AUTOMATISMES

Jean-François MAZOIN
(Département Génie chimique-IUT Paul Sabatier de Toulouse)

Sommaire

Introduction

On entend par automatismes tout ce qui met en œuvre des actions discontinues.

Appareillage procédé :

- Capteurs :
détection de seuils (LH)
détecteurs d'états (Flamme)
interrupteurs

- Actionneurs :

vannes Tout Ou Rien - TOR - (Ouverte ou Fermée)
moteurs - de pompe, d'agitation… - (Marche ou Arrêt)
lampes

Signaux :

Les signaux véhiculent une information discontinue, ils sont donc eux-mêmes discontinus. Ce sont des signaux en tension du type "circuit sous tension" et "circuit hors tension". A l'inverse de l'information sur une mesure, l'information sur un seuil ne requiert pas de précision, elle exige simplement que l'on puisse différencier deux états. Les pertes de tension dans les lignes entre le procédé et la salle technique ne constituent pas ici une perte d'information.
Ainsi, un message du type "seuil atteint" sera représenté par "circuit sous tension".

Loi de Commande :

Elle met en œuvre des fonctions logiques et ainsi est nettement plus simple à régler que la loi de commande de la régulation continue.
Exemple : Si "seuil atteint" Alors "vanne TOR fermée".
Remarque : les automatismes portent également le nom d'automatique logique.

Technologies

Câblée : c'est l'ancienne technologie des automatismes, elle met en œuvre des contacts, des relais, des bobines… Cette technologie est abandonnée au profit de technologies plus modernes et plus souples. En effet, dans la technologie câblée, la loi de commande est figée dans le câblage.
Programmée : elle fait appel à des outils d'informatique industrielle que l'on appelle les automates programmables. Elle est de plus en plus systématiquement employée car grâce à l'aspect de programmation de la loi de commande, celle-ci est très facilement adaptable aux besoins et aux évolutions du processus.

Applications :

Gestion des phases utilitaires d'un procédé comme le démarrage ou l'arrêt en décrivant et automatisant l'ensemble des phases de ce démarrage ou de cet arrêt.
Elle se retrouvera également dans la gestion permanente des sécurités.

On voit que ces applications présentent une différence fondamentale : le temps. Le premier cas est un procédé séquentiel. C'est à dire que la loi de commande doit intégrer le paramètre temps, les actions sont gérées dans un ordre bien déterminé. Dans le deuxième cas, les sécurités doivent être actives en permanence et leur apparition peut s'effectuer dans un ordre indéterminé et à un instant indéterminé.
Il existe deux types de lois de commande :

Lois de commandes combinatoires
Lois de commande séquentielles

Lois de commandes combinatoires

$S_i = f(e_i)$

Les sorties (actionneurs) sont une fonction logique des entrées (capteurs). On parle de loi de commande combinatoire parce que les sorties résultent exclusivement de la combinaison des entrées.

Lois de commandes séquentielles

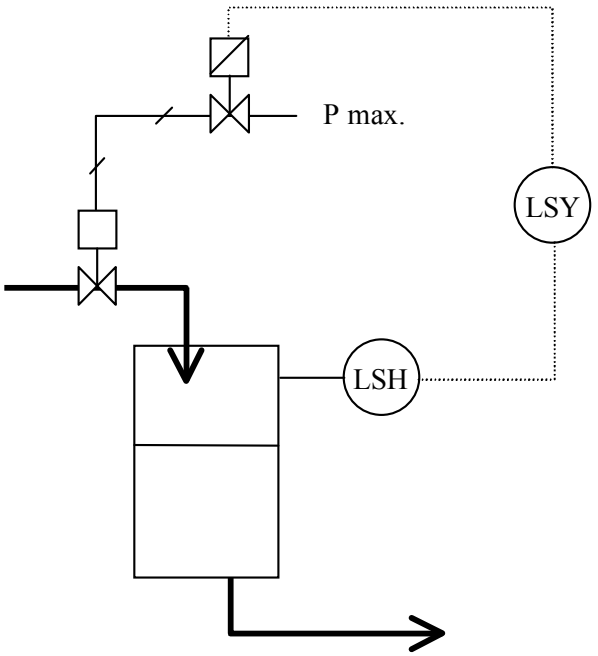
$S_i = f(e_i , t)$ ou $S_i = f(t)$

Les sorties (actionneurs) sont une fonction logique des entrées (capteurs) et du temps (degré d'avancement de l'opération). On parle de séquentiel parce que la structure de la loi de commande est définie par séquences. Cette chronologie devra être intégrée dans la loi de commande.

Nous allons d'abord définir la logique combinatoire en utilisant le langage de programmation "ladder" ou "à contact", héritage de l'automatique câblée. Nous verrons ensuite qu'il existe un outil de programmation de logique séquentielle appelé le Grafcet. Grâce à cet outil, un programme séquentiel peut être construit et la plupart du temps la programmation sur automate sera effectuée sur Grafcet. Cependant, tous les automates n'intègrent pas le langage Grafcet, nous verrons comment à partir de la structure Grafcet d'une application, mettre ce programme en équations combinatoires.

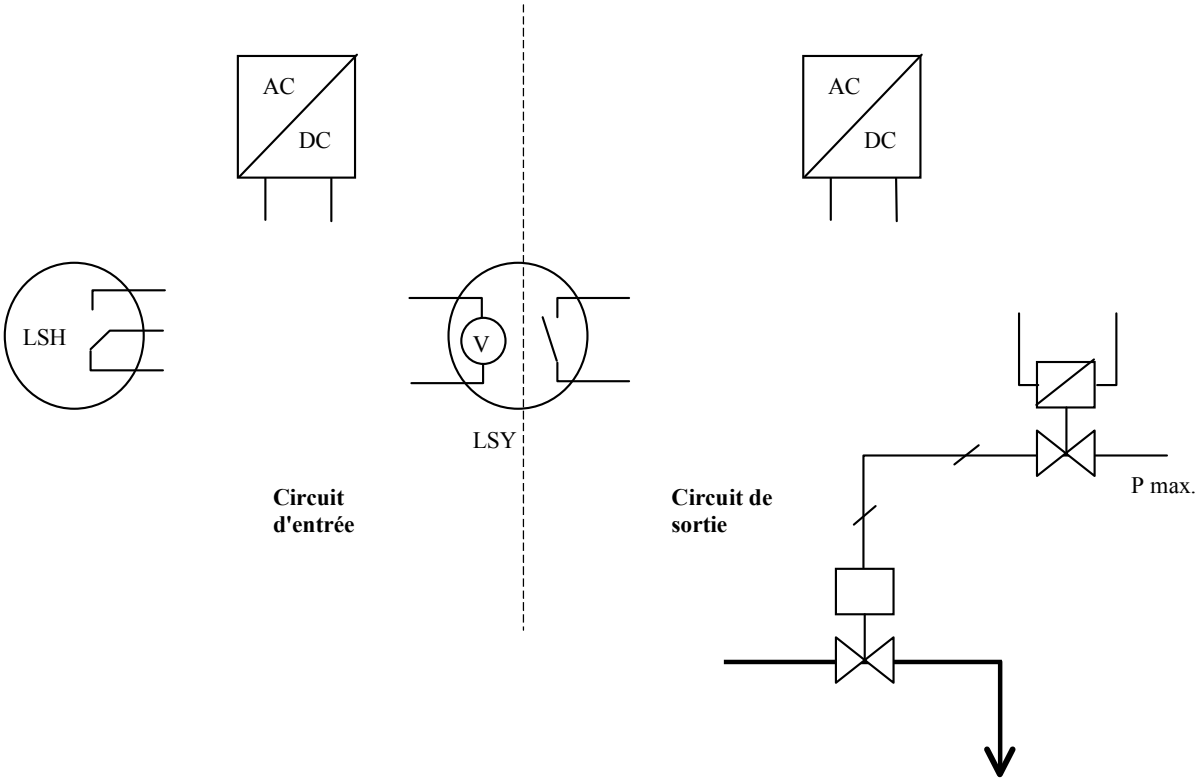
Chapitre 1 : Fonctions Logiques - Combinatoire

