UNIVERSITE SIDI MOHAMMED BEN ABDELLAH **Mr KHATORY**
Ecole **S**upérieure **de T**echnologie de **F**ès
Filière Génie **I**ndustriel et **M**aintenance

INITIATION INFORMATIQUE I

(Système de numération)

(1° GIM)

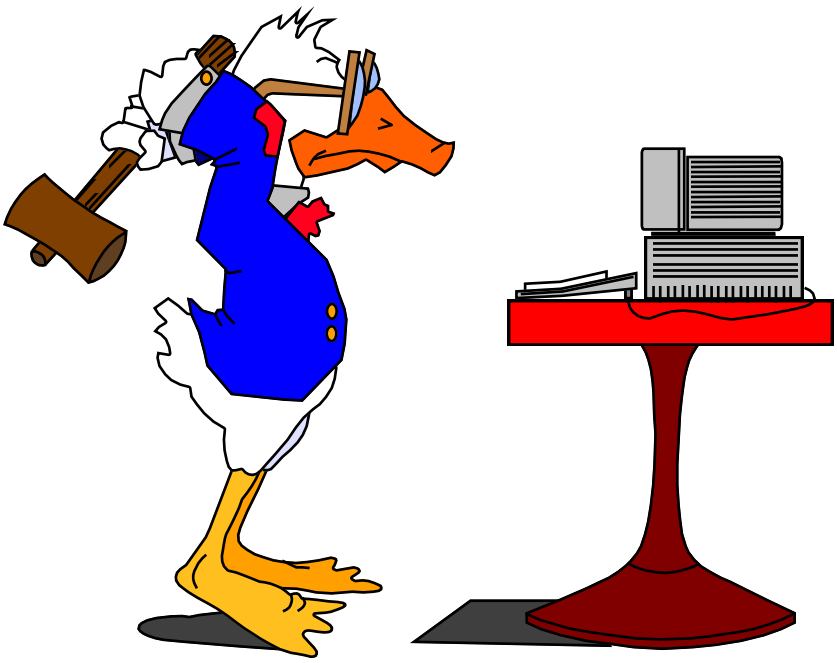


TABLE DES MATIÈRES

INTRODUCTION

La création de la numération est un des faits les plus marquants de l'histoire de l'humanité. Si la plupart des civilisations ont adopté le système décimal, c'est qu'il a toujours été naturel de compter sur ses doigts. L'utilisation des phalanges et des articulations permit même d'améliorer ce simple procédé connu de tous.

I. SYSTEME DE NUMERATION

On utilise les " systèmes de numération" pour compter des objets et de les représenter par des nombres.

Trois notions interviennent dans un système:

- la ***base B*** du système, c'est un nombre entier quelconque.
- Les ***digits*** du système sont des caractères tous différents et représentent chacun un élément de la base; il y en a donc ***B*** au total
- ***Poids*** du digit selon son rang

Ecriture d'un nombre A dans la ***base B*** :

$(A)_B = a_3a_2a_1a_0$ (4 chiffres)

$a_i < B \ (\forall_i)$

$(A)_B = a_0B^0 + a_1B^1 + a_2B^2 + a_3B^3$; Poids $a_i = B^i$

1. système décimal

Dans la base 10 "système décimal ", il y a ***dix digits***: 0,1,2,3,4,5,6,7,8 et 9 appelés ***chiffre***

$(1234)_{10} = 4x10^0 + 3x10^1 + 2x10^2 + 1x10^3$

$=4 + 30 + 200 + 1000$

B=10;

Poids:

- du premier digit est $10^0=1$ (Unité)
- du deuxième digit est $10^1=10$ (Dizaine)
- du troisième digit est$10^2=100$ (Centaine)
- du quatrième digit est $10^3=1000$ (Milliers)

2. Système Binaire

Dans ce système, la base B vaut 2, et il y a donc ***2 digits*** 0 et 1 appelés dans ce cas " **BIT**" (**B**inary dig**IT**).

Par exemple, le nombre 1011 exprimé en binaire signifie:

$(1011)_2 = 1x2^0 + 1x2^1 + 0x2^2 + 1x2^3$

$=1 + 2 + 8$

$=(11)_{10}$

3. Système octal

Dans ce système, la base vaut 8 et il y a ***8 digits***: 0,1,2,3,4,5,6 et 7. Il n'y a pas de chiffres 8 et 9.

