

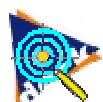
CODAGE DE L'INFORMATION



Références au programme de formation

- Traitement de l'information (ET 3.1.4)

I/ Objectifs et prérequis



Objectifs

A la fin de la séquence, l'élève doit être capable

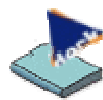
- de distinguer une information logique d'une information analogique
- de réaliser une conversion entre les bases 1, 10 et 16
- de décrypter une table ASCII



Pré-requis

- Caractérisation de l'information analogique et logique

II/ Information logique



Information logique

Une information est un message que l'on reçoit ou que l'on souhaite transmettre. En ce sens, il est fondamental que le message **arrive à destination sans modifications ou déformations**. Il est tout aussi fondamental que le message **puisse être manipulé** sans trop de difficulté

Ces deux contraintes ont conduit les concepteurs de systèmes techniques, au fil du temps, à la conclusion que le message le plus simple à manipuler est formé d'informations ne possédant que deux états distincts.

Ces deux états sont appelés état logique 0 (0) et état logique 1 (1) ou niveau logique bas (NL0) et niveau logique haut (NL1)

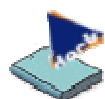
Application :



Dans le domaine de la musique, les disques optiques (appelés disques *Compact Disc*®) ont remplacé les disques vinyles en raison de l'amélioration de la qualité sonore.

- Pour les **disques vinyles** une différence de profondeur des sillons permettait de restituer un son, mais l'usure ou l'encrassement des sillons conduisaient à une dégradation de la musique.
- Pour les **CD**, seuls deux niveaux distincts sont gravés dans la surface. Ces différences de niveau permettent, grâce à un rayon laser et des circuits électroniques de reconstituer la musique. L'absence de contact mécanique grâce au rayon laser et la surface plane empêche tout encrassement et donc toute déformation.

III/ Etat logique et variable logique



L'état logique

L'état logique caractérise l'information que transmet une variable, ou un élément ne pouvant prendre que deux états. On l'appelle le **bit (binary unit)**

Remarque : Selon la convention adoptée un état logique peut avoir la valeur 0 ou 1.

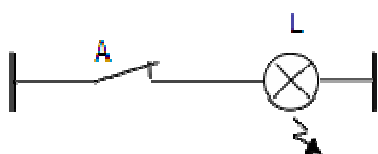
Application 1 : Circuit électrique 1



On considère qu'un état de repos est l'état 0.

- Lorsque A est relâché, A est à 0 et L est à 0
- Lorsque A est actionné, A est à 1 et L est à 1

Application 2 : Circuit électrique 2

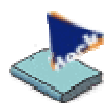


- Lorsque A est relâché, A est à 0 et L est à 1
- Lorsque A est actionné, A est à 1 et L est à 0

Application 3 : Lettre A

Soit la lettre A formée d'un carré de 8 x 8 pixels. Si on considère qu'un pixel blanc correspond à un NL0 et qu'un pixel noir correspond à un NL1, la lettre peut être décrite par une matrice de pixels à 0 ou 1.

0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

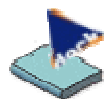


Variable logique

Une variable logique est une grandeur qui peut prendre deux états logiques.

Grâce aux variables logiques, il est possible d'établir des équations logiques qui décrivent une fonction.

IV/ Equation logique et fonctions logiques

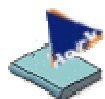
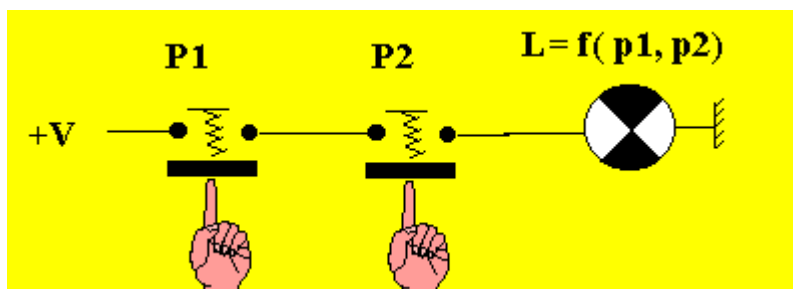


Définition

Comme en arithmétique, une équation caractérise une variable en fonction d'autres variables et/ou de constantes.

