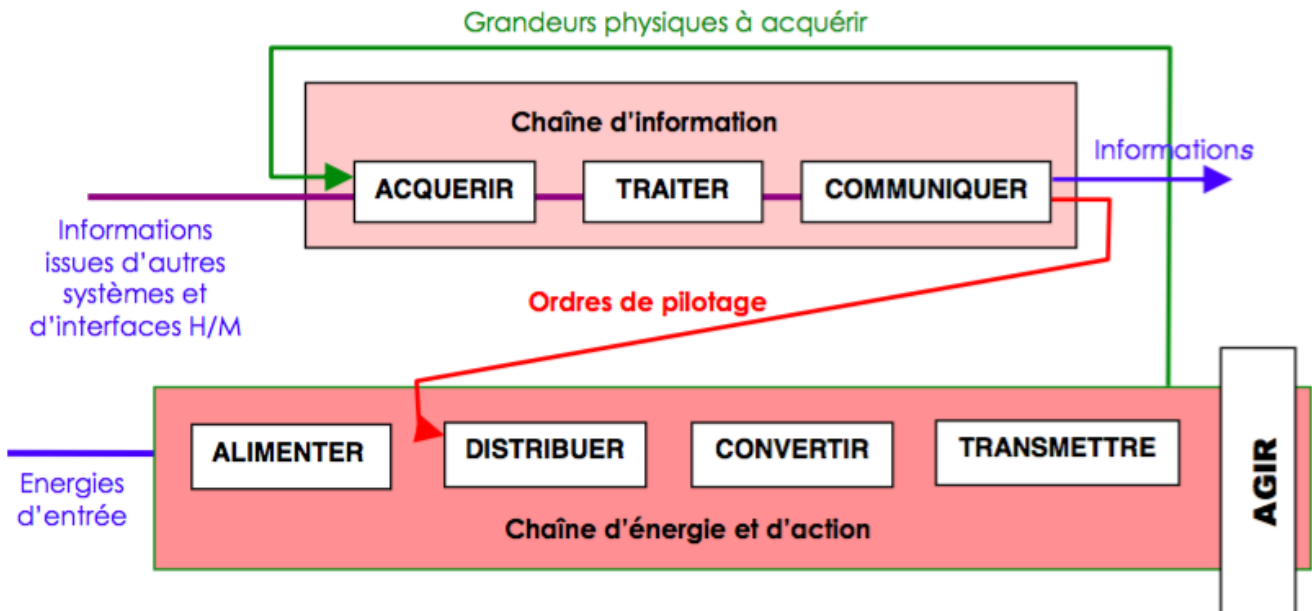


Traitement de l'information

Nous avons vu dans les cours précédents que les systèmes pluritechniques possèdent une chaîne d'information et une chaîne d'énergie.

Ce cours va permettre de comprendre comment coder une information, la traiter afin de commander la chaîne d'énergie.



Introduction aux systèmes logiques

I . LES SYSTEMES LOGIQUES

Un système à logique est un système qui traite l'information de façon numérique. C'est donc la fonction **TRAITER** de la chaîne d'informations qui décrit le comportement logique d'un système.

L'objectif est de traduire le comportement du système sous la forme d'équations ou de représentations graphiques, qui seront implantées dans le composant réalisant la fonction Traiter.

Vocabulaire :

Variables d'entrées : Ce sont les informations fournies par la fonction Acquérir. Ce sont donc des entrées de la fonction Traiter.

Variables de sorties : Ce sont les informations fournies à la fonction Communiquer. Ce sont les sorties de la fonction Traiter.

Variables binaires : Les variables binaires sont des variables (d'entrées et de sorties) qui ne peuvent prendre que 2 valeurs (0 ou 1). On parle alors de niveaux logiques.

Grandeurs analogiques : C'est une grandeur physique qui peut varier de façon continue en fonction du temps. (ex : température, vitesse, tension aux bornes d'une prise réseau, ...)

Grandeurs binaires : C'est une grandeur physique à laquelle on associe deux valeurs conventionnelles 0 et 1.

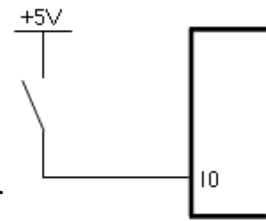
Exemple d'un signal électrique : - 0V → 0 logique

- 5V → 1 logique

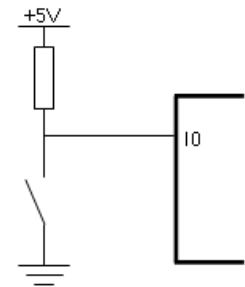
Lorsque l'on souhaitera traiter des grandeurs analogiques, il sera nécessaire de les transformer en grandeurs binaires. Nous verrons à la fin du cours comment passer d'un système de numération à un autre.

Logique positive et logique négative :

On parle de logique positive, lorsqu'une entrée ou une sortie est active lorsque le niveau logique associé est un « 1 » logique (présence d'une tension par exemple). Pareillement, elle est dite inactive, lorsque le niveau logique associé est un « 0 » (absence d'une tension par exemple). C'est le cas le plus utilisé.



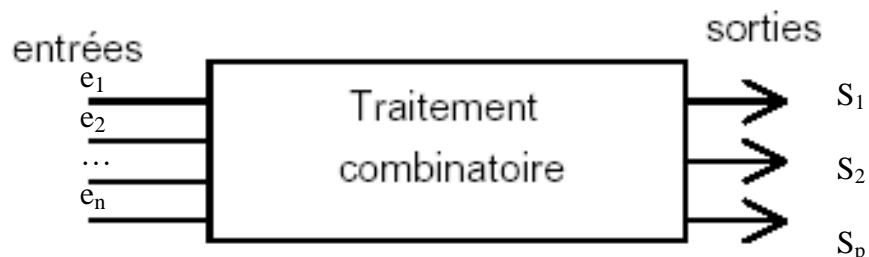
On parle de logique négative, lorsque une entrée ou une sortie est active lorsque le niveau logique associé est un « 0 » logique (absence d'une tension par exemple). Pareillement, elle est dite inactive, lorsque le niveau logique associé est un « 1 » (présence d'une tension par exemple).



II . LOGIQUE COMBINATOIRE ET SEQUENTIELLE

I. Logique combinatoire :

Un système est dit à logique combinatoire si les variables de sorties ne dépendent que des variables d'entrées.



Explication : La même cause (même combinaison des entrées e_1, e_2, \dots, e_n) produit toujours le même effet (même état des sorties S_1, S_2, \dots, S_p). L'effet disparaît lorsque la cause disparaît.

Le but étant de traduire le comportement du système, il faut donc définir les fonctions logiques reliant les sorties aux entrées.

$$S_1 = f_1(e_1, e_2, \dots, e_n)$$

$$S_2 = f_2(e_1, e_2, \dots, e_n)$$

.

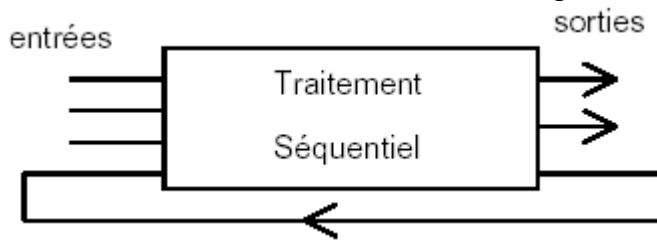
$$S_p = f_p(e_1, e_2, \dots, e_n)$$

Les fonctions f_1, f_2, \dots, f_p sont appelées fonctions logiques, car elles manipulent des variables logiques binaires. Ces fonctions permettent de décrire le comportement global du système étudié.

Exemples : machine outil, afficheur 7 segments...

II. Logique séquentielle :

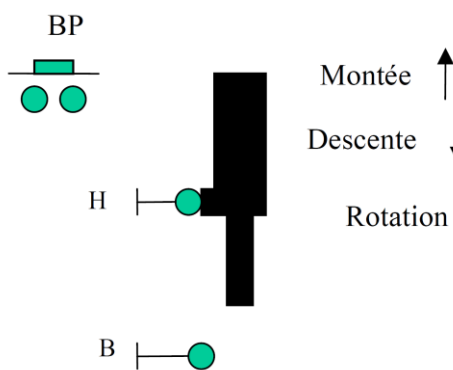
Un système est dit à logique séquentielle, lorsque la ou les sorties dépendent de la combinaison des entrées mais aussi de l'état précédent des sorties et de la variable temps.



Explication : Une même cause (même combinaison des entrées e_1, e_2, \dots, e_n) peut produire des effets différents (états différents des sorties S_1, S_2, \dots, S_p). L'effet peut persister si la cause disparaît.

$$S_i = f(e_1, e_2, \dots, e_n; S_1, S_2, \dots, S_p; t)$$

1. Exemple 1 : Perceuse à colonne



Une perceuse à colonne permet de percer des pièces métalliques.

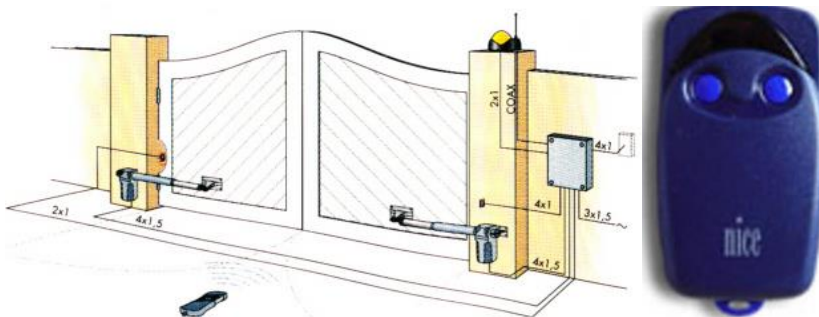
La mise en fonctionnement de la perceuse est réalisée par un appui bref sur le bouton BP. Ceci provoque la mise en rotation du mandrin.

La perceuse en position haute (capteur H) descend alors jusqu'en position basse (capteur B), puis remonte en position haute. L'arrêt de la perceuse après un cycle de perçage n'est autorisé que lorsque la perceuse est en position haute, et que l'utilisateur ré appuie sur le bouton BP.

Identifier les variables binaires de ce système :

Justifier que le comportement du système est celui d'un système à logique séquentielle.

2. Exemple 2 : Télécommande d'un portail à deux battants



Une télécommande d'un portail à 2 battants est composée d'un 1^{er} bouton poussoir gérant l'ouverture fermeture d'un seul battant, et un 2nd bouton poussoir gérant l'ouverture fermeture des 2 battants.

Quel est le type de logique d'un tel système ? Justifier votre réponse.

