SOMMAIRE AUTOMATIQUE

1) Objectifs

A l'issue de ce cours, vous connaîtrez les principaux codages utilisés en programmation :

- binaire
- hexadécimal
- BCD
- Virgule flottante
- ASCII

D'autres codages seront abordés :

- Gray
- ASCII
- Détection d'erreur
- CODE BARRE

2) Les nombres en base décimale (base 10)

Tels Monsieur Jourdain, vous pratiquez la base 10 sans le savoir.

2.1) Naturels, relatifs et décimaux (avec virgule)

Nous nous intéresserons à trois ensembles de nombres :

- les entiers naturels { 0, 1, 2, 3, … }
- les entiers relatifs { …, -2, -1, 0, 1, 2, 3, … }
- les nombres à virgule

2.2) Base, rang et poids

La **BASE** d'un système de numération est la quantité de symboles différents qu'utilise ce système.
En base 10, les symboles disponibles sont {0, 1, 2, 3, 4, 5, 6, 7, 8, 9}.
Chaque chiffre a un **POIDS** selon son **RANG**. Ainsi dans le nombre à virgule 9486,25 '9' est le chiffre des milliers, de poids 10^3, etc.

rang	Poids	
3	Millier	$10^3 = 1000$
2	Centaine	$10^2 = 100$
1	Dizaine	$10^1 = 10$
0	Unité	$10^0 = 1$
-1	Dixième	$10^{-1} = 0,1$
-2	Centième	$10^{-2} = 0,01$

2.3) Représentation polynomiale

Tout nombre en base 10 peut s'écrire sous la forme d'un polynôme :
Soit N, un nombre entier de n chiffres $N = a_{n-1}a_{n-2}.....a_2a_1a_0$

$$N = \sum_{i=0}^{n-1} a_i \times B^i$$

Avec ***i*** : rang du chiffre **B** : base de numération

a_i : chiffre compris entre 0 et 9

Ainsi la décomposition polynomiale de 4201,3 est :

$4201,3 = a_3 \cdot 10^3 + a_2 \cdot 10^2 + a_1 \cdot 10^1 + a_0 \cdot 10^0 + a_{-1} \cdot 10^{-1}$

avec $a_3 = 4$, $a_2 = 2$, $a_1 = 0$, $a_0 = 1$, et $a_{-1} = 3$.

Exercice :
Donner la représentation polynomiale de N = 5 670 220

On donne $a_7 = 3$, $a_4 = 2$, les autres chiffres étant nuls. Que vaut N ?

3) Les codes binaires pondérés

3.1) Codage des entiers naturels

Il s'agit des nombres entiers positifs à partir de 0 : { 0, 1, 2, 3, … }.

3.1.1) Le système binaire naturel

En base 2, les symboles disponibles sont {0,1}.
L'élément le plus simple est le **BINARY DIGIT**, appelé communément le BIT.
Il ne prend que les deux valeurs 0 et 1.
La représentation usuelle en programmation est sur 8 bits, aussi appelé octet.
Le poids de chaque chiffre est alors compris entre 2^7 et 2^0 :

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

Le bit de gauche est le MSB : **Most Significant Bit**, bit de poids fort
Celui de droite est le LSB : **Least Significant Bit**, bit de poids faible.
Ces huit bits forment un OCTET : **Byte** en anglais.

Avec un octet, on peut coder des entiers naturels de 0 à 255.

En base B avec n chiffres, on peut écrire B^n nombres différents de 0 à $B^n - 1$
Selon l'environnement de travail utilisé (programmation en Assembleur, en Basic, en langage C, …) plusieurs notations existent pour identifier un nombre écrit en base 2 :

0011 1001$_{(2)}$ = %00111001 = 00111001B

Exercice : Compléter le tableau

base 10	Binaire naturel sur 8 bits
0	%0000 0000
1	
2	
3	
4	
5	
6	
7	
8	
9	
10	
11	
12	
13	
14	
15	
16	%0001 0000
17	
32	
64	
127	
128	
129	
255	%1111 1111

Remarque : Avec 8 bits , La valeur maxi est 255.