Par exemple: le nombre 275 exprimé en octal:

$(275)_8 = 5x8^0 + 7x8^1 + 2x8^2$

$= 5 + 56 + 128$

$= (189)_{10}$

4. Système hexadécimal

Dans ce système, la base B vaut 16 et il y a ***16 digits***: 0,1,2,3,4,5,6,7,8,9,A,B,C,D,E et F. Les dix premiers digits de 0 à 9 sont les chiffres du système décimal et les digits de 10 à 15 sont les premières lettres majuscules de l'alphabet.

Exemple, le nombre BAC exprimé en hexadécimal :

$(BAC)_{16} = Cx16^0 + Ax16^1 + Bx16^2$

$=12 + 10x16 +11x256$

$=12 + 160 + 2816$

$=(2988)_{10}$

$(3F9)_{16} =9x16^0 + 15x16^1 + 3x16^2$

$=9 + 240 + 768$

$=1017$

5. Changement de base :

a. conversion octal → binaire (binaire → octal)

On peut remarquer que $8 = 2^3$;

On peut donc faire correspondre à chaque digit d'un nombre exprimé en octal un ensemble de 3 bits du même nombre exprimé en binaire. Par exemple:

$(763)_8 = (111)(110)(011)$

$=(111110011)_2$

La conversion inverse, binaire → octal, se fait de la même façon, en décomposant le nombre binaire par ensembles de 3 bits à partir de la droite.

Par exemple: $(10111011101)_2=(2735)_8$

b. conversion hexadécimal → binaire (binaire → hexadécimal)

De la même manière, on peut remarquer que $16=2^4$

On fera donc correspondre à chaque digit d'un nombre hexadécimal 4 bits du nombre binaire correspondant.

Par exemple : $(A28)_{16}=(101000101000)_2$

La conversion inverse, binaire hexadécimal, se fait en décomposant le nombre binaire par ensembles de 4 bits à partir de la droite.

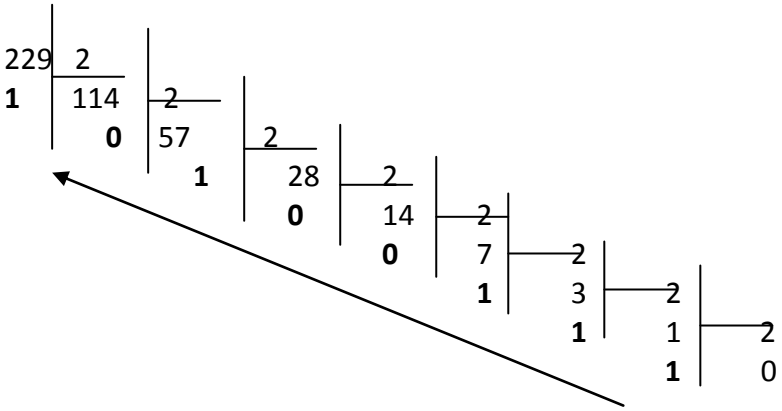
Par exemple: $(101110011101001)_2$ = (0101)(1100)(1110)(1001)

$=(5CE9)_{16}$

L'expression hexadécimal d'un nombre binaire est très utilisée pour interpréter des résultats fournis par un "microprocesseur".

c. conversion décimal → binaire , décimal → octal, ou décimal → hexadécimal

La conversion de l'expression décimale d'un nombre en son expression binaire, octale ou hexadécimale repose sur la recherche des multiples des puissances successives de la base (2,8 ou 16 selon le cas) que contient ce nombre. La méthode pratique consiste à effectuer des divisions successives: du nombre par la base, puis du quotient obtenu par la base, puis du nouveau quotient par la base,… jusqu'à ce que le quotient devienne nul. L'expression cherchée est constituée par l'ensemble des restes successifs des divisions, lu à l'envers.



$(\mathbf{11100101})_2=(229)_{10}$

la même méthode serait applicable pour les conversions :

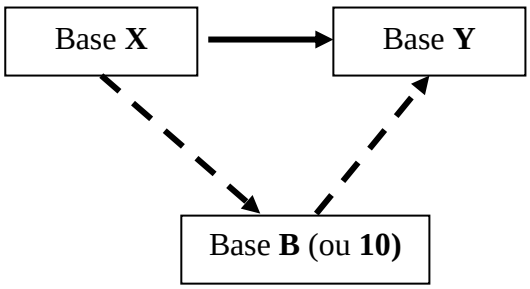
- décimal → octal (des divisions successives par 8)
- décimal → hexadécimal(des divisions successives par 16).

d. conversion d'une base X vers base Y

si $X = B^m$ et $Y= B^n$

Alors convertir le nombre de la base X (B^m) vers B puis de la base B vers la base Y (B^n)

Sinon Convertir de la base X vers la base **10** puis de la base **10** vers la base Y



II. CODAGE

On distingue deux catégories de codes: les "codes numériques" qui permettent seulement le codage des nombres, et les "codes alphanumériques" qui permettent le codage d'une information quelconque (ensembles de lettres, de chiffres et de symboles).

