IV. Transcodage information-signal (étude des principaux codes)

IV.1. Les systèmes de numération

IV.1.1. Définition

Un système de numération se définit par deux éléments :

a. La base du système b.
b. Les symboles du système.

Exemples : Système décimal (0, 1, 2, 3, 4, 5, 6, 7, 8, 9).
Système binaire (0, 1).
Système octal (0, 1, 2, 3, 4, 5, 6, 7).
Système décimal (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F).

IV.1.2. Notation

Pour un nombre N constitué de a_i symboles dans une base b,

se represente : $N = (a_{n-1}, a_{n-2}, a_{n-3},\dots a_0)_b$

s'écrit : $N = a_{n-1}.b^{n-1} + a_{n-2}.b^{n-2} + a_{n-3}.b^{n-3}+\dots a_0.b^0$

Applications:

$(2021)_{10}$, $(11010)_2$, $(2701)_8$, $(11AF)_{16}$.

$N_1 = 10^3.2 + 10^2.0 + 10^1.2 + 10^0.1 = 2000 + 0 + 20 + 1 = 2021$ en décimal.

$N_2 = 2^4.1 + 2^3.1 + 2^2.0 + 2^1.1 + 2^0.0 = 16 + 8 + 0 + 2+0 = 26$ en décimal.

$N_3 = 8^3.2 + 8^2.7 + 8^1.0 + 8^0.1 = 1024 + 448 + 0 + 1 = 1473$ en décimal.

$N_4 = 16^3.1 + 16^2.1 + 16^1.A + 16^0.F = 4096 + 256 + 160 + 16 = 4528$ en décimal.

IV.2. Le transcodage

Le transcodage est l'opération de conversion d'une base d'un système de numération vers une autre base.

Ce que nous avons vu aux applications précédentes, est le transcodage des systèmes de numération (binaire, octal, hexadécimal) vers le système décimal.

IV.2.1.Le transcodage du décimal vers une autre base (binaire, hexadécimal)

L'opération se fait par la division successivement du nombre entier à transcodes par la base b(2), les calcules s'arrêtent lorsque le quotient arrive à 0. Puis par la multiplication successivement du nombre décimal à transcoder par b(2), les calcules s'arrêtent lorsque le quotient arrive à 1,00**[10]**.

Exemple :

Conversion de 28,8625 en binaire,

La partie entier 28

28/2=14 0 , 14/2=7 0, 7/2=3 1, 3/2=1 1, 1/2=0 1,

donc $(28)_{10} \equiv (11100)_2$.

La partie décimal 0,8625

– Conversion de 0,8625 :

- 0,8625 x 2 = 1,725 = 1 + 0,725
- 0,725 x 2 = 1,45 = 1 + 0,45
- 0,45 x 2 = 0,9 = 0 + 0,9
- 0,9 x 2 = 1,8 = 1 + 0,8
- 0,8 x 2 = 1,6 = 1 + 0,6
- 0,6 x 2 = 1,2 = 1 + 0,2
- 0,2 x 2 = 0,4 = 0 + 0,4
- 0,4 x 2 = 0,8 = 0 + 0,8 …

28,8625 peut être représenté par $(11100,11011100...)_2$

Et par la même méthode on peut transformer le nombre $(29.\ 975)_{10}$ en hexadécimal

La partie entier 29 : 29/16=1 D, 1/16=0 1,

Donc $(28)_{10} \equiv (1D)_{16}$.

