1. La logique

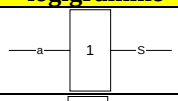
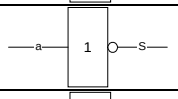
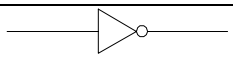
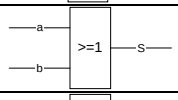
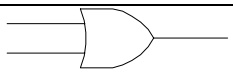
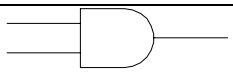
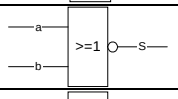
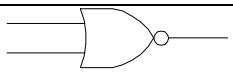
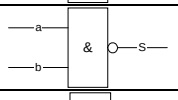
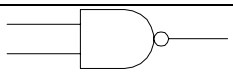
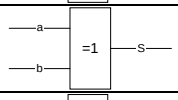
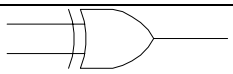
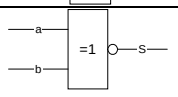
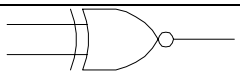
1.1. Rappels

Conventions en logique positive
le niveau haut (état 1) correspond à une affirmation logique (vrai)
le niveau bas (état 0) correspond à une infirmation logique (faux)

Conventions en logique négative
le niveau haut (ou état 1) correspond à une infirmation logique (faux)
le niveau bas (ou état 0) correspond à une affirmation logique (vrai)

dans un changement de convention (passage d'une logique positive à une logique négative)
le non reste non, le et devient ou et le ou devient et !

a	b	$a+b$ ☺ou	$a \cdot b$ ☺et	$\overline{a}$ ☺barre
0	0	0	0	1
0	1	1	0	1
1	0	1	0	0
1	1	1	1	0

fonction	équation	logigramme	schéma électronique
oui (assignation)	$S = a$		
non (inversion)	$S = \overline{a}$		
ou (OR)	$S = a + b$		
et (AND)	$S = a \cdot b$		
Non ou (NOR)	$S = \overline{a + b}$		
Non et (NAND)	$S = \overline{a.b}$		
ou exclusif (XOR)	$S = a \oplus b = \overline{a} \cdot b + a \cdot \overline{b}$		
Identité (NXOR)	$S = \overline{a \oplus b} = \overline{a} \cdot \overline{b} + a \cdot b$ $S = a \otimes b$		

1.2. Propriétés

commutativité	$a+b=b+a$ $a\cdot b=b\cdot a$
élément neutre	$a+0=a$ $a\cdot 1=a$
distributivité	$a+(b\cdot c)=(a+b)\cdot(a+c)$ $a.(b+c)=(a\cdot b)+(a\cdot c)$
compléments	$a+\bar{a}=1$ $a\cdot\bar{a}=0$ $a+1=1$ $a\cdot 0=0$
idempotence	$a+a=a$ $a\cdot a=a$
unicité du complément	$1=\bar{0}$ $0=\bar{1}$ $a=\bar{\bar{a}}$
absorption	$a\cdot(a+b)=a$ $a+(a\cdot b)=a$
associativité	$(a+b)+c=a+(b+c)$ $(a\cdot b)\cdot c=a\cdot(b\cdot c)$
inclusion de 1ere espèce	$a+a\cdot b+a\cdot b\cdot c=a$
inclusion de 2ième espèce	$a\cdot b+\bar{a}\cdot c+b\cdot c=(a+c)\cdot(\bar{a}+b)$
inclusion de 3ième espèce	$a+\bar{a}\cdot b=a+b$

théorème de De Morgan
$\overline{a\cdot b}=\bar{a}+\bar{b}$
$\overline{a+b}=\bar{a}\cdot\bar{b}$

1.3. Simplification par la méthode graphique de Karnaugh

le tableau de karnaugh n'est autre que la table de vérité de l'équation logique *S* dont on désir connaître la forme minimale. Mais :

☞on représente la table sous la forme d'un tableau (si l'équation logique comporte n variables indépendantes, nous aurons alors un tableau à 2^n cases au lieu d'avoir une table à 2^n lignes !).

