Chapitre 1

Les systèmes de numération et codes

I. Introduction

Le système décimal est malheureusement difficile à adapter aux mécanismes numériques, car il est difficile de concevoir du matériel électronique fonctionnant sur dix plages de tensions différentes. Le système de numération binaire est le plus important de ceux utilisés dans les circuits numériques. Il est le seul que ces circuits soit capable d'utiliser. Il ne faut pour autant ne pas négliger l'importance des autres systèmes. Le système décimal revêt de l'importance en raison de son utilisation universelle pour représenter les grandeurs du monde courant. De ce fait, il faudra parfois convertir des valeurs décimales en valeurs binaires avant de pouvoir les traiter dans un circuit numérique. Par exemple, lorsque vous composez un nombre décimal sur votre calculatrice (ou sur le clavier de votre ordinateur), les circuits internes convertissent ce nombre décimal en une valeur binaire.

Nous connaissons les systèmes binaire et décimal, mais il existe deux autres systèmes de numération très répandus dans les circuits numériques. Il s'agit des systèmes de numération octale (base de 8) et hexadécimal (base de 16) qui servent tous les deux au même but, soit celui de constituer un outil efficace pour représenter de gros nombres binaires. Comme nous le verrons, ces systèmes de numération ont l'avantage d'exprimer les nombres de façon que leur conversion en binaire, et vice versa, soit très facile.

Dans un système numérique, il peut arriver que plusieurs systèmes de numération cohabitent, d'où l'importance de pouvoir convertir un système dans un autre. Le présent chapitre se propose de vous montrer comment effectuer de telles conversions. Dans ce chapitre nous introduisons également certains codes binaires utilisés pour représenter divers genres d'information.

II. Représentation des nombres

Le nombre de symboles utilisés caractérise le numéro de la base. Celui que nous connaissons le mieux est le système décimal mais nous allons aussi définir les systèmes binaire, octale et hexadécimal.

Exemple :

- en base **10**, nous avons les **10 symboles** (0, 1,..,9) ;
- en base **2**, nous avons les **2 symboles** (0, 1) ;
- en base **8**, nous avons les **8 symboles** (0, 1,…, 7) ;
- en base **16**, nous avons besoin de **16 symboles**, nous utiliserons les 10 chiffres plus les lettres de A à F, soit 0, 1,…, 9, A, B, C, D, E, F.

Le poids d'un chiffre dépend de sa position dans le nombre. Nous parlons de numération de position, soit :

Un nombre dans une base "$\boldsymbol{b}$" entière positive s'écrit :

$$\boldsymbol{N_b = (a_{n-1}a_{n-2}\cdots a_1a_0a_{-1}a_{-2}\cdots a_{-m})_b}$$

ce qui correspond aux opérations :

$$N_b = a_{n-1}b^{n-1} + a_{n-2}b^{n-2} + a_{n-3}b^{n-3} + \ldots + a_2b^2 + a_1b^1 + a_0b^0 + a_{-1}b^{-1} + a_{-2}b^{-2} + \cdots + a_{-m}b^{-m}$$

L'indice de **b** indique la base dans laquelle le nombre est calculé.

II.1. Le système décimal ou base 10

Comporte 10 symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9. Pour le distinguer d'un autre système, on peut préciser la base d'un nombre en plaçant cette dernière en indice à la fin du nombre.

Ex. 2017 peut s'écrire $(2017)_{10}$

Ce nombre se décompose ainsi :

milliers	centaines	dizaines	unités
10^3	10^2	10^1	10^0
2	**0**	**1**	**7**

$(2017)_{10} = \mathbf{2} \cdot 10^3 + \mathbf{0} \cdot 10^2 + \mathbf{1} \cdot 10^1 + \mathbf{7} \cdot 10^0$

Le système décimal est malheureusement difficile à adapter aux mécanismes numériques, car il est difficile de concevoir du matériel électronique fonctionnant sur dix plages de tensions différentes.