La partie décimale 0,975

0,975x16 = 15.6 = F+0,6

0,6 x 16 = 9,6 = 9+0.6,

Donc : $(29,975)_{10}$ peut être représenté par $(1D,F99...)_{16}$

IV.2.2. Le transcodage d'une base 2 vers une autre base (octal, hexadécimal)

IV.2.2.1. Le transcodage du binaire vers l'octal

Dans ce cas, on devise la série des bits en groupes de trois bits, par exemple :
$(1100010011110)_2 \equiv 1\ 100\ 010\ 011\ 110 \equiv (14\ 2\ 3\ 6)$

IV.2.2.2. Le transcodage du binaire vers l'hexadécimal

Dans ce cas, on devise la série des bits en groupes de trois bits, par exemple :
$(1100010011110)_2 \equiv 1\ 1000\ 1001\ 1110 \equiv (189E)_{16}$

IV.2.2.3. Le transcodage de l'hexadécimal vers le binaire

Dans ce cas, on va dissocier chaque symbole de l'hexadécimal en quatre bits, exemple : $(5CFA)_{16} \equiv (101\ 1100\ 1111\ 1010)_2$

IV.3. Les principaux codes

IV.3.1. le codage des caractères alphanumériques

Les premiers codes ont été établis pour répondre au besoin de transmettre des messages rapidement**[10]**.

On ne retiendra ici que les codes les plus utilisés actuellement : ASCII et Unicode.

IV.3.2. Code ASCII (American Standard Code for Information Interchange) [10]

Le codage ASCII (normalisé en 1967) représente un jeu de 128 caractères au moyen de 7 signaux à 2 états.

La table de correspondance est représentée ci-dessous, sous forme rectangulaire pour être plus compacte.

Ce code comprend 26 majuscules, 26 minuscules, 10 chiffres, 32 symboles, 33 codes de contrôle et un espace. Ce codage n'est réellement adapté qu'à la langue anglaise qui est dépourvue de toute décoration (ou signes diacritiques : accent, cédille, tilde, etc..) sur les caractères.

Des codes de commande pour les anciens terminaux complètent cette liste : seuls quelques uns sont encore utilisés aujourd'hui (NUL, BS, HT, LF, VT, FF, CR, ESC, DEL).

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

Figure 13. Code ASCII[10]

Ce code (**Figure 13)** est très limité, mais il suffit à de nombreuses activités techniques dans le domaine du traitement de l'information (systèmes d'exploitation, claviers, imprimantes, internet, etc..) **[10, 11]**.

IV.3.3. L'unicode

Après des tentatives d'extension du code ASCII par les firmes informatiques, Unicode a été créé de manière concertée. Au départ, les caractères étaient codés sur 16 bits. Depuis, il a été étendu à 32 bits.

La première version a été développée en 1991. L'unicode est mis à jour régulièrement pour prendre en compte de nouveaux alphabets : 5.2 (2009), 6.0 (2012), 6.1 (2012), 6.3 (2013).

Dans la version 6.0 (janvier 2012), l'Unicode permettait de coder (voir Figure 14) :
– 137 468 caractères à usage privé.
– 109 242 lettres ou syllabes, chiffres ou nombres, symboles divers, signes diacritiques (accent...) et signes de ponctuation.
– plusieurs centaines de caractères de contrôle ou modificateurs spéciaux.

Pour assurer la compatibilité, les codes des caractères du jeu ASCII sont les mêmes en ASCII qu'en Unicode, il suffit d'ajouter des zéros à gauche pour arriver à 16 bits.

ASCII/8859-1 Text		Unicode Text	
A	0100 0001	A	0000 0000 0100 0001
S	0101 0011	S	0000 0000 0101 0011
C	0100 0011	C	0000 0000 0100 0011
I	0100 1001	I	0000 0000 0100 1001
I	0100 1001	I	0000 0000 0100 1001
/	0010 1111		0000 0000 0010 0000
8	0011 1000	天	0101 1001 0010 1001
8	0011 1000	地	0101 0111 0011 0000
5	0011 0101		0000 0000 0010 0000
9	0011 1001	س	0000 0110 0011 0011
-	0010 1101	ل	0000 0110 0100 0100
1	0011 0001	ا	0000 0110 0010 0111
	0010 0000	م	0000 0110 0100 0101
t	0111 0100		0000 0000 0010 0000
e	0110 0101	α	0000 0011 1011 0001
x	0111 1000	≰	0010 0010 0111 0000
t	0111 0100	γ	0000 0011 1011 0011

Figure 14. Développement du code ASCII en ASCII8859-1 puis en Unicode[10].