Exemple :



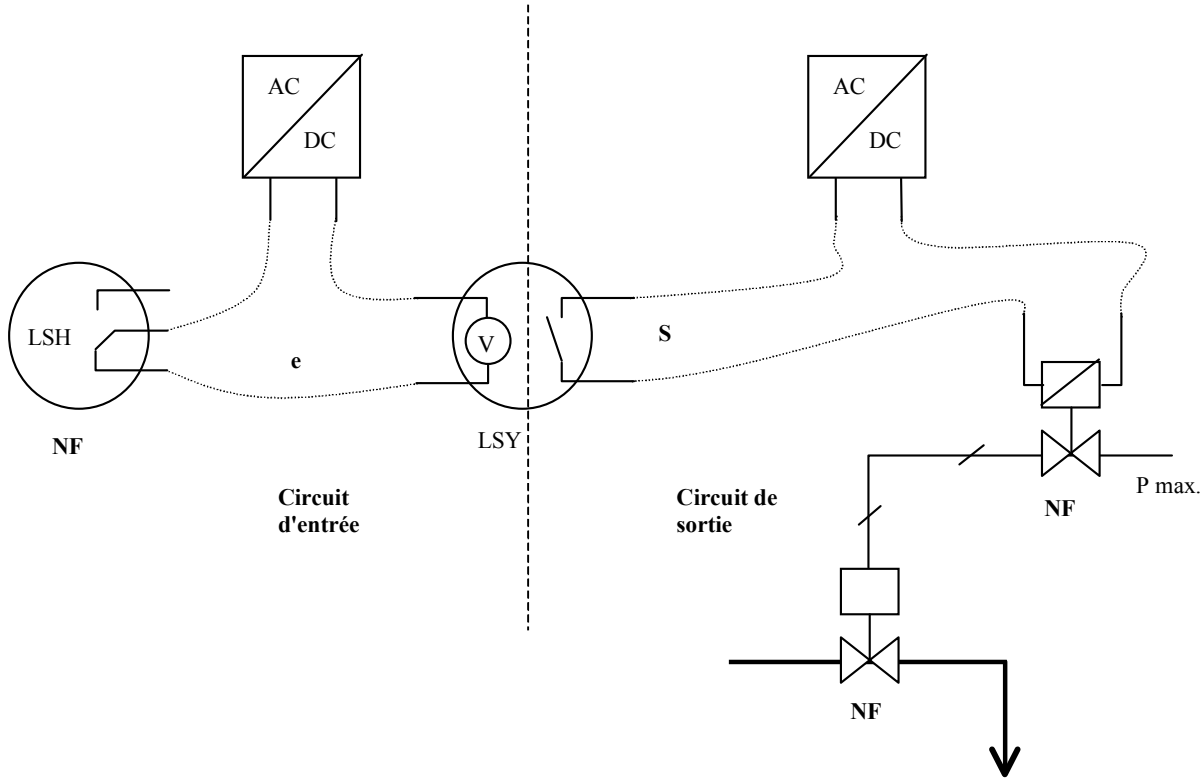
Le capteur LSH est un détecteur de seuil pouvant délivrer un contact ouvert au repos (normalement ouvert) ou un contact fermé au repos (normalement fermé). L'organe de correction est une vanne pneumatique fonctionnant sur du 0 de pression (pression atmosphérique) ou une pression maximale en général comprise entre 4 et 10 bar relatifs. La commande pneumatique est admise dans le servomoteur de la vanne procédé au moyen d'une électrovanne sur le circuit pression appareil. Cette électrovanne de commande fonctionne en général en 24 V continu.
LSY est un automate programmable ou simplement un pilote électronique, son rôle est d'assurer la loi de commande.

Câblages :



Le transformateur redresseur n'est présent que lorsque l'automate LSY présente des contacts secs en entrée et en sortie. L'automate peut lui-même alimenter le circuit d'entrée et le circuit de sortie. L'entrée du LSY fonctionne comme un voltmètre, en fait le LSY détecte la continuité électrique dans le circuit d'entrée et assure ou non (contact) la continuité électrique dans le circuit de sortie.
On constate qu'un premier choix doit être fait : sur le détecteur, il faut choisir un contact NO ou NF, sur l'électrovanne de commande, il faut choisir une électrovanne NO ou NF et sur la vanne de commande procédé, il faut choisir une vanne NO ou NF.
La sécurité du procédé est assurée par la vanne pneumatique, dans notre cas, la sécurité du procédé impose une vanne NF (éviter le débordement du réservoir en cas de problème) ; cette position correspond à un défaut de pression dans le servomoteur de la vanne pneumatique. La position de repos de la vanne pneumatique doit correspondre à la position de repos de l'électrovanne, ainsi, en cas de coupure du circuit de commande de l'électrovanne, celle-ci prend sa position de repos, coupant la pression dans le servomoteur de la vanne pneumatique pour que celle-ci rejoigne sa position de repos et donc sa position de sécurité. La pression d'air devant être toujours coupée, il conviendra de choisir toujours une électrovanne NF.
Enfin, le choix au niveau du détecteur doit faire correspondre une rupture du circuit d'entrée (discontinuité électrique) à la position de repos de la vanne procédé : vanne procédé au repos (fermée) quand le niveau haut est atteint (capteur actif). Donc, lorsque le capteur est actif, le contact est ouvert pour assurer la discontinuité électrique ; au repos, le capteur doit être fermé. Il convient donc de choisir un capteur NF.

Câblage :



Loi de commande :

Elle résulte de la proposition logique suivante : "Si le niveau haut est atteint alors fermer la vanne.". Cette proposition doit être traduite en terme de continuité électrique des circuits d'entrée et de sortie puisque l'automate détecte la continuité électrique du circuit d'entrée et assure celle du circuit de sortie.

Cette proposition est modifiée de la façon suivante :

Niveau haut atteint → Capteur actif → Contact ouvert (NF) → Discontinuité électrique du circuit d'entrée.
Vanne procédé fermée → Pression du servomoteur nulle (NF) → Electrovanne fermée → Servomoteur hors tension (NF) → Discontinuité électrique du circuit de sortie.

Nous avons alors : "Si Discontinuité électrique du circuit d'entrée alors Discontinuité électrique du circuit de sortie."