Systèmes à logique combinatoire

III DECRIRE UN SYSTEME A LOGIQUE COMBINATOIRE

L'outil mathématique qui permet de décrire les systèmes combinatoires est l'**algèbre de Boole**. Par la combinaison des quatre opérateurs (appelés aussi fonctions logiques) de base que sont le *NON*, le *OU* (inclusif) et le *ET*, nous allons pouvoir décrire chacune des sorties du système étudié en fonction des entrées.

La description du comportement d'un système combinatoire peut être faite soit par :

- ✓ Une table de vérité ;
- ✓ Une équation logique avec les opérateurs logiques de base;
- ✓ Un schéma à contacts ;
- ✓ Un logigramme ;
- ✓ Un chronogramme.

I. La table de vérité

C'est un tableau donnant l'état des variables de sorties d'un système logique en fonction de l'ensemble des états possibles des variables d'entrées.

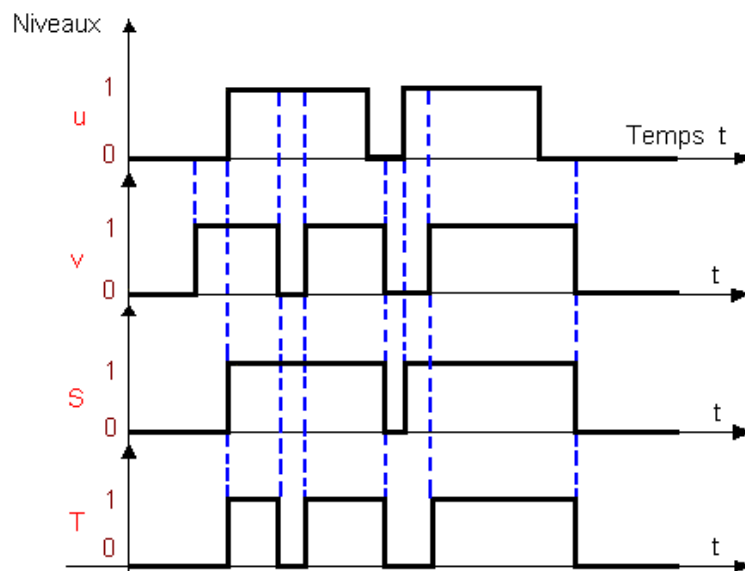
Exemple : Système à 2 variables d'entrées (a et b) et 2 variables de sorties (X et Y)

Entrées		Sorties	
a	b	X	Y
0	0	1	1
0	1	0	1
1	0	0	0
1	1	1	1

II. Le chronogramme

C'est une représentation graphique qui décrit les évolutions temporelles des variables d'entrées et de sorties du système étudié.

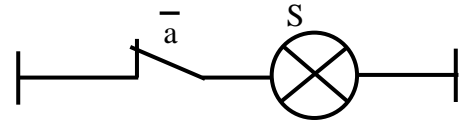
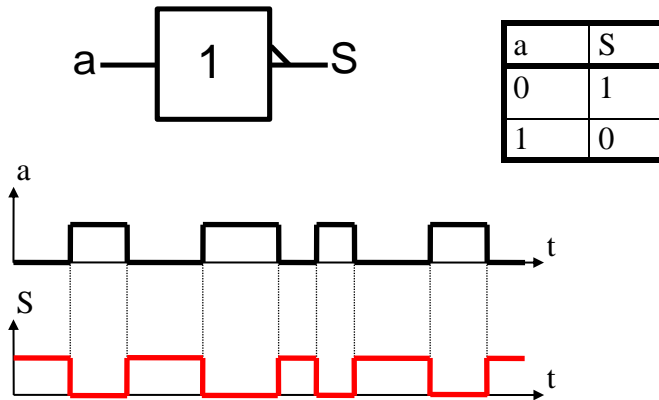
Exemple de chronogramme à 4 variables (u, v, S et T) :



IV LES OPERATEURS LOGIQUES DE BASE

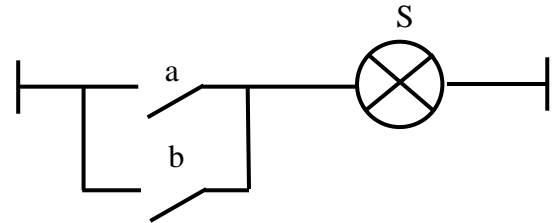
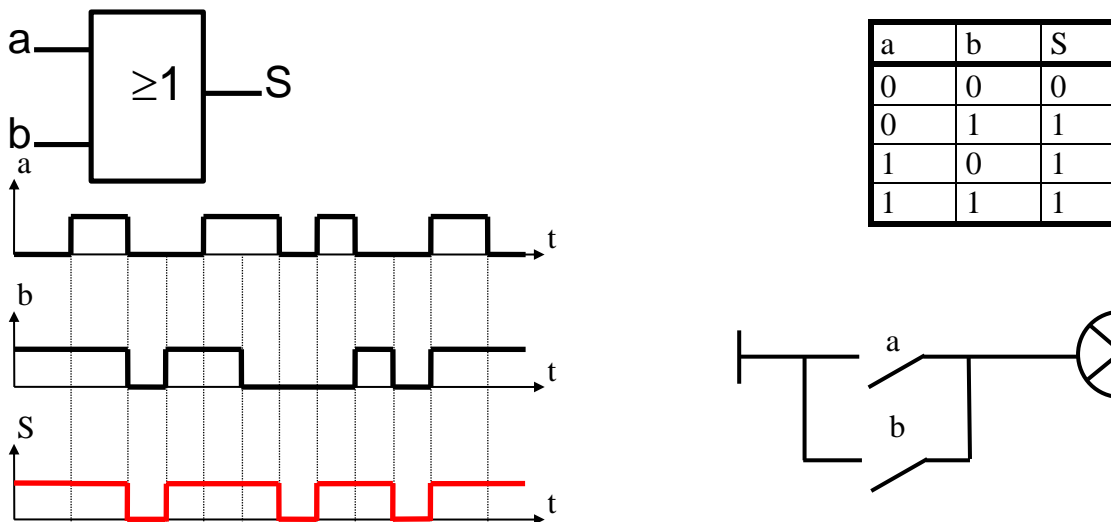
I. La fonction NON : $S = \bar{a}$

Cette fonction complémente le niveau logique présent sur son entrée.



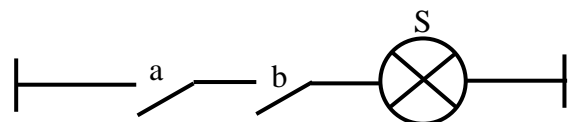
II. La fonction OU Inclusif : $S = a + b$

Cette fonction présente un niveau logique haut sur sa sortie dès qu'au moins l'une de ses entrées est au niveau logique haut.



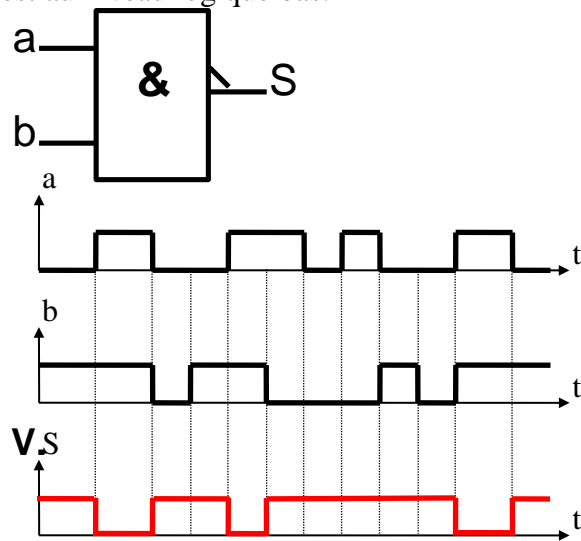
III. La fonction ET : $S = a.b$

Cette fonction positionne sa sortie au niveau logique haut si ses entrées sont au niveau haut.

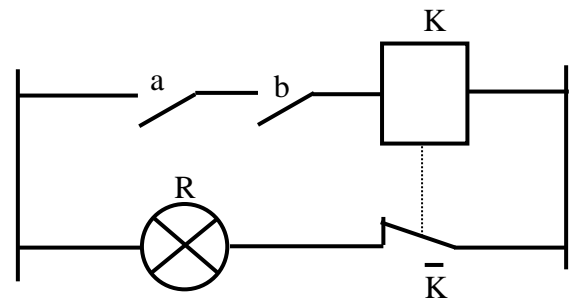


IV. La fonction NON ET : $S = \overline{a \cdot b}$

Cette fonction présente un niveau logique haut en sortie lorsqu'au moins l'une de ses entrées est au niveau logique bas.

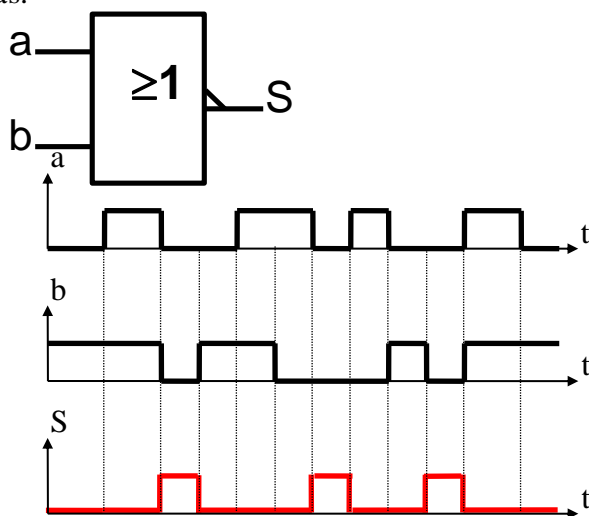


a	b	S
0	0	1
0	1	1
1	0	1
1	1	0

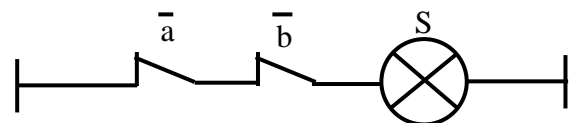


VI. La fonction NON OU : $S = \overline{a + b}$

Cette fonction présente un niveau logique haut en sortie si ses entrées sont au niveau logique bas.



a	b	S
0	0	1
0	1	0
1	0	0
1	1	0

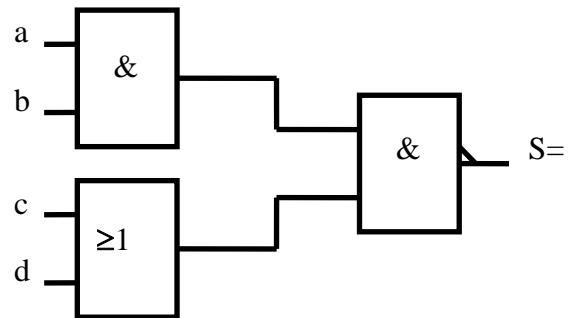
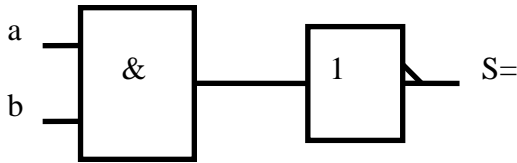


Remarque : les fonctions logiques de base présentées ci-dessus, possèdent 2 entrées, mais il est possible d'en avoir 3 ou 4.

