LES AUTOMATISMES

LES AUTOMATES PROGRAMMABLES INDUSTRIELS

GJC

Lycée L.RASCOL 10, Rue de la République
BP 218. 81012 ALBI CEDEX

SOMMAIRE

Introduction

à la logique

programmée

I.SYSTEMES LOGIQUES

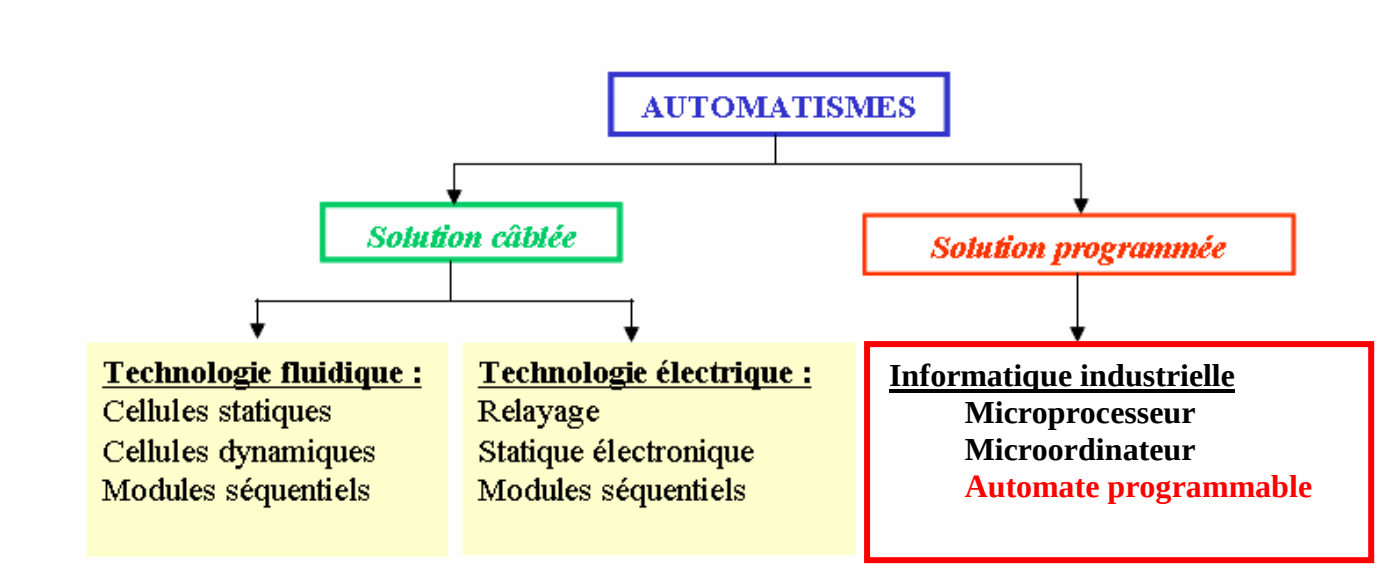
La partie commande élabore des ordres à partir des mesures et des consignes selon la loi de commande de l'automatisme. Celle-ci, pour des procédés logiques caractérisés par la nature tout ou rien ou binaire des informations, est constituée d'un ensemble d'équations Booléennes. Dans les procédés continus caractérisés par des grandeurs analogiques, il s'agirait d'équations fonctionnelles du type équations différentielles ou fonctions de transferts. L'automaticien dispose de nombreux outils technologiques pour réaliser la commande de son système. On les classe généralement en deux catégories : les solutions câblées et les solutions programmées.
Les outils câblés de toutes technologies, malgré leurs qualités éprouvées, souffrent d'un certain nombre de limitations :

- l'encombrement (poids et volume),
- le manque de souplesse pour la mise au point et l'évolution des commandes,
- la difficulté pour maîtriser les problèmes complexes ainsi que ceux liés au dépannage,
- le coût des composants avec rentabilité financière limitée aux fonctions simples en raison de l'apparition des technologies programmées.

Les techniques programmées se distinguent par un seuil de rentabilité constamment décroissant. Ce sont des outils informatiques destinés à traiter de l'information. L'utilisation en gestion et en calcul est connue. Les applications techniques relèvent de l'informatique industrielle. Depuis 1970 environ, l'automaticien dispose d'un outil spécialisé :

L'API : Automate Programmable Industriel

L'informatique industrielle est une discipline conjuguant les théories de l'automatique et les moyens de l'informatique dans le but de résoudre des problèmes de nature industrielle en particulier ceux liés au traitement de l'information en temps réel.



Technologies possibles pour la réalisation de la PC

II.LOGIQUE COMBINATOIRE

II.1. Définition

II.2. Equations

La logique combinatoire se caractèrise par des èquations de la forme :

$$S_i = f (e_1, e_2, ...e_n)$$

II.3. Exemples d'opérateurs combinatoires

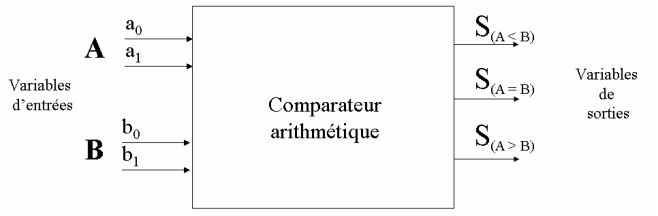
II.3.1.COMPARAISON ENTRE DEUX INFORMATIONS

COMPARATEUR ARITHMETIQUE

Les deux nombres seront exprimés en binaire pur, on veut savoir si:

A ⟶ a1 a0

B ⟶ b1 b0



$ Analyse

b_1	b_0	a_1	a_0	$S_{(A=B)}$	$S_{(A>B)}$	$S_{(A<B)}$
0	0	0	0	1	0	0
0	0	0	1	0	1	0
0	0	1	0	0	1	0
0	0	1	1	0	1	0
0	1	0	0	0	0	1
0	1	0	1	1	0	0
0	1	1	0	0	1	0
0	1	1	1	0	1	0
1	0	0	0	0	0	1
1	0	0	1	0	0	1
1	0	1	0	1	0	0
1	0	1	1	0	1	0
1	1	0	0	0	0	1
1	1	0	1	0	0	1
1	1	1	0	0	0	1
1	1	1	1	1	0	0

$ Equations

$$S_{(A=B)} = /a_0\ /a_1\ /b_0\ /b_1 + a_0\ /a_1\ b_0\ /b_1 + /a_0\ a_1\ /b_0\ b_1 + a_0\ a_1\ b_0\ b_1$$

$$S_{(A=B)} = /a_0\ /b_0\ (/a_1\ /b_1 + a_1\ b_1) + a_0\ b_0\ (/a_1\ /b_1 + a_1\ b_1)$$

$$S_{(A=B)} = (/a_1\ /b_1 + a_1\ b_1) + (/a_0\ /b_0 + a_0\ b_0)$$

$$S_{(A=B)} = (a_1 \odot b_1) + (a_0 \odot b_0)$$

$$S_{(A>B)} = a_0\ /a_1\ /b_0\ /b_1 + /a_0\ a_1\ /b_0\ /b_1 + a_0\ a_1\ /b_0\ /b_1 + /a_0\ a_1\ b_0\ /b_1 + a_0\ a_1\ b_0\ /b_1 + a_0\ a_1\ /b_0\ b_1$$

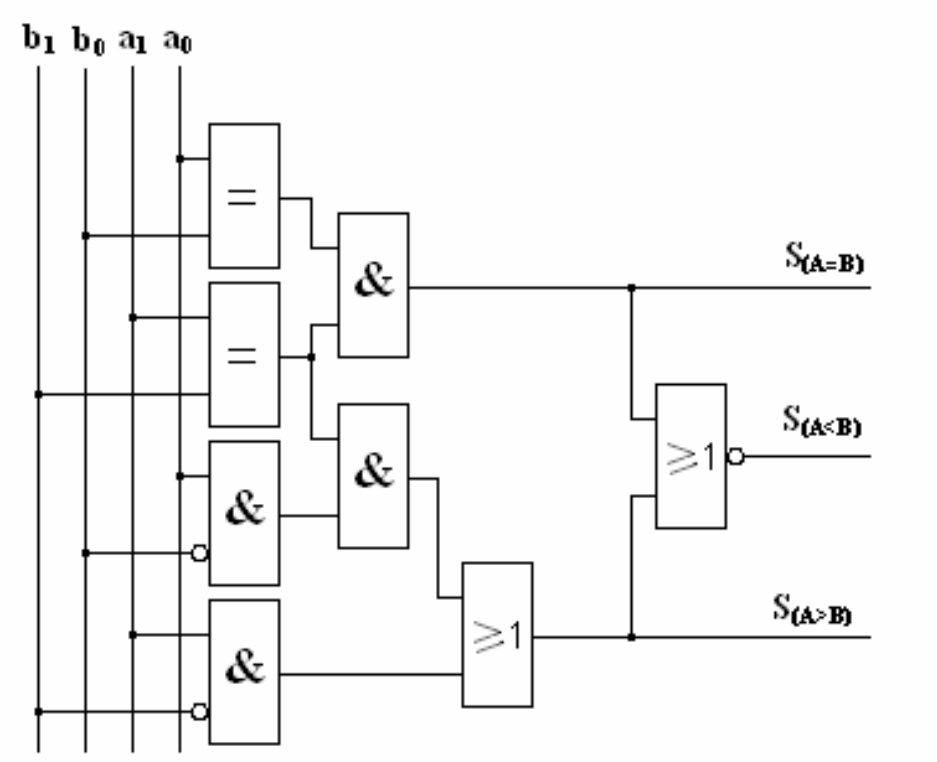
$$S_{(A>B)} = a_1\ /b_1\ (/a_0\ /b_0 + a_0\ /b_0 + /a_0\ b_0 + a_0\ b_0) + a_0\ /b_0\ (/a_1\ /b_1 + a_1\ b_1)$$

$$S_{(A>B)} = a_1\ /b_1\ [(a_0 \odot b_0) + \overline{(a_0 \odot b_0)}] + a_0\ /b_0\ (a_1 \odot b_1)$$

$$S_{(A>B)} = a_1\ /b_1 + a_0\ /b_0\ (a_1 \odot b_1)$$

$$S_{(A<B)} = \overline{S_{(A=B)}} \cdot \overline{S_{(A<B)}} = \overline{S_{(A=B)} + S_{(A<B)}}$$

$ Schéma



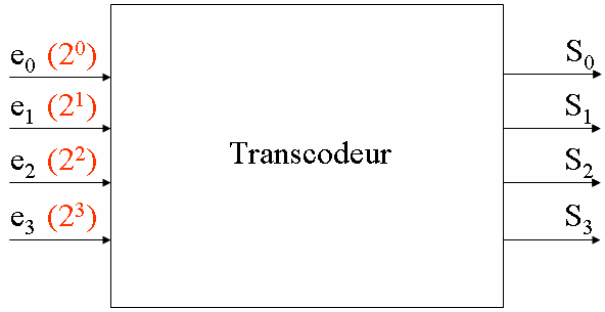
II.3.2.TRANSFORMATION DE CODES

Les ensembles logiques ne peuvent traiter des informations que si elles sont sous forme binaire (0 ou 1). Il en résulte que tout problème, avant d'être présenté à la machine doit être transcrit sous forme binaire, c'est le **codage**.
De même le résultat donné sous forme binaire par le système de traitement de l'information doit être transcrit dans le langage original seul exploitable par l'homme c'est le **décodage**.

Enfin le système de traitement de l'information peut avoir à travailler avec différents codes binaires c'est le **transcodage**.

TRANSCODAGE

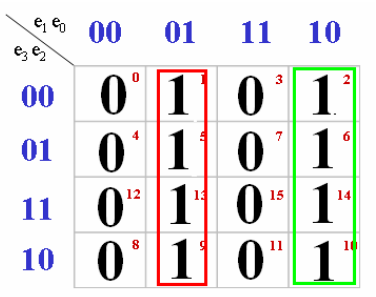
$ Transcodeur Binaire / GRAY



$ Analyse

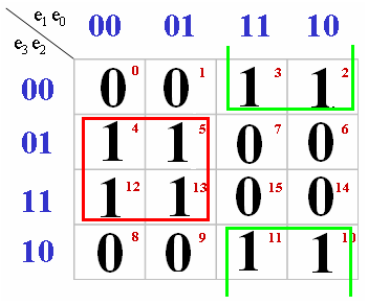
Binaire				GRAY			
e_3	e_2	e_1	e_0	S_3	S_2	S_1	S_0
0	0	0	0	0	0	0	0
0	0	0	1	0	0	0	1
0	0	1	0	0	0	1	1
0	0	1	1	0	0	1	0
0	1	0	0	0	1	1	0
0	1	0	1	0	1	1	1
0	1	1	0	0	1	0	1
0	1	1	1	0	1	0	0
1	0	0	0	1	1	0	0
1	0	0	1	1	1	0	1
1	0	1	0	1	1	1	1
1	0	1	1	1	1	1	0
1	1	0	0	1	0	1	0
1	1	0	1	1	0	1	1
1	1	1	0	1	0	0	1
1	1	1	1	1	0	0	0

$ Equations



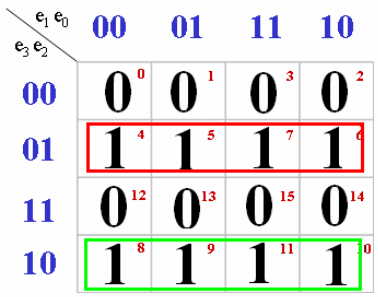
$e_3 e_2$ \ $e_1 e_0$	00	01	11	10
00	0	1	0	1
01	0	1	0	1
11	0	1	0	1
10	0	1	0	1

$S_0 = /e_0\, e_1 + e_0\, /e_1 = e_0 \oplus e_1$



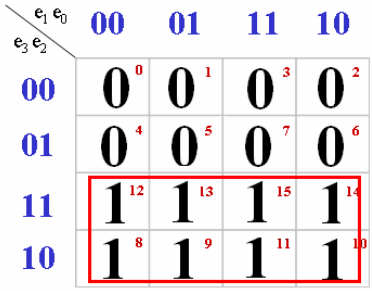
$e_3 e_2$ \ $e_1 e_0$	00	01	11	10
00	0	0	1	1
01	1	1	0	0
11	1	1	0	0
10	0	0	1	1

$S_1 = /e_1\, e_2 + e_1\, /e_2 = e_1 \oplus e_2$



$e_3 e_2$ \ $e_1 e_0$	00	01	11	10
00	0	0	0	0
01	1	1	1	1
11	0	0	0	0
10	1	1	1	1

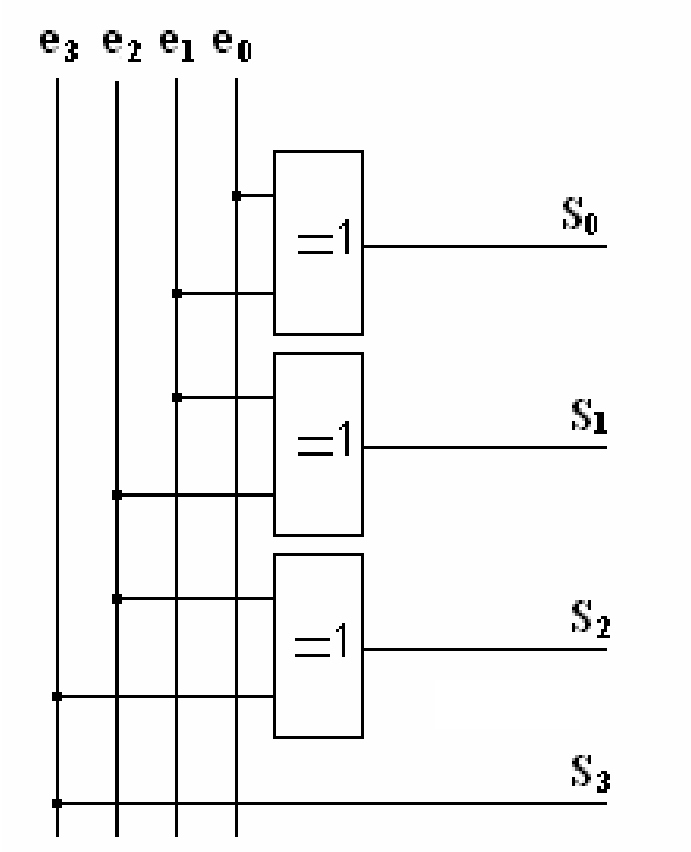
$S_2 = e_2\, /e_3 + /e_2\, e_3 = e_2 \oplus e_3$



$e_3 e_2$ \ $e_1 e_0$	00	01	11	10
00	0	0	0	0
01	0	0	0	0
11	1	1	1	1
10	1	1	1	1

$S_3 = e_3$

$ Schéma

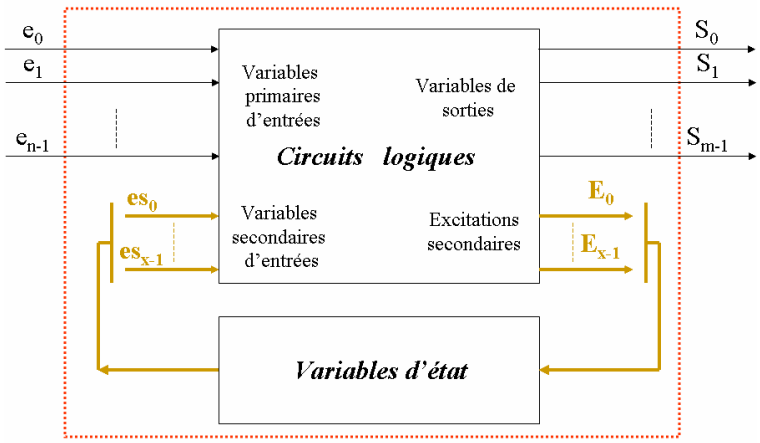


III. LOGIQUE SEQUENTIELLE

III.1. Définition

Dans un système combinatoire, la valeur de la sortie dépend uniquement des entrées, il ne garde pas en mémoire ce qui c'est passé avant. Dans un système séquentiel, la valeur de la sortie dépend des entrées et du temps, il y a une mémoire. $S = f(e, t)$

On ne va pas garder la variable **t** comme continue, on va la quantifier et s'intéresser au "temps" comme un ordre de succession. On va donc créer des variables supplémentaires que l'on bouclera sur les entrées par l'intermédiaire d'un bloc mémoire.



On aura un système à (n + x) entrées et à m sorties.

L'état de la sortie à l'instant (t), dépend de l'histoire antérieure du système. Les excitations secondaires E(t), misent en place dans la nouvelle commande permettent de mémoriser cette histoire.
Les vecteurs d'entrées e(t) et de sorties S(t) sont les représentations des informations respectivement acquises du procédé par les capteurs et qui lui sont appliquées par les actionneurs. Les vecteurs d'état es(t) résument le passé du procédé et sa situation présente. L'état est en quelque sorte la mémoire du système.

III.2. Equations

La logique séquentielle demandera deux types d'équations:

- *des équations caractéristiques des sorties du procédé*
- *des équations caractéristiques de l'état du système à l'instant (t).*

Ces deux types d'équations logiques auront la forme suivante:

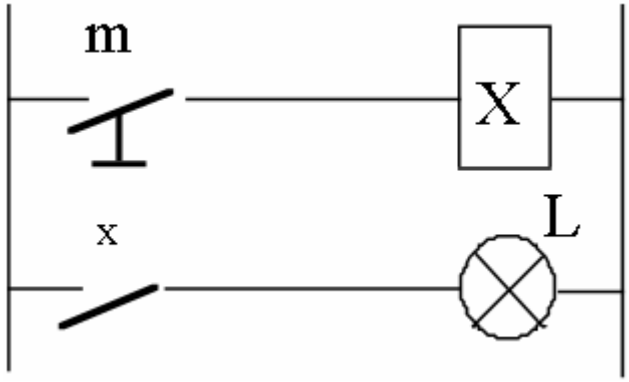
$$S_i = f(e_0, e_1, \ldots. e_{n-1}, e_{s0}, \ldots.. e_{sx-1})$$

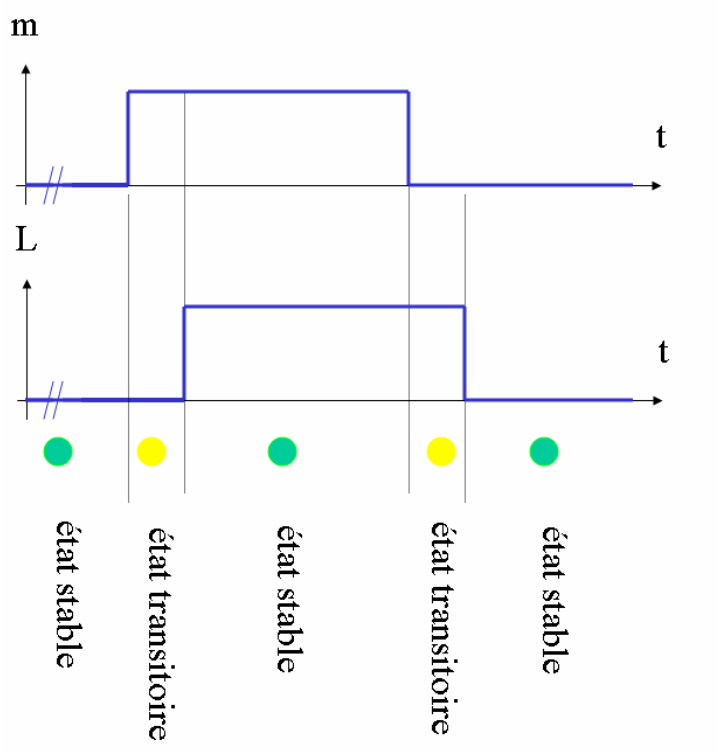
$$E_i = f(e_0, e_1, \ldots. e_{n-1}, e_{s0}, \ldots.. e_{sx-1})$$

S(t): variable de sorties du système
e(t): variable primaire d'entrée du système
E(t): Excitation ou fonction secondaire
es(t): variable secondaire d'entrée générée par la fonction secondaire

III.3. Etat stables et état transitoires

$ Etude d'un relais monostable





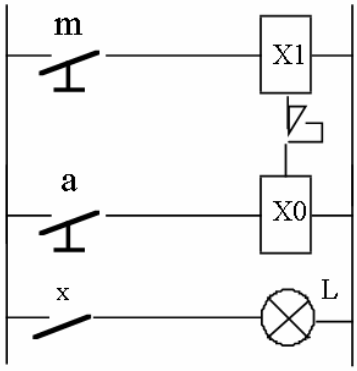
Dans un automatisme assurant une succession de séquences il est indispensable de considérer les retards apportés par le temps de réponse des différents organes.

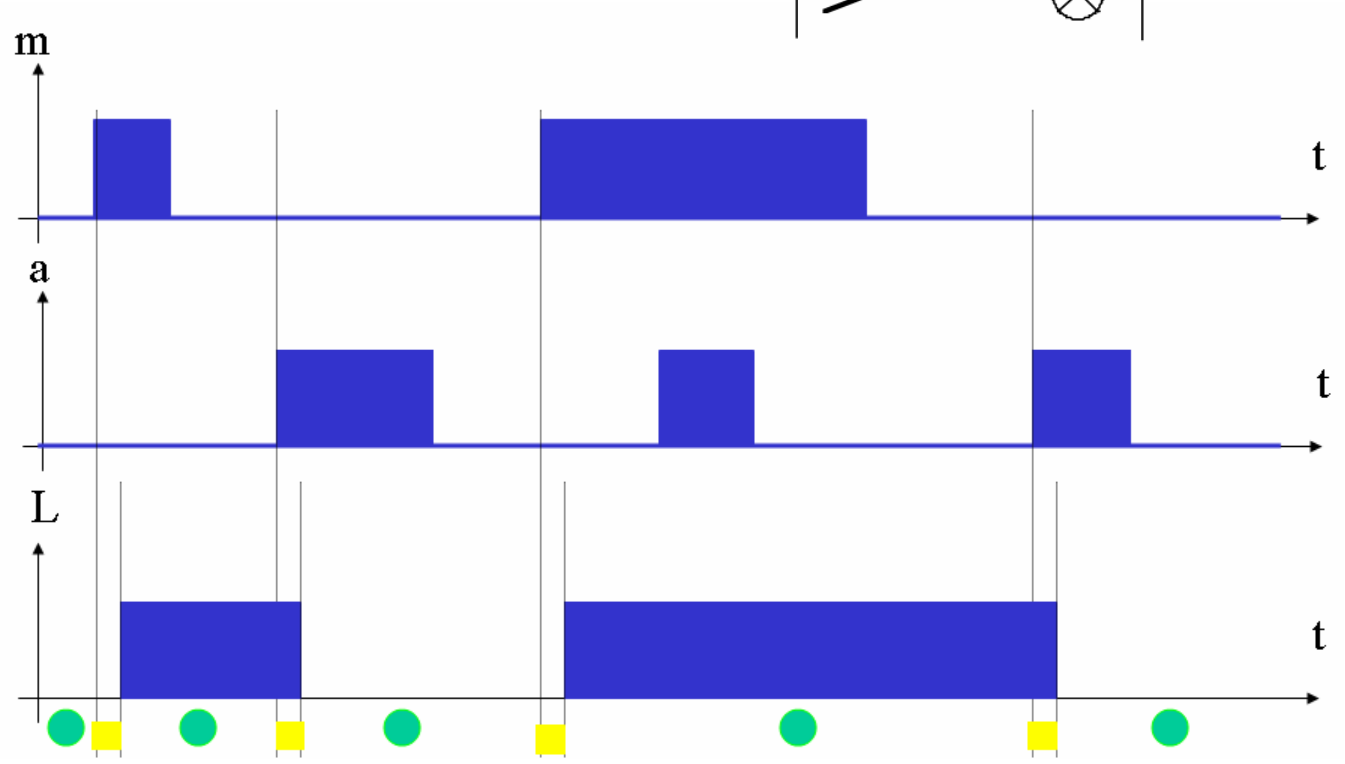
Dans l'exemple ci-contre, on considère:

- les **états stables**: la valeur de la variable d'entrée et celle de la variable de sortie sont constantes dans le temps.