Le capteur est câblé sur l'entrée e du LSY et la vanne est câblée sur la sortie s du LSY.

Conventions :

Lorsque une entrée ou une sortie digitale (2 positions) est en état de continuité électrique, elle est prise à 1, lorsqu'elle est en état de discontinuité électrique, elle est prise à 0.

La loi de commande s'écrit alors : "Si e=0 alors S=0" ou encore plus simplement :

S = e

Il s'agit de la fonction logique Identité.

Remarque : on peut vérifier que cette fonction logique assure le complémentaire de la proposition donnée au départ : "Si le niveau haut n'est pas atteint alors ouvrir la vanne.".

I. Variables Booléennes et Fonctions Logiques

Une variable booléenne est une variable qui ne peut prendre que deux valeurs discrètes que l'on nomme 0 ou 1.
Une fonction logique est une combinaison donnée des variables d'entrées.
Lorsque l'on réalise une combinaison de n variables d'entrées, on peut obtenir 2^n combinaisons différentes.

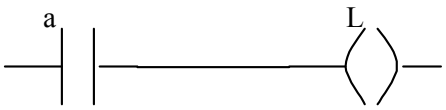
Nous avons déjà vu la fonction identité, la fonction complément défini le contraire de l'identité. Il existe 2 autres fonctions logiques de base, le ET et le OU, puis 4 fonctions particulières méritant d'être citées.

I.1. Fonction Identité

L est une sortie, a et b des entrées.
Proposition logique : "L est à 1 quand a=1"
Table de Vérité :

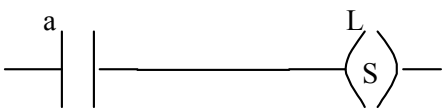
a	L
0	0
1	1

Forme algébrique : L=a
Forme schématique :

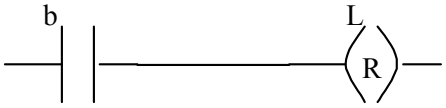


Cette notation vient de la technologie câblée, elle est appelée "ladder" ou "langage à contact". L'équation combinatoire précédente peut être lue de la façon suivante : "Tant que l'entrée a est à 1, la sortie L est maintenue à 1." Et aussi : "Dès que l'entrée a retourne à 0, la sortie L est repositionnée à 0." La bobine L est excitée tant que le contact est maintenu à 1.

Noter la différence avec la proposition suivante : "Dès que l'entrée a passe à 1, la sortie L est positionnée à 1." Et y demeure même si a retourne à 0. Ceci correspond à un "set" :



Seul un "reset" peut ramener à 0 une sortie positionnée à 1 par un "set". Par exemple, retour à zéro de L sur passage à 1 du contact b :

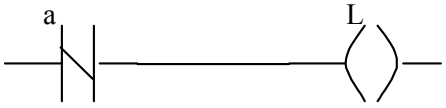


I.2. Fonction Complément

Proposition logique : "L est à 1 quand a=0"
Table de Vérité :

a	L
0	1
1	0

Forme algébrique : $L = \overline{a}$
Forme schématique :

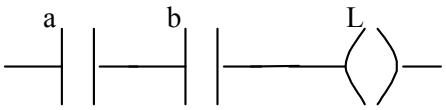


I.3. Fonction ET

Proposition logique : "L est à 1 quand a=1 et b=1"
Table de Karnaugh :
la présentation en table de vérité est toujours valable mais jusqu'à 4 variables d'entrées, la table de Karnaugh permet la simplification rapide des fonctions logiques ; cette présentation sera donc préférée à la table de vérité.

a / b	0	1
0	0	0
1	0	1

Forme algébrique : L=a.b
Forme schématique :

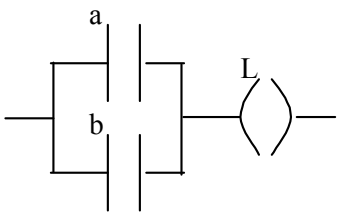


I.4. Fonction OU

Proposition logique : "L est à 1 quand a=1 ou b=1"
Table de Karnaugh :

a / b	0	1
0	0	1
1	1	1

Forme algébrique : L=a+b
Forme schématique :

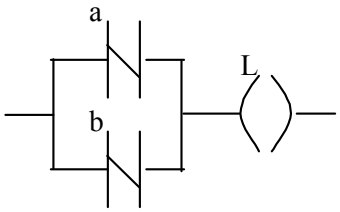


I.5. Fonction NON ET (Nand)

Proposition logique : Proposition complémentaire de ET
Table de Karnaugh : Complémentaire.

a \ b	0	1
0	1	1
1	1	0

Forme algébrique : $L = \overline{a.b}$ ou $L = \bar{a} + \bar{b}$ (Théorème de Morgan : § II.1)
Forme schématique :

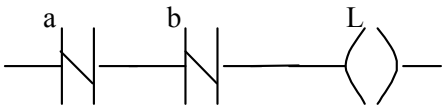


I.6. Fonction NON OU (Nor)

Proposition logique : Proposition complémentaire du OU
Table de Karnaugh : Complément

a \ b	0	1
0	1	0
1	0	0

Forme algébrique : $L = \overline{a + b}$ ou $L = \bar{a}.\bar{b}$ (Théorème de Morgan : § II.1)
Forme schématique :



I.7. Fonction OU EXCLUSIF

Table de Karnaugh :.

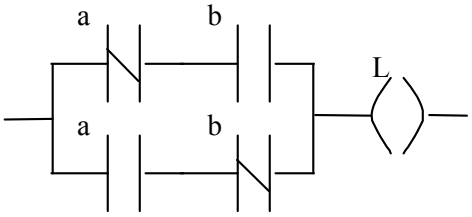
a / b	0	1
0	0	1
1	1	0

Forme algébrique : $L = \bar{a}.b + a.\bar{b}$

Cette forme algébrique s'appelle la <u>première forme canonique</u>. Elle s'obtient en exprimant toutes les combinaisons rendant vraie la fonction décrite dans la table.

Dans notre cas : "L=1 si a=0 *et* b=1 *ou* si a=1 *et* b=0"

Forme schématique :

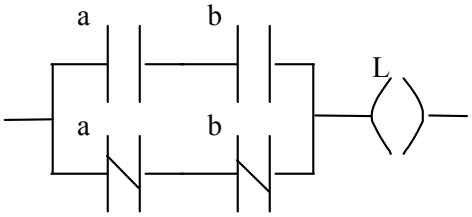


I.8. Fonction COINCIDENCE

Table de Karnaugh :

a / b	0	1
0	1	0
1	0	1

Forme algébrique : $L = a.b + \bar{a}.\bar{b}$
Forme schématique :



II . Simplification des fonctions logiques

On obtient la première forme canonique d'une fonction logique à partir de l'expression de toutes les combinaisons des entrées qui rendent vraies la fonction. Cette forme est toujours la plus coûteuse. Le coût évalue le nombre de fonctions ET et OU intervenant dans l'expression.

Par exemple : pour la fonction coïncidence ci-dessus, le coût est de 3.