IV.3.4. Le codage RZ

Le codage RZ (voir figure 15) est le plus simple et le plus facile codage à utiliser et réaliser, en effet, pour établir ce codage, il suffit de réaliser une porte logique ET(AND) qui relie le signal de l'horloge et l'information à transmettre**[11, 12]**,

$$\begin{cases} e(i) = 0 \Rightarrow s(i) = 0 \\ e(i) = 1 \Rightarrow \begin{cases} s(i) = 1 \; pour \; 0 \leq t \leq T/2 \\ s(i) = 0 \; pour \; T/2 \leq t \leq T \end{cases} \end{cases}$$

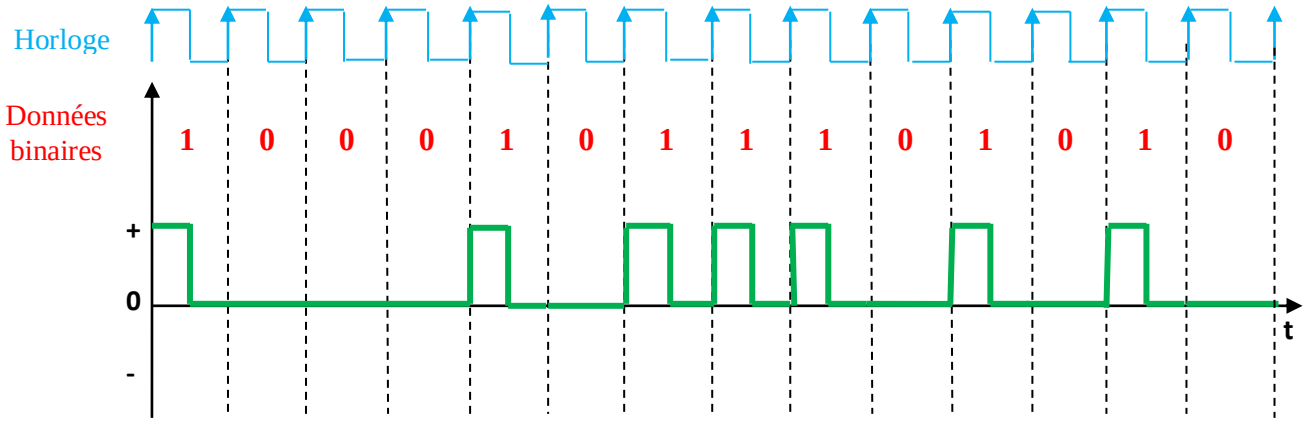

Figure 15. Principe du code RZ.

IV.3.5. Le codage NRZ[11]

Le codage NRZ est plus simple et plus efficace que le codage RZ, il consiste à transformer les niveaux 1 en +V et les niveaux 0 en –V (V étant une valeur de tension continue). Lorsque le front montant de l'horloge capte la présence d'un 1 en entrée, le signal de sortie prend la valeur +V jusqu'au front montant suivant Si l'information suivante est un 0 alors le signal prendra la valeur –V et ainsi de suite (voir figure 16).

$$\begin{cases} e(i) = 0 \Rightarrow S(i) = -V \\ e(i) = 1 \Rightarrow S(i) = +V \end{cases}$$