256 et plus s'écrivent alors avec un 9^e bit à 1 (appelé aussi bit de retenue)

3.1.2) Le système octal
L'octal est la base 8.
Les symboles disponibles sont {0, 1, 2, 3, 4, 5, 6, 7}.

Dans ce système, le poids est une puissance de 8.

Exemple : $N = 6548_{(8)}$

$N = 6\times8^3 + 5\times8^2 + 4\times8^1 + 8\times8^0$

$N = 3432_{(10)}$

Pour reconnaître un nombre en octal, plusieurs notations existent :

$$175_{(8)} = @175 = 175Q = 0175$$

Dans la pratique, ce système est rarement rencontré.

3.1.3) Le système hexadécimal

Le système hexadécimal est la base 16.
Les symboles disponibles sont {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}.
Les lettres A à F correspondent aux nombres 10 à 15 en base 10.
Dans ce système, le poids est une puissance de 16.

Exemple : $N = AC53_{(16)}$

$N = A\times16^3 + C\times16^2 + 5\times16^1 + 3\times16^0$

$N = 10\times16^3 + 12\times16^2 + 5\times16^1 + 3\times16^0$

$N = 44115_{(10)}$

Selon l'environnement de travail utilisé (programmation en assembleur, en basic, en C, …), plusieurs notations existent pour identifier un nombre écrit en base 16 :

$$F6B1_{(16)} = \$F6B1 = F6B1H = 0xF6B1$$

Exercice :

$N = F5D3_{(16)} =$

$N = 1F0B_{(16)} =$

$N = 10\times16^3 + 12\times16^2 + 5\times16^1 + 3\times16^0 =$

3.1.4) Le BCD

Le BCD (**Binary Coded Decimal**) est un système intermédiaire entre la base 10 et le binaire : chaque chiffre de la base 10 est codé sur 4 bits.

Exemple : $9847_{(10)}$ s'écrit $1001\ 1000\ 0100\ 0111_{(BCD)}$.

Les codes binaires compris entre $1010_{(2)}$ et $1111_{(2)}$ n'existent pas en BCD.

Exercice :
Dans un automate programmable industriel, la date et l'heure sont disponibles en mémoire au format BCD.
Le jour est écrit sur 4 octets selon le format JJMMAAAA.
L'heure est sur 3 octets selon le format HHMMSS.

Que contiendrait ces 7 octets à l'instant présent ?

3.1.5) Changements de base

3.1.5.1) Tableau de correspondance

Le tableau des correspondances entre les quatre principaux systèmes, décimal / binaire / hexadécimal / BCD, est à connaître parfaitement.

Base 10	Binaire (base 2)	Hexadécimal (base 16)	BCD
0	%0000 0000	$00	0000 0000
1			
2			
3			
4			
5			
6			
7			
8			
9			
10			
11			
12			
13			
14			
15			
16	%0001 0000	$10	%0001 0110
17			

3.1.5.2) Conversion du décimal vers une autre base

Un nombre N étant donné en base 10, comment trouver son écriture dans une base B quelconque ? Les deux méthodes proposées effectuent des divisions euclidiennes successives.

Première méthode (par soustraction) : du poids fort vers le poids faible

Nous allons convertir N = $3786_{(10)}$ en hexadécimal (base 16).
Le première division est par la puissance de 16 la plus élevée présente dans N.

La plus grande puissance de 16 contenue dans N est 16^2.
En effet, $16^2 = 256 < N$ et $16^3 = 4096 > N$.

3786	256
202	14

La division euclidienne de N par 16^2 montre que $N = 14 \times 16^2 + 202$.
Il reste 202.

202	16
10	12

La division euclidienne du reste précédent par 16^1 montre que $202 = 12 \times 16^1 + 10$.

En résumé, nous avons $N = 14 \times 16^2 + 12 \times 16^1 + 10 \times 16^0$,
Ou encore $N = E \times 16^2 + C \times 16^1 + A \times 16^0$,
Donc N = $ECA_{(16)}$

Deuxième méthode (par division) : du poids faible vers le poids fort

Nous allons à nouveau convertir N = $(3786)_{10}$ en hexadécimal (base 16).
Ici, nous enchaînons les divisions par 16.