- Les **états transitoires**: la valeur de la variable de sortie va rejoindre la valeur logique de la variable d'entrée.

$ Etude d'un relais bistable





Etat stable Etat transitoire

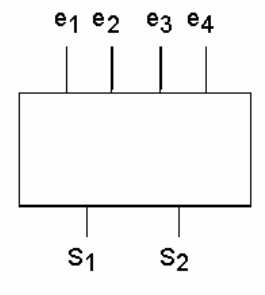
III.4. Représentation des fonctions séquentielles

$ Chronogramme ou diagramme des temps
$ Table de vérité
$ Grafcet
$ Réseau de PETRI
etc .....

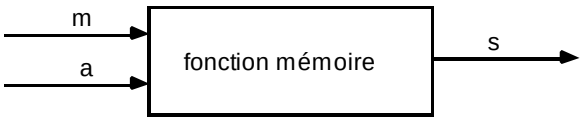
III.5. Fonctions séquentielles asynchrones

III.5.1.Définition

Dans une fonction séquentielle asynchrone chaque nouvelle combinaison des variables d'entrées est aussitôt prise en compte par le circuit.



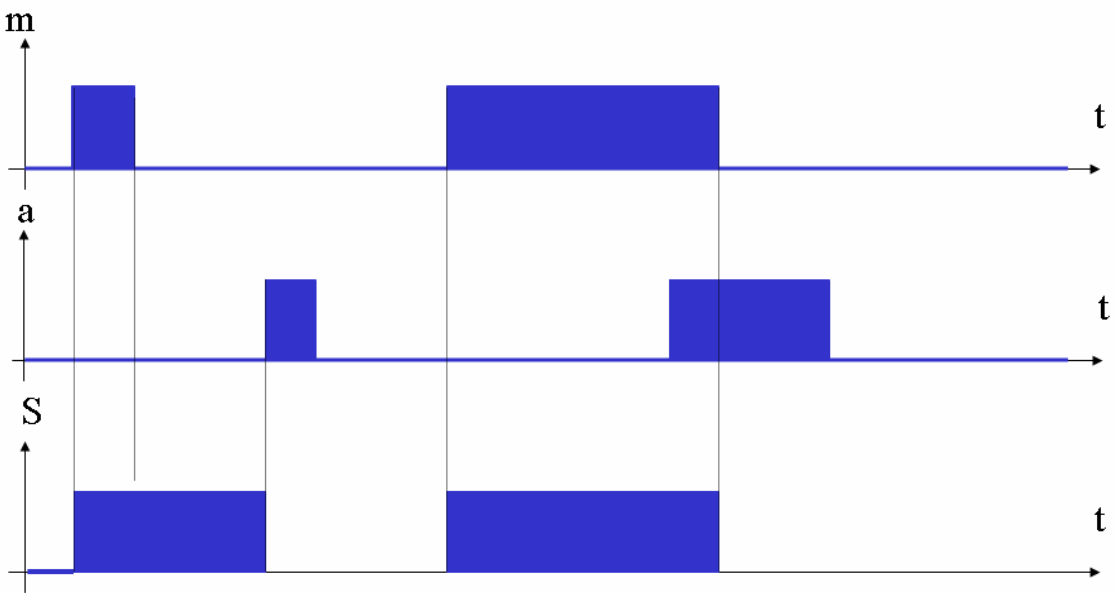
III.5.2.Mémoire monostable



Remarque : On supposera que l'intervalle de temps qui sépare deux changements d'états des entrées **m** et **a** est toujours supérieur au temps de réponse global de l'ensemble de la mémoire.

1 er CAS Mémoire à marche prioritaire

$ Traduction du cahier des charges sous forme logique (diagramme des temps).



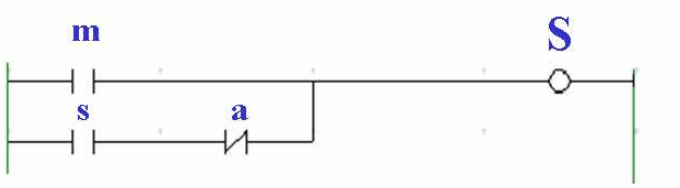
$ Analyse du fonctionnement mise en équation

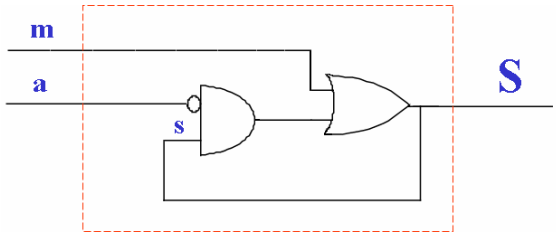
La sortie **S** est active quand l'information **m** apparaît : **S = m**

La sortie **S** reste active quand l'information **m** disparaît :
nécessité de se valider elle même **S = m + s**

La sortie **S** est désactive quand l'information **a** apparaît ($\bar{a}$), sauf si **m** est encore actif :
$\bar{a}$ n'intervient que sur l'auto-alimentation **S = m + s . $\bar{a}$**

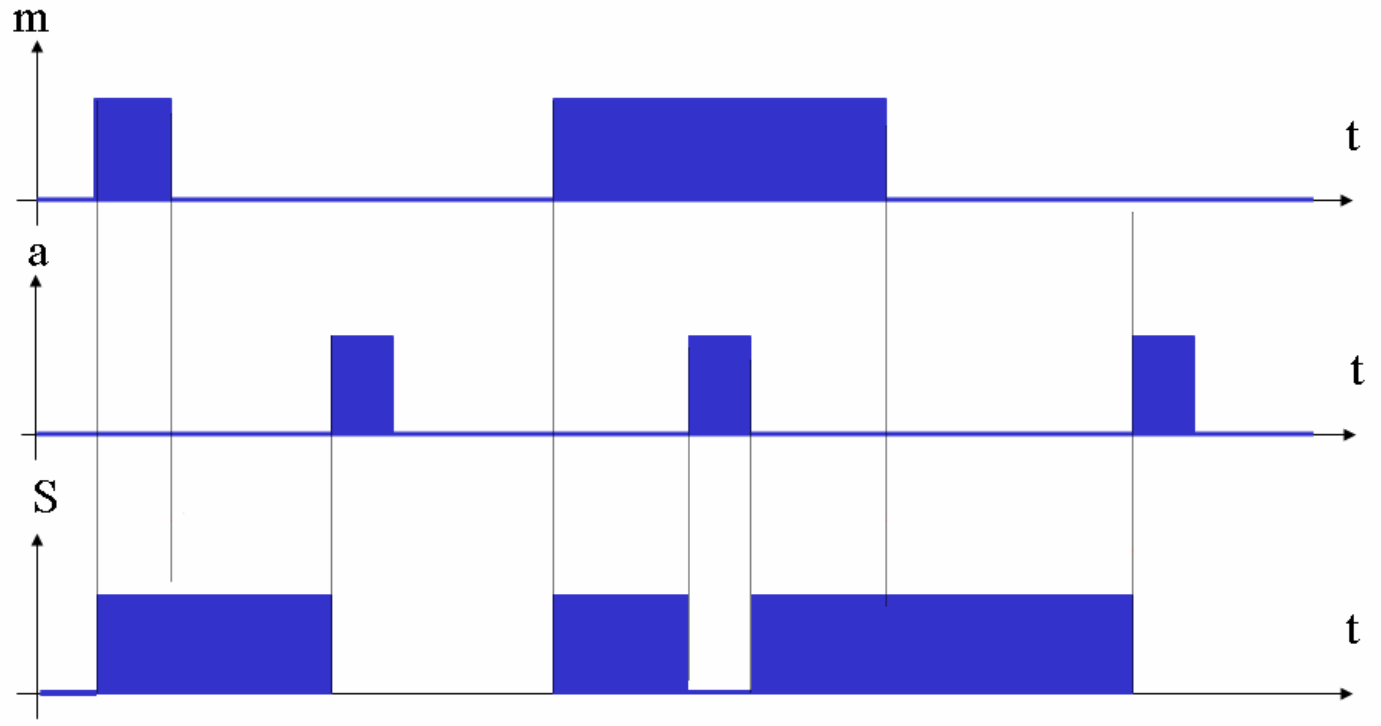
$ Schéma





2 em CAS Mémoire à arrêt prioritaire

$ Traduction du cahier des charges sous une forme logique (diagramme des temps).



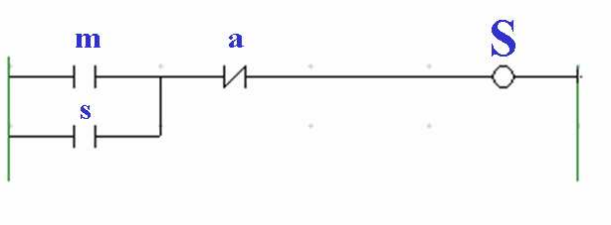
$ Analyse du fonctionnement mise en équation

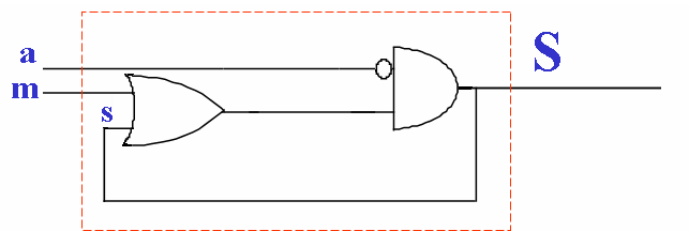
La sortie **S** est active quand l'information **m** apparaît : **S = m**

La sortie **S** reste active quand l'information **m** disparaît :
nécessité de se valider elle même **S = m + s**

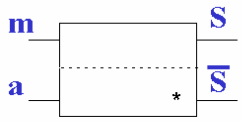
La sortie **S** est désactive quand l'information **a** apparaît ($\bar{a}$), quelquesoit l'état de **m** :
$\bar{a}$ intervient sur **m** et sur l'auto-alimentation **S = (m + s). $\bar{a}$**

$ Schéma

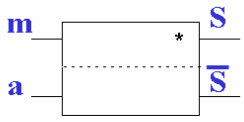




$ Symbole normalisé

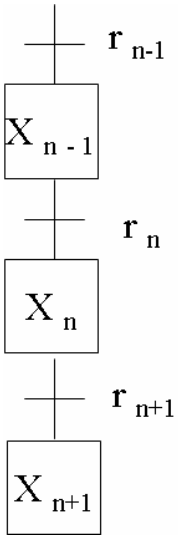


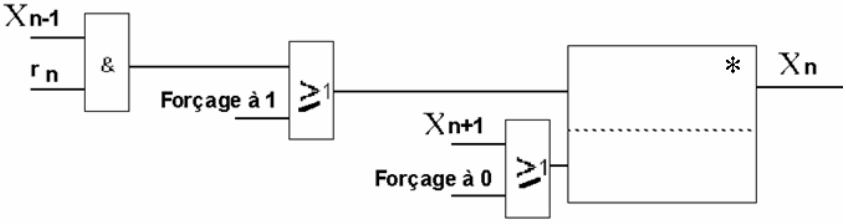
Arrêt prioritaire



Marche prioritaire

$ Exemple d'application

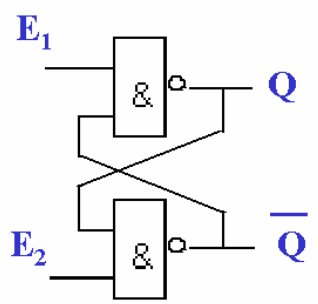




III.5.3.Mémoire bistable « Bascule /R/S / Bascule R S »

Bascule /R /S

$ Schéma

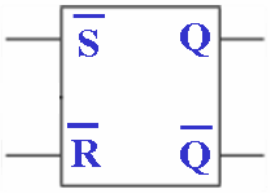


$ Fonctionnement

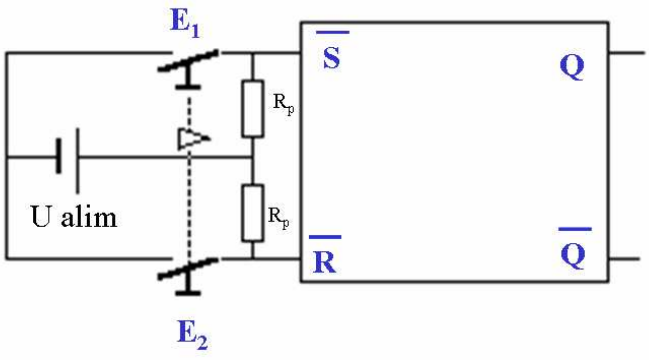
Table de vérité

Entrées		Sorties		
E1	E2	Q	$\overline{Q}$	
0	0	1	1	Combinaison à interdire
0	1	1	0	Mise à 1
1	0	0	1	Mise à 0
1	1	0 1	1 0	Mémorisation de l'état précédent donc 0 est le niveau actif des entrées.

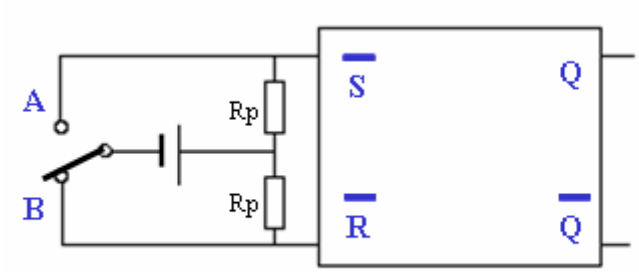
$ Symbole normalisé

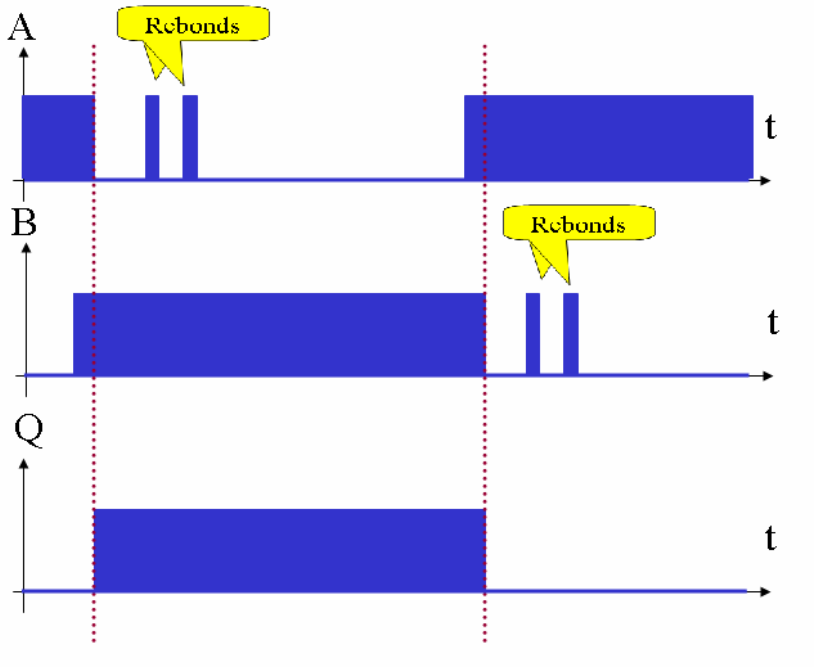


$ Montage d'application

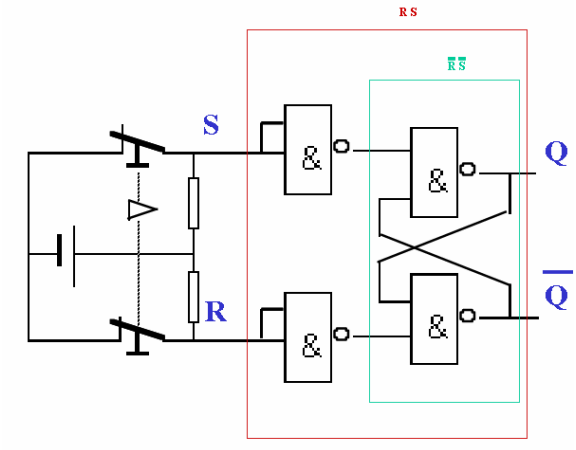


$ Utilisation de la bascule RS (montage anti rebonds)

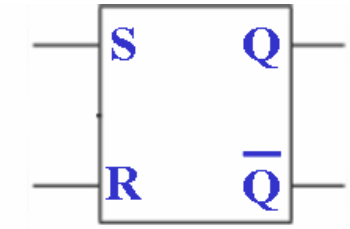




Bascule RS

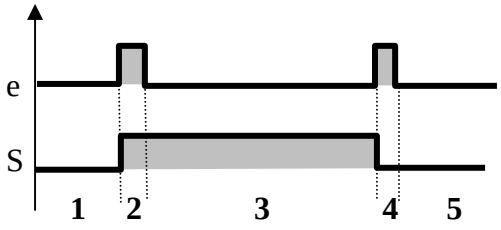


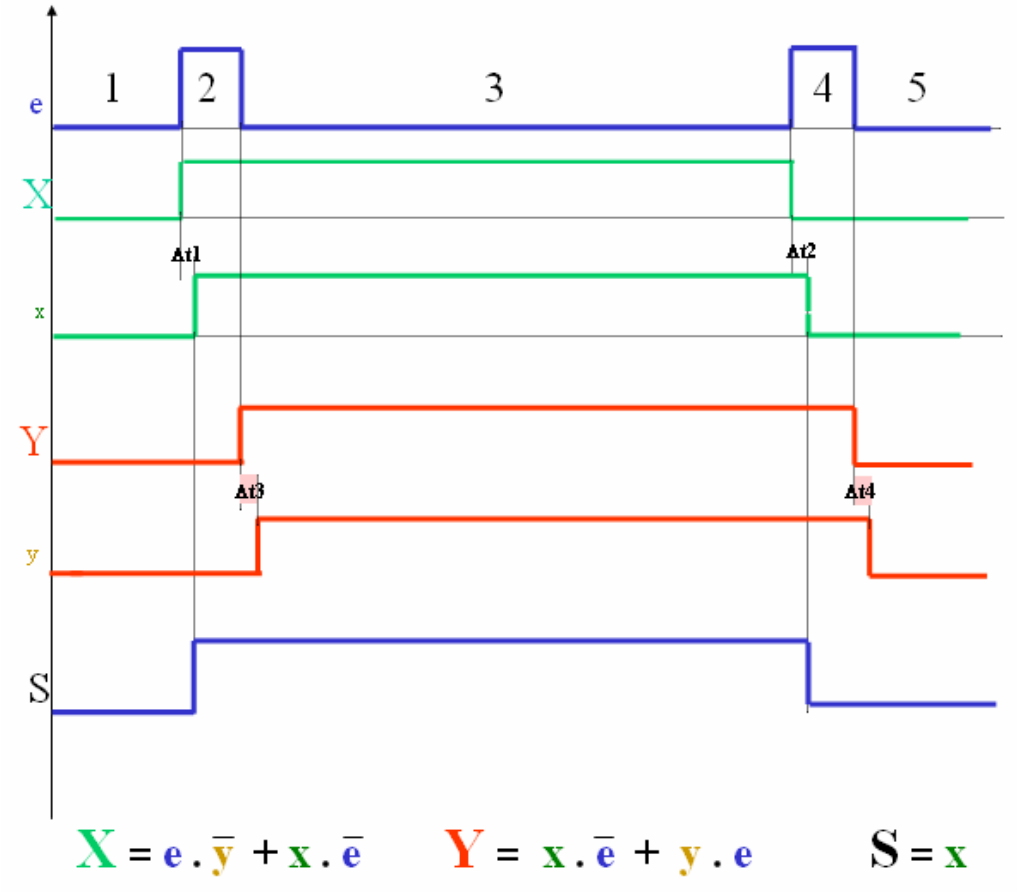
$ Symbole normalisé



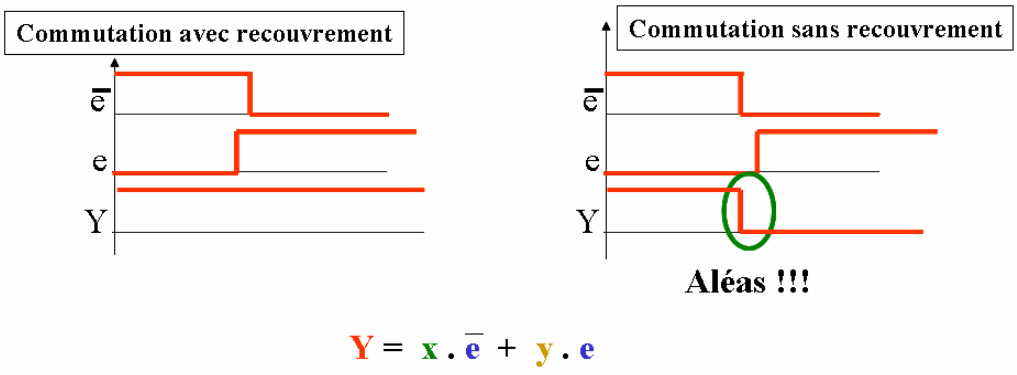
III.5.4. Aléas de fonctionnement

	1	2	3	4	5
e	0	1	0	1	0
S	0	1	1	0	0





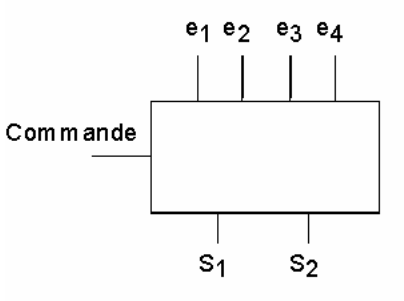
Examinons le comportement de la fonction auxiliaire Y lors du passage de l'état 3 à l'état 4:
Les temps de propagation des signaux dans un circuit ajoutés aux temps de commutation font apparaître des aléas de fonctionnements qui, dans notre exemple pourraient faire évoluer la commande non pas de 3 vers 4 mais de 3 vers 2, remettant la fonction Y à zéro au lieu de la maintenir à 1.



III.6. Fonctions séquentielles synchrones

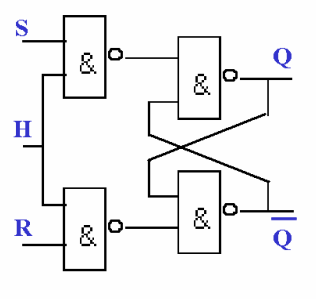
III.6.1.Définition

Dans une fonction séquentielle synchrone la prise en compte d'une nouvelle combinaison des variables d'entrées ne s'effectue que sur l'ordre d'une entrée de commande.
On maîtrise maintenant les instants de commutations à l'aide de l'entrée de commande.
Les circuits synchrones ne sont pas sujets aux aléas.