Pour réduire ce coût, on va procéder à une simplification de la fonction, lorsque celle-ci est possible.
Cette simplification peut se faire par une méthode algébrique prenant en compte les postulats et théorèmes de l'algèbre de Boole ou par une méthode graphique appelée Méthode de Karnaugh.

II.1. Postulats et théorèmes de l'algèbre de Boole.

Opérations sur la même variable :	$a + a = a$ $a \,.\, a = a$
Commutativité :	$a + b = b + a$ $a \,.\, b = b \,.\, a$
Associativité :	$a + (b + c) = (a + b) + c = a + b + c$ $a.(b.c) = (a.b).c = a.b.c$
Distributivité :	$a.(b + c) = (a.b) + (a.c)$ $a + (b.c) = (a + b).(a + c)$
Eléments neutres :	$0 + b = b$ $a \,.\, 1 = a$
Eléments absorbants :	$1 + b = 1$ $a \,.\, 0 = 0$
Complémentarité :	$a + \bar{a} = 1$ $a \,.\, \bar{a} = 0$
Loi d'involution :	$\overline{(\bar{a})} = a$
Théorème de Morgan :	$\overline{a + b} = \bar{b} \,.\, \bar{a}$ $\overline{a \,.\, b} = \bar{b} + \bar{a}$
Absorption du multiple :	$a + a.b = a$
Absorption du complément :	$a + \bar{a}.b = a + b$

II.2. Méthode de Karnaugh

Cette méthode utilise la table de Karnaugh. Attention dans la table de Karnaugh lorsque l'on passe d'une case à une case voisine, il ne doit y avoir qu'un seul bit qui change : on appelle cette propriété la propriété d'adjacence.

La méthode de Karnaugh consiste à effectuer des regroupement de cases à 1 par puissance de 2 les plus grandes possibles.

Exemple : soit une fonction à 3 entrées a, b et c. La table comporte 8 cases soit 2 à la puissance 3.

On peut faire un paquet de 8 :

a **c b**	0 0	1 0	1 1	0 1
0	1	1	1	1
1	1	1	1	1

L=1

On peut faire un paquet de 4 :

a **c b**	0 0	1 0	1 1	0 1
0	1	1	0	0
1	1	1	0	0

On voit que L est à 1 quelque soit la valeur de a et de c pour b=0, donc $L = \overline{b}$

a **c b**	0 0	1 0	1 1	0 1
0	1	0	0	1
1	1	0	0	1

On voit que L est à 1 quelque soit la valeur de b et de c pour a=0, donc $L = \overline{a}$

On peut faire un paquet de 2 :

a **c b**	0 0	1 0	1 1	0 1
0	1	1	0	0
1	0	1	0	0

On a $L = \overline{b}.\overline{c} + a.\overline{b}$

a **c b**	0 0	1 0	1 1	0 1
0	1	1	0	1
1	0	0	0	0

On a $L = \overline{b}.\overline{c} + \overline{a}.\overline{c}$

On ne peut faire qu'un paquet de 1 (2 à la puissance 0) :

a c b	0 0	1 0	1 1	0 1
0	1	1	0	0
1	0	0	1	0

On a $L = \overline{b}.\overline{c} + a.b.c$

Généralisation : Soit une fonction à n entrées, :

Les paquets de 2^n donnent une réduction à L=1
Les paquets de 2^{n-1} donnent une expression en fonction d'une seule entrée
Les paquets de 2^{n-2} donnent une expression en fonction de deux entrées
…
Les paquets de 2^1 donnent une expression en fonction de n-1 entrées
Les paquets de 2^0 donnent une expression en fonction des n entrées (Première forme canonique).

II.3. Utilisation des combinaisons physiquement impossibles

Il arrive que certaines combinaisons des entrées soient physiquement impossibles, par exemple sur un réservoir, on positionne un détecteur de niveau haut LSH et un détecteur de niveau bas LSL, on suppose que le capteur est au repos quand le seuil n'est pas atteint.
La combinaison LSL =1 et LSH=1 est physiquement impossible !

Puisque ces combinaisons d'entrées ne se produiront jamais, on peut leur affecter la valeur 1 si cela nous permet de réduire le coût de la fonction.

Exemple :

a c b	0 0	1 0	1 1	0 1
0	1	1	0	0
1	1	1	x	0

La combinaison a.b.c est impossible mais ne nous permet pas de simplifier davantage la fonction, on l'affecte donc à 0.

a c b	0 0	1 0	1 1	0 1
0	1	1	1	0
1	1	1	0	0

Cette fonction se simplifie : $L = \overline{b} + a.\overline{c}$

a **c b**	0 0	1 0	1 1	0 1
0	1	1	1	0
1	1	1	x	0

Si maintenant a.b.c est une combinaison impossible alors au lieu d'un paquet de 2, on peut faire un paquet de 4 et la fonction se simplifie davantage : $L = \bar{b} + a$

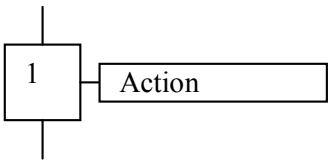
L'utilisation de combinaisons impossibles dans une méthode de simplification algébrique n'est pas aussi simple que dans la méthode de karnaugh.

Chapitre 2 : Fonctions Séquentielles Conditionnelles : Grafcet

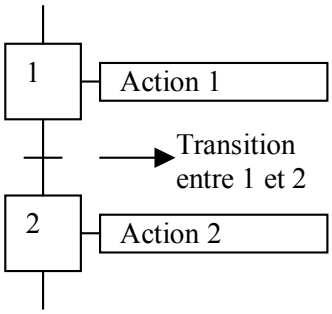
Le Grafcet est un outil de programmation et d'écriture des lois de commande dont l'évolution dépend du temps et de l'état du procédé. Il est basé sur une succession d'étapes séparées par des transitions.

I. Description du Grafcet

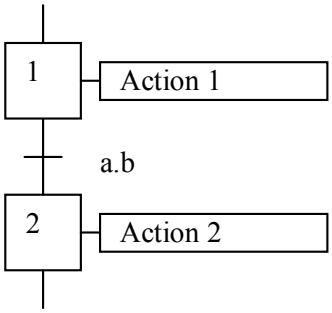
Etape : lors d'une étape, tous les actionneurs conservent la même valeur. En général, une étape correspond à une action, c'est-à-dire au passage à 1 de la sortie de l'automate sur laquelle est câblé l'actionneur. Plusieurs actions peuvent être effectuées dans une étape.



Transition : située entre deux étapes, elle contient la condition logique qui autorise le passage d'une étape à l'autre.



Réceptivité : C'est la condition logique qui autorise le passage de la transition.

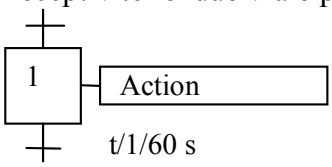


Dans ce cas, il faut que l'équation logique "a et b" soit vraie pour que la transition soit franchie. a et b sont en général des capteurs (ou des variables internes).

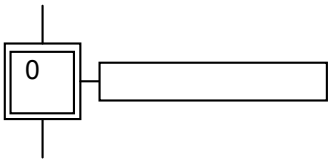
Remarques :

Sens d'évolution d'un Grafcet : descendant, sinon on met une flèche pour signaler un sens inhabituel.

