UNITE ATC

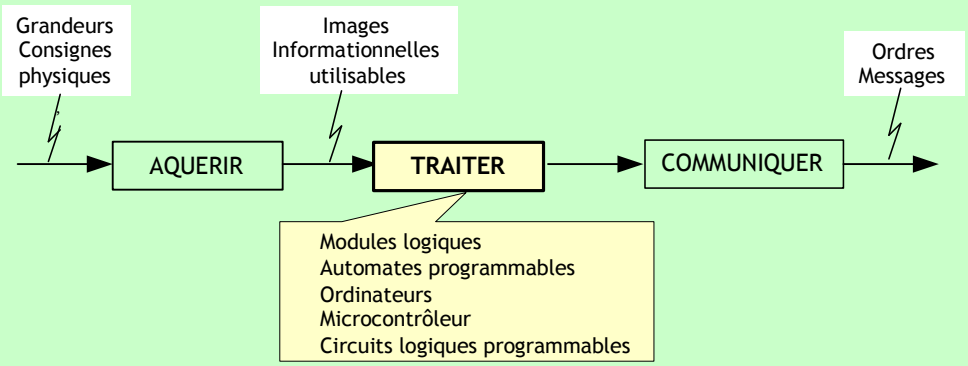
FONCTION

T R A I T E R

PRESENTATION

Dans un système automatisé, le traitement des informations concernant principalement l'état de la PO (capteurs, consignes utilisateur, etc.) nécessitent des organes de commande dotée d'une certaine intelligence relativement élevée, allant du simple circuit logique combinatoire jusqu'au microordinateur sophistiqué.

La position de la fonction "Traiter" dans une chaîne d'information, ainsi que les différentes réalisations principales sont représentées par la figure suivante.



COMPETENCES ATTENDUES

A partir de tout ou partie d'un produit support avec son cahier des charges et son dossier technique :

- Exprimer les entrées/sorties d'une unité de traitement des données acquises
- Mettre en œuvre une unité de traitement de l'information

CHAPITRES INCLUS DANS LA FONCTION TRAITER

- Représentation et codage de l'information binaire
- Fonctions combinatoires de bases
- Simplification des fonctions logiques
- Fonctions combinatoires avancées
- Fonctions séquentielles
- Familles logiques TTL et CMOS
- Circuits intégrés de temporisation
- Circuits logiques programmables
- GRAFCET
- Les systèmes programmables : Automate programmable industriel

CHAPITRE 1

REPRESENTATION ET CODAGE DE L'INFORMATION BINAIRE

INTRODUCTION :

La numération arabe est universellement adoptée, étant donné sa capacité à faire les calculs. Il s'agit du système de numération avec la base 10. Comme on le sait, dans cette base familière :

- On utilise les 10 symboles, appelés chiffres, de l'ensemble : {0, 1, 2, 3, 4, 5, 6, 7, 8, 9} ;
- Un nombre quelconque peut s'écrire en utilisant les puissances de 10 ;

  Exemple : 571 = 5 x 102 + 7 x 101 + 1 x 100 .

Mais la représentation des nombres avec le système décimal (base 10) n'est pas la seule utilisée. On peut donc en utiliser d'autres, en particulier le système binaire (base 2). Les circuits logiques ne connaissent que les valeurs 0 et 1 ; alors on peut faire des calculs et des traitements comme on le fait avec le système décimal. Ceci permet de rendre le traitement de l'information automatique et rapide.

1. LES SYSTEMES DE NUMERATION :

1.1. Principe :

D'une façon générale, soit une base B, donc associée à B symboles : { S_0, S_1, …, S_{B-1}} ; un nombre N, a les caractéristiques suivantes :

- Il s'écrit N = A_{n-1} … A_i … A_1A_0 avec $A_i \in \{ S_0, S_1, …, S_{B-1}\}$;
- Il a pour valeur N = $A_{n-1}.B^{n-1} + … + A_iB^i + … + A_1B^1 + A_0B^0$ (forme polymoniale) ;
- Ai est le chiffre (digit) de rang i et de poids Bi ;
- An est le chiffre le plus significatif (MSD : Most Significant Digit) ;
- A0 est le chiffre le moins significatif (LSD : Less Significant Digit).

On va étudier les bases 2 et 16 pour leur intérêt dans les circuits logiques. La référence à la base 10 est d'un usage pratique, on étudiera alors la conversion de la base 2 vers la base 10 et vise versa.

1.2. Système binaire (base 2) :

L'homme connaît la base 10 ; il fait alors ses calculs dans cette base. Alors, puisque les systèmes numériques ne reconnaissent que 2 états 0 ou 1, ils seront très aptes de faire les calculs dans la base 2. La base 2 a les caractéristiques suivantes :

- Un nombre N s'écrit : N = A_nA_{n-1} … A_i … A_1A_0 avec $A_i \in \{0, 1\}$; chaque chiffre est appelé couramment bit, contraction de binary digit (chiffre binaire) ; Ce nombre N est couramment désigné aussi par "Mot de n bits" ;
- A_n est le chiffre le plus significatif, couramment appelé MSB (Most Significant Bit) ;
- A_0 est le chiffre le moins significatif, couramment appelé LSB (Less Significant Bit).
- Ce nombre a pour valeur N = $A_{n-1}2^{n-1} + … + A_i2^i + … + A_12^1 + A_02^0$ (forme polymoniale);
  Exemple : N = 110101 ; il a pour valeur N = $1x2^5 + 1x2^4 + 0x2^3 + 1x2^2 + 0x2^1 + 1x2^0$.

1.3. Conversion de la base 2 vers la base 10 :

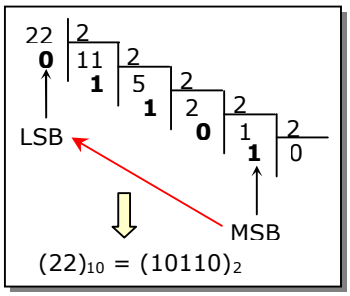
On exploite directement la forme polynomiale.

Exemple pour la base 2 : $(1011)_2 = 1x2^3 + 0x2^2 + 1x2^1 + 1x2^0 = 8 + 0 + 2 + 1 = 8 + 4 + 2 = (11)_{10}$.

1.4. Conversion de la base 10 vers la base 2 :

La méthode de division est la plus utilisée ; elle consiste en des divisions successives du nombre $(N)_{10}$ par 2, jusqu'à obtenir un quotient nul. Les restes des divisions successives, écrits dans l'ordre inverse, constituent le nombre N dans la base 2 $(N)_2$.

Exemple : $(22)_{10} = (?)_2$.



2. CODAGE DE L'INFORMATION BINAIRE :

Un système électronique traite les informations en binaire. Or, ces informations à traiter sont de différentes natures. Par exemple, en traitement de texte, on manipule des caractères ; pour qu'un ordinateur traite ces caractères, il faut associer alors à chaque caractère un nombre binaire. Cette association s'appelle "Codage" de l'information binaire et permet d'utiliser plusieurs codes suivant le domaine d'application. L'opération inverse s'appelle "Décodage" ou "Transcodage". On étudie en particulier : Le code binaire pur, le code GRAY, le code BCD et le code ASCII.

2.1. Le code binaire pur :

Il est aussi appelé code binaire naturel. C'est le code binaire sans aucune codification, c'est à dire qui découle directement du principe général de la numération. C'est le code naturel utilisé dans les systèmes numériques (ordinateur, etc.). Le tableau suivant donne le code binaire pur pour un exemple d'un mot de 4 bits $(A_3\ A_2\ A_1\ A_0)$:

Valeur décimale	Code binaire			
	A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1
10	1	0	1	0
11	1	0	1	1
12	1	1	0	0
13	1	1	0	1
14	1	1	1	0
15	1	1	1	1

Pour remplir rapidement une table de vérité avec toutes les combinaisons possibles des variables d'entrée, on procède comme en décimal :

- On part du poids faible (A_0), qui balaye la plage 0 à 1 ;
- On passe au poids suivant (A_1), qui reste à 0 pour la plage 0 à 1 de A_0, puis à 1 pour la même plage ;
- Et ainsi de suite.

A_2	A_1	A_0
0	0	0
0	0	1
0	1	0
0	1	1

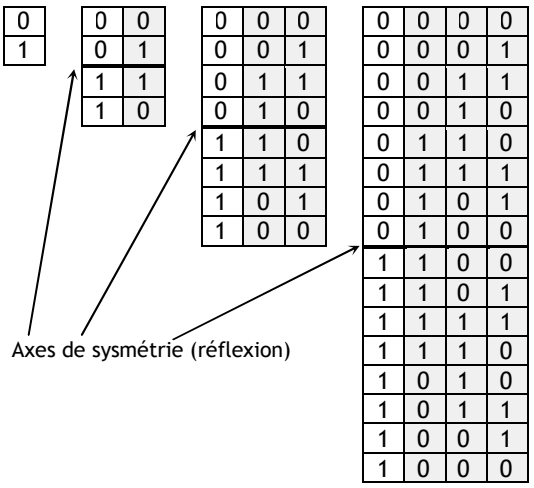
2.2. Le code GRAY :

Dans les systèmes industriels où on a besoin de mesurer un déplacement linéaire ou angulaire, on utilise le "code GRAY". La raison de ce choix est que si le système qui mesure le déplacement (capteur) utilise le code binaire pur, le déplacement d'une position à une autre voisine génère des combinaisons intermédiaire fausses, car plusieurs bits varient en même temps.

Pour remédier à ce problème, il suffit de coder chaque position de façon que les valeurs de positions successives ne différent que d'un seul bit. C'est pour cela qu'on l'appelle "code à distance unité". On l'appelle aussi "code binaire réfléchi" parce que pour le construire, on procède par réflexion comme l'indique le tableau suivant avec 4 bits :

- on a 16 combinaisons différentes ;
- dans le passage d'une combinaison à une autre, il n'y a qu'un bit qui change.

Valeur décimale	Code binaire				Code GRAY			
	A_3	A_2	A_1	A_0	G_3	G_2	G_1	G_0
0	0	0	0	0	0	0	0	0
1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	0	1	1
3	0	0	1	1	0	0	1	0
4	0	1	0	0	0	1	1	0
5	0	1	0	1	0	1	1	1
6	0	1	1	0	0	1	0	1
7	0	1	1	1	0	1	0	0
8	1	0	0	0	1	1	0	0
9	1	0	0	1	1	1	0	1
10	1	0	1	0	1	1	1	1
11	1	0	1	1	1	1	1	0
12	1	1	0	0	1	0	1	0
13	1	1	0	1	1	0	1	1
14	1	1	1	0	1	0	0	1
15	1	1	1	1	1	0	0	0



2.3. Le code ASCII :

Le code ASCII (American Standard Code for Information Interchange) est un code qui représente les caractères éditables ou non éditables : éditables parce que l'on peut les éditer comme la caractère "A" et non éditables comme le cratère "Escape" ou "Return". Il est codé sur 7 bits (b6 b5 b4 b3 b2 b1 b0), ce qui permet de représenter 128 (2^7) caractères différents. La table suivante montre un tel codage. Par exemple, Le code de la lettre "A" (majuscule) est :

- en binaire : b6 b5 b4 b3 b2 b1 b0 = 1000001 ;
- en hexadécimal 41 ;
- en décimal 65.

Binaire				b6	0	0	0	0	1	1	1	1
				b5	0	0	1	1	0	0	1	1
				b4	0	1	0	1	0	1	0	1
				Hexadécimal	0	1	2	3	4	5	6	7
b3	b2	b1	b0	Décimal	0	16	32	48	64	80	96	112
0	0	0	0	0	+0	NUL	TC7 (DEL)	SP	0	@	P	`	p
0	0	0	1	1	+1	TC1 (SOH)	DC1	!	1	A	Q	a	q
0	0	1	0	2	+2	TC2 (STX)	DC2	"	2	B	R	b	r
0	0	1	1	3	+3	TC3 (ETX)	DC3	#	3	C	S	c	s
0	1	0	0	4	+4	TC4 (EOT)	DC4	$	4	D	T	d	t
0	1	0	1	5	+5	TC5 (ENQ)	TC8 (NAK)	%	5	E	U	e	u
0	1	1	0	6	+6	TC6 (ACK)	TC9 (SYN)	&	6	F	V	f	v
0	1	1	1	7	+7	BEL	TC 10 (ETB)	'	7	G	W	g	w
1	0	0	0	8	+8	FE0 (BS)	CAN	(	8	H	X	h	x
1	0	0	1	9	+9	FE1 (HT)	EM	)	9	I	Y	i	y
1	0	1	0	A	+10	FE2 (LF)	SUB	*	:	J	Z	j	z
1	0	1	1	B	+11	FE3 (VT)	ESC	+	;	K	[	k	é
1	1	0	0	C	+12	FE4 (FF)	IS4 (FS)	,	<	L	\	l	ù
1	1	0	1	D	+13	FE5 (CR)	IS3 (GS)	-	=	M	]	m	è
1	1	1	0	E	+14	SO	IS2 (RS)	.	>	N	^	n	¨
1	1	1	1	F	+15	SI	IS1 (US)	/	?	O	_	o	DEL

2.4. Le code BCD :

Le code BCD (Binary Coded Decimal) qui veut dire **B**inaire **C**odé en **D**écimal est la traduction en binaire des 9 premiers chiffres du systèmes décimal.

Si on a un nombre décimal N à m chiffres, il sera codé en BCD sur (m x 4) bits : chaque chiffre décimal est traduit en code BCD sur 4 bits.

Exemple : $(571)_2$ = 1000111011 en binaire pur
= 0101 0111 0001 en BCD
5 7 1

Valeur décimale	Code BCD			
	A_3	A_2	A_1	A_0
0	0	0	0	0
1	0	0	0	1
2	0	0	1	0
3	0	0	1	1
4	0	1	0	0
5	0	1	0	1
6	0	1	1	0
7	0	1	1	1
8	1	0	0	0
9	1	0	0	1

3. NOTIONS D'ARITHMETIQUE BINAIRE :

Comme on l'a signalé, avec les circuits logiques, on peut faire des calculs arithmétiques, puisqu'on a les mêmes règles de calcul qui s'appliquent à toutes les bases. On traite alors de l'arithmétique binaire et on se limite aux cas de l'addition et de la soustraction.

3.1. Cas de l'addition et la soustraction :

	Retenue
0 + 0 = 0	0
0 + 1 = 1	0
1 + 0 = 1	0
1 + 1 = 0	1

	111
11	1011
+	+
07	0111
18	10010

	Retenue
0 - 0 = 0	0
0 - 1 = 1	1
1 - 0 = 1	0
1 - 1 = 0	0

19	11011
-	-
05	00111
	11
14	01110

Des circuits logiques simples réalisent ces opérations ; de plus une soustraction est facilement ramenée à une addition grâce à une représentation adéquate des nombres

3.2. Représentation des nombres :

Dans les calculs, on manipule des nombres positifs et négatifs ; il faut alors coder le signe algébrique. Plusieurs modes de représentation sont adoptés en fonction des calculs à effectuer et les caractéristiques technologiques des systèmes de traitement.

3.2.1. Représentation par valeur absolue et signe :

Pour le bit de signe, on adopte la convention 0 (+) et 1 (-). Par exemple :

$$(+35)_{10} = 0\ 100011 \quad \text{et} \quad (-35)_{10} = 1\ 100011.$$

Cette solution a comme inconvénient la complexité de la réalisation technologique due à :

- Un traitement spécifique du signe ;
- Une double représentation du 0.

3.2.2. Représentation par complément à 2 :

Soit un nombre binaire A sur n bits et son complément $\overline{A}$ (nommé aussi complément à 1 de A), on a :

$$A + \overline{A} = 2^n - 1$$

$$\begin{array}{r} A_{n-1}\ A_{n-2} \ldots\ A_1\ A_0 \\ + \\ \overline{A_{n-1}}\ \overline{A_{n-2}} \ldots \overline{A_1}\ \overline{A_0} \\ \hline 1\ \underbrace{1\ \ldots\ 1}_{(2^n - 1)}\ 1 \end{array}$$

On déduit de cette relation que : $-A = \overline{A} + 1 - 2n$.

Comme le (2n) ne rentre pas dans le format défini (An-1 An-2… A1 A0), il sera ignoré. On a alors ($-A = \overline{A} + 1$) :

Le terme ($\overline{A} + 1$) s' identifie donc à l'opposé arithmétique de A

Le terme ($\overline{A} + 1$) est appelé complément à 2.

Exemple : Pour n = 4, on obtient :

$(A)_{10}$	A	$\overline{A}$	$(\overline{A}+1)$	$(-A)_{10}$
7	0111	1000	1001	-7
6	0110	1001	1010	-6
5	0101	1010	1011	-5
4	0100	1011	1100	-4
3	0011	1100	1101	-3
2	0010	1101	1110	-2
1	0001	1110	1111	-1
0	0000	1111	0000	0

On remarque que :

- Le MSB représente le signe avec 0 (+) et 1 (-).
- Le zéro n'a qu'une seule représentation ;
- Alors pour effectuer une soustraction, il suffit de faire une addition avec le complément à 2. Le résultat se lit directement en complément à 2 :
  - Si le signe est + la lecture est directe ;
  - Si le signe est - , on convertit le résultat en recherchant le complément à 2 de celui-ci.

E X E R C I C E S R E S O L U S

EXERCICE N° 1 :

$(40)_{10} = (\ ?\ )_2$

EXERCICE N° 2 :

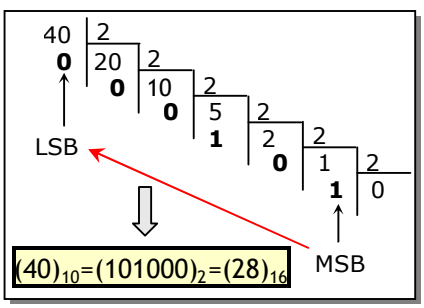
Trouver le code ASCII du chiffre 9 et de la lettre "a".

EXERCICE N° 3 :

Donner le complément à 2 du nombre $A = (13)_{10} = (1101)_2$.

CORRIGE :

EXERCICE N° 1 :



EXERCICE N° 2 :

D'après la table du code ASCII présenté dans le cours, on trouve :

→ pour "9" : $(47)_{10} = (0111001)_2$.

→ pour "a" : $(97)_{10} = (1100001)_2$. A noter qu'il est différent de "A" (majuscule).

EXERCICE N° 3 :

On a $(13)_{10} = (1101)_2$.

→ On fait le complément à 1 du nombre A, soit $\overline{A} = 0010$;
→ On ajoute 1 au résultat, $\overline{A} + 1 = 0011$.

FONCTIONS COMBINATOIRES DE BASE

INTRODUCTION :

De nombreux dispositifs ont deux états stables de fonctionnement. Par exemple, un interrupteur peut être ouvert ou fermé ; un transistor, sous certaines conditions, peut être bloqué ou saturé, etc. On convient d'affecter, par convention, à un des deux états la valeur " 0 " et "1" à l'autre état. L'algèbre de Boole est l'outil mathématique pour étudier ces dispositifs et les circuits logiques représentent l'outil technologique pour réaliser pratiquement les opérations de cette algèbre. Les circuits qu'on va étudier dans ce chapitre sont dits "**combinatoires**", car l'état de leurs sorties ne dépend que de l'état des entrées.

1. OPERATIONS BOOLEENNES ELEMENTAIRES :

Trois opérations élémentaires suffisent pour définir une algèbre de Boole :

- l'inversion : Non (Not) ;
- le produit logique : ET (AND) ;
- la somme logique : OU (OR).

1.1. Opération Inversion :

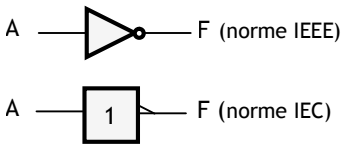
C'est une opération définie sur une seule variable. La sortie prend la valeur que n'a pas l'entrée. On dit que la sortie est l'inverse ou le complément de l'entrée.

Table de vérité

A	F
0	1
1	0

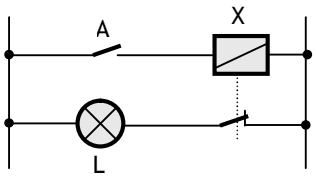
$F = \overline{A}$ (Se lit A barre)

Symbole



IEEE: Institute of Electrical and Electronics Engineers.
IEC : International Electrotechnical Commission

Illustration électrique



- L'interrupteur A ouvert (A=0) ; le relais X est non excité et le contact qui lui est associé reste fermé (position de repos) ; la lampe L est allumée (L=1) : A=0 ⇨ L=1.
- L'interrupteur A fermé (A=1) ; Le relais X est excité et le contact qui lui est associé est devenu ouvert ; La lampe L est éteinte (L=0) : A=1 ⇨ L=0 ; alors, L = Not A : $L=\overline{A}$

Propriété

- $\overline{\overline{F}} = F$

1.2. Opération ET (AND) :

C'est une opération sur 2 variables d'entrée au moins. Dans le cas simple de 2 entrées A et B, la sortie est vraie (égale à 1) si A **ET** B sont vraies aussi.

Table de vérité

A	B	F
0	0	0
0	1	0
1	0	0
1	1	1

F = A . B = AB (se lit A ET B)

Symbole

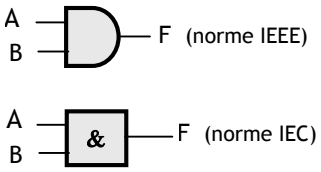
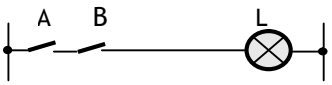


Illustration électrique



La lampe L est allumée (L=1) si l'interrupteur A **ET** l'interrupteur B sont fermés (A=B=1), soit L = A . B

Propriétés

- La fonction AND est commutative associative: F = A.B = B.A.
- La fonction AND est : F = A.(B.C) = (A.B).C = A.B.C.
- La fonction AND est généralisable pour n entrées.
- Identités remarquables : $X.0 = 0$; $X.1 = X$; $X.X = X$ et $X.\overline{X} = 0$.

1.3. Opération OU (OR) :

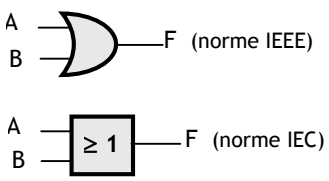
C'est une opération sur 2 variables d'entrée au moins. Dans le cas simple de 2 entrées A et B, la sortie est vraie (égale à 1) si seulement A OU B est vraie. Cette opération est dite OU inclusive, car on inclut le cas (A=B=1 ; F=1). On verra qu'il y a une autre fonction appelée OU exclusive.

Table de vérité

A	B	F
0	0	0
0	1	1
1	0	1
1	1	1

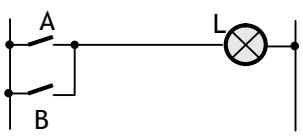
F = A + B (se lit A OU B)

Symbole



Le signe "≥" indique que la sortie est égale à 1 si le nombre des entrées à "1" est supérieur ou égal à 1 ; autrement dit, une entrée au moins égale à "1".

Illustration électrique



L est allumée (L=1) si A OU B est fermé (A=1 **OU** B=1), soit L = A + B.

Propriétés

- Comme la fonction AND, la fonction OR est commutative, associative et généralisable pour n entrées : $X+0 = X$; $X+1 = 1$; $X+X = X$ et $X+\overline{X} = 1$.
- Identités remarquables :

1.4. Propriétés et théorèmes remarquables :

Propriétés

- $A \cdot (B + C) = AB + AC$ (Distributivité du produit par rapport à la somme) ;
- $A + (B \cdot C) = (A + B) \cdot (A + C)$ (Distributivité de la somme par rapport au produit) ;
- $AB + A\overline{B} = B$: $B(A + \overline{A}) = B \cdot 1 = B$ (Factorisation) ;
- $A + AB = A$: $A(1 + B) = A \cdot 1 = A$ (Loi d'absorption)
- $A + \overline{A}B = A + B$; en effet, $A + \overline{A}B = A + \overline{A}B + AB = A + B$;

Théorème de De Morgan :

Ce théorème d'une grande utilité, permet de calculer le complément d'une expression logique quelconque (somme de produits ou produit de sommes) :

- $\overline{X + Y} = \overline{X} \cdot \overline{Y}$;
- $\overline{X \cdot Y} = \overline{X} + \overline{Y}$.

D'une façon générale, Le complément d'une expression quelconque s'obtient en complémentant les variables et en permutant les opérateurs "+" et ".".

Exemple : $F = \overline{A}BD + A\overline{D} \Rightarrow \overline{F} = \overline{\overline{A}BD + A\overline{D}} = (A + \overline{B} + \overline{D}) \cdot (\overline{A} + D)$

2. AUTRES OPERATIONS

2.1. Opération NAND :

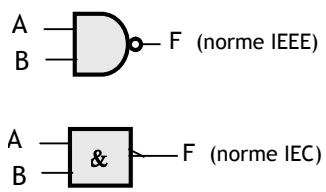
C'est le complément de l'opération AND.

Table de vérité

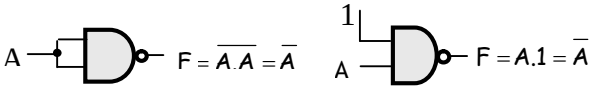
A	B	F
0	0	1
0	1	1
1	0	1
1	1	0

$F = \overline{A \cdot B}$ (se lit (A ET B) tout barre)

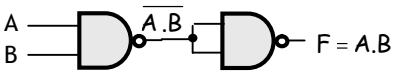
Symbole



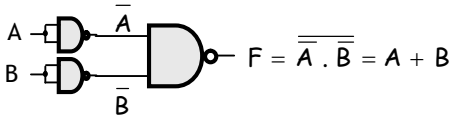
Propriétés

- La fonction NAND est commutative :. $F = \overline{A.B} = \overline{B.A}$
- La fonction NAND n'est pas associative : $F = \overline{A.(\overline{B.C})} \neq \overline{(\overline{A.B}).C} \neq \overline{A.B.C}$
- La fonction NAND est généralisable pour n entrées.
- L'opérateur NAND est dit "système logique complet", car il permet de réaliser toutes les opérations de base : Not, AND et OR ; et par conséquent, toute fonction logique :
  - Réalisation d'un inverseur : 

  - Réalisation d'une AND (F = A . B) : En appliquant le théorème de De Morgan, $F = \overline{\overline{F}} = \overline{\overline{A.B}}$



- Réalisation d'une OR (F = A + B) : De même, $\overline{\overline{F}} = \overline{\overline{A+B}} \Rightarrow F = \overline{\overline{A}.\overline{B}}$ (Morgan)



2.2- Opération NOR :

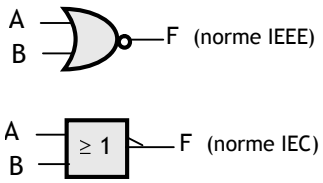
C'est le complément de l'opération OR.

Table de vérité

A	B	F
0	0	1
0	1	0
1	0	0
1	1	0

$F = \overline{A.B}$ (se lit (A ET B) tout barre)

Symbole



Propriétés

- Comme la fonction NAND, la fonction NOR n'est ni combinatoire, ni associative ; elle est aussi généralisable pour n entrées
- L'opérateur NOR est un système logique complet, comme le NAND.

2.3. Opération XOR :

Comme on l'a signalé précédemment, cette opération diffère du OR classique ou inclusif ; l'examen de sa table de vérité ci dessous montre que F est égale à 1 si [(A=0 ET B=1) OU (A=1 ET B=0)] ; formellement, on écrit :

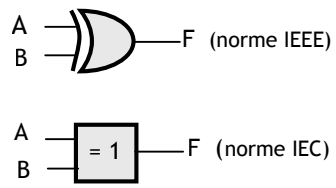
$F = \overline{A}B + A\overline{B}$ qu'on note $F = A \oplus B$

Table de vérité

A	B	F
0	0	0
0	1	1
1	0	1
1	1	0

$F = A \oplus B$ (se lit A OU exclusif B)

Symbole



Le signe "=" indique que la sortie est égale à "1" si une entrée et une seule est égale à 1.

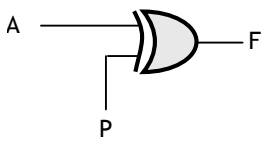
Propretés

- L'opération XOR est commutative : F = A⊕B = B⊕A.
- L'opération XOR est associative : F = A⊕(B⊕C) = (A⊕B)⊕C = A⊕B⊕C.
- L'opération XOR n'est pas généralisable pour n entrées.

Remarque :

L'opérateur OU Exclusif est considéré comme l'opérateur programmable le plus élémentaire. En effet, considérons sa table de vérité dans la figure suivante, on remarque que suivant l'état de P, l'opérateur réalise la fonction OUI ou la fonction NON. Alors P est l'entrée de programmation de cet opérateur.

P	A	F	Fonction réalisée par l'opérateur
0	0	0	SI P = 0 ⇒ F = A ⇒ Fonction Identité
0	1	1	
1	0	1	SI P = 1 ⇒ F = Not A ⇒ Fonction Inversion
1	1	0	



3. REPRESENTATION DES FONCTIONS LOGIQUES :

Pratiquement, une fonction logique est représentée par :

- son **équation logique** qui n'est qu'une association de sommes et de produits logiques ;
- sa table de vérité ou son tableau de Karnaugh, qu'on verra dans le prochain chapitre ;
- Son logigramme qui est une représentation symbolique, sous forme d'un schéma, formé par les différentes liaisons entres les symboles des opérateurs élémentaires.

Exemple : Voilà les 3 représentations d'une certaine fonction F à 3 variables A, X et Y :

- L'équation logique donnée est : F (X, Y, A) = $\bar{A}X+A\bar{Y}$;
- La table de vérité, déduite à partir de l'équation, est : On a 3 variables d'entrées ⇒ on a 2^3 combinaisons possibles (2^3 lignes de la table). D'une façon générale, on a 2^n combinaisons pour n variables d'entrée. On déduit l'équation logique de la fonction F, à partir de la table de vérité suivant le raisonnement suivant :

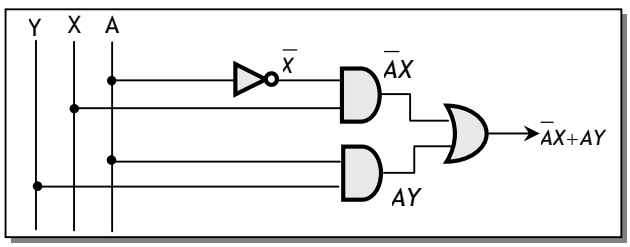
A	X	Y	F
0	0	0	0
0	0	1	0
0	1	0	1
0	1	1	1
1	0	0	0
1	0	1	1
1	1	0	0
1	1	1	1

  - On cherche les lignes où la fonction F est égale à 1 ;
  - On note la combinaison des entrées pour chacune de ces lignes ;
  - On somme logiquement ces combinaisons.

Ainsi, la fonction F est égale à 1 si on a $\bar{A}X\bar{Y}$ **OU** $\bar{A}XY$ **OU** $A\bar{X}Y$ **OU** AXY, ce qui donne :

$$F = \bar{A}X\bar{Y} + \bar{A}XY + A\bar{X}Y + AXY$$
$$= \bar{A}X(Y+\bar{Y}) + AY(X+\bar{X})$$
$$= \bar{A}X + AY$$

- Le logigramme déduit de l'équation est :



On remarque que cette petite fonction emploie différents types de portes logiques : inverseur, AND et OR. Il est évident qu'il serait rentable de réaliser cette fonction logique avec le minimum de matériel (circuits logiques), ce qui demande une bonne analyse du problème pour simplifier la fonction en question.