V ASSOCIATION DE FONCTIONS LOGIQUES DE BASE

Il est possible d'associer des opérateurs logiques de base en les connectant en « cascade ». La sortie du premier opérateur logique devient alors l'entrée du second.

Calculer S dans les schémas suivants :



VI PROPRIETES ET EQUATIONS (ALGEBRE DE BOOLE).

4.1 La commutativité :

$$a + b = b + a$$

$$a \bullet b = b \bullet a$$

4.2 L'associativité :

$$(a + b) + c = a + (b + c)$$

$$(a \bullet b) \bullet c = a \bullet (b \bullet c)$$

4.3 La distributivité :

$$a \bullet (b + c) = (a \bullet b) + (a \bullet c)$$

$$a + (b \bullet c) = (a + b) \bullet (a + c)$$

4.4 Les éléments neutres :

$$a + 0 = a$$

$$a \bullet 1 = a$$

4.5 La complémentation :

$$a + \bar{a} = 1$$

$$a \bullet \bar{a} = 0$$

4.6 L'idempotence :

$$a + a = a$$

$$a \bullet a = a$$

4.7 Les éléments absorbants :

$$a \bullet 0 = 0$$

$$a + 1 = 1$$

4.8 L'involution :

$$\overline{(\bar{a})} = a$$

$$\overline{(\overline{a})} = a$$

4.9 L'inclusion :

$$a \bullet b + a \bullet \bar{b} = a$$

4.10 Théorème de De Morgan :

$$\overline{a + b} = \bar{a} \bullet \bar{b}$$

$$\overline{a \bullet b} = \bar{a} + \bar{b}$$

VII DETERMINATION DE FONCTIONS LOGIQUES A BASE D'UNE TABLE DE VERITE

En supposant que, suite à une étude d'un cahier des charges, nous obtenons la table de vérité suivante :

a	b	S
0	0	0
0	1	1
1	0	0
1	1	0

Nous nous apercevons que $S = 1$ quand $a = 0$ et $b = 1$.

Si nous codons $S=1$ par \bar{S} , $a=0$ par \bar{a} et $b=1$ par b , nous pouvons écrire :

$$\bar{S} = \bar{a} \cdot b$$

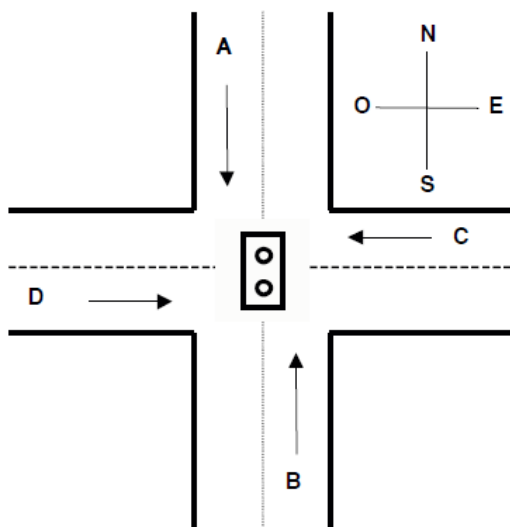
Selon la même méthode, écrivez l'équation logique pour les 2 tables de vérité suivantes :

a	b	S
0	0	0
0	1	0
1	0	1
1	1	0

a	b	S
0	0	1
0	1	0
1	0	1
1	1	1

VIII DEMARCHE DE DESCRIPTION DU COMPORTEMENT

Exemple : Intersection entre une route principale et une route secondaire :



Des capteurs de présence de voitures ont été placés le long des voies :

- c et d pour la route principale ;
- a et b pour la route secondaire.

Ce sont les variables d'entrées.

Les variables logiques de ces capteurs sont à 1 quand il y a des voitures, à 0 quand il n'y en a pas.

Les feux de circulation Est-Ouest (E-O) et Nord-Sud (N-S) sont les variables de sorties. On impose à la variable E-O la valeur binaire 0 lorsque le feu est Rouge, et la valeur binaire 1 lorsque le feu est Vert. De la même façon pour la variable binaire N-S.

Le comportement souhaité (élément du cahier des charges) des feux à cette intersection est décrit par les règles suivantes :

- S'il y a des voitures dans la voie C et dans la voie D alors le feu E-O est vert ;
- S'il y a des voitures dans la voie C ou dans la voie D, et dans la voie A ou dans la voie B mais pas dans les deux alors le feu FE-O est vert ;
- S'il y a des voitures dans la voie A et dans la voie B et dans la voie C ou dans la voie D mais pas dans les deux alors le feu N-S est vert ;
- S'il y a des voitures dans la voie A ou dans la voie B et aucune dans les voies C et D alors le feu N-S est vert.

Etape 1 : Définir la frontière d'étude du système étudié.

On n'étudie que la commande des feux E-O et N-S en fonction des capteurs a, b, c et d. Les commandes des passages piétons ne sont pas prises en compte.

Etape 2 : Recenser les grandeurs d'entrées et de sorties, leurs associer des noms de variables binaires (les noms associés aux variables sont parfois appelés mnémoniques) et définir la logique utilisée.

Grandeurs binaires	Mnémoniques	Logique utilisée
Présence voiture dans la voie A	Va	Logique positive
Présence voiture dans la voie A	Vb	Logique positive
Présence voiture dans la voie A	Vc	Logique positive
Présence voiture dans la voie A	Vd	Logique positive
Commande du feu Est-Ouest	FEO	Logique positive
Commande du feu Nord-Sud	FNS	Logique positive

Etape 3 : Traduire le comportement observé ou le comportement souhaité à l'aide d'une table de vérité et en déduire les fonctions logiques reliant les variables de sorties aux variables d'entrées.

Combien de lignes faut-il dans la table de vérité ?

Il y a 4 variables binaires en entrée, donc il y a 2^4 combinaisons, soit 16 lignes.

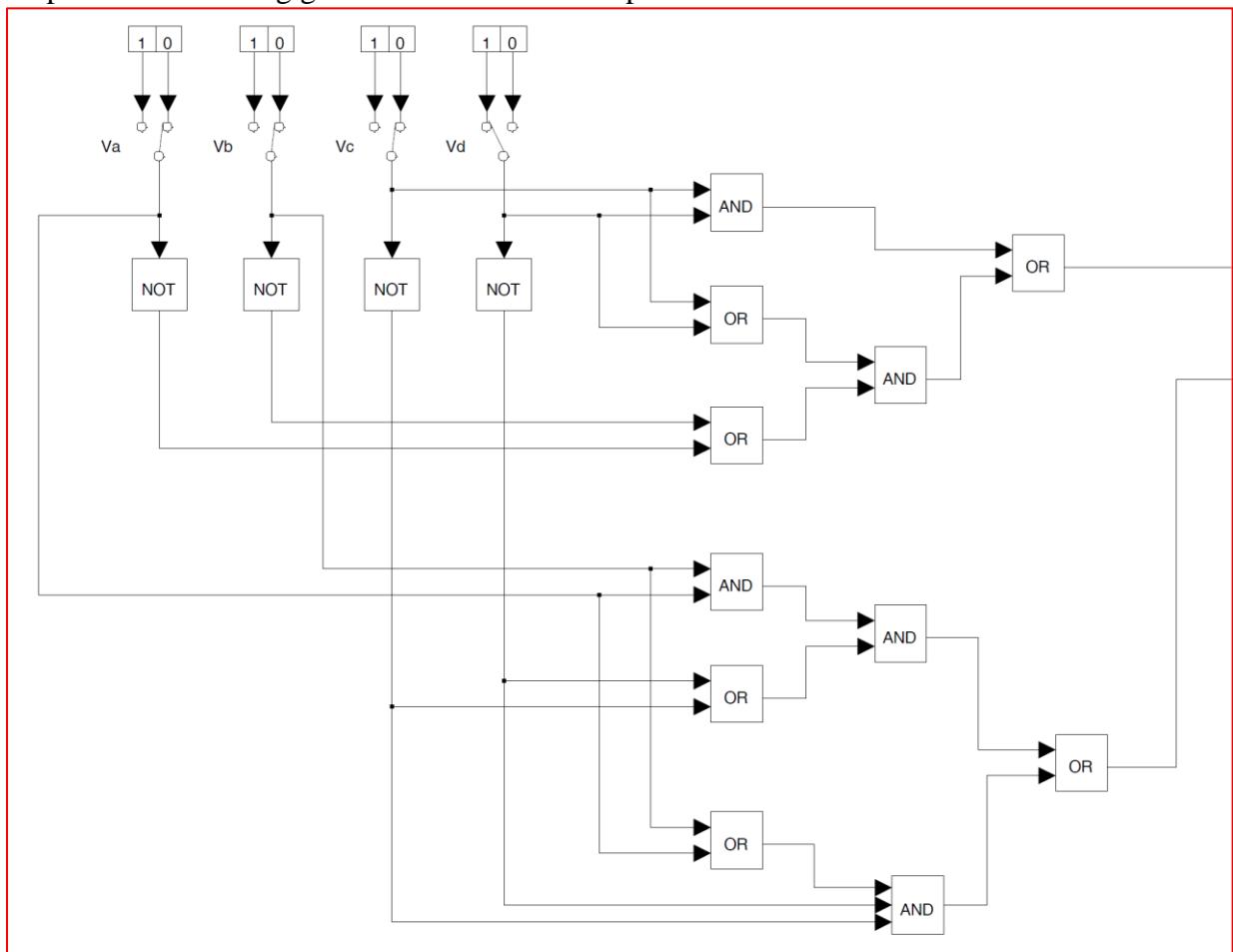
Va	Vb	Vc	Vd	FEO	FNS
0	0	0	0	0	0
0	0	0	1	1	0
0	0	1	0	1	0
0	0	1	1	1	0
0	1	0	0	0	1
0	1	0	1	1	0
0	1	1	0	1	0
0	1	1	1	1	0
1	0	0	0	0	1
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	0	1
1	1	0	1	0	1
1	1	1	0	0	1
1	1	1	1	1	0

Quelles sont les fonctions logiques liant les variables d'entrées aux variables de sorties ?

$$FEO = Vc.Vd + (Vc + Vd).(Va + Vb)$$

$$FNS = Va.Vb.(Vc + Vd) + (Va + Vb).Vc.Vd$$

Etape 4 : Etablir le logigramme décrivant le comportement.



Description des systèmes à logique séquentielle

Problématique : Comment décrire le comportement séquentiel d'un système ?

