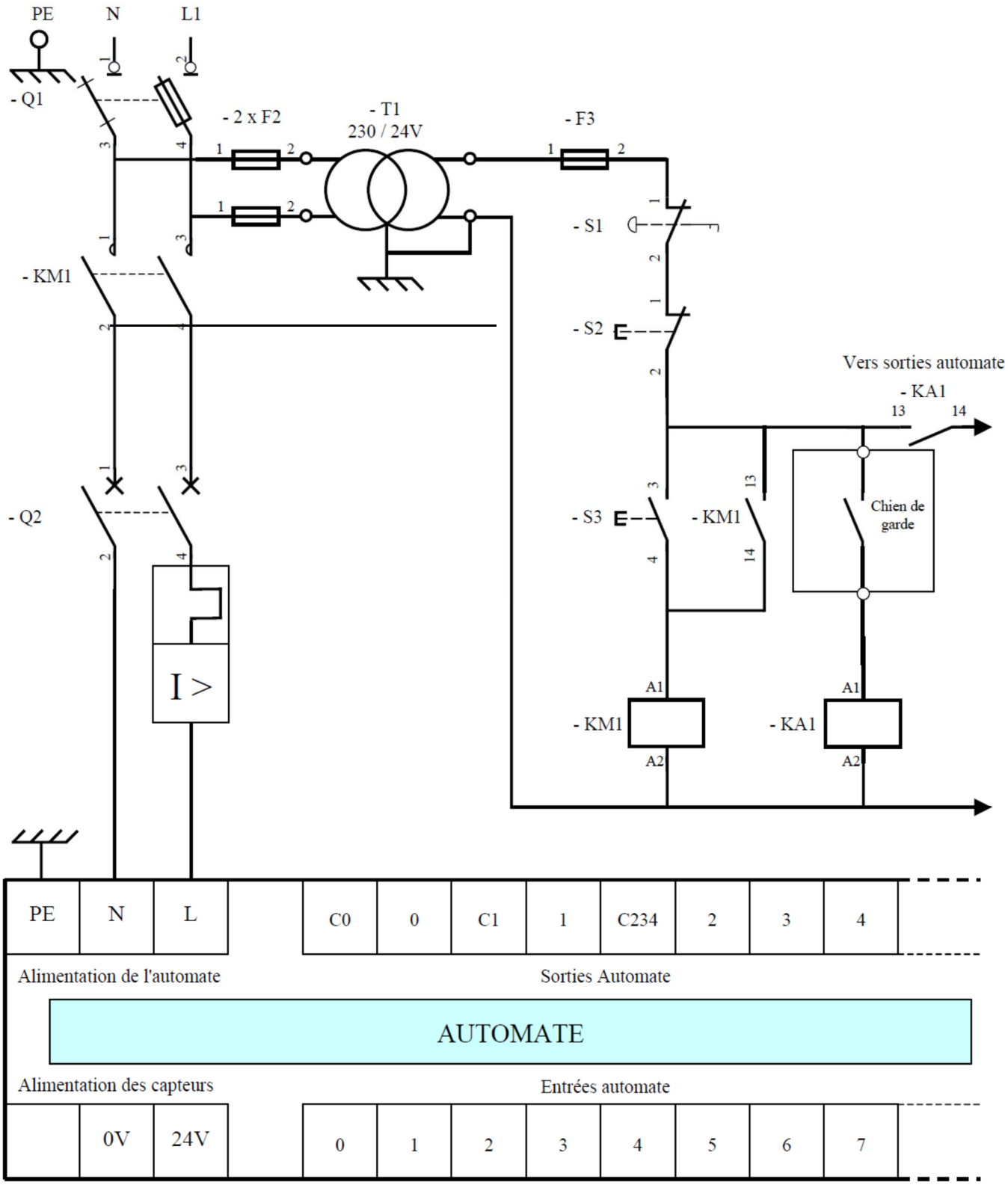


Structure interne d'un Automate Programmable Industriel (API)