3786	16	
10	236	16
	12	14

Après trois divisions successives par 16, le quotient est inférieur au diviseur donc on arrête les divisions.
Il vient :
$N = 14 \times 16^2 + 12 \times 16^1 + 10 \times 16^0$,
Donc N = $ECA_{(16)}$

Exercice :

a) traduire en hexadécimal 41, 73, 625, 1028 et 23222

b) ramener en base 10 les nombres $23, $100, $A010, $BC00, $FE00

3.1.5.3) Autres conversions

Les conversions entre bases binaires, octales et hexadécimales se font directement de droite à gauche par paquets de 3 ou 4 bits.
conversion d'octal à binaire : *@57 = %00101111*
conversion de binaire à octal : *%10111100 = @274*
conversion d'hexadécimal en binaire : *$FC = %11111100*
conversion de binaire en hexadécimal : *%00111011 = $3B*

$B = 2^k \rightarrow 2$
codage de chaque chiffre de la base B en binaire sur k bits
$2 \rightarrow B = 2^k$
regroupement de k bits écrit en base B

Mise en forme : Puces et numéros

Pour s'entraîner aux conversions, voir les exercices page 8 du livre ELECTRONIQUE NUMERIQUE

3.2) Codage des entiers relatifs

Il s'agit des nombres entiers signés (binaire signé).
3.2.1) Le complément à deux (CPL2) et le binaire signé
Pour coder les entier relatifs, on opère comme suit
Soit N, un nombre entier relatif de n bits $N = a_{n-1}a_{n-2}.....a_2a_1a_0$
Son codage en binaire signé répond à la relation

$$N = -a_{n-1} \times 2^{n-1} + \sum_{i=0}^{n-2} a_i \times 2^i$$

Compléter le tableau

base 10	Binaire signé sur 8 bits	Hexadécimal signé sur 2 chiffres
-128	$1000\ 0000_{(BS)}$	$\$80_{(HS)}$
-127		
-3		
-2		
-1		
0	$0000\ 0000_{(BS)}$	$00_{(HS)}$
1		
2		
3		
127	$0111\ 1111_{(BS)}$	$7F_{(HS)}$

Le bit de poids fort renseigne sur le signe du nombre : c'est le BIT DE SIGNE. Il est à 1 pour les nombres négatifs.

Exercice
Un entier signé codé sur un octet (8 bits) peut prendre des valeurs entre –128 ($80_{(HS)}$.) et 127 ($7F_{(HS)}$.).
Quelles seront les valeurs possibles sur deux octets (16 bits) ?

Et sur quatre octets (32 bits) ?

Pour passer de $N_{(BS)}$ *à* $-N_{(BS)}$ et réciproquement, on peut utliser la méthode du COMPLEMENT A 2 (CPL2).
Chercher le CPL2 d'un nombre binaire signé revient à changer son signe :

$CPL2[+N_{(BS)}] = -N_{(BS)}$ $\quad CPL2[-N_{(BS)}] = +N_{(BS)} \quad \rightarrow \quad -[-N_{(BS)}] = +N_{(BS)}$

Deux opérations sont nécessaires :
- Complémenter tous les bits ($1 \leftrightarrow 0$)
- Ajouter 1 (addition en binaire)

Exemple : recherche du codage hexadécimal signé de (–17)
Le codage binaire signé de +17 est 0001 0001$_{(BS)}$*…..*
Son complément bit à bit est %1110 1110.
On ajoute 1 : %1110 1110 + %0000 0001 = %1110 1111.
Résultat : –17 = 1110 1111$_{(BS).}$ *=* $EF_{(HS)}$.

Ce codage a la propriété d'être cyclique.

Dernière propriété du codage CPL2 : il permet l'addition d'entiers signés (voir plus loin).

Exercice : vérifiez que le codage hexadécimal signé de (–17) = $EF_{(HS)}$. est bien (+17)

3.2.2) Addition et soustraction d'entiers relatifs

Sous réserve de dépassement de capacité, l'*addition* d'entiers signés est immédiate si les nombres négatifs sont codés en CPL2.
La *soustraction* consiste à ajouter le CPL2 du nombre.
Le bit de signe renseigne sur la polarité du résultat.