La description du comportement d'un système séquentiel peut être faite soit par :

- ✓ Un schéma électrique ;
- ✓ Un GRAFCET ;
- ✓ Un graphe d'état ;
- ✓ Un algorithme (ou algorithme).

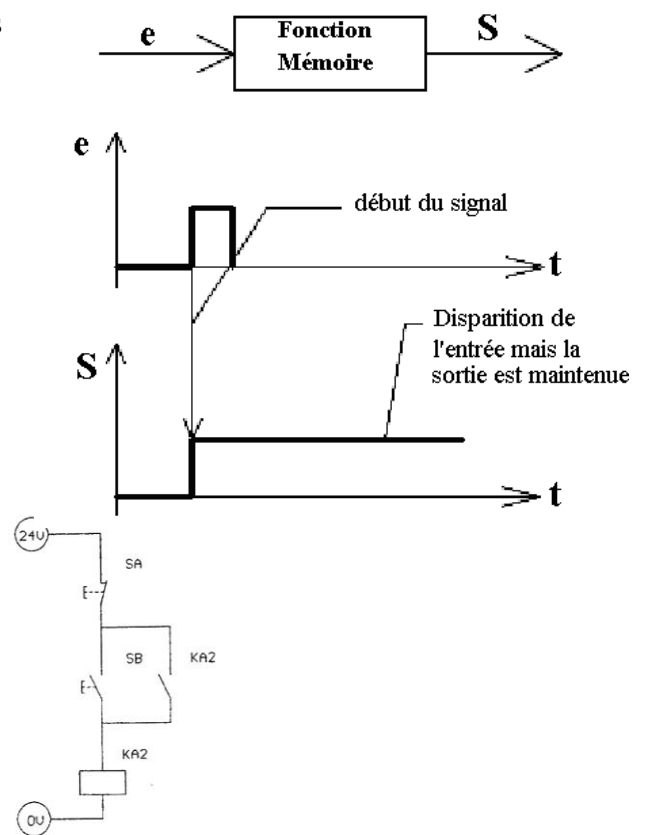
I. Rappel général

Contrairement aux systèmes à logique combinatoire, les variables de sortie dépendent des variables d'entrées, mais aussi des variables de sorties aux instants précédents et du temps. Il y a donc une des états précédents.

Par mémoire, on exprime le phénomène qui consiste à conserver l'effet d'un événement après sa disparition.

À l'apparition du signal e , la sortie change d'état. À la disparition du signal e , la sortie reste dans le même état. Le maintien de la sortie est l'effet mémoire.

Exemple de montage électrique avec effet mémoire :

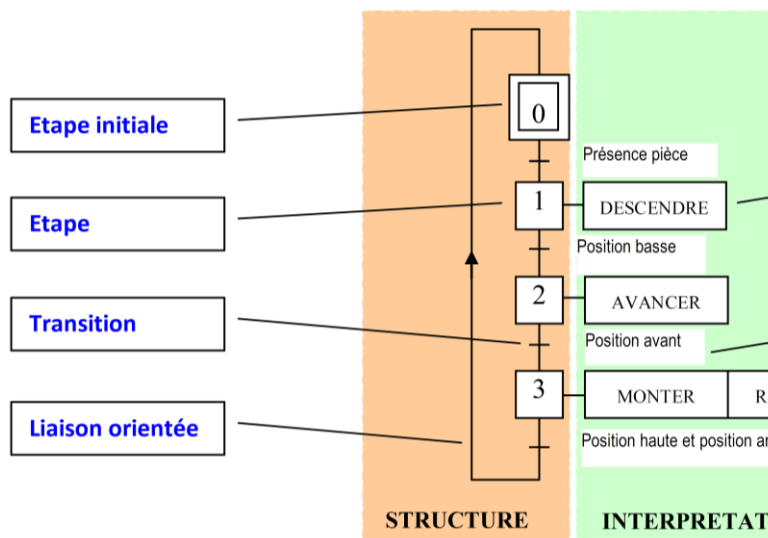


II. Description par l'outil Grafcet

Le **GRA**phe **F**onctionnel de **C**ommande **É**tapes / **T**ransitions ou **GRAFCET** est apparu dans les années 70. Il est utilisé pour représenter graphiquement l'évolution des cycles d'un système séquentiel.

Le GRAFCET se compose :

- ✓ d'**étapes** auxquelles sont associées des **actions** ;
- ✓ de **transitions** auxquelles sont associées des **conditions** (ou **réceptivités**) ;
- ✓ de **liaisons orientées** reliant les étapes aux transitions et les transitions aux étapes.



I. Règles d'évolution d'un GRAFCET

Règle 1 : Situation initiale (étapes initiales)

La situation initiale du Grafcet caractérise le comportement initial de la P.C. vis-à-vis de la P.O. Elle correspond aux étapes actives au début du fonctionnement (étapes initiales indiquées par un carré double). Il y a toujours au moins une étape initiale.

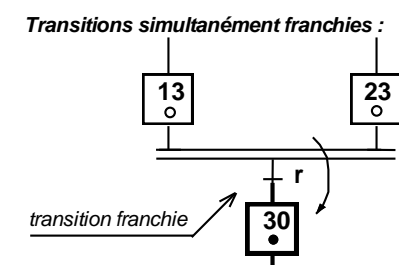
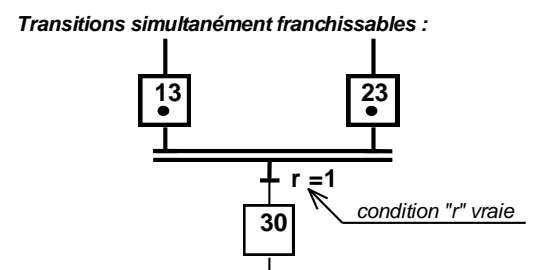
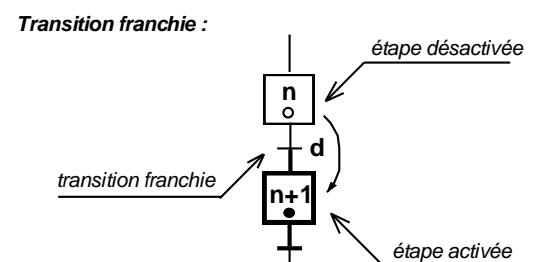
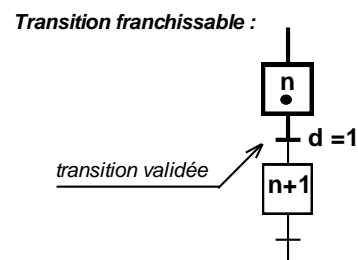
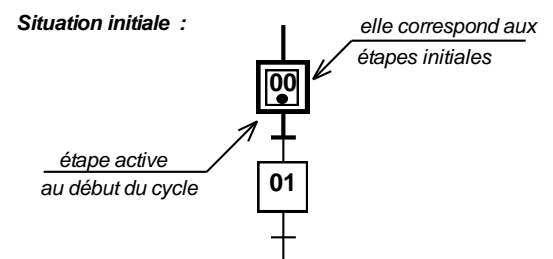
Règle 2 : Franchissement d'une transition

L'évolution de la situation du Grafcet correspond au franchissement d'une transition. On passe de l'étape **n** à l'étape **n+1** lorsque :

- ✓ la transition est validée : étapes précédentes actives.
- ✓ la réceptivité (condition) associée à cette transition est vraie.

Si ces deux conditions sont réalisées, la transition est obligatoirement franchie.

Règle 3 : Evolution des étapes actives



Le franchissement d'une transition entraîne simultanément l'activation de toutes les étapes immédiatement suivantes et la désactivation de toutes les étapes immédiatement précédentes.

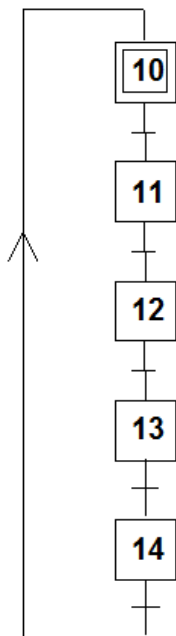
Règle 4 : Franchissements simultanés

Plusieurs transitions simultanément franchissables sont simultanément franchies. Pour montrer cette condition, on représente le groupement de liaisons par deux traits parallèles : divergence ou convergence en "ET".

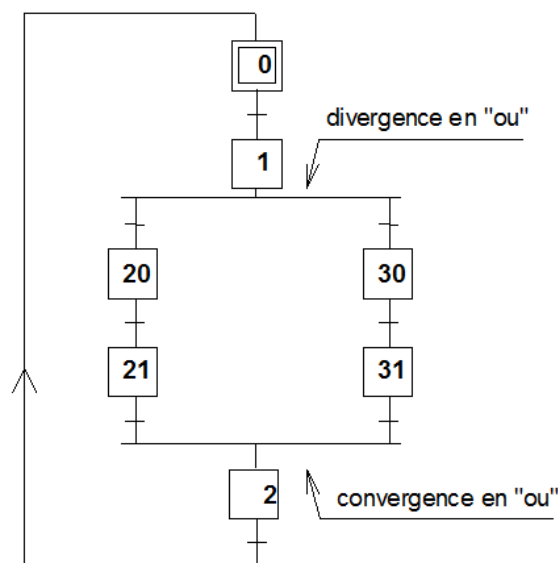
Règle 5 : Activation et désactivation simultanées

Si, au cours du fonctionnement, une étape est simultanément activée et désactivée, alors elle reste active.