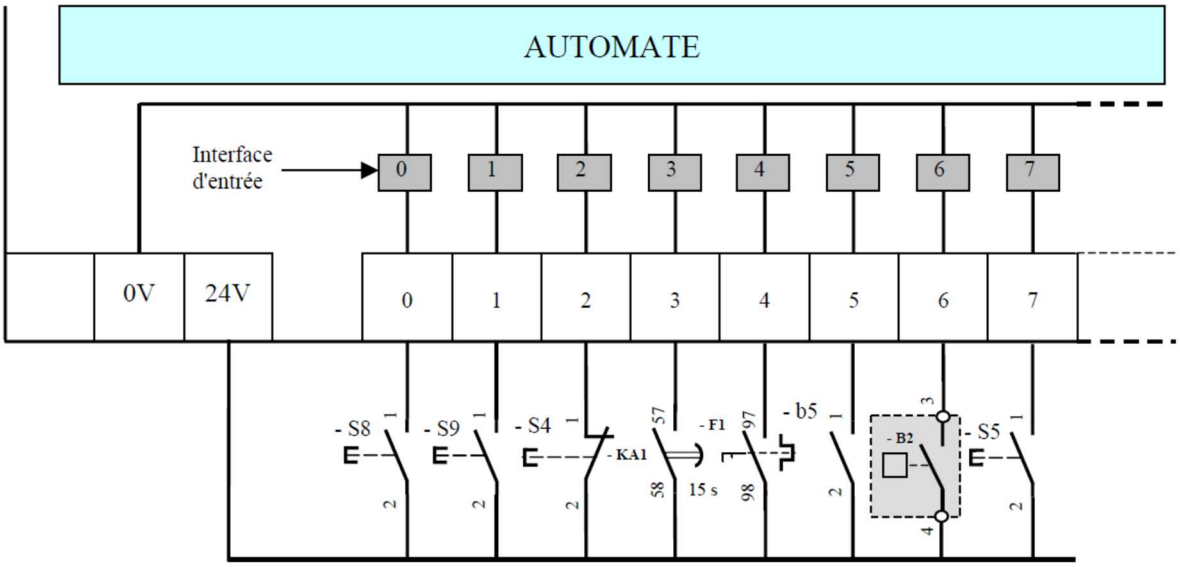
c) Câblage d'un A.P.I.

- Câblage de l'alimentation

On note généralement une alimentation monophasée de l'automate et une alimentation secondaire pour les sorties. Diverses protections sont nécessaires (cf. page suivante)


PE
N
L1
- Q1
- 2 x F2
- T1
230 / 24V
- F3
- S1
- S2
- KM1
- Q2
- S3
- KM1
- KA1
Vers sorties automate
Chien de garde
I >
A1
A2
- KM1
- KA1
PE
N
L
C0
0
C1
1
C234
2
3
4
Alimentation de l'automate
Sorties Automate
AUTOMATE
Alimentation des capteurs
Entrées automate
0V
24V
0
1
2
3
4
5
6
7