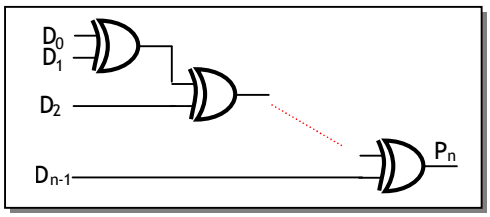
E X E R C I C E S R E S O L U S

EXERCICE N° 1 :

Réaliser un XOR uniquement avec 4 avec des NAND à 2 entrées.

EXERCICE N° 2 :

D'après la table de vérité du XOR, on remarque que la sortie vaut 1 uniquement si le nombre de 1 formé par les 2 entrées est impair ; sinon, elle est à 0. Le schéma ci-contre généralise cette détection de parité pour un mot de n bits.



Montrer par le raisonnement de récurrence que c'est vrai.

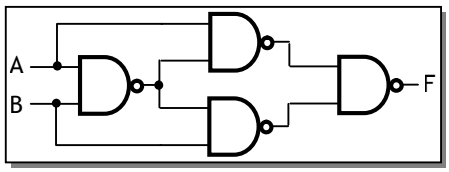
CORRIGE :

EXERCICE N° 1 :

F = $\bar{A}B+A\bar{B}=\bar{A}B+\bar{A}A+A\bar{B}+\bar{B}B$ (Ajout de 2 termes nuls).

F = $A(\bar{A}+\bar{B})+B(\bar{A}+\bar{B})=A(\overline{A.B})+B(\overline{A.B})$

$\bar{\bar{F}}=\overline{\overline{A(\overline{A.B})+B(\overline{A.B})}}=\overline{\overline{A(\overline{A.B})}\,\overline{B(\overline{A.B})}}$, ce qui correspond à 4 portes NAND



EXERCICE N° 2 :

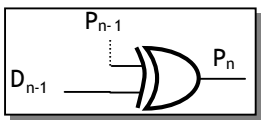
Alors, on suppose que c'est vrai pour (n-1) :

→ P_{n-1} = 1 ($D_{n-2}...D_0$ contient un nombre impaire de 1)
- Si D_{n-1} = 0, P_n = 1
- Si D_{n-1} = 1, P_n = 0

Ce qui est vrai.

→ P_{n-1} = 0 ($D_{n-2}...D_0$ contient un nombre pair de 1)
- Si D_{n-1} = 0, P_n = 0
- Si D_{n-1} = 1, P_n = 1

Ce qui est vrai.



E X E R C I C E S N O N R E S O L U S

EXERCICE N° 1 :

Comme le NAND, le NOR est un système logique complet. Réaliser alors les opérations de base avec cet opérateur.

EXERCICE N° 2 :

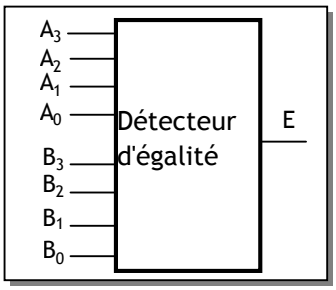
1- Montrer que L'opération NOR n'est pas associative : $F=\overline{A+(\overline{B+C})}\neq\overline{(\overline{A+B})+C}\neq\overline{A+B+C}$.

2- Faire de même pour la porte NAND.

EXERCICE N° 3 :

Réaliser un comparateur entre 2 mots A et B de 4 bits, en se basant sur l'aspect élémentaire de "détecteur d'égalité" du OU EXCLUSIF et en utilisant des portes logiques de base :

- E = 0, s'il n'y a pas d'égalité ;
- E = 1, s'il y a égalité ;



SIMPLIFICATION DES FONCTIONS LOGIQUES

INTRODUCTION :

Une fonction logique est sous forme normale ou canonique si chacun de ses termes contient toutes les variables, directes ou inverses, dont elle dépend. Exemple : $F(X,Y) = X.\overline{Y} + \overline{X}.Y$.

Si l'une de ses variables ne figure pas dans un de ses termes, alors elle est sous forme simplifiée. Cette forme est fort bien recherchée pour aboutir à la réalisation pratique avec un minimum de matériel et à moindre coût. Pour cette fin, on utilise, en général, 3 méthodes :

- La méthode algébrique ;
- La méthode graphique à base du diagramme de Karnaugh ;
- Les méthodes programmables.

Dans ce chapitre, on s'intéresse uniquement aux 2 premières méthodes.

1. METHODE ALGEBRIQUE :

Cette méthode utilise les principes de l'Algèbre de Boole. On en rappelle ci-après 3 parmi les plus importants :

- $A + AB = A(1+B) = A$
- $AB + A\overline{B} = B(A + \overline{A}) = B$
- $A + \overline{A}B = A + B$; en effet, $A + \overline{A}B = (A + \overline{A})(A + B) = AA + AB + A\overline{A} + \overline{A}B = A + B$

Le principe consiste à mettre en oeuvre ces propriétés, dans l'expression à simplifier, par exemple en ajoutant un terme déjà existant :

$$Z = ABC + \overline{A}BC + A\overline{B}C + AB\overline{C}$$

$$Z = \underbrace{ABC + \overline{A}BC}_{BC} + \underbrace{ABC + A\overline{B}C}_{AC} + \underbrace{ABC + AB\overline{C}}_{AB}$$

Cette méthode apprend de la rigueur, mais elle n'est pratiquement plus utilisée systématiquement.

2. METHODE GRAPHIQUE :

2.1. Tableau de Karnaugh et principe de simplification :

Cette méthode plus simple utilise le tableau de Karnaugh pour simplifier des fonctions booléennes ayant jusqu'à 6 variables. Le tableau de Karnaugh d'une fonction logique est la transformation de sa table de vérité sous forme d'une table contractée à 2 dimensions. La méthode consiste principalement à mettre en évidence graphiquement ou visuellement, les groupements de cases, de type :

$$AB + A\overline{B} = A(B + \overline{B}) = A$$

Aussi, dans un tableau de Karnaugh, on peut utiliser une case plusieurs fois selon la relation de la redondance :

$$X + X + \ldots + X = X$$

Le passage de la table de vérité au tableau de Karnaugh se fait selon la procédure suivante:

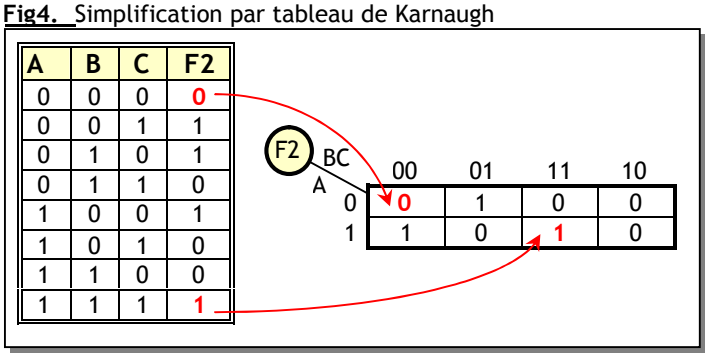
- Chaque ligne de la table de vérité correspond à une case du tableau de Karnaugh ;
- Les cases sont disposées de telle sorte que le passage d'une case à une case voisine se fasse par changement de l'état d'une seule variable à la fois en utilisant le code GRAY.

La mise en œuvre de cette méthode se fait alors en 2 phases :

- La transcription de la fonction à simplifier dans le tableau de Karnaugh ;
- La recherche des groupements de cases qui donneront des expressions simplifiées.

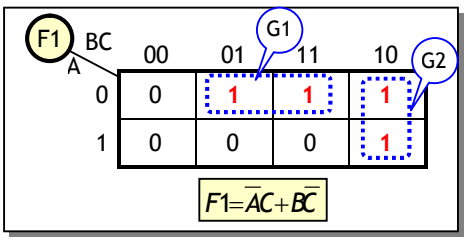
Pour illustrer le passage de la table de vérité au tableau de Karnaugh, la figure 4 montre un exemple pour 3 variables.

Fig4. Simplification par tableau de Karnaugh



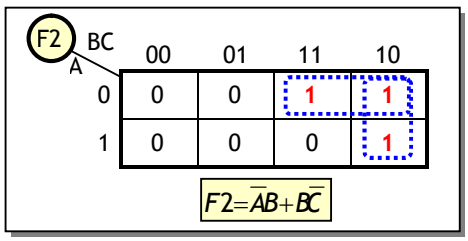
2.2. Exemples de simplification par tableau de Karnaugh :

Exemple 1 :



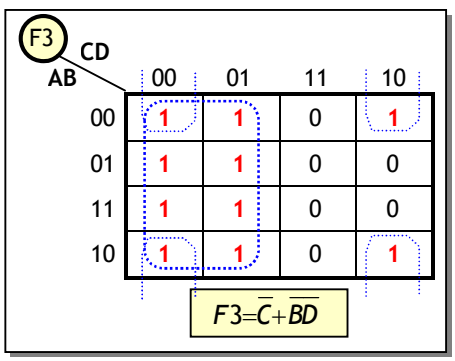
- Dans le groupement 1, c'est B qui a varié, ce qui donne $\overline{A}C$;
- Dans le groupement 2, c'est A qui a varié, ce qui donne $B\overline{C}$.

Exemple 2 :



- Dans ces 2 groupements, on réutilise une case utilisée par les 2 groupement suivant la loi de redondance (X + Y + X = X +Y).

Exemple 3 :



2.3. Conclusion :

- On ne regroupe pas des cases qui ne sont pas symétriques, car cela ne donne pas de termes vérifiant la forme simplificatrice : $AB+A\overline{B}=A(B+\overline{B})=A$.
- Le nombre de variables supprimées dépend de la taille du groupement. Ainsi :
  - Un groupement de 2 cases symétriques entraîne la suppression d'une variable ;
  - Un groupement de 4 cases symétriques entraîne la suppression de 2 variables ;
  - En général, un groupement de 2^k cases entraîne la suppression de k variables.

E X E R C I C E S R E S O L U S

EXERCICE N° 1 :

Simplifier la fonction suivante en utilisant le principe qui consiste à multiplier un terme égal à 1 $(X + \overline{X})$:

$$Z = AB + \overline{B}C + AC$$

EXERCICE N° : Store automatisé

Le système de commande du store étudié dans cet exemple est simplifié par rapport à la réalité pour des raisons didactiques ; en effet le fonctionnement correct du système nécessite des temporisations et des fonctions de mémoire qui ne sont pas étudiées ici :

- Si la luminosité du soleil (s), captée par une cellule solaire, dépasse un seuil prédéfini, on descend le store (D) ;
- 2 boutons poussoirs permettent la descente (d) ou la montée (m) du store ; un appui simultané sur les 2 boutons entraîne la descente du store ;
- Si la vitesse du vent (v), captée par un anémomètre, dépasse un seuil prédéfini, on remonte le store ; ce fonctionnement de sécurité est prioritaire sur tous les autres.

Donner l'équation de la montée du Store M, ainsi que son logigramme, en dressant la table de vérité du système et en utilisant le tableau de Karnaugh.

CORRIGE :

EXERCICE N° 1 :

$$Z = AB + \overline{B}C + AC = AB + \overline{B}C + AC.(1 + \overline{B})$$

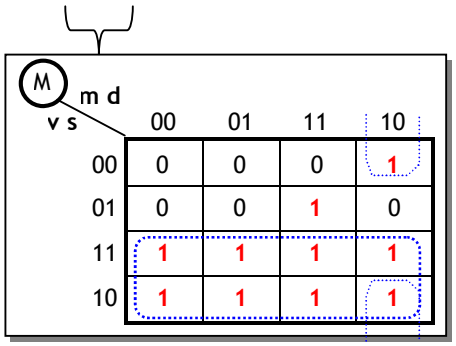
$$Z = AB + \overline{B}C + ACB + AC\overline{B} = AB.(1 + C) + \overline{B}C.(1 + A)$$

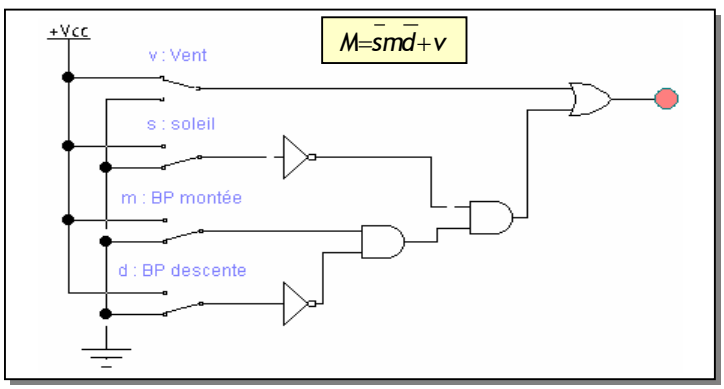
$$Z = AB + \overline{B}C$$

EXERCICE N° 2 :

Table de vérité

v	s	m	d	M	D
0	0	0	0	0	0
0	0	0	1	0	1
0	0	1	0	1	0
0	0	1	1	0	1
0	1	0	0	0	1
0	1	0	1	0	1
0	1	1	0	0	1
0	1	1	1	0	1
1	0	0	0	1	0
1	0	0	1	1	0
1	0	1	0	1	0
1	0	1	1	1	0
1	1	0	0	1	0
1	1	0	1	1	0
1	1	1	0	1	0
1	1	1	1	1	0





CHAPITRE 4

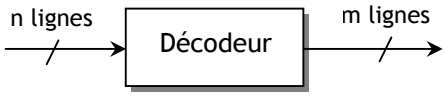
FONCTIONS COMBINATOIRES AVANCEES

INTRODUCTION :

Dans les systèmes numériques, on utilise souvent des fonctions qui on justifié leurs réalisations en circuits intégrés. On note en particulier les décodeurs, les multiplexeurs, les démultiplexeurs et les circuits arithmétiques. Bien qu'ils soient plus ou moins remplacés actuellement par les systèmes programmables (circuits logiques programmables et microprocesseur), ils sont encore utilisés.

1. LES DECODEURS :

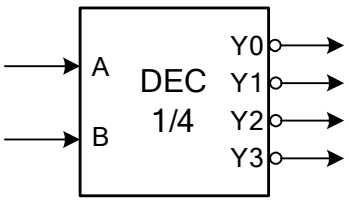
La fonction de décodage consiste à faire correspondre à un code présent en entrée sur n lignes, un autre code en sortie sur m lignes avec en général $m \neq n$:



1.1. Décodeur 1 parmi n:

Ce type de décodeur permet de faire correspondre à un code présent en entrée sur n lignes une sortie et une seule active parmi les N = 2n sorties possibles. On le désigne aussi par décodeur m lignes vers n lignes. Pour comprendre le principe d'un tel décodeur, étudions le décodeur 1 parmi 4 ou 2 vers 4, donné à la figure 1 ; le niveau active des sorties est le 0, car c'est souvent le cas :

Fig. 1 : Décodeur 1 parmi 4 avec sorties actives sur niveau bas



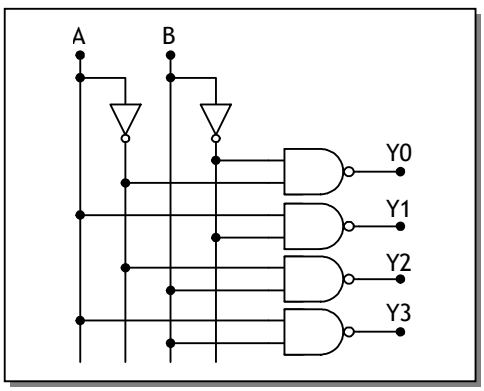
ENTREES		SORTIES			
B	A	Y0	Y1	Y2	Y3
0	0	0	1	1	1
0	1	1	0	1	1
1	0	1	1	0	1
1	1	1	1	1	0

Directement ou à l'aide de la table de Karnaugh, on détermine les équations de sorties :

$$\overline{Y_0} = \overline{B}.\overline{A}$$
$$\overline{Y_1} = \overline{B}.A$$
$$\overline{Y_2} = B.\overline{A}$$
$$\overline{Y_3} = A.B$$

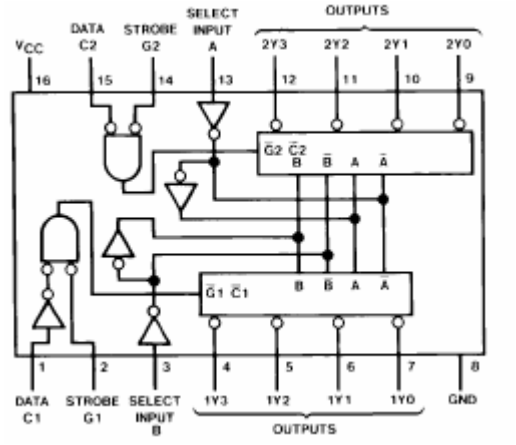
Le schéma d'implémentation du décodeur sera alors celui de la figure 2 ci-contre :

Fig. 2 : Logigramme de décodeur 1 parmi 4



Les circuits intégrés réalisant cette fonction contiennent des entrées de validation comme G ou E permettant de sélectionner le circuit. On peut citer comme exemple le double décodeur 74LS156 dont le brochage et la table de fonction sont donnés à la figure 3 :

Fig. 3 : Diagramme de brochage et table de fonctionnement du 74LS156



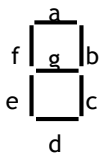
2-Line-to-4-Line Decoder or
1-Line-to-4-Line Demultiplexer

Inputs				Outputs			
Select		Strobe	Data				
B	A	G1	C1	1Y0	1Y1	1Y2	1Y3
X	X	H	X	H	H	H	H
L	L	L	H	L	H	H	H
L	H	L	H	H	L	H	H
H	L	L	H	H	H	L	H
H	H	L	H	H	H	H	L
X	X	X	L	H	H	H	H

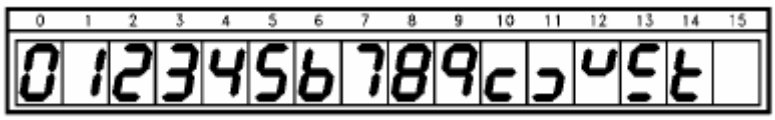
1.2. Décodeur BCD – 7 segments :

Ce type de décodeur permet de convertir le code BCD 4bits à l'entrée pour obtenir à la sortie un code 7 segments permettant de commander un afficheur 7 segments permettant l'écriture de tous les chiffres et aussi d'autres symboles comme le montre la figure 4 :

Fig. 4 : Afficheur 7 segments



Identification des segments



Désignations numériques et résultat de l'affichage

Pour mettre en équation ce type de décodeur, il faut dresser la table de vérité suivante :

Nombre BCD à décoder	ENTREES				SORTIES						
	D	C	B	A	a	b	c	d	e	f	g
0	0	0	0	0	1	1	1	1	1	1	0
1	0	0	0	1	0	1	1	0	0	0	0
2	0	0	1	0	1	1	0	1	1	0	1
3	0	0	1	1	1	1	1	1	0	0	1
4	0	1	0	0	0	1	1	0	0	1	1
5	0	1	0	1	1	0	1	1	0	1	1
6	0	1	1	0	0	0	1	1	1	1	1
7	0	1	1	1	1	1	1	0	0	0	0
8	1	0	0	0	1	1	1	1	1	1	1
9	1	0	0	1	1	1	1	0	0	1	1

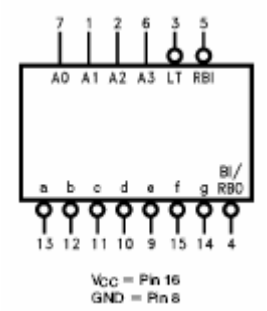
La table de karnaugh de chaque segment permet alors d'obtenir les équations de ce décodeur. Les 0 étant les moins nombreux, l'écriture des équations de commande d'extinction des segments sera plus facile :

$$\overline{a} = A.\overline{B}.\overline{C}.\overline{D} + \overline{A}.C \qquad \overline{b} = A.\overline{B}.C + \overline{A}.B.C \qquad \overline{c} = \overline{A}.B.\overline{C} \qquad \overline{d} = \overline{A}.\overline{B}.C + A.B.C + A.\overline{B}.\overline{C}$$

$$\overline{e} = A + \overline{B}.C \qquad \overline{f} = A.\overline{C}.\overline{D} + A.B + B.\overline{C} \qquad \overline{g} = \overline{B}.\overline{C}.\overline{D} + A.B.C$$

Comme exemple de décodeur, on peut citer le circuit intégré 74LS47 dont le schéma de brochage et la table de vérité sont données à la figure 5 :

Fig. 5 : Diagramme de brochage et table de fonctionnement du 74LS47



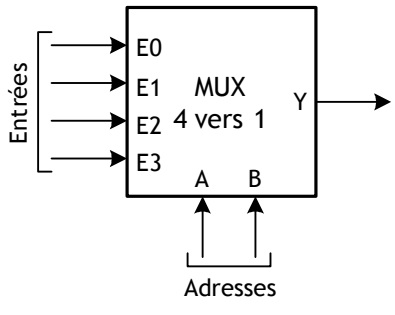
Truth Table

Decimal or Function	Inputs							Outputs						
	$\overline{LT}$	$\overline{RBI}$	A3	A2	A1	A0	$\overline{BI/RBO}$	$\overline{a}$	$\overline{b}$	$\overline{c}$	$\overline{d}$	$\overline{e}$	$\overline{f}$	$\overline{g}$
0	H	H	L	L	L	L	H	L	L	L	L	L	L	H
1	H	X	L	L	L	H	H	H	L	L	H	H	H	H
2	H	X	L	L	H	L	H	L	L	H	L	L	H	L
3	H	X	L	L	H	H	H	L	L	L	L	H	H	L
4	H	X	L	H	L	L	H	H	L	L	H	H	L	L
5	H	X	L	H	L	H	H	L	H	L	L	H	L	L
6	H	X	L	H	H	L	H	H	H	L	L	L	L	L
7	H	X	L	H	H	H	H	L	L	L	H	H	H	H
8	H	X	H	L	L	L	H	L	L	L	L	L	L	L
9	H	X	H	L	L	H	H	L	L	L	H	H	L	L
10	H	X	H	L	H	L	H	H	H	H	L	L	H	L
11	H	X	H	L	H	H	H	H	H	L	L	H	H	L
12	H	X	H	H	L	L	H	H	L	H	H	H	L	L
13	H	X	H	H	L	H	H	L	H	H	L	H	L	L
14	H	X	H	H	H	L	H	H	H	H	L	L	L	L
15	H	X	H	H	H	H	H	H	H	H	H	H	H	H
$\overline{BI}$	X	X	X	X	X	X	L	H	H	H	H	H	H	H
$\overline{RBI}$	H	L	L	L	L	L	L	H	H	H	H	H	H	H
$\overline{LT}$	L	X	X	X	X	X	H	L	L	L	L	L	L	L

2. LE MULTIPLEXEUR :

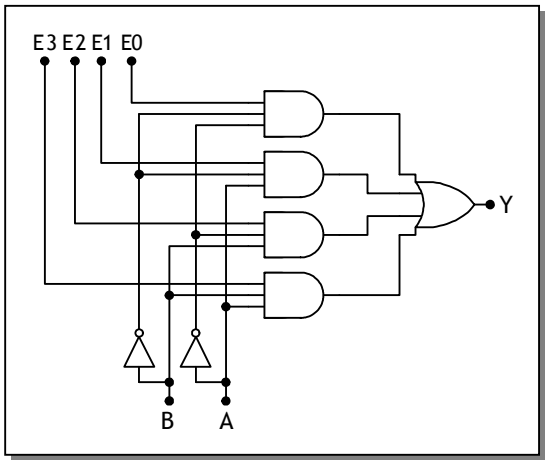
Un multiplexeur permet de sélectionner une entrée parmi 2n pour transmettre l'information portée par cette ligne à un seul canal de sortie. La sélection de l'entrée se fait alors à l'aide de n lignes d'adressage. Pour comprendre le principe, considérons un multiplexeur à quatre entrées (figure 6), donc deux lignes d'adressage et une ligne de sortie :

Fig. 6 : Multiplexeur 4 vers 1



ADRESSES		SORTIE
B	A	Y
0	0	E0
0	1	E1
1	0	E2
1	1	E3

Fig. 7 : Logigramme de multiplexeur 4 vers 1



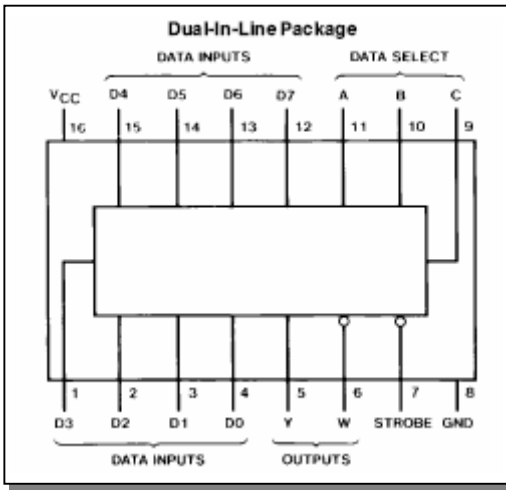
De la table de vérité, on déduit l'expression logique de la sortie Y :

$$Y = \overline{A}.\overline{B}.E0 + A.\overline{B}.E1 + \overline{A}.B.E2 + A.B.E3$$

Le schéma d'implantation du multiplexeur 4 vers 1 sera celui de la figure 7 ci-contre.

Les circuits intégrés réalisant cette fonction contiennent des entrées de validation (Strobe - Enable) permettant de sélectionner le circuit comme le 74LS151 qui est multiplexeur 8 vers 1 (figure 8) :

Fig. 8 : Diagramme de brochage et table de fonctionnement du 74LS151

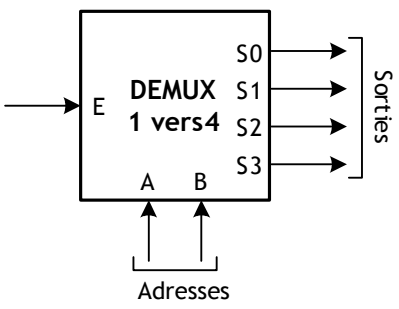


Inputs				Outputs	
Select			Strobe		
C	B	A	S	Y	W
X	X	X	H	L	H
L	L	L	L	D0	$\overline{D0}$
L	L	H	L	D1	$\overline{D1}$
L	H	L	L	D2	$\overline{D2}$
L	H	H	L	D3	$\overline{D3}$
H	L	L	L	D4	$\overline{D4}$
H	L	H	L	D5	$\overline{D5}$
H	H	L	L	D6	$\overline{D6}$
H	H	H	L	D7	$\overline{D7}$

3. LE DEMULTIPLEXEUR :

Le démultiplexeur effectue l'opération inverse d'un multiplexeur à savoir il permet de distribuer l'information présente à l'entrée vers l'une des 2n sorties. La sélection de la sortie se fait à l'aide de n lignes d'adressage. Pour comprendre le principe, considérons un démultiplexeur à quatre sorties (voir figure 9), donc deux lignes d'adressage et une ligne d'entrée :

Fig. 9 : Demultiplexeur 1 vers 4



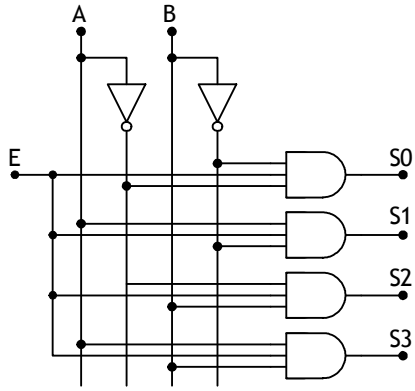
ADRESSES		SORTIES			
B	A	S0	S1	S2	S3
0	0	E	0	0	0
0	1	0	E	0	0
1	0	0	0	E	0
1	1	0	0	0	E

A partir de la table de vérité, on détermine les équations de sortie suivantes :

$$S_0 = E.\overline{B}.\overline{A}$$
$$S_1 = E.\overline{B}.A$$
$$S_2 = E.B.\overline{A}$$
$$S_3 = E.A.B$$

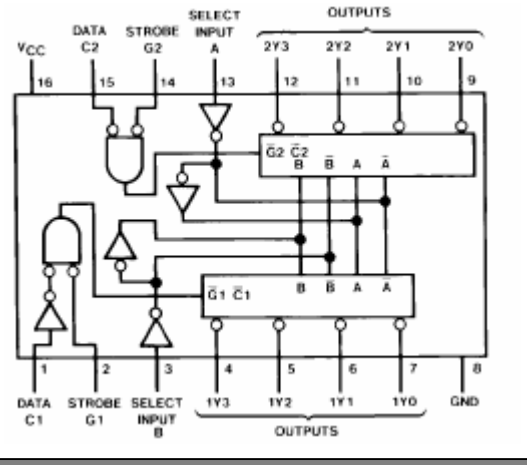
Le schéma d'implémentation du démultiplexeur sera alors celui de la figure 10 ci-contre :

Fig. 10 : Logigramme de démultiplexeur 1 vers 4



Les circuits intégrés réalisant cette fonction contiennent des entrées de validation (Strobe et Enable) permettant de sélectionner le circuit comme le 74LS155 qui est un double démultiplexeur 1 vers 4 dont le schéma de brochage et la table de vérité sont données à la figure 11 :

Fig. 11 : Diagramme de brochage et table de fonctionnement du 74LS155



2-Line-to-4-Line Decoder or
1-Line-to-4-Line Demultiplexer

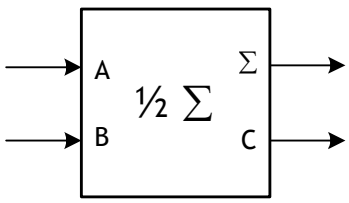
Inputs				Outputs			
Select		Strobe	Data				
B	A	G1	C1	1Y0	1Y1	1Y2	1Y3
X	X	H	X	H	H	H	H
L	L	L	H	L	H	H	H
L	H	L	H	H	L	H	H
H	L	L	H	H	H	L	H
H	H	L	H	H	H	H	L
X	X	X	L	H	H	H	H

4. L'ADDITIONNEUR :

4.1- Le demi-additionneur :

C'est un circuit permettant d'effectuer l'addition de deux bits A et B pour générer leur somme Σ et leur retenue C (Carry) comme le montre le schéma et la table de vérité de la figure 12 :

Fig. 12 : Le Demi-Additionneur



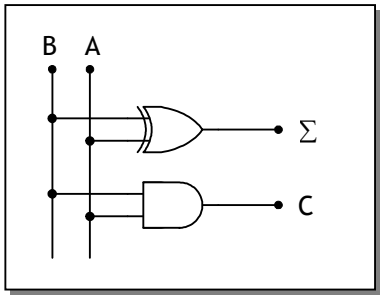
ENTREES		SORTIES	
B	A	Σ	C
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

A partir de la table de vérité, on peut écrire les deux fonctions sous la forme suivante :

$$\sum = A.\overline{B} + \overline{A}.B = A \oplus B \qquad C = A.B$$

Ce qui peut être réalisé par le circuit schématisé sur le logigramme de la figure 13 ci-contre.