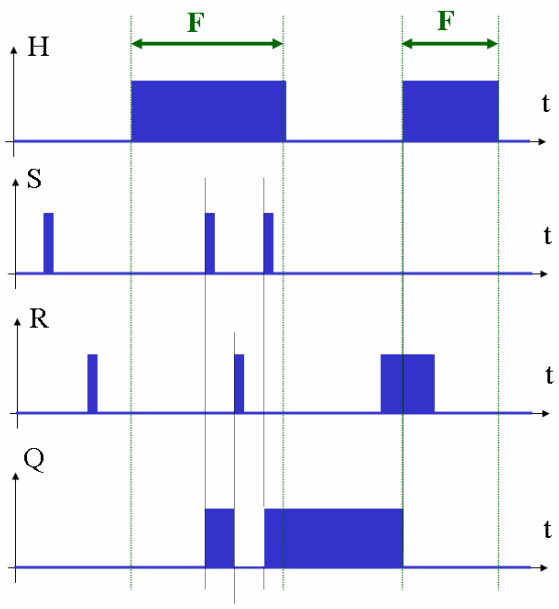


III.6.2. Bascule RSH

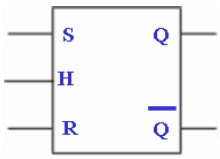
$ Schéma



$ Fonctionnement

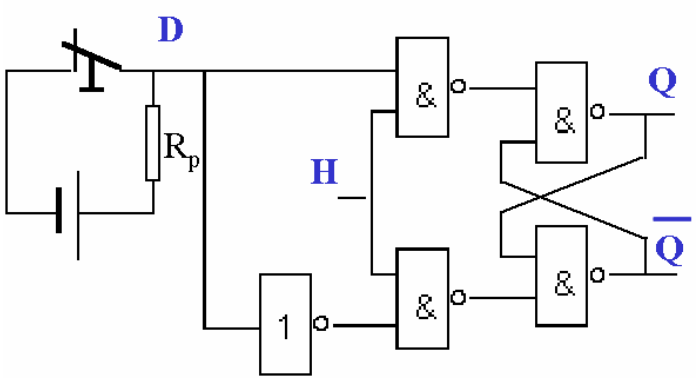


$ Symbole

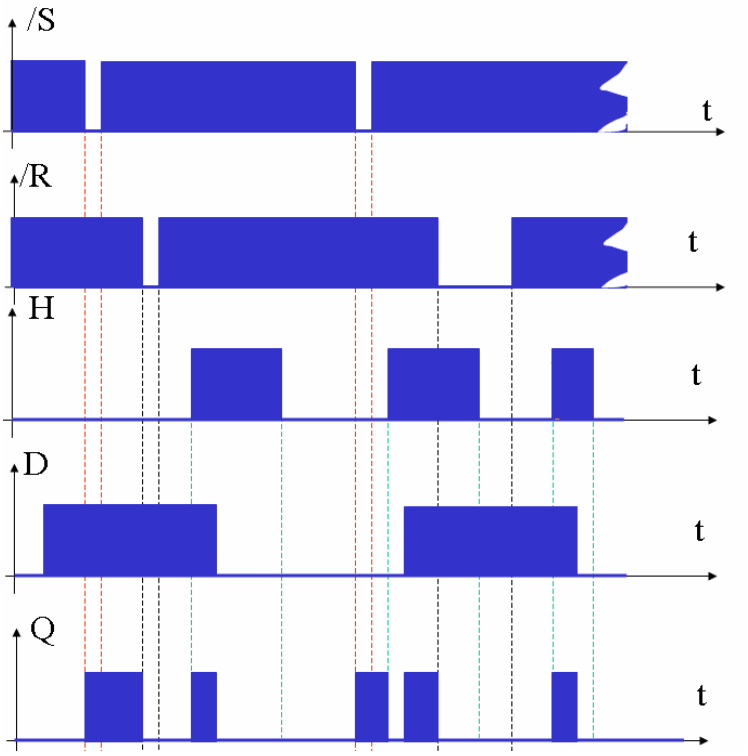


III.6.3. Bascule D

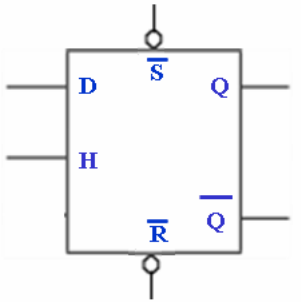
$ Schéma



$ Fonctionnement



$ Symbole



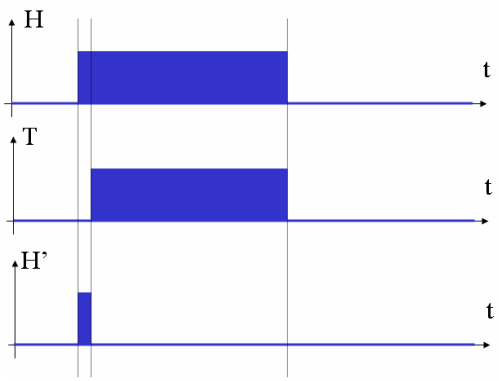
Cette bascule D est active sur le niveau 1 du signal de commande ' H 'elle est dite *bascule D type LACH*

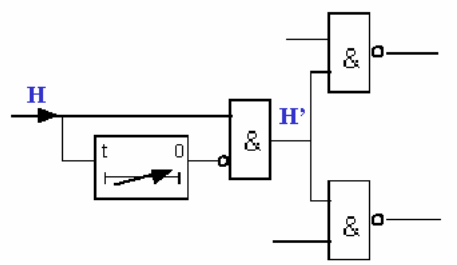
Remarque

Il existe aussi une bascule D qui n'est active uniquement sur le front montant ou le front descendant du signal de commande ' H '.

Elle est dite bascule **D type EDGE.**

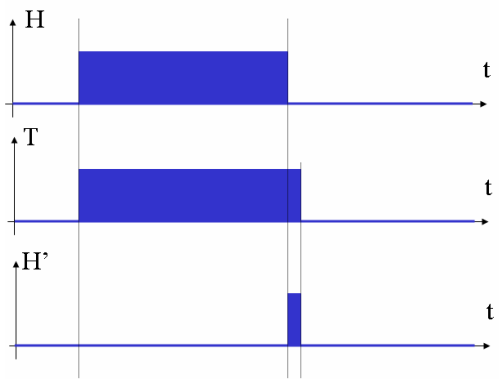
Création d'un front montant

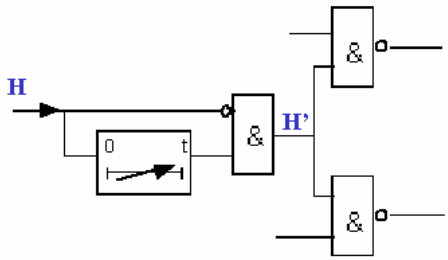




$$H' = H \cdot \overline{T}$$

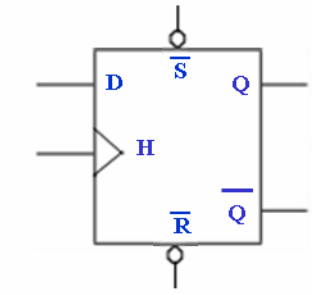
Création d'un front descendant



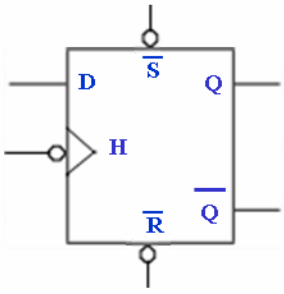


$$H' = \overline{H} \cdot T$$

$ Symboles

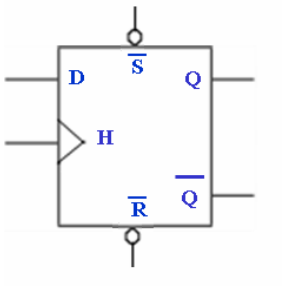


Bascule D à front montant



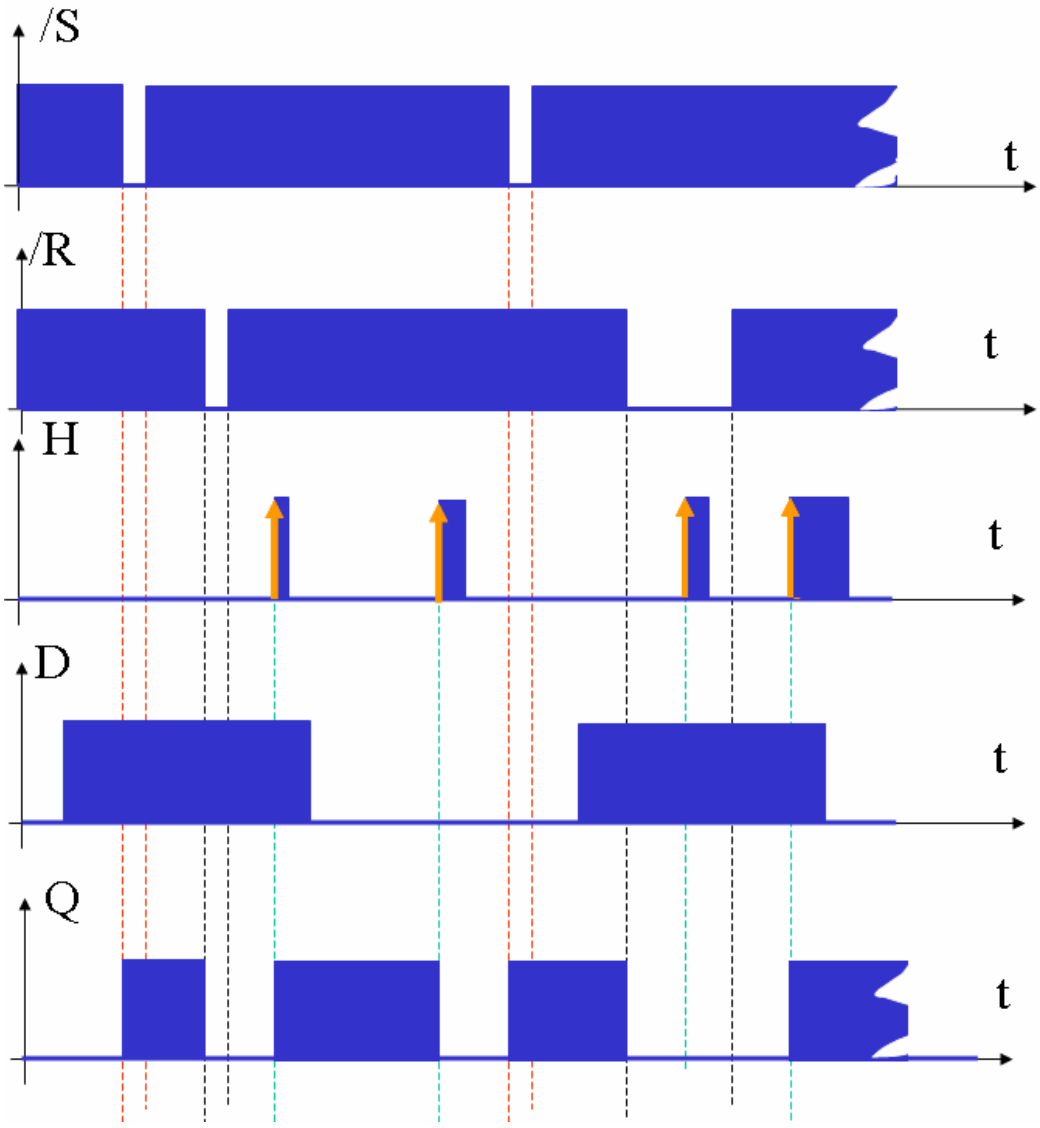
Bascule D à front descendant

$ Fonctionnement



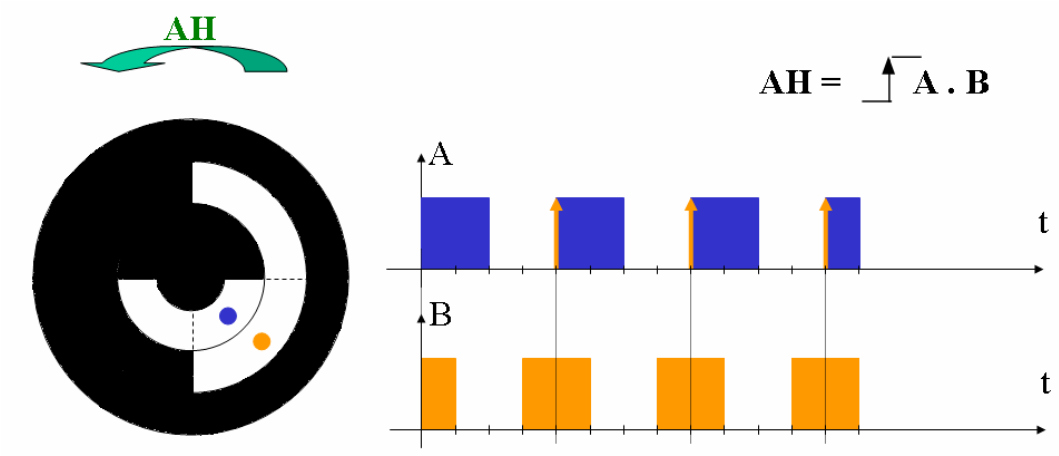
L'entrée **D** est une entrée synchrone, elle n'est prise en compte qu'au front de H

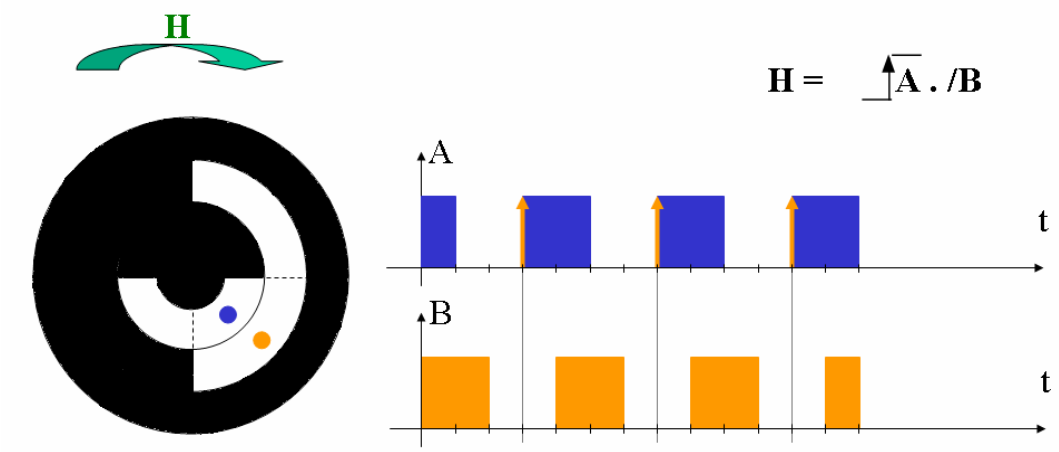
Les entrées **S** et **R** sont des entrées asynchrones, elles sont prises en compte dès quelles sont actives. **Indépendamment de H !**

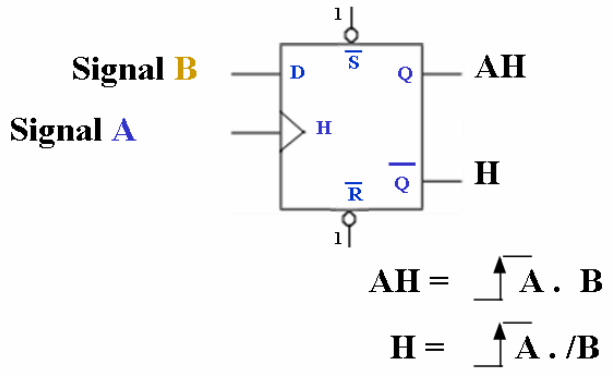


Exemple d'application

Détection du sens de déplacement d'un système





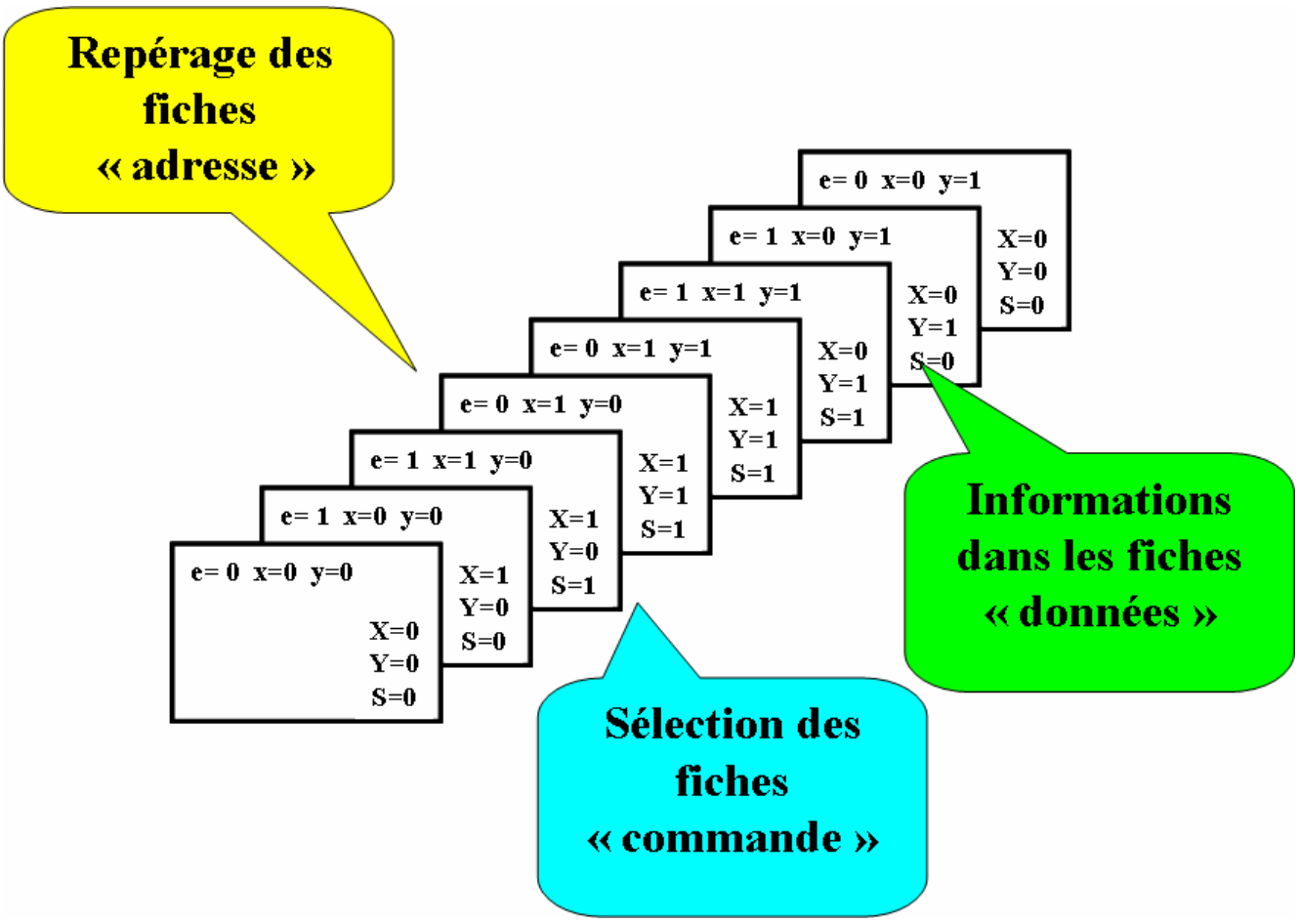


IV. LOGIQUE PROGRAMMEE

Supposons qu'un opérateur réalise manuellement la commande définie par le chronogramme étudié page 15.
Il devra, à intervalles de temps réguliers, observer l'état des entrées e, x, y puis, suivant cet état, effectuer ou non la mise à jour des sorties S, X, Y. Pour se faciliter le travail, il va se constituer un fichier sur lequel il portera les différentes combinaisons des entrées et les états des sorties correspondants. Il ne lui restera plus alors qu'a :

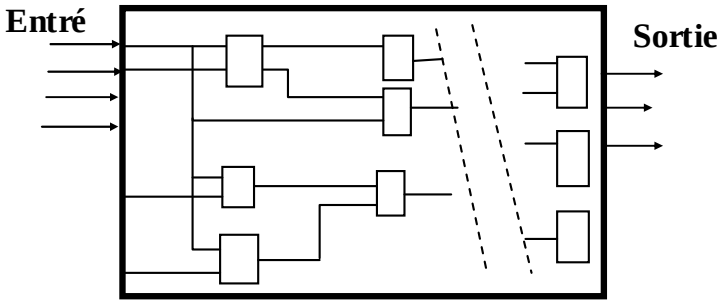
- repérer la fiche adéquate
- observer l'état présent des entrées
- lire la fiche et mettre les sorties dans l'état indiqué
- passer à la fiche suivante

Chaque fiche est repérée par un numéro de fiche que l'on appelle "adresse". La combinaison représentant les entrées et les sorties est la partie "donnée".
L'automatisation de la commande nécessite la sélection automatique de cette adresse à intervalles de temps réguliers. Fixer cet intervalle demande l'emploi d'une horloge commandant un élément de stockage de l'adresse. Cet élément est réalisé par des registres synchrones. Le fichier est réalisé par des circuits séquentiels appelés "mémoires".

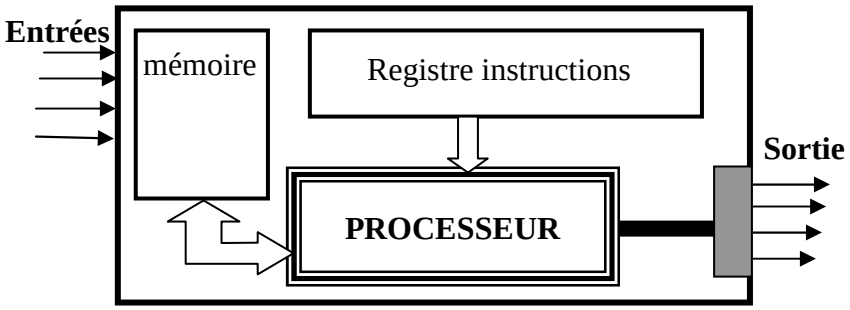


IV.1. Traitements parallèles ou séquentiels

Le traitement est dit parallèle lorsque tous les signaux sont pris en compte simultanément par l'organe de traitement. C'est le cas de la logique câblée. En adoptant cette représentation simplificatrice, on distingue que le traitement parallèle recèle un certain degré de séquentiel provoqué par les transitions de couche à couche.



Un traitement est dit séquentiel lorsque les signaux concernés à un instant donné sont traités successivement, dans un ordre prédéfini. Le principe de la logique programmée est de substituer aux différentes couches d'une logique câblée, une seule couche appelée "PROCESSEUR" capable de réaliser les fonctions de toutes les couches. Ce processeur exécute des instructions qui lui indiquent quelle fonction il doit réaliser à un instant donné et sur quels signaux. Le processeur est donc lui-même une logique séquentielle synchrone. Sa réalisation est matérialisée par un MICROPROCESSEUR.



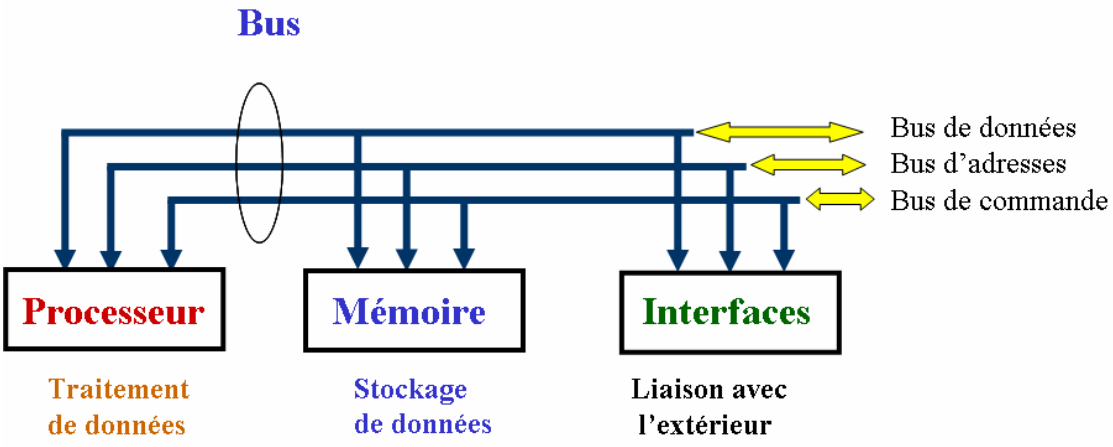
Traitement séquentiel

A chaque top d'horloge, le processeur exécute une instruction et une seule. Les données doivent être mémorisées afin d'être disponibles lorsque leur tour de traitement viendra. La manière de traiter ces données, c'est à dire les instructions ou ensemble d'ordre commandant les fonctions du processeur, est également conservée en mémoire.
Les logiques programmables sont dites séquentielles car elles traitent séquentiellement les instructions, mémorisées, donc les signaux représentés par les données.

IV.2. Système Minimum

Pour permettre le fonctionnement défini, un système programmable doit au minimum être composé

- PROCESSEUR susceptible de se charger de l'exécution du programme.
- MEMOIRE permettant de stocker les informations nécessaires au travail du processeur.
- INTERFACES d'entrées / sorties chargées des relations du système avec les périphériques.
- BUS de liaison entre les trois autres éléments pour la communication et les échanges d'informations.



V.STRUCTURE D'UN API

Un automate programmable industriel (API) est une machine électronique, programmable par un personnel non informaticien et destiné à piloter, en temps réel et en ambiance industrielle, des procédés logiques séquentiels et combinatoires. (Norme NF C 63-850).
C'est un outil :

- adapté vis à vis de la partie opérative. Entrées/Sorties industrielles, traitement de fonctions logiques courantes (séquentielles et/ou combinatoires),
- coupleurs spécialisés, délocalisation,
- adapté vis à vis de l'environnement, température, humidité, vibration, parasites électriques et électromagnétiques,
- adapté aux tâches de conception et de réalisation. Par un outil de dialogue permettant de décrire et d'implanter l'application dans un langage simplifié adapté aux tâches de mise au point,
- Outil de dialogue permettant l'édition et la modification ponctuelle, le transfert de programme, la simulation, la visualisation dynamique, etc …
- adapté aux tâches d'exploitation en mode normal et en mode dégradé.

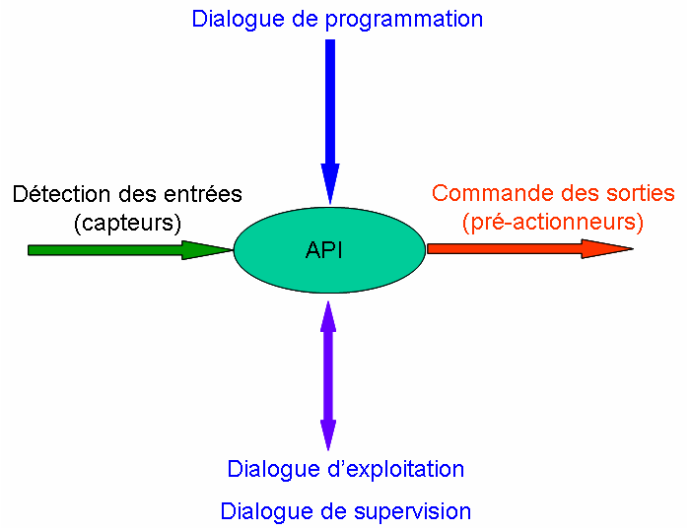
V.1. Structure Fonctionnelle

Les 5 fonctions principales autour d'un automate programmable sont

- la détection depuis des capteurs répartis sur la machine,
- la commande d'actions vers les pré-actionneurs et les actionneurs,
- le dialogue d'exploitation,
- le dialogue de programmation,
- le dialogue de supervision de production.