Les opérateurs logiques présents dans une équation sont :

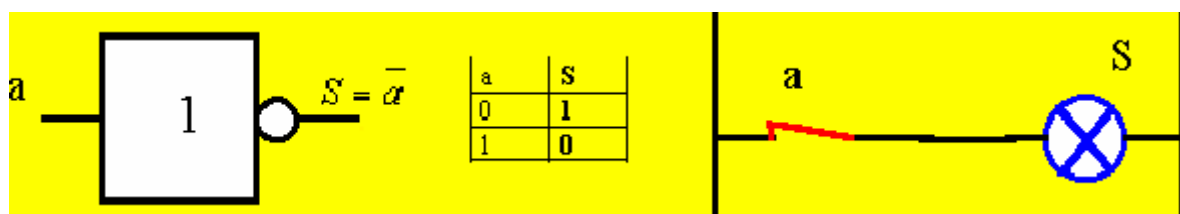
- **NON** : inverse l'état logique d'une variable. Le caractère $\bar{\quad}$ ou $/$ spécifie l'opérateur NON. Exemple : Si $A=0$, $/A=1$. Si $A=1$, $/A=0$. On dit qu'on **complémente** la valeur.
- **OU** : Représentée par le signal plus "+", la fonction OU impose que au moins une des variables liée à l'opérateur soient à 1 pour que le résultat soit à 1
- **ET** : Représentée par un point ".", la fonction ET impose que les variables liées à l'opérateur soient à 1 pour que le résultat soit à 1. Ci-dessous : $L=P1 . P2$



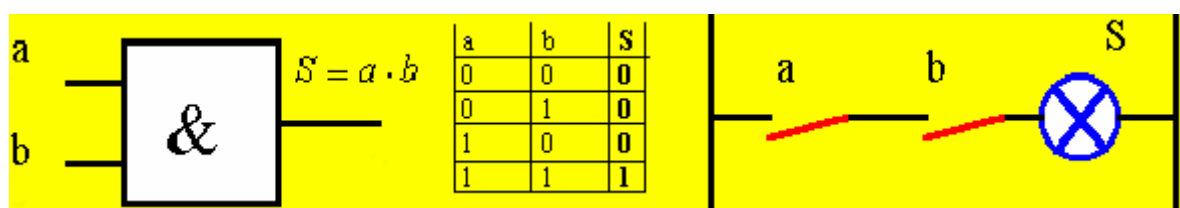
Fonctions logiques

Une *fonction logique*, appelée aussi *Porte Logique*, est une structure matérielle, un composant électronique, qui assure une opération logique.

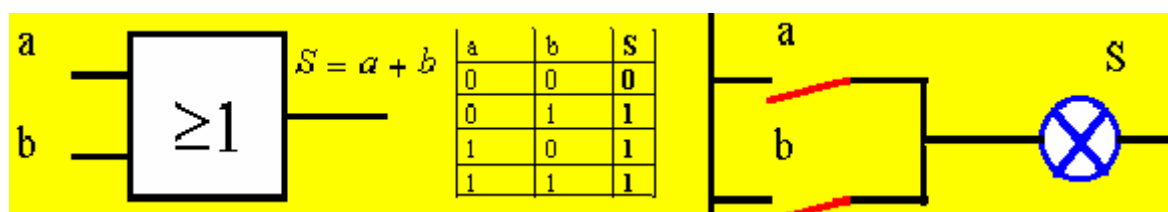
Fonction NON



Fonction ET



Fonction OU

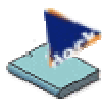


- L'association des fonctions ET et NON donnent une fonction NON-ET (NAND)
- L'association des fonctions OU et NON donnent une fonction NON-OU (NOR)

V/ L'Algèbre de Boole

Georges Boole, mathématicien anglais né en 1815 a élaboré des relations mathématiques qui ont facilité les manipulations des équations logiques.