1. Codes numériques

a. code binaire naturel

Le code binaire naturel est le code dans lequel on exprime un nombre selon le système de numération binaire.

Quelques notions:

- un quartet : c'est un mot de 4 bits (0-15)
- un octet : c'est un mot de 8 bits (0-255)
- un "kilo" : unité de capacité de traitement numérique (10 bits: 0-1023)

Inconvénients du code binaire naturel:

- nécessite une grande quantité de bits pour exprimer un nombre
- peut introduire des erreurs lors du codage de grandeurs variant de façon ordonnée. Entre deux codes successifs, plusieurs bits pourront alors être amenés à changer simultanément:

01 → 10 (01 →11→ 10 ou 01→ 00→ 10)

b. code binaire réfléchi

Dans ce code, appelé code GRAY, un seul bit change de valeur entre deux codages successifs. Il est construit de proche en proche, de telle sorte que chaque fois que l'on ajoute au code un bit sur sa gauche, on recopie au dessous de combinaisons existantes les mêmes combinaisons, mais en les écrivant dans l'ordre opposé.

Code binaire Naturel…….	..Code binaire réfléchi
Sur 3 bits…….000……..	..000
001……..	..001
010……..	..011
011……..	..010
100……..	..110
101……..	..111
110……..	..101
111……..	..100

	Code binaire Naturel…….	..Code binaire réfléchi
	Sur 4 bits	
0	0000……	..0000
1	0001……	0001
2	0010……	..0011
3	0011……	..0010
4	0100……	..0110
5	0101……	..0111
6	0110……	..0101
7	0111……	..0100
8	1000……	..1100
9	1001……	..1101
10	1010……	..1111
11	**1011……**	**..1110**
12	1100……	..1010
13	1101……	..1011
14	1110……	..1001
15	1111……	..1000

Méthode

La ***valeur numérique*** d'un nombre binaire réfléchi s'obtient en donnant aux chiffres successifs pris de droite à gauche les poids 1,3,7,15,…2^{n+1} –1 et en effectuant la somme des produits non nuls, de signes alternés.

Exemple:

- 1011 → **+**15 -3 +1 = 13
- 0100→ +7 =7
- 10O1→ +15 -1 = 14
- **1110→ +15 -7 +3 = 11**
- 0111→…… = 5

} voir le tableau ci-dessus

Autre méthode:

Pour trouver l'expression d'un nombre binaire dans le code réfléchi, on l'additionne sans effectuer la retenue, avec le nombre obtenu en le décalant vers la gauche d'un rang et on abandonne le chiffre du plus petit poids.

Exemple:

```
    1 0 1 1
+   1 0 1 1
-----------
  1 1 1 0 1
```

→ **1110** en code réfléchi correspond à $(11)_{10}$

c. code décimaux

On code chaque chiffre (0-9) en binaire sur 4 bits ($2^3 < 10 \leq 2^4$). Ce code est appelé **DCB**: (**D**écimal **C**odé en **B**inaire) en anglais **BCD**: **B**inary **C**oded **D**ecimal

$(1297)_{10} = (0001\ 0010\ 1001\ 0111)_{BCD}$

d. complément à 2

Question :comment représenter un nombre négatif en représentation binaire?

- arithmétique binaire

somme avec retenue

a	b	*S*	*R*
0	0	0	0
0	1	1	0
1	0	1	0
1	1	0	1

S=a⊕b

R=a•b

Produit

a	b	*P*
0	0	0
0	1	0
1	0	0
1	1	1

P=a•b

"NON" Complément

a	$\bar{a}$
0	1
1	0

- représentation en complément à 2

soit A = $a_{n-1}\ a_{n-2}\ ....a_0$ un nombre binaire à n bits.