– Addition de deux nombres positifs

$$\begin{array}{rl} & (+17) = 0001\ 0001_{(BS)} \\ + & (+12) = 0000\ 1100_{(BS)} \\ \hline = & (+29) = 0001\ 1101_{(BS)} \end{array}$$

– Addition de deux nombres de signes contraires

$$\begin{array}{rl} & (+17) = 0001\ 0001_{(BS)} \\ + & (-12) = 1111\ 0100_{(BS)} \\ \hline = & (+29) = 0000\ 0101_{(BS)} \end{array}$$

$$\begin{array}{rl} & (+17) = 0001\ 0001_{(BS)} \\ + & (-33) = 1101\ 1111_{(BS)} \\ \hline \end{array}$$

– Addition de deux nombres négatifs

$$\begin{array}{rl} & (-12) = 1111\ 0100_{(BS)} \\ + & (-33) = 1101\ 1111_{(BS)} \\ \hline = & (-45) = 1101\ 0011_{(BS)} \end{array}$$

Pour s'entraîner au CPL2, voir les exercices 8, 9 et 10, page 8 du livre ELECTRONIQUE NUMERIQUE

3.2.3) Multiplication et division

L'opération (X) · (Y) entre deux entiers positifs se pose comme une *multiplication* traditionnelle en base 10, avec les règles suivantes :

- $0 \cdot 0 = 0$
- $0 \cdot 1 = 0$
- $1 \cdot 0 = 0$
- $1 \cdot 1 = 1$

```
          %00101101
x         %01000011
-----------------
          00101101
         00101101
        00101101 . . . .
-----------------
%0000101111000111
```

Pour des entiers négatifs, il sera nécessaire de les changer de signes avant d'effectuer l'opération et, le cas échéant, de prendre le CPL2 du résultat.

La *division* est identique à une division euclidienne entre deux entiers positifs en base 10.

```
  110110 | 1010
- 1010   | 101
-------  |
  001110 |
- 1010   |
-------  |
  0100   |
```

Pour des entiers négatifs, l'approche est similaire à celle de la multiplication. Il est nécessaire de les changer de signe et, selon les cas, de prendre le CPL2 du résultat.

3.3) Codage des nombres à virgule

3.3.1) Virgule flottante

Les nombres à virgule sont décomposés en une mantisse et un exposant.
Exemples en base 10 :

72	$= 0{,}72 \cdot 10^{2}$	*mantisse 0,72*	*exposant 2*
0,03	$= 0{,}3 \cdot 10^{-1}$	*mantisse 0,3*	*exposant négatif –1*
–250	$= -0{,}25 \cdot 10^{3}$	*mantisse négative –0,25*	*exposant 3*

La représentation binaire des nombres à virgule est très variable d'un environnement de programmation à l'autre. Pour information, voici la représentation sur 4 octets en ANSI C du type **Float** :
- 1bit de signe de la mantisse
- 23 bits de mantisse
- 1 bit de signe de l'exposant
- 7 bits d'exposant

Avec ce codage, les valeurs extrêmes sont approximativement $\pm 3{,}4 \cdot 10^{\pm 38}$

3.3.2) Nombre fractionnaires

Nous nous intéressons ici au codage des nombres à virgule compris entre 0 et 1 ($0 < N < 1$), tels que les mantisses vues précédemment.
En base 10, nous savons décomposer un tel nombre et le mettre sous forme de fraction.

Exemple: N= 0,8273 = $8\times10^{-1} + 2\times10^{-2} + 7\times10^{-3} + 3\times10^{-4} = \frac{8273}{10000}$

Plus généralement, dans une base B quelconque :
Soit N = $0, a_1 a_2 a_3 .....a_n$ (B)

La partie fractionaire est égale à $\sum_{i=1}^{n} a_i \times B^{-i}$

Où a_i est un chiffre tel que ° $0 \le a_i \le B-1$

Ainsi en base 2: N= $0{,}1011_{(2)} = 1\times2^{-1} + 0\times2^{-2} + 1\times2^{-3} + 1\times2^{-4} = \frac{11}{16} = 0{,}6875$

Comment convertir un nombre fractionnaire de la base 10 à la base B ?

Avec des nombres entiers, nous avions procédé par divisions successives. Ici, nous opérerons par multiplications successives pour faire apparaître les éléments a_i.