Propriétés

Commutativité : $A + B = B + A$ et $A.B = B.A$

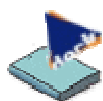
Associativité : $A + (B + C) = (A + B) + C$ et $(A.B).C = A.(B.C)$

Élément neutre : $A + 0 = A$ et $A.1 = A$

Complémentarité : $A . /A = 0$ et $A + /A = 1$

Distributivité : $(A + B).C = A.C + B.C$ et $(A . B) + C = (A + C).(B + C)$

VI/ Les codages



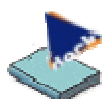
Définition

Pour transmettre une information, des codes sont nécessaires pour les représenter.

Dans ce qui nous intéresse, les codes reposent toujours sur un ensemble de bits (un mot), soit en respectant une base de numération soit en suivant un code arbitraire.

	120	121	122	123	124	125	126	127	128	129	12A	12B
0	0	1	2	3	4	5	6	7	8	9	A	B
1	C	D	E	F	G	H	I	J	K	L	M	N
2	O	P	Q	R	S	T	U	V	W	X	Y	Z
3	[a]	[b]	[c]	[d]	[e]	[f]	[g]	[h]	[i]	[j]	[k]	[l]
4	[m]	[n]	[o]	[p]	[q]	[r]	[s]	[t]	[u]	[v]	[w]	[x]
5	[y]	[z]	[aa]	[ab]	[ac]	[ad]	[ae]	[af]	[ag]	[ah]	[ai]	[aj]
6	[ak]	[al]	[am]	[an]	[ao]	[ap]	[aq]	[ar]	[as]	[at]	[au]	[av]

VII/ Les bases de numération



La numération

Dans un système de numération pondéré de base B, un entier N est représenté par un polynôme qui est de la forme :

$$N = a_n . B^n + a_{n-1} . B^{n-1} + \dots + a_2 . B^2 + a_1 . B^1 + a_0 . B^0$$

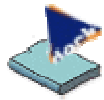
Avec les coefficients $a_n : 0 \leq a_n \leq B-1$

Par exemple :

En base 10 1982 est égal à $1000 + 900 + 80 + 2$ soit

$$1982 = 1 \times 10^3 + 9 \times 10^2 + 8 \times 10^1 + 2 \times 10^0$$

Cette relation s'applique quelque soit la base B

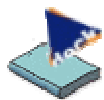


La base 2

Elle est bien sûr la base de prédilection pour une numération en automatisme et en informatique.

Remarque : pour spécifier un nombre binaire, on le fait précéder souvent par le caractère "%".

Nombre décimal	Code binaire pur			
	8	4	2	1
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



La base 16

Pour des raisons de commodité de lecture et d'écriture, on utilise la base 16 - Base hexadécimale - lorsqu'un grand nombre de bits doivent être utilisés.

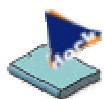
La base 16 est un regroupement de 4 bits (2^4), elle comporte donc 16 caractères : de 0 à 9 et de A à F.

Décimal	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Héxa	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F

Remarque : pour spécifier un nombre hexadécimal, on le fait précéder généralement par le caractère "\$" ou on le fait suivre par la lettre "h"

VIII/ Les codes binaires « arbitraires »

Ces codes ne respectent pas la forme polynomiale d'un nombre, mais permettent de s'adapter à certaines situations particulières.



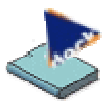
Le code de GRAY

Ce code trouve son intérêt dans le fait que d'une valeur à l'autre, seul un bit change d'état ce qui correspond à une situation réelle en automatisme : deux capteurs logiques ne changent jamais d'état en même temps.

Certains capteurs de position restituent des informations selon ce code.

Il est également utilisé dans une technique mise au point par l'automaticien KARNAUGH, qui permet de simplifier des équations logiques.