☞on représente les différents états des variables logiques composant l'équation en utilisant le code GRAY.

On remarque alors que le passage d'une case à l'autre dans le sens vertical ou horizontal, entraîne la variation d'une seule variable à la fois.

code gray
0000
0001
0011
0010
0110
0111
0101
0100
1100
1111
1111
1110
1010
1001

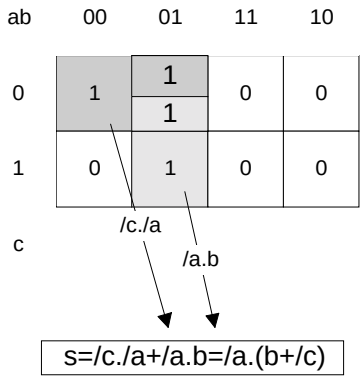
On réalise des groupements de 2^i cases adjacentes qui sont à 1 (si on groupe les 0 alors on obtient la forme minimale de $\overline{S}$.)

On cherche toujours les groupements maximaux. On peut prendre plusieurs fois une même case (idempotence).

La forme « minimale » de l'équation est alors obtenue en écrivant la somme logique des produits des variables qui ne changent pas d'état pour chaque groupement.

abc	s
000	1
001	0
010	1
011	1
100	0
101	0
110	0
111	0

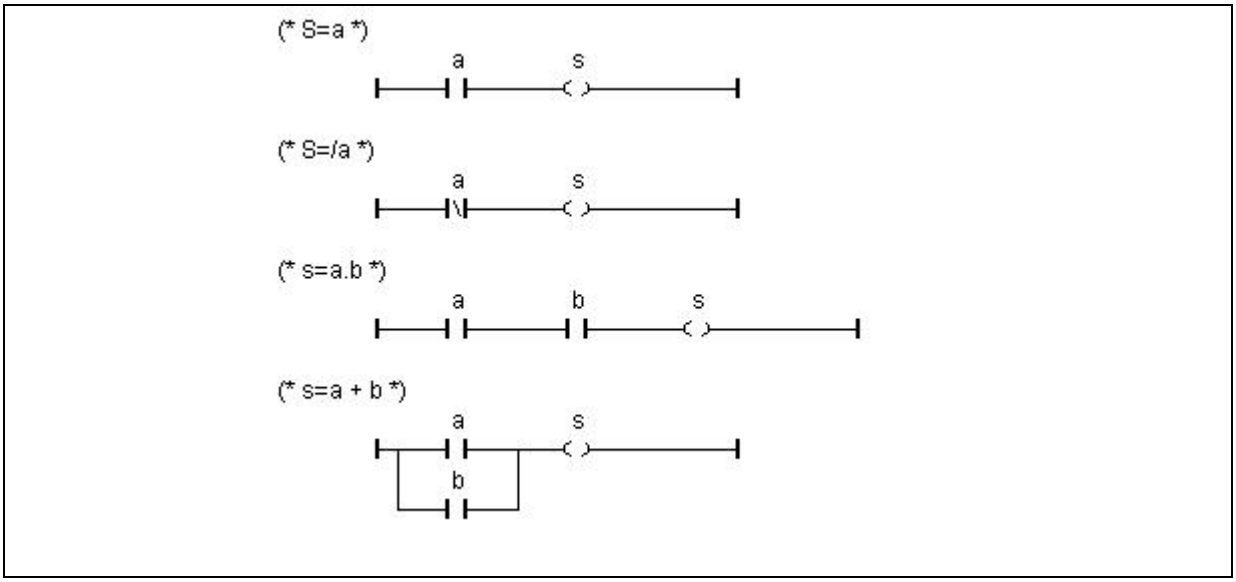
☞



1.4. Programmation en langage automate (norme IEC1131)

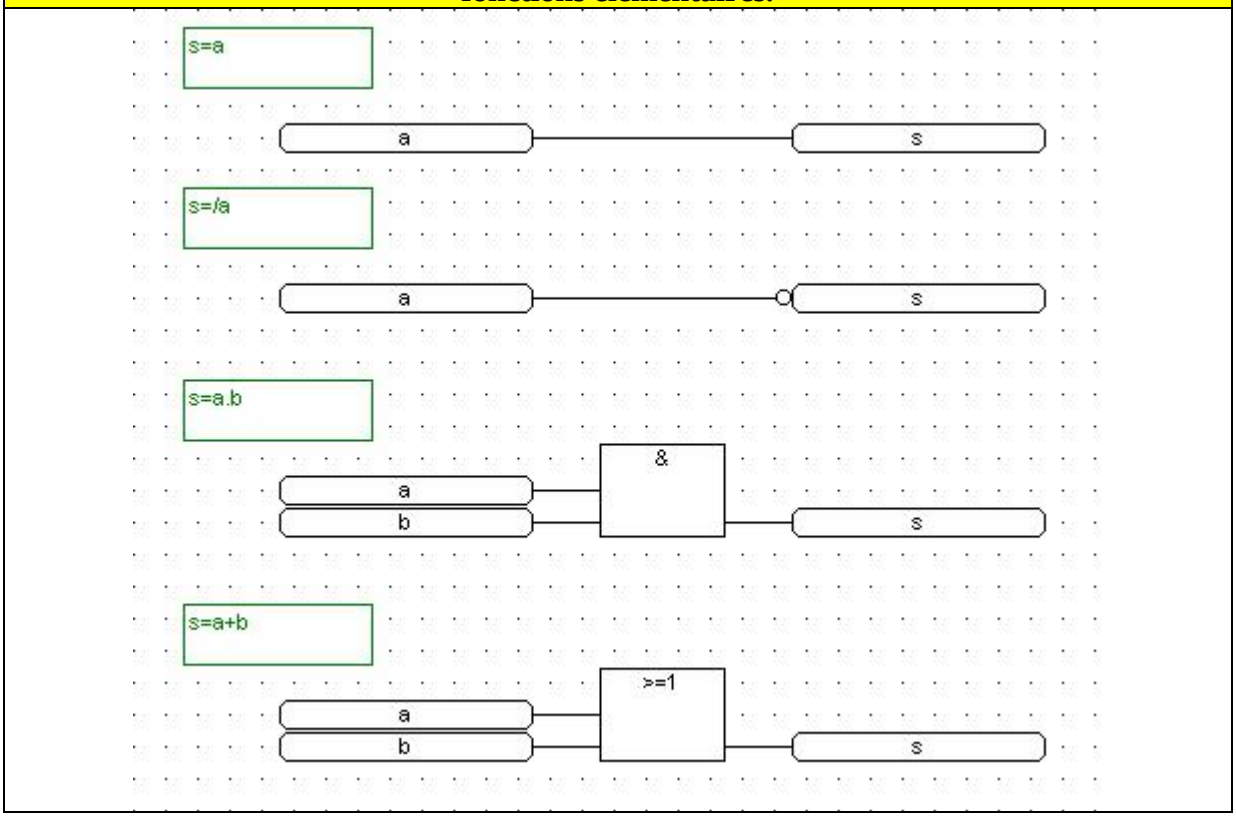
L'assignation, l'inversion booléenne, le ET logique, le OU logique et le OU exclusif logique sont disponibles sous des formes différentes dans les 4 langages LD, IL, FBD et ST.

Le langage LD (Ladder Diagram),
C'est une représentation graphique d'équations booléennes
Combinant des contacts (en entrée) et des relais (en sortie).