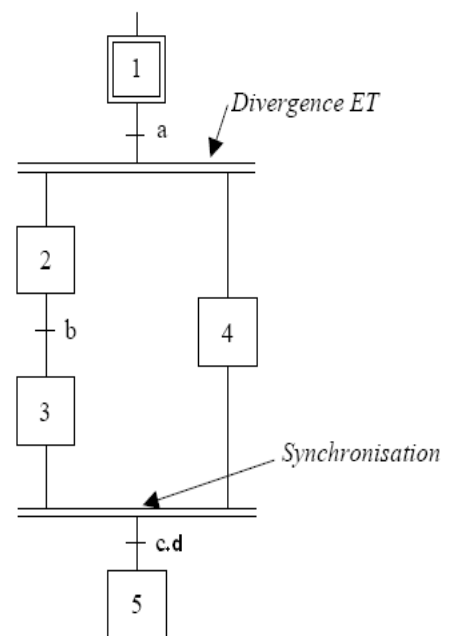
II. Structures de base d'un GRAFCET



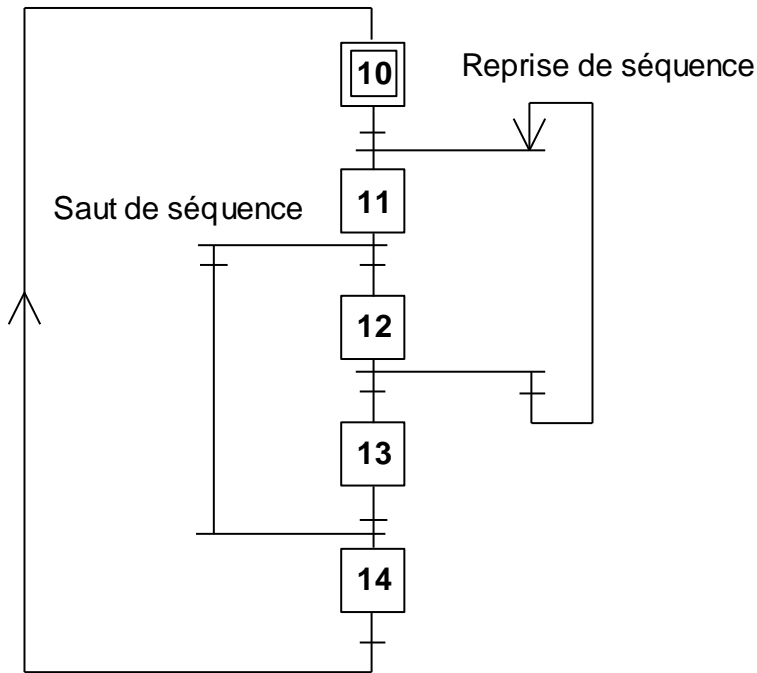
Structure linéaire :
les actions sont
réalisées les unes
après les autres



Structures alternatives : elles permettent de passer par
une branche ou l'autre, ou les deux en même temps.



1. Structure à reprise ou saut de séquences



C'est une structure qui permet :

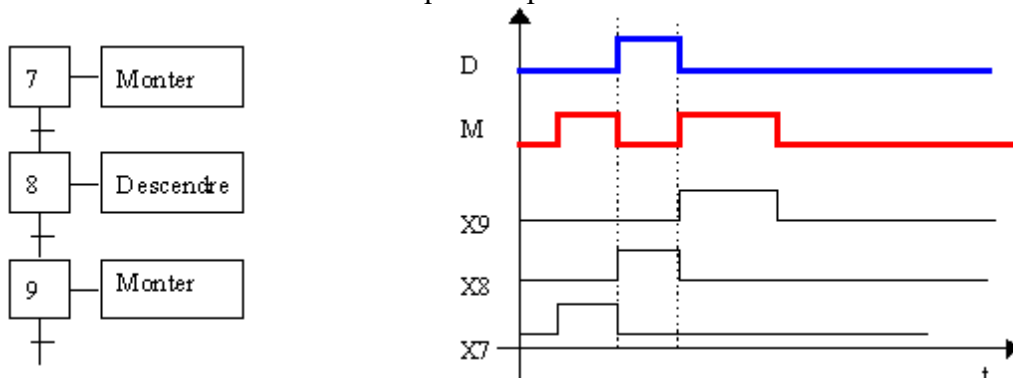
- ✓ De « sauter » certaines actions (saut de séquence);
- ✓ De réaliser plusieurs fois les mêmes actions (reprise de séquence). Elle est très utilisée lorsque l'on veut réaliser un nombre de fois défini une série d'actions (notion de compteur).

III. Description des actions associées aux étapes et des réceptivités

1. Actions associées aux étapes

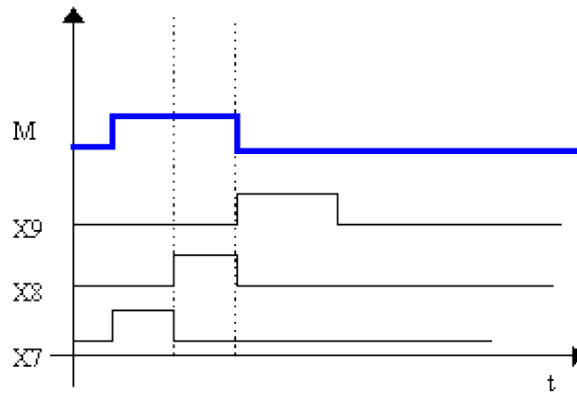
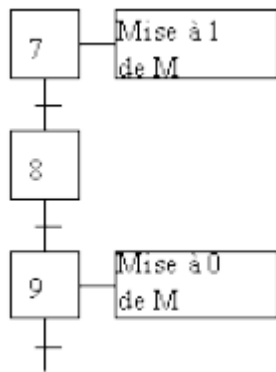
Les actions sont précisées dans un cadre lié à l'étape, de manière générale, l'action n'est vraie que si l'étape est active. La norme européenne CEI précise la nature de l'action par une lettre précisant la nature de l'action.

La ou les actions sont réalisées lorsque l'étape est active.



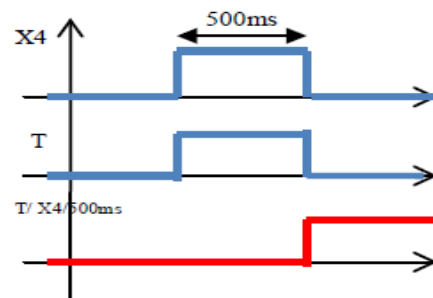
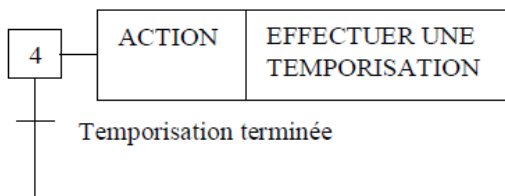
C'est le cas le plus fréquemment rencontré.

2. Actions mémorisées



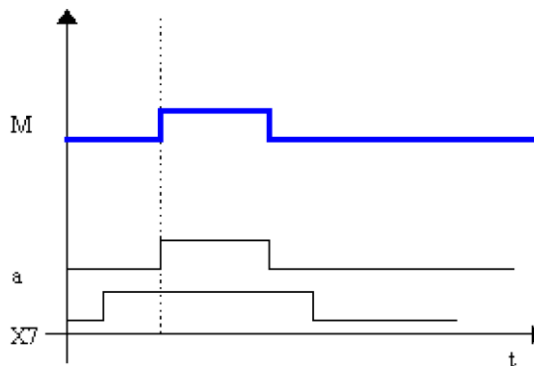
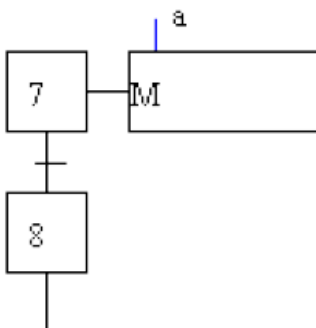
La mise à 1 correspond au Set (S) et la mise à 0 correspond au Reset (R).

3. Actions temporisées



L'action est réalisée lorsque l'étape 4 est active et durant une durée prédéterminée.

4. Actions conditionnelles



L'action M est réalisée si l'étape est active (7) et si la condition est vraie (a).

5. Réceptivités associées aux transitions

Les réceptivités sont des fonctions logiques combinatoires et une réceptivité est vraie ou fausse. Il s'agit donc d'une variable binaire.

VIII.III.5.a Réceptivités sur fronts

Parfois, il peut être intéressant dans certaines applications de s'intéresser aux fronts plutôt qu'aux niveaux (0 ou 1) d'une variable.

On parle alors de front montant ou descendant.

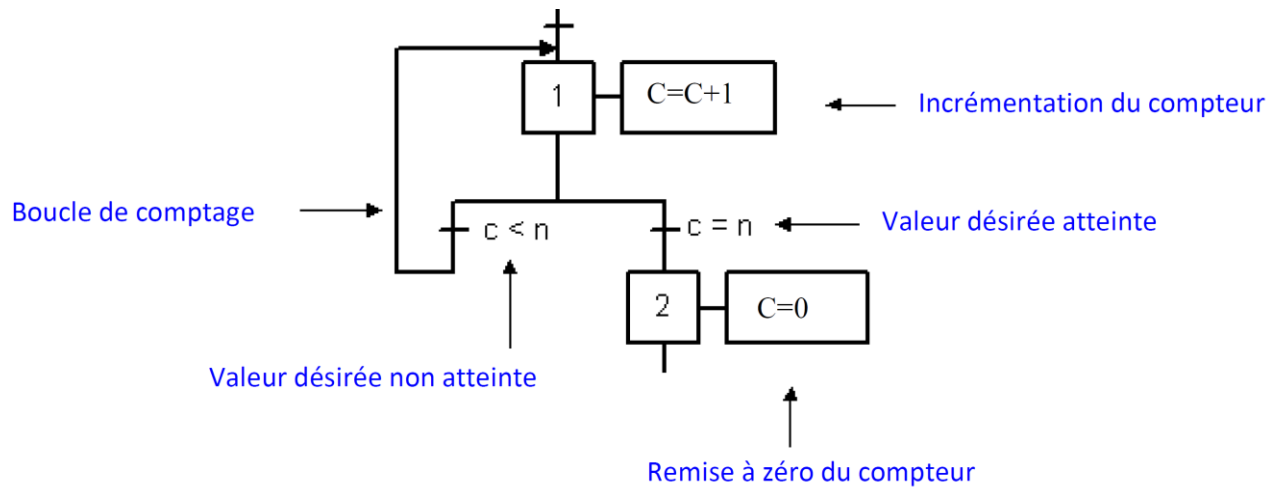
Ils sont notés par :

Front montant : $\uparrow a$ Front descendant : $\downarrow a$

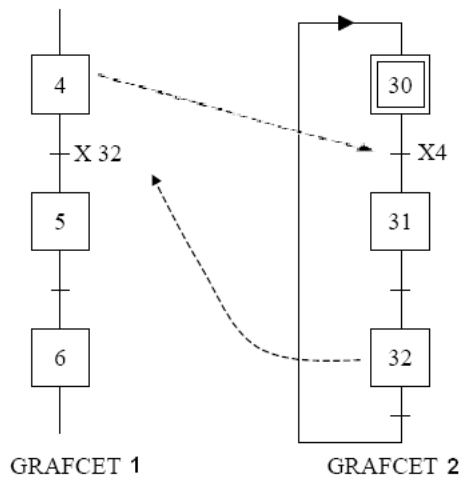
La variable $\uparrow a$ vaut 1 lorsque la variable a passe de l'état logique 0 à 1, et 0 le reste du temps.

La variable $\downarrow a$ vaut 1 lorsque la variable a passe de l'état logique 1 à 0, et 0 le reste du temps.