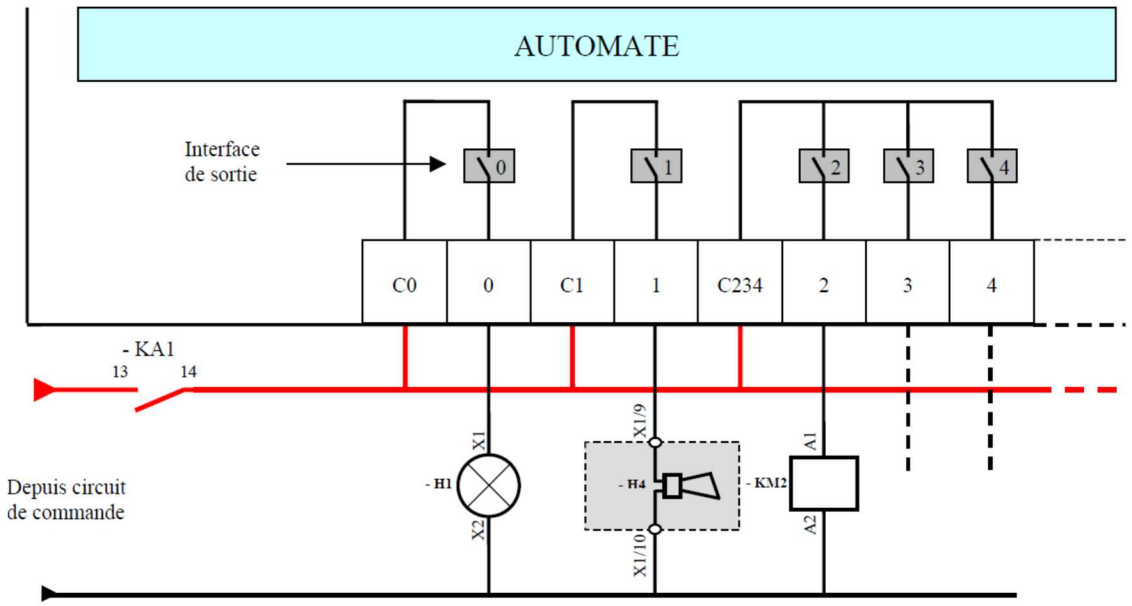
➢ Câblage des entrées automates



L'automate est pourvu généralement d'une alimentation pour les capteurs/détecteurs (attention au type de logique utilisée : logique positive ou négative).

Les entrées sont connectées au OV (commun) de cette alimentation. Les informations des capteurs/détecteurs sont traitées par les interfaces d'entrées.

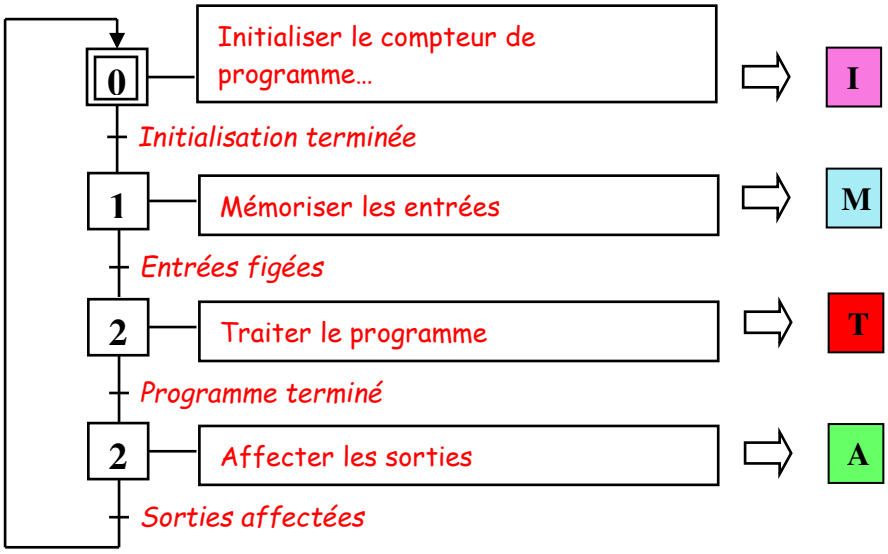
➢ Câblage des sorties



Les interfaces de sorties permettent d'alimenter les divers pré-actionneurs.

d) Cycle de fonctionnement d'un A.P.I.

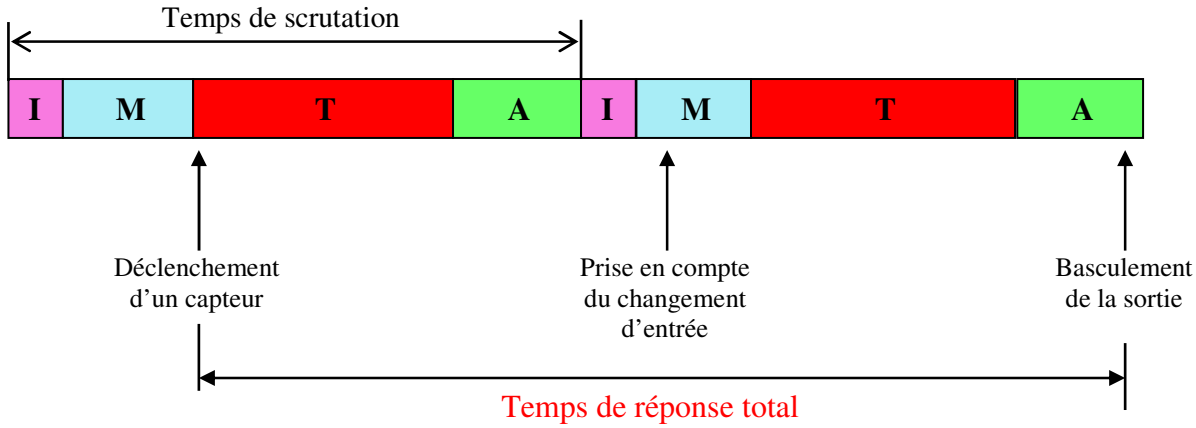
Le Grafcet suivant montre le cycle :



Ces opérations sont effectuées continuellement par l'automate (fonctionnement cyclique).