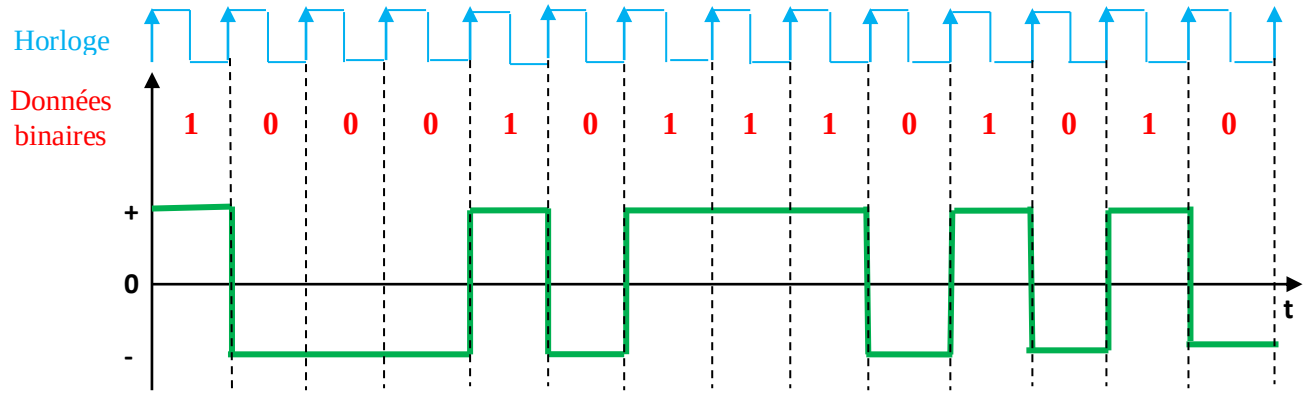

Figure 16. Principe du codage NRZ.

Ce type de codage fait osciller le signal d'informations à transmettre entre deux états logiques non nuls, le récepteur pourra déterminer ainsi la présence ou non du signal.

IV.3.6. Le codage Manchester[11, 12]

Le code Manchester est obtenu a l'aide de la réalisation de la porte logique OU exclusif (XOR) entre le signal de l'horloge et l'information binaire à transmettre, le signal résultant de la porte logique aura la forme : un front montant en T/2 pour les 1 logiques et un front descendant en T/2 pour les 0 logiques (voir figure 17).

$$\begin{cases} e(i) = 0 \Rightarrow \begin{cases} S(i) = +V \\ S\left(i + \frac{T}{2}\right) = -V \end{cases} & \text{ou} \downarrow \\ e(i) = 1 \Rightarrow \begin{cases} S(i) = -V \\ S\left(i + \frac{T}{2}\right) = +V \end{cases} & \text{ou} \uparrow \end{cases}$$

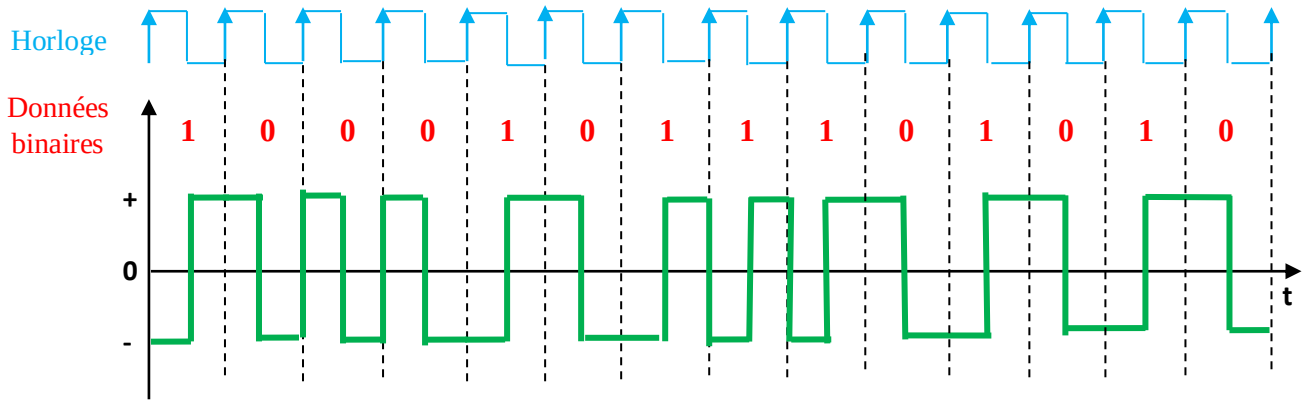

Figure 17. Principe du code Manchester.