II.2. Le système binaire naturel

II.2.1 Introduction

Les systèmes électroniques sont caractérisés par deux états :

- Interrupteur : **ouvert** ou **fermé**
- Transistor : **saturé** ou **bloqué**

De cette constatation est née l'idée d'utiliser le système binaire ou le système à base 2.

La base 2 n'utilise que deux symboles : **0** et **1**

Le système de numération binaire est un système de numération de position où le poids de chaque bit est un multiple de puissance de 2 (base) ;

EXEMPLE I.1 :

2^6	2^5	2^4	2^3	2^2	2^1	2^0	2^{-1}	2^{-2}	2^{-3}
1	**0**	**1**	**1**	**0**	**0**	**1,**	**0**	**1**	**1**

les différentes **puissances de 2** sont :

2^0	2^1	2^2	2^3	2^4	2^5	2^6	2^7	$\mathbf{2^8}$	2^9	$\mathbf{2^{10}}$	...
1	2	4	8	16	32	64	128	**256**	512	**1024**	...

un ensemble de **8 bits** est appelé « **octet** » (ou ***byte***).

II.2.2 Comptage binaire

On présente les nombres binaires en général avec un nombre fixe de bits, nombre imposé par les circuits mémoires utilisés pour représenter ces nombres.

Suite des nombres binaires à 4 bits :

Poids :	2^3	2^2	2^1	2^0	B_{10}
	0	0	0	0	0
	0	0	0	1	1
	0	0	1	0	2
	0	0	1	1	3
	0	1	0	0	4
	0	1	0	1	5
	0	1	1	0	6
	0	1	1	1	7
	1	0	0	0	8
	1	0	0	1	9
	1	0	1	0	10
	1	0	1	1	11
	1	1	0	0	12
	1	1	0	1	13
	1	1	1	0	14
	1	1	1	1	15

Le bit le plus significatif - le bit le plus à gauche- est appelé « **bit de poids fort** » ou **MSB** (**M**ost **S**ignificant **B**it).

Le bit le moins significatif - le bit le plus à droite- est appelé « **bit de poids faible** » ou **LSB** (**L**east **S**ignificant **B**it).

Si on utilise **N** bits, on peut représenter 2^N valeurs différentes de 2^0 à 2^{N-1}

EXEMPLE I.2 :

N = 8 ;

$0 \leftrightarrow 00000000 \rightarrow 11111111 \leftrightarrow (255)_{10}$

II.3. Le système octal

Le système de numération octal a comme base huit (**base 8**), ce qui signifie qu'il comprend **huit symboles** possibles, soit **0**, **1**, **2**, **3**, **4**, **5**, **6** et **7**. Ainsi, chaque chiffre dans un nombre octal a une valeur comprise entre 0 et 7. Voici les poids de chacune des positions d'un nombre octal.

…	8^3	8^2	8^1	8^0	.	8^{-1}	8^{-2}	8^{-3}	…

EXEMPLE I.3 :

$$(127,65)_8 = 1.8^2 + 2.8^1 + 7.8^0 + 6.8^{-1} + 5.8^{-2} = (87.828125)_{10}$$

❖ ***Remarque 1.1 :***

L'intérêt de ce système est que la **base 8** est une puissance de **2** (**8 = 2^3**), donc les poids sont aussi des puissances de **2**. Chaque symbole de la **base 8** est exprimé sur **3 éléments binaires** : $(a_i)_8 = b_{i2}b_{i1}b_{i0}$

II.4. Le système hexadécimal

Le système hexadécimal a comme **base 16**, ce qui implique **16 symboles** de chiffres possibles, qui, dans ce cas, sont les dix chiffres **0 à 9** plus les lettres majuscules **A, B, C, D, E et F**. Le Tableau I. 1 expose les rapports entre les systèmes hexadécimal, décimal et binaire. Remarquez que chaque chiffre hexadécimal a comme équivalent binaire un groupe de quatre bits.