VIII.III.5.b Réceptivités de comptage



VIII.III.5.c Réceptivités relatives à l'état d'une étape



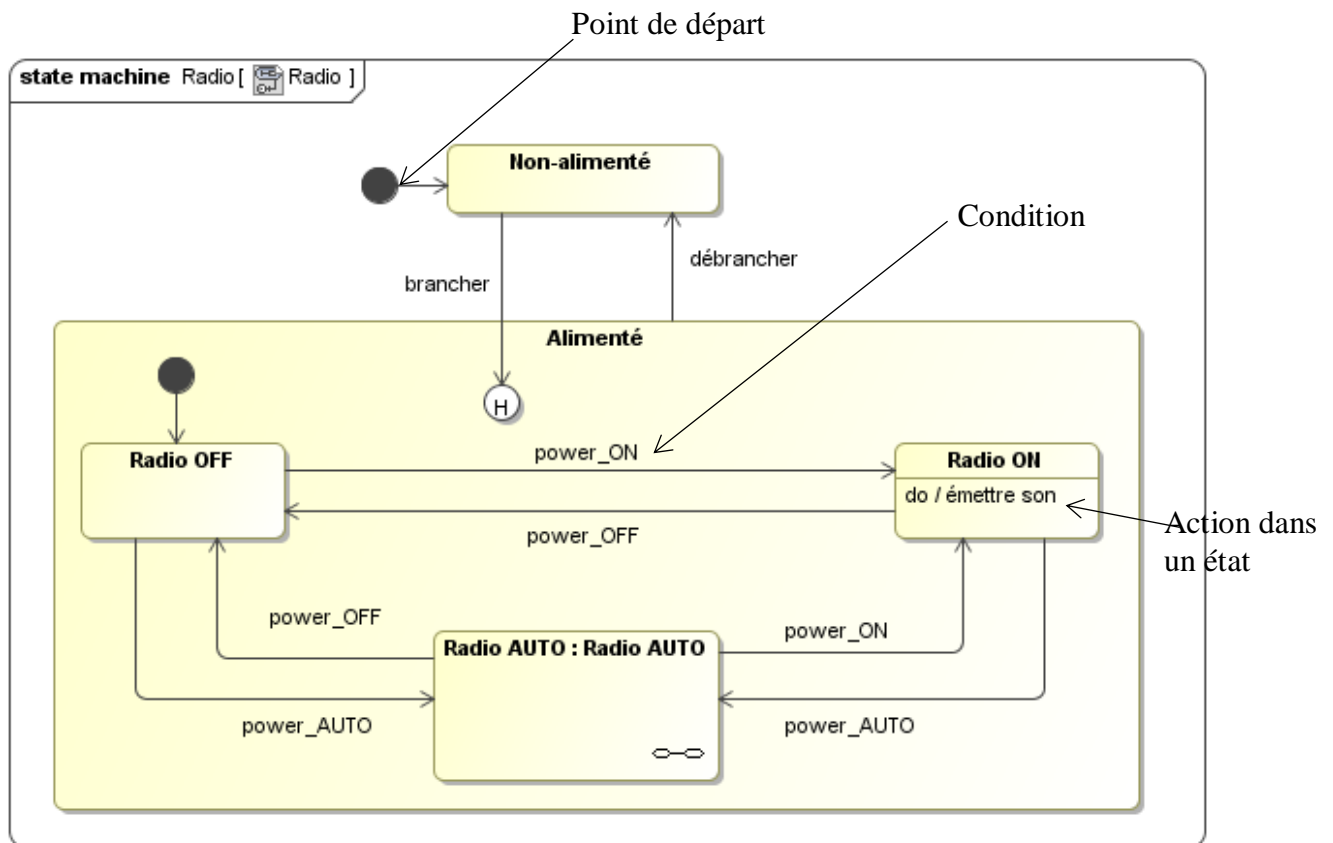
Parfois, il est nécessaire dans le cas de plusieurs GRAFCET, de vouloir synchroniser ces GRAFCET entre eux. Pour cela, il est nécessaire de connaître l'état « d'avancement » d'un autre GRAFCET. Les réceptivités relatives aux étapes sont notées X_i , où i est le numéro de l'étape considéré. La variable X_i vaut 1 si l'étape i est active, sinon, elle vaut 0. Exemple :

IX . DESCRIPTION PAR L'OUTIL GRAPHE D'ETATS

Le graphe d'états, appelé aussi diagramme états-transitions, est un des outils permettant de décrire le comportement séquentiel d'un système.

I. Constitution de l'outil graphe d'états

Les différents constituants d'un graphe d'états sont indiqués sur l'exemple ci-dessous :



Remarque :

Les états peuvent être représentés graphiquement soit par des rectangles, soit par des ovales. Lorsqu'un état est composé de sous-états, on parle de super états ou d'état composite.


II. Descriptif des constituants d'un graphe d'états




Etat OU : Les états OU représentent des états de fonctionnement mutuellement exclusif. Aucun des 2 états OU (Ou exclusif) ne peut être actif ou exécuter en même temps.



Etat ET : Les états ET représentent des états de fonctionnement totalement indépendant. 2 ou plusieurs états de même niveau hiérarchique peuvent être actifs simultanément. Ces états sont représentés graphiquement par un rectangle en trait pointillé, et un numéro indique l'ordre d'exécution.

 **Transitions :** Les transitions sont représentées par des flèches orientées, et permettent de décrire les évolutions du système d'un état source vers un état destination.

 **Transition par défaut :** Cette transition indique l'état (ou super-état) qui doit être actif à l'état initial. Elle ne peut apparaître que sur des états OU.

Nom_etat
entry: actions;
during: actions;
exit: actions;

Actions dans un état : Il s'agit de définir les actions à effectuer lorsque l'état est actif.

On définit 3 types d'actions :

- **entry: actions** Action à l'activation de l'état.
- **during: actions** Action durant l'état :
- **exit: actions** Action à la désactivation de l'état.

Evènement et condition associés à une transition: Il s'agit de définir la condition pour passer d'un état source à un état destination. Nous ne traiterons que le cas des conditions. La condition est une équation logique qui est binaire (booléenne, vraie ou fausse).

La syntaxe utilisée dans le logiciel Matlab pour une condition est :

- Écriture entre crochet de la condition [*condition*] ;
- Écriture sans crochet pour les évènements (pas utilisé en SSI) ;
- Le ET logique s'écrit &, le OU logique s'écrit ||, la fonction NON s'écrit ! (exemple : [BPMa & (!BPAr || !BPAu)].

III. Évolution d'un graphe d'états

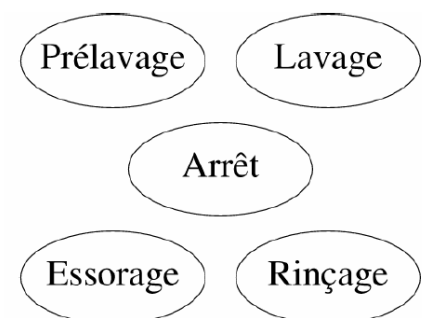
Dans un graphe d'états, la loi d'évolution des états n'est évidemment pas aléatoire.

Cette loi est soigneusement choisie par le concepteur du système afin que celui-ci remplisse la fonction souhaitée. Le graphe d'états représente graphiquement l'évolution séquentielle des états d'un système.

Exemple : Le lave-linge

Un cycle complet d'un lave-linge est composé de 5 états :

- Prélavage ;
- Lavage ;
- Rinçage ;
- Essorage ;
- Arrêt.



On complète ensuite le graphe d'états en plaçant les transitions entre les états (flèche orientée).

On précise ensuite les conditions (ou évènements) permettant le passage d'un état à un autre.

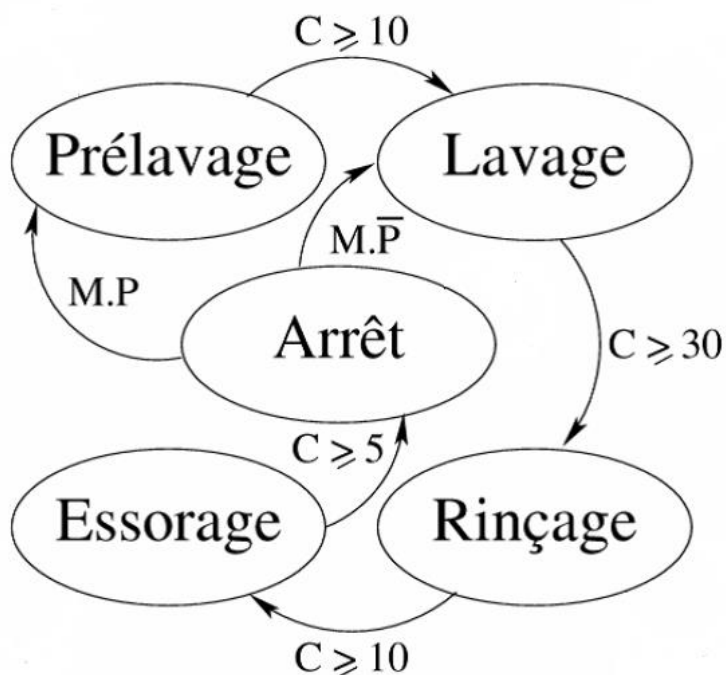
Soit les entrées suivantes :

- **M** : bouton poussoir « Marche » ;
- **P** : « Prélavage » sélectionné ;
- **C** : valeur courante (en minutes) d'un compteur permettant de mesurer la durée d'un état et qui est remis à zéro automatiquement au début de chaque état.

Les durées des différentes étapes de lavage (états) sont fixées par le constructeur :

- prélavage : 10 minutes ;
- lavage : 30 minutes ;
- rinçage : 10 minutes ;
- essorage : 5 minutes.

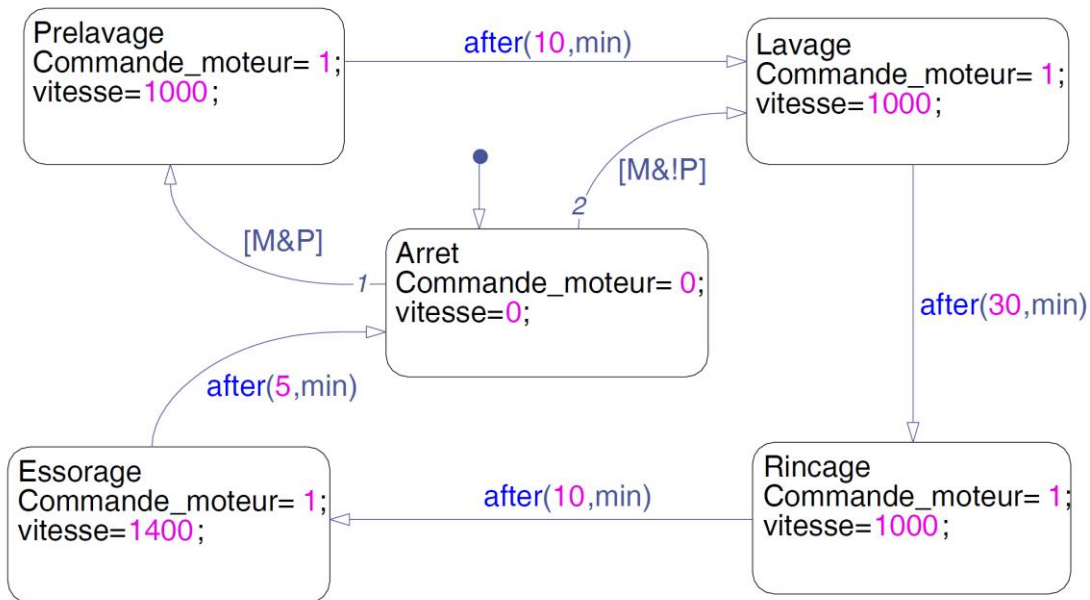
À partir de ces informations, nous pouvons compléter les conditions logiques associées à chaque transition.