Ces fonctions utilisent des moyens de communication différents selon leur spécificité.

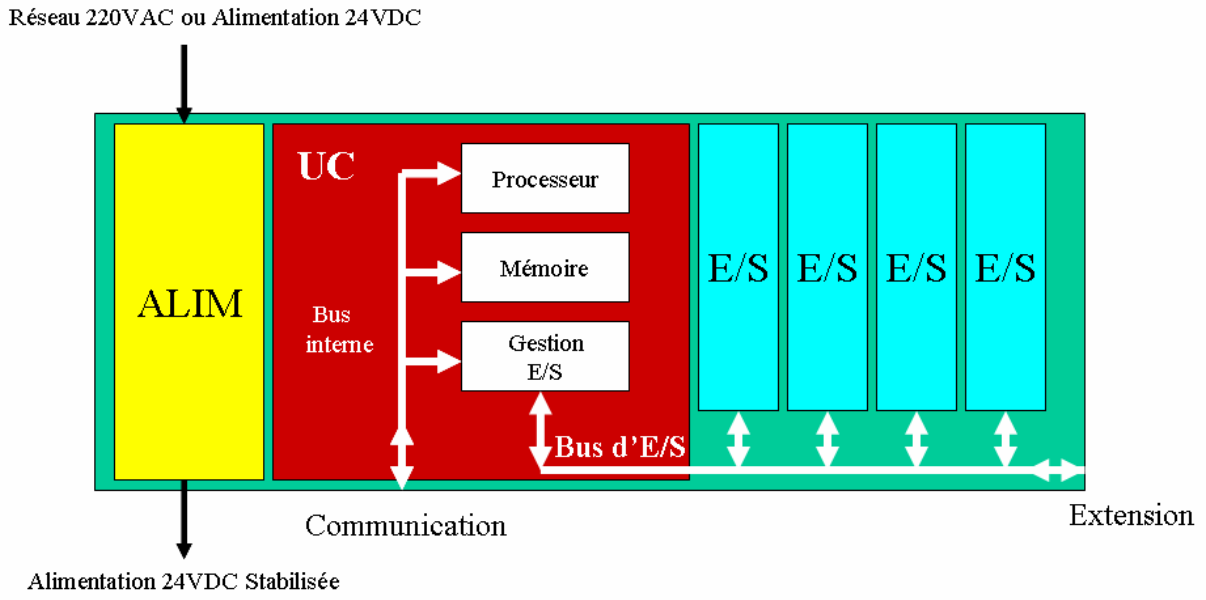
- soit des liaisons "Fil à Fil" pour des modules d'entrées sorties logiques (TOR) par exemple,
- soit des liaisons "séries" ou "parallèles" pour des coupleurs spécialisés et pour des modules de programmation.



V.2.Structure Matérielle

L'API se présente sous forme d'un ensemble de blocs fonctionnels s'articulant autour d'un canal de communication : LE BUS.
Cette organisation modulaire permet une grande souplesse de configuration. Il en résulte que, pour une application particulière, un automate doit être équipé d'un certain nombre de constituants capables de satisfaire aux spécifications imposées.
Cette organisation modulaire se traduit par une structure matérielle générale comprenant :

- un RACK de base constituant l'ossature métallique d'un API avec la carte "Fond de Panier" (BUS + connecteurs)
- une ALIMENTATION chargée de fournir l'énergie nécessaire au bon fonctionnement de l'automate.
- une **U**NITE **C**ENTRALE ou **UC** comprenant le processeur, élément fondamental de l'API et la CARTE MEMOIRE séparée ou intégrée à l'unité centrale et un Gestionnaire d'Entrées / Sorties.
- des COUPLEURS d'Entrées /Sorties de type TOR ou spécialisés, suivant l'application.



V.2.1 Le rack

Il s'agit de structures métalliques pouvant recevoir un certain nombre de cartes électroniques de même dimension, selon un pas donné .Un équipement peut se présenter sous la forme:

- un système compact, fermé avec un nombre d'E/S défini par le constructeur, généralement peu important et réservé aux API bas de gamme.
- un rack principal avec un (ou plusieurs) rack d'extension liés par le bus de l'API ou par coupleurs d'extensions si le nombre d'entrées / sorties est important.

V.2.2 Le module d'alimentation

Il permet de fournir l'énergie nécessaire au bon fonctionnement de l'automate.
Les principales tensions utilisées dans un API sont du +12v et ± 5v adaptés au fonctionnement des cartes électroniques internes.
Pour pouvoir agir directement sur la PO en cas de défaillance, certains modules d'alimentation sont équipées d'un contact et d'un voyant.
Le module d'alimentation à un emplacement réservé dans le rack principal.
On distingue :

- le module d'alimentation alternatif qui fourni l'energie nécessaire à partir du secteur 220V.
- le module d'alimentation continu qui fourni l'energie nécessaire à partir d'alimentations externes (24 ou 48V).

Le choix du module d'alimentation d'un automate programmable se fera à partir de sa configuration et d'un bilan énergétique des consommations des coupleurs installés.

V.2.3 L'Unité Centrale

L'Unité Centrale est le cœur de l'automate, elle regroupe l'ensemble des dispositifs nécessaires au fonctionnement logique interne de l'API :

- le Processeur,
- la Mémoire,
- le Gestionnaire d'entrées / sorties.

Voir paragraphe correspondant

V.2.4 Les Coupleurs d'entrées /sorties

Les coupleurs d'entrées / sorties assurent la fiabilité des échanges des informations entre l'API et la partie opérative dans un milieu industriel fortement parasité.
Les constructeurs offrent une grande variété de coupleurs *voir paragraphe correspondant*.

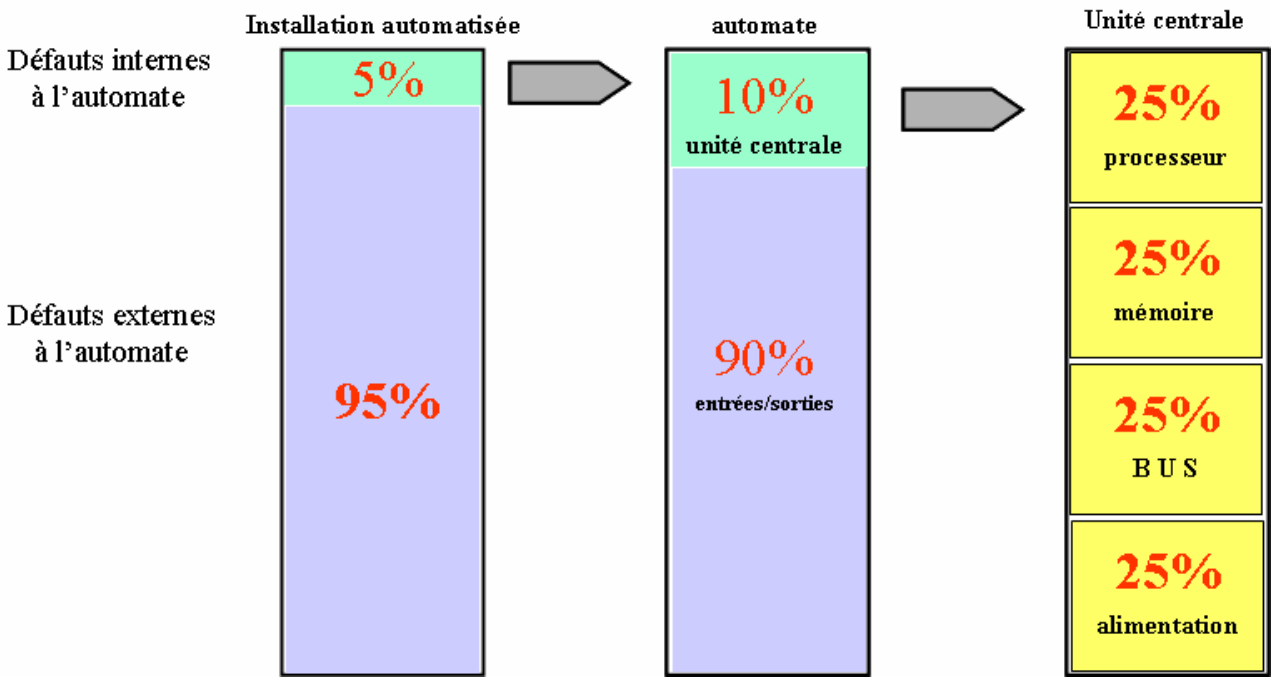
V.2.5 Le Bus

Les constructeurs d'API ont rapidement fait évoluer leurs machines vers une architecture multibus pour augmenter les performances de leurs systèmes. Nous identifierons les divers BUS suivant leur niveaux d'utilisation:

- Niveau 0: correspondant au bus du microprocesseur confiné à la carte d'unité centrale « Bus Interne ».
- Niveau 1: c'est par lui que s'effectuent les échanges entre cartes, c'est le « Bus fond de panier » ou « Bus d'entrées/sorties ».

VI.FIABILITE, SECURITE, DISPONIBILITE

La fiabilité d'un API se mesure à partir du taux de défaillance de chacun de ses constituants.



D'après une étude des constructeurs, 95% des défauts survenant dans une installation automatisée sont d'origine externe à l'automate. Parmi les 5% restants, 90% concernent les éléments d'entrées/sorties. Les 10% restants sont affectés à l'unité centrale et se répartissent de façon à peu pré uniforme entre le microprocesseur, la mémoire, le Bus, l'alimentation.
Cette analyse montre que :

- l'unité centrale de l'API est fiable et présente un haut niveau de disponibilité.
- que pour augmenter la disponibilité de l'installation dans son intégralité, il faut s'attaquer par des fonctions de diagnostics aux 95% des pannes occasionnées par les défauts extérieurs.

Disponibilité

La disponibilité est définie comme la rapport entre la somme des temps de production et du temps total pouvant être selon les cas considérée comme:

- temps de mise à disposition
- temps d'engagement
- temps normal de production

La disponibilité correspond à l'aptitude du système à réaliser sa fonction en permanence.

L'Automate est fiable et présente un haut niveau de disponibilité

L’Unité Centrale

I.INTRODUCTION

Le chapitre précédent a montré que la fonction "traiter les données" d'un système automatisé était dévolue à l'Automate Programmable (API).
L'analyse fonctionnelle globale de l'API montre la réalisation de deux fonctions principales

- Fonction de traitement des données essentiellement exécutée par l'Unité Centrale.
- Fonction de traitement des signaux exécutée par les coupleurs d'entrées/sorties.

l'UC présente une structure matérielle dérivée de la définition d'un système minimum programmable c'est à dire composé de

- le ***PROCESSEUR*** pour traiter les données
- la ***MEMOIRE*** pour le stockage
- le ***GESTIONNAIRE D'E/S*** pour la gestion des E/S industrielles

Tous les échanges d'informations entre ces différents éléments se font par le BUS. La machine est dite *UNIBUS* lorsque toutes les informations transitent par un seul BUS. Cependant, les performances des API modernes sont liées en général à une structure ***MULTIBUS***, le cas le plus courant consiste à réserver un BUS d'E/S pour les échanges avec l'extérieur.
L'Unité Centrale d'un API doit savoir :

- traiter des variables binaires (travail sur BIT) mais aussi permettre le travail sur MOT pour le traitement numérique et analogique.
- travailler en temps réel ou du moins avoir un temps de cycle très court.
- interpréter le langage utilisateur.

Actuellement la majorité des API sont à base de microprocesseur. L'Unité Centrale est également caractérisée par :

- le nombre et le type des instructions, fonctions de base ou optionnelles (le jeu d'instruction),
- le format des données et des instructions,
- le mode de fonctionnement et la vitesse de traitement d'une instruction,
- la capacité d'adressage mémoire.

II.LES FONCTIONS LOGICIELLES DE L'UC

II.1. Logiciel de base

Le logiciel de base à pour objet de permettre la mise en œuvre du matériel selon les spécifications indiquées par l'utilisateur au moyen du langage de programmation. Ce logiciel est, sauf exception, livré avec chaque machine par le constructeur. Le logiciel de base est constitué de l'ensemble des programmes destinés à permettre ou à faciliter la mise en œuvre du matériel pour la production et l'exploitation des applications. Il se présente comme une couche logicielle entre le matériel et l'extérieur. Cette couche est composée du système d'exploitation et du logiciel de production de programmes.

II.2. Logiciel d'exploitation

Le logiciel d'exploitation (ou système) est constitué du sous-ensemble du logiciel de base nécessaire et suffisant à l'exploitation des applications. Cette mission revêt deux aspects :

- la satisfaction des sollicitations provenant des applications
- la gestion interne de la machine.

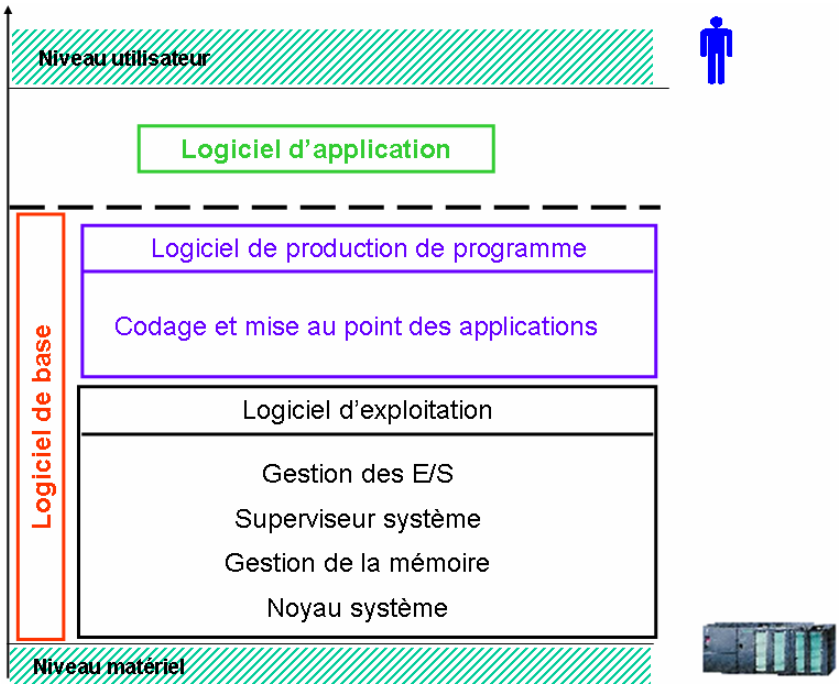
On distingue différents niveaux de systèmes d'exploitation selon leurs capacités:
Mono-utilisateur lorsqu'ils ne supportent qu'un utilisateur, ils sont *multi-utilisateurs* dans le cas contraire
Monoprogrammation s'ils ne peuvent être simultanément en configuration d'exploitation et de développement de programme, *multiprogrammation* dans le cas contraire
Mono-tâche quand ils ne savent exploiter qu'une seule tâche, *multitâches* dans le cas inverse.
L'architecture du système d'exploitation est composée de tout ou partie des modules ou fonctions suivants

- le noyau du système d'exploitation, toujours résident, chargé de traduire les fonctions systèmes en actions matérielles
- le module de gestion de la mémoire
- le superviseur chargé d'administrer le système d'exploitation au regard des requêtes que lui adresse les programmes d'application
- des utilitaires spécialisés (communication, gestion des périphériques,…)

Son volume mémoire dépend des fonctions qu'il offre. On notera que de 1K mot au début il peut être actuellement de 20K mots et plus.

II.3. Logiciel de production de programme

Le logiciel de production des programmes est un sous-ensemble du logiciel de base nécessaire et suffisant au codage et à la mise au point des applications. Il travaille sous l'autorité du système d'exploitation, avec des contraintes de temps de réponse plus faible car l'opérateur est plus lent que le procédé.
L'ordinateur est une machine universelle, conçu pour exécuter un jeu d'instruction suffisamment riche pour répondre à l'ensemble des besoins en traitements logiques. Avec des langages de haut niveau (langages évolués) il permet l'accès à de grandes classes d'applications. Pour satisfaire certains besoins on trouve des langages temps réels, orientés problèmes, langages très évolués mais spécialisés et de ce fait accessible à des personnels non informaticiens, en particulier aux automaticiens.
Le langage des API est orienté problème. Mais, la machine elle-même est spécialisée, par sa conception, pour résoudre une gamme de problèmes particuliers. On dit que l'API est une machine orientée problème, ceci permet de déporter l'effort de développement vers le constructeur (coût et recherche) au profit de l'utilisateur qui profite de la souplesse des logiques programmables.



III.LA MEMOIRE

III.1. Définition

Une mémoire est un dispositif technologique destiné à conserver l'information. La mémoire élémentaire (ou point mémoire) mémorise une valeur binaire un **BIT** (0 ou 1).

III.2. Caractéristiques des mémoires

Les principaux paramètres caractéristiques des mémoires:

Le mode d'accès :
Accès aléatoire ou direct, l'information est stockée à une adresse précise que l'on atteint directement et rapidement en lui appliquant une information binaire correspondant à l'adresse sélectionnée.
Accès séquentiel; le temps d'accès est plus long que l'information se trouve stockée plus ou moins loin de l'organe de lecture /écriture.

Le mode de fonctionnement:
à lecture / écriture, permet l'inscription et le prélèvement d'une information, à lecture seule, les données sont écrites par le constructeur, et ne peuvent qu'être lues par l'utilisateur.

La volatilité :
une mémoire est volatile si elle perd les informations lors d'une coupure de son alimentation.

La vitesse :
temps d'accès: temps pour l'obtention d'une information après la demande de lecture.

La capacité :
nombre d'éléments binaire qui peuvent être stockés. L'information est rangée dans N registres de p éléments.

Capacité = N x p

1 Octet = 1 Byte = 8 Bits
1 KO = 2^{10} Octets = 1024 Octets = 1024 x 8 = 8192 Bits
1 MO = 1024 KO
1 GO = 1024 MO

III.3. Technologies

Différentes technologies de mémoires sont utilisées par les constructeurs d'API.

Mémoires ROM (mortes non volatiles)

- Non reprogrammable par l'utilisateur :PROM logiciel d'exploitation
- Reprogrammable par l'utilisateur :
  - REPROM après effacement aux UV
  - EEPROM reprogrammable par signal électrique.

Ce type de mémoire est utilisé pour le logiciel d'application.

Mémoires RAM (vives volatiles)

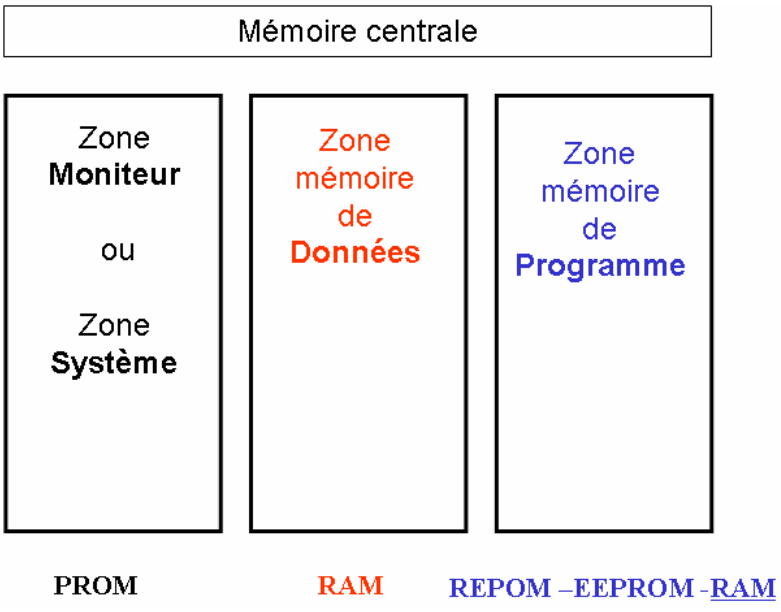
Elles peuvent être utilisées pour le programme d'application et/ou les données. Dans un API elles sont protégées par batteries ou pile au lithium (1000 heures). Elles ont l'avantage de pouvoir se programmer directement dans l'UC donc autorisent des outils de programmation plus simple.

III.4. Architecture de la mémoire

La mémoire d'un API quelle soit modulaire, extensible ou de capacité fixe, sert a stocker, d'une part le programme utilisateur, et d'autre part les données de fonctionnement. Un programme est une suite d'instruction écrite par l'utilisateur pour effectuer un certain travail. Les données constituent l'ensemble des informations et des valeurs qui caractérisent le fonctionnement, elles peuvent évoluer au cours de l'exécution du programme. La mémoire de l'API sera donc scindée en plusieurs zones. Ces zones seront physiquement distinctes ou non selon les constructeurs, car elles imposent des contraintes technologiques différentes.

III.4.1 zones de la mémoire

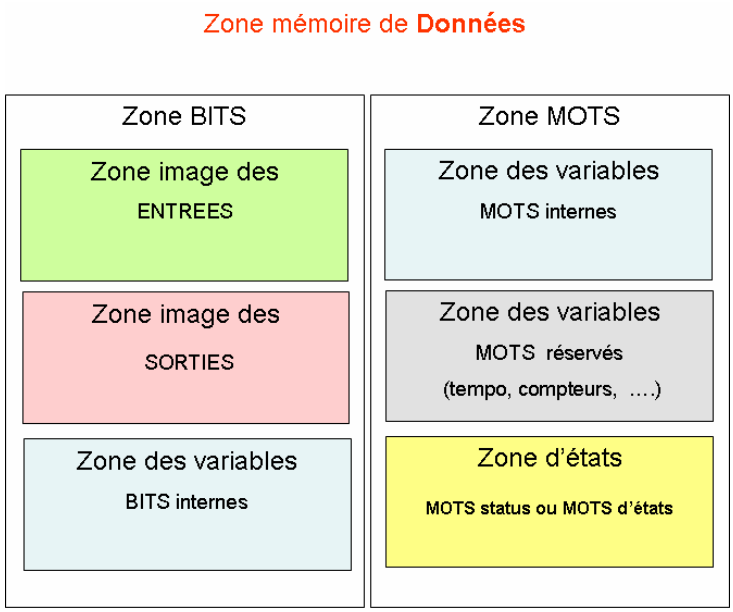
- La **zone MONITEUR** est réservée au système d'exploitation de l'automate,
- la **zone DONNEES** pour le stockage des données
- la **zone PROGRAMME** pour l'écriture du programme applicatif



III.4.2 zone de mémoire de données

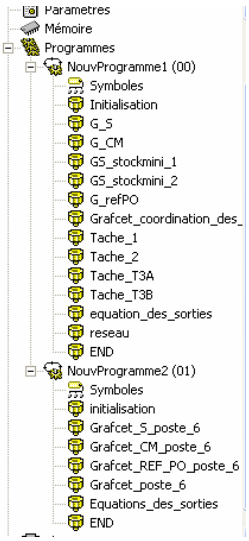
Les données utilisées sont rangées en mémoire dans des mots. L'utilisateur n'a pas à se soucier de l'adresse réelle de chaque donnée, en effet le langage de programmation lui permet de donner une adresse symbolique que le moniteur transformera en adresse physique dans la mémoire. Les données à stocker sont du type bits ou/et mots.

- **zone bit :** Des bits affectés aux entrées et aux sorties sont les images de l'état des coupleurs, le processeur travaillant avec ces images. Des bits de données appelés bit internes ou variables internes, permettent de réaliser les équations logiques, ils peuvent être monostables ou bistables, sauvegardés ou pas.
- **zone mot** : Permet le stockage de variables numériques.



III.4.3 mémoire programme

Reçoit le(s) programme(s) d'application(s). Elle est organisée en mots. Un programme est repéré par l'adresse de sa première instruction ou par un nom (label).
Suivant les constructeurs la mémoire programme peut être découpée en zones ou ne faire qu'une entité.



IV.LE PROCESSEUR

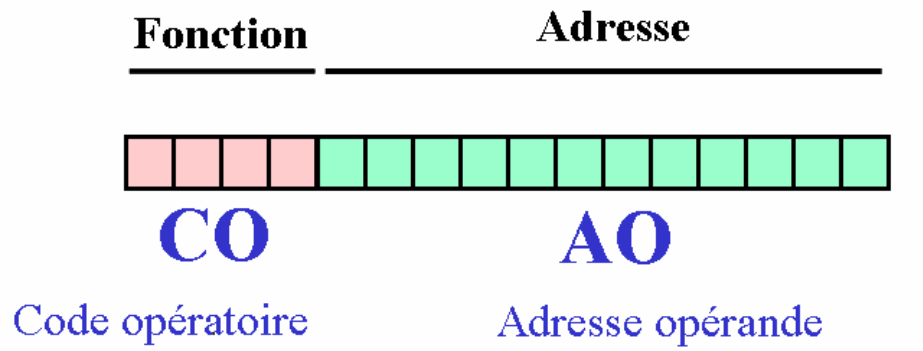
Le processeur d'un API doit répondre à des contraintes:

- de traitement sur bits pour les variables logiques, E/S TOR, séquentiel et combinatoire
- de traitement sur mots pour les variables numériques.
- de travail en temps réel ou avec un temps de cycle court.
- d'interprétation du langage utilisateur pour le codage en instructions du processeur.