Le symbole \ dans le contact ou dans la bobine traduit le NON.

Langage FBD (Function Block Diagram), langage graphique. Il permet la construction d'équations complexes à partir des fonctions élémentaires.



*Le symbole —O— traduit le **NON.***

Langage IL (Instruction List), C'est un langage textuel de bas niveau, particulièrement adapté aux applications de petite taille.
(*s=a*) LD a ST s (*s=/a*) LDN A ST s (*s=a.b*) LD a AND b ST s (*s=a+b*) LD a OR b ST s (*s=/(a+b)*) LD a OR b STN s

LD charge un opérande (variable ou constante).
ST stocke le résultat courant dans la variable.
N traduit le ***NON.***

langage ST (Structured Text), c'est un langage textuel de haut niveau.
s:=a; (*fonction OUI*) s:=not(a); (*fonction NON*) s:=a or b; (*fonction OU*) s:=a and b; (*fonction ET*) s := not(a xor b) ; (*fonction identité*)

Un programme ST est une suite d'énoncés.
Un énoncé se termine par un point virgule (";").
Notez bien la façon d'écrire l'assignation (variable := expression;)

2. Numération et bases

Le système de numération le plus répandu sur la terre est le système décimal. Pourquoi ? peut-être tout simplement parce que l'homme a dix doigts! Les ordinateurs n'ont pas de doigts, mais des portes logiques qui peuvent avoir deux états : ouvert et fermé. Ils utilisent le système binaire.

2.1. Expression d'un nombre x dans une base B

La base B possède B chiffres $\{C_0,...,C_{B-1}\}$.

$$x=\sum_{j=-\infty}^{j=+\infty} n_j.B^j \text{ avec } n_j \in \{C_0,...,C_{B-1}\}$$

2.2. Système de numération binaire

Les chiffres (BIT pour Binary Digit) du système de numération binaire sont $\{0,1\}$.
Le Most Significant Bit (MSB) est le bit de poids fort.
Le Lost Significant Bit (LSB) est le bit de poids faible
Si on utilise N bits, on parle de mot de N bits.

2.2.1. Exemple d'écriture

$$1992_{(10)}=1.2^{10}+1.2^9+1.2^8+1.2^7+1.2^6+0.2^5+0.2^4+1.2^3+0.2^2+0.2^1+0.2^0$$

MSB LSB

$$1992_{(10)}=11111001000_{(2)}$$

$$3{,}25_{(10)}=1.2^1+1.2^0+0.2^{-1}+1.2^{-2}$$
$$3{,}25_{(10)}=11{,}01_{(2)}$$

2.2.2. remarques

- En utilisant un mot de N bits, on peut dénombrer 2^N termes. La valeur du dernier terme dans le système décimal est 2^N-1.
- Le nombre de bits nécessaire pour représenter un nombre décimal D est $N=E(\frac{\log(D+1)}{\log(2)})+1$
- L'octet désigne un mot de 8 bits.
- $1M_o=1024K_o \quad 1K_o=1024 octets$

2.3. Le système de numération hexadécimal

La facilité avec laquelle se font les conversions entre les systèmes binaire et hexadécimal, explique pourquoi le système hexadécimal est devenu une façon abrégée d'exprimer de grands nombres binaires.

Les chiffres hexadécimaux sont $\{0,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F\}$ avec **A=10, B=11, C=12, D=13, E=14 et F=15.**

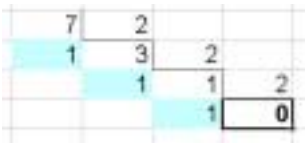
2.3.1. Exemple d'écriture

$$1992_{(10)} = 7.16^2 + C.16^1 + 8.16^0 = 7C8_{(16)}$$
$$2000{,}0390625_{(10)} = 7.16^2 + D.16^1 + 0.16^0 + 0.16^{-1} + A.16^{-2} = 7D0{,}0A