On appelle **scrutation** l'ensemble des opérations réalisées par l'automate et le temps de scrutation est le temps mis par l'automate pour traiter la même partie de programme. Ce temps est de l'ordre de la dizaine de millisecondes pour les applications standards.

Le **temps de réponse total (TRT)** est le temps qui s'écoule entre le changement d'état d'une entrée et le changement d'état de la sortie correspondante :

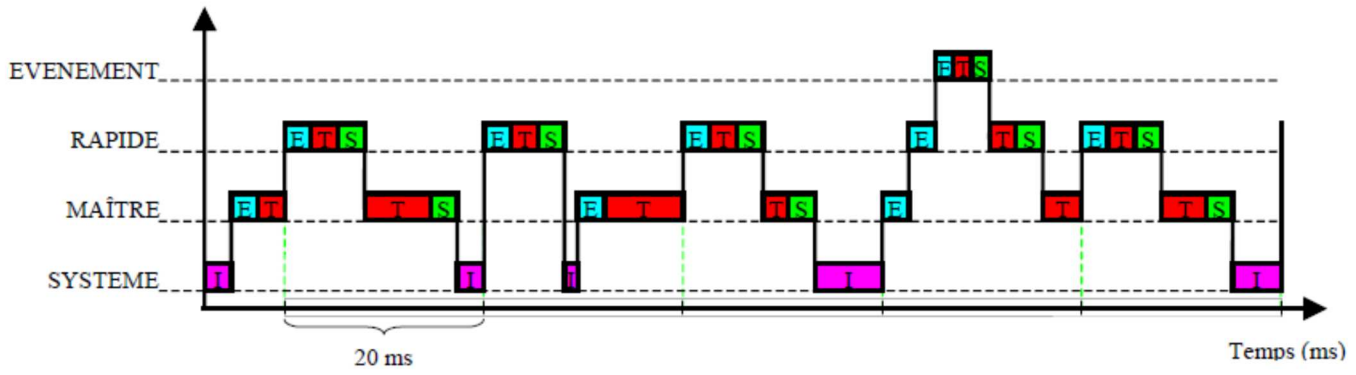


Le temps de scrutation est directement lié au programme implanté. Ce temps peut être fixé à une valeur précise (fonctionnement périodique), le système indiquera alors tout dépassement de période.

Dans certains cas, on ne peut admettre un temps de réponse aussi long pour certaines entrées : ces entrées pourront alors être traitées par l'automate comme des événements (traitement événementiel) et prises en compte en priorité (problème de sécurité, coupure d'alimentation ...).

Certains automates sont également pourvus d'entrées rapides qui sont prises en compte avant le traitement séquentiel mais le traitement évènementiel reste prioritaire.

Le traitement peut être multitâche :



On observe ici les quatre tâches précédemment décrites dans le Grafcet. La tâche rapide est cyclique et revient toute les 20 ms dans notre cas. Les tâches Maître et système ne sont pas cycliques. La tâche événementielle est prioritaire.

e) Programmation d'un A.P.I.

Plusieurs langages sont normalisés au plan mondial par la norme CEI 61131-3.

- Le grafcet ou spc est un langage de haute élaboration qui nécessite un logiciel et un post processeur adapté
- Liste d'instructions (IL : Instruction list) : Langage textuel de même nature que l'assembleur (programmation des microcontrôleurs). Très peu utilisé par les automaticiens.

- Langage littéral structuré (ST : Structured Text) : langage informatique de même nature que le Pascal, il utilise les fonctions comme if ... then ... else ... (si ... alors ... sinon ...) Peu utilisé par les automaticiens.

- Langage à contacts (LD : Ladder diagram) : Langage graphique développé pour les électriciens. Il utilise les symboles tels que : contacts, relais et blocs fonctionnels et s'organise en réseaux (labels). C'est le plus utilisé.

- Blocs Fonctionnels (FBD : Function Bloc Diagram) : Langage graphique ou des fonctions sont représentées par des rectangles avec les entrées à gauche et les sorties à droites. Les blocs sont programmés (bibliothèque) ou programmables. Utilisé par les automaticiens.