IV.1. L'instruction

Une instruction est un ordre exécutable par la logique. Une logique est dite programmable lorsque sa mise en œuvre nécessite l'exécution d'instructions stockées dans une mémoire. Un programme est une suite d'instructions concourant à un traitement donné.
Le processeur ne sait pas tout faire, une instruction est reconnue par lui si elle appartient à l'ensemble des opérations qu'il sait exécuter: ***le jeu d'instructions***. Une instruction précise au processeur le **QUOI FAIRE** et sur **QUOI LE FAIRE,** elle comporte donc deux parties.
Le format de l'instruction est fonction du type de processeur, il peut être fixe (8 à 16 bits) ou variable suivant le type de fonction à réaliser

- le CODE OPERATOIRE (CO) qui indique-le quoi faire, c'est à dire quelle opération il faut effectuer
- l'ADRESSE OPERANDE (AO) qui précise le sur quoi faire, c'est à dire les adresses des données à traiter.



IV.2. Architecture générale

Un processeur est généralement composé de deux éléments principaux:

- Unité de COMMANDE
- Unité de TRAITEMENT

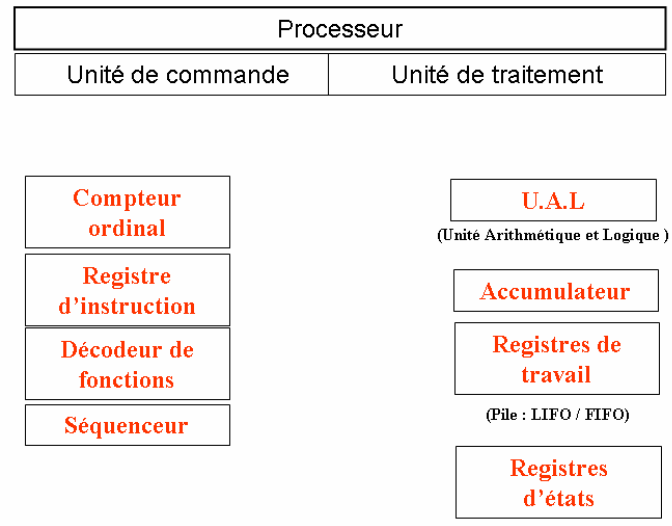
IV.2.1 unité de commande

L'unité de commande est composée de :

- **compteur ordinal** ou **pointeur programme** chargé de l'adresse de l'instruction en cours
- **registre d'instruction** qui stocke l'instruction en cours pendant tout le temps de son exécution. Le registre instruction est en deux parties liées au format de l'instruction. La partie adresse liée à la mémoire et la partie fonction liée au **décodeur de fonctions** chargé de reconnaître l'opération à exécuter.

- **séquenceur** qui génère les signaux "horloges" d'enchaînement des trois phases d'exécution d'une instruction :

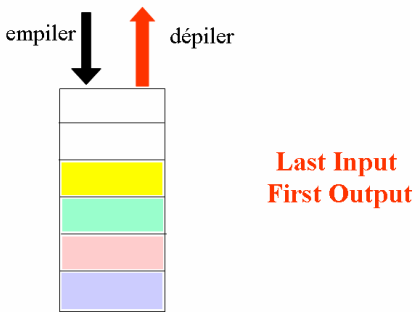
- 1° appel de l'instruction
- 2° exécution de l'instruction
- 3° préparation instruction suivante



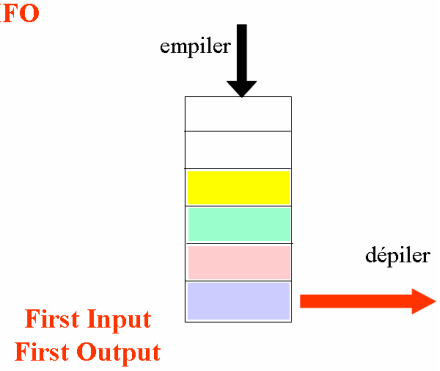
IV.2.2 unité de traitement

L'unité de traitement comprend essentiellement :
UAL unité arithmétique et logique. C'est le circuit le plus complexe chargé d'effectuer toutes les opérations spécifiées par le décodeur de fonction.
Accumulateur est un registre étroitement lié à l'UAL. Il travaille en tant que "source" de données pour l'UAL mais aussi en "destination" pour stocker le résultat d'un travail.
Registres de travail (index, piles[1]) servant au stockage temporaire de données avant traitement.
Registres d'états pour le compte rendu des différentes opérations exécutées par le processeur.

Pile type LIFO

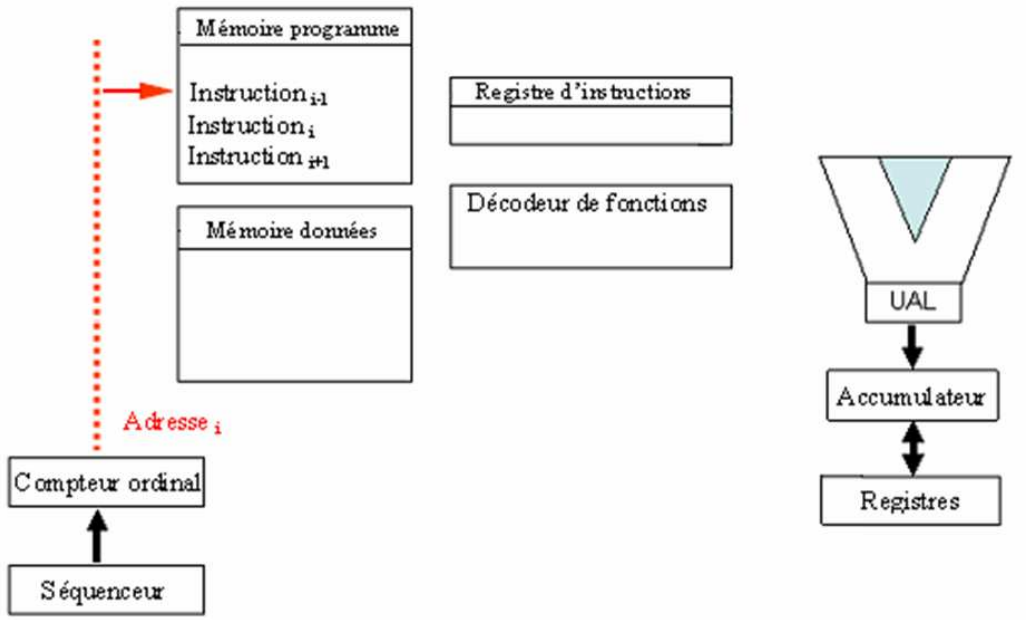


Pile type FIFO



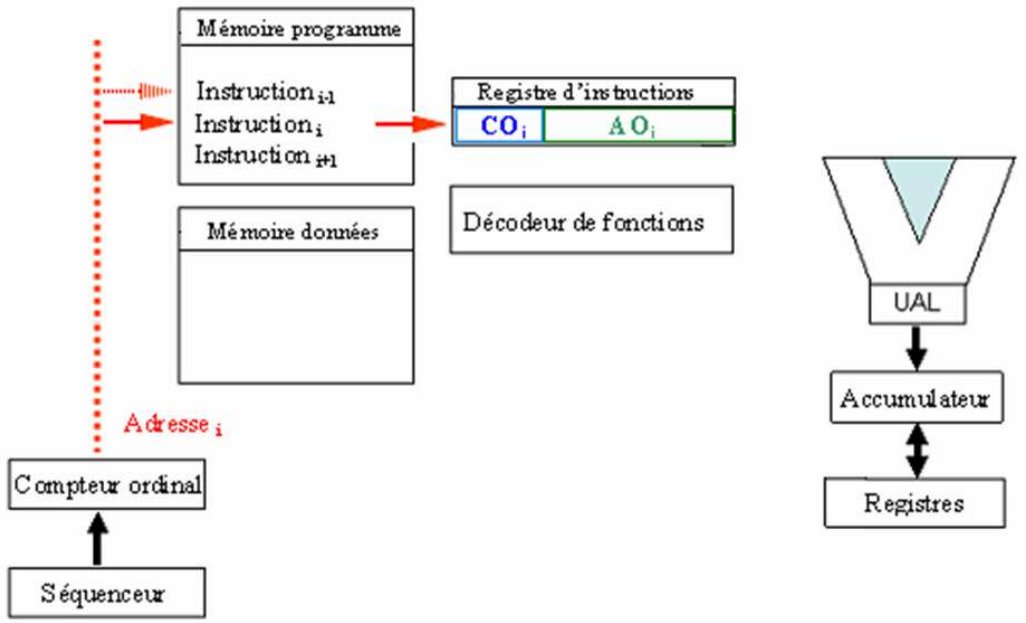
[1] Une pile est constituée d'un ensemble ordonné d'informations et d'une politique de gestion de leur acquisition et de leur restitution. Deux types sont disponibles dans un processeur :
LIFO – dernier entré / premier sorti
FIFO – premier entré / premier sorti

IV.3. fonctionnement simplifié de processeur.



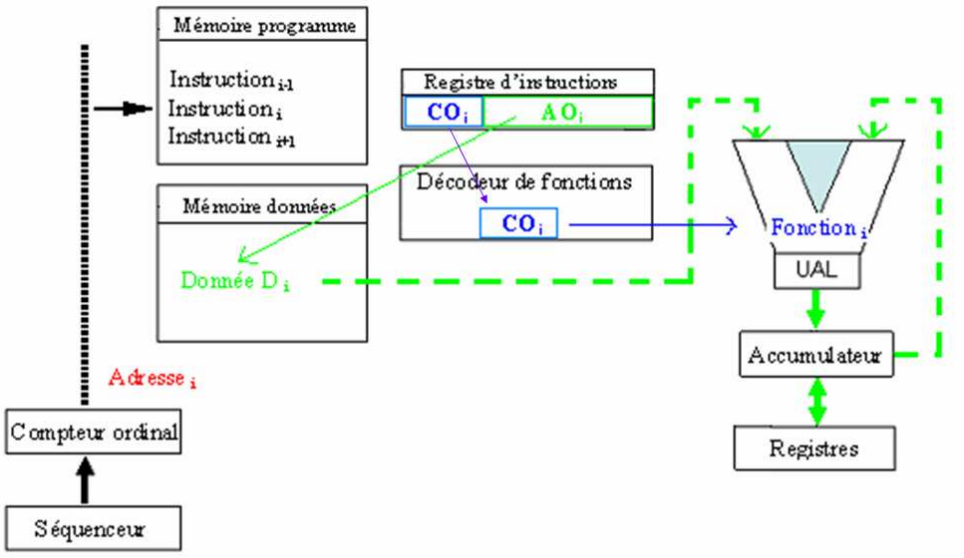
« état initial »

1° PHASE: le compteur ordinal désigne l'adresse de l'instruction en cours dans la mémoire, par l'intermédiaire des Bus l'instruction est ramenée de la mémoire dans le registre d'instruction.



Phase 1 « appel de l'instruction »

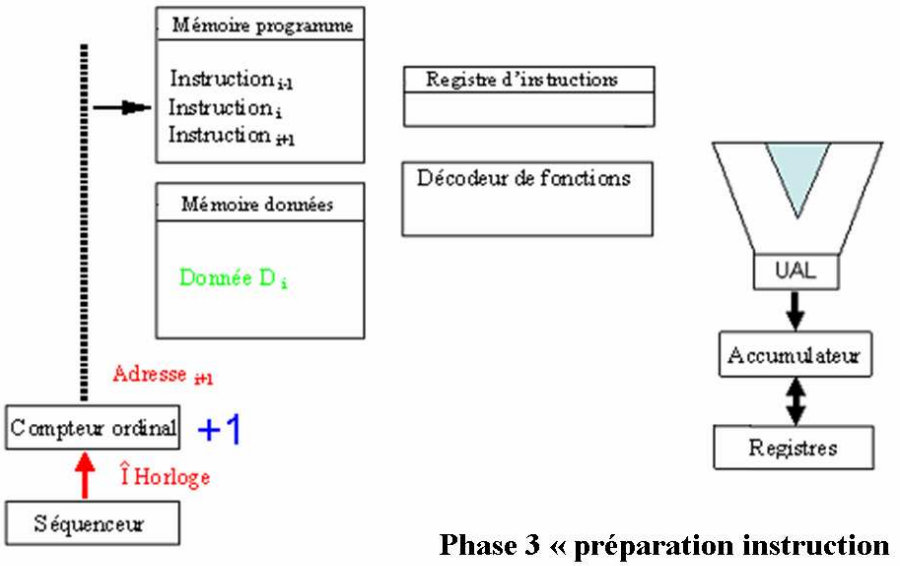
2° PHASE: le décodeur de fonctions interprète et détermine les commandes à appliquer sur l'UAL pour réaliser l'opération désignée sur les données d'adresse AO et contenue dans l'accumulateur. Le résultat est remis dans l'accumulateur.



Phase 2 « exécution de l'instruction »

3° PHASE: le compteur ordinal est incrémenté pour contenir l'adresse de l'instruction suivante.

Un processeur complet, capable de réaliser toutes les opérations logiques et arithmétiques, constitué de registre et d'une mémoire morte contenant les microprogrammes nécessaires à la réalisation de toutes ces opérations, intégré dans un seul boîtier, sur un seul support, est appelé **MICROPROCESSEUR**



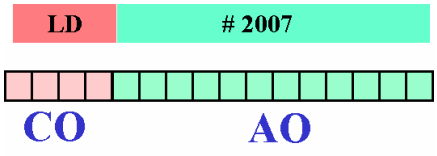
Phase 3 « préparation instruction suivante »

MICROPROGRAMME:Dans un microprocesseur chaque instruction est interprétée, décodée et exécutée sous le contrôle d'un microprogramme formé d'une succession de micro-instructions[2] commandant une phase de l'exécution de l'instruction.

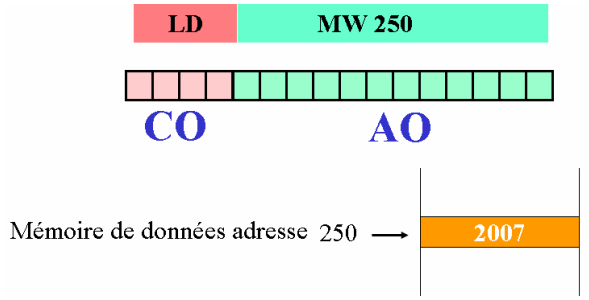
[2] Micro-instruction: instructions localisées dans la mémoire de commande (ROM) du microprocesseur. Macro-instruction: instructions écrites dans la mémoire centrale par l'utilisateur

IV.4. Adressage

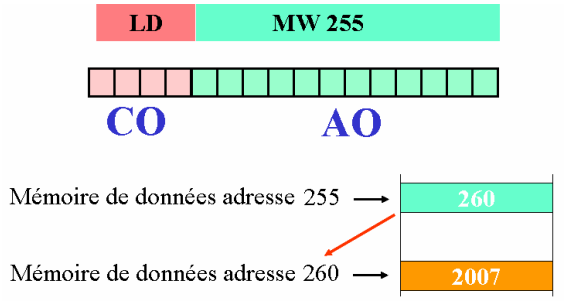
L'adressage doit permettre un maximum de souplesse dans la manipulation des informations, pour cela plusieurs techniques d'adressage sont utilisées.
Adressage immédiat: la donnée est dans la zone adresse opérande.



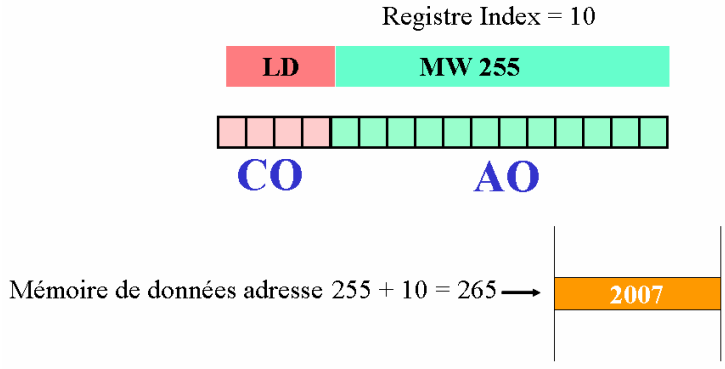
Adressage absolu: la donnée est à l'adresse qui figure dans la zone adresse opérande.



Adressage indirect: l'adresse opérande indique une adresse ou se trouve l'adresse de la donnée



Adressage indexé : l'adresse de la donnée est obtenue en rajoutant la valeur du registre index à la valeur de l'adresse opérande.



V.LE CYCLE DE FONCTIONNEMENT

V.1. Le cycle automate

Lorsque l'utilisateur souhaite faire exécuter un programme, il doit fournir à la logique programmable (moniteur de la machine) des informations sous forme de directives. Dans le cas général, il suffit de charger le compteur ordinal de l'adresse de début de programme et de demander l'exécution. La dernière instruction du programme est un ordre de "fin d'exécution" qui rend la main à l'utilisateur ou qui arrête la machine. Dans ce cas le programme est exécuté à la demande, on dit que la machine à un fonctionnement asynchrone, par rapport à une horloge interne rythmant le fonctionnement propre de la machine.

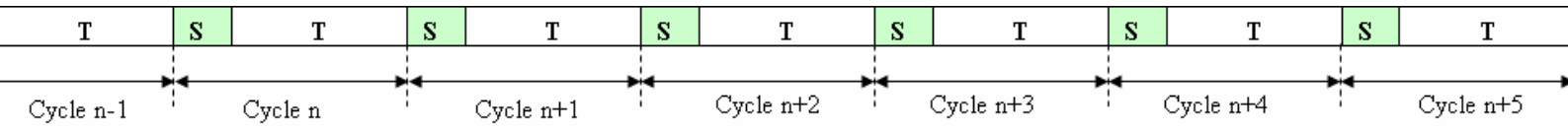
Une caractéristique unique des automates programmables est au contraire le fonctionnement cyclique, c'est à dire synchrone, de l'unité centrale.

Le pointeur programme, en décrivant successivement tous les mots de la mémoire programme jusqu'au dernier, avant de recommencer à partir du premier, assure le fonctionnement cyclique de l'unité centrale et synchrone par rapport à l'horloge interne. La puissance d'une UC est donc directement liée à sa vitesse de scrutation.

On appelle période d'un API, ou vitesse de scrutation, le temps mis pour l'exécution de 1K instructions logiques.

Lors d'un tour de cycle, l'UC assure les fonctions système (gestion interne, autodiagnostics, échanges) et le traitement des données spécifié par le programme. Un cycle réel de la machine comprend donc deux phases :

- ***la phase système***
- ***la phase traitement***



Exemple de fonctionnement

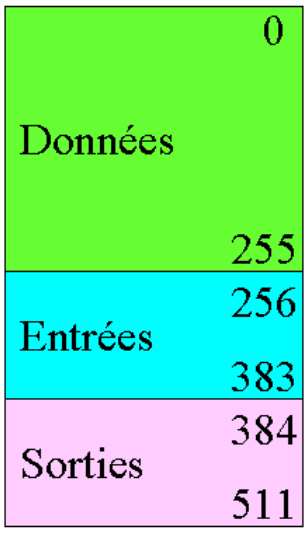
Equation

S1 = (E1+E2) ($\overline{E3}$.X1 + $\overline{E8}$.X2)

Jeu d'instructions

instruction	mnémonique	code	Fonction réalisée
CHARGER	SI SI/	0001 0010	Test d'une variable Test du complément d'une variable
FONCTIONS LOGIQUES	ET ET/ OU OU/	0011 0100 0101 0110	Faire une ET Faire une ET complémenté Faire une OU Faire une OU complémenté
AFFECTATION	OUT OUT/	0111 1000	Affecter une valeur à la variable Affecter le complément

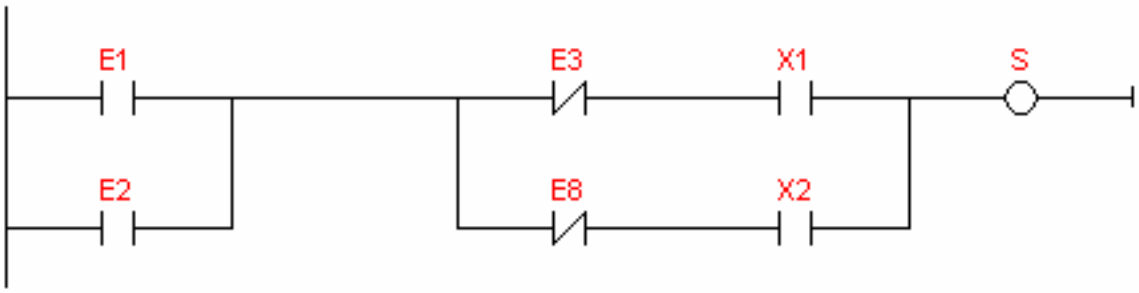
Mémoire de données



Adressage

E1	256
E2	257
E3	258
E8	259
S	384
X1	000
X2	001

Programme

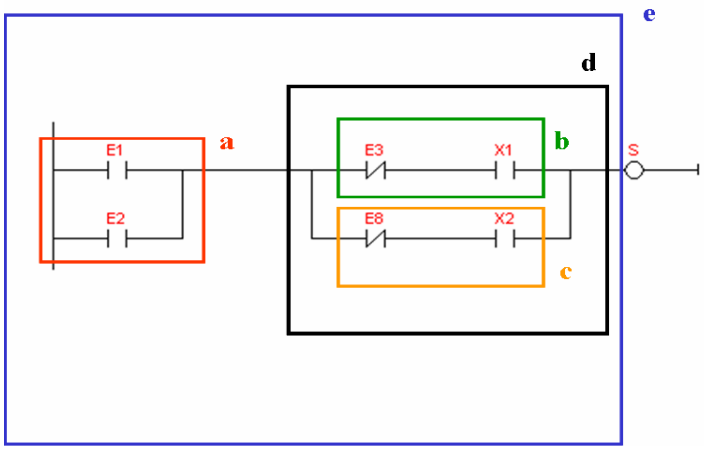


Fonctionnement littéral

0, Charger E1 } a
1, Faire OU avec E2
2, Charger complément E3 } b
3, Faire ET avec X1
4, Charger complément E8 } c
5, Faire ET avec X2
6,Faire OU (c + b) } d
7, Faire ET (d. a) } e
8, Affecter résultat à S

Fonctionnement Automate

Instruction N°	0	1	2	3	4	5	6	7	8	
Contenu de l'accumulateur	E1	a	$\overline{E3}$	b	$\overline{E8}$	c	d	e		
Contenu de la pile			a	a	b	b	a			
					a	a				



0, SI	0256	} a
1, OU	0257	
2, SI/	0258	} b
3, ET	0000	
4, SI/	0259	} c
5, ET	0001	
6, OU		} d
7, ET		} e
8, OUT	0384	

V.2. Gestion des entrées-sorties

Les informations entrent et sortent de l'API au travers de coupleurs adaptés à leur nature.
Le système d'entrées-sorties, constitué du gestionnaire d'entrées/sorties et de l'ensemble des coupleurs, doit permettre, sur le plan fonctionnel, la résolution de deux problèmes :

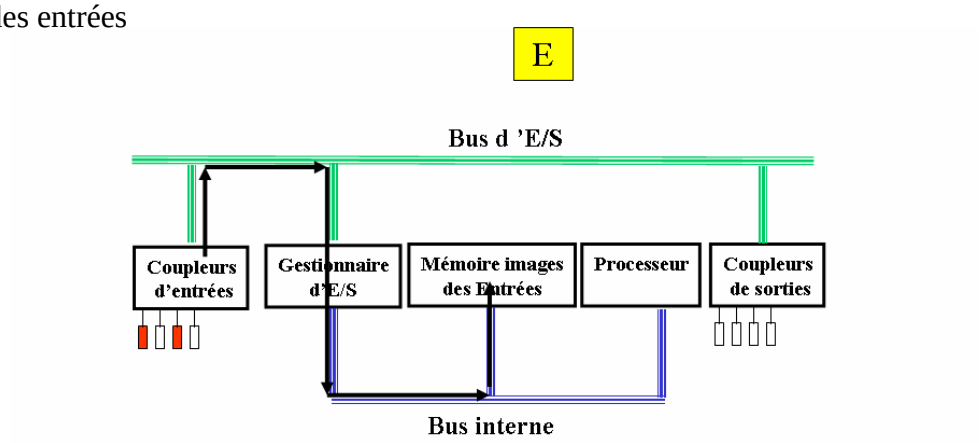
- La liaison technologique entre l'extérieur et le bus d'entrée/sortie, c'est le rôle des coupleurs.
- La cohérence entre les informations, sachant que le processus a ses temps de réactions propres et le processeur le sien, de par le déroulement séquentiel du programme.