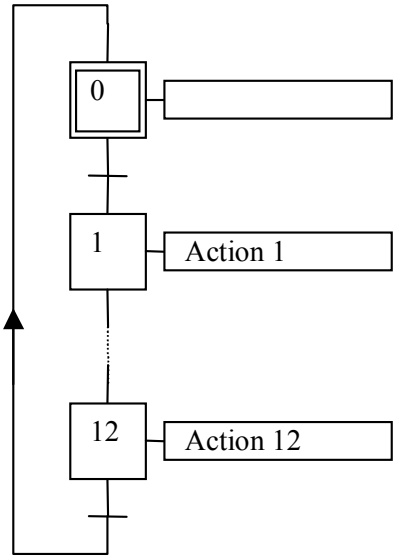
Entre 2 étapes, il y a toujours 1 et 1 seule transition.
Le contenu d'une étape fait référence aux actionneurs (sorties).
Le contenu d'une réceptivité fait référence aux capteurs (entrées).
Si une réceptivité est toujours vraie, on note "=1" en face de la transition.
Réceptivité rendue vraie par temporisation.



Etape Initiale : l'étape initiale marque le démarrage du programme. Lorsque on initialise le grafcet, on se positionne dans cette étape. Elle ne contient pas d'action mais permet d'initialiser des valeurs de sortie si besoin est.



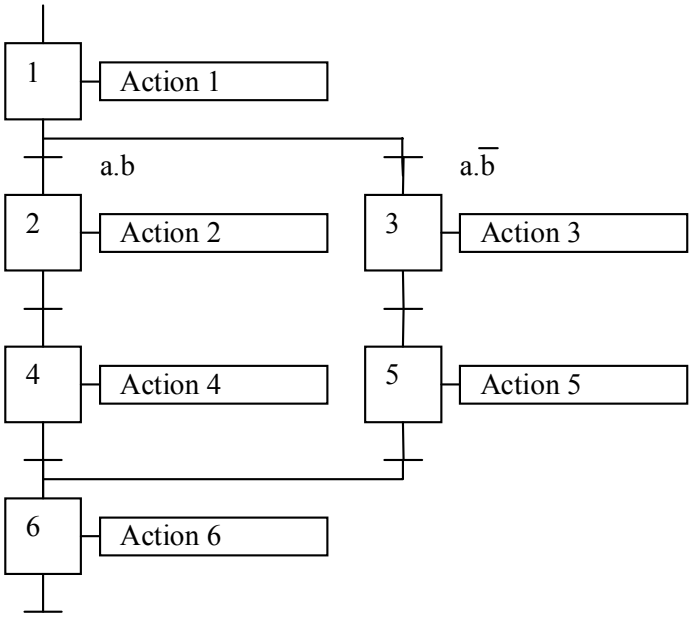
Fin de Programme : elle s'effectue par un retour à l'étape initiale.



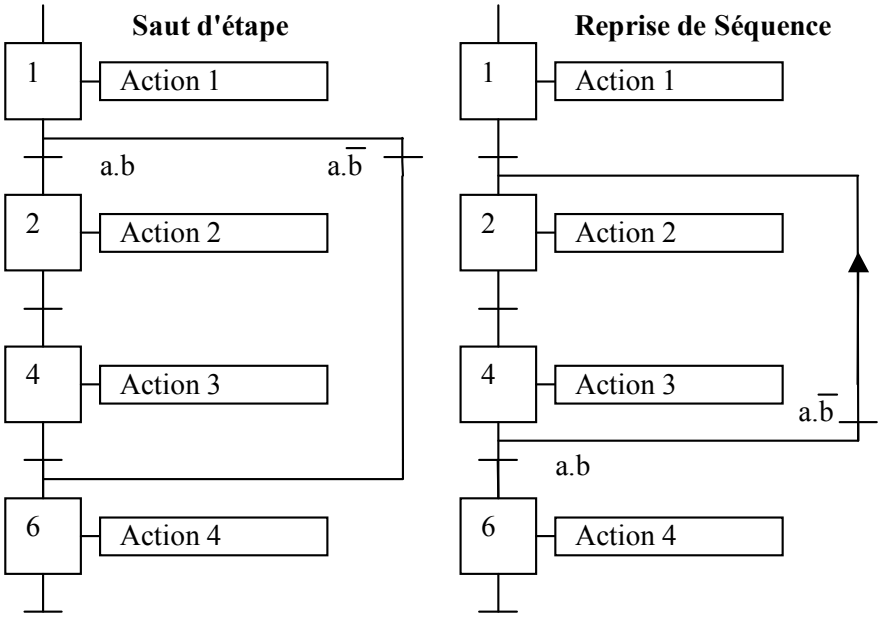
Règles d'évolution du Grafcet :

Une étape est dite active lorsque le programme est pointé sur cette étape. Dans le cas de séquences simultanées, plusieurs étapes sont actives en même temps.
Une transition est dite validée lorsque toutes les étapes immédiatement précédantes sont actives.
Une transition est franchie lorsqu'elle est validée et lorsque la réceptivité associée est vraie.
Au franchissement d'une transition, la ou les étapes précédantes sont désactivées et la ou les étapes suivantes sont activées simultanément.

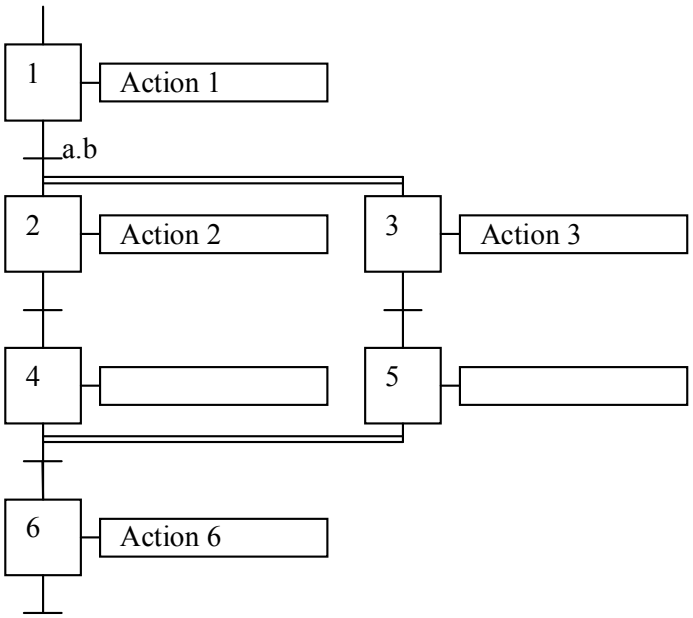
Divergence en OU : elle permet de choisir entre plusieurs séquences. Les transitions qui orientent le choix de la séquence doivent être associées à des réceptivités incompatibles de façon à ce qu'une seule séquence puisse se dérouler en fonction de l'état du procédé.



Cas particulier de la divergence en OU : le saut d'étape et la reprise de séquence.