Les Travaux pratiques permettront de se familiariser avec la programmation par Grafcet et par langage à contact (Ladder)

f) Choix d'un A.P.I.

Le choix nécessitera de s'interroger sur :

- La méthode de programmation (logiciel, interface…)

- La formation du personnel

- Les caractéristiques propres de l'API en fonction du cahier des charges du système :

  - 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 (Profibus ...).

3) Programmation des A.P.I.

3-1) Type de langages :

- Liste d'instruction : IL (Instruction List)

C'est un pseudo assembleur, sous forme de texte, chaque instruction assembleur représente une instruction machine (Le langage machine est écrit sous forme d'octets).

Exemple :

Démarrer un moteur si le bouton «start» est pressé et si on n'est pas dans un condition d'alarme

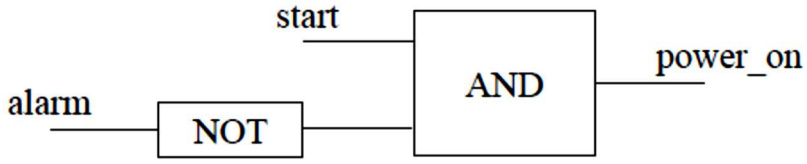
```
VAR
      start : BOOL AT \%IX0.1;
      alarm : BOOL AT \%MX1.5;
      power_on : BOOL AT \%OX3.2;
END_VAR

LD start
ANDN alarm
ST power_on
```

VAR: I/M/O → input/internal/output
X/B/W/L → bit, byte, word, double word

- Langage de haut niveau : ST (Structured Test)

```
if (start and (not alarm) )
      then power_on
end if
```

- Blocs connectables entre eux réalisant des opérations : FDB (Function Block Diagram)



➢ Programmation Ladder :

C'est un langage graphique qui permet d'écrire un programme de contrôle sous la forme d'un schéma électrique comportant des interrupteurs et relais.

Le chapitre suivant est dédié à son apprentissage

3-2) La programmation Ladder :

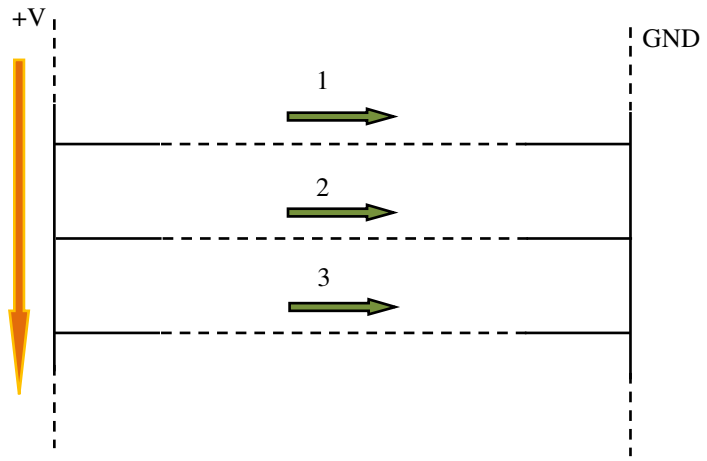
C'est une représentation graphique qui peut facilement être interprétée par une personne ayant des connaissances en électrique sans pour autant avoir appris le langage spécifique pour une programmation simple

a) Représentation de la structure générale :

Ladder signifie échelle en anglais. L'alimentation électrique, représentée par deux verticales (masse à droite), sont reliés par des barreaux (rung = échelons en anglais). La circulation du courant de la gauche vers la droite sera conditionnée par les éléments placés sur chaque ligne (rung).

Chaque rung représente une instruction de programme. Ils se lisent de haut en bas (le programme les lit dans l'ordre à chaque cycle)

Sur chaque ligne, l'évaluation des valeurs se fait de gauche à droite



b) Eléments fondamentaux :

Contacts (ou interrupteurs) :

Représentent les données d'entrée de l'instruction. Ils sont positionnés dans la partie gauche du rung.

On peut définir un contact NO ou NC

Ainsi que des contacts qui détectent des fronts montant ou descendant

Front montant — Front descendant

Bobines (ou relais)

Représentent les données de sortie de l'instruction.

