Dans la vie de tous les jours, nous sommes entourés de valeurs analogiques. L'homme a vite choisi d'utiliser des **grandeurs numériques**, faciles à traiter et à transmettre. Selon l'usage que l'on souhaite en faire, un codage trouve une application plus appropriée par rapport à tel autre : par exemple, nous avons une base de 10 chiffres pour compter (0.. 9), une base 24 ou 60 pour nous repérer dans le temps (heures, minutes)…

I. Les systèmes de numération

On a l'habitude de compter l'argent, les quantités, les masses, les distances… dans la base 10. On pourrait décomposer par exemple 1324€ en 1324 = 1000 + 300 + 20 + 4 = 1 x 10^3 + 3 x 10^2 + 2 x 10^1 + 4 x 10^0. On peut généraliser les décompositions de la base 10 par : $abcd_{10}$ = a x 10^3 + b x 10^2 + c x 10^1 + d x 10^0. Cette généralisation fonctionne pour toutes les bases :

$$abcd_n = a \times n^3 + b \times n^2 + c \times n^1 + d \times n^0$$

II. Correspondance de quelques bases usuelles

Décimal	Binaire naturel (8bits)	Hexadécimal
0	0000 0000	0
1	0000 0001	1
2	0000 0010	2
3	0000 0011	3
4	0000 0100	4
5	0000 0101	5
6	0000 0110	6
7	0000 0111	7
8	0000 1000	8
9	0000 1001	9
10	0000 1010	A
11	0000 1011	B
12	0000 1100	C
13	0000 1101	D
14	0000 1110	E
15	0000 1111	F
16	0001 0000	10

III. Les changements de base

De la base 2 vers la base 10 :

$1\ 0110_{(2)}$ =

De la base 10 vers la base 2 :

$120_{(10)}$ =

$120_{(10)}$ =

De la base 2 vers la base 16 :

$10\ 1011_{(2)}$ =

De la base 16 vers la base 2 :

$EDF_{(16)}$ =

De la base 16 vers la base 10 :

$F8A_{(16)}$ =

De la base 10 vers la base 16 :

$124_{(10)}$ =

IV. Autres codages utilisés

Binaire signé :

Nous avons jusqu'à présent parlé de nombres entiers naturels. Ils ne peuvent par nature qu'être positifs ou nuls. Envisageons maintenant les **nombres entiers relatifs** ou autrement dit, munis d'un signe '+' ou '-'. Le problème est que les circuits électroniques digitaux ne peuvent enregistrer que des 0 ou des 1 mais pas de signes + ou -. Le seul moyen est alors de convenir que si un nombre est susceptible d'être négatif on lui **réserver un bit** pour indiquer le signe.

Le bit le plus à gauche du mot binaire est celui qui va représenter le signe. Signe **négatif** si ce bit vaut 1, signe **positif** quand ce bit vaut 0. Si on admet que le nombre peut représenter des valeurs négatives, on parle de nombres "signés".

Exemple :

- codage binaire naturel 8 bits de $21_{(10)}$: $0001\ 0101_{(2)}$
- codage binaire signé 8 bits de + $21_{(10)}$: $0001\ 0101_{signé}$
- codage binaire signé 8 bits de - $21_{(10)}$: $1110\ 1011_{signé}$

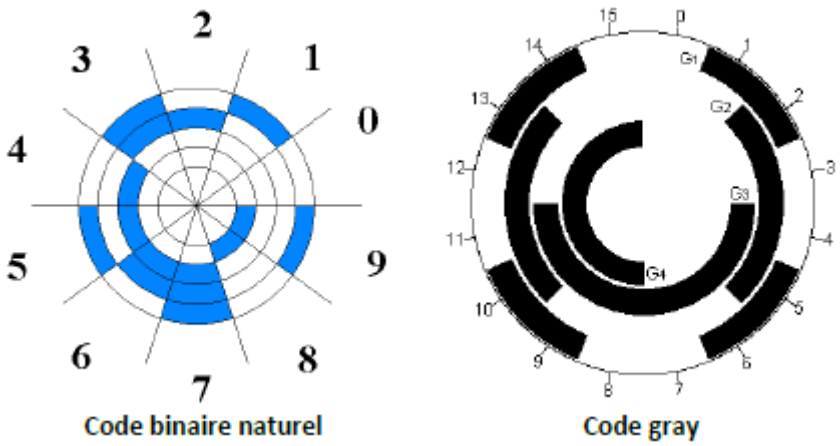
Le calcul du code pour les nombres négatifs se fait en deux étapes :

- Remplacer tous les 0 par des 1, et tous les 1 par des 0 ;
- Ajouter 1 (+1) au code précédent.

Code gray :

C'est un codage qui a la particularité de ne **changer qu'un bit lors du passage d'une valeur à sa suivante**. Cela permet dans le codage des systèmes en rotation (antenne radar par exemple) d'éviter les erreurs de synchronisation de lecture d'un codage binaire naturel lorsque plusieurs changent à la fois.

Exemple :



Le fait de modifier plusieurs bits lors d'une simple incrémentation peut mener, selon le circuit logique, à **un état transitoire indésirable** dû au fait que le chemin logique de chaque bit dispose d'un délai différent. Ainsi, lors du passage de la valeur "01" à la valeur "10" en binaire naturel, il est possible d'observer un état transitoire "00" si le bit de droite commute en premier ou "11" dans le cas contraire. Si le circuit dépendant de cette donnée n'est pas synchrone, l'état transitoire peut perturber les opérations en faisant croire au système qu'il est passé par un état normalement non atteint à ce stade.

Code BCD :

Le décimal codé binaire (**B**inary **C**oded **D**ecimal en Anglais) est un système de numération se rapprochant de la représentation humaine usuelle, en base 10. Il suffit de **remplacer chaque chiffre décimal par son image binaire** codé sur quatre bits.

Exemple :

$2018_{(10)} = 0010\ 0000\ 0001\ 1000_{(BCD)}$

Code ASCII :

Le codage ASCII (American Standard Code for Information Interchange) est devenu au fil du temps le standard pour coder les informations alphanumériques et autres caractères de commande. C'est un codage sur sept bits qui se présente de la façon suivante :

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

Lors des communications (Internet par exemple) un **bit de parité** est ajouté, ce qui donne un ensemble de huit bits qui représente le codage complet d'un caractère. Ce bit de parité est mis à zéro si la somme des autres bits est paire, et à un si elle est impaire, pour détecter des erreurs de transmission.

Exemple :

Code hexadécimal transmis si on tape les lettres BONJOUR :

$B = 42_{(16)} = 0100\ 0010_{(2)} = 42_{(16)}$
$O = 4F_{(16)} = 0100\ 1111_{(2)} \Rightarrow 1100\ 1111_{(2)} = CF_{(16)}$
$N = 4E_{(16)} = 0100\ 1110_{(2)} = 4E_{(16)}$
$J = 4A_{(16)} = 0100\ 1010_{(2)} \Rightarrow 1100\ 1010_{(2)} = CA_{(16)}$
$O = 4F_{(16)} = 0100\ 1111_{(2)} \Rightarrow 1100\ 1111_{(2)} = CF_{(16)}$
$U = 55_{(16)} = 0101\ 0101_{(2)} = 55_{(16)}$
$R = 52_{(16)} = 0101\ 0010_{(2)} \Rightarrow 1101\ 0010_{(2)} = D2_{(16)}$

On transmet donc en série la suite 42 CF 4E CA CF 55 D2