Fig. 13 : Le Demi-Additionneur



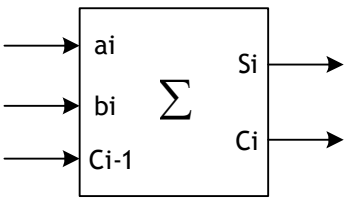
4.2- L'additionneur complet :

Pour effectuer une addition de deux nombres binaires de n bits, on additionne successivement les bits du même poids en tenant compte de la retenue de l'addition précédente comme le montre l'exemple suivant :

	a_3	a_2	a_1	a_0	Nombre A
+	b_3	b_2	b_1	b_0	Nombre B
	S_3	S_2	S_1	S_0	Somme : S = A+B
←	C_3	C_2	C_1	C_0	Retenues

Il faut donc concevoir une cellule élémentaire appelée additionneur complet et qui permet de réaliser l'addition des bits ai et bi en plus de la retenue Ci-1 de l'addition précédente. Un tel circuit est définit par le schéma et la table de vérité de la figure 14 :

Fig. 14 : Additionneur complet



ENTREES			SORTIES	
ai	bi	Ci-1	Si	Ci
0	0	0	0	0
0	0	1	1	0
0	1	0	1	0
0	1	1	0	1
1	0	0	1	0
1	0	1	0	1
1	1	0	0	1
1	1	1	1	1

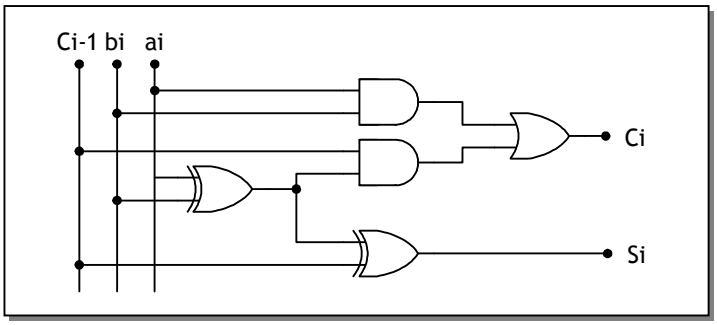
A l'aide de la table de Karnaugh, on détermine les équations de sorties suivantes :

$$S_i = a_i.\overline{b_i}.\overline{C_{i-1}} + \overline{a_i}.b_i.\overline{C_{i-1}} + \overline{a_i}.\overline{b_i}.C_{i-1} + a_i.b_i.C_{i-1} = a_i \oplus b_i \oplus C_{i-1}$$

$$C_i = a_i.b_i + a_i.\overline{b_i}.C_{i-1} + \overline{a_i}.b_i.C_{i-1} = a_i.b_i + (a_i \oplus b_i).C_{i-1}$$

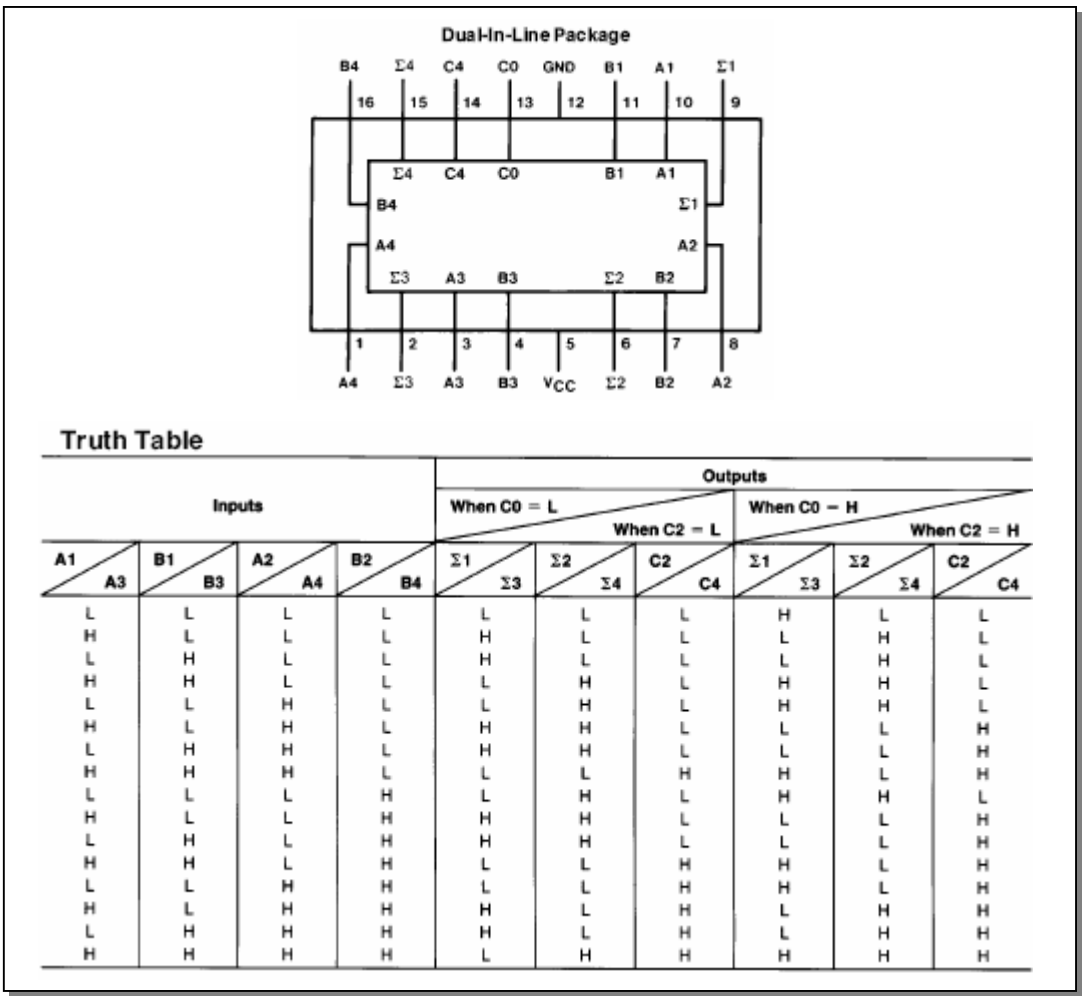
Le schéma d'implantation de l'additionneur complet sera celui de la figure 15 :

Fig. 15 : Logigramme d'un additionneur complet



Comme exemple d'additionneur complet de mots de 4 bits , on peut citer le circuit intégré 74LS83 dont le schéma de brochage et la table de vérité sont données à la figure 16 :

Fig. 16 : Additionneur 4bits 74LS83



Truth Table

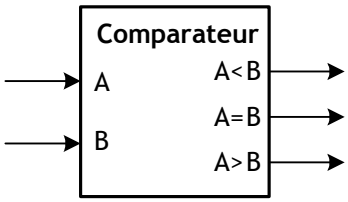
Inputs				Outputs					
				When C0 = L			When C0 = H		
				When C2 = L			When C2 = H		
A1 / A3	B1 / B3	A2 / A4	B2 / B4	Σ1 / Σ3	Σ2 / Σ4	C2 / C4	Σ1 / Σ3	Σ2 / Σ4	C2 / C4
L	L	L	L	L	L	L	H	L	L
H	L	L	L	H	L	L	L	H	L
L	H	L	L	H	L	L	L	H	L
H	H	L	L	L	H	L	H	H	L
L	L	H	L	L	H	L	H	H	L
H	L	H	L	H	H	L	L	L	H
L	H	H	L	H	H	L	L	L	H
H	H	H	L	L	L	H	H	L	H
L	L	L	H	L	H	L	H	H	L
H	L	L	H	H	H	L	L	L	H
L	H	L	H	H	H	L	L	L	H
H	H	L	H	L	L	H	H	L	H
L	L	H	H	L	L	H	H	L	H
H	L	H	H	H	L	H	L	H	H
L	H	H	H	H	L	H	L	H	H
H	H	H	H	L	H	H	H	H	H

5. LE COMPARATEUR :

Un comparateur est un circuit permettant de détecter l'égalité de deux nombres et éventuellement d'indiquer le nombre le plus grand ou le plus petit.

Pour comprendre le principe, on va réaliser un comparateur simple permettant de comparer deux mots de 1 bit. La table de vérité d'un tel comparateur est donnée à la figure 17 :

Fig. 17 : Comparateur de 2 mots de 1 bit



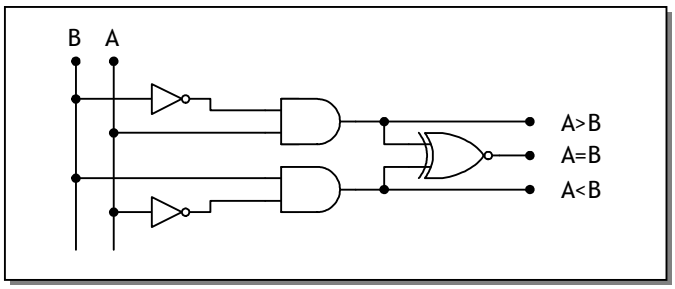
ENTREES		SORTIES		
B	A	S1:A<B	S2:A=B	S3:A>B
0	0	0	1	0
0	1	0	0	1
1	0	1	0	0
1	1	0	1	0

A partir de la table de vérité, on peut écrire les trois fonctions sous la forme suivante :

$$S1 = \overline{A}.B \qquad S2 = A.\overline{B} \qquad S3 = \overline{S1 \oplus S3}$$

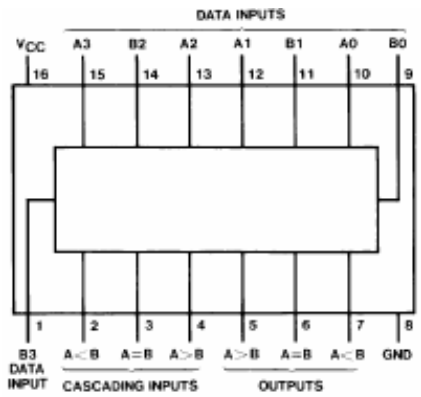
Le schéma d'implantation de ce comparateur 2 bits sera celui de la figure 18 :

Fig. 18 : Logigramme du comparateur de 2 mots de 1 bit



Comme exemple de comparateur binaire, on peut citer le circuit intégré 74LS85 dont le schéma de brochage et la table de vérité sont données à la figure 19 :

Fig. 19 : Comparateur 4 bits 74LS85



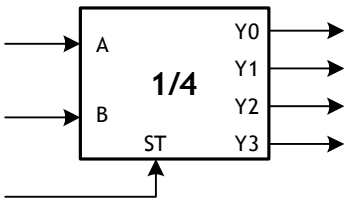
Function Table

Comparing Inputs				Cascading Inputs			Outputs		
A3, B3	A2, B2	A1, B1	A0, B0	A > B	A < B	A = B	A > B	A < B	A = B
A3 > B3	X	X	X	X	X	X	H	L	L
A3 < B3	X	X	X	X	X	X	L	H	L
A3 = B3	A2 > B2	X	X	X	X	X	H	L	L
A3 = B3	A2 < B2	X	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 > B1	X	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 < B1	X	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 > B0	X	X	X	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 < B0	X	X	X	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	L	L	H	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	H	L	L	H	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	X	X	H	L	L	H
A3 = B3	A2 = B2	A1 = B1	A0 = B0	H	H	L	L	L	L
A3 = B3	A2 = B2	A1 = B1	A0 = B0	L	L	L	H	H	L

E X E R C I C E S R E S O L U S

EXERCICE N° 1:

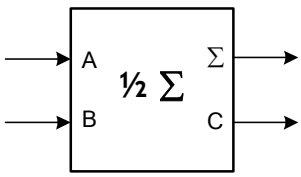
En utilisant deux décodeurs 1/4, réaliser un décodeur 1/8 :



ST : entrée de validation active sur niveau haut

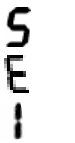
EXERCICE N° 2:

Réaliser un additionneur complet en utilisant deux demi-additionneurs :



EXERCICE N° 3:

On désire afficher le résultat de la comparaison de deux nombres binaires 4 bits A et B avec un afficheur 7 segments. Etudier le circuit qui permet de rendre lumineux les segments de façon à afficher :



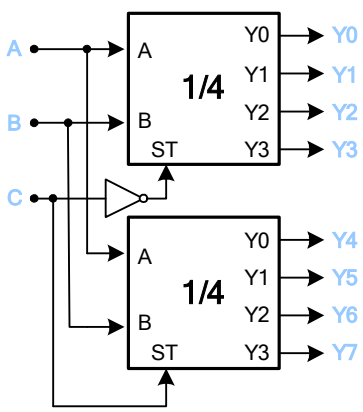
Si A > B

Si A = B

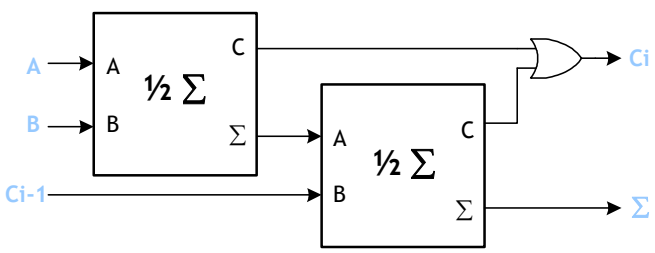
Si A < B

CORRIGES :

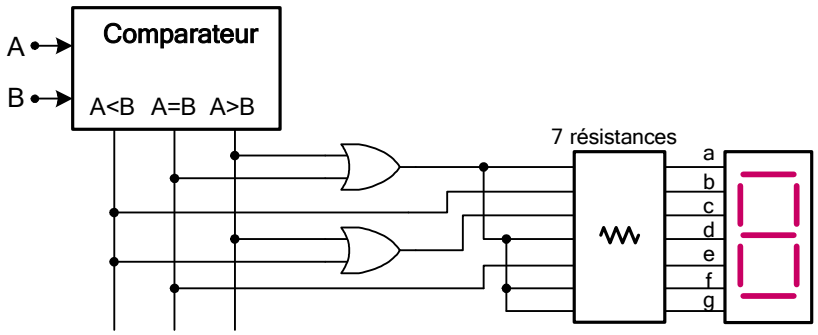
EXERCICE N° 1 :



EXERCICE N° 2 :

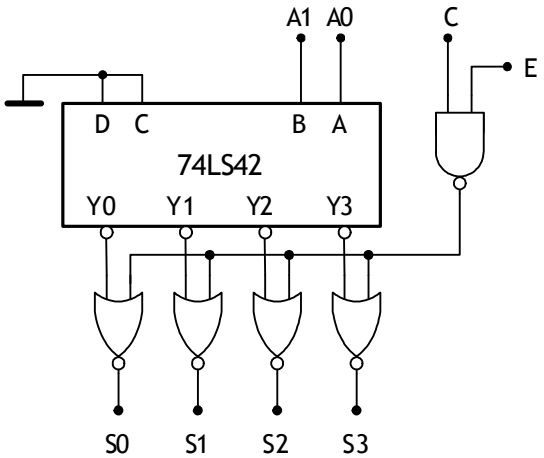


EXERCICE N° 3 :



E X E R C I C E N O N R E S O L U

L'étude suivante permettra de déterminer le rôle du circuit logique du schéma suivant :



Le circuit intégré 74LS42 est un décodeur 1 parmi 10 qui est utilisé uniquement comme un décodeur 1 parmi 4 :

1. C = 0. Donner l'état direct de S0, S1, S2 et S3 ;
2. C = 1 :
   a. compléter la table de vérité suivante :
   b.

C	A1	A0	S0	S1	S2	S3
1	0	0	E			
1						
1						
1						

   c. c. Donner alors l'expression de S0, S1, S2 et S3 en fonction de C, A1, A0 et E.

3. Quelle est alors la fonction de l'ensemble du circuit ?

CHAPITRE 5

NOTION DE MEMOIRE

INTRODUCTION :

A la différence d'un circuit combinatoire, l'état d'un circuit séquentiel dépend de l'état de ses entrées et de l'état précédent de ses sorties ; il doit donc "se rappeler" ou avoir de la "mémoire". Par mémoire, on exprime le phénomène qui consiste à conserver l'effet d'un événement après sa disparition.

1. CIRCUIT MEMOIRE EN TECHNOLOGIE ELECTRIQUE :

Pour introduire à ce type de circuit, on étudie un exemple simple et classique dans ce domaine ; il s'agit de la commande d'un moteur d'une perceuse par exemple :

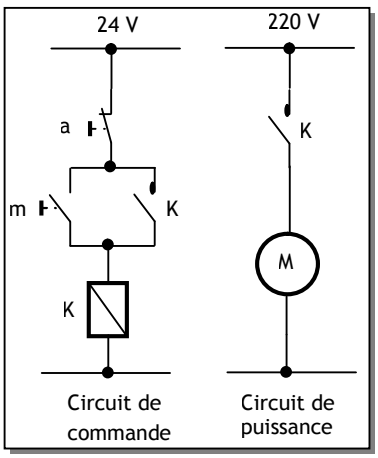
- Un bouton "m" permet de mettre en marche le moteur et un bouton "a" permet de l'arrêter ;
- Quand on appuie sur le bouton m, le moteur démarre ; quand on relâche le bouton, le moteur continue à tourner. L'ordre de mise en marche a donc été mémorisé ;
- Il en est de même pour le bouton a ;
- L'action arrêt est prioritaire : si m et a sont appuyés en même temps, on arrête le moteur.

On connaît déjà la solution de ce problème, figurant dans le circuit d'auto maintien ou d'auto-alimentation :

- Le bouton m est un contact ouvert au repos ; le bouton a est un contact fermé au repos. Le relais K dispose de 2 contacts : un utilisé dans le circuit de commande et l'autre utilisé dans le circuit de puissance.

- Quand l'utilisateur appuie sur m, la bobine du relais est alimentée. Les contacts K associés se ferment. Si l'utilisateur relâche m, le courant continue à circuler par K ; le relais est alors auto-alimenté et le moteur continue à tourner. L'équation du relais X et du moteur M est :

$$K=(K+m)\bar{a}$$

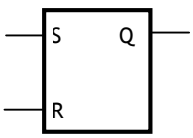


Il s'agit du circuit mémoire élémentaire en technologie électrique.

2. CIRCUIT MEMOIRE EN TECHNOLOGIE ELECTRONIQUE :

Les circuits mémoire électroniques sont d'une grande variété à différents champs d'application ; on étudie ici le circuit le plus élémentaire, qu'on peut qualifier de circuit de base pour tous les autres ; il s'agit de la bascule SR :

- Son symbole est représenté par la figure ci-contre ;
- On l'appelle bascule, car elle bascule d'un état à l'autre suivant l'état de ses entrées S et R ;
- S (Set) est l'entrée de mise à 1 de la sortie Q ;
- R (Reset) est l'entrée de mise à 0 de la sortie Q.



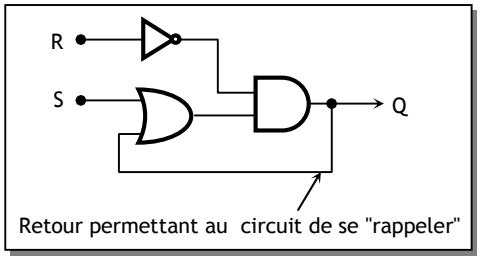
On développera le circuit de cette bascule en utilisant 2 approches :

- Dans l'approche 1, on exploite le résultat du paragraphe 1 ;
- Dans l'approche 2, on utilise le raisonnement comme pour un circuit combinatoire.

2.1. Approche 1 :

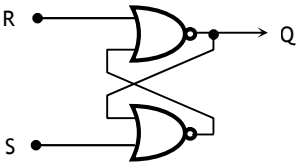
Dans cette approche, on part de l'équation d'auto-alimentation du relais et on fait la correspondance logique : S correspond à m, R correspond à a et Q correspond à K. On en déduit alors l'équation de la bascule SR avec Reset prioritaire, ainsi que son logigramme :

$$Q=(Q+S)\overline{R}$$



Ce circuit est plus connu par sa réalisation simplifiée avec l'utilisation de portes NOR :

$$Q=\overline{\overline{Q}}$$
$$=\overline{\overline{(Q+S).\overline{R}}}$$
$$=\overline{\overline{(Q+S)}+R}$$



S	R	Q	Fonction de la bascule
0	0	x	Mémorisation
0	1	1	Action Reset
1	0	0	Action Set
1	1	0	Action Reset (*)

X indique l'état précédent (0 ou 1)
(*) Etat indéterminé si on passe à SR=00.

2.2. Approche 2 :

Dans cette approche, on raisonne comme pour un circuit combinatoire. Il est donc nécessaire de connaître l'état de Q pour connaître l'état de la sortie lorsque les deux entrées sont à 0 (état de mémoire). On introduit alors une variable supplémentaire qui indique l'état précédent de Q. On note "q" cette variable.

S	R	q	Q
0	0	0	0
0	0	1	1
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Q / S \ Rq	00	01	11	10
0	0	1	0	0
1	1	1	0	0

$$Q=S\overline{R}+\overline{R}q$$
$$=(q+S)\overline{R}$$

Puisqu' on a (q = Q), alors :

$$Q=(Q+S)\overline{R}$$

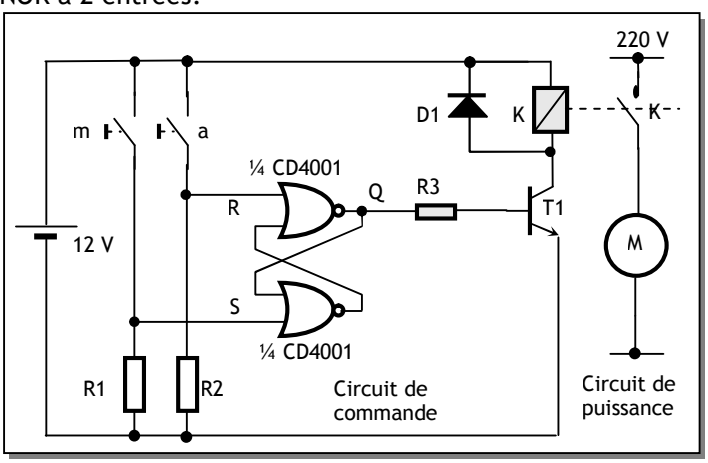
EXERCICE RESOLU

Il s'agit de trouver la version électronique du montage Marche/Arrêt du moteur ; le circuit utilisé est le circuit CMOS CD4011 comportant 4 portes NOR à 2 entrées.

CORRIGE :

Le montage est le suivant :

→ L'appui sur m correspond à l'action Set de la bascule SR, ce qui fait conduire le transistor T1 et exciter le relais K ; le moteur tourne ; le relâchement de ce bouton correspond à la mémorisation de cet état ;



→ L'appui sur a correspond à l'action Reset, ce qui fait bloquer T1 et désexciter le relais K ; le moteur s'arrête ; le relâchement de ce bouton correspond à la mémorisation de cet état.

CHAPITRE 6

FONCTIONS SEQUENTIELLES

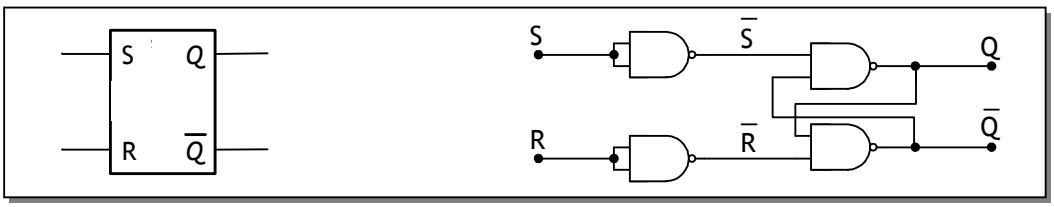
1. LES BASCULES :

A l'instar des opérateurs logiques élémentaires en logique combinatoire, les bascules (appelées aussi flip-flop) sont les éléments de base de la logique séquentielle

1.1. Bascule SR Asynchrone :

Les bascules RS sont à la base de tous les éléments de mémorisation. Il s'agit d'un montage utilisant deux portes NAND et capable de mémoriser un niveau logique choisi à l'aide de deux sorties complémentaires. Son schéma est donné à la figure 1 :

Fig. 1 : Bascule SR asynchrone

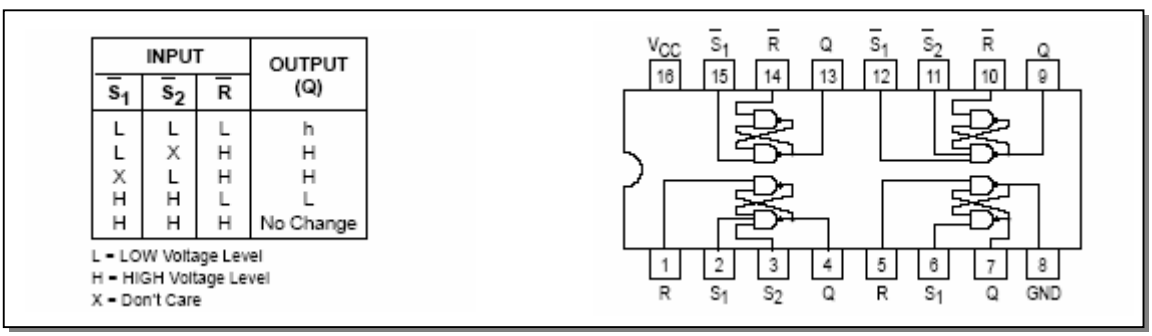


La table de vérité d'une telle bascule est la suivante :

Entrées		Etat précédent	Sorties		Commentaires
S	R	Q_{n-1}	Q_n	$\overline{Q}_n$	
0	0	0	0	1	Mémorisation de l'état précédent
0	0	1	1	0	
0	1	X	0	1	Remise à zéro de la sortie Q_n
1	0	X	1	0	Remise à un de la sortie Q_n
1	1	X	1	1	Etat indéfini

Parmi les circuits intégrés à bascules $\overline{S}\overline{R}$, on trouve le 74279 dont la table de fonction et le schéma de brochage sont donnés à la figure 2 :

Fig. 2 : Brochage et table de vérité du 74279



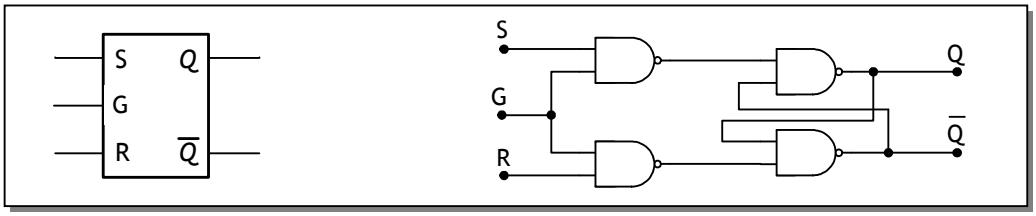
Cette bascule présente deux inconvénients majeurs :

- Sensibilité de la sortie Q aux changements indésirables (parasites) des entrées S et R ;
- La configuration S = R = 1 est à éviter parce qu'elle conduit à l'égalité entre les deux sorties Q *et* $\overline{Q}$ et donc il n'y a plus complémentarité comme c'est indiqué dans la définition de la bascule ;

1.2. Bascule SR Synchrone :

La bascule SR synchrone (figure 3) permet de résoudre le premier inconvénient de la bascule SR asynchrone. Les ordres Set et Reset ne changent l'état de la sortie qu'après l'autorisation d'un signal de commande G (Gate) :

Fig. 3 : Bascule SR synchrone

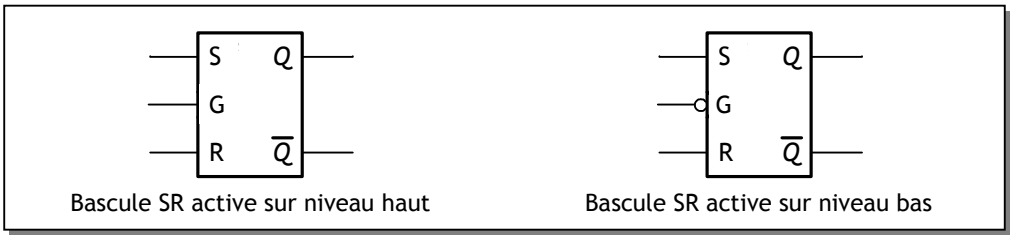


La table de vérité d'une telle bascule est la suivante :

G	S	R	Q_n	Commentaires
0	X	X	Q_{n-1}	Mémorisation de l'état de la sortie
1	0	0	Q_{n-1}	Fonctionnement normal de la bascule SR
1	0	1	0	
1	1	0	1	

Le changement d'état de cette bascule est autorisé sur niveau logique 1 du signal de commande G. On dit que c'est une bascule commandée de manière statique (sur niveau logique 0 ou 1) et sa représentation est donnée à la figure 4 :

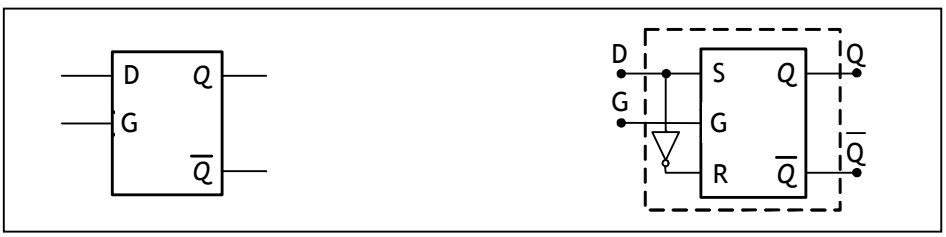
Fig. 4: Bascule SR statique



1.3. Bascule D :

La bascule D est dérivée de la bascule SR en ajoutant un inverseur entre Set et Reset pour n'avoir plus qu'une seule entrée pour fixer le niveau à mémoriser. Avec un tel montage, il n'y a plus de combinaisons d'entrées invalides (S=R=1) :

Fig. 5: Bascule D



La table de vérité d'une telle bascule est la suivante :

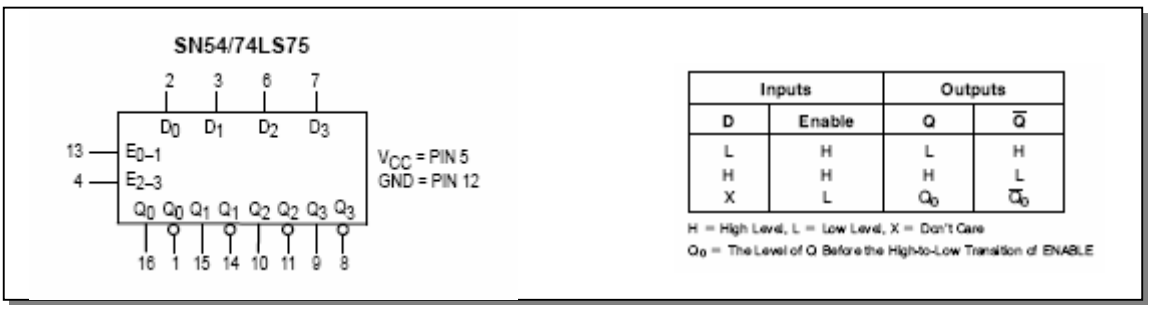
G	D	Q_n	Commentaires
0	X	Q_{n-1}	Mémorisation de l'état de la sortie
1	0	0	Mise à zéro de la sortie Q
1	1	1	Mise à un de la sortie Q