Tableau I. 1. Rapport entre hexadécimal, décimal et binaire

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

EXEMPLE I.4 :

$$(1A7,6)_{16} = 1.16^2 + 10.16^1 + 7.16^0 + 6.16^{-1} = (423.375)_{10}$$

❖ ***Remarque 1.2 :***

L'intérêt de ce système est que la base 16 est une puissance de 2 (**16=2^4**), donc les poids sont aussi des puissances de 2. Chaque symbole de la base **16** est exprimé sur **4** éléments binaires : $(a_i)_{16} = b_{i3}b_{i2}b_{i1}b_{i0}$

EXEMPLE I.5 :

$$(1A7,6)_{16} = (1\ 1010\ 0111.0110)_2$$

III. Conversion d'un système de numération à un autre

III.1. Conversion de la base 10 à une base X

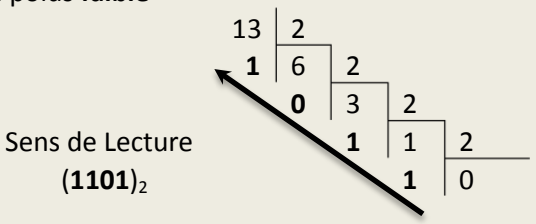
III.1.1 Conversion de la partie entière

Pour passer d'un **nombre décimal** à un nombre exprimé dans une **autre base,** on utilise la méthode des **divisions successives.**

On divise alors le nombre décimal N_{10} par la base b (binaire, octal, hexadécimal…). Le reste de la division est un digit du résultat**.**

EXEMPLE I.6 :

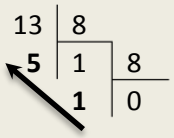
$(13)_{10} = (?)_2$ Bit de poids **faible**



Sens de Lecture
$(\mathbf{1101})_2$

Bit de poids **fort**

$(13)_{10} = (?)_8$



Sens de Lecture
$(\mathbf{15})_8$

III.1.2 Conversion de la partie fractionnaire

La conversion de la partie fractionnaire s'obtient par l'opérateur inverse, soit la multiplication. Cette méthode de conversion est illustrée ci-après pour le nombre $(\mathbf{0,375})_{10}$. Nous utilisons des **multiplications successives par 2** du nombre décimal à convertir. A chaque multiplication nous obtenons une partie entière et un reste. Il est important de noter que le nombre binaire résultant s'obtient en écrivant le premier chiffre à la position du bit de poids le plus fort.

EXEMPLE I.7 :

```
0,375 X 2 = 0,75 → partie entière 0   Poids fort (MSB)
                   Reste 0,75
0,75 X 2 = 1,5   → partie entière 1
                   Reste 0,5
0,5 X 2 = 1      → partie entière 1
                   Reste 0
```

Donc $(\mathbf{0,375})_{10} = (\mathbf{0,011})_2$

III.2. Conversion d'une base X à la base 10

Le principe général pour passer de base X vers la base 10 consiste à faire l'évaluation à partir de la définition précédente $val_X (a_{n-1} \dots a_1 a_0) = a_{n-1} b^{n-1} + \dots + a_1 b_1 + a_0 b_0$.

Cette conversion est assez simple puisque il suffit de faire le développement en **polynôme** de ce nombre dans la base **X**, et de faire la **somme** par la suite.

III.3. Conversion d'une base B_1 à une base B_2

III.3.1 Base 2^n vers base 2

Chaque symbole de la base $\mathbf{B = 2^n}$ peut être représenté par **n éléments binaires**. (Voir Exemple I.5)

III.3.2 Base 2 vers base 2^n

Dans ce cas, il suffit de regrouper les éléments binaires par **paquets de n**.