Nombre décimal	Code Gray			
	G4	G3	G2	G1
0	0	0	0	0
1	0	0	0	1
2	0	0	1	1
3	0	0	1	0
4	0	1	1	0
5	0	1	1	1
6	0	1	0	1
7	0	1	0	0
8	1	1	0	0
9	1	1	0	1
10	1	1	1	1
11	1	1	1	0
12	1	0	1	0
13	1	0	1	1
14	1	0	0	1
15	1	0	0	0



Le code BCD

Il s'agit du codage des nombres Décimaux Codés en Binaire (**B**inary **C**oded **D**ecimal).

Nombre décimal	Code BCD			
	8	4	2	1
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

Chaque nombre décimal est codé sur 4 bits et les valeurs vont donc de 0000 à 1001.

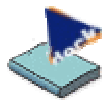
Par exemple : Le nombre décimal 9225 se code en 1001 0010 0010 0101

Les codes 1010 à 1111 sont interdits.

Ce codage a particulièrement été utilisé dans les premiers logiciels de gestion ou le signe et la partie décimale d'un nombre avait besoin d'être précis et d'avoir une valeur finie.

Il facilite l'interfaçage Machine-Homme.

IX/ Codage des nombres signés et décimaux



Nombres signés

Comme son nom l'indique, un nombre signé est un nombre pourvu d'un signe positif ou négatif.

Pour coder le signe d'un nombre binaire on utilise le bit de poids le plus fort (Le MSB). Ce qui implique qu'un mot de 8 bit peut être codé de -128 à +127

Par convention, si le MSB est à 1, le nombre est considéré comme négatif.

Pour obtenir la valeur négative d'un nombre on cherche son **complément à 2** en appliquant la séquence suivante :

Soit N le nombre

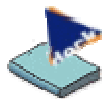
- Prendre le complément du nombre
- Ajouter 1 à sa valeur.

Exemple : -100

- 100 en décimal est égal à 01100100 en binaire
- Complément du nombre : 10011011
- On ajoute 1 : $10011011+1=10011100$

-100 est donc égal à 10011100

Remarque : En arithmétique signée, la valeur 0 vaut 10000000 soit \$80 en hexadécimal.



Nombre décimaux

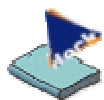
Pour représenter des nombres réels qui peuvent être signés, on utilise selon la norme de l'IEEE trois zones distinctes dans un mot de 32 bits (simple précision):

- le signe sur 1 bit
- l'exposant sur 8 bits en arithmétique signé
- la mantisse (ensemble des chiffres à droite de la virgule) sur 23 bits qui correspond au nombre décimal.

Exemple -12,25

- On transforme la valeur en nombre décimal : $-0,1225 \times 10^2$
- Le signe est négatif
- Exposant : +2 soit 00000010
- La mantisse : 0000000000010010001001
- Le résultat est donc : 1000 0001 0000 0000 0000 0100 1000 1001 = \$81000489

XI/ Codage de caractères



Codage ASCII

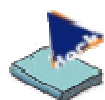
Le codage ASCII (American Standard Code for Informant Interchange) inventé en 1961, permet le codage de 256 caractères sur 8 bits chacun.

Un tableau lignes-colonnes permet de retrouver le code d'un caractère :

MSB \ LSB	0	1	2	3	4	5	6	7
	000	001	010	011	100	101	110	111
0	0000 NUL	0001 DLE	0010 SP	0011 @	0100 P	0101 `	0110 p	0111
1	0001 SOH	0010 DC1	0011 I	0100 1	0101 A	0110 Q	0111 a	1000 q
2	0010 STX	0011 DC2	0100 "	0101 2	0110 B	0111 R	1000 b	1001 r
3	0011 ETX	0100 DC3	0101 #	0110 3	0111 C	1000 S	1001 c	1010 s
4	0100 EOT	0101 DC4	0110 \$	0111 4	1000 D	1001 T	1010 d	1011 t
5	0101 ENQ	0110 NAK	0111 %	1000 5	1001 E	1010 U	1011 e	1100 u
6	0110 ACK	0111 SYN	1000 &	1001 6	1010 F	1011 V	1100 f	1101 v
7	0111 BEL	1000 ETB	1001 '	1010 7	1011 G	1100 W	1101 g	1110 w
8	1000 BS	1001 CAN	1010 (1011 8	1100 H	1101 X	1110 h	1111 x
9	1001 HT	1010 EM	1011)	1100 9	1101 I	1110 Y	1111 i	
A	1010 LF	1011 SUB	1100 *	1101 :	1110 J	1111 Z		
B	1011 VT	1100 ESC	1101 +	1110 ;	1111 K		[k
C	1100 FF	1101 FS	1110 ,	1111 <		L	\	l
D	1101 CR	1110 GS	1111 -		=	M]	m
E	1110 SO		RS	.	>	N	^	n
F	1111 SI	US	/	?	O	_	o	DEL