Il reste ensuite à définir les variables de sorties :

- Commande_moteur : égale à 1 si le moteur doit tourner, sinon 0 ;
- Vitesse : égale à
 - 0 à l'arrêt ;
 - 1 000 tr/min en prélavage, en lavage et en rinçage ;
 - 1 400 tr/min en essorage.

Le graphe d'états complet du lave-linge peut donc se représenter sous la forme :



Exercice : Compléter ce diagramme d'état afin de pouvoir interrompre le cycle du lave-linge quel que soit l'état de celui-ci. La variable associée à Arrêt est la variable binaire A.

IV. Règles d'évolution d'un graphe d'états

Lorsque dans un graphe d'états, seuls des états OU sont présents, il est nécessaire de vérifier :

- Qu'il y a toujours au moins un état destination (Condition 1) ;
- Qu'il n'existe qu'un seul état destination (Condition 2).

Condition 1 : Il doit être complet ou non ambigu.

Ceci signifie que le comportement est toujours défini : à chaque évolutions des variables d'entrées (conditions ou évènements), et quel que soit l'état dans lequel se trouve le graphe d'états, on doit pouvoir connaître l'état suivant.

Condition 2 : Il doit être non contradictoire.

La deuxième règle signifie qu'à tout changement des variables d'entrées (condition et évènement), une seule transition est possible. Si plus d'une transition à sa condition associée vraie, le graphe est contradictoire (par exemple 2 actions incompatibles sont simultanément possibles).

X. DESCRIPTION PAR L'OUTIL ALGORITHME (OU ALGORIGRAMME)

I. Définitions

Algorithme : C'est l'ensemble de règles opératoires ordonnant à un processeur d'exécuter dans un ordre déterminé un nombre d'opérations élémentaires.

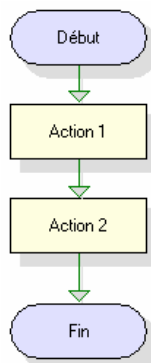
Algorigramme : C'est une représentation graphique de l'algorithme utilisant des symboles normalisés.

II. Constitution de l'outil Algorigramme

SYMBOLE	DÉSIGNATION	SYMBOLE	DÉSIGNATION
	début ou fin d'un algorithme		Test ou Branchement conditionnel décision d'un choix parmi d'autres en fonction des conditions
	symbole général de « traitement » opération sur des données, instructions, ... ou opération pour laquelle il n'existe aucun symbole normalisé		sous-programme appel d'un sous-programme
	entrée / sortie		Liaison Les différents symboles sont reliés entre eux par des lignes de liaison. Le cheminement va de haut en bas et de gauche à droite. Un cheminement différent est indiqué à l'aide d'une flèche
	commentaire		

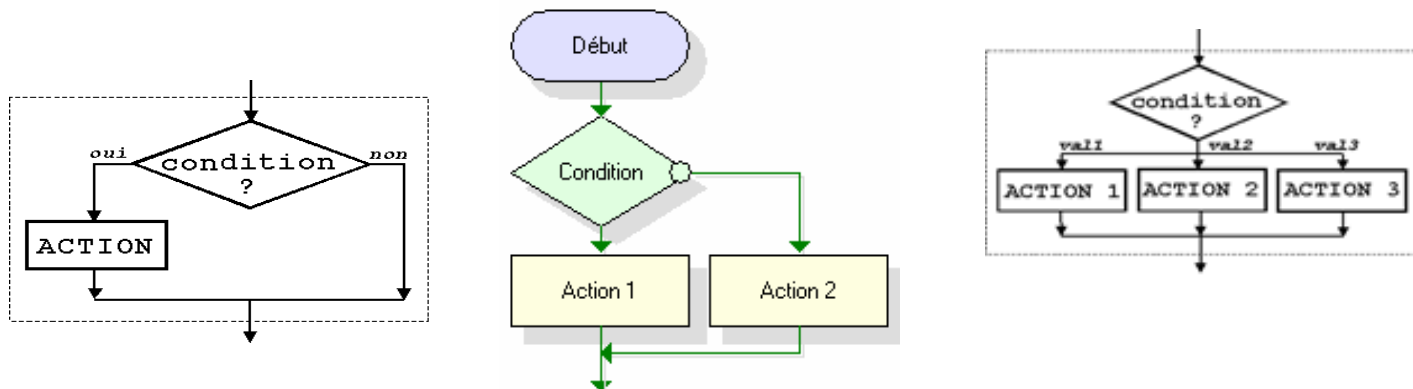
III. Structures de base d'un algorithme - algorithme

1. Structure linéaire (appelée aussi structure unique)



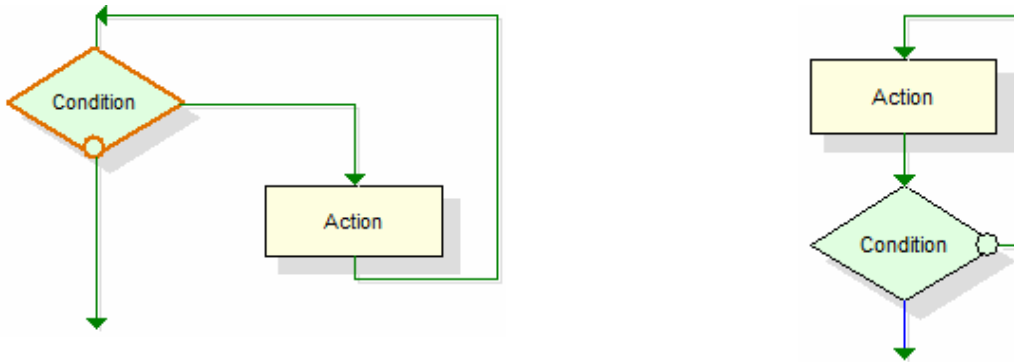
On exécute successivement une suite d'action dans l'ordre de leur énoncé.

2. Structure alternative



On fait un test (condition) et on exécute l'action 1, 2 ou 3 suivant la réponse à ce test.

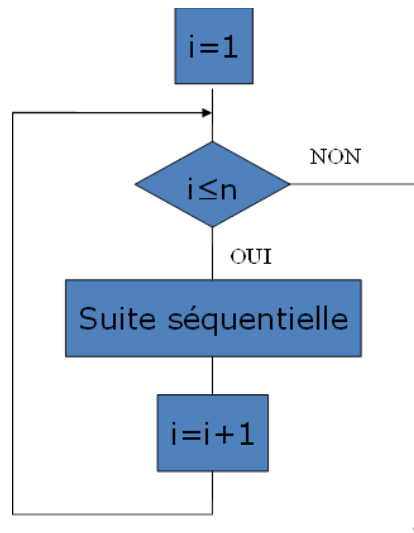
3. Structures itératives



La structure **Tant que Faire** est très utilisée, notamment lorsque l'on veut faire une boucle infinie. On écrit alors :

Tant que 1
Faire Suite d'actions
FinTantQue

4. Structure à reprise de séquence



Numération et codage de l'information

Nous avons vu dans les parties précédentes que nous sommes capables de traiter aisément le langage binaire, or les systèmes qui nous entourent ne le sont pas forcément, d'où la nécessité d'apprendre à passer d'un système de numération à un autre.

XI . LES SYSTEMES DE NUMERATION

I. Numération décimale :

Ce système de numération, utilisé dans la vie quotidienne, dispose de dix symboles différents qui sont : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9. On parle de base 10.

Un nombre entier positif N écrit en base 10, s'écrit toujours :

$$(N)_{10} = a_n * 10^n + a_{n-1} * 10^{n-1} + \dots + a_2 * 10^2 + a_1 * 10^1 + a_0 * 10^0$$

Remarque : pour spécifier la base utilisée, généralement on place le nombre entier positif entre parenthèse, suivi en indice de la base utilisée. Pour la base 10, généralement, on oublie de spécifier la base, car il s'agit de la base utilisée couramment.

Exemple : $(7239)_{10} = 7239 = 7 \cdot 10^3 + 2 \cdot 10^2 + 3 \cdot 10^1 + 9 \cdot 10^0$

Vocabulaire :

- Les digits correspondent aux coefficients a_n . Ils ne peuvent prendre que des valeurs de la base.
- Les puissances de 10 sont appelées les poids.
- Le poids est égal à la base élevée à la puissance de son rang.

	Unité	Dizaine	Centaine	Milliers	10 Milliers	100 Milliers
Digit	a_0	a_1	a_2	a_3	a_4	a_5
Rang	0	1	2	3	4	5
Poids	10^0	10^1	10^2	10^3	10^4	10^5

Pour faciliter l'écriture, nous utilisons la représentation suivante :

$$(N)_{10} = N = (a_n, a_{n-1}, \dots, a_1, a_0)_{10} = (a_n, a_{n-1}, \dots, a_1, a_0)$$

II. Numération binaire :

La numération binaire ou en base 2 utilise deux symboles (0 et 1). Cette base est très commode pour distinguer les 2 états logiques fondamentaux (lampe allumée ou éteinte, présence ou non d'une pièce, couleur noir ou blanc, ...).

On écrit : $(a_n a_{n-1} a_{n-2} \dots a_1 a_0)_2 = a_n \cdot 2^n + a_{n-1} \cdot 2^{n-1} + a_{n-2} \cdot 2^{n-2} + \dots + a_1 \cdot 2^1 + a_0 \cdot 2^0$

En numération binaire, les digits sont appelés bit (abréviation de Binary digIT).

Exemple : $(5)_{10} = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 = (101)_2$

Un code binaire à n bits (en base 2) distingue 2^n états ou combinaisons.

n	0	1	2	3	4	5	6	7	8
2 ⁿ	1	2	4	8	16	32	64	128	256

Les puissances successives de 2 (1, 2, 4, 8, 16, 32, ...) sont appelées poids binaires.

Vocabulaire :

Le bit de poids le plus fort (a_n) est appelé MSB (Most Significant Bit).

Le bit de poids le plus faible (a_0) est appelé LSB (Low Significant Bit).