EXEMPLE I.8 :

$$(1010110)_2 = (\underbrace{001}_{1}\ \underbrace{010}_{2}\ \underbrace{110}_{6})_2 = (126)_8$$

$$= (\underbrace{0101}_{5}\ \underbrace{0110}_{6})_2 = (56)_{16}$$

III.3.3 Base *i* vers base *j*

Dans le cas où ***i*** et ***j*** sont des puissances de **2**, on utilise la **base 2** comme **relais** ;

Exemple : `base 8 → base 2 → base 16`

Sinon, on utilise la **base 10** comme relais ;

Exemple : `base 5 → base 10 → base 8`

IV. Les codes numériques

Dans une machine, toutes les informations sont codées sous forme d'une suite de "**0**" et de "**1**" (**langage binaire**). Mais l'être humain ne parle généralement pas couramment le langage binaire. Il doit donc tout "traduire" pour que la machine puisse exécuter les instructions relatives aux informations qu'on peut lui donner. Le codage étant une opération purement humaine, il faut produire des algorithmes qui permettront à la machine de traduire les informations que nous voulons lui voir traiter. C'est une opération établissant une bijection entre une information et une suite de " 0 " et de " 1 " qui sont représentables en machine. Le codage est donc une opération qui établit une correspondance entre un ensemble source (nombre, caractère, symbole) vers un ensemble destination contenant des combinaisons de 0 et de 1.

Un nombre ou caractère peut se présenter dans plusieurs codes. Les codes les plus utilisé sont :

- Le code binaire pur ;
- Le code **DCB** (**D**écimal **C**odé **B**inaire) ou BCD ;
- Le code binaire réfléchi (**code Gray**) ;
- Le code **ASCII** (**A**mericain **S**tandard **C**ode for **I**nformation **I**nterchange).

IV.1. Code binaire pur

Le code binaire pur est un code pondéré par des puissances de 2, utilisé en arithmétique binaire. Ses dérivées sont le code octal et le code hexadécimal.

IV.2. Code binaire en complément vrai (Complément à 2)

Tout comme le code binaire pur, c'est un code pondéré par des puissances de 2 dont le bit de **poids fort a un poids négatif**. C'est le code le plus utilisé en arithmétique binaire.

EXEMPLE I.9 :

$$(10101100)_{CA2} = -2^7 + 2^5 + 2^3 + 2^2 = (-84)_{10}$$

IV.3. Code DCB (décimal codé binaire)

Dans le code **DCB**, chaque chiffre décimal (**0**,**1**, . . . ,**9**) est codé en binaire avec 4 éléments binaires. C'est un code pondéré avec les poids 1, 2, 4, 8, 10, 20, 40, 80, 100, 200,… Plus facile pour coder des grands nombre, il est surtout utilisé pour l'affichage des nombres.

EXEMPLE I.10 :

$$(129)_{10} = (10000001)_2 = (\underbrace{0001}_{1}\ \underbrace{0010}_{2}\ \underbrace{1001}_{9})_{DCB}$$

IV.4. Code de Gray (ou code binaire réfléchi)

C'est un code qui permet d'éviter les erreurs de transition lors des changements d'état en binaire. Dans ce code, un seul bit change entre deux nombres consécutifs (notion d'adjacence).

EXEMPLE I.11 :

Hexadécimal	Binaire pur				Gray			
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

Le code Gray présente 4 symétries miroir. Il est cyclique et il se referme sur lui-même.

Ce code est utilisé dans les tableaux de Karnaugh que nous allons développer dans les chapitres suivants, dans des circuits d'entrée/sortie, et dans certains convertisseurs analogique/numérique.

IV.5. Code ASCII (American Standard Code for Information Interchange)

Le code utilisé par la majorité des ordinateurs pour reconnaître les caractères **(lettres, chiffres, symboles)** est le code **ASCII**, on l'appelle aussi code alpha numérique.

Le code **ASCII** à 7 bits permet de coder 2^7 = 128 caractères.

EXEMPLE I.12 :

Code ASCII							Signification
0	1	1	0	0	0	1	**Chiffre 1**
1	0	0	0	0	0	1	**Lettre latine capitale A**
1	0	0	0	1	0	0	**Lettre latine capitale D**
0	1	1	1	1	1	1	**Point d'intérogation ?**
0	1	0	0	0	0	0	**Espacement**
0	0	0	1	1	0	1	**Reteur à la ligne**