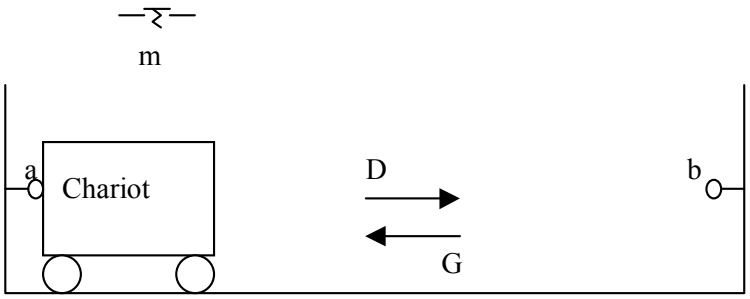


Divergence en ET : elle permet de réaliser simultanément plusieurs séquences. La transition amont une fois franchie, active les étapes immédiatement suivantes. Il est nécessaire de mettre des étapes de synchronisation avant la reprise de séquence, dans l'exemple suivant ce sont les étapes 4 et 5 qui jouent le rôle d'étapes de synchronisation.



II. Exemple :

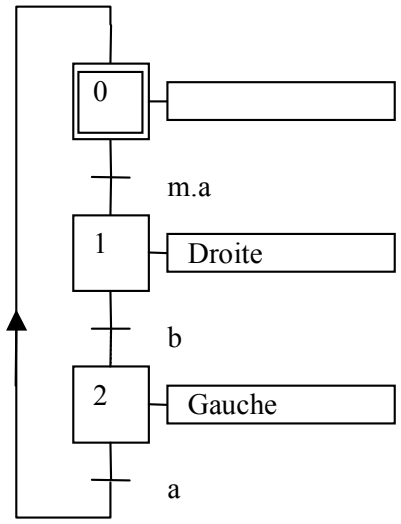
A partir d'un exemple simple, nous allons écrire le programme, l'expliciter pour passer à une programmation sur Automate April 2000 et finalement le traduire en équations combinatoires.



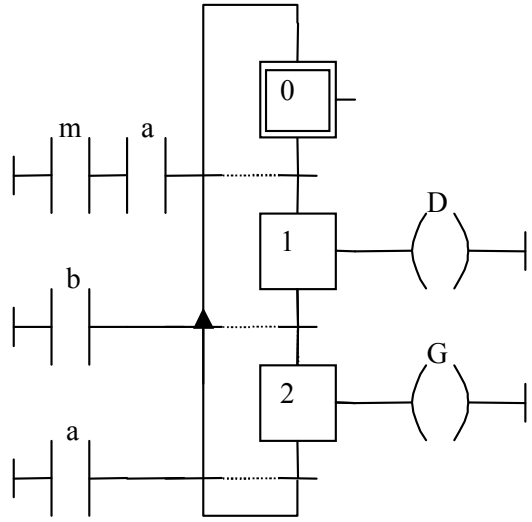
Cahier des Charges : A l'état initial, le chariot est à gauche et le capteur de position a est actif. Lorsque l'opérateur appuie sur le bouton poussoir (NO) m, le moteur D démarre et amène le chariot à droite. Arrivé à droite, le capteur b est actif, l'actionneur D est arrêté et le chariot repart aussitôt vers la gauche (G actif), revenu à gauche il s'arrête.

Ecriture du programme (papier) :

Dans ce programme, une étape délimite une action mais il peut y avoir des actionneurs à manœuvrer en début de programme et à désactiver en fin de programme. On différenciera dans le vocabulaire ces deux types d'actions. Ainsi, le terme "Droite" défini dans l'étape 1 signifie que la sortie correspondant à l'actionneur Moteur de Mouvement vers la Droite est maintenu à 1 pendant toute la durée de l'étape. Si l'action se déroule sur plusieurs étapes, on parlera de "Mise en Marche de l'actionneur" à la première étape, ceci correspond à un "set" de la sortie correspondant à l'actionneur ; on parlera "d'Arrêt de l'Actionneur" après la dernière étape où son fonctionnement doit être assuré, ceci correspond au "reset" de la sortie.



Ecriture Machine (Automate) :



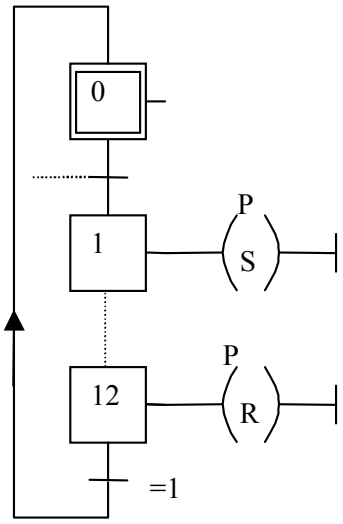
Sur les automates programmables industriels, les réceptivités et les actions s'écrivent dans des pages programmes associées et non directement sur le schéma du Grafcet.

Lorsqu'elles sont validées, les transitions sont vraies dès que l'équation combinatoire sur les capteurs est vraie.

Les actions demeurent à 1 (actives) tant que l'étape est active.

Remarque :

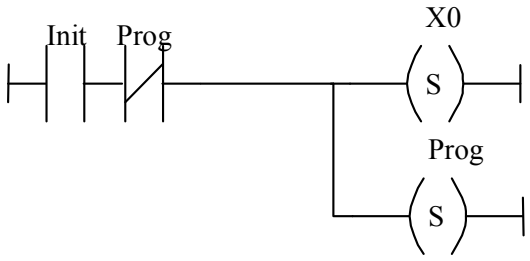
Si une action opère sur plusieurs étapes, par exemple de l'étape 1 à 11, alors on utilisera des "set" et "reset". La réceptivité des étapes d'arrêt d'actionneurs sont en général inconditionnelles d'où "=1".



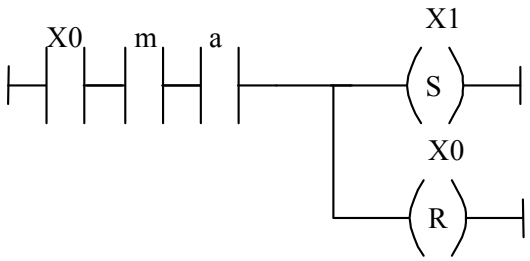
Mise en équation du Grafcet :

Elle est utilisée lorsque la fonction Grafcet n'est pas disponible sur l'automate. Elle est la traduction combinatoire du programme machine standard.

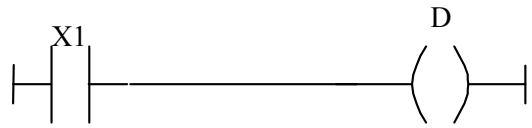
L'étape d'initialisation est activée au moyen d'un contact Init. La variable Prog est initialisée dans le programme principal à 0 et mise à 1 dès le début du premier cycle de l'automate de manière à ce que Init ne repositionne pas X0 à 1 aux cycles suivants tant que le programme n'est pas terminé.