Exemple : Convertir N = $0{,}72145_{(10)}$ en hexadécimal sur 4 octets

$0{,}72145 \times 16 = 11{,}5432 \rightarrow a_1 = B$
$0{,}5432 \times 16 = 8{,}6912 \rightarrow a_2 = 8$
$0{,}6912 \times 16 = 11{,}0592 \rightarrow a_3 = B$
$0{,}0592 \times 16 = 0{,}9472 \rightarrow a_4 = 0$
$0{,}9472 \times 16 = 15{,}1552 \rightarrow a_5 = F$
$0{,}1552 \times 16 = 2{,}4832 \rightarrow a_6 = 2$
$0{,}4832 \times 16 = 7{,}7312 \rightarrow a_7 = 7$
$0{,}7312 \times 16 = 11{,}6992 \rightarrow a_8 = B$

$$N = 0{,}B8B0F27B_{(16)} = \frac{\mathbf{\$B8B0F27B}}{\mathbf{\$100000000}} = \frac{\mathbf{3098604155}}{\mathbf{16^8}} = \mathbf{0{,}72144999}$$

Exercice

1) Convertir en binaire sur 8 bits des nombres fractionnaires
N= 0,7

N=0,17

2) N = $0{,}72145_{(10)}$ en hexadécimal sur 4 octets

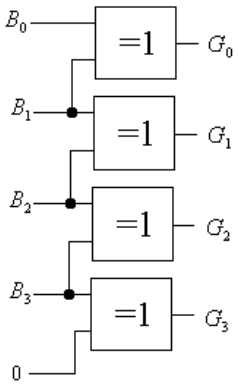
2) Convertir en décimal le nombre fractionnaire N=$0{,}243F6A8A_{(16)}$

4) Autres codes usuels

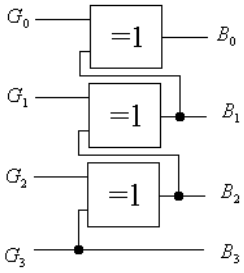
4.1) Le code GRAY

Le code binaire que nous avons vu jusqu'à maintenant s'appelle code binaire naturel. Il existe de nombreux autres codes dont le code GRAY (code binaire réfléchi).
Dans les conversions d'une grandeur analogique (par exemple la position d'un axe d'un moteur) en une grandeur numérique on a besoin d'un code dans lequel les grandeurs successives ne diffèrent que d'un bit. Cela évite des erreurs de détections. Ainsi, au passage de 7 à 8, les quatre bits changent. S'ils ne changent pas en même temps, on détectera des valeurs intermédiaires erronées :

$$0111 \rightarrow 0110 \rightarrow 0100 \rightarrow 0000 \rightarrow 1000$$



Binaire	entier	GRAY
$B_3\ B_2\ B_1\ B_0$		$G_3\ G_2\ G_1\ G_0$
0000	0	0000
0001	1	0001
0010	2	0011
0011	3	0010
0100	4	0110
0101	5	0111
0110	6	0101
0111	7	0100
1000	8	1100
1001	9	1101
1010	10	1111
1011	11	1110
1100	12	1010
1101	13	1011
1110	14	1001
1111	15	1000
10000	16	11000

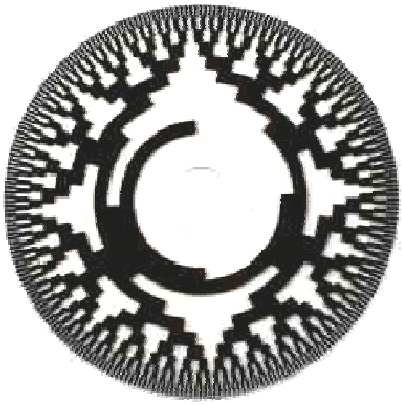


Le code GRAY est aussi appelé code binaire réfléchi : en effet, de nombreuses symétries horizontales apparaissent dans le tableau.