MSB \ LSB	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0	0000	Ç	É	á	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
1	0001	ç	é	á	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
2	0010	é	Æ	ó	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
3	0011	æ	ø	ú	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
4	0100	à	ö	ñ	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
5	0101	á	ò	Ñ	Á	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
6	0110	â	ó	ª	À	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
7	0111	ç	ù	•	À	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
8	1000	ê	ÿ	¿	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
9	1001	e	Û	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
A	1010	è	Ü	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
B	1011	í	⌘	½	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
C	1100	î	£	¼	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
D	1101	ï	⌘	;	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
E	1110	Ë	×	«	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘
F	1111	Ä	ƒ	»	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘	⌘

Exemple : la lettre D se code \$44



Codage Unicode - UTF8

Introduit en 1991, il permet de coder tout caractère indépendamment de tout système de programmation ou système d'exploitation.

Ce code est utilisé dans les pages WEB, les emails, etc...

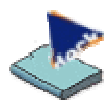
Ce code est normalisé ISO/CEI 10646

Les caractères sont groupés en blocs en fonction de leur usage et des écritures supportées, et reçoivent une identification numérique unique appelée *point de code*, identifiée généralement sous la forme U+xxxx (où xxx est un nombre hexadécimal de 4 à 6 chiffres, entre U+0000 et U+10FFFF). La plage définie permet d'attribuer jusqu'à 1 114 112 points de code.

Par exemple les points de code U+0000 à U+FFFF contient la plupart des caractères utilisés par les langues modernes les plus courantes dans le monde.

Exemple de la lettre A : U+0041 : 01000001 mais pour certains autres caractères 4 octets peuvent être nécessaires.

XII/ Conversion de bases



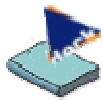
De la base 10 à la base 2

La conversion consiste à retrancher des puissances de 2 au nombre à convertir.

Exemple pour la valeur 228 :

En 228 il y a 128
 $228 - 128 = 100$
 En 100, il y a 64
 $100 - 64 = 36$
 $36 = 32 + 4$
 Les colonnes vides sont mises à 0
 Le résultat est donc : 11100100

128	64	32	16	8	4	2	1
1	1	1	0	0	1	0	0

***De la base 2 vers la base 10***

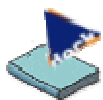
Il s'agit de faire l'opération précédente par une succession d'addition après avoir placé le nombre dans un tableau des puissances de 2.

Exemple : %10010011

1×128

128	64	32	16	8	4	2	1
1	0	0	1	0	0	1	1

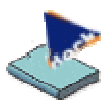
L'addition donne : $128 + 16 + 2 + 1 = 147$

***De la base 2 vers la base 16***

Comme 4 caractères binaires sont condensés en un caractère hexadécimal, il suffit de faire des regroupements des bits 4 à 4 puis de coder ces regroupements en [hexadécimal](#).

Exemple :

$$\%1111001100 = 0011\ 1100\ 1100 = \$3CC$$

***De la base 16 à la base 2***

On procède à l'opération inverse de la précédente. Chaque caractère hexadécimal est décomposé en binaire.

Exemple :

$$\$15EF = \% 0001\ 0101\ 1110\ 1111$$