On a également des bobines directes (NO) ou inversée (NF)

Si alimentée alors X=1 sinon X=0 — Si alimentée alors X=0 sinon X=1

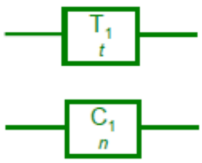
Et des bobines positionnées en Set ou Reset :

Si alimentée, alors X positionné définitivement à 1 — Si alimentée, alors X positionné définitivement à 0

Blocs fonctionnels :

Permettent de réaliser des fonctions avancées (temporisation, comptage…)

On place seul bloc par rung généralement. La syntaxe exacte est fonction du logiciel utilisé



c) Exemple : programmation d'un OU exclusif :

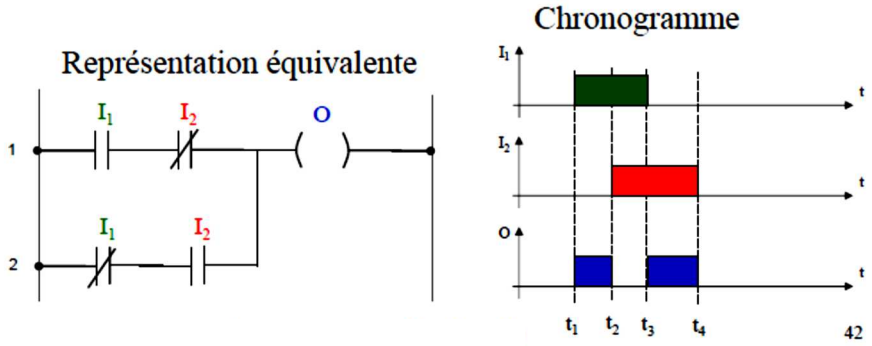
Entrées		Sortie
I1	I2	O1
0	0	0
0	1	1
1	0	1
1	1	0

La programmation structurée donnerait :

XOR

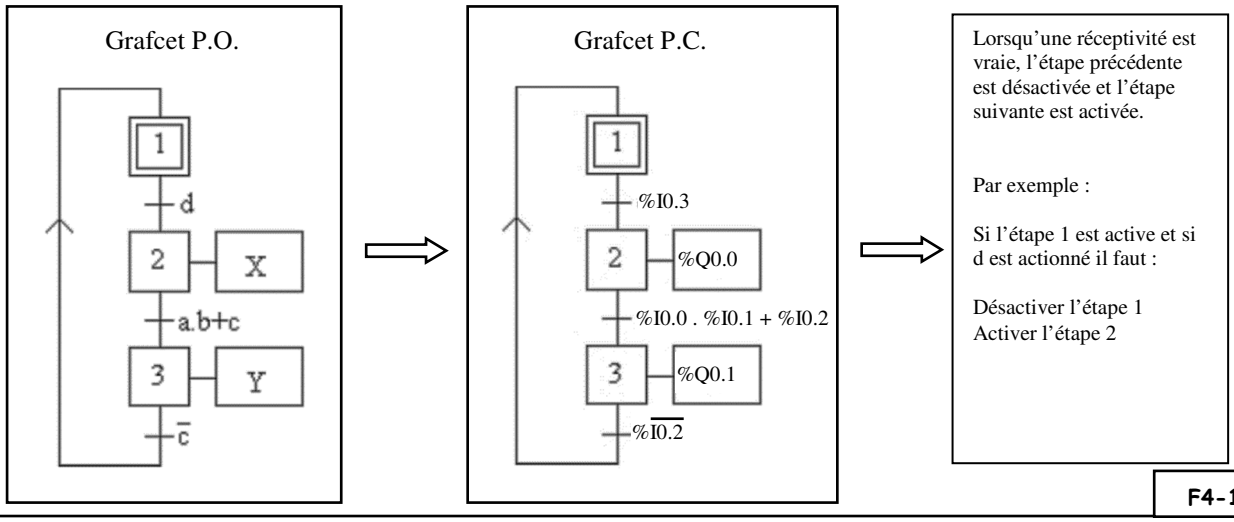
if (I_1 **and** (**not** I_2)) or ((**not** I_1) **and** I_2)
then O = 1
else O = 0
end if

La programmation Ladder donne



4) Du Grafet vers le Ladder

La programmation va mettre en évidence les règles de fonctionnement d'un Grafcet :