Pour construire une suite de nombres en code GRAY, il suffit de remarquer que pour obtenir la première colonne du tableau (G_0), nous avons écrit un fois 0 deux fois 1 deux fois 0 … et faire une observation similaire pour G_1, **…(Axes de symétries)**

Exercices :

a) Combien de cercles sur ce codeur de position angulaire absolu ?



b) Conversion de code binaire vers code Gray :
   - Exprimer G_3 en fonction de B_3
   - Exprimer G_2 en fonction B_3 et B_2
   - Exprimer G_1 en fonction B_2 et B_1
   - Exprimer G_0 en fonction B_1 et B_0

c) Conversion de code Gray vers code binaire :
   - Exprimer B_3en fonction de G_3
   - Exprimer B_2 en fonction B_3 et G_2
   - Exprimer B_1en fonction B_2 et G_1
   - Exprimer B_0 en fonction B_1 et G_0

4.2) Le code ASCII

Pour la mémorisation, l'édition et/ou la transmission de textes, il est nécessaire de pouvoir aussi coder sous forme binaire des caractères tels que les lettres de l'alphabet, la ponctuation, des symboles, des instructions de changement de ligne, de « retour chariot », etc.
Le code ASCII (American Standard Code for Information Interchange) a été adopté en Informatique.
Vous en trouverez un extrait p. 148 du livre ELECTRONIQUE NUMERIQUE.

Exercice :
Donner les codes ASCII correspondants au message « Hello world ! » .

caractères	*H*	*e*	*l*	*l*	*o*		*w*	*o*	*r*	*l*	*d*		*!*	CR	LF
Code ASCII															

Liste des codes ASCII
(d'après www.asciitable.com)

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

codes étendus

128	Ç	144	É	160	á	176	░	193	┴	209	╤	225	ß	241	±
129	ü	145	æ	161	í	177	▒	194	┬	210	╥	226	Γ	242	≥
130	é	146	Æ	162	ó	178	▓	195	├	211	╙	227	π	243	≤
131	â	147	ô	163	ú	179	|	196	─	212	╘	228	Σ	244	⌠
132	ä	148	ö	164	ñ	180	┤	197	┼	213	╒	229	σ	245	⌡
133	à	149	ò	165	Ñ	181	╡	198	╞	214	╓	230	µ	246	÷
134	å	150	û	166	ª	182	╢	199	╟	215	╫	231	τ	247	≈
135	ç	151	ù	167	º	183	╖	200	╚	216	╪	232	Φ	248	°
136	ê	152	_	168	¿	184	╕	201	╔	217	┘	233	Θ	249	·
137	ë	153	Ö	169	_	185	╣	202	╩	218	┌	234	Ω	250	·
138	è	154	Ü	170	¬	186	║	203	╦	219	█	235	δ	251	√
139	ï	156	£	171	½	187	╗	204	╠	220	▄	236	∞	252	_
140	î	157	¥	172	¼	188	╝	205	═	221	▌	237	ϕ	253	²
141	ì	158	_	173	¡	189	╜	206	╬	222	▐	238	ε	254	■
142	Ä	159	ƒ	174	«	190	╛	207	╧	223	▀	239	∩	255	
143	Å	192	└	175	»	191	┐	208	╨	224	α	240	≡		

4.3) Les codes à détection d'erreur

Ces codes permettent de détecter, et pour certains même de corriger des erreurs dues à des perturbations dans la transmission des données, le stockage, etc.
La technique de codage la plus simple consiste à transmettre avec chaque octet un neuvième bit, le BIT DE PARITÉ.
Dans le cas de la parité paire (**Even Parity**), ce neuvième bit est tel que la somme des bits à 1 transmis est un nombre pair. Ce sera l'inverse pour la parité impaire.
Soient à transmettre les caractères M et I, codés en ASCII %0100 1101 et %0100 1001. Les bits de parité seront respectivement 0 et 1.
Une erreur simple sera ainsi détectée par le destinataire qui demandera alors une retransmission des données.

Exercice :
Reprendre l'exercice sur le code ASCII, et calculer le bit de parité paire pour chaque caractère.

4.4) Les codes barres

Ces dernières années est apparu un système optique de codage numérique appelé CODE BARRE.

Parmi les codes existants, le code EAN 13 (**European Article Number**) est utilisé sur tous les produits de grande consommation. Il est décrit dans le MEMOTECH 4ème édition page 4.40 : un trait fin code 1, un espace fin code 0, un système de détection des erreurs est intégré dans les 13 caractères.