C'est pourquoi les informations d'entrées, lues sur les coupleurs d'entrées, sont mises en mémoire pour devenir des variables d'entrées. Le processeur ne travaillant que sur ces images aura une vision cohérente du processus durant tout le déroulement du programme.
De même, les valeurs des états des sorties, calculées par le processeur tout au long du programme, sont mises en mémoire pour devenir des variables de sorties qui seront ensuite transmises, groupées, aux coupleurs de sorties. Les pré-actionneurs recevront donc une image cohérente des actions à accomplir, en assurant la continuité des états des sorties entre deux mises à jour.
Ces opérations de lecture/écriture des variables d'entrées/sorties sont effectuées automatiquement par le processeur.
Les coupleurs d'entrées/sorties assurent la liaison entre l'unité centrale de l'API et l'extérieur. Les traitements effectués dans les coupleurs d'entrées/sorties se situent au niveau de la nature des signaux et de celui de l'information pour les rendre compatibles, d'un coté avec les caractéristiques internes de l'API, et de l'autre avec les caractéristiques externes du processus. Ces traitements d'informations, au niveau des coupleurs, correspondent à une décentralisation du traitement qui modifie de plus en plus le rôle des coupleurs.
De la simple fonction d'interface technologique, le coupleur évolue vers un partage du travail avec le processeur. Ce partage permet de soulager le processeur de tâches standard figées ou paramétrables, pour se concentrer aux tâches spécifiques du process. Ce traitement au niveau du coupleur permet également d'obtenir des performances de vitesses qui n'auraient pût être atteintes avec un processeur unique. Il est également possible d'obtenir de ces coupleurs "intelligents" des informations sur leur fonctionnement.
Les coupleurs se caractérisent donc par la nature des traitements effectués sur les signaux et sur les informations, ainsi que sur le sens des échanges.

V.3. Cycle synchrone vis à vis des entrées-sorties

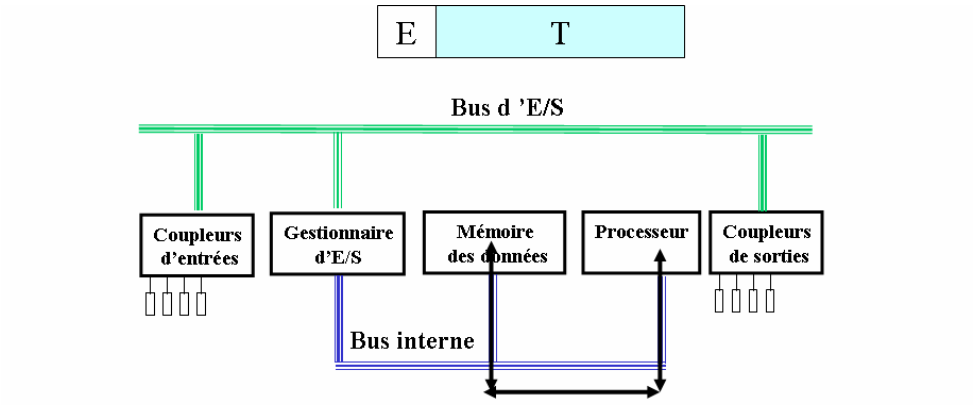
C'est la procédure la plus simple et la plus utilisée. Le cycle automate se décompose de la manière ci-dessus.
Un tel **cycle de scrutation** va permettre une programmation élémentaire. Bien que limité en puissance, cette méthode permet d'éviter un bon nombre d'aléas, et l'écriture programme sous forme synchrone assurera une implantation conforme du GRAFCET.
Il est à noter que l'affectation des variables internes s'effectue au fur et à mesure de leur traitement dans le programme.
Les divers automates du marché n'ont pas tous le même type de cycle. La différence essentielle vient de la politique d'acquisition des entrées/sorties du système. Intervient aussi la répartition de l'intelligence dans la structure multiprocesseur.

Acquisition des entrées



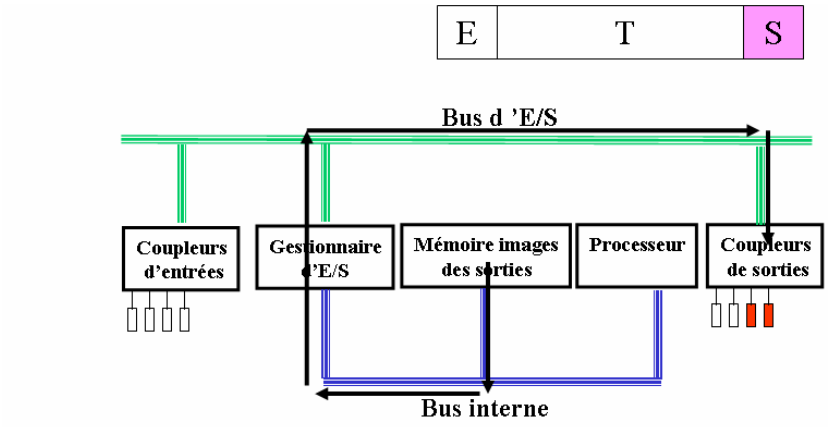
Lire toutes les entrées de tous les coupleurs d'entrées et, via le gestionnaire d'E/S, les recopier en mémoire de données dans la zone image des entrées.

Traitement



Traiter les équations du programme d'application à partir des entrées mémorisées et :
- **pour les sorties, écrire les valeurs en zone image des sorties,**
- **pour les variables internes, écrire les valeurs en zone correspondante.**

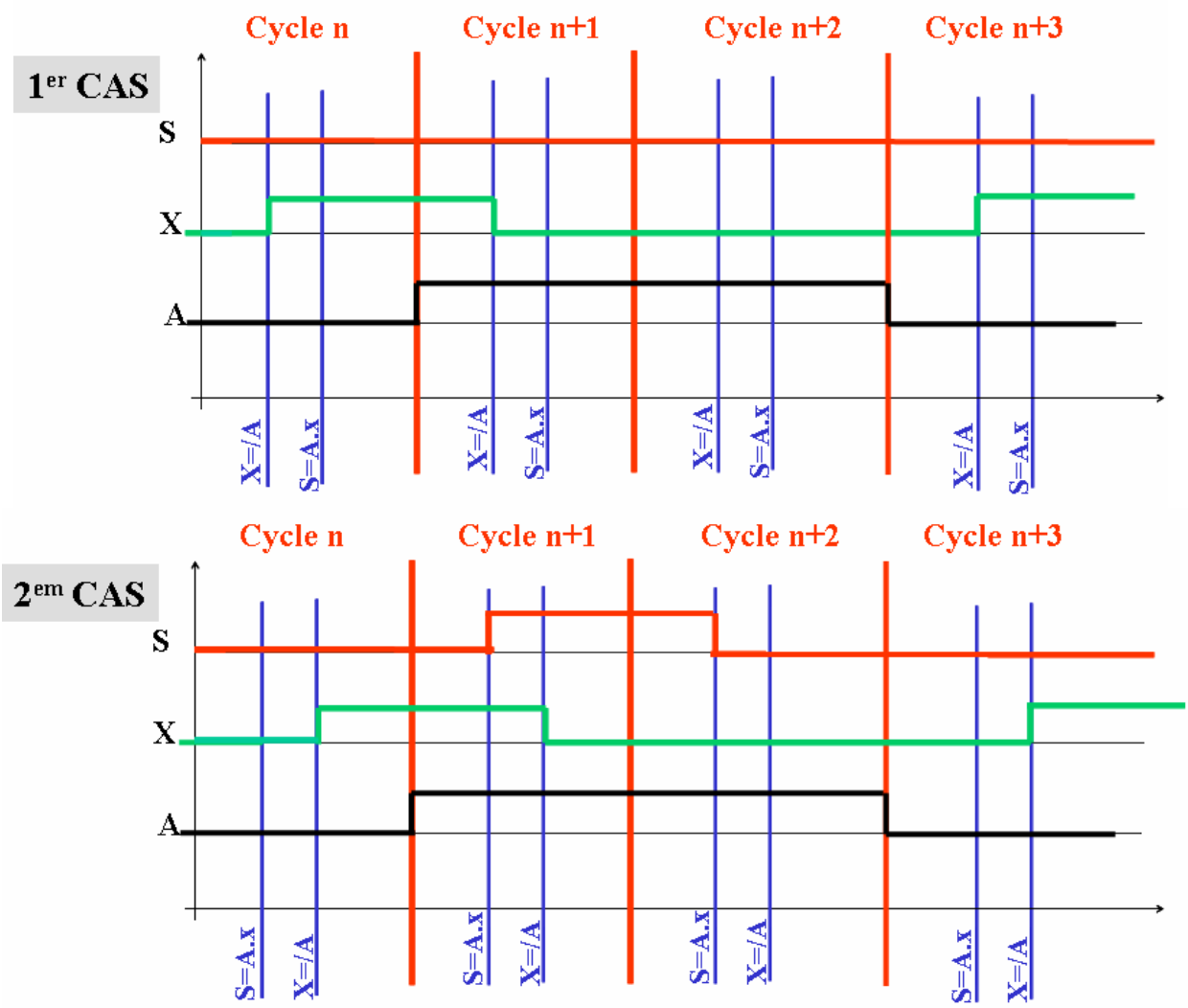
Affectation des sorties



Lire les valeurs des sorties en zone image des sorties et, via le gestionnaire d'E/S, recopier les états de toutes les sorties sur tous les coupleurs de sorties.

V.4. Influences du cycle sur la programmation

Les procédures de fonctionnement des Unités Centrales, malgré les avantages quelles procurent, engendrent aussi des difficultés de programmations.
Soit les équations de fonctionnement suivantes, deux programmations sont possibles pour ces équations.
S=A.X et X=/A



Dans le deuxième cas, la sortie passe à (1) pendant un tour de cycle**.** Ceci provient du fait que les opérations logiques exécutées les unes après les autres à partir de résultats antérieurs s'apparentent davantage à des opérations séquentielles qu'à des opérations combinatoires. Ce phénomène est aussi du au mode séquentiel de scrutation.
NOTA:
Si une sortie est commandée à plusieurs endroits du programme (plusieurs équations), l'état mis à jour sur le coupleur de sortie est celui pris en dernier dans l'exécution du programme (dernière équation).

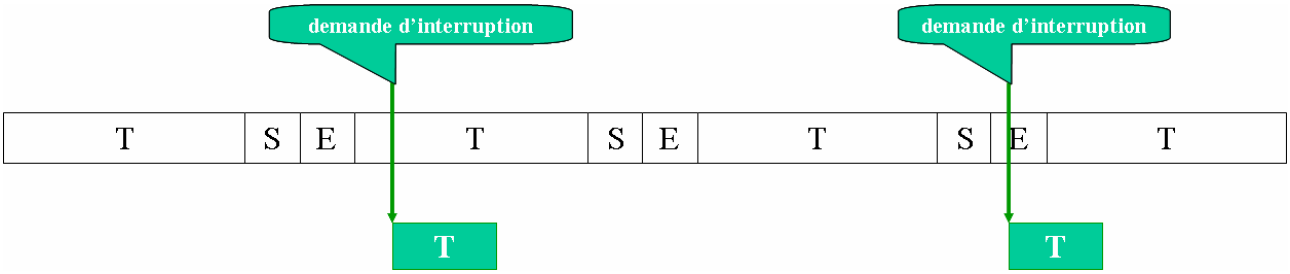
Pour qu'il n'y ait pas risque d'erreur avec les variables de type monostables, le programme ne doit comporter qu'une seule équation pour chaque variable, pour n'être lue et exécutée qu'une fois à chaque cycle de scrutation.

V.5. Cycles d'interruption

Les automates de dernière génération permettent le travail par interruption du cycle. A la configuration de l'API il est possible de déclarer une ou plusieurs entrées comme « entrée interruptive » ou/et une fréquence fixe d'interruption du cycle. On possède alors deux types d'interruption.

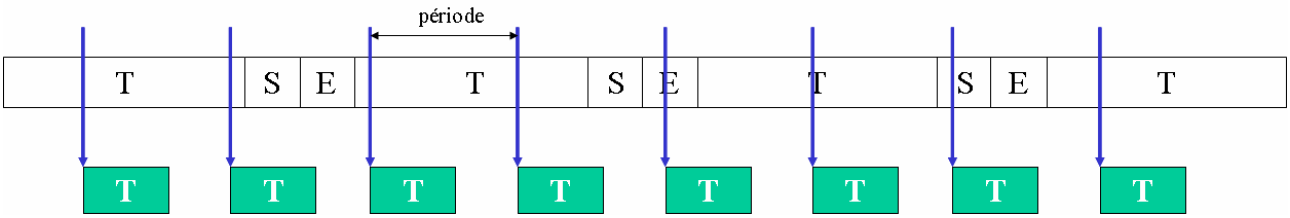
V.5.1 Interruption commandée.

Le changement d'état d'une entrée déclarée comme interruptive provoque l'interruption du cycle en cours pour traiter la partie de programme liée à l'interruption.
On appelle cela travailler en « tâche d'interruption ».



V.5.2 Interruption périodique

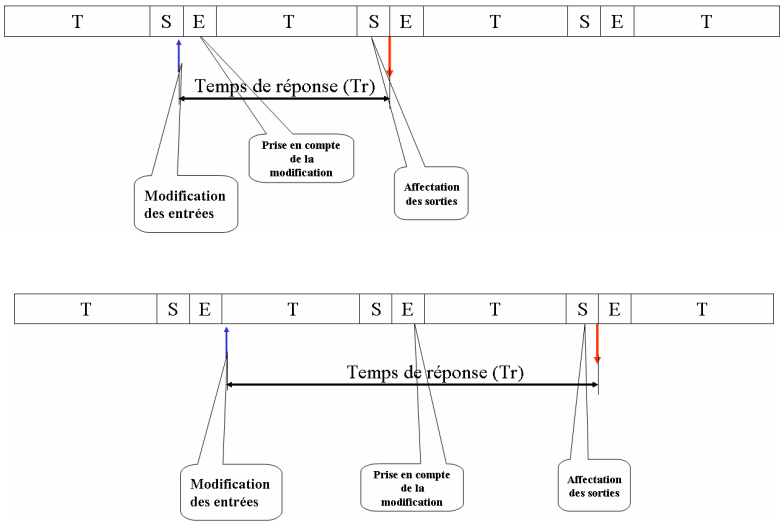
Dans ce cas, l'interruption est fixée par le système à un intervalle de temps défini lors de la configuration.
On appelle cela travailler en « tâche rapide ».



Dans tous les cas, le nombre d'interruption est limité par le système et la gestion est liée a un certain nombre de critères dont les priorités entre-elles n'est pas le moins important.

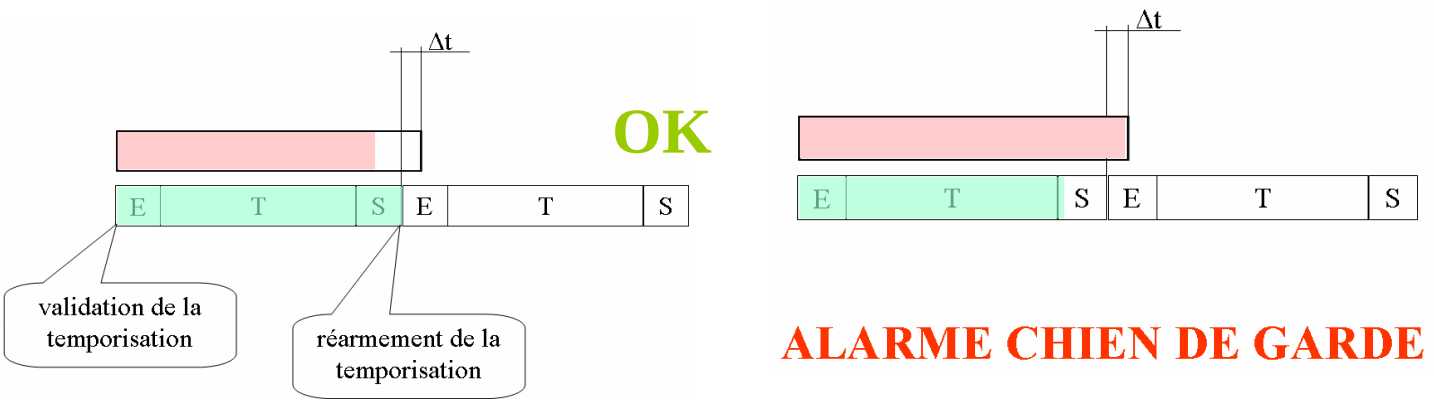
VI. LA SECURITE DE FONCTIONNEMENT

Le temps de cycle est défini comme le temps mis par l'UC pour exécuter les trois phases de la scrutation ; acquisition des entrées, traitement, affectation des sorties.
Temps de réponse d'un API est le temps nécessaire à l'automate pour prendre en compte une modification des entrées, traiter cette modification et affecter les sorties correspondantes.



1 cycle =< Tr =< 2 cycles

Pour éviter des défauts de fonctionnement graves au niveau de la partie opérative en cas de défaillance de l'unité Centrale ou pour vérifier le bon déroulement du programme, l'Unité Centrale possède un système de surveillance appelé ***"CHIEN DE GARDE"*** ou ***WATCHDOG.*** Le principe de ce dispositif est le suivant:
Le temps total d'une scrutation ne peut pas excéder une valeur maximale qui dépend de la vitesse de l'UC et de la capacité maxi de la mémoire. Une temporisation, par décomptage du temps, rafraîchie automatiquement à chaque tour de cycle (passage par un point obligé), déclenche ***l'arrêt de la scrutation*** *et* ***la remise à zéro des sorties*** si la durée du cycle devient supérieure à la valeur de seuil de la temporisation.



Exemple:

Prenons un API dont la mémoire de données permet la configuration maximum de :

- 128 entrées- 8x16 cartes E
- 144 sorties - 9x16 cartes S
- Mémoire programme 4K
- Le temps d'accès à une carte d'entrée/sortie est de 29µs
- La période de 1,85 ms.

Acquisition des entrées $8 \times 29 = 232\ \mu s$

Affectation des sorties $9 \times 29 = 261\ \mu s$

Durée du traitement $1{,}85 \times 4 = 7{,}4$ ms

Temps de cycle $0{,}232 \text{ ms} + 7{,}4 \text{ ms} + 0{,}261 \text{ ms} = 7{,}893$ ms

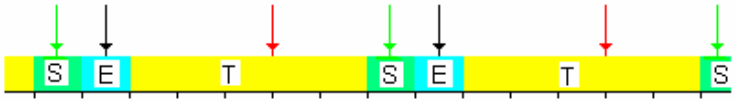
Durée du chien de garde > 7,893 ms (9 ms par exemple)

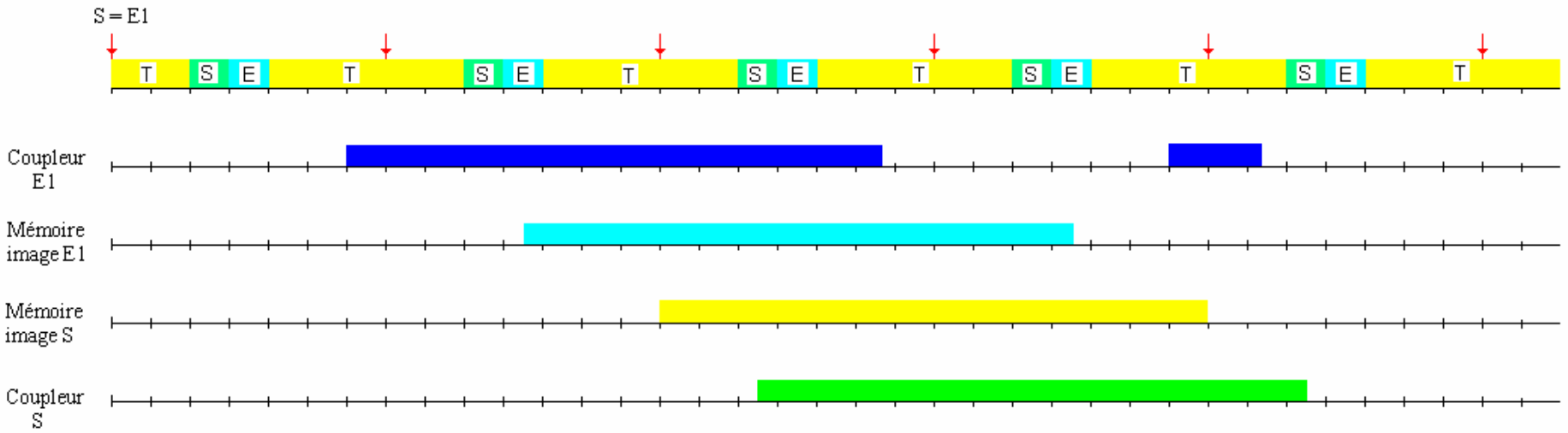
$7{,}893 \text{ ms} \leq Tr \leq 15{,}786 \text{ ms}$

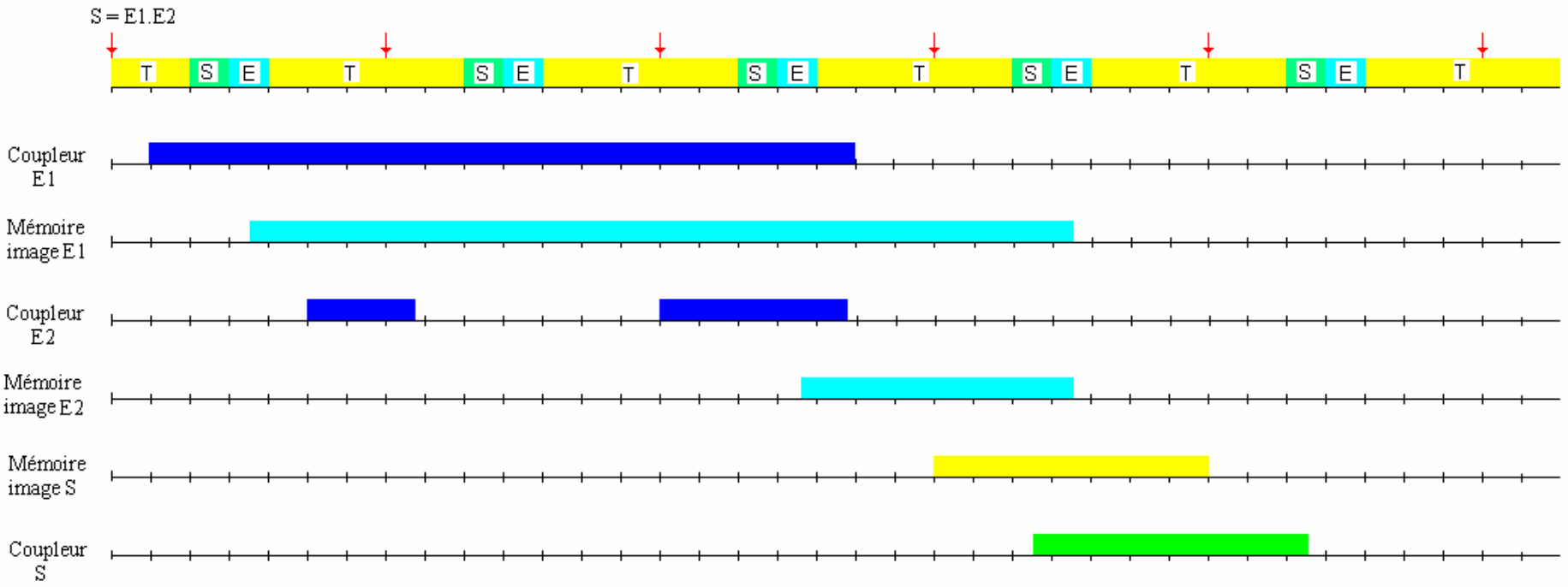
On considère que ↓ représente l'instant de l'acquisition de l'entrée E1.

On considère que ↓ représente l'instant de l'affectation de la sortie S.

On considère que ↓ représente l'instant du traitement de l'équation S = E1






S = E1.E2
T
S
E
Coupleur E1
Mémoire image E1
Coupleur E2
Mémoire image E2
Mémoire image S
Coupleur S

Les coupleurs industriels

I. GENERALITES

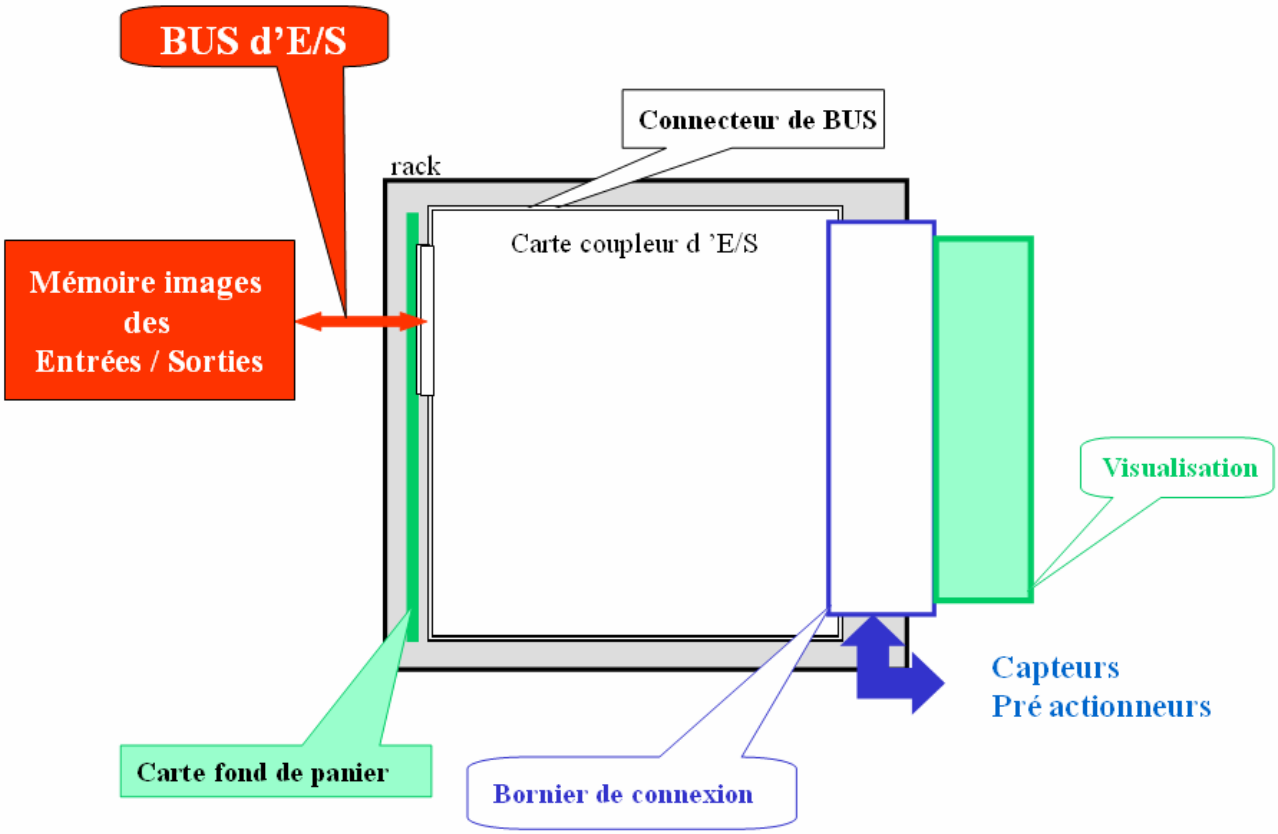
Les coupleurs industriels assurent la liaison entre l'Unité Centrale de l'automate et l'extérieur, c'est à dire le process.