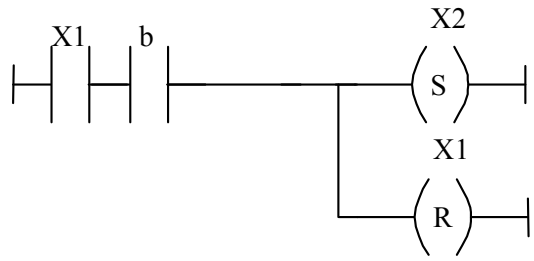
La transition qui permet le passage de X0 à X1. Elle assure la désactivation de l'étape 0 et l'activation de l'étape 1. Le contact X0 représente la validation de la transition, il est suivi de la réceptivité. Pour qu'une transition soit franchie, il faut qu'elle soit validée et que la réceptivité associée soit vraie.



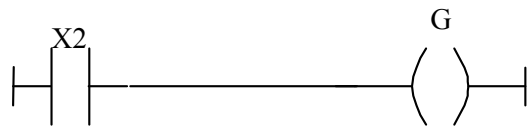
L'étape 1 contient une action : l'actionneur D est maintenu à 1 (actif) tant que l'étape 1 (X1) est active (=1).



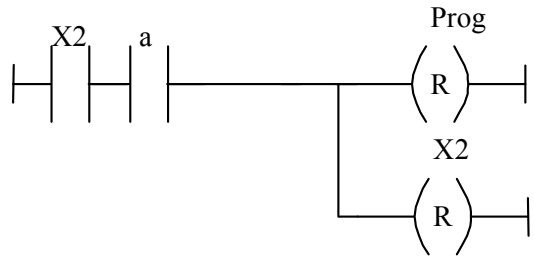
Transition 2 :



Etape 2 :

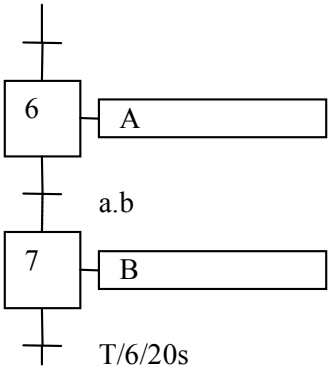


Transition Finale : ici la variable interne Prog est remise à 0 autorisant alors le passage à 1 de X0 si la variable Init est toujours à 1.

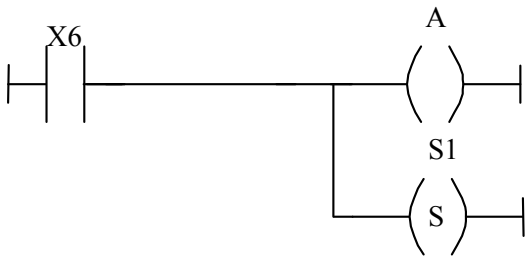


Remarque :

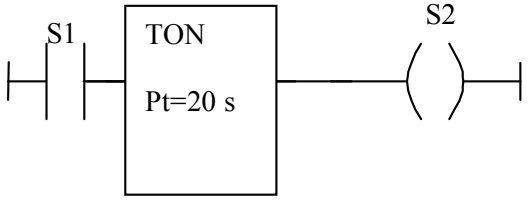
Mise en équation de réceptivité de temporisation. Soit le bout de programme :



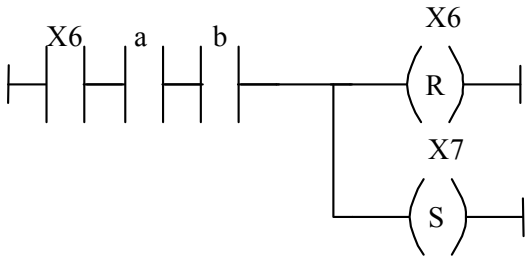
Se traduit en :

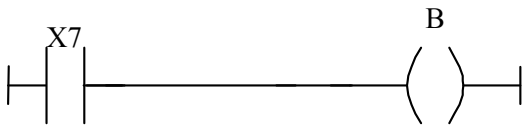


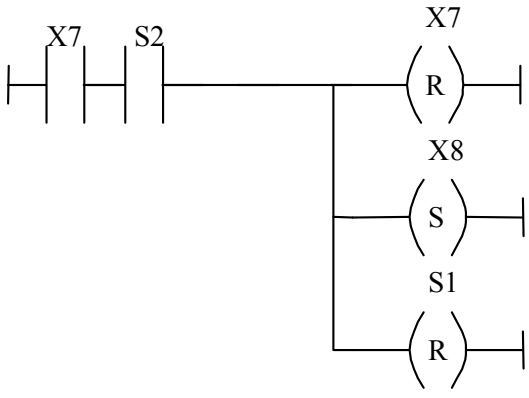
S1 est une variable interne.



S2 est une variable interne.