Le changement d'état d'une bascule D peut aussi être autorisé par le front montant ou le front descendant du signal de commande G. On parle alors de bascule D dynamique (figure 6) :

Fig. 6: Bascule D dynamique

Parmi les circuits intégrés à bascules D, on trouve le 7475 dont la table de fonction et le schéma de brochage sont donnés par la figure 7 :

Fig. 7: Brochage et table de vérité du 74LS75

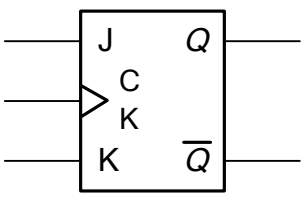


Inputs		Outputs	
D	Enable	Q	$\overline{Q}$
L	H	L	H
H	H	H	L
X	L	Q_0	$\overline{Q}_0$

H = High Level, L = Low Level, X = Don't Care
Q_0 = The Level of Q Before the High-to-Low Transition of ENABLE

1.4. Bascule JK :

Le principe des bascules dynamiques permet de mieux protéger la bascule contre les changements indésirables des entrées. La bascule JK permet en plus de lever l'ambiguïté qui existe pour l'état S=R=1 d'une bascule SR. Son symbole et sa table de vérité sont les suivants :



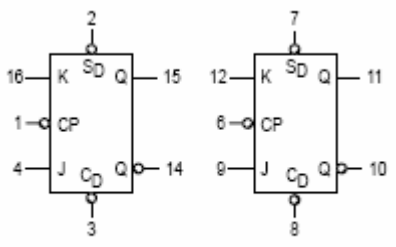
CK	J	K	Q_n	Commentaires
↑	0	0	Q_{n-1}	Mémorisation de l'état de la sortie
↑	0	1	0	Mise à 0 de la sortie Q
↑	1	0	1	Mise à 1 de la sortie Q
↑	1	1	$\overline{Q}_{n-1}$	Basculement de l'état de la sortie

Parmi les circuits intégrés à bascules JK, on trouve le 7476 dont la table de fonction et le schéma de brochage sont donnés à la figure 8 :

Fig. 8: Brochage et table de vérité du 74LS76

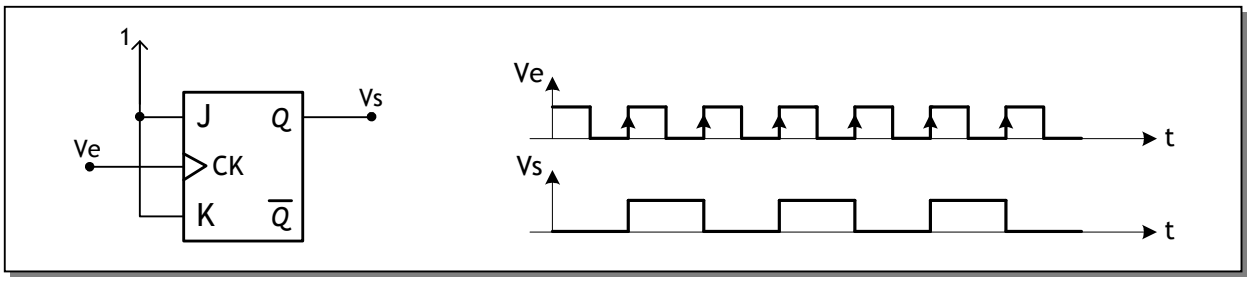
INPUTS				OUTPUTS	
S_D	C_D	J	K	Q	Q
L	H	X	X	H	L
H	L	X	X	L	H
L	L	X	X	H	H
H	H	h	h	q	q
H	H	l	h	L	H
H	H	h	l	H	L
H	H	l	l	q	q

l, h (q) = Lower case letters indicate the state of the referenced input (or output) one setup time prior to the HIGH-to-LOW clock transition



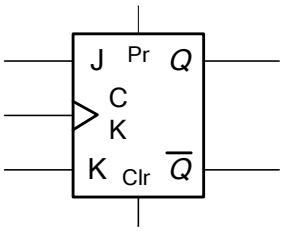
Si on utilise la bascule JK avec J = K = 1, on obtient l'une des principales applications de la bascule JK à savoir le diviseur de fréquence par 2 (figure 9) :

Fig. 9: Diviseur de fréquence par 2



1.5. Fonctionnement forcé des bascules :

Il est parfois nécessaire d'affecter le niveau de sortie d'une bascule de manière non synchrone c'est-à-dire indépendamment de l'horloge. C'est le rôle des entrées de forçage asynchrone Preset (Set) et Clear (Clr) qui permettent d'initialiser la bascule :



PR	Clr	Q_n	Commentaires
0	0	Q_n	Fonctionnement normal de la bascule
0	1	0	Forçage à 0 de la sortie Q
1	0	1	Forçage à 1 de la sortie Q

2. LES COMPTEURS :

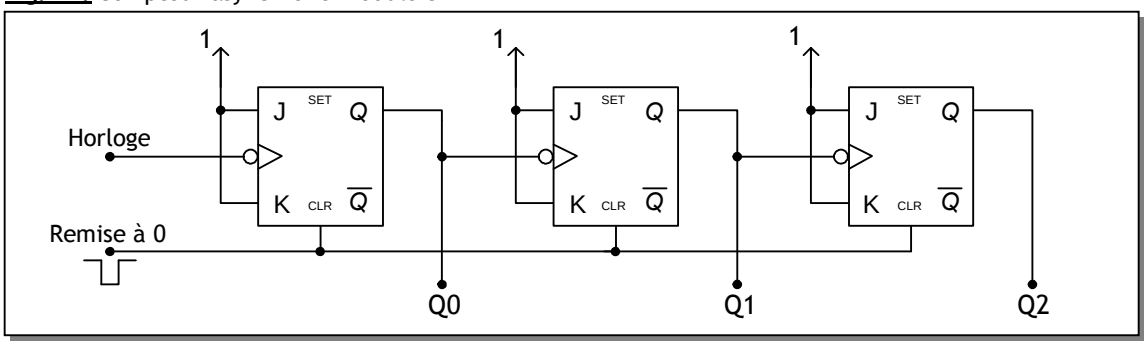
Un compteur est un ensemble de n bascules interconnectées par des portes logiques. Ils peuvent décrire, au rythme d'un signal de commande appelé horloge, une suite d'états binaires. Il ne peut y avoir au maximum que 2n combinaisons et le nombre total N des combinaisons successives est appelé le modulo du compteur. Les compteurs binaires peuvent être classés en deux catégories :

- Les compteurs asynchrones ;
- Les compteurs synchrones ;

2.1. Compteur asynchrone modulo N = 2^n :

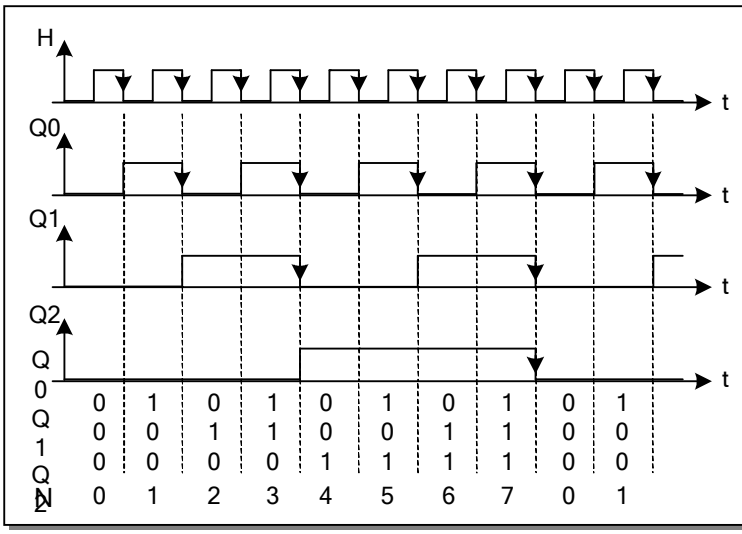
Ce type de compteur est constitué de n bascules JK fonctionnant en mode T (Toggle) : J=K=1. Ces bascules sont montées en cascade c'est-à-dire le signal d'horloge commande uniquement la première bascule tandis que pour chacune des autres bascules le signal d'horloge est fourni par la sortie de la bascule de rang immédiatement inférieur. Pour bien comprendre le principe, réalisons un compteur modulo 8 permettant de compter de 0 à 7 comme le montre la figure 10 :

Fig. 10: Compteur asynchrone module 8



Les chronogrammes et al table de vérité d'un tel compteur sont donnés à la figure 11 :

Fig. 11: Chronogrammes et table de vérité d'un compteur asynchrone module 8



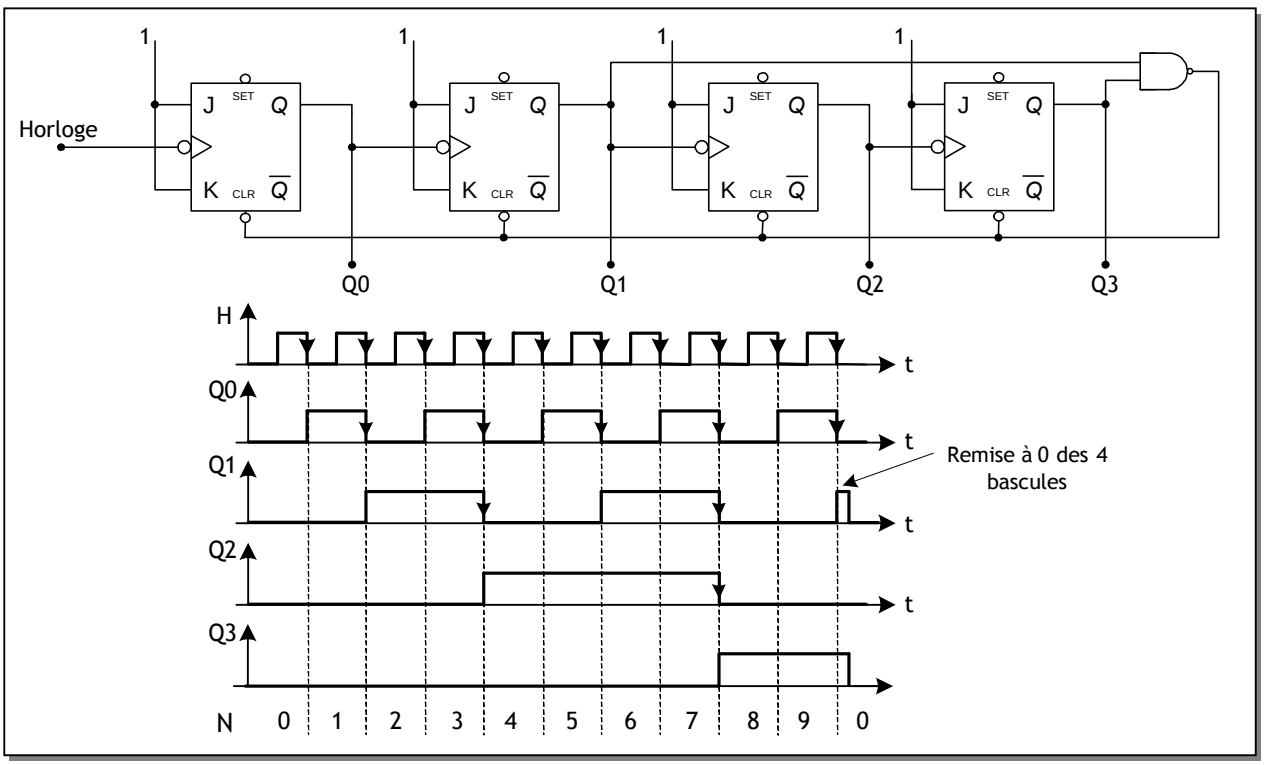
H	Q3(MSB)	Q2	Q1(LSB)
...	0	0	0
↓	0	0	1
↓	0	1	0
↓	0	1	1
↓	1	0	0
↓	1	0	1
↓	1	1	0
↓	1	1	1
↓	0	0	0

2.2. Compteur asynchrone modulo N ≠ 2n :

Pour ce type général de compteur qui compte de 0 à N-1, on va étudier l'exemple d'un compteur asynchrone modulo 10 (0 à 9). Pour le réaliser, il y a deux étapes :

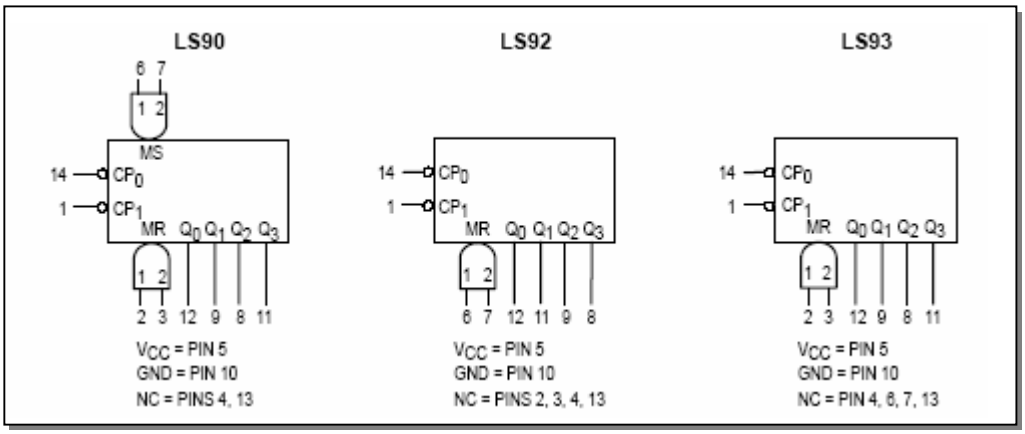
- On cherche d'abord la puissance de 2 immédiatement supérieure à N et qui est pour notre compteur $2^4 = 16$. L'exposant de cette puissance de 2 donne le nombre de bascules JK à monter en cascade, 4 pour notre exemple ;
- On détecte ensuite l'état N qui remettra le compteur à 0 et qui est pour notre compteur $10 = (1010)_2$. On relie les sorties Q = 1 (Q1 et Q3) pour N aux entrées d'une porte NAND dont la sortie commandera l'entrée CLR de chaque bascule (figure 12).

Fig. 12: Compteur asynchrone modulo 10



Les compteur 7490 (modulo 10), 7492 (modulo 12) et 7493 (modulo 16) sont des compteurs asynchrones (figure 13) composés de 4 bascules dont les connexions internes varient suivant le type du compteur :

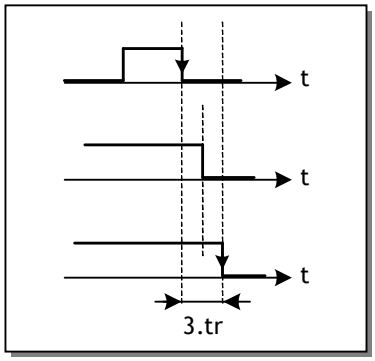
Fig. 13: Exemples de compteur asynchrones



2.3. Compteur synchrone modulo N :

L'inconvénient du compteur asynchrone c'est le temps de réponse de chaque bascule. Ainsi, le signal d'horloge ne parvient pas simultanément sur toutes les bascules. Ceci a pour conséquence de provoquer des états transitoires qui peuvent être indésirables.

Fig. 14: Temps de réponse



Supposons un temps de réponse tr identique pour toutes les bascules et considérons la chronologie du passage d'un compteur asynchrone 3 bits de 011 à 100 dans la figure ci-contre. Nous constatons que le compteur passe par les états transitoires 011 et 001 qui sont faux en plus d'un temps de propagation qui a triplé.

Dans un compteur synchrone toutes les bascules reçoivent en parallèle le même signal d'horloge. Pour faire décrire au compteur une séquence déterminée il faut à chaque impulsion d'horloge définir les entrées synchrones J et K.

Pour cela on utilise la table de transition de la bascule J-K ainsi que la table de vérité décrivant la séquence du compteur. Prenons l'exemple d'un compteur modulo 8 :

Q3	*Q2*	*Q1*
0	0	0
0	0	1
0	1	0
0	1	1
1	0	0
1	0	1
1	1	0
1	1	1

Q_n	Q_{n+1}	*J*	*K*
0	0	0	X
0	1	1	X
1	0	X	1
1	1	X	0

Table d'excitation

La résolution du problème consiste à chercher les équations des entrées J et K de chaque bascule à l'aide de la table des états recherchés :

J_0

Q2 \ Q1 Q0	00	01	11	10
0	**1**	**x**	**X**	**1**
1	**1**	**X**	**X**	**1**

K_0

Q2 \ Q1 Q0	00	01	11	10
0	**X**	**1**	**1**	**X**
1	**X**	**1**	**1**	**X**

$J_0 = K_0 = 1$

J_1

Q2 \ Q1 Q0	00	01	11	10
0	0	**1**	**X**	X
1	0	**1**	**X**	X

K_1

Q2 \ Q1 Q0	00	01	11	10
0	X	**X**	**1**	0
1	X	**X**	**1**	0

$J1 = K1 = Q0$

J_2

Q2 \ Q1 Q0	00	01	11	10
0	0	0	**1**	0
1	X	X	**X**	X

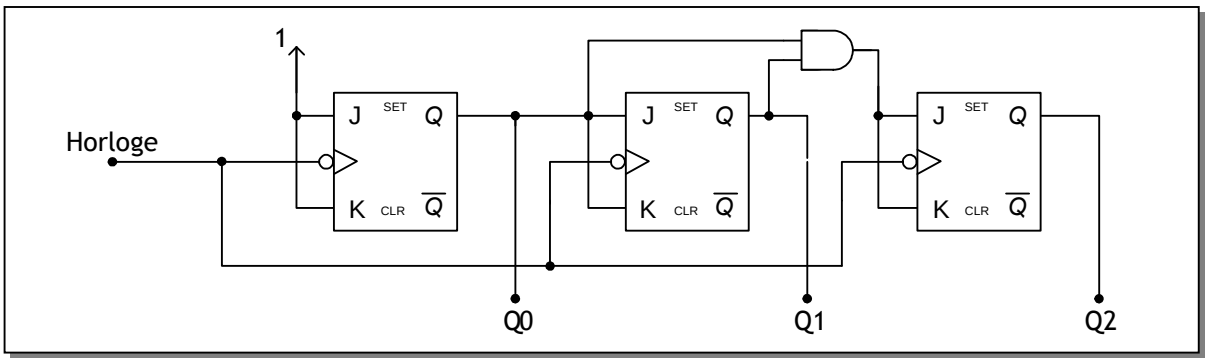
K_2

Q2 \ Q1 Q0	00	01	11	10
0	X	X	**X**	X
1	0	0	**1**	0

$J2 = K2 = Q0.Q1$

Le montage du compteur synchrone 3 bits est donné à la figure 15 :

Fig. 15: Compteur synchrone modulo 8

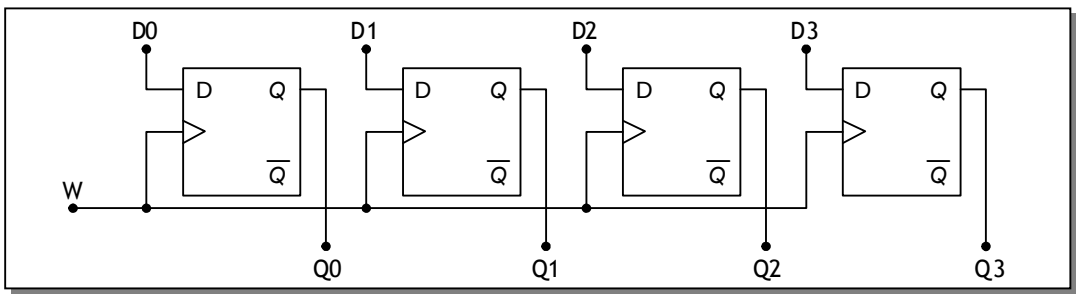


3. LES REGISTRES :

3.1. Présentation :

Un registre est constitué d'un assemblage de n bascules D permettant la mémorisation temporaire de n bits avec ou sans décalage. L'information est emmagasinée sur un signal de commande et ensuite conservée et disponible en lecture. La figure 16 donne un exemple de registre de mémorisation 4 bits avec le signal d'écriture W (Write) qui commande la mémorisation des données D0, D1, D2 et D3:

Fig. 16: Registre de mémorisation 4bits à lecture / écriture parallèles



3.2. Les registres à décalage :

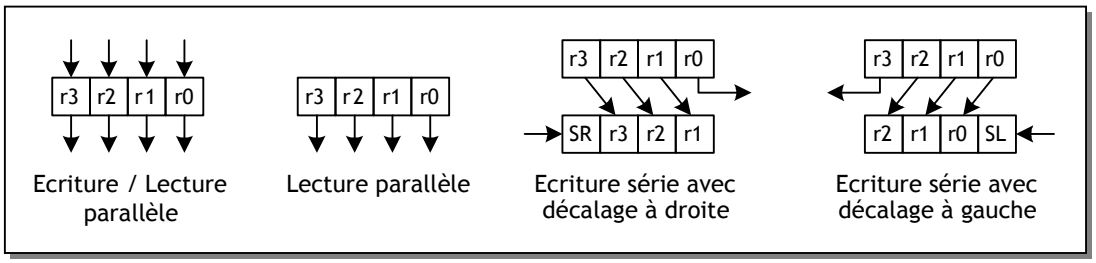
Dans un registre à décalage les bascules sont interconnectées de façon à ce que l'état logique de la bascule de rang i puisse être transmis à la bascule de rang i+1 (ou i-1) quand un signal d'horloge est appliqué à l'ensemble des bascules.

L'information peut être chargée de deux manières :

- Entrée parallèle : comme dans le cas d'un registre de mémorisation ;
- Entrée série : l'information est présentée séquentiellement bit après bit à l'entrée de la première bascule. Le décalage peut alors être vers la gauche ou la droite.

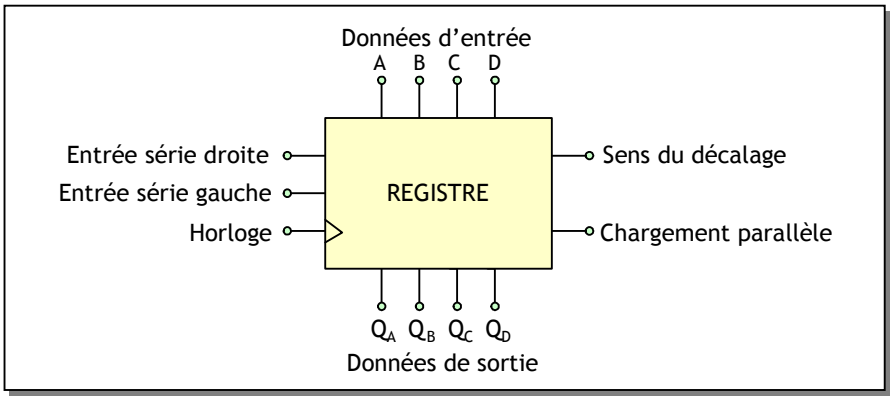
De même, l'information peut être lue en série ou en parallèle. La figure 17 résume les modes de fonctionnement d'un registre à décalage :

Fig. 17: Modes de fonctionnement d'un registre à décalage



Un registre à décalage universel aura donc la structure de la figure 18 :

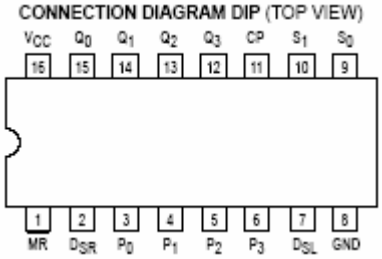
Fig. 18: Structure d'un registre à décalage universel



Parmi les registres universels, on trouve le 74194 qui est un registre à chargement parallèle ou série, avec la possibilité d'un déplacement de l'information vers la droite (QA vers QD) ou la gauche (QD vers QA). La description et le schéma de brochage sont donnés à la figure 19 :

Fig. 19: Brochage et table de fonctionnement du registre universel 74LS194

CONNECTION DIAGRAM DIP (TOP VIEW)



MODE SELECT — TRUTH TABLE

OPERATING MODE	INPUTS						OUTPUTS			
	MR	S_1	S_0	D_{SR}	D_{SL}	P_n	Q_0	Q_1	Q_2	Q_3
Reset	L	X	X	X	X	X	L	L	L	L
Hold	H	l	l	X	X	X	q_0	q_1	q_2	q_3
Shift Left	H	h	l	X	l	X	q_1	q_2	q_3	L
	H	h	l	X	h	X	q_1	q_2	q_3	H
Shift Right	H	l	h	l	X	X	L	q_0	q_1	q_2
	H	l	h	h	X	X	H	q_0	q_1	q_2
Parallel Load	H	h	h	X	X	P_n	P_0	P_1	P_2	P_3

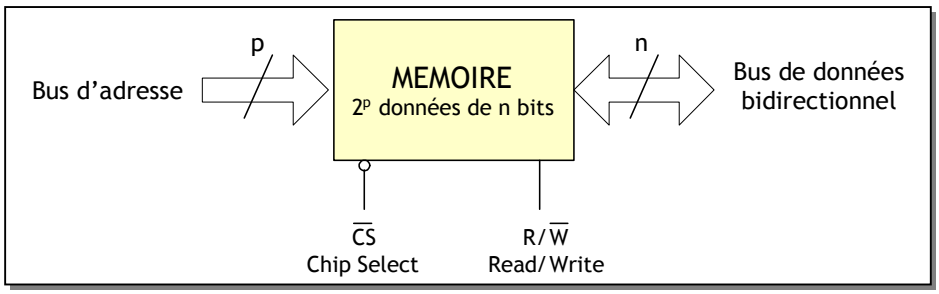
4. LES MEMOIRES :

4.1. Introduction :

Les systèmes modernes sont souvent à microprocesseur de tels systèmes exigent une capacité mémoire importante pour le stockage des données. Les mémoires électroniques ou à semi-conducteurs sont les éléments qui répondent à ce besoin.

Une mémoire est un dispositif capable d'emmagasiner puis de restituer une information. L'unité d'information (bit, octet, etc.) s'appelle «point mémoire » ou «cellule».

Fig. 19: Schéma fonctionnel d'une mémoire



- L'adresse fournie par le bus d'adresses est le mot binaire de p bits qui permet de localiser la donnée ;
- La donnée de n bits entre (écriture) et sort (lecture) par le bus de données qui est bidirectionnel : deux sens possibles, en liaison avec le signal R/W ;
- La mémoire peut stocker 2p données de n bits chacune ;
- Le signal CS permet la sélection du circuit ou le mettre en haute impédance ; cette possibilité permet, comme on le verra, l'extension de la capacité mémoire d'un système.

On peut donc utiliser une mémoire soit en :

- **lecture :**
  - ✓ Appliquer le mot adresse sur le bus d'adresse ;
  - ✓ Sélectionner le boîtier mémoire en appliquant un niveau logique bas sur la ligne *CS* ;
  - ✓ Sélectionner le mode lecture en appliquant un niveau logique haut sur la ligne R/W ;
- **écriture :**
  - ✓ Appliquer le mot d'adresse sur le bus d'adresse ;
  - ✓ Le mot de donnée sur le bus de données ;
  - ✓ Sélectionner le boîtier mémoire en appliquant un niveau logique bas sur la ligne CS ;
  - ✓ Sélectionner le mode écriture en appliquant un niveau logique bas sur la ligne R/W ;

4.2. Caractéristiques des mémoires :

- ***La capacité*** : c'est la quantité d'information qui peut être stockée dans la mémoire. Elle s'exprime en bits ou en mots de n bits. Par exemple :

  64b, 4Kb, 8Ko (o : octet ou byte) avec 1o = 8b ; 1K = 2^{10} = 1024 ; 1M = 2^{20} = 1048576

- ***L'organisation*** : elle définit le nombre de mots et la longueur de chaque mot. Par exemple :
  - ✓ Une mémoire de 64Kx 1 est constituée de 65536 mots de 1 bit. Sa capacité est donc de 64Kb (8Ko) ;
  - ✓ Une mémoire de 8Kx8 contient 8192 mots de 8 bits. Sa capacité est de 64Kb (8Ko) ;
- ***Le temps d'accès*** : c'est le temps qui s'écoule entre une demande d'information et le moment où elle est effectivement disponible.

4.3. Les différents types de mémoires

Les mémoires sont classées suivant deux familles :

- Mémoires mortes (**ROM** pour Read Only Memory) : mémoire à lecture seule;
- Mémoires vives (**RAM** pour Random Access Memory) : mémoire à lecture et écriture.

4.3.1. Les mémoires mortes :

Les ROM sont utilisées pour stocker des informations figées telles que des programmes fixes dans des machines programmées ou les tables de conversion de données.
Le contenu est fixé à la construction ou par l'utilisateur et la disparition de l'alimentation électrique n'altère pas le contenu.

4.3.2. Les mémoires vives

Dès qu'un système doit conserver temporairement des informations, la RAM trouve sa place. En informatique, elles sont largement mises en oeuvre en quantités importantes (plus de 16 Mo en microinformatique et plusieurs centaines de méga octets en mini-informatique).

4.3.3. Les mémoires programmables et effaçables par l'utilisateur

Les mémoires programmables sont intermédiaires entre les RAM et les ROM. Leur contenu peut être défini par l'utilisateur et subsister sans alimentation électrique.
On en rencontre de différentes familles :

- ✓ Les **PROM** (Programmable ROM) : sont composées de fusibles que l'on peut détruire une seule fois ;
- ✓ Les **EPROM** (Erasable PROM) : ce sont des mémoires effaçables par ultraviolet et programmables électriquement ;
- ✓ Les **EEPROM** (Electrical Erasable PROM) : ce sont des mémoires effaçables et programmables électriquement .

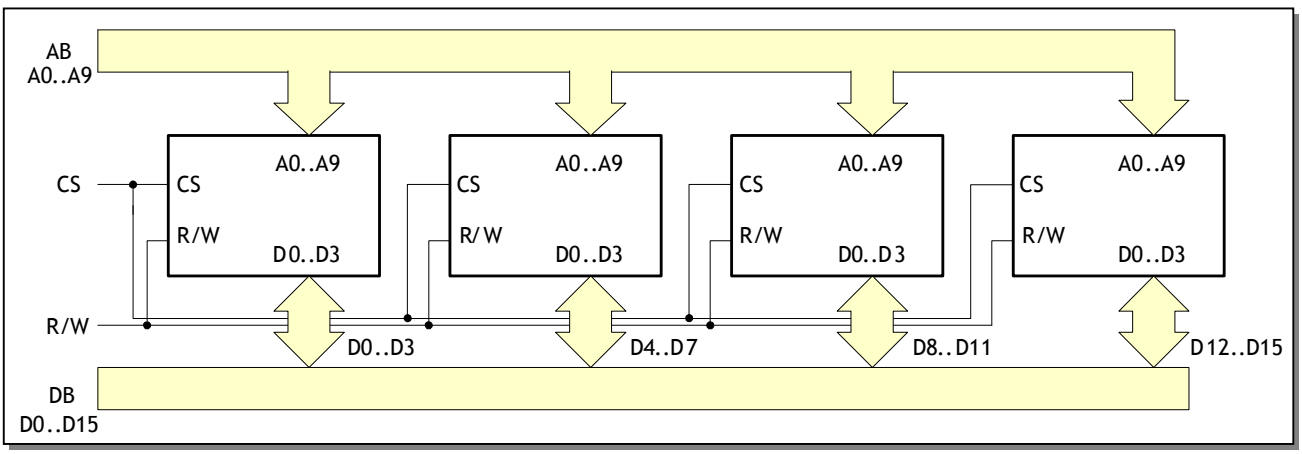
4.4. Extension de capacité :

Il est courant dans un système microinformatique de grouper plusieurs circuits pour augmenter la capacité (nombre des mots et/ou longueur des mots). Par exemple, à l'aide de 4 boîtiers mémoires de 1Kx4bits, on peut réaliser les mémoires suivantes : 1Kx16bits, 4Kx4bits, 2Kx8bits.