Un regroupement successif de 4 bits s'appelle un quartet.

Un regroupement successif de 8 bits s'appelle un octet.

Un regroupement successif de k bits ($k > 8$) s'appelle un mot de k bits.

III. Numération hexadécimale :

Ce système de numération est très utilisé dans les systèmes ordinateurs et micro-ordinateurs ainsi que dans le domaine des transmissions de données.

Il comporte 16 symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F.

$$(N)_{16} = a_{n-1}.16^{n-1} + a_{n-2}.16^{n-2} + \dots + a_1.16^1 + a_0.16^0$$

Pour indiquer la base 16, on peut la noter en indice suivant la manière générale, mais dans la pratique, on peut aussi utiliser le caractère \$ (dollar) devant le nombre, ou la lettre H derrière ou alors 16# devant le nombre.

Exemple : $(A8)H = 16\#A8 = \$A8 = A.16^1 + 8.16^0 = 10.16 + 8.1 = (168)_{10}$

XII . CHANGEMENT DE BASE - CONVERSION

I. Conversion directe :

Du binaire vers l'hexadécimal :

Un nombre hexadécimal est « découparable » en quartets facilement codables en binaire. Donc, pour convertir du binaire vers l'hexadécimal, on divise le nombre binaire en quartet, en partant de la droite. Chacun des paquets est ensuite converti en hexadécimal.

Exemple : $(110101110001)_2 = (1101 \ 0111 \ 0001)_2 = 16\#D71$

De l'hexadécimal vers le binaire :

C'est le processus directement inverse, on écrit chaque quartet sur 4 bits en complétant éventuellement avec des zéros.

Exemple : $BC34H = (1011 \ 1100 \ 0011 \ 0100)_2$

II. Conversion indirecte :

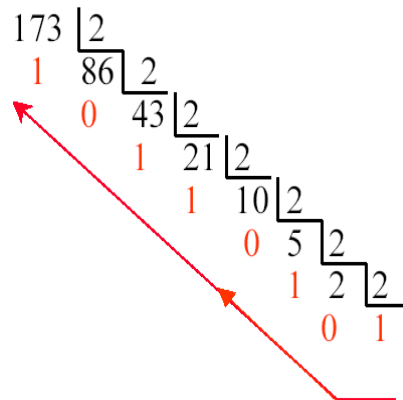
Un nombre entier positif **N** étant donné en base 10, cherchons à l'écrire dans une autre base notée **b**.

La méthode consiste à diviser le nombre décimal à convertir par la base **b**, et nous conservons le reste (division entière). Le quotient obtenu est ainsi successivement divisé tant qu'il n'est

pas nul. Les restes successifs sont écrits, en commençant par le dernier, de la gauche vers la droite, pour former l'expression de **N** dans le système de base **b**.

Du décimal vers le binaire :

Exemple : Convertir 173 en base 2



Le résultat est donc **(10101101)₂**

Du décimal vers l'hexadécimal :

On reprend la même méthode mais en divisant par 16.

XIII . CODAGE DES NOMBRES

I. Codages pondérés :

Les codes pondérés sont des codes où à chaque bit est associé un poids.

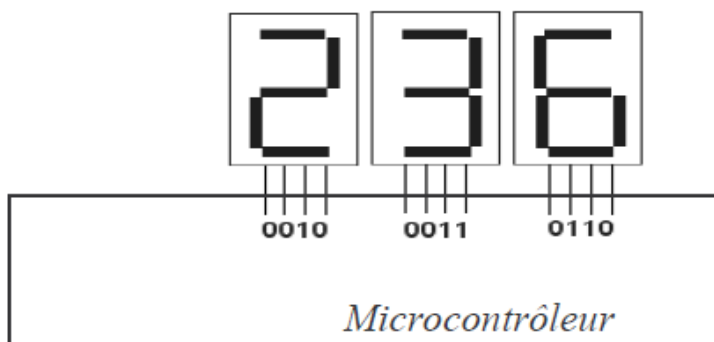
Les codes naturels :

Les codes binaires, décimaux et hexadécimaux répondent aux règles classiques de l'arithmétique des codes pondérés. Ce sont donc des codes pondérés.

Le code binaire codé décimal (BCD) :

Ce codage est destiné à l'affichage de valeurs décimale, chaque digit (unités, dizaines, centaines, ...) doit être codé en binaire sur 4 bits. Il ne permet aucun calcul, il est uniquement destiné à la saisie et à l'affichage de données.

Exemple :



Le code Complément à 1 :

Le code Complément à 1 appelé aussi Complément Restreint (CR) d'un nombre est obtenu en complémentant chaque bit un à un.

Exemple : $CR(22_{10}) = CR(10110_2) = (01001)$

Le code Complément à 2 : Représentation des nombres négatifs

Le code Complément à 2 appelé aussi complément vrai (CV) permet de représenter les nombres entiers négatifs utilisés dans les calculateurs.

Il est obtenu par $-x = CV(x) = CR(x) + 1$. Le signe est le bit de poids fort (MSB). Par convention, 0 pour une valeur positive et 1 pour une valeur négative.

Exemple :

$$-22 = CV(22_{10}) = CV(010110_2) = CR(010110_2) + 1 = (101001) + 1 = 101010$$

II. Codages non pondérés :

Les codes non pondérés correspondent aux codes où chaque digit n'a pas de poids.

Le code binaire réfléchi ou code Gray :

Dans ce code, un seul bit change entre 2 valeurs adjacentes. Le bit changeant est celui le plus à droite ne provoquant pas une combinaison déjà apparue. Il est employé dès que l'on doit représenter une évolution réelle des variables où une seule change d'état à un instant.

Exemple d'utilisation du code binaire réfléchi : Capteur de position angulaire absolu.

Le code p parmi n :

Dans ce code, à chaque nombre correspondent n bits, dont p valent 1 et n-p valent 0. Il permet de détecter jusqu'à une erreur : si lors d'une communication, il y a réception d'un nombre de 1 différent de p, cela signifie qu'il y a eu une erreur de transmission.

Le nombre de combinaisons est défini par : $C_n^p = \frac{n!}{p! \cdot (n-p)!}$

N	2 parmi 5
0	1 1 0 0 0
1	0 0 0 1 1
2	0 0 1 0 1
3	0 0 1 1 0
4	0 1 0 0 1
5	0 1 0 1 0
6	0 1 1 0 0
7	1 0 0 0 1
8	1 0 0 1 0
9	1 0 1 0 0

Exemple : Le code 2 parmi 5

Avec le code 2 parmi 5, il y a

$$C_5^2 = \frac{5!}{2! \cdot (5-2)!} = \frac{5 \cdot 4 \cdot 3 \cdot 2 \cdot 1}{2 \cdot 1 \cdot (3 \cdot 2 \cdot 1)} = \frac{120}{12} = 10 \text{ combinaisons.}$$

Exemple d'utilisation : le codage 2 parmi 5 entrelacé pour les codes-barres sur les enveloppes.

XIV . EXTENSION AUX CODES NON NUMERIQUES

Pour manipuler d'autres éléments que des nombres, il est aussi nécessaire de les coder. Le plus connu de ces codes, et le plus utilisé en particulier dans le monde informatique, est le code ASCII (American Standard Code for Information Interchange) présenté ci-dessous.

ASCII	Caract.	Signification	ASCII	Caract.	Signification
00	NUL	<i>null</i> , nul	16	DLE	<i>data link escape</i> , échap. liaison données
01	SOH	<i>start of heading</i> , début d'en-tête	17	DC1	<i>device control 1</i> , commande unité 1
02	STX	<i>start of text</i> , début de texte	18	DC2	<i>device control 2</i> , commande unité 2
03	ETX	<i>end of text</i> , fin de texte	19	DC3	<i>device control 3</i> , commande unité 3
04	EOT	<i>end of transmission</i> , fin de transmission	20	DC4	<i>device control 4</i> , commande unité 4
05	ENQ	<i>enquiry</i> , interrogation	21	NAK	<i>negative acknowledge</i> , acc. récep. nég.
06	ACK	<i>acknowledge</i> , accusé de réception	22	SYN	<i>synchronous idle</i> , inactif synchronisé
07	BEL	<i>bell</i> , sonnerie	23	ETB	<i>end of transmission block</i> , fin tran. bloc
08	BS	<i>backspace</i> , espacement arrière	24	CAN	<i>cancel</i> , annuler
09	HT	<i>horizontal tabulation</i> , tabulation horiz.	25	EM	<i>end of medium</i> , fin du support
10	LF	<i>line feed</i> , saut de ligne	26	SUB	<i>substitute</i> , substitut
11	VT	<i>vertical tabulation</i> , tabulation verticale	27	ESC	<i>escape</i> , échappement
12	FF	<i>form feed</i> , saut de page	28	FS	<i>file separator</i> , séparateur de fichiers
13	CR	<i>carriage return</i> , retour chariot	29	GS	<i>group separator</i> , séparateur de groupes
14	SO	<i>shift out</i> , hors code	30	RS	<i>record separator</i> , sép. d'enregistr.
15	SI	<i>shift in</i> , en code	31	US	<i>unit separator</i> , séparateur d'unités

Ces codes ne sont quasiment plus utilisés à l'exception de SOH, STX, ETX, EOT, LF et CR.

Table des caractères imprimables (32 à 127) — ou table ASCII standard

ASCII	Caractère.
32	SP (<i>space</i> , espace)
33	!
34	"
35	#
36	\$
37	%
38	&
39	'
40	(
41)
42	*
43	+
44	,
45	-
46	.
47	/
48	0
49	1
50	2
51	3
52	4
53	5
54	6
55	7
56	8
57	9
58	:
59	;
60	<
61	=
62	>
63	?

ASCII	Caractère
64	@
65	A
66	B
67	C
68	D
69	E
70	F
71	G
72	H
73	I
74	J
75	K
76	L
77	M
78	N
79	O
80	P
81	Q
82	R
83	S
84	T
85	U
86	V
87	W
88	X
89	Y
90	Z
91	[
92	\
93]
94	^
95	_

ASCII	Caractère
96	`
97	a
98	b
99	c
100	d
101	e
102	f
103	g
104	h
105	i
106	j
107	k
108	l
109	m
110	n
111	o
112	p
113	q
114	r
115	s
116	t
117	u
118	v
119	w
120	x
121	y
122	z
123	{
124	
125	}
126	~
127	DEL (<i>delete</i> , sup.)

Ces codes alphanumériques sont accessibles sur un ordinateur fixe avec un clavier ou depuis un ordinateur portable disposant d'un pavé numérique.

Pour cela, ouvrir un éditeur de texte, et appuyer sur la touche Alt, et saisir un nombre compris entre 32 et 127 sur le clavier numérique. Les caractères associés aux codes ASCII apparaîtront.