- les coupleurs d'entrées reçoivent des signaux de l'extérieur (capteurs), les traitent pour les rendre compatibles avec la logique interne de l'API.
- les coupleurs de sorties reçoivent les informations de l'UC, les amplifient pour les rendre compatibles avec les récepteurs extérieurs (pré actionneurs).

Les entrées/sorties d'automates ont fait l'objet de développements importants dés l'origine du fait de la finalité de la machine. De la simple fonction d'interface technologique, le coupleur a évolué vers un partage du travail avec le processeur central de l'UC. Ce partage permet de soulager le processeur de tâches standard, pour se consacrer aux tâches spécifiques du process. Ce traitement au niveau des coupleurs "intelligents" permet d'obtenir des performances de vitesse, mais aussi des informations sur le fonctionnement interne du coupleur. Chaque constructeur offre dans son catalogue une panoplie plus ou moins étendue de coupleurs différents. Les types les plus représentatifs dans l'utilisation des automates programmables peuvent être rassemblés en trois catégories:

- Les coupleurs ou interfaces d'entrées/sorties tout ou rien (TOR)
- Les coupleurs intelligents
- Les coupleurs de communication (sous-ensemble des coupleurs intelligents)

Composition d'un coupleur standard[3]



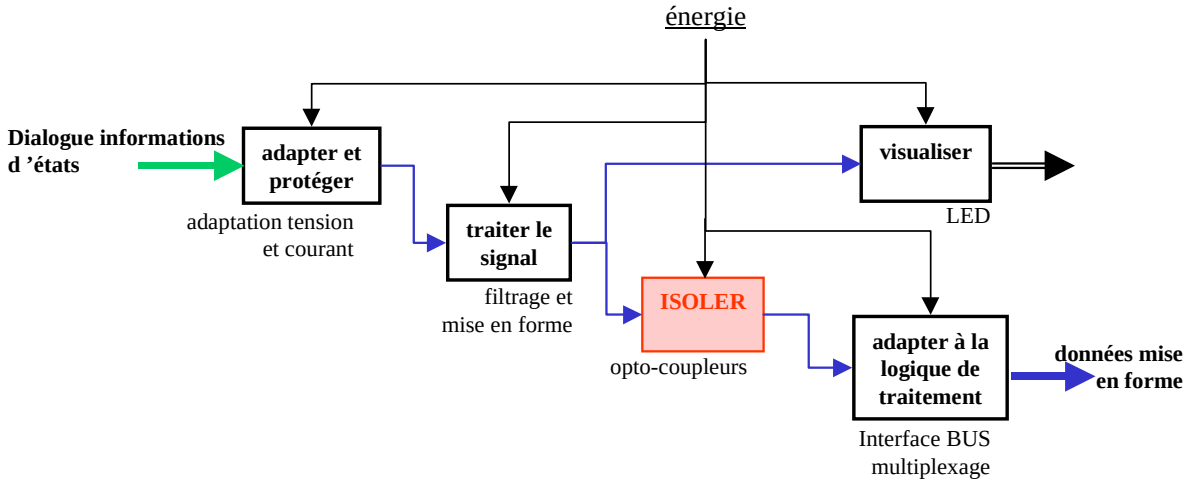
[3] L'étude porte sur le coupleur au sens fonctionnel. Le point de vue technologique sera évoqué plus tard.

II. Les coupleurs d'entrées

Le module d'entrées permet à l'UC de l'API d'effectuer une lecture de l'état logique des capteurs qui lui sont associés. A chaque entrée correspond une voie qui traite le signal pour élaborer un BIT. L'ensemble des bits forme un MOT recopié périodiquement dans la zone mémoire image des entrées de la mémoire de données.
Les caractéristiques principales d'un coupleur d'entrées sont :

- la nature et la tension d'entrée
- les seuils de détection des niveaux logiques
- le temps de filtrage
- la modularité
- l'indépendance ou non des entrées (point commun)

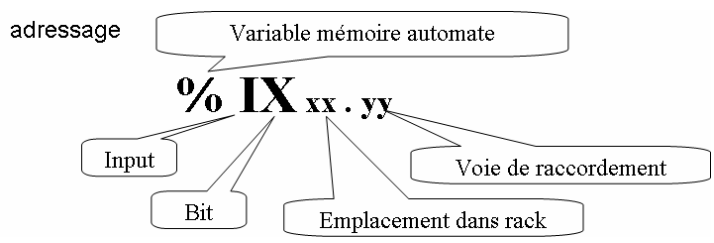
Structure fonctionnelle



II.1. L'adressage

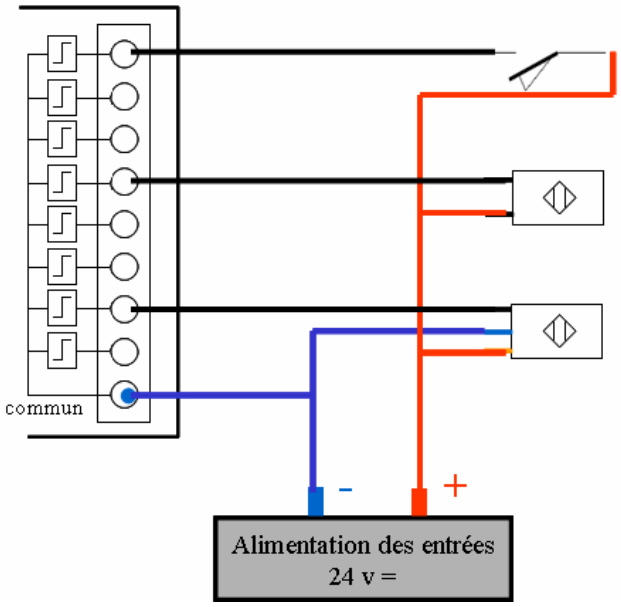
La norme de programmation CEI 1131 préconise une syntaxe d'adressage du type

% symbole générique de variables mémoire automate
I input
X travail sur bit
xx n° d'emplacement de la carte dans le rack (ou n° du mot)
yy n° de la voie de raccordement

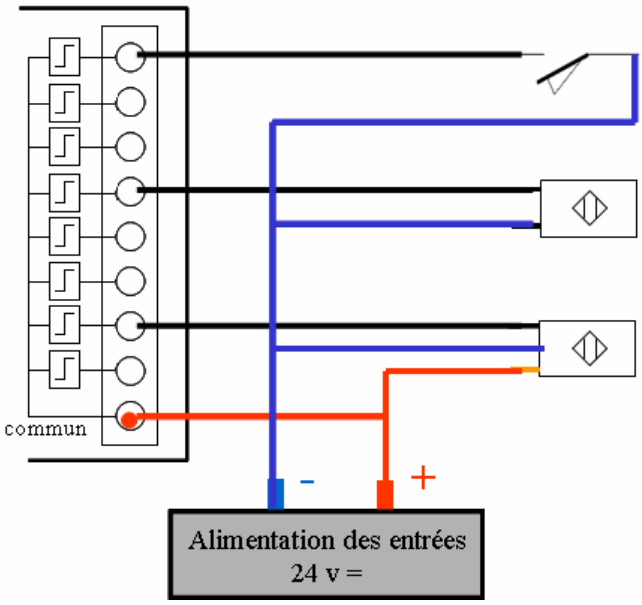


II.2. câblage et raccordement

Les coupleurs d'entrées TOR acceptent toutes les technologies de capteurs en fonction de leur technique interne[4] (Capteurs 3 fils, 2 fils ou contact).
On trouve des coupleurs (ou cartes) au modulo 8, 16, 32 entrées, avec des possibilité de raccordements d'entrées indépendantes ou avec un commun pour 8 ou 16.
Les entrées sont généralement alimentées par l'automate (alimentation interne 24V=, 500mA)



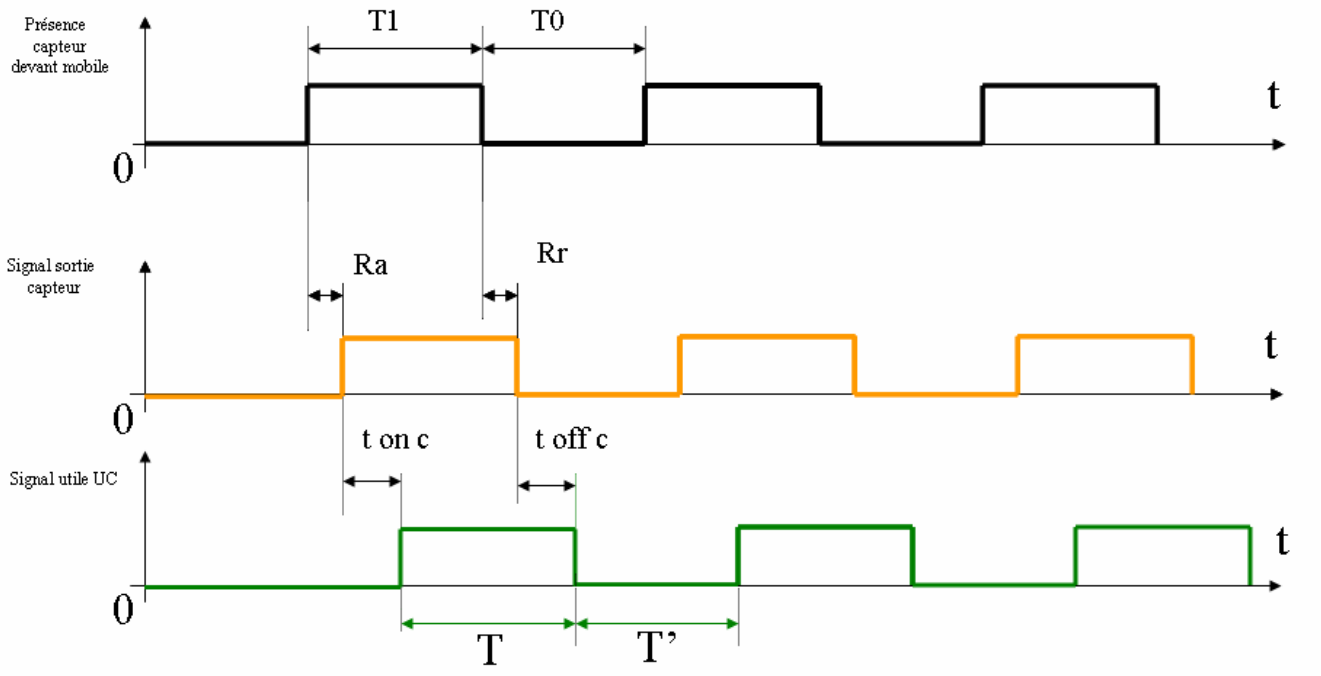
Type PNP



Type NPN

[4] Voir cours de technologie.

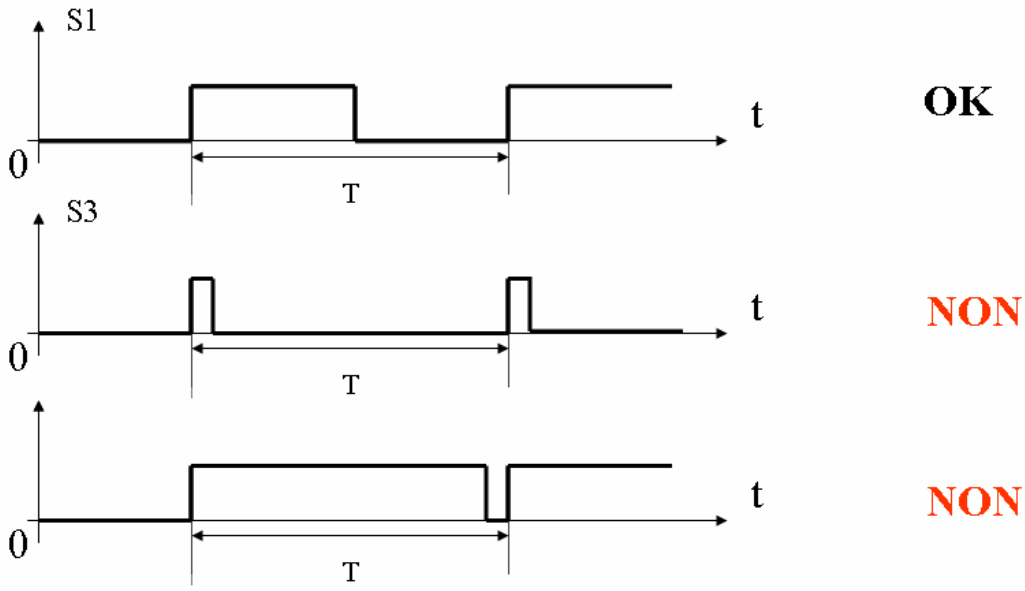
Conditions à remplir pour que l'information d'un capteur soit prise en compte correctement par l'automate programmable.



T = T1 + Rr + t off c – Ra – t on c > t cycle API

T' = T0 + Ra + t on c – Rr – t off c > t cycle API

Malgré une fréquence d'évolution très faible du signal sur une entrée, l'automate peut avoir des problèmes pour suivre !

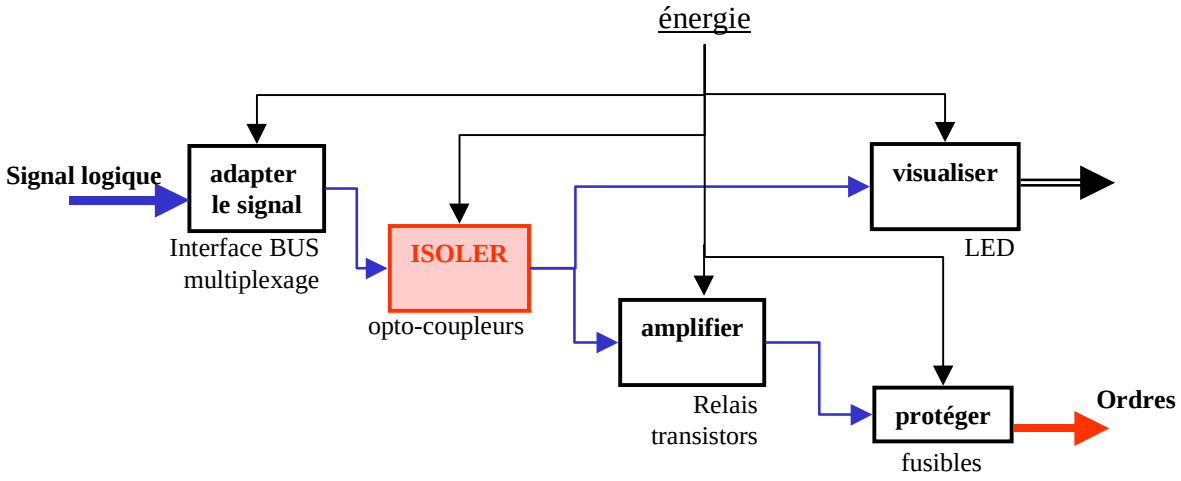


III. LES COUPLEURS DE SORTIES

Les modules de sorties permettent d'agir sur les pré actionneurs. Les informations fournies par le processeur, stockées en mémoire image des sorties, sont envoyées au coupleur via le gestionnaire d'entrées/sorties.
Le signal est ensuite traité par le coupleur pour réaliser la correspondance état logique et signal électrique.
Les caractéristiques principales d'une interface de sorties sont:

- la technologie des sorties
- la nature, l'amplitude de la tension commandée et son intensité (ou puissance)
- la modularité
- l'indépendance ou non des sorties.

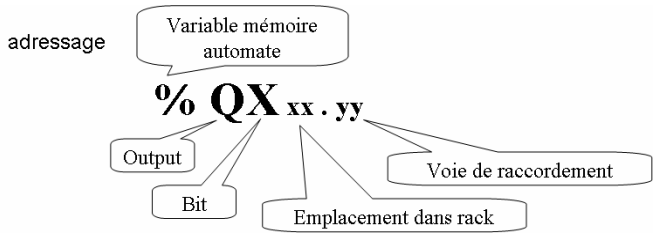
Structure fonctionnelle



III.1. L'adressage

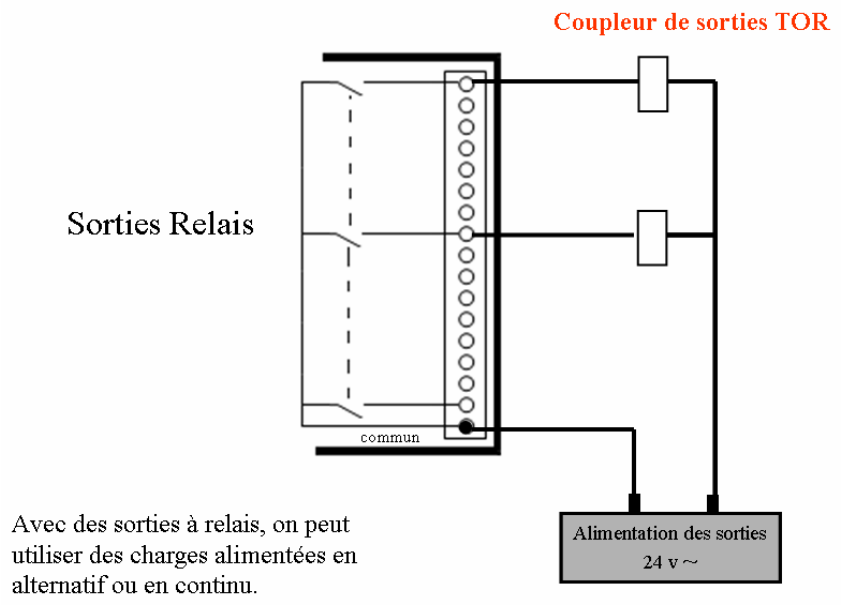
La norme de programmation CEI 1131 préconise une syntaxe d'adressage du type

% symbole générique de variables mémoire automate
Q output
X travail sur bit
xx n° d'emplacement de la carte dans le rack (ou n° du mot)
yy n° de la voie de raccordement

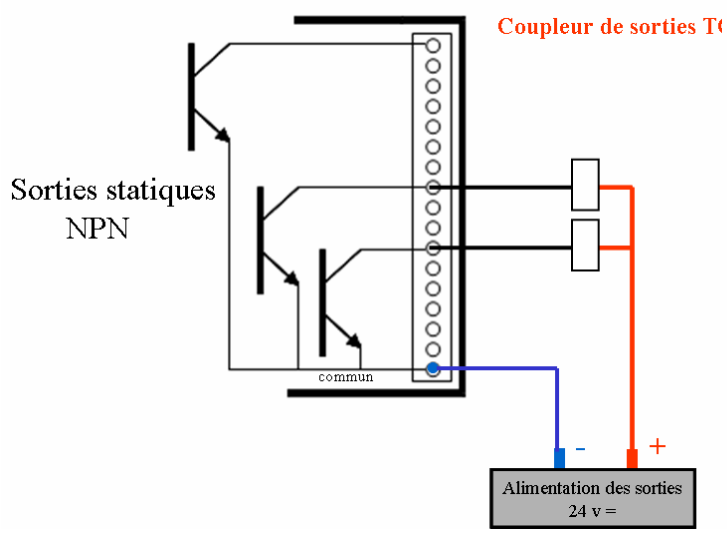


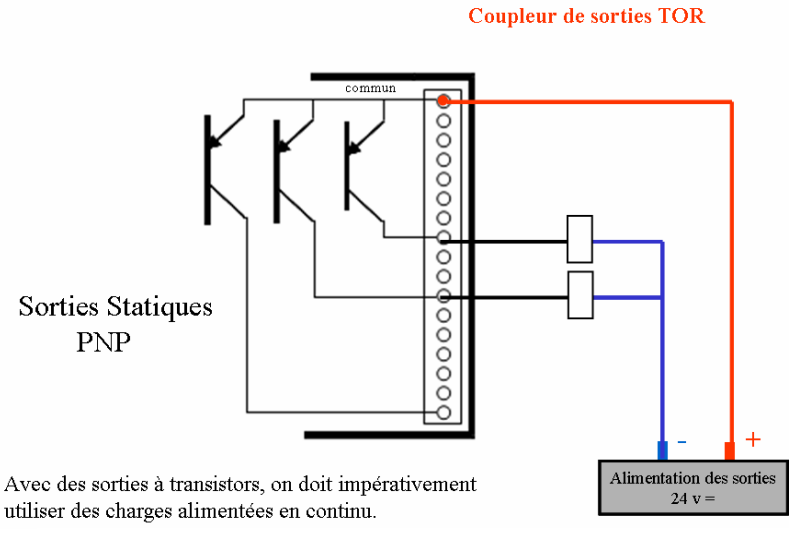
III.2. câblage et raccordement

Les coupleurs de sorties TOR acceptent toutes les technologies de pré-actionneurs en fonction de leur technique interne[5].
On trouve des coupleurs (ou cartes) au modulo 8, 16, 32 sorties, avec des possibilités de raccordements indépendants ou avec un commun pour 8 ou 16.
Les sorties sont généralement alimentées par une alimentation externe.



Avec des sorties à relais, on peut utiliser des charges alimentées en alternatif ou en continu.





Avec des sorties à transistors, on doit impérativement utiliser des charges alimentées en continu.

[5] Voir cours de technologie.

IV. TYPES D'ARCHITECTURES

L'évolution des automates programmables a entraîné une modification des systèmes automatisés, qui n'ont cessé d'évoluer des architectures centralisées vers des architectures distribuées. Deux causes principales à ce mouvement :

- L'évolution des besoins.

La concurrence internationale pousse l'entreprise à accroître en permanence sa compétitivité en automatisant ses installations. Les gains de productivité ne sont toutefois sensibles que si cette automatisation reste flexible. Une contrainte qui est peu compatible avec une architecture centralisée faisant appel à un seul automate.

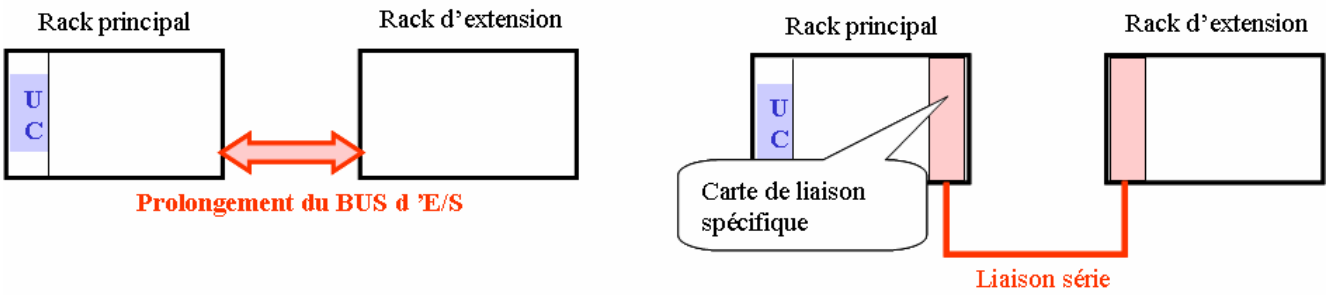
- Les progrès technologiques.