A :		a_{n-1}	a_{n-2} ...	a_0
C :		$\bar{a}_{n-1}$	$\bar{a}_{n-2}$ ...	$\bar{a}_0$
A+C:		1	1 ...	1
+1				1
A+C+1	1	0	0 ...	0

A+C+1 = 2^n, or on travaille sur n bits, et 2^n est représenté par n zéro, on a alors une représentation unique de 0.

B= 2^n – A est appelé le *complément à 2* du nombre A.

A +B s'écrit 0 sur n bits.

B=2^n –A =C+1

Conclusion:

Pour avoir la représentation d'un nombre négatif en complément à 2, on **complémente** tous les bits et on **ajoute 1**

Exemple: code binaire signé sur 4 bits.

	Positif	négatif		Formule: Complément à 1 +1
0	.....0000.....	0000 !!!.....		1111 + 1= 1(0000)
1	.....0001.....	1111.....	-1	1110 +1 =1111
2	.....0010.....	1110.....	-2	1101 +1 =1110
3	.....0011.....	1101.....	-3	1100 +1 =1101
4	.....0100.....	1100.....	-4	1011 +1 =1100
5	.....0101.....	1011.....	-5	1010 +1 =1011
6	.....0110.....	1010.....	-6	1001 +1 =1010
7	.....0111.....	1001.....	-7	1000 +1 =1001
	Pas de 8 !!	1000.....	-8	(-8)

Idem sur n bits: A= $a_{n-1}a_n ...a_0$ on complémente à 1 A puis on additionne 1.

e. nombres fractionnaires

Rappel:

Soit une base b associée à b symboles $\{S_0, S_1, S_2, ..., S_{b-1}\}$

Un nombre positif N dans un système de base b s'écrit sous la forme polynomiale:

$$N = a_{n-1} \cdot b^{n-1} + a_{n-2} \cdot b^{n-2} + \cdots + a_1 \cdot b^1 + a_0 \cdot b^0 + a_{-1} \cdot b^{-1} + a_{-2} \cdot b^{-2} + \cdots + a_{-m+1} \cdot b^{-m+1} + a_{-m} \cdot b^{-m}$$

La représentation simple de position est la suivante:

$$\left(a_{n-1}a_{n-2}a_1a_0, a_{-1}a_{-2}a_{-m+1}a_{-m}\right)$$

a_i est le chiffre de rang i (a_i appartient à un ensemble de b symboles)

a_{n-1} est le chiffre le plus significatif

a_{-m} est le chiffre le moins significatif

$\left(a_{n-1}a_{n-2}...a_0\right)$ partie entière

$\left(a_{-1}a_{-2}...a_{-m}\right)$ partie fractionnaire (<1)

Méthode:

On multiplie la partie fractionnaire par la base en répétant l'opération sur la partie fractionnaire du produit jusqu'a ce qu'elle soit nulle (ou que la précision voulue soit atteinte).

Pour la partie entière, on procède par divisions comme pour un entier.

Exemple : conversion de $(54,25)_{10}$ en base 2

Partie entière : $(54)_{10} = (110110)_2$ par divisions.

Partie fractionnaire :

0,25 x 2 = 0,50 → $a_{-1} = 0$

0,50 x 2 = 1,00 → $a_{-2} = 1$

0,00 x 2 = 0,00 → $a_{-3} = 0$

$(54,25)_{10} = (110110,01)_2$

Autre exemple : $(0,45)_{10}$ en base 2 **?**

0,45 * 2 = 0,90	0
0,90 * 2 = 1,8	1
0,8 * 2 = **1,6**	1
0,6 * 2 = 1,2	1
0,2 * 2 = 0,4	0
0,4 * 2 = 0,8	0
0,8 * 2 = **1,6**	1
0,6 * 2 = 1,2	

$(0,45)_{10} = (0,0111001...)_2$!!!

NB: Une longueur finie en base 10 peut être infinie en base B
On conserve la précision relative 10^{-3} est approximée par 2^{-10}

f. représentation des nombres réels

Le codage en complément à deux sur *n* bits ne permet de représenter qu'un intervalle de 2^n valeurs. Pour un grand nombre d'applications, cet intervalle de valeurs est trop restreint. La représentation à virgule flottante (*floating-point*) a été introduite pour répondre à ce besoin.
Pour des mots de 32 bits,:

- la représentation en ***complément à deux*** permet de coder un intervalle de $\boldsymbol{2^{32}}$ ***valeurs***
- tandis que la représentation à ***virgule flottante*** permet de coder un intervalle d'environ $\boldsymbol{2^{255}}$ ***valeurs***.