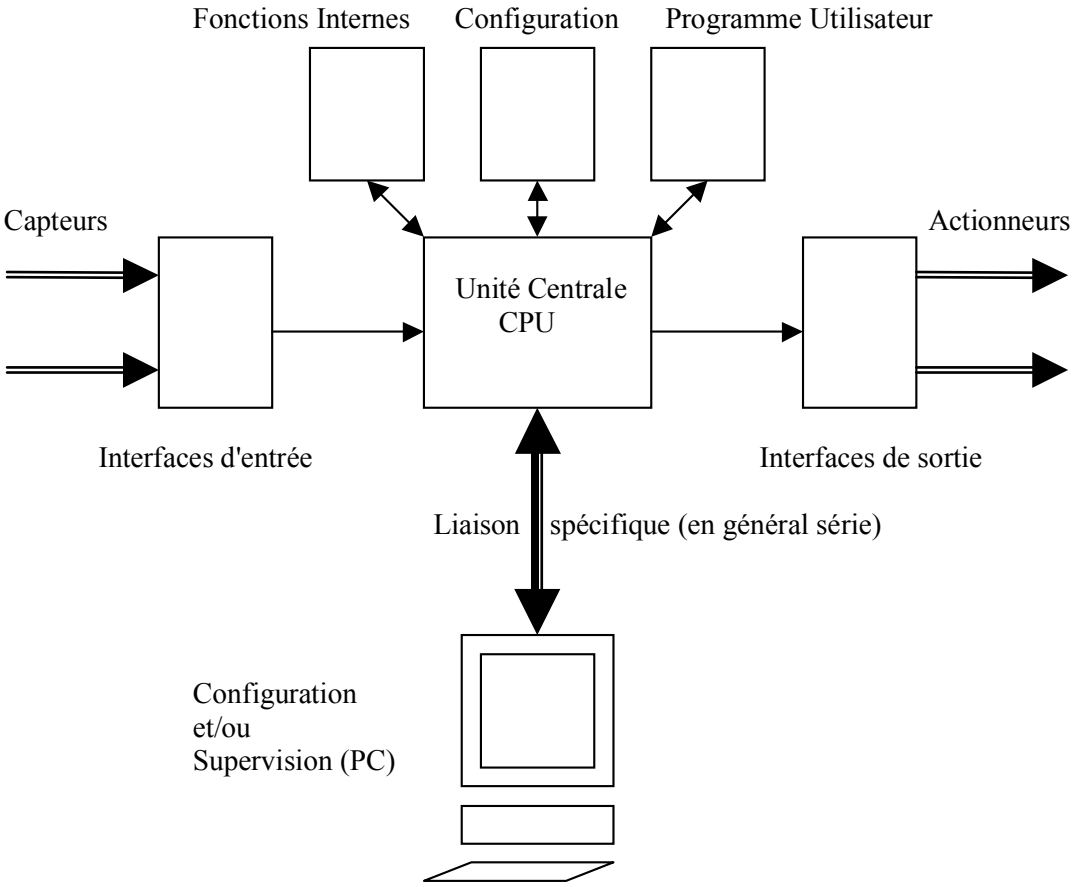






Chapitre 3 : L'Automate Programmable Industriel

I. Structure des automates



Unité centrale : elle gère l'ensemble du processus, elle contient le processeur, les mémoires vives et des mémoires mortes pour une taille débutant à 40 koctets. Elle est programmable directement par console ou par le biais d'une liaison série et d'un logiciel adapté. Cette CPU peut être en RUN (elle exécute le programme), en STOP (toutes les sorties sont au repos, contacts ouverts).
Configuration : elle contient les paramètres liés à la structure de l'API et à la structure du réseau informatique.
Fonctions Internes : ce sont des fonctions pré-programmées livrées avec l'API qui permettent par exemple d'assurer des temporisations, des régulations… Ces fonctions peuvent être résidentes dans l'automate ou disponible dans le logiciel de programmation.
Programme Utilisateur : c'est la loi de commande, il assure la gestion des sorties en fonction de l'état des entrées et éventuellement du temps. Ce programme est exécuté sous forme cyclique par l'API, le temps de cycle est bien sûr dépendant de la taille du programme et ne doit pas excéder la centaine de millisecondes.

Supervision : c'est un ordinateur standard. Il contient le logiciel de programmation (Orphée pour April et Step7 pour Siemens). Ce logiciel permet d'écrire le programme, de le compiler et de le transférer à l'automate. L'ordinateur peut également servir de poste opérateur pour assurer la conduite de l'unité. Un autre logiciel est alors nécessaire pour assurer le dialogue avec l'automate et une interface opérateur conviviale. Si la liaison entre le PC et la CPU est rompue, l'API continue de dérouler son programme.

Interfaces : elles assurent le lien avec le procédé. Ces interfaces peuvent alimenter les boucles d'entrées ou de sorties, dans ce cas, l'automate sera dotée d'une alimentation 24V continue. Elles peuvent être garnies de contacts secs, dans ce cas, une alimentation extérieure devra être intégrée aux boucles d'entrée et de sortie (Voir Chapitre I).

II. Interfaces d'entrées et de sorties

On trouve comme interfaces des cartes d'entrées logiques (ou analogiques), des cartes de sortie logiques (ou analogiques). Le coût d'une carte varie entre 1500 et 4000 francs en fonction du constructeur.

Cartes d'entrées logiques : elles assurent la connexion de capteurs logiques. Une carte standard assure jusqu'à 32 connections. Une carte en Exi (sécurité intrinsèque) assure jusqu'à 16 connections. Les cartes Exi assurent une isolation électronique (barrière Zener) entre le process en zone ADF et la zone électrique standard.

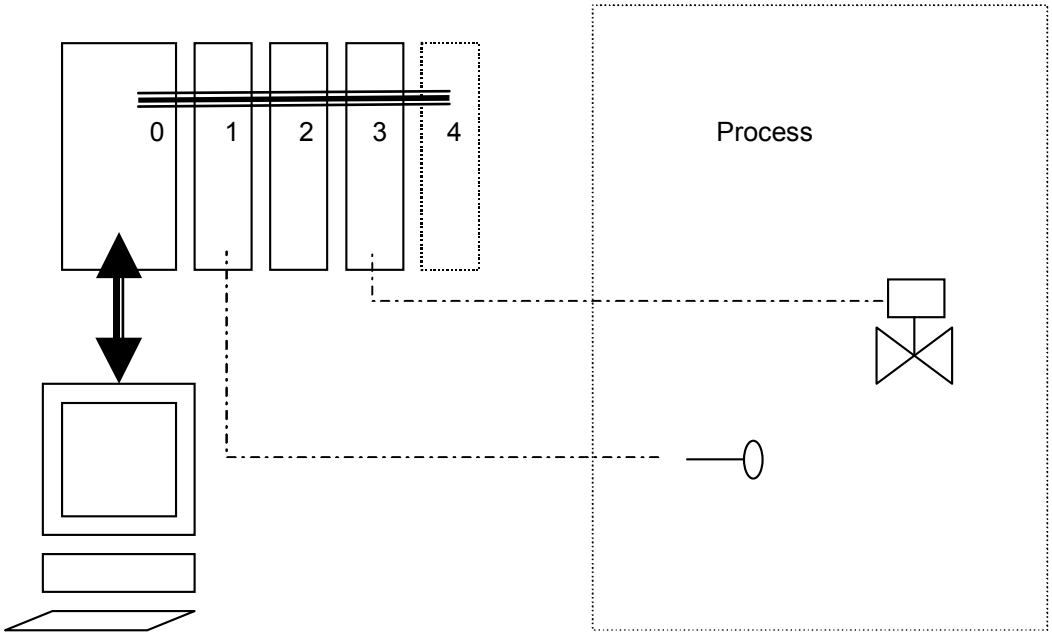
Cartes d'entrées analogiques : elles assurent la connexion des capteurs analogiques (4-20 mA). Une carte standard assure jusqu'à 8 connections. Une carte en Exi (sécurité intrinsèque) assure jusqu'à 4 connections.

Cartes de sorties logiques : elles assurent la connexion des actionneurs logiques. Une carte standard assure jusqu'à 16 connections. Une carte en Exi (sécurité intrinsèque) assure jusqu'à 8 connections.

Cartes de sorties analogiques : elles assurent la connexion des capteurs analogiques (4-20 mA). Une carte standard assure jusqu'à 4 connections. Une carte en Exi (sécurité intrinsèque) assure également jusqu'à 4 connections.

Il existe d'autres types d'interfaces (entrées et sorties logiques, régulateurs entrées et sorties analogiques,…). Chaque capteur ou actionneur câblé possède une adresse dont la description varie d'un type d'automate à l'autre. Cette adresse peut être très proche de l'adresse mémoire machine.

Exemple :



Cet automate de type April 2000 dispose de 5 emplacements de cartes. A l'emplacement 0, on trouve l'unité centrale. Aux emplacements 1 et 2, deux cartes d'entrées logiques (32 voies). A l'emplacement 3, une carte de sortie logique (16 voies). Il n'y a pas de carte d'alimentation, les contacts de l'automates sont donc secs. On câble un capteur sur la première carte d'entrée (voie 12) et un actionneur sur la carte de sortie (voie 3).

<u>Adresse du capteur :</u> % I X 1 11

Type de Carte	Nature	Emplacement	Canal
I=Entrée	X=Logique	1	=Voie-1

<u>Adresse Actionneur :</u> % Q X 3 02

Type de Carte	Nature	Emplacement	Canal
Q=Sortie	X=Logique	3	=Voie-1

Cet adressage est simple mais beaucoup d'automates ont un adressage qui fait référence à la localisation en mémoire de la variable associée à l'entrée ou à la sortie ; un bit pour une E/S logique, un mot soit 2 octets pour une E/S analogique.