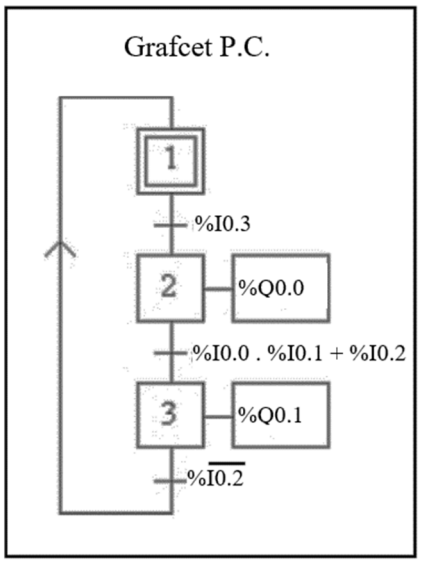


Lorsqu'une réceptivité est vraie, l'étape précédente est désactivée et l'étape suivante est activée.

Par exemple :

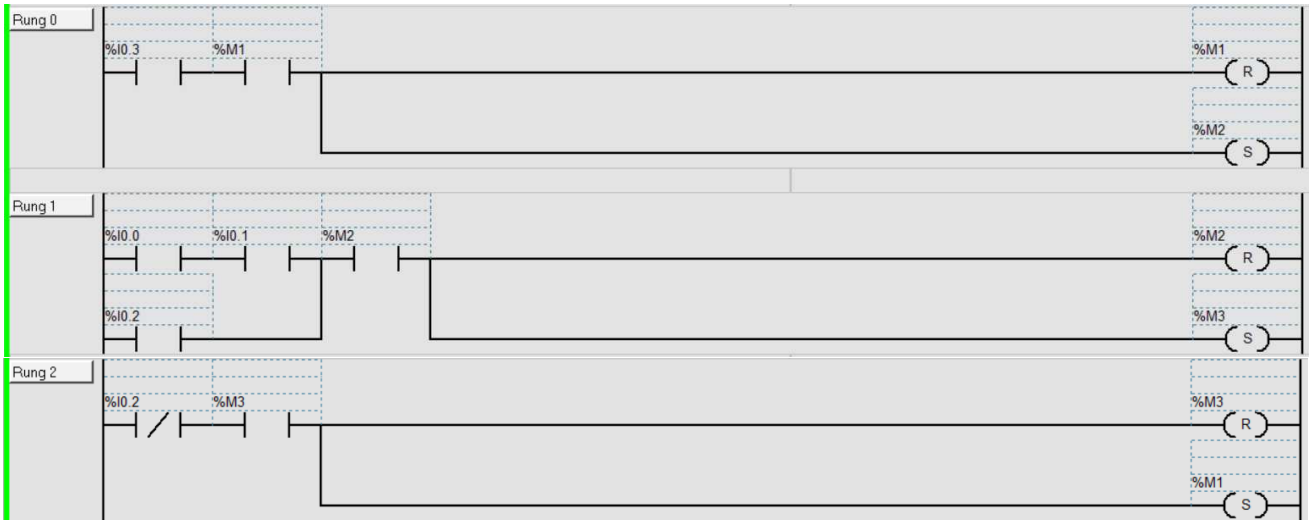
Si l'étape 1 est active et si d est actionné il faut :

Désactiver l'étape 1
Activer l'étape 2

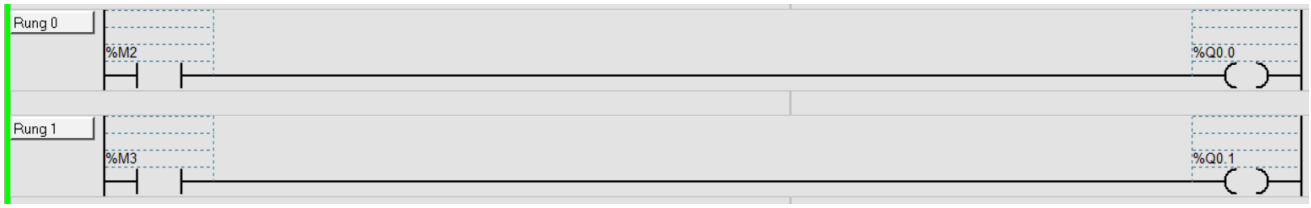


On utilisera les mémoires internes %M :
La variable (mot = octet) %Mn sera associée à l'étape n

Programmation des activations et désactivations des étapes



Programmation des sorties (actionneurs)



Programmation de l'initialisation (permet ici d'activer l'étape initiale)