La représentation en virgule flottante a été normalisée (norme IEEE 754

Figure 1. Représentation des nombres à virgule flottante dans la norme IEEE 754

Signe	***Exposant***	***Fraction***
S	*e*	*f*

Nombre de bits	**Taille de *s***	**Taille de *f***	**Taille de *e***	***Emin***	***Emax***
32(simple précision)	1	23	8	- 126	127
64 (double précision)	1	52	11	-1022	1023

Dans cette représentation, la valeur d'un nombre sur **32 bits** est donnée par l'expression

$$(-1)^{s} \times \left(1 + \sum_{i=1}^{23} f_i 2^{-i}\right) \times 2^{e - E_{ma}}$$

où *fi* correspond au *i*ème bit de la fraction *f*.

Exemple : ***n*=32 bits**:
A = **1** 10000100 01010000000000000000000
S= 1 (-) , e=132-127=5, f= $2^{-2}+2^{-4}$=0,25+0,0625=0,3125
A= **-** 1,3125x2^5

2. Codes alphanumériques

Les codes "alphanumériques" sont des codes destinés à la transmission d'informations quelconques; ils ont donc à représenter au moins 36 caractères (10 chiffres plus 26 lettres). Comme 36 est compris entre 2^5 et 2^6, ils devront comporter au moins 6 bits. En fait, ils sont souvent à 8 bits, d'une part pour avoir une certaine souplesse d'utilisation (codes de commande réservés),d'autre part pour permettre la détection des erreurs (avec un bit de parité).

a. codes ASCII

Le code ASCII (American Standard Code for Information Interchange) comporte 7 bits d'informations et 1 bit de parité. Il est utilisé en particulier pour l'échange d'informations entre une unité centrale et des périphériques en informatique (clavier, imprimante,..)
Exemple:

Code ASCII	Caractère
01000001	"A"
01000010	"B"
11000011	"C"
……	…..
11100001	"a"

Bit de parité ↑

Code ASCII	Caractère
00110000	"0"
10110001	"1"
…..	…

Plusieurs points importants à propos du code ASCII :

- Les codes compris entre 0 et 31 ne représentent pas des caractères, ils ne sont pas affichables. Ces codes, souvent nommés *caractères de contrôles* sont utilisés pour indiquer des actions comme passer à la ligne (CR, LF), émettre un bip sonore (BEL), etc.
- Les lettres se suivent dans l'ordre alphabétique (codes 65 à 90 pour les majuscules, 97 à 122 pour les minuscules), ce qui simplifie les comparaisons.
- On passe des majuscules aux minuscules en modifiant le 5ième bit, ce qui revient à ajouter $32=2^5$ au code ASCII décimal.
- Les chiffres sont rangés dans l'ordre croissant (codes 48 à 57), et les 4 bits de poids faibles définissent la valeur en binaire du chiffre.

Décimal	Hexa	Binaire	Caractère		Décimal	Hexa	Binaire	Caractère
0	0	X0000000	NUL		32	20	X0100000	ESPACE
1	1	X0000001			33	21	X0100001	!
2	2	X0000010	STX		34	22	X0100010	"
3	3	X0000011	ETX		35	23	X0100011	#
4	4	X0000100	EOT		36	24	X0100100	$
5	5	X0000101			37	25	X0100101	%
6	6	X0000110	ACK		38	26	X0100110	&
7	7	X0000111	BEL		39	27	X0100111	'
8	8	X0001000			40	28	X0101000	(
9	9	X0001001			41	29	X0101001	)
10	A	X0001010	LF		42	2A	X0101010	*
11	B	X0001011			43	2B	X0101011	+
12	C	X0001100			44	2C	X0101100	,
13	D	X0001101	CR		45	2D	X0101101	-
14	E	X0001110			46	2E	X0101110	.
15	F	X0001111			47	2F	X0101111	/
16	10	X0010000			48	30	X0110000	0
17	11	X0010001			49	31	X0110001	1
18	12	X0010010			50	32	X0110010	2
19	13	X0010011			51	33	X0110011	3
20	14	X0010100	NAK		52	34	X0110100	4
21	15	X0010101			53	35	X0110101	5
22	16	X0010110			54	36	X0110110	6
23	17	X0010111			55	37	X0110111	7
24	18	X0011000			56	38	X0111000	8
25	19	X0011001			57	39	X0111001	9
26	1A	X0011010			58	3A	X0111010	:
27	1B	X0011011			59	3B	X0111011	;
28	1C	X0011100			60	3C	X0111100	<
29	1D	X0011101			61	3D	X0111101	=
30	1E	X0011110			62	3E	X0111110	>
31	1F	X0011111			63	3F	X0111111	?