La miniaturisation et la réduction des coûts conduisent à intégrer plus de fonctions dans des équipements plus petits, plus performants et moins chers.
Ces technologies ont permis la mise en œuvre de deux architectures distribuées par :

- décentralisation du traitement
- décentralisation des entrées/sorties

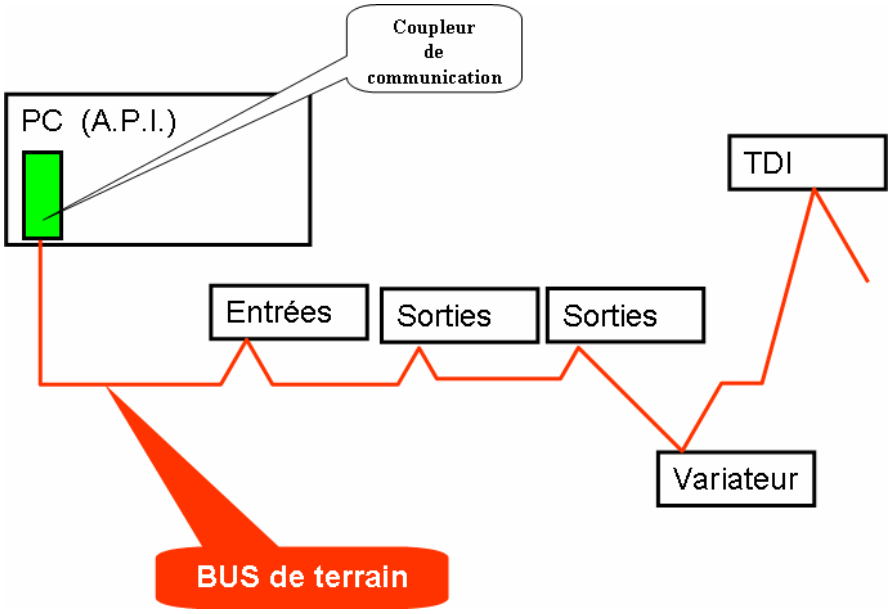
IV.1. architecture centralisée

Utilisée dans des petites applications avec un nombre d'entrées / sorties limitées ne demandant qu'un développement logiciel simple. Une concentration géographique est aussi nécessaire pour éviter un câblage important. La mise en œuvre de systèmes automatisés plus complexes demande une étude logicielle très structurée pour une mise au point et une maintenance facilitée. L'utilisation de configuration d'API compact ou modulaire est possible dans ce cas. La possibilité de rack d'extension permet une augmentation en nombre d'E/S mais «charge » le câblage et le logiciel.



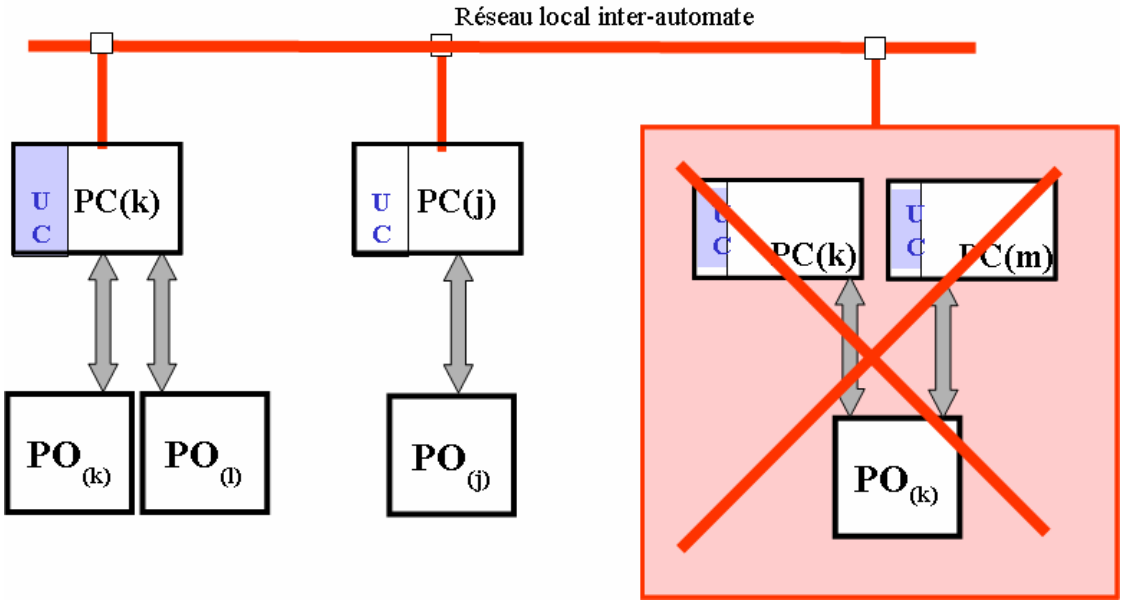
IV.2.Architecture décentralisée

Dans ce type d'architecture, l'API pilote des éléments concentrateurs d'entrées/sorties. Les coupleurs ne se trouvant plus dans l'armoire, il faut établir un lien avec l'UC : c'est le rôle du BUS de TERRAIN et des BUS Capteurs Actionneurs.
Ce type d'architecture peut être utilisé lorsque le procédé n'est pas facilement « découpable ». La décentralisation des entrées/sorties favorise la réalisation de machines modulaires. Elle facilite la décentralisation des postes de conduite et de diagnostic au cœur de l'installation. Le câblage est très simplifié, il faut faire attention, dans ces conditions, de ne pas retrouver un gros automate unique là ou plusieurs plus petits auraient simplifié le programme.



IV.3. Décentralisation du traitement

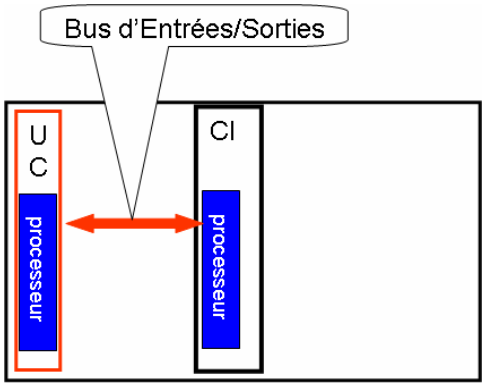
Une Partie Commande peut commander plusieurs Parties Opératives, mais une Partie Opérative ne peut être en liaison avec plusieurs Parties Commandes.
Dans ce cas plusieurs PC coopèrent pour le fonctionnement d'un système complexe, faisant appel à différentes formes de commandes (API, commande d'axes, régulation, robot …)
Dans cette structure un réseau est nécessaire pour établir une coordination étroite entre les différents équipements.
Ce type d'architecture est particulièrement adapté à la coordination de machines indépendantes ou pour des systèmes « découpables » en sous-ensembles fonctionnels.



Dans le découpage d'une installation automatisée, il faut respecter la règle hiérarchique suivante :
Une Partie Commande peut commander plusieurs Parties Opératives, mais une Partie Opérative ne peut être en liaison avec plusieurs Parties Commandes.

V. Les coupleurs intelligents

Les coupleurs intelligents réalisent en plus du traitement du signal un traitement de l'information reçue pour mettre à la disposition du processeur une information plus élaborée ou, réciproquement, qui élaborera à partir des informations du processeur un signal adapté au pré actionneur.
Chaque coupleur utilise pour ce traitement un microprocesseur. Cette prise en charge par le coupleur, d'une partie du traitement, soulage le processeur de l'Unité Centrale et améliore les performances de l'automate programmable.



V.1. Coupleur analogique

Les modules analogiques permettent d'établir la correspondance entre valeurs numériques et grandeurs analogiques (courant ou tension). Voir cours Analogique TS2.

V.2 Coupleur de régulation

Ce sont des applications particulières des cartes d'entrées/sorties analogiques qui permettent de faire de la régulation de température en utilisant des régulateurs du type PID.

V.3 Coupleur de comptage rapide

Ces coupleurs permettent de compter ou de décompter des impulsions émises par des codeurs de type "incrémental". Ils sont appelés "cartes de comptage rapide" et permettent du comptage indépendant du temps de cycle automate, avec une fréquence pouvant aller jusqu'à 100kHz.

V.4 Coupleur de positionnement

Plus connu sous le nom de **"carte d'axe"** par analogie avec les fonctions remplies par les axes de machines outils à commande numérique, ils permettent d'assurer le cycle de positionnement complet d'un mobile à partir de paramètres envoyés par l'UC.

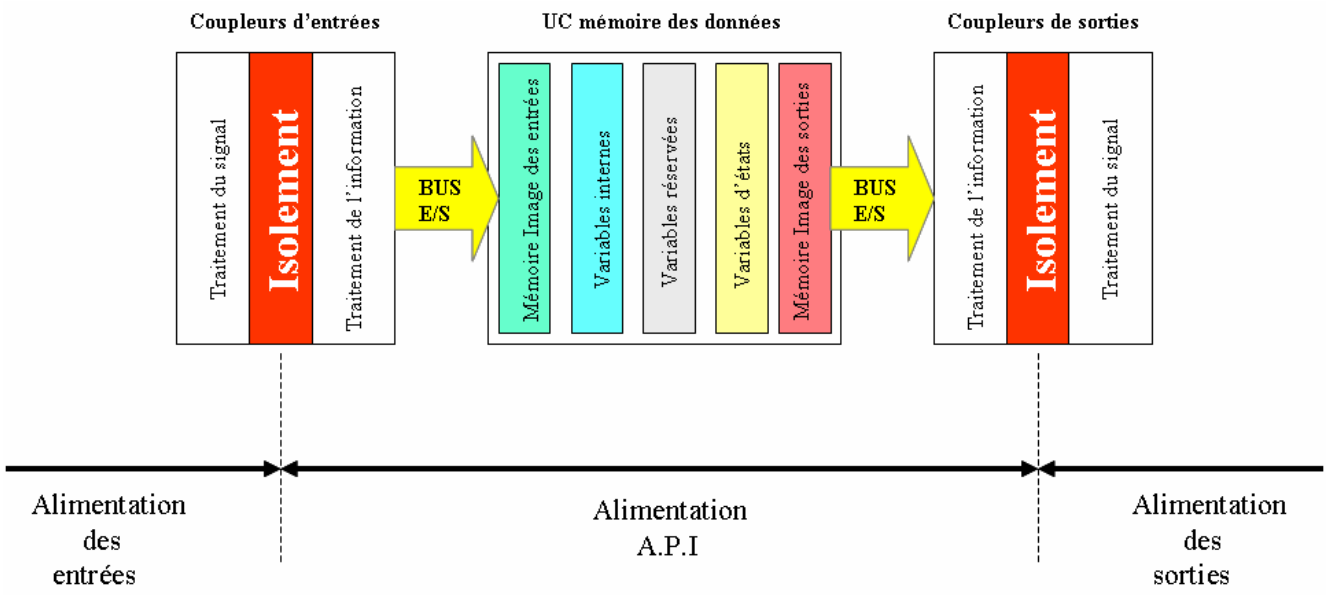
V.5 Coupleur de communication

Lorsque le volume d'informations à échanger devient important, dialogue entre un API et des périphériques type écran, clavier ou échange avec un autre API. On utilise :
- des coupleurs de communication série (ce coupleur n'intègre pas tout le protocole d'échange, une partie reste gérée par le programme utilisateur).
- des coupleurs réseaux (ce coupleur prend en charge tout le protocole d'échange qui est **"transparent"** pour l'utilisateur). Voir cours Réseaux Industriels TS2.

VI. Synthèse

La structure générale des coupleurs peut se représenter par deux modules, isolés l'un par rapport à l'autre, qui sont:

- le module **traitement du signal**,
- le module **traitement de l'information**.

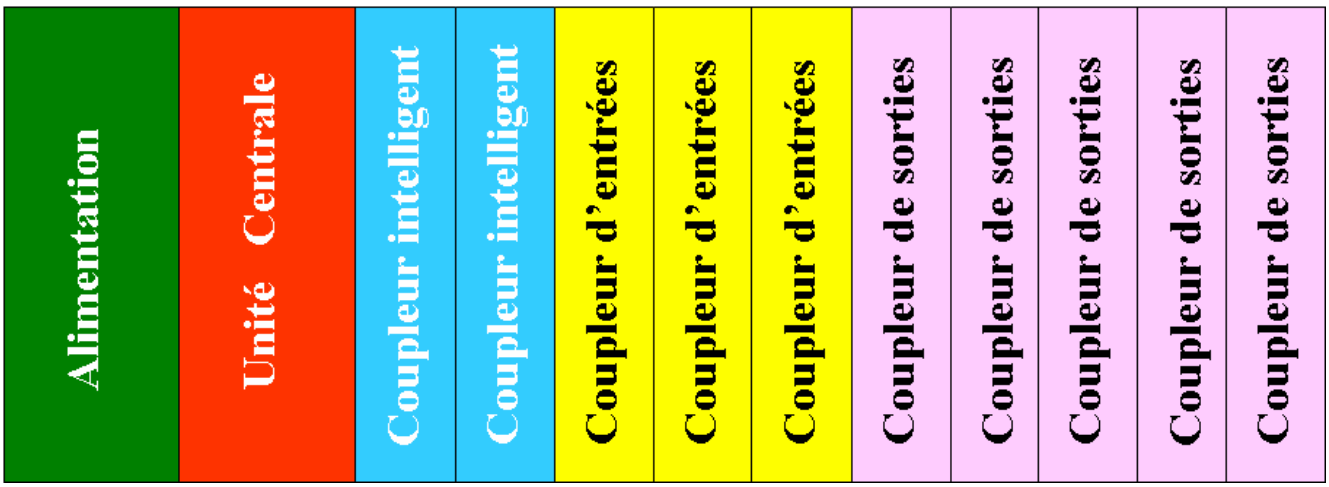


La définition d'un automate programmable commence par le choix des coupleurs appropriés aux signaux à traiter.

Configuration d'un automate

La configuration se monte en partant de l'alimentation :

- Alimentation
- Unité Centrale
- Coupleurs intelligents placés de suite après l'UC
- Coupleurs d'E/S



VII. Mise en énergie d'un système automatisé

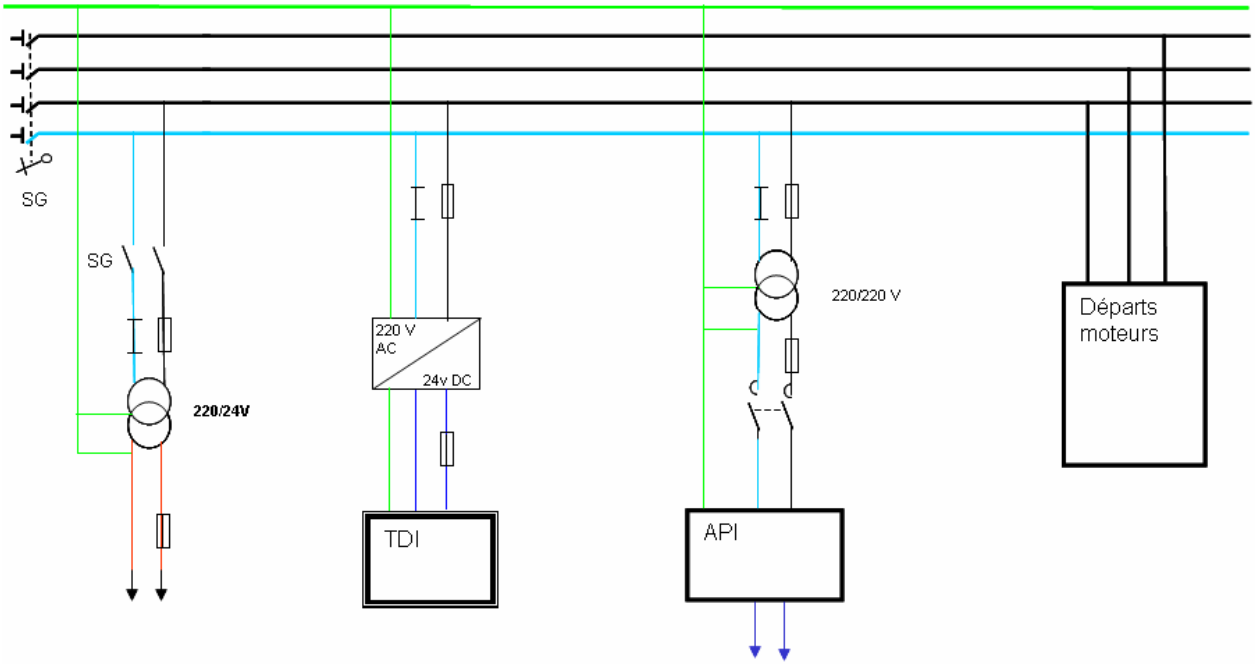
La majorité des systèmes automatisés sont actuellement basés sur un mixage d'énergie électrique et pneumatique. La distinction entre énergie de puissance et énergie de commande se fait arbitrairement par les valeurs de tensions ou de pressions utilisées.
En puissance les principales énergies utilisées sont:

- Électrique, 220/240 monophasé et 380/400 triphasé,
- Pneumatique, valeur industrielle entre 5 et 8 bars,
- Hydraulique, pression de 25 à 250 bars, plus dans les systèmes de presse.

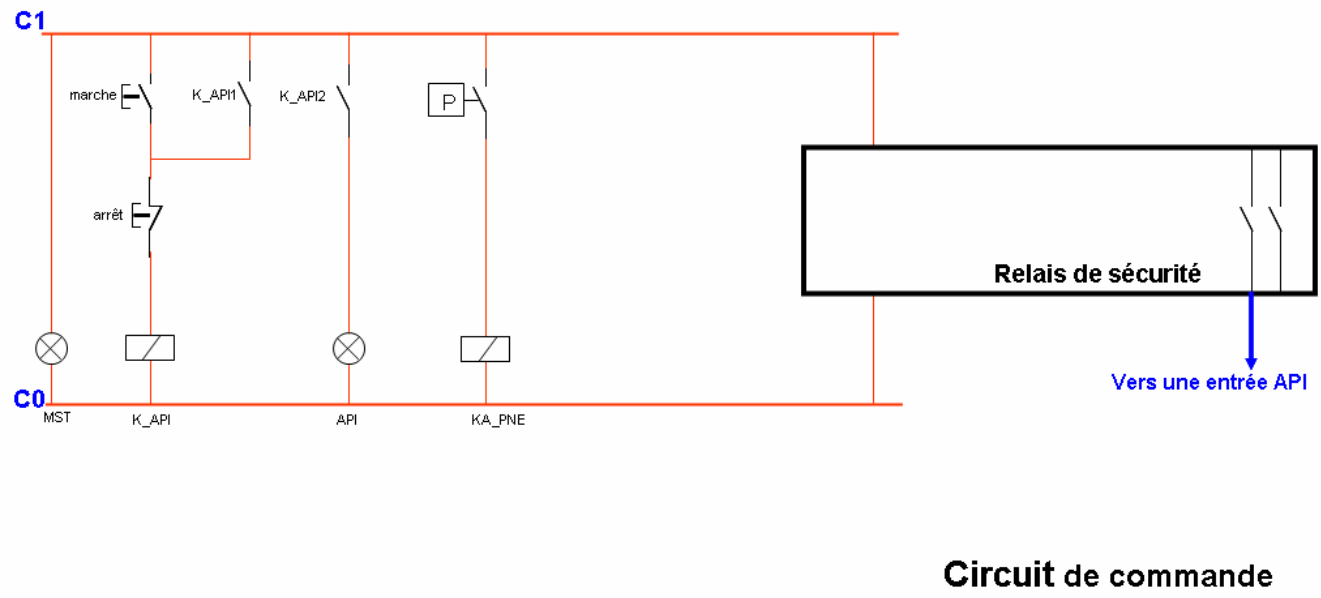
En commande on utilise essentiellement l'énergie électrique en continu ou alternatif :

- On utilise le 24V alternatif pour la partie électromécanique des préactionneurs et pour le circuit câblé de mise en service du système,
- On utilise le 24V continu (généralement fourni par l'API) pour l'alimentation des entrées via les capteurs.

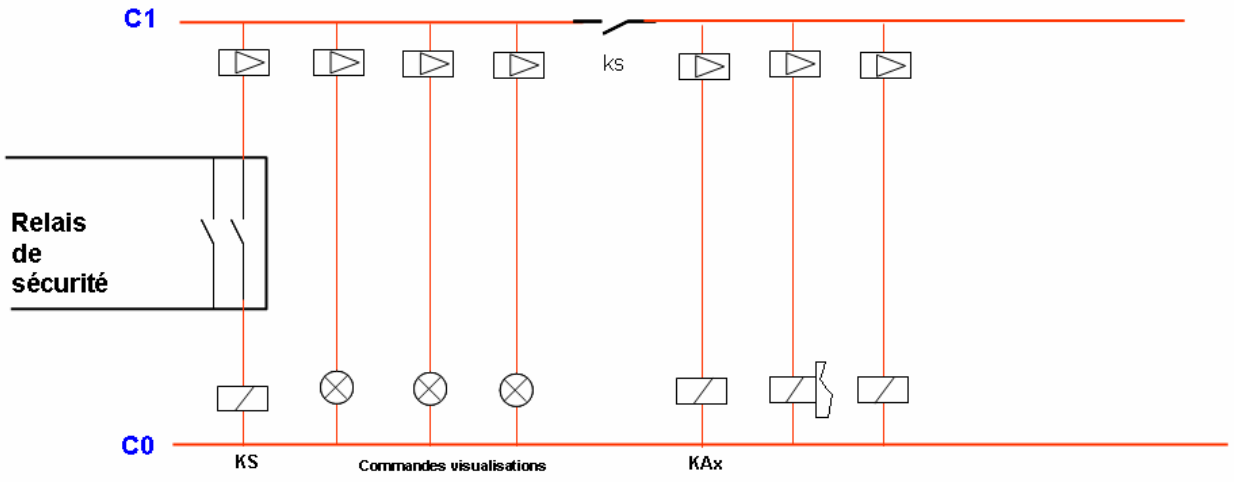
VII .I. Schéma générique d'une alimentation électrique



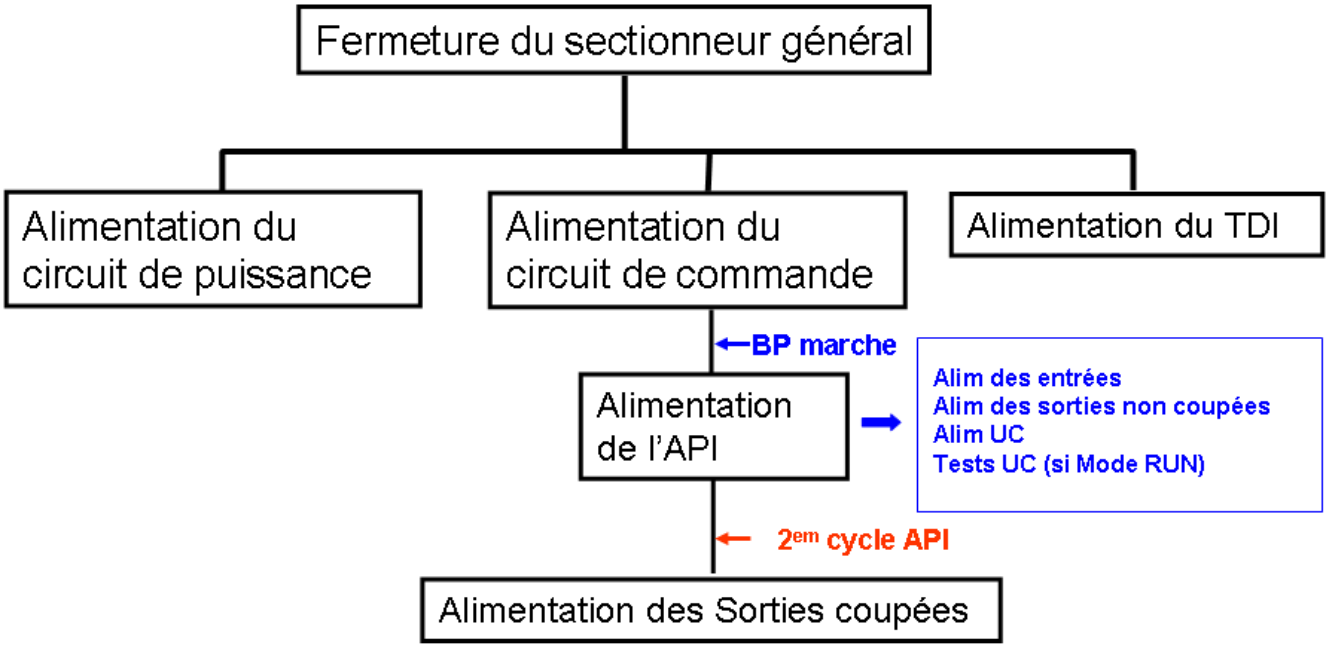
La mise en énergie d'un système automatisé piloté par un automate programmable doit se faire en respectant un ordre chronologique, symbolisé par le diagramme et les schémas de principe suivants



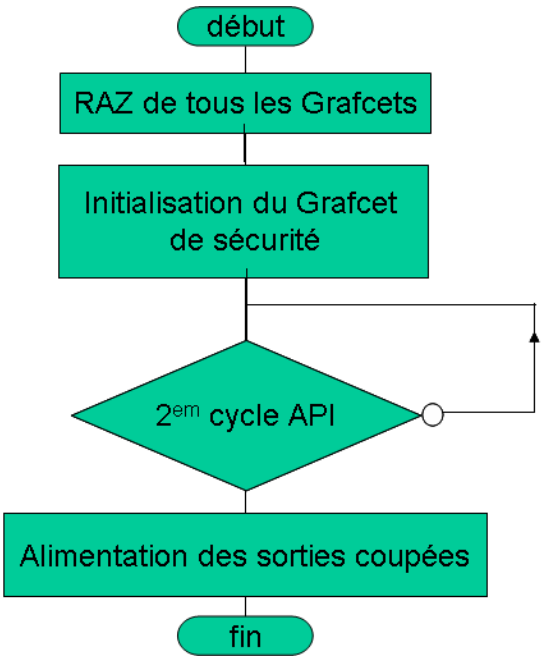
Circuit de commande



Circuit des sorties



L'application logicielle est elle aussi soumise à un minimum de méthodologie au moment de la mise sous tension. L'organigramme ci-dessous indique le principe de la partie logicielle gérant la mise sous tension.



VII .II. Schéma générique d'une alimentation pneumatique