Le schéma de la figure 20 réalise une mémoire de 1Kx16bits à partir d'une mémoire élémentaire de 1Kx4bits :

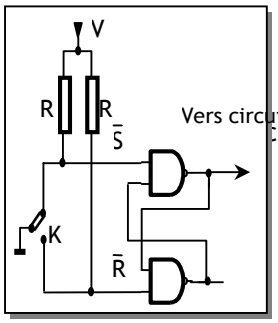
- Nécessité de 4 boîtiers ;
- Nécessité de 10 bits d'adresses A_0 à A_9 ; (nombre de mots = $2^{\text{nombre de bits d'adresse}}$)
- Nécessité de 16 bits de données D_0 à D_{15} ;

Fig. 20: Extension de format d'une mémoire

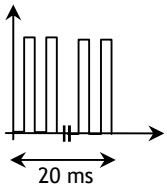


EXERCICES RESOLUS

EXERCICE N° 1 :



Lorsqu'un circuit actif sur un front est attaqué par un interrupteur directement, les rebondissements de ce dernier provoque plusieurs front d'une durée approximative de 20 ms ; ce phénomène cause un dysfonctionnement du circuit commandé. La solution à ce problème est un circuit anti-rebond qu'on peut réaliser de plusieurs façons ; ici on étudie la réalisation à base d'une bascule SR. Analyser alors le fonctionnement d'un tel circuit.



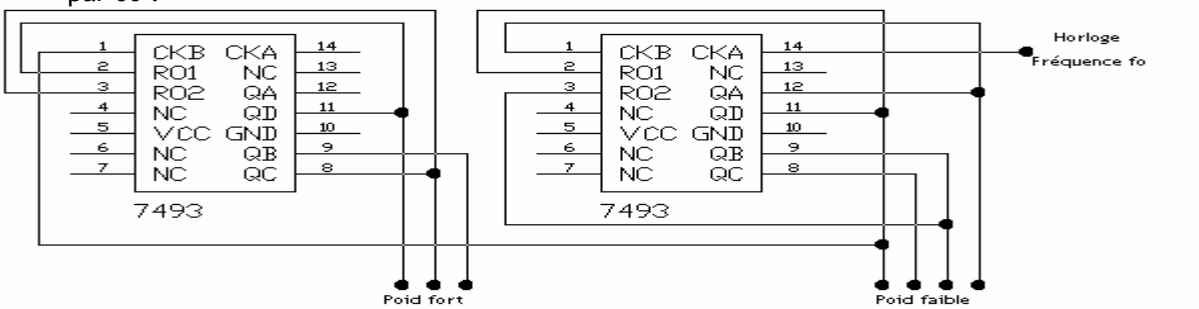
EXERCICE N° 2 :

On désire réaliser un compteur asynchrone modulo 13 à base du circuit intégré 74LS76 :

2.1. Combien de circuits intégrés 74LS76 doit-on utiliser ?
2.2. Proposer alors un schéma pour la réalisation d'un tel compteur en prévoyant un signal INIT pour l'initialisation du compteur ?

EXERCICE N° 3 : Etude du compteur binaire 74LS93

3.1. De quel type de compteur s'agit-t –il ?
3.2. Montrer comment raccorder les bornes du compteur binaire 74LS93 pour obtenir un compteur modulo 16 ?
3.3. Soit le montage suivant permettant de réaliser un compteur modulo 60 ou diviseur de fréquence par 60 :



Expliquez le fonctionnement du montage en commençant par indiquer le modulo de chaque compteur 74LS93 ?

EXERCICE N° 4 : Etude du registre à décalage universel 74LS194

On désire utiliser le registre à décalage universel 74LS194 pour un chargement parallèle des données et une lecture série de ces données avec décalage vers la droite :

4.1. Remplir le tableau de fonctionnement ci-dessous pour permettre un chargement parallèle du mot binaire 1011 dans le registre 74LS194 ?
4.2. En supposant que le chargement du mot binaire a été effectué, donnez le câblage du 74LS194 pour réaliser un décalage rotatif de la gauche vers la droite ?

ENTREES								SORTIES			
Clear	Mode		Clock	Parallèle				Q_D	Q_C	Q_B	Q_A
	S1	S0		D	C	B	A				

CORRIGES :

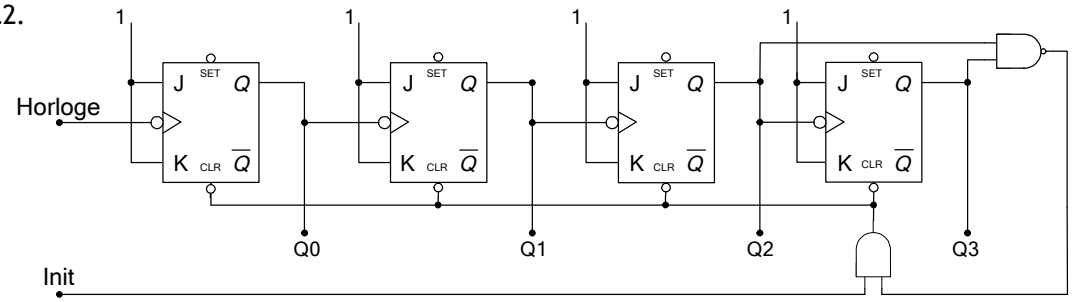
EXERCICE N° 1 :

* Au repos, l'interrupteur K est en position 1, S = 0 et R = 1, alors Q = 1 ;
* Si on passe à la position 2, S = 1 et R = 0, alors Q = 0 ;
* Mais l'interrupteur rebond entre cette position et la position intermédiaire : où S = 1 et R =1, ce qui correspond à l'état de mémoire de la bascule SR (Q = 0) ;
* De même, si on revient à la position 1 ; A la sortie, on n'a pas de rebondissemets.

EXERCICE N° 2 :

1.1. On aura besoin de 4 bascules JK donc de 2 circuits 74LS76.

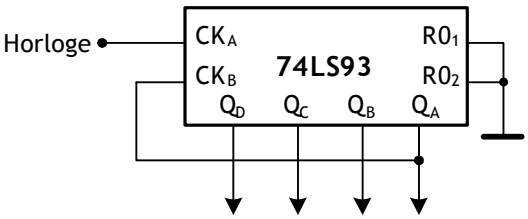
1.2.



EXERCICE N° 3 :

3.1. C'est un compteur asynchrone binaire 4 bits.

3.2.



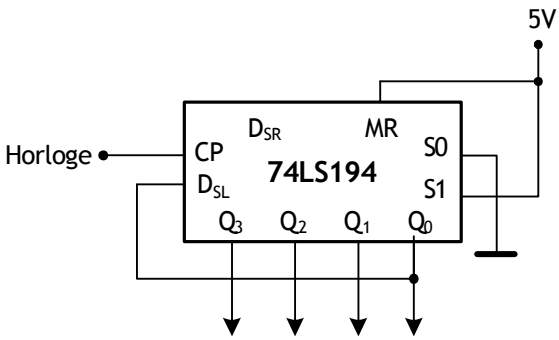
3.3. Le 1er compteur 74LS93 est un modulo 10 tandis que le 2éme est un modulo 6. Les deux compteurs sont montés en cascade puisque la dernière sortie de 1er compteur attaque l'horloge du deuxième.

EXERCICE N° 4 :

4.1.

ENTREES							SORTIES			
Clear	Mode		Parallèle				Q_D	Q_C	Q_B	Q_A
	S1	S0	D	C	B	A				
1	1	1	1	0	1	1	1	0	1	1

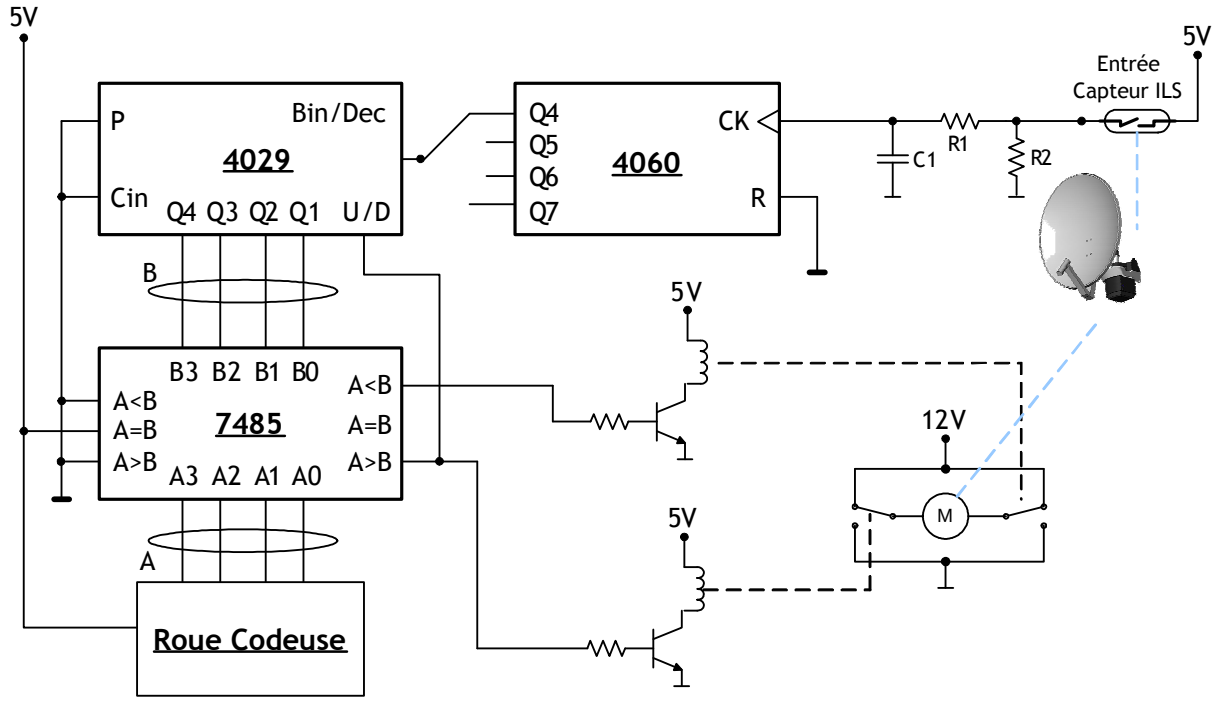
4.2.



E X E R C I C E N O N R E S O L U

Le montage proposé permet un asservissement de position pour un positionneur d'antenne parabolique :
Le montage suivant représente une solution câblée pour la chaîne d'information du système "Positionneur d'antenne parabolique". Le fonctionnement d'une telle structure est comme suit :

- La position désirée (consigne) est fournie par une roue codeuse ;
- Quand le moteur tourne vers le "West" ou vers l'"East", le capteur ILS se fermant et s'ouvrant régulièrement, donne des impulsions au compteur (le 4060 et le 4029) qui compte ces dernières ; de ce fait, on détecte la position de l'antenne ;
- Le comparateur (le 7485) compare la consigne à la position réelle et en fonction du résultat fait tourner le moteur vers le "W" ou vers le "E" ou l'arrête.



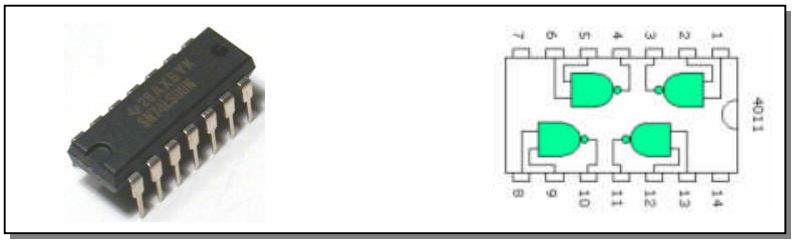
1. Pour les 2 sens de rotation du moteur, donner la fonction du compteur 4029 (comptage ou décomptage), l'état des 2 relais ainsi que le schéma correspondant à la commande du moteur.
2. 16 impulsions du capteur ILS correspondent approximativement à un déplacement de 0.1 mm, de la tige du vérin électrique (moteur + système vis-écrou). Quel est le rôle du compteur 4060 ?
3. Le relais est un circuit selfique. Qu'est ce qui manque alors dans ce circuit de commande ?

FAMILLES LOGIQUES TTL ET CMOS

1. NOTION DE FAMILLE DE CIRCUIT LOGIQUE

Un circuit logique se présente sous forme de circuit intégré qui permet de regrouper dans un même boîtier un maximum de composants électroniques dont le plus important est le transistor.

Fig. 1 : Exemple de circuit logique intégré



Les circuits intégrés logiques sont classés suivant leur technologie de fabrication en plusieurs familles logiques. Chaque famille logique a pour point commun la technologie employée.

Dans ce chapitre, on étudiera les familles les plus populaires actuellement, à savoir :

- ***La famille TTL (Transistor Transistor Logic)*** : utilise une technologie à base de transistors bipolaires ;
- ***La famille CMOS (Complementary Metal Oxide Semiconductor)*** : utilise une technologie a base de transistor MOS.

Chaque famille logique est caractérisée par des paramètres électriques comme l'alimentation et la consommation, et des performances dynamiques comme le temps de propagation.

2. LES VARIANTES TECHNOLOGIQUES DES FAMILLES LOGIQUES TTL ET CMOS

2.1. Les variantes technologiques de la famille TTL :

- ***La série TTL standard*** : Série standard qui est peu rapide avec une consommation élevée ;
- ***La série TTL L*** : Série à faible consommation (Low Power) mais au détriment de la rapidité ;
- ***La série TTL H***: Série rapide (High speed) mais au détriment de la consommation ;
- ***La série TTL S*** : Série Schottky qui est 2 fois plus rapide que la série H pour la même consommation ;
- ***La série TTL LS*** : Série qui constitue un compromis entre la série TTL L et la série TTL S ;
- ***Les séries TTL AS (Advanced Schottky), ALS (Advanced Low power Schottky)*** : Séries dérivées des technologies présentées précédemment. Les technologies avancées de TTL mettent en oeuvre les progrès récent en matière de circuits intégrés bipolaires.

2.2. Les variantes technologiques de la famille CMOS :

- ***La série 4000*** : Série classique de base.
- ***Les séries AC et ACT*** : La série AC (Advanced Cmos) est l'évolution de la série de base. Le suffixe T indique la compatibilité des entrées et sorties avec les séries TTL.
- ***Séries HC, HCT, AHC et AHCT*** : Les séries disposant du suffixe H (*High speed*) sont fondées sur la technologie CMOS rapide. Elles existent dans les différentes déclinaisons suivant qu'elles sont avancées (A), compatibles TTL (T) ou combinant ces caractéristiques.

3. LA STRUCTURE DE BASE DES FAMILLES LOGIQUES TTL ET CMOS

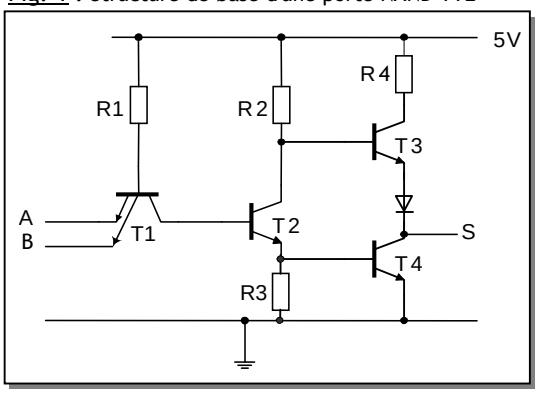
3.1. La structure TTL :

Le montage de la figure 1 est la structure de base d'une porte NAND TTL à base de transistors bipolaires ; l'analyse suivante montrera qu'il s'agit bien d'une porte "NAND".

- Lorsque les deux entrées a et b sont au niveau haut (5V), le transistor T1 est bloqué et le transistor T2 est saturé ce qui bloque T3 et sature T4 ; S est alors au niveau logique bas (0V) si la porte est chargée normalement (par exemple par une autre porte TTL);

- Si l'une des entrées ou les deux entrées sont au niveau bas (0V), le transistor T1 est conducteur, ce qui bloque T2 et T4 ; T3 peut alors conduire et assure à la sortie S un niveau logique haut (voisin de 5V) si la porte est chargée normalement (par exemple par une autre porte TTL) ;

Fig. 1 : Structure de base d'une porte NAND TTL



3.2. Structure CMOS :

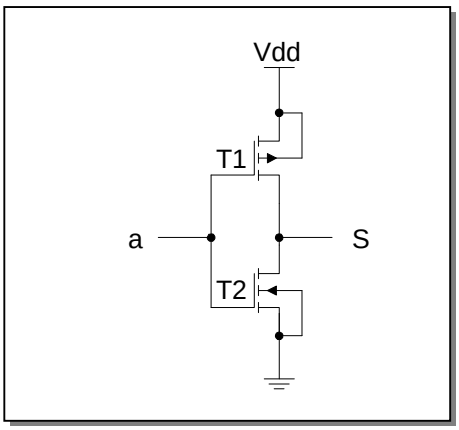
Le montage de la figure 2 représente la structure de base d'une porte INVERSEUSE CMOS à base de deux transistors MOS complémentaires :

- Si on applique un niveau bas à l'entrée, le transistor T2 (MOS N) est bloqué car sa tension grille est nulle. Par contre le transistor T1 (MOS P) est conducteur car sa tension grille source est égale à -5V : le sortie S est au niveau logique 1 ;

- Si on applique un niveau haut à l'entrée, le transistor T1 est bloqué car sa tension grille source est nulle. Le transistor T2 est conducteur car sa tension grille source est égale à 5V : la sortie S est au niveau logique 0 ;

- L'ensemble réalise une porte inverseuse.

Fig. 2 : Structure de base d'une porte CMOS



4. LES PARAMETRES ELECTRIQUES DES CIRCUITS LOGIQUES

4.1. Tension d'alimentation :

- Famille logique TTL : L'alimentation doit être fixe et égale à 5V avec une tolérance de ± 5%.

- Famille logique CMOS : Le choix de la tension d'alimentation est plus large de 3V à 18V.

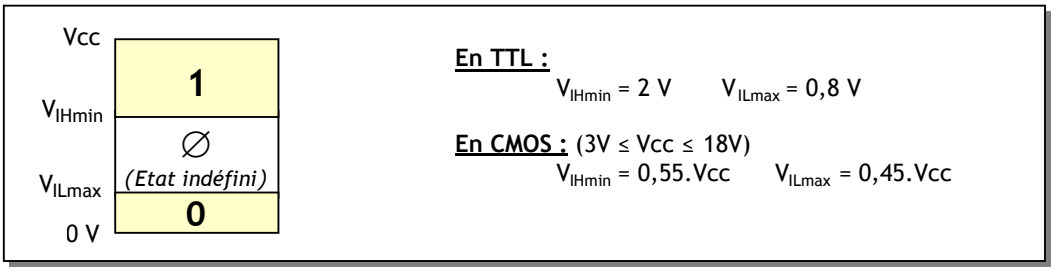
4.2. Les niveaux logiques :

Pour une famille donnée, les niveaux logiques « 0 » ou "L" (Low) et « 1 » ou "H" (High) ne correspondent pas à une tension précise, mais à une certaine « plage » de tension.

La terminologie utilisée pour les valeurs de la tension en entrée (Input):

- **VIHmin** : Tension minimale en entrée qui assure le niveau logique haut.
- **VILmax** : Tension maximale en entrée qui assure le niveau logique bas.

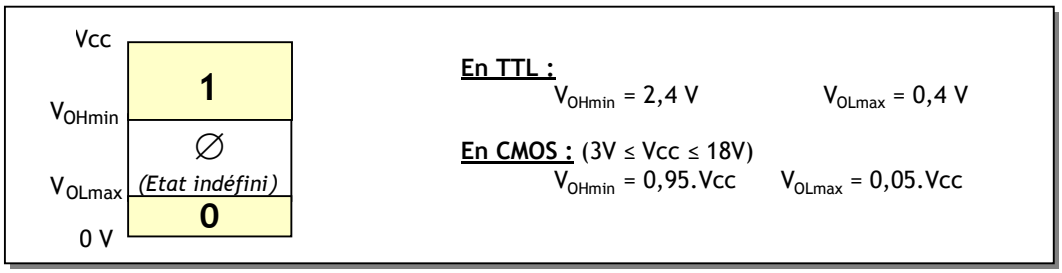
Fig. 3 : Niveaux logiques d'entrées



La terminologie utilisée pour les valeurs de la tension de sortie (Output) :

- **VOHmin** : Tension minimale de sortie à l'état logique haut.
- **V0Lmax** : Tension maximale de sortie à l'état logique bas.

Fig. 4 : Niveaux logiques de sorties



4.4. Consommation des circuits logiques

La consommation d'un circuit logique est la puissance demandée par son boîtier au circuit d'alimentation. Elle doit être la plus faible possible et elle est nécessaire pour dimensionner l'alimentation des circuits logiques.

Le constructeur indique une puissance consommée moyenne liée aux niveaux de sortie des circuits logiques. Ainsi pour un opérateur NAND TTL, elle est de 50mW

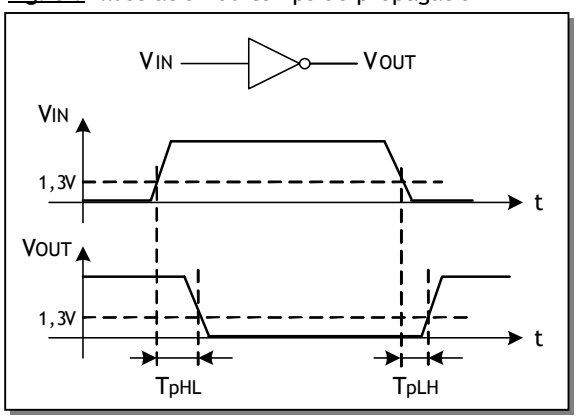
5. LES PERFORMANCES DYNAMIQUES DES CIRCUITS LOGIQUES

Dans une porte logique, les grandeurs sont transmises avec un retard caractéristique de la porte : c'est le temps de propagation de l'information dans la porte. On distingue alors la transition haut-bas (front descendant) ou bas-haut (front montant) :

- **TpHL** : temps de propagation du signal logique lorsque la sortie passe de l'état haut à l'état bas (*Propagation Time High to Low*).
- **TpLH** : temps de propagation du signal logique lorsque la sortie passe de l'état bas à l'état haut (*Propagation Time Low to High*).

Pour assurer la mesure de ces durées, une référence de tension est fixée par les constructeurs (1,3 V en TTL) pour le début et la fin de la propagation comme le montre la figure 5 :

Fig. 5 : Illustration du temps de propagation



6. PERFORMANCES COMPAREES DES DIFFERENTES FAMILLES TTL ET CMOS

	TTL	TTL-LS	C-MOS	NC-MOS
Temps de propagation tpd (ns)	10	10	40	10
Tension d'alimentation (V)	+5V à ± 5%	+5V à ± 5%	+3V à +15V	+3V à +15V
Puissance consommée Pc (mW)	10	2	0,3	0,3
Fréquence maximale (MHz)	30	30	10	30
Nombres d'entrées recommandées	10	20	50 mini	50 mini

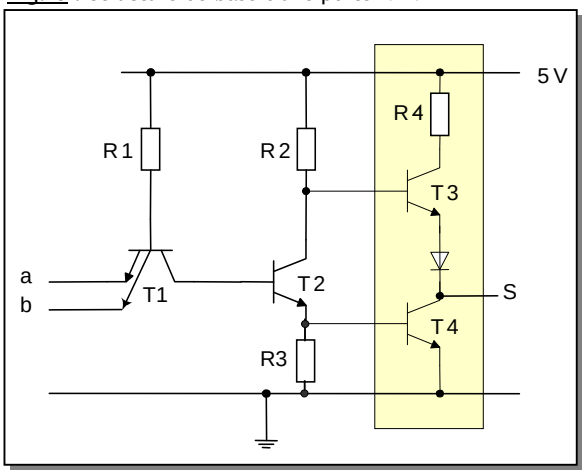
7. LES DIFFERENTS ETAGES DE SORTIE

7.1. Etage de sortie Totem pole :

L'étage de sortie, appelé totem pole ajoute un étage à transistors pour améliorer les commutations de la sortie et la stabilité des niveaux logiques.

La figure 6 montre la structure d'une porte NAND avec sortie totem pole :

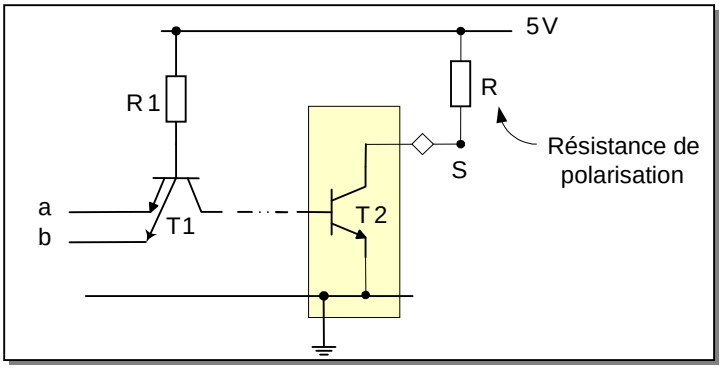
Fig. 6 : Structure de base d'une porte NAND TTL



7.2. Etage de sortie à collecteur ou drain ouvert :

L'étage de sortie à collecteur ouvert ou drain ouvert est utilisé lorsque la charge nécessite une tension d'alimentation que ne peut lui fournir la porte logique. Dans cette configuration, la sortie est le collecteur ou le drain du dernier transistor dont il faut terminer la polarisation. Ce type de sortie est identifié par le signe montré à la figure suivante qui représente une porte NAND à sortie collecteur ouvert :

Fig. 7 : Porte NAND TTL avec sortie collecteur ouvert



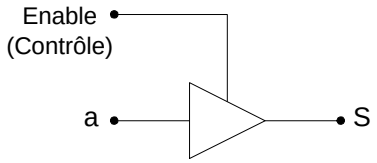
7.3. Etage de sortie trois états :

Dans certaines applications, il s'avère nécessaire de déconnecter électriquement la sortie d'une porte logique pour l'isoler du montage. Il faut donc que la sortie soit comme un circuit ouvert vis à vis du reste du montage. Un nouvel état apparaît en supplément du niveau haut et du bas : l'état de haute impédance (HiZ) où l'impédance ou la résistance de sortie devient très grande, voire infinie.

Pour mettre en oeuvre une telle porte, il faut une entrée supplémentaire sélectionnant l'état haute impédance ou troisième état.

La figure 8 montre le symbole d'un opérateur OUI 3 états avec sa table de vérité :

Fig. 8 : Porte OUI 3 états



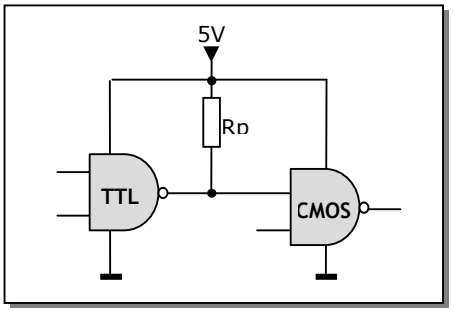
Contrôle	Entrée	Sortie
Enable	a	S
1	0	0
1	1	1
0	X	HiZ

8. INTERFACAGE DES CIRCUITS LOGIQUES

8.1. Interface CMOS/TTL et TTL/CMOS (Figure 9) :

- Une sortie CMOS peut commander sans problème une entrée TTL ;
- Par contre, une sortie TTL délivre une tension VOH très proche de VIH(min) de l'entrée CMOS. La solution la plus utilisée pour assurer la compatibilité consiste à utiliser une résistance de rappel à la source Rp comprise entre 1KΩ à 10KΩ.

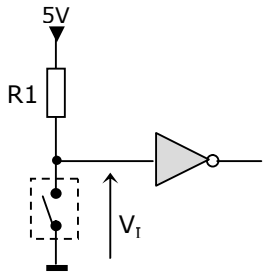
Fig. 9 : Interfaçage entre TTL et CMOS



8.2. Commande d'une entrée logique TTL par un contact :

Souvent, l'entrée d'un circuit logique TTL change d'état en fonction de l'état d'un contact (interrupteur, contact d'un capteur TOR, etc.). Dans ce cas, il faut prévoir un interfaçage pour assurer le bon fonctionnement comme le montre les figures 10 et 11 :

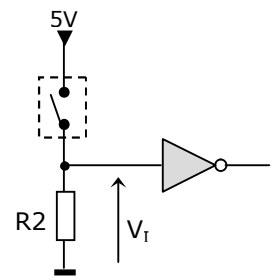
Fig. 10 : Contact commandant un niveau bas



Sans la résistance R1, la fermeture du contact provoque un court-circuit de l'alimentation 5V. La résistance R1 résout ce problème :

- Au repos, le contact est ouvert, R1 assure le '1' logique sur l'entrée de la porte logique ; R1 est typiquement de 10kΩ ;
- lorsque le contact est fermé, l'entrée logique est au niveau "0" et l'alimentation n'est pas court-circuitée.

Fig. 11 : Contact commandant un niveau haut

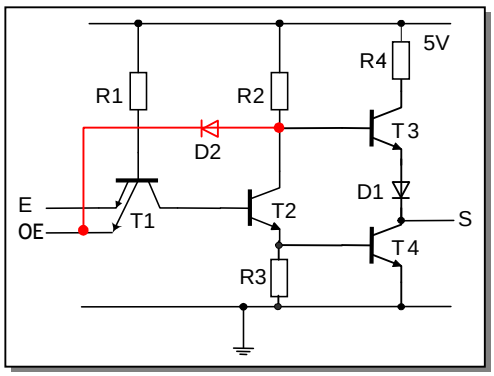


- Au repos, le contact est ouvert, R1 assure le '0' logique sur l'entrée de la porte logique ; R1 est typiquement de 1kΩ ;
- lorsque le contact est fermé, l'entrée logique est au niveau "1" et l'alimentation n'est pas court-circuitée.

EXERCICES RESOLUS

EXERCICE N° 1:

1.1- Donner le schéma d'une porte NAND à 4 entrées.
1.2- Peut-on généraliser le principe sur n entrés ?



EXERCICE N° 2:

Soit le montage ci-contre, d'une porte logique. Analyser le montage et en déduire le rôle de l'entrée OE (Output Enable) et le rôle de la porte ?