Décimal	Hexa	Binaire	Caractère		Décimal	Hexa	Binaire	Caractère
64	40	**X**1000000	@		96	60	**X**1100000	`
65	41	**X**1000001	A		97	61	**X**1100001	a
66	42	**X**1000010	B		98	62	**X**1100010	b
67	43	**X**1000011	C		99	63	**X**1100011	c
68	44	**X**1000100	D		100	64	**X**1100100	d
69	45	**X**1000101	E		101	65	**X**1100101	e
70	46	**X**1000110	F		102	66	**X**1100110	f
71	47	**X**1000111	G		103	67	**X**1100111	g
72	48	**X**1001000	H		104	68	**X**1101000	h
73	49	**X**1001001	I		105	69	**X**1101001	i
74	4A	**X**1001010	J		106	6A	**X**1101010	j
75	4B	**X**1001011	K		107	6B	**X**1101011	k
76	4C	**X**1001100	L		108	6C	**X**1101100	l
77	4D	**X**1001101	M		109	6D	**X**1101101	m
78	4E	**X**1001110	N		110	6E	**X**1101110	n
79	4F	**X**1001111	O		111	6F	**X**1101111	o
80	50	**X**1010000	P		112	70	**X**1110000	p
81	51	**X**1010001	Q		113	71	**X**1110001	q
82	52	**X**1010010	R		114	72	**X**1110010	r
83	53	**X**1010011	S		115	73	**X**1110011	s
84	54	**X**1010100	T		116	74	**X**1110100	t
85	55	**X**1010101	U		117	75	**X**1110101	u
86	56	**X**1010110	V		118	76	**X**1110110	v
87	57	**X**1010111	W		119	77	**X**1110111	w
88	58	**X**1011000	X		120	78	**X**1111000	x
89	59	**X**1011001	Y		121	79	**X**1111001	y
90	5A	**X**1011010	Z		122	7A	**X**1111010	z
91	5B	**X**1011011	[		123	7B	**X**1111011	
92	5C	**X**1011100	\		124	7C	**X**1111100	|
93	5D	**X**1011101	]		125	7D	**X**1111101	
94	5E	**X**1011110	^		126	7E	**X**1111110	~
95	5F	**X**1011111	_		127	7F	**X**1111111	

NB: X est le bit de parité

b. code E.I.A

Le code **EIA** (**E**lectronic **I**ndustries **A**ssociation) comporte également 7 bits d'informations et 1 bit de parité. Il est plus particulièrement utilisé dans la commande numérique-outils.

Exemple:

Code E.I.A	Caractère	
01110001	"A"	1: parité

Bit de parité ↑

c. code Unicode

Au lieu d'utiliser seulement les codes 0 à 127, il utilise des codes de valeurs bien plus grandes.

Le code UNICODE permet de représenter tous les caractères spécifiques aux différentes langues. De nouveaux codes sont régulièrement attribués pour de nouveaux caractères: caractères latins (accentués ou non), grecs, cyrillics, arméniens, arabe, hébreux,…

Caractéristiques du code(mis au point en 1991):

- 16 bits pour représenter 65 536 caractères (0 a 65 535)
- Compatible avec ASCII
- Code la plupart des alphabets : Arabe, Chinois, ….
- On en a défini environ 50 000 caractères pour l'instant..

d. le Code Barre

Ce principe de codage, apparu dans les années 80, est largement utilisé sur les produits de grande consommation, car il facilite la gestion des produits.

Le marquage comporte un certain nombre de barres verticales ainsi que 13 chiffres :

- Le 1er chiffre désigne le pays d'origine : 3 = France, 4 = Allemagne, 0 = U.S.A, Canada etc. …
- Les cinq suivants sont ceux du code « fabricant »,
- Les six autres sont ceux du code de l'article,
- Le dernier étant une clé de contrôle
- Les barres représentent le codage de ces chiffres sur 7 bits, à chaque chiffre est attribué un ensemble de 7 espaces blancs ou noirs.

Site Mr. KHATORY: **http://e-khatory.com**