CORRIGES :

EXERCICE N° 1 :

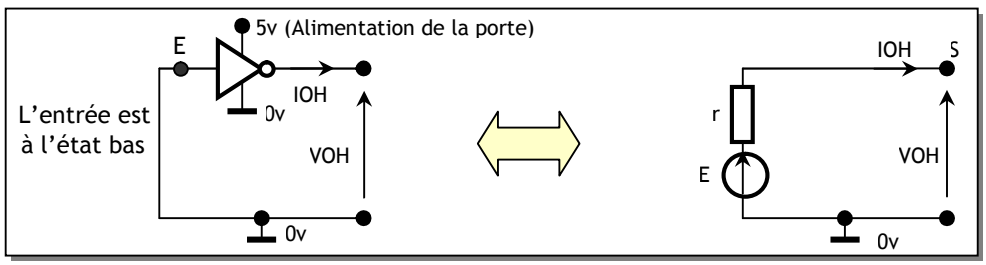
1.1- C'est le même schéma que pour le schéma de base, mais avec un transistor à 4 émetteurs..
1.2- Oui.

EXERCICE N° 2 :

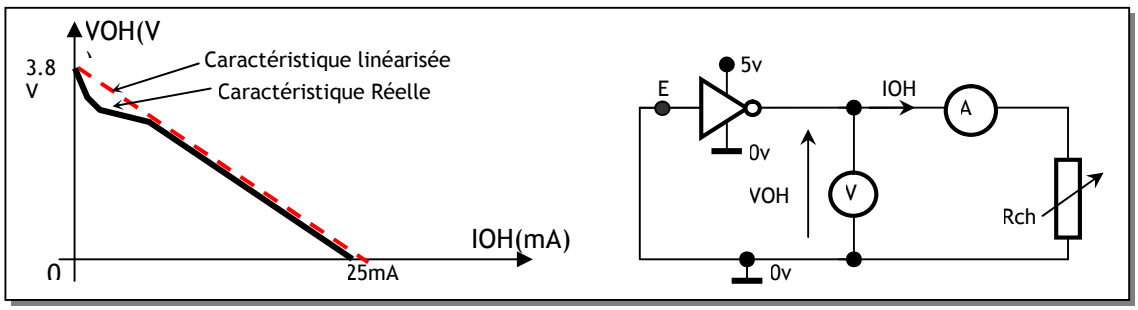
→ Si OE=0 (0V), D2 conduit, ce qui fait conduire T1 bloquant par conséquence T2 et T4. La conduction de D2 bloque T3 et D1. En conclusion, l'étage TOTEM POLE est bloqué, ce qui met la sortie de la porte en haute impédance.
→ Si OE= (5V), D2 est bloqué ; elle n'a aucune influence sur le montage. Par analyse simple, on trouve que la porte est inverseuse ($S = \overline{E}$). En conclusion, c'est une porte inverseuse Tri-State.

EXERCICE NON RESOLU

La sortie d'une porte logique à l'état haut est équivalente à un dipôle actif, comme il est modélisé dans la figure ci-dessous :



On se propose d'étudier sa caractéristique courant-tension VOH = f(IOH). Le relevé d'une telle caractéristique est donné à la figure ci après :



Donner les caractéristiques (E,r) de ce dipôle, en considérant la caractéristique linéaire

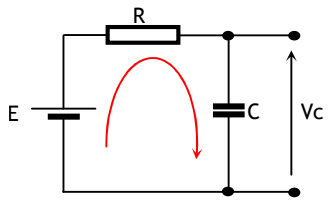
TEMPORISATEURS A BASE DE CIRCUITS INTEGREES

INTRODUCTION

Dans les systèmes numériques, on a souvent besoin q'une action soit effectuée pendant une durée déterminée ; on parle de temporisation. Aussi, on a besoin d'un signal périodique qui synchronise ou cadence les opérations d'un systèmes séquentiel ; on parle de base de temps ou d'horloge (Clock).

1. LE CIRCUIT DE BASE : LE CIRCUIT RC

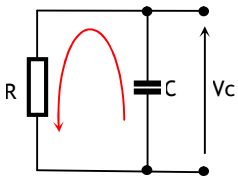
A la base de pas mal de systèmes de temporisation, on trouve un circuit à base d'une résistance R et d'un condensateur C (circuit RC). Ci -dessous, on étudie les aspects de base de ce circuit.



Dans un circuit RC, avec C déchargé (Vc=0V), alimenté par une tension continue Vcc, la tension Vc aux bornes de C augmente ; on dit que C se charge. La loi de variation de Vc est de la forme :

$$Vc = E(1-e^{-t/RC})$$

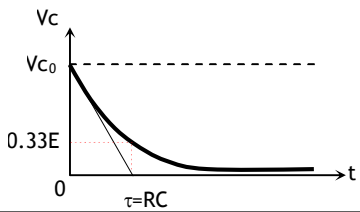
- Pour t=0 → Vc=0V ;
- Pour t→∞ → Vc→E ;
- Pour t=τ → Vc=0.63E.



Dans un circuit RC, avec C déjà chargé (Vc=Vc_0), la tension aux bornes de C diminue ; on dit que C se décharge. la loi de variation de Vc est de la forme :

$$Vc = Vc_0(e^{-t/RC})$$

- Pour t=0 → Vc=Vc0 ;
- Pour t→∞ → Vc→0 ;
- Pour t=τ → Vc=0.33Vc0.



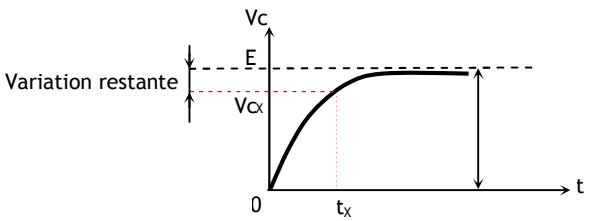
On démontre que pour atteindre une certaine valeur VcX, il faut un certain temps tX, tel que :

$$t_X = RC[\ln(\frac{Varaition\_totale}{Variation\_restante})]$$

où :

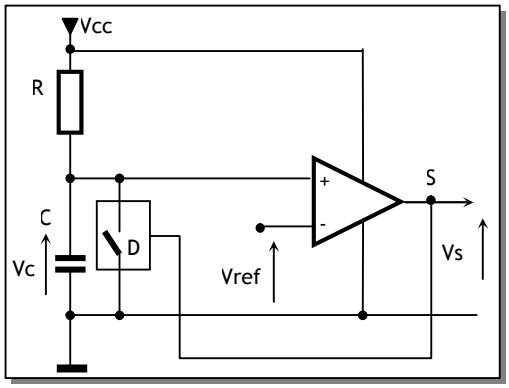
- ln est le logarithme népérien ;
- R est la valeur de la résistance ;
- C est la capacité du condensateur.

$$t_X = RC[\ln(\frac{E}{E - Vc_X})]$$



2. PRINCIPE DE BASE :

Il correspond au schéma ci-contre. Son fonctionnement est comme suit :

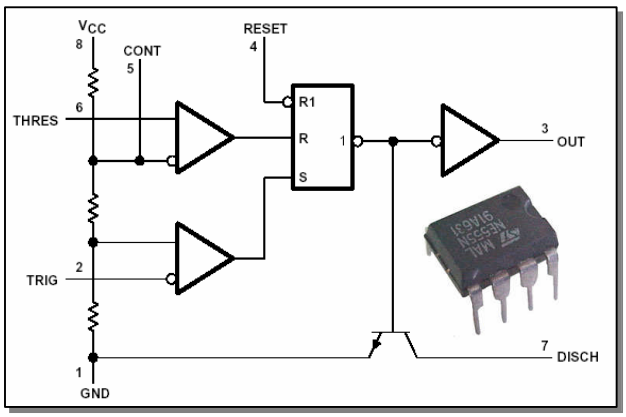


- Initialement, le condensateur C est déchargé et la tension de sortie Vs du comparateur est au niveau bas ce qui ouvre l'interrupteur électronique D ;
- Le condensateur C se charge exponentiellement à travers la résistance R ;
- Vs reste à l'état bas jusqu'à ce que Vc atteigne la tension de la référence Vref ;
- Lorsque Vc atteint Vref, le comparateur bascule à l'état haut provoquant ainsi la fermeture de l'interrupteur ;
- Le condensateur C se décharge rapidement à travers la résistance de l'interrupteur qui est relativement faible par rapport à R.

Un tel principe est utilisé régulièrement dans les systèmes électroniques ; une variété de circuits intégrés le concrétisent (74121 en TTL, le 4047 en CMOS, NE555, etc.). Le plus célèbre de ces circuits est le fameux NE555, qui fera l'objet de l'étude suivante.

3. LE TEMPORISATEUR NE555 :

Introduit en 1972, le NE555 était le premier circuit intégré temporisateur qui représente une sorte de standard en matière de fonctions de temporisation, en particulier le circuit monostable et le circuit astable. Comme l'indique son schéma interne, il est consistitué de :



- 2 comparateurs dont les seuils sont fixés par le pont des 3 réistances R ;
- d'une bascule SR, avec une entrée de forcage à 0 (RESET) ;
- d'un transistor pour la décharge de condensateur externe.

4. MODES DE FONCTIONNEMENT DU NE555

Plusieurs fonctions peuvent être réalisées par ce circuit ; les plus importantes sont le Monostable qui permet d'avoir des temporisations et l'Astable qui permet d'avoir des oscillations libres.

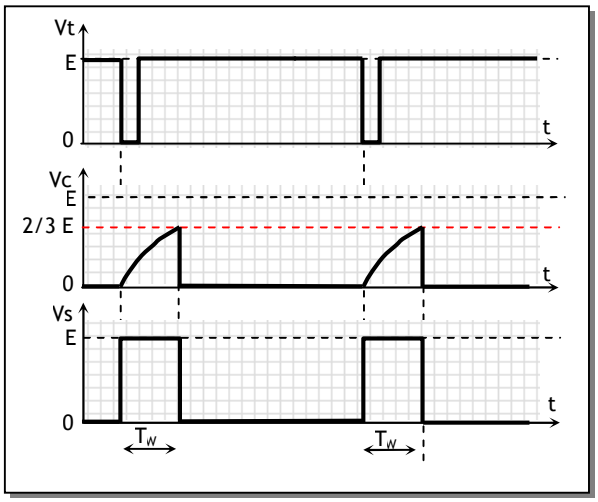
4.1. Le monostable

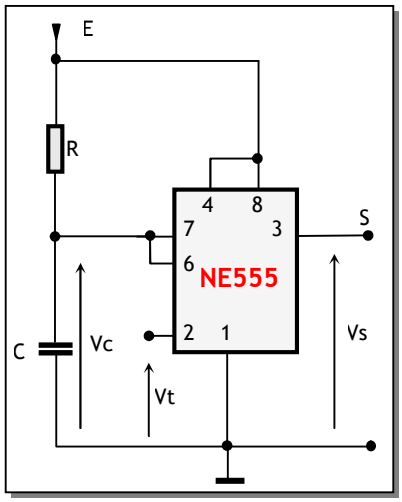
Le schéma à base du NE555 et son fonctionnement sont donnés ci-après :

- On suppose qu'au démarrage, le condensateur C est déchargé et la sortie Vs à 0 ;
- L'entrée Vt est au repos au niveau logique 1 ; alors les variables du montage sont positionnées ainsi :
  - Le entrées de la bascule (S=0) et (R=0) → Etat mémoire de la bascule et Vs reste à 0 ;
  - le transistor est saturé ; c'est l'état stable du monostable.
- Si Vt passe à 0 pendant une durée très courte par rapport à la durée qu'on veut du monostable, alors :
  - (S=1) et (R=0) → la sortie Vs est à 1 ;
  - le transistor est bloqué, ce qui permet au condensateur C de se charger à travers R.

- Quand Vc, après un temps qui dépend de R et C, atteint 2/3 de E, on a :
  - (S=0) et (R=1) → la sortie Vs est à 0 ;
  - le transistor est saturé ; on revient à l'état de repos.
- L'expression de la durée de temporisation T_W est :

$$T_W = RC[\ln(\frac{E}{E-\frac{2E}{3}})] \quad \rightarrow \quad T_W = RC\ln 3 \quad \rightarrow \quad \boxed{T_W \approx 1.1RC}$$



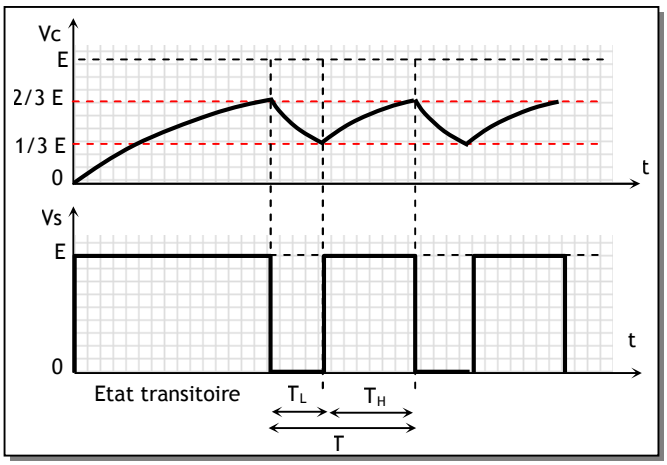


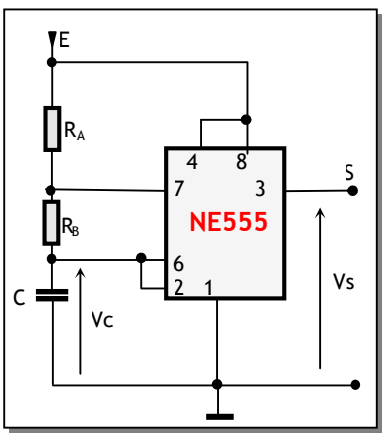
4.2. L'astable

Il y a plusieurs variantes de cette fonction ; on va étudier la plus courante. Son schéma et son fonctionnement sont donnés ci-après :

- On suppose qu'au démarrage, le condensateur C est déchargé et la sortie Vs à 1 ; alors les variables du montage sont positionnées ainsi :
  - Le entrées de la bascule (S=0) et (R=0) → Etat mémoire de la bascule et Vs reste à 1 ;
  - le transistor est bloqué, ce qui permet au condensateur C de se charger à travers (R_A+ R_B).
- Quand Vc, après un temps qui dépend de (R_A+ R_B) et C, atteint 2/3 de E, on a :
  - (S=0) et (R=1) → la sortie Vs est à 0 ;
  - le transistor est saturé ; ce qui permet au condensateur C de se décharger à travers la résistance R_B. La tension Vc diminue alors.
- Quand Vc, après un temps qui dépend de R_B et C, atteint 1/3 de E, on a :
  - (S=1) et (R=0) → la sortie Vs est à 1 ;
  - le transistor est bloqué ; le cycle recommence. Il s'agit bien d'un oscillateur.
- L'expression de la période est $T = T_L + T_H$, avec :

$$T_L = R_B C\ln 2 \quad \text{et} \quad T_H = (R_A + R_B)C\ln 2 \quad \rightarrow \quad \boxed{T \approx 0.7(2R_A + R_B)C}$$

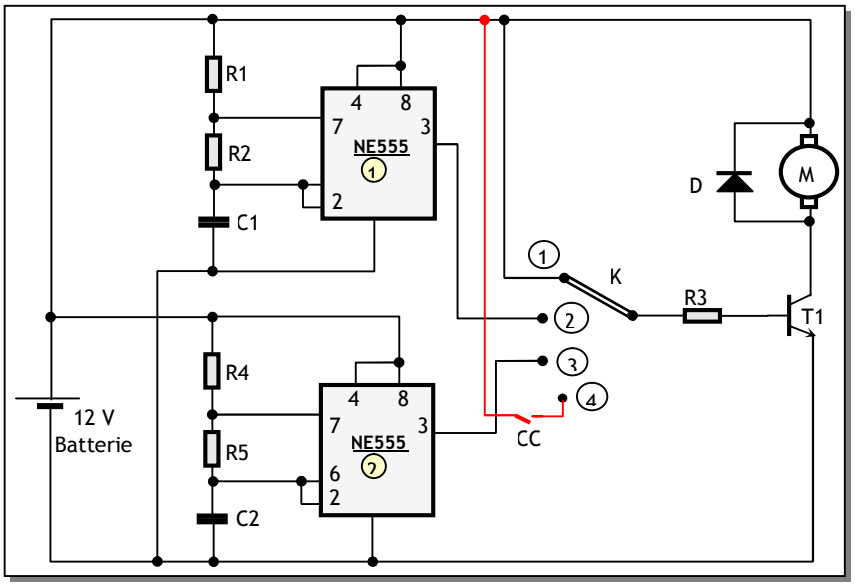


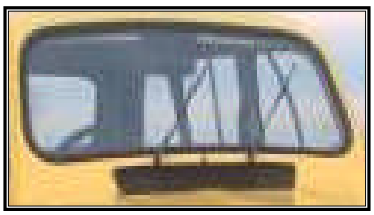


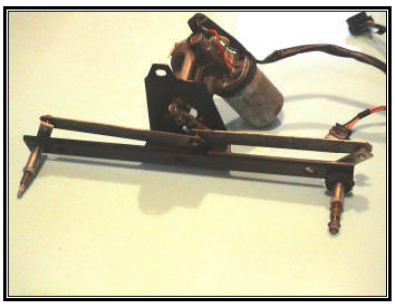
E X E R C I C E S R E S O L U S

Le montage ci-dessous représente la partie commande d'un essuie-glace, sachant que sa partie opérative est principalement constituée autour d'un moteur à courant continu et d'un système Bielle-manivelle. Analyser son fonctionnement, sachant que :

- Le NE555 (1) est un astable de (T=1ms) ;
- Le NE555 (2) est un astable de (T=7s) avec 2s de durée du niveau haut ;
- le capteur "cc" est un contact incorporé dans le moteur, qui est fermé tant que l'essuie-glace est dans sa course normale de balayage (pas dans la position initiale de repos).







CORRIGE :

Le système permet d'améliorer la visibilité d'un pare-brise. Il fonctionne de la façon suivante :

→ Si le commutateur K est en position 1, le moteur est alimenté par 12V en permanence (1ere vitesse) ;
→ Si le commutateur K est en position 2, le moteur est alimenté par un signal périodique de période 1 ms par exemple, tantôt à 12 V, tantôt à 0 V, ce qui fait fonctionner le moteur avec une tension moyenne inférieure de celle la position 1 ; par conséquent, la vitesse de balayage est plus petite (2eme vitesse) ;
→ Si le commutateur K est en position 3, le moteur est alimenté par un signal, dont la durée de l'état 1 (12 V) est plus petite que celle de l'état 0 (0V), ce qui donne comme résultat : l'essuie-glace fait un aller retour (en 2s par exemple) et s'arrête pendant un temps plus grand (5s par exemple) ; et le cycle recommence ;
→ Si on passe de n'importe quelle position à la position 4 (arrêt), il risque que les balais ne reviennent pas à leur position initiale ; le capteur "cc" qui est fermé tant que l'essuie-glace est en course, permet alors d'alimenter le moteur et par conséquent faire revenir les balais à leur position de repos.

CIRCUITS LOGIQUES PROGRAMMABLES

INTRODUCTION :

Actuellement les systèmes techniques utilisent de plus en plus de circuits numériques, qui impliquent l'utilisation d'un nombre important de circuits intégrés logiques. Ceci a pour conséquence, une mise en oeuvre complexe, un circuit imprimé de taille importante et un prix de revient élevé. Le développement incessant de la technologie des mémoires électroniques utilisées en informatique a alors permis la naissance des circuits logiques programmables (Programmable Logic Devices ou en abrégé PLD).

1. PRINCIPES ET TECHNIQUES DE BASE :

Un circuit programmable est un assemblage de circuits de base qui sont génériques (opérateurs logiques combinatoires, multiplexeurs, bascules, etc.). On trouve en particulier dans un PLD :

- des **multiplexeurs** permettant la création de chemins entre ses différents composants ;
- des **OU Exclusif** permettant la complémentarité des fonctions ;
- des **bascules** généralement de type D, permettant la mémorisation des fonctions ;
- et des réseaux logiques programmables qu'on détaille ci-après.

Toute fonction logique combinatoire peut se mettre sous forme d'une somme (OU) de produits (ET). Partant de ce principe, on en déduit une structure appelée PLA (Programmable Logic Array). Elle est constituée :

- D'un ensemble d'opérateurs « ET » (figure 1) sur lesquels viennent se connecter les variables d'entrée et leurs compléments ;
- D'un ensemble d'opérateurs « OU » (figure 2) sur lesquels les sorties des opérateurs « ET » sont connectées.

Fig. 1 : Porte ET

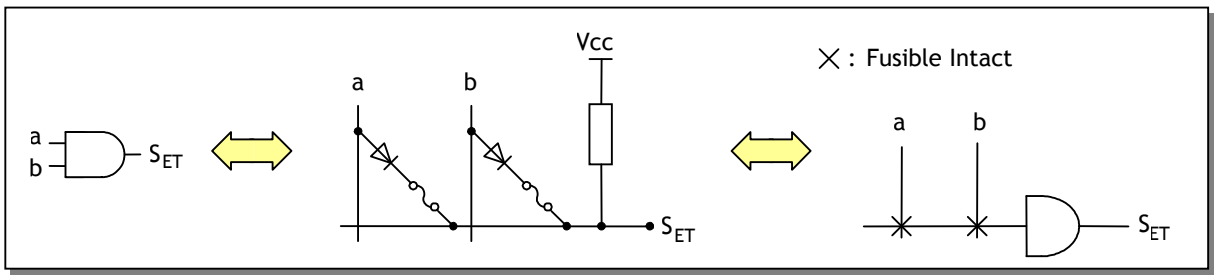
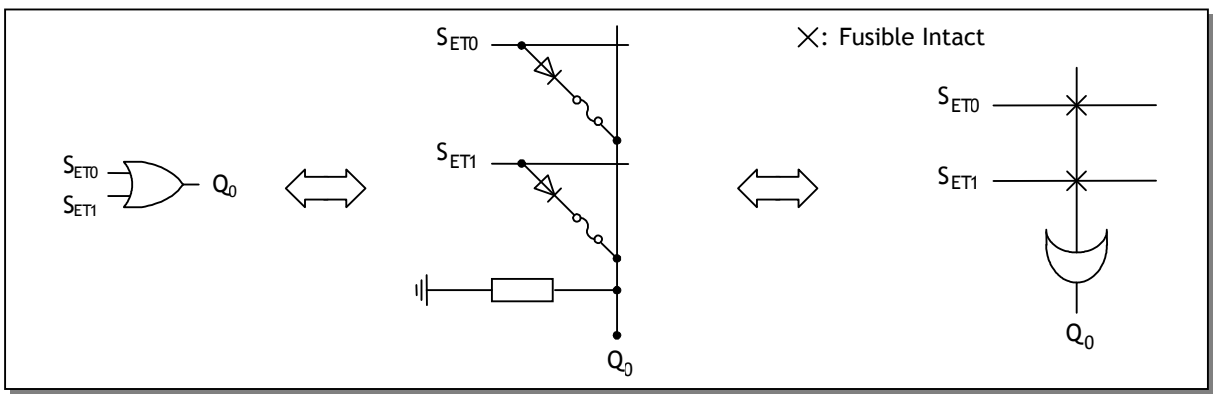


Fig. 2 : Porte OU



L'ensemble des opérateurs forment des matrices (matrice OU et matrice ET). Les interconnexions de ces matrices sont assurées par des fusibles qui sont « grillés » lors de la programmation. Lorsqu'un PLD est vierge toutes les connexions sont assurées (fusibles intacts).

La figure 3 représente la structure interne d'un PLA et la figure 4 contient la représentation simplifiée d'une telle structure :

Fig. 3 : Structure interne d'un PLA 2 entrées et 1 sortie

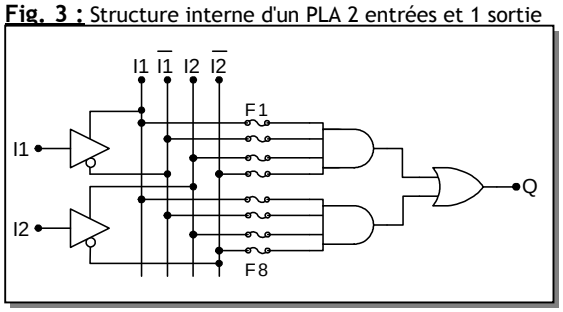
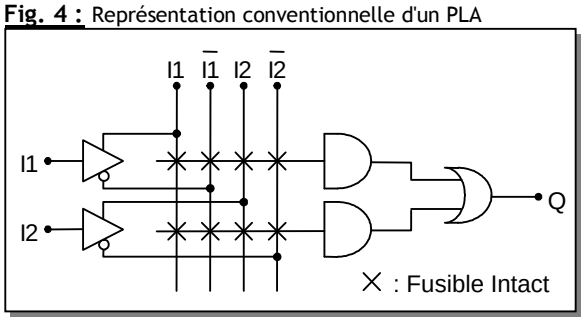


Fig. 4 : Représentation conventionnelle d'un PLA



Pour matérialiser les interconnexions :

- la technique des fusibles est en voie de disparition ; en effet, dans cette technique, on l'aura compris, le PLD est programmé un fois pour toutes ;
- Parmi les techniques utilisées actuellement, on retient celle qui est la plus courante ; il s'agit de la technique utilisant le transistor "MOS à grille isolée". Cette technique permet de programmer et reprogrammer le PLD. Le circuit est donc effaçable électriquement. De point de vue aspect physique, le fusible est tout simplement remplacé par un transistor MOS à grille isolée.

2. LA CLASSIFICATION DES PLD :

Il existe plusieurs familles de PLD qui sont différenciées par leurs structures internes. Dans e tableau suivant, on présente 2 classes de base de ces familles :

TYPE	Densité	MATRICE ET	MATRICE OU	EFFAÇABLE
PAL (Programmbale Array Logic)	10 à 100 portes	Programmable	Fixe	Non
GAL (Generic Array Logic)	10 à 100	Programmable	Fixe	Electriquement

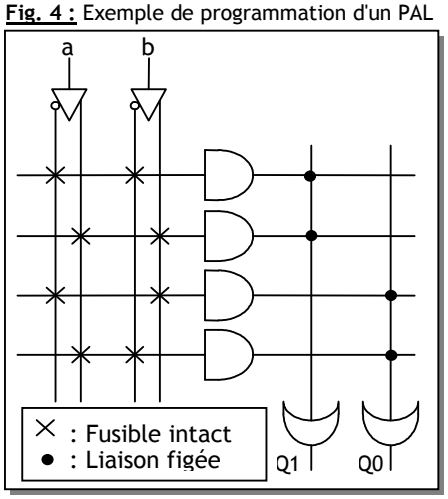
3. LES PAL (Programmable Array Logic) :

3.1. Structure de base d'un PAL :

Les PAL sont les premiers circuits programmables à être utilisés et ont été développés par le constructeur AMD il y a une vingtaine d'années. Ils possèdent une matrice « ET » programmable et une matrice « OU » fixe ou figée.

Les PAL sont programmés par destruction de fusibles et ne sont donc programmables qu'une fois. La fusion des fusibles est obtenue en appliquant à leurs bornes, à l'aide d'un programmateur adapté, une tension de 12 à 15 V pendant 10 à 50 µS.

Fig. 4 : Exemple de programmation d'un PAL



Soit à programmer les fonctions suivantes :

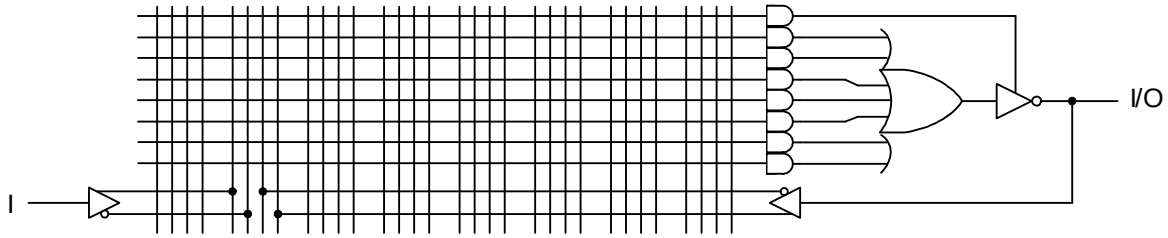
$$Q0 = a.\bar{b} + \bar{a}.b \qquad \text{et } Q0 = \bar{a}.\bar{b} + a.b$$

On « grillera » des fusibles de façon à obtenir le schéma de la figure 4 ci-contre, où seuls les fusibles intacts qui sont représentés ; les autres grillés ou sautés ne sont pas représentés.

3.2. Les différents types de sortie d'un PAL :

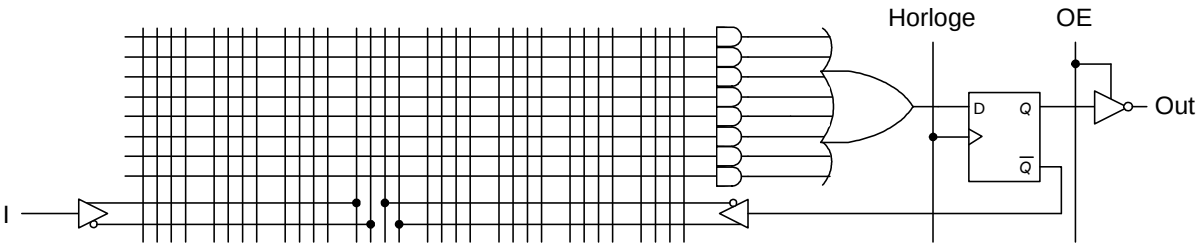
3.2.1. Structure de sorties combinatoires :

Ces sorties 3 états sont rebouclées vers la matrice de fusibles. En mode haute impédance, on peut utiliser une broche de sortie comme étant une variable d'entrée intermédiaire :



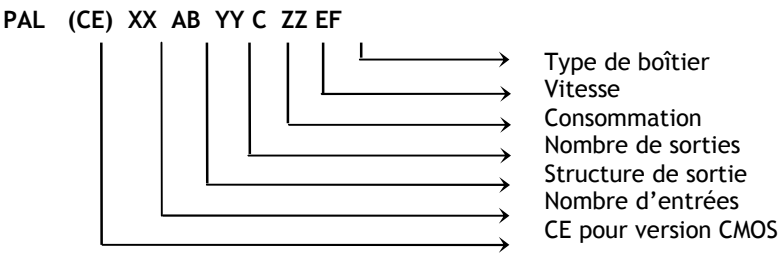
3.2.2. Structure de sortie séquentielle :

Ces sorties utilisent des bascules D dont les sorties sont de type trois états. Elles sont contrôlées par un signal de validation OE (Output Enable) et une horloge commune à toutes les bascules :

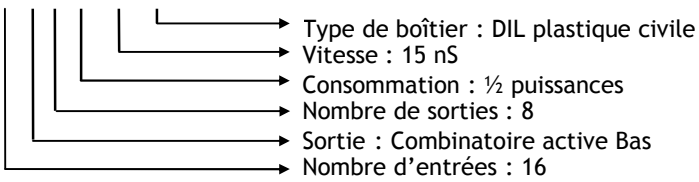


3.2.3. Référence des PAL (D'après AMD) :

Les constructeurs ont défini une nomenclature permettant de décoder facilement la référence des PAL :



Nous allons voir le **PAL 16 L 8 H 15 PC**



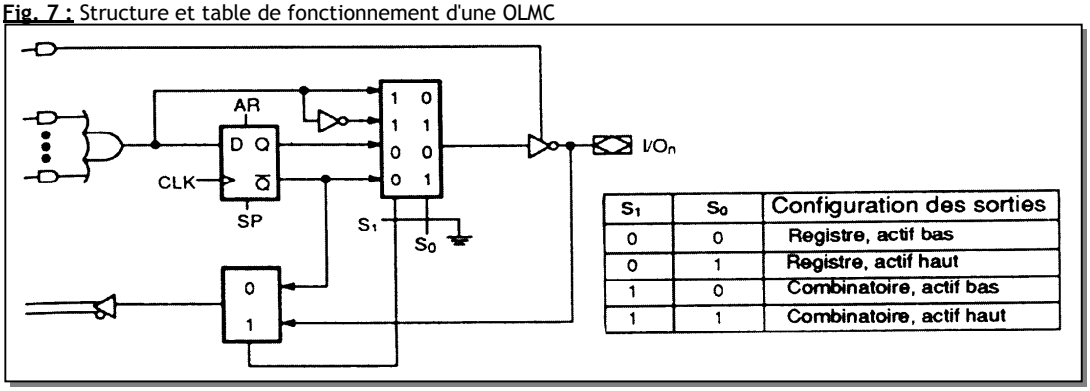
4. LES GAL (GENERIC ARRAY LOGIC) :

L'inconvénient majeur d'un PAL est qu'il ne peut être programmé qu'une seule fois, ce qui impose un "gaspillage" lors du développement. On a donc pensé à remplacer les fusibles par des transistors MOS pouvant être régénérés, d'où le terme "Generic Array Logic"(Réseau logique programmable). GAL est marque déposée de LATTICE.

Les GAL sont donc des PAL effaçables, avec en plus macro-cellules de sortie programmables, ce qui le rend versatile. La partie nommée OLMC (OUTPUT LOGIC MACROCELL) est versatile, ce qui veut dire qu'il est possible par programmation de choisir entre une configuration de sortie combinatoire ou séquentielle. La figure 7 montre la structure et la table de fonctionnement d'une OLMC :

- Le multiplexeur 4 vers 1 permet de mettre en circuit ou non la bascule D, en inversant ou non les signaux ;
- Le multiplexeur 2 vers 1 permet de réinjecter soit la sortie, soit l'entrée du buffer de sortie vers le réseau programmable.

Fig. 7 : Structure et table de fonctionnement d'une OLMC

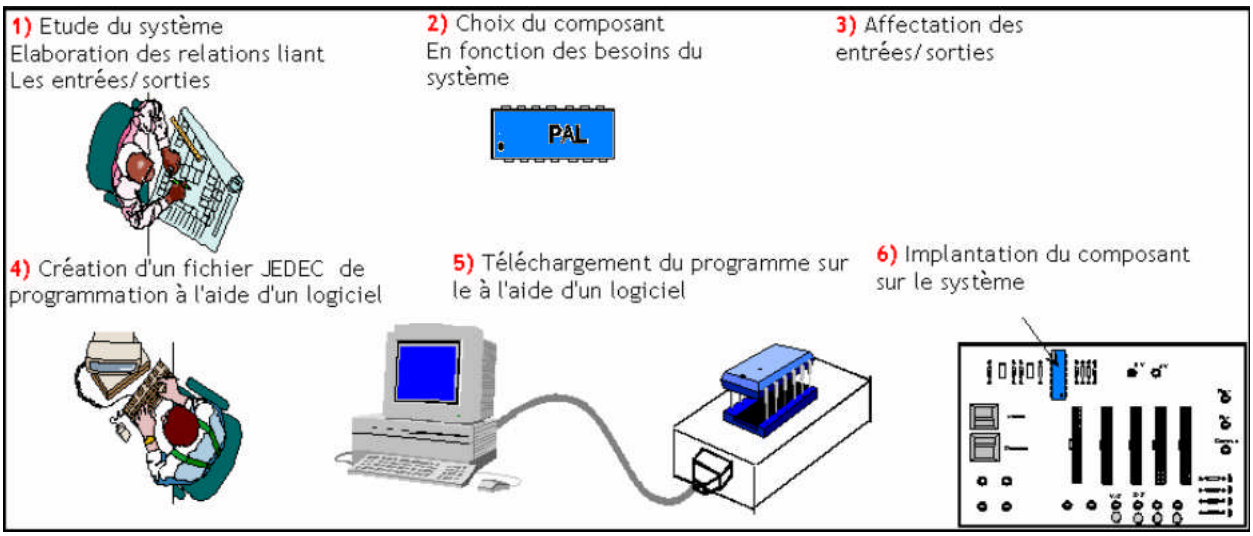


S_1	S_0	Configuration des sorties
0	0	Registre, actif bas
0	1	Registre, actif haut
1	0	Combinatoire, actif bas
1	1	Combinatoire, actif haut

5. PROGRAMMATION DES PLD

La programmation des PLD nécessite :

- Le logiciel de développement permettant de simplifier les équations et de générer un fichier JEDEC à partir des données rentrées par l'opérateur. Le fichier JEDEC étant un ensemble de données binaires indiquant au programmateur les fusibles à "griller" ;
- Un programmateur permettant de "griller" les fusibles du PLD en fonction des données du fichier JEDEC. Il est en général associé à un logiciel de pilotage. Les programmateurs utilisés sont en général les mêmes que ceux permettant la programmation des EPROM.

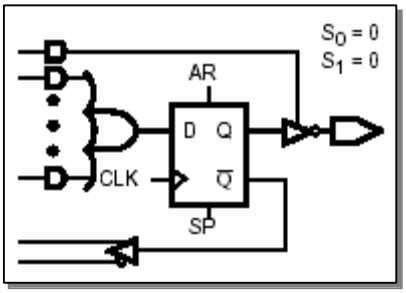


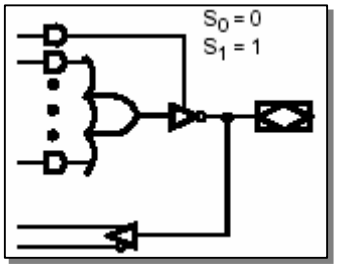
E X E R C I C E S R E S O L U S

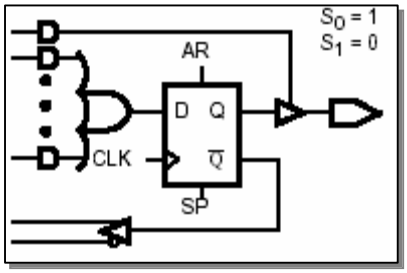
Donner pour chaque combinaison de S1S0 de la macro-cellule d'un GAL, le schéma équivalent.

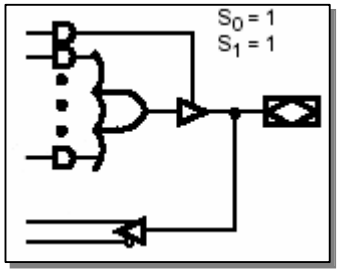
CORRIGE :

EXERCICE N° 1 :









LE GRAFCET

INTRODUCTION :

La représentation graphique permet de décrire le fonctionnement séquentiel d'un système automatisé sans ambiguïté et d'une façon compréhensible par toutes les catégories de personnel : de l'ingénieur au technico-commercial. En effet, l'œil humain est capable de saisir, d'un regard, une évolution séquentielle représentée graphiquement. Parmi les méthodes possibles, on trouve l'organigramme et le GRAFCET qui est l'objet d'étude de ce chapitre. Le GRAFCET provient de GRAphe Fonctionnel de Commande par Etapes et Transitions. Il est normalisé sur le plan international, depuis 1988 sous le nom de "Sequential Function Chart (SFC)" (norme CEI 848). Pour illustrer les notions traitées dans ce chapitre, on se basera sur le système de perçage automatisé, décrit ci-dessous :

Fig1. Perceuse automatisée

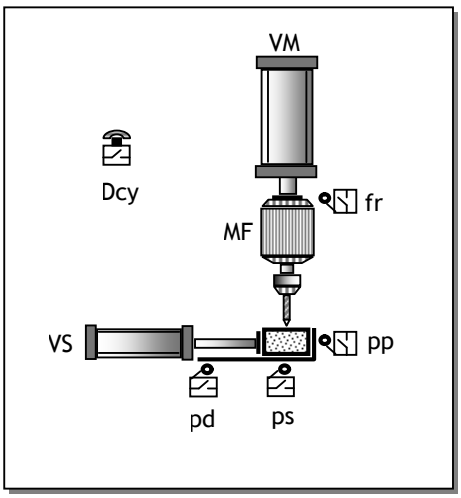


Fig2. Fonctionnement du système

- L'appui sur le bouton **D**épart **cy**cle (Dcy) lance le cycle ;
- Le vérin de serrage (VS) déplace la pièce pour la serrer ; le capteur (ps) indique que la pièce est serrée ;
- Le moteur supportant le forêt (MF) commence à tourner et le vérin (VM) pousse le moteur vers le bas ;
- Le perçage de la pièce commence et le capteur (pp) indique que la pièce est percée ;
- Alors le vérin VM remonte ; quand le capteur (fr) est actionné, cela indique que le forêt est retourné ;
- Le moteur MF et le vérin VM sont arrêtés ;
- Le vérin VS retourne dans l'autre sens ; le capteur (pd) indique que la pièce est desserrée :
- On revient alors à l'état initial.

Note : Pour la clarté, les noms des capteurs sont en minuscule et ceux des actionneurs sont en majuscule.

1. TYPES DE GRAFCET :

Suivant les différents points de vue (utilisateur, technico-commercial, concepteur-réalisateur, etc.), on peut distinguer plusieurs types de GRAFCET. Pour simplifier, on les résume dans 2 types :

- GRAFCET niveau 1 ;
- GRAFCET niveau 2.

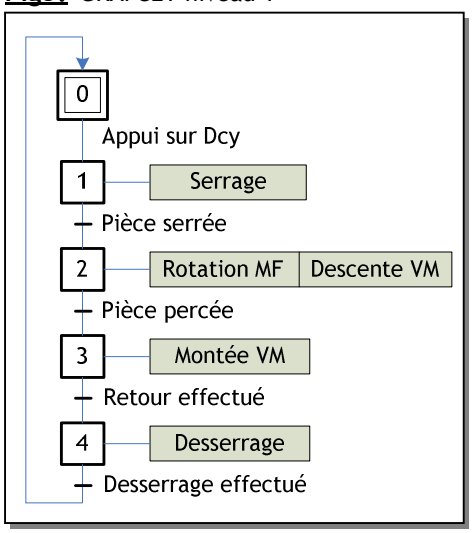
1.1- GRAFCET niveau 1 :

Dans ce type de GRAFCET, apparaissent les actions à réaliser et les informations nécessaires à leur exécution. Ce modèle est purement descriptif. Le choix des actionneurs et des capteurs n'est pas encore fait. On le désigne aussi par "GRAFCET point de vue système" ou "GRAFCET fonctionnel".

1.2- GRAFCET niveau 2 :

Une étude détaillée conduit au choix des solutions technologiques pour la partie opérative (PO) et la partie commande (PC). On le désigne aussi par "GRAFCET point de vue PO et PC".

Fig3. GRAFCET niveau 1



2. ELEMENTS DE BASE :

Le GRAFCET se compose d'un ensemble :

- D'**étapes** auxquelles sont associées des actions ;
- De **transitions** auxquelles sont associées des réceptivités ;
- De **liaisons orientées** reliant les étapes aux transitions et les transitions aux étapes.

2.1- Etape :

2.1.1- Définition :

Une étape caractérise un état qui est un comportement invariant d'une partie ou de la totalité de la partie commande. C'est une situation dans laquelle les variables d'entrée et de sortie de la partie commande conservent leur état. Une variable d'étape est associée à chaque étape (en général repéré par Xi, ou i est l'identificateur de l'étape). Cette variable booléenne a pour valeur logique :

- "1" lorsque l'étape associée est active. Par exemple pour l'étape 0, X0 = 1 ;
- "0" logique lorsque celle-ci est inactive. Par exemple pour l'étape 0, X0 = 0.

Fig4. Symbole d'une étape

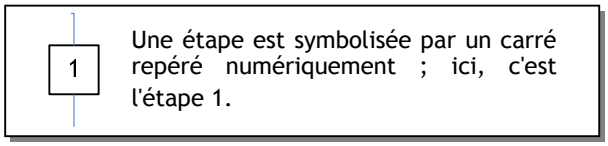


Fig5. Etape initiale

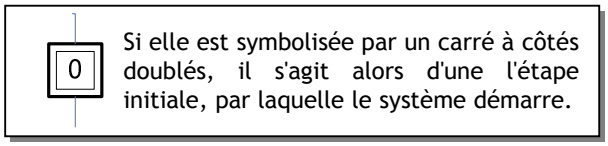
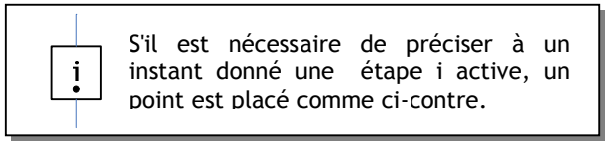


Fig6. Etape active

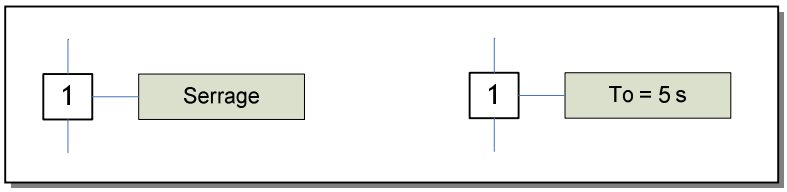


2.1.2- Actions associées à une étape :

Une ou plusieurs actions peuvent être associées à une étape. Elles traduisent ce qui doit être fait chaque fois que l'étape à laquelle elles sont associées est active. On symbolise les actions par un rectangle relié au symbole de l'étape. Elles peuvent être :

- Externes correspondant aux ordres vers la PO ;
  Exemple : Serrer la pièce.
- Internes correspondant à des fonctions qui n'agissent pas sur la PO, telles qu'une temporisation, un comptage, etc.
  Exemple : Lancer une temporisation de 5 s.

Fig7. Exemples d'actions associées à une étape



2.2- Les transitions :

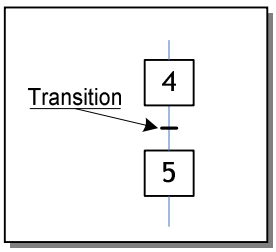
2.2.1- Définition :

Un système lors de son fonctionnement séquentiel change d'état. Une transition indique la possibilité d'évolution entre étapes :

- Elle est symbolisée par une barre perpendiculaire aux liaisons orientées ;
- Elle définit la condition d'évolution entre étapes, appelée réceptivité.

Une transition est validée si l'étape ou les étapes précédentes sont actives.

Fig8. Transition



2.2.2- Réceptivité associée à une transition :

a. Définition :

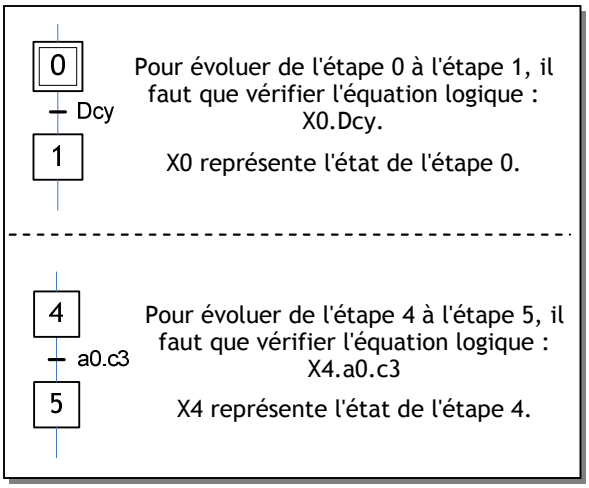
A chaque transition est associée une condition logique appelée réceptivité ou condition de franchissement d'étape. La réceptivité est une fonction combinatoire d'informations booléennes telles que l'état :

- d'un capteur ;
- d'un bouton de l'Interface Homme/Machine ;
- d'une temporisation ;
- d'une étape, etc.

Pour franchir une étape, il faut que :

- la transition soit validée ;
- **ET** la réceptivité soit vraie.

Fig9. Exemples de réceptivités



b. Cas particuliers :

Il y a des cas particuliers de réceptivité, on en cite 2 :

- **Temporisation** : Pour faire intervenir le temps dans une réceptivité, il suffit d'indiquer après le repère "t" son origine et sa durée. L'origine sera l'instant de début de l'activation de l'étape déclenchant la temporisation. La notation t/14/5 signifie que la réceptivité sera vraie 5 secondes après l'activation de l'étape repérée 14. La notation normalisée s'écrit 5s/X14.
- **Réceptivité toujours vraie** : une telle réceptivité s'écrit "= 1". Le franchissement de cette transition se fera dès que la ou les étapes immédiatement antérieures seront actives sans autre condition.

Fig10. Réceptivité toujours vraie

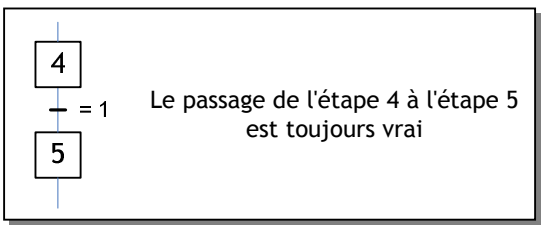
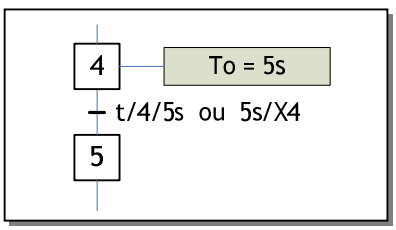


Fig11. Temporisation



3. LES REGLES D'EVOLUTION D'UN GRAFCET :

Règles N° 1 : Situation initiale

L'initialisation précise l'étape ou les étapes actives au début du fonctionnement. Elle caractérise le comportement initial de la partie commande vis-à-vis de la partie opérative. Les étapes initiales sont activées inconditionnellement en début de cycle.

Règles N° 2 : Franchissement d'une transition

Une transition est validée lorsque toutes les étapes immédiatement précédentes sont actives. Elle ne peut alors être franchie que :

- Lorsqu'elle est validée ;
- ET que la réceptivité associée à la transition est vraie.

Règles N° 3 : Evolution des étapes actives

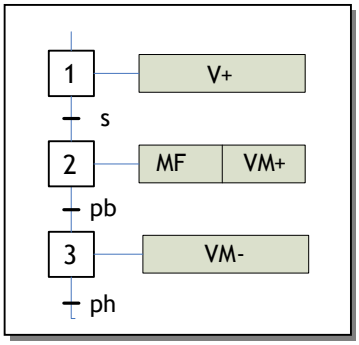
Le franchissement d'une transition entraîne l'activation simultanée de toutes les étapes immédiatement suivantes et la désactivation de toutes les étapes immédiatement précédentes.

4. STRUCTURES DE BASE D'UN GRAFCET :

4.1- La séquence linéaire :

Une séquence linéaire est composée d'un ensemble d'étapes successives où chaque étape est suivie d'une seule transition et chaque transition n'est validée que par une seule étape.

Fig12. Séquence linéaire

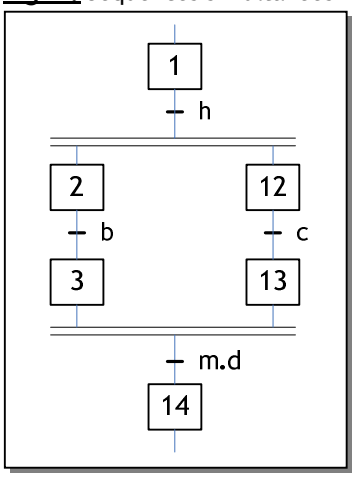


4.2- Les séquences simultanées :

Lorsque le franchissement d'une transition conduit à activer simultanément plusieurs séquences d'étapes, on obtient des séquences simultanées qui s'exécuteront parallèlement mais indépendamment. C'est-à-dire, l'évolution de chacune des séquences d'étapes dépendra des conditions d'évolution du système automatisé.

Pour représenter la structure des séquences simultanées, on utilise deux traits parallèles pour indiquer le début et la fin des séquences.

Fig13. Séquences simultanées



Considérons l'exemple de la figure 13 :

- La transition 'h', qui possède 2 étapes de sortie, représente l'exécution en parallèle de plusieurs séquences. On appelle cette structure ***divergent ET*** :

  Si l'étape 1 est active et la réceptivité h = 1, les étapes 2 et 12 sont activées

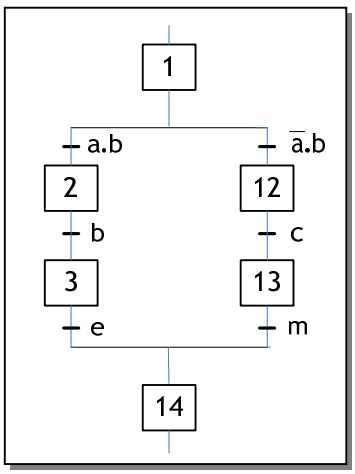
- La transition 'm.d', qui possède plusieurs étapes d'entrée, représente la synchronisation de plusieurs séquences. On appelle cette structure ***convergent ET***:

  Si les 2 étapes 3 et 13 sont actives et la réceptivité m.d = 1, l'étape 14 est activée.

4.3- Sélection de séquences :

Une structure alternative permet d'effectuer un choix unique d'évolution entre plusieurs étapes en aval à partir d'une seule étape en amont. Pour représenter la structure alternative, on utilise un simple trait horizontal pour indiquer le début et la fin des séquences. Considérons l'exemple de la figure 14 :

Fig14. Sélection de séquences



- De l'étape 1 :
  - On active l'étape 2 si la réceptivité b = 1 et a =1 ;
  - **Ou** on active l'étape 12 si la réceptivité b = 1 mais a = 0.

  On appelle cette structure "***divergence en OU***". Il est à noter que les branches d'une divergence en OU doivent avoir des réceptivités exclusives, c'est-à-dire ne peuvent pas être vraies simultanément.

- L'activation de l'étape 14 peut provenir :
  - de l'étape 3 si elle est active et e = 1 ;
  - **OU** de l'étape 13 si elle est active et m = 1.

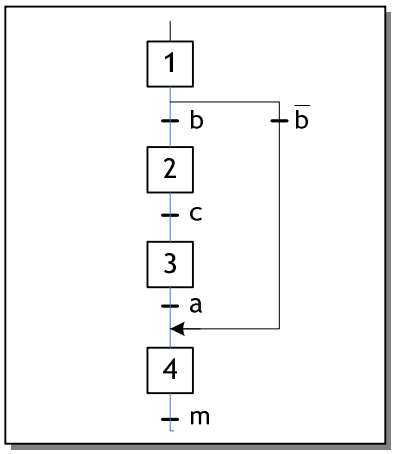
  On appelle cette structure "***convergence en OU***".

4.4- Le saut d'étapes :

Le saut d'étape représente un saut conditionnel permettant de sauter plusieurs étapes pour activer une étape en aval dans la séquence.

Dans l'exemple de la figure 15, il y a un saut de l'étape 1 à l'étape 4 mais conditionné par la réceptivité b.

Fig15. Saut d'étapes

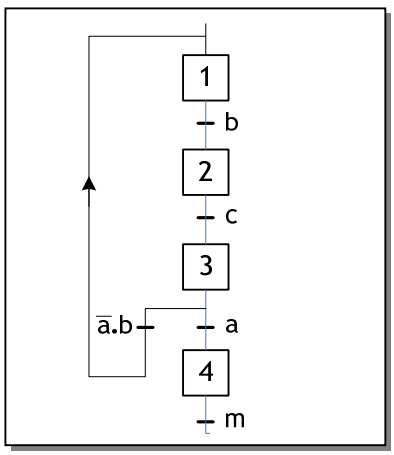


4.5- Structure répétitive :

Une structure répétitive appelée aussi une reprise de séquence, est un saut conditionnel permettant la reprise d'une séquence plusieurs fois (boucle) tant qu'une condition logique fixée n'est pas obtenue.

Il y a reprise des étapes 1, 2 et 3 tant que la réceptivité a n'est pas obtenue. On dit aussi que c'est un saut d'étape 3 à 1 par la réceptivité a.b.

Fig16. Reprise de séquence



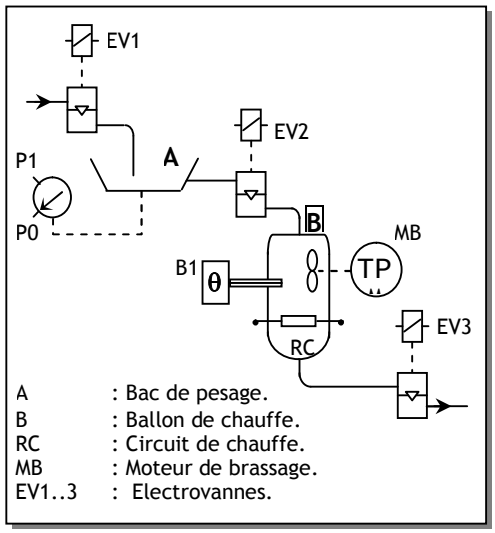
E X E R C I C E S R E S O L U S

EXERCICE N° 1 : Malaxeur agroalimentaire

Le malaxeur étudié est un système utilisé dans des usines de produits agro-alimentaires. Il décrit le processus de traitement d'un produit liquide assurant le dosage d'une certaine quantité du liquide pour la porter à une température donnée θ_0 (°c). Le système est réalisé autour de :

- Un bac de dosage A permettant de peser la quantité du liquide à chauffer ;
- Un ballon de chauffe permettant le chauffage et le brassage (mélange) du liquide pesé.

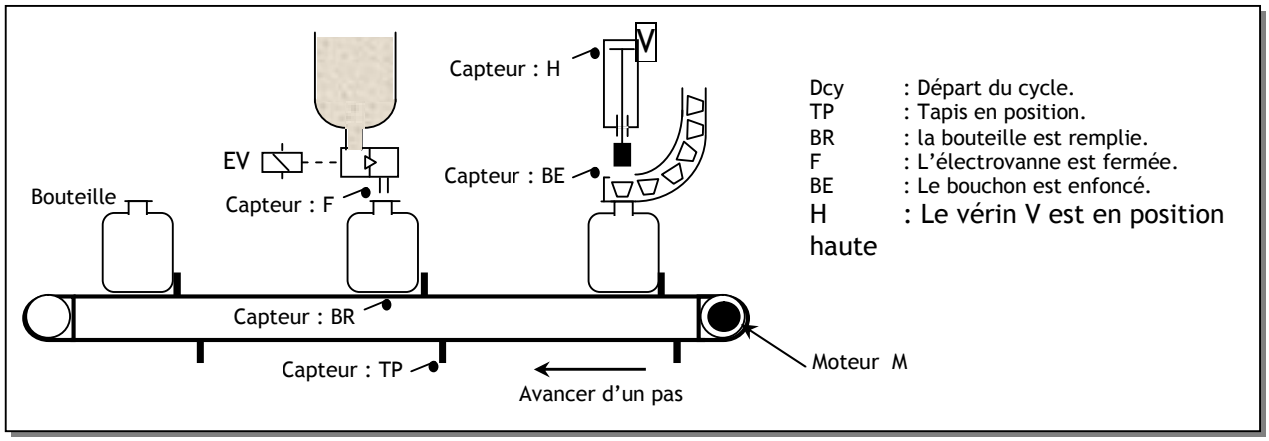
Le mode de marche du système est cycle par cycle. Le début de chaque cycle est commandé par l'appui sur le bouton poussoir **Dcy**. Les étapes suivantes sont alors exécutées :



- L'ouverture de EV1 autorise le remplissage du bac doseur A jusqu'à une valeur préaffichée P1 du système de pesage.
- Lorsque P1 est atteinte, on arrête le remplissage et on ouvre EV2 pour autoriser le déversement du liquide du bac vers le ballon de chauffe B.
- A la fin du déversement (information P0), le circuit de chauffage RC et le moteur de brassage MB sont alimentés.
- La température de chauffage est contrôlée par le capteur B1. Lorsque la température θ0 est atteinte, le chauffage et le brassage sont arrêtés et on ouvre EV3 pour autoriser la circulation du liquide chauffé vers la suite du processus.

Au bout de 20 secondes, EV3 est désactivée et un nouveau cycle peut commencer.

EXERCICE N° 2 : Etude d'une chaîne d'embouteillage



Il s'agit d'un système utilisé dans les usines de production des boissons liquides. Il décrit une partie du processus assurant les fonctions de remplissage et de bouchage des bouteilles.

Le système est réalisé autour de :

- Un tapis roulant permettant le déplacement des bouteilles.
- Un poste de remplissage P1 commandé par l'électrovanne EV.
- Un poste de bouchage P2 commandé par un vérin presseur V à double effet.

Le déclenchement de la chaîne d'embouteillage se fait par action sur l'interrupteur Dcy. Le moteur ' Avance Tapis : M ' tourne d'un pas jusqu'à l'action du capteur ' Tapis en position : TP '. Une bouteille est alors présente à chacun des postes P1 et P2. Les opérations de remplissage et de bouchage s'effectueront simultanément sur les deux bouteilles :

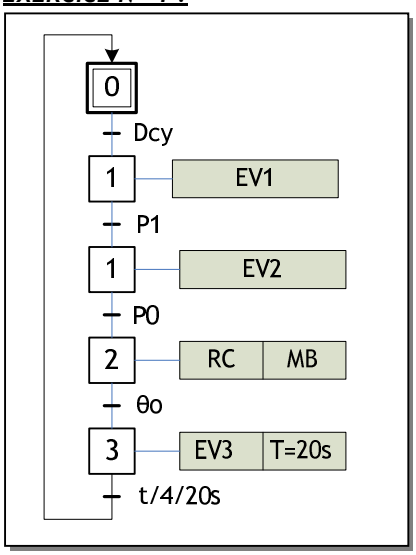
- Le remplissage se fera en deux étapes :
  - Ouverture de l'électrovanne EV ;
  - Fermeture de EV après le remplissage de la bouteille. Le capteur 'Bouteille remplie : BR' permettra de contrôler le niveau de remplissage des bouteilles.

- Le bouchage se fera en deux étapes :
  - Descente du vérin presseur V ;
  - Remonte du vérin V après l'enfoncement du bouchon.

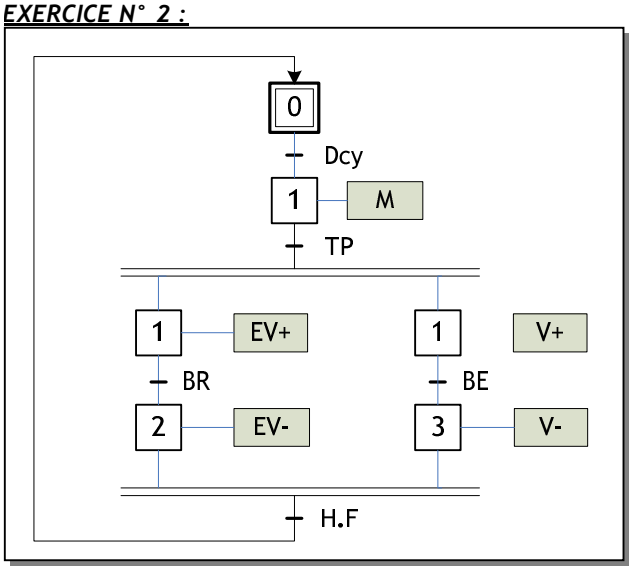
Il est à noter que le cycle ne recommencera que si les deux opérations de remplissage et de bouchage sont achevées.

CORRIGE :

EXERCICE N° 1 :



EXERCICE N° 2 :



AUTOMATE PROGRAMMABLE INDUSTRIEL

INTRODUCTION

La matérialisation d'un GRAFCET peut être réalisée de deux façons :

- Logique câblée à base de bascules : elle est simple et adaptée à des petits systèmes figés ;
- Logique programmée à base d'ordinateur, de microcontrôleur ou d'automate programmable industriel (API) : cette solution a l'avantage d'être flexible et évolutive puisqu'elle s'adapte facilement à tout changement du système par un simple changement de programme.

1. LOGIQUE CABLEE

L'élément de base dans cette logique est la bascule SR. L'action Reset étant prioritaire (S = R = 1 alors Q = 0), l'équation de la sortie Q a pour expression :

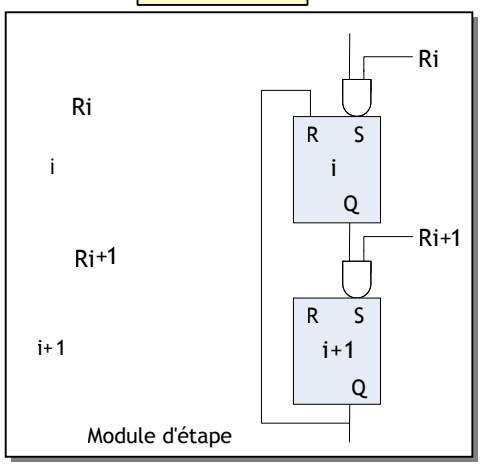
$$Q = (Q + S).\overline{R}$$

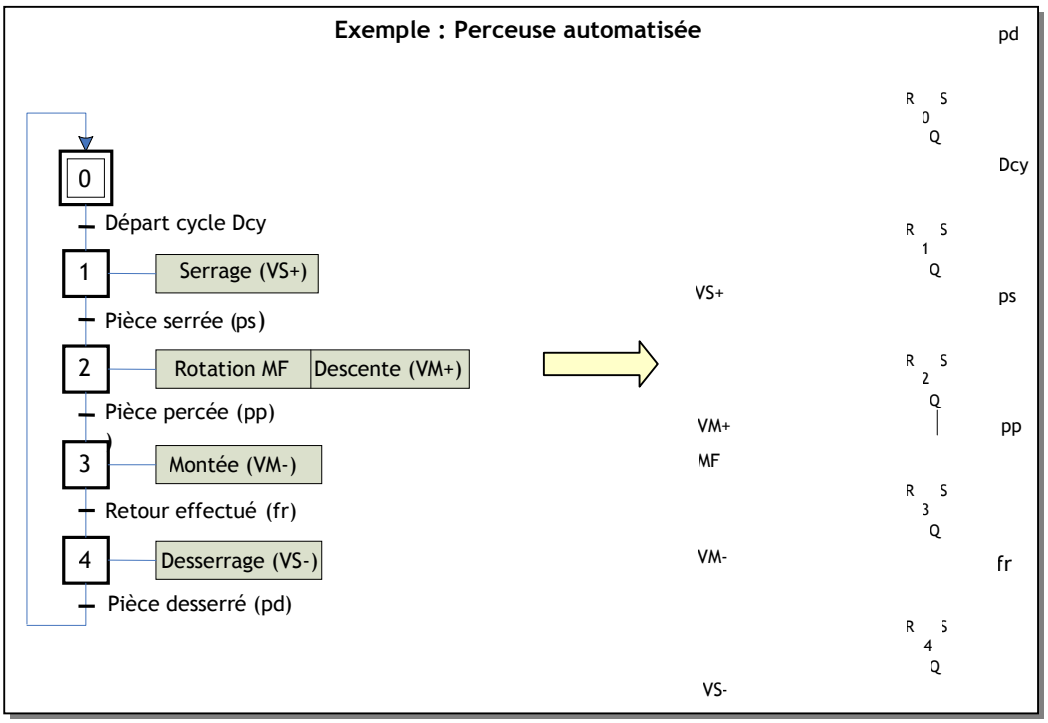
Pour matérialiser un GRAFCET, on associe à chaque étape une bascule SR ; en effet, dans un GRAFCET, une étape :

- est activée (action **Set**) par la condition (Etape précédente **ET** réceptivité vraie) ;
- reste activée même si la condition (Etape précédente **ET** réceptivité vraie) devient fausse ;
- est désactivée (action **Reset**) si l'étape suivante est activée.

Ce fonctionnement est donc traduit par le schéma ci-contre. On appelle l'ensemble formée par la bascule SR et la porte ET "module d'étape", car chaque étape sera matérialisée par ce module de base :

- i indique une étape i ;
- Ri indique la réceptivité associée à l'étape i.





- On passe du GRAFCET au schéma avec modules d'étapes comme expliqué plus haut ;
- L'équation d'une sortie se détermine :
  - en cherchant toutes les étapes où cette sortie est active ; et
  - en liant les sorties Q des bascules (X) associées à ces étapes par un opérateur logique OU.

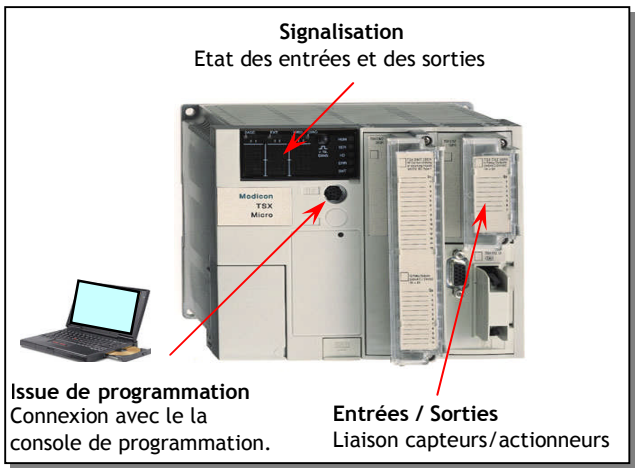
Dans le cas de l'exemple ci-dessus :

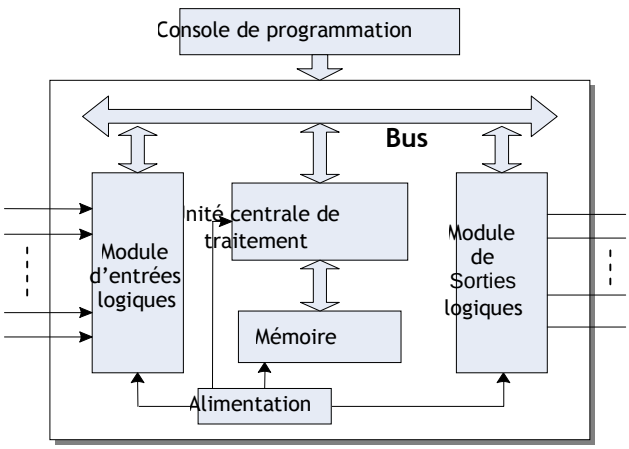
- La sortie V est active dans l'étape 1 OU 2 OU 3, alors V = X1 + X2 + X3 ;
- La sortie MB est active uniquement dans l'étape 1, alors MB = X1 ; de même, MK = X3.

2. AUTOMATE PROGRAMMABLE INDUSTRIEL :

2.1- Structure :

Un Automate Programmable Industriel (API) est une machine électronique programmable destinée à piloter en ambiance industrielle et en temps réel des systèmes automatisés. La structure interne d'un API est représentée par la figure suivante :



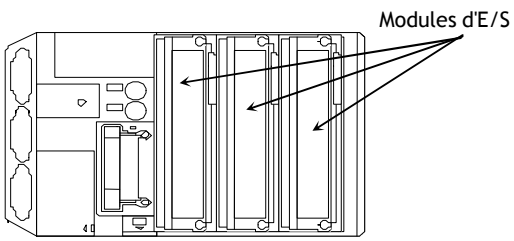


- **La mémoire :** Elle permet :
  - De recevoir les informations issues des entrées ;
  - De recevoir les informations générées par le processeur et destinées à la commande des sorties (valeur des sorties, des temporisations, etc.) ;
  - De recevoir et conserver le programme du processus.

- **L'unité de traitement :** Elle réalise toutes les fonctions logiques et arithmétiques à partir d'un programme contenu dans sa mémoire : elle lit et écrit dans la mémoire et actualise les sorties. Elle est connectée aux autres éléments (mémoire et interface E/S) par un "Bus" parallèle qui véhicule les informations entre ces éléments.

- **Les interfaces d'entrées/sorties :**
  - Les entrées reçoivent des informations en provenance des éléments de détection et du pupitre opérateur ;
  - Les sorties transmettent des informations aux pré-actionneurs et aux éléments de signalisation du pupitre.

Ces interfaces d'Entrée/Sortie (E/S) se présentent généralement sous forme d'interfaces modulaires qu'on ajoute selon le besoin.



L'interface d'entrée a pour fonction de :

- Recevoir les signaux logiques en provenance des capteurs ;
- Traiter ces signaux en les mettant en forme, en éliminant les parasites d'origine industrielle et en isolant électriquement l'unité de commande de la partie opérative (isolation galvanique) pour la protection ;
- Généralement les entrées sont désignées ainsi : %Ii.j où i est le numéro du module et j le numéro de l'entrée dans ce module, le signe "%" est spécifique au constructeur (ici Telemecanique) . Exemple : %I0.3 représente l'entrée 3 du module 0.

L'interface de sortie a pour fonction de :

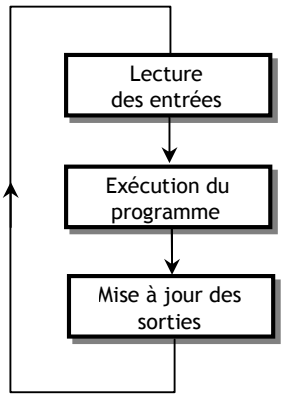
- Commander les pré-actionneurs et éléments de signalisation du système ;
- Adapter les niveaux de tension de l'unité de commande à celle de la partie opérative du système en garantissant une isolation galvanique entre ces dernières ;
- Généralement les sorties sont désignées ainsi : %Qi.j où i est le numéro du module et j le numéro de la sortie dans ce module. Exemple : % Q1.5 représente la sortie 5 du module 1.

- **La console de programmation** : C'est généralement un PC où est installé qui le logiciel de programmation spécifique à l'API. Ce logiciel permet d'éditer le programme, de le compiler et de le transférer à l'automate. Le PC peut également servir de poste opérateur pour assurer la conduite de l'unité. Un autre logiciel est alors nécessaire pour assurer le dialogue avec l'automate.

2.2- Cycle d'exécution d'un automate :

Durant son fonctionnement, un API exécute le même cycle de fonctionnement qu'on appelle "cycle automate" ; la durée de ce cycle est typiquement de 1 à 50 ms :

- Avant chaque traitement, l'API lit les entrées et les mémorise durant le cycle automate ;
- Il calcule les équations de fonctionnement du système en fonction des entrées et d'autres variables et les mémorise ;
- Les résultats sont recopiés dans les sorties.

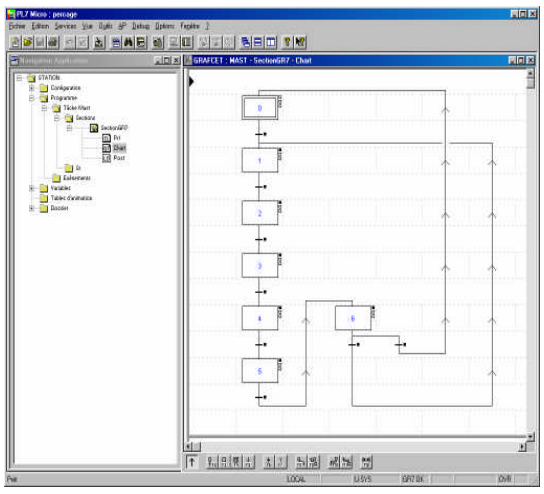


2.3- PROGRAMMATION DE L'API

La programmation d'un API consiste à traduire dans le langage spécialisé de l'automate, les équations de fonctionnement du système à automatiser. Parmi les langages normalisés, on cite quelques-uns des plus connus et plus utilisés :

- Langage à contacts (LADDER) ;
- Langage List d'instructions (Instruction List) ;
- Langage GRAFCET (Sequential Function Chart : SFC).

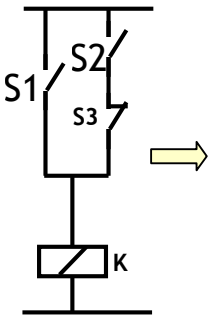
Généralement, les constructeurs d'API proposent des environnements logiciels graphiques pour la programmation. Un exemple typique d'interface graphique se présente comme ci-contre :

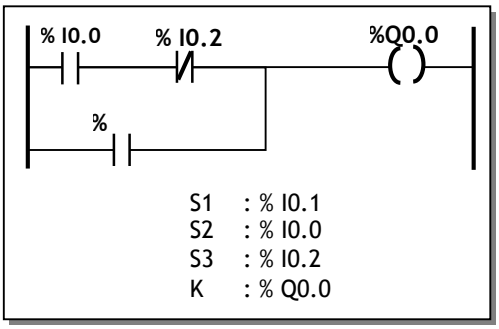


2.3.1- Le LADDER

a. Description :

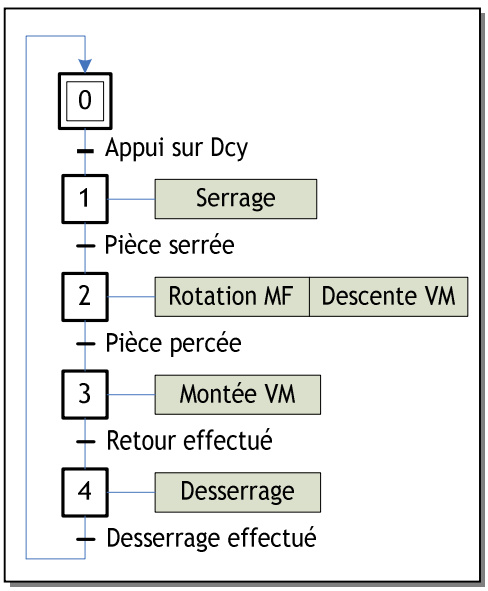
Le langage Ladder est une succession de "réseaux de contacts" véhiculant des informations logiques depuis les entrées vers les sorties. C'est une simple traduction des circuits de commande électriques.





b. Exemple :

Dans cet exemple, on traduit le GRAFCET correspondant à la perceuse automatisée en LADDER :



Pour matérialiser ce GRAFCET, on utilise les associations suivantes :

- Pour les fin de course :

Départ de cycle	**% I0.1**
Pièce serrée	**% I0.2**
Pièce percée	**% I0.3**
Retour effectué	**% I0.4**
Desserrage effectué	**% I0.5**

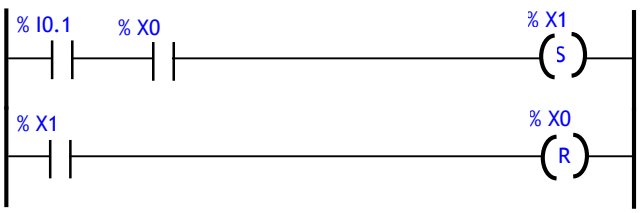
- Pour chaque étape, on associe une variable automate dont l'indice est le numéro d'étape : **X0** pour l'étape 0, **X1** pour l'étape 1, etc.
- Pour chaque action, on associe une variable de sortie de l'API.

Serrage	**% Q1.1**
Rotation MF	**% Q1.2**
Descente VM	**% Q1.3**
Montée VM	**% Q1.4**
Desserrage	**% Q1.5**

On a vu dans la matérialisation par bascules que :

- Si l'étape i est active et si la réceptivité suivante est vraie alors l'étape (i+1) est activée.
- L'activation de cette étape (i+1) désactive l'étape i.

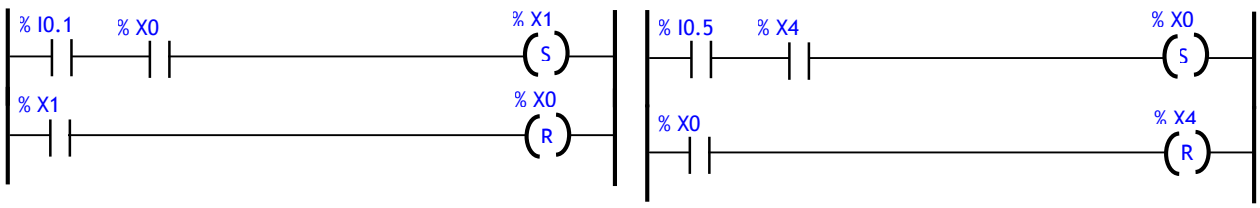
Pour l'étape1, en LADDER, ceci est représenté par :

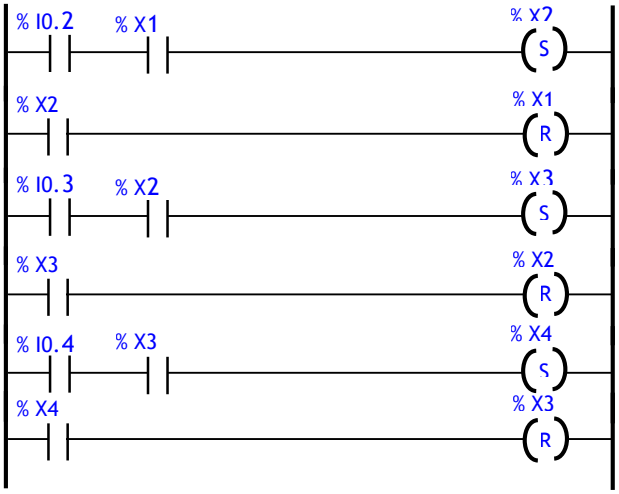


Si X0=1 et I0.1=1 alors Set X1

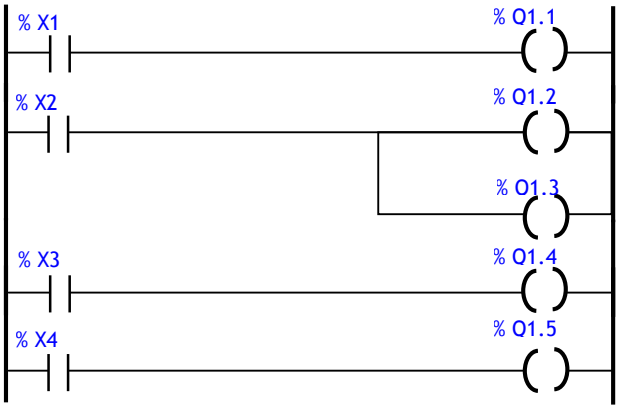
Si X1=1 alors Reset X0

Le programme complet sera alors





Traitement postérieur : Activation des sorties



2.3.2- Liste d'instructions "IL" :

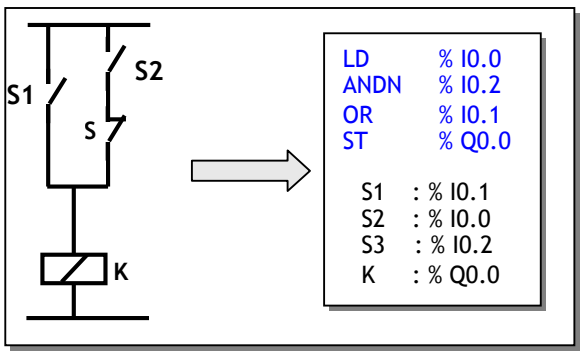
a. Description :

L'IL est un langage dans lequel toutes les opérations sont décrites par des instructions mnémoniques. Ce n'est pas un langage graphique. Le tableau suivant donne une liste représentative de ce langage :

INSTRUCTION	FONCTION
LD	Lire une entrée ou une variable interne.
LDN	Lire l'inverse d'une entrée ou d'une variable interne.
AND	ET logique entre le résultat de l'instruction précédente et l'état de l'opérande.
ANDN	ET logique entre le résultat de l'instruction précédente et l'état inverse de l'opérande.
OR	OU logique entre le résultat de l'instruction précédente et l'état de l'opérande.
ORN	OU logique entre le résultat de l'instruction précédente et l'état inverse de l'opérande.
N	Négation du résultat de l'instruction précédente
ST	L'opérande associé prend la valeur du résultat de la zone test.
STN	L'opérande associé prend la valeur inverse du résultat de la zone test.
S	L'opérande associé est mis à 1 lorsque le résultat de la zone test est à 1.
R	L'opérande associé est mis à 0 lorsque le résultat de la zone test est à 1.
END	Fin de programme.

b. Exemple : le programme suivant est une mise en œuvre du GRAFCET de l'exemple de la perceuse en langage IL

Principe de base




```
LD% I0.1
AND     % X0
S       % X1
R       % X0
LD      % I0.2
AND     % X1
S       % X2
R       % X1
LD      % I0.3
AND     % X2
S       % X3
R       % X2
LD      % I0.4
AND     % X3
S       % X4
R       % X3
```

```
LD      % I0.5
AND     % X4
S       % X0
R       % X4
Traitement postérieur
LD      % X1
S       % Q1.1
LD      % X2
S       % Q1.2
S       % Q1.3
LD      % X3
S       % Q1.4
LD      % X4
S       % Q1.5
END
```

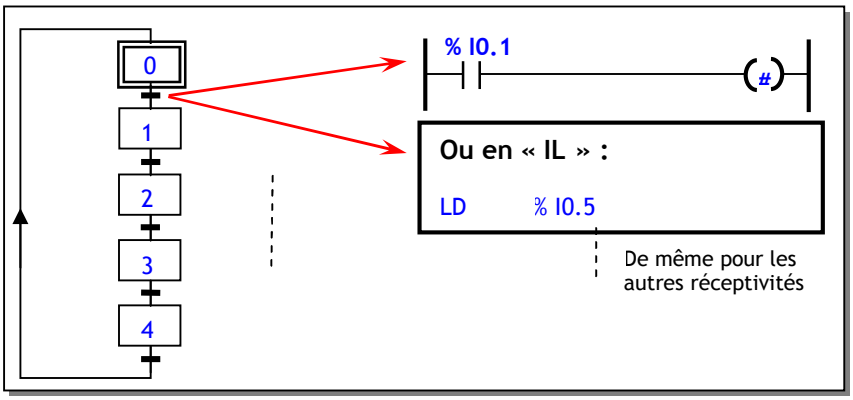
2.3.3- Sequential Function Chart (SFC)

a. Description :

Le SFC est le langage graphique qui traduit un GRAFCET sur un API. Par abus de langage, on l'appelle aussi le langage GRAFCET. Pour éditer un programme GRAFCET on passe par les étapes suivantes :

- On commence par construire graphiquement le GRAFCET ;
- On traduit les réceptivités dans le langage IL ou le langage LADDER ;
- La programmation des actions se fait dans le traitement postérieur en LADDER ou en "IL".
- En langage GRAFCET l'activation et la désactivation des étapes se fait automatiquement.

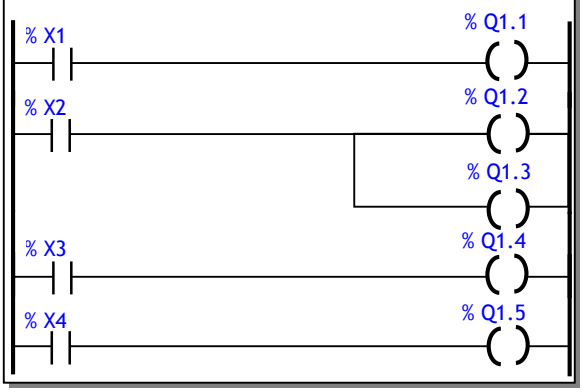
Exemple du poste de perçage



Traitement postérieur en IL

```
LD    % X1
S     % Q1.1
LD    % X2
S     % Q1.2
S     % Q1.3
LD    % X3
S     % Q1.4
LD    % X4
S     % Q1.5
END
```

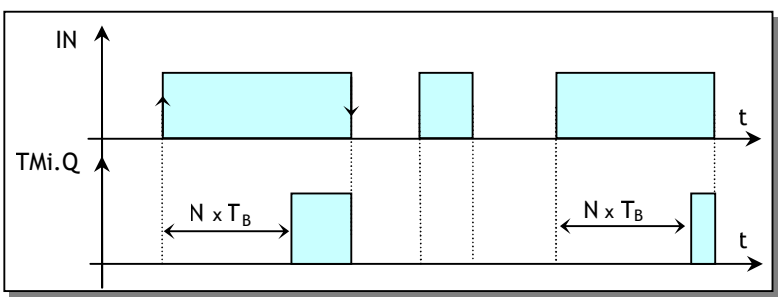
Traitement postérieur en LADDER



2.3.4- Fonctionnement des timers dans un API

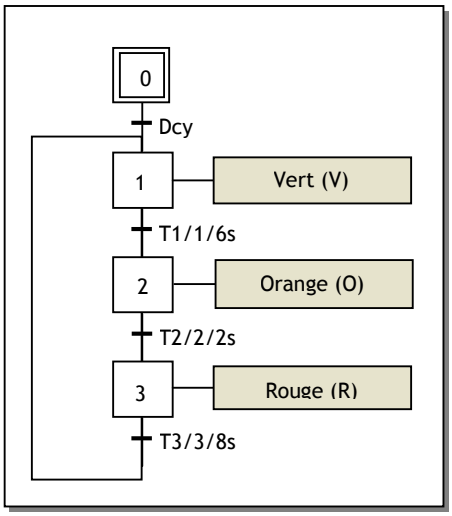
Un temporisateur (Timer), a pour fonction de retarder une action. Dans un API on dispose généralement de trois type de temporisateur, dont le plus utilisé est le temporisateur TON. Un temporisateur TON, lorsqu'il est lancé, fait un retard t = N x TB, comme l'indique la figure suivante :

N : valeur de présélection ;
T_B : Base de temps.



EXERCICE N° 1 :

Le GRAFCET suivant représente le fonctionnement des feux de croisement. On se propose de le matérialiser par langage GRAFCET.



On utilise les association suivantes :

- Pour les fin de courses :

  Départ de cycle **% I0.1**

- Pour les timers on utilise trois timers de type TON dont les sorties sont :

  % TM0 **% TM0.Q**
  % TM1 **% TM1.Q**
  % TM2 **% TM2.Q**

- Pour chaque action, on associe une variable de sortie de l'API.

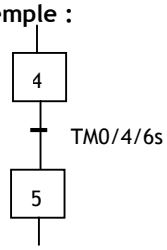
  Vert (V) **% Q1.1**
  Orange (O) **% Q1.2**
  Rouge (R) **% Q1.3**

On adopte le fonctionnement simple suivant :

- On étudie uniquement une voie ;
- Le vert s'allume pendant 6s ;
- L'orange s'allume pendant 2s.
- Le rouge s'allume pendant 8s.

Traduire les réceptivités en LADDER et le traitement postérieur en IL

Exemple :

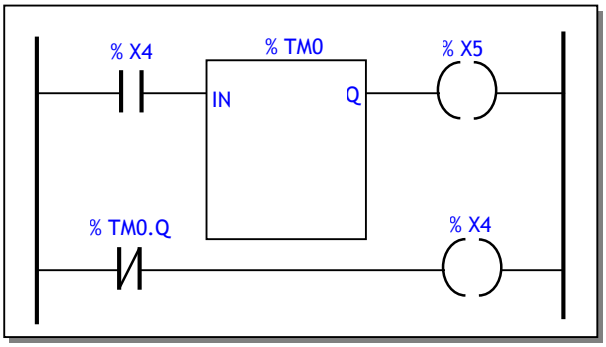


- Le timer TM0 est lancé par la variable d'étape X4.
- Quand la sortie du timer TM0.Q passe à 1 il doit activer l'étape X5 et désactiver l'étape X4.
- La désactivation de X4 permet d'initialiser le timer pour le préparer pour une nouvelle utilisation.

CORRIGE :

EXERCICE N° 1 :

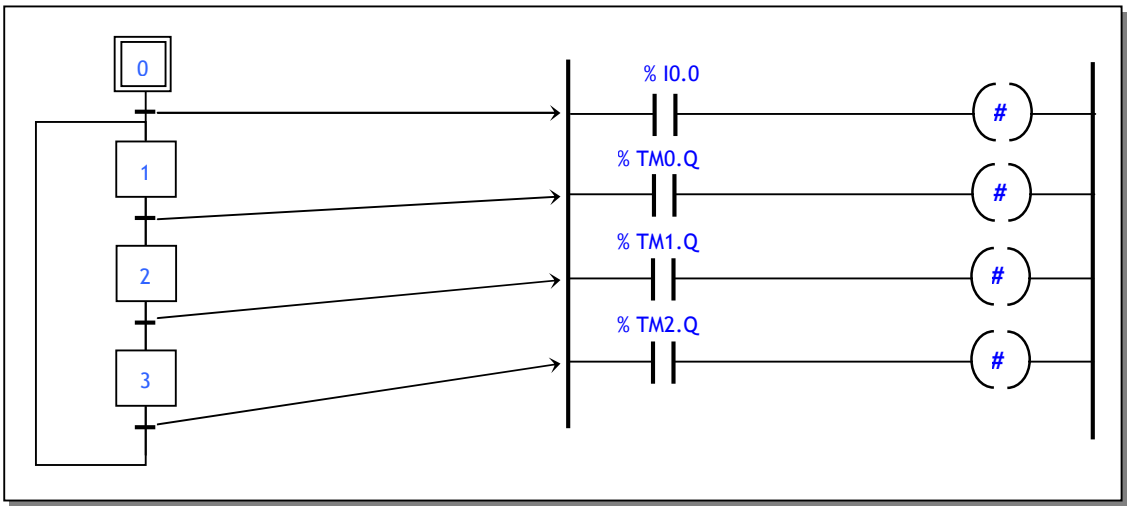
L' interprétation de ceci en LADDER et en IL se fait comme suit :



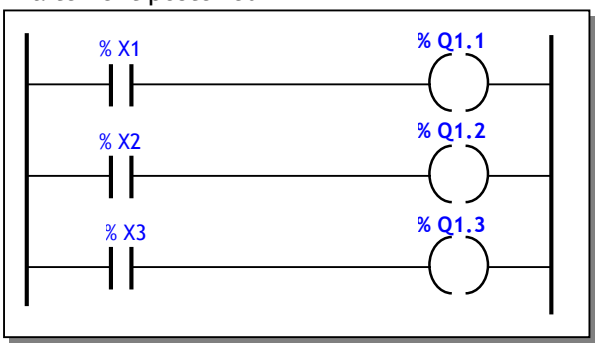

```
LD   % X4
ST   % TM0.IN
LD   % TM0.Q
S    % X5
R    % X4
```

On propose dans ce qui suit la correction de l'application en traduisant les réceptivités et le postérieur en LADDER.

Le GRAFCET



Traitement postérieur



Préliminaire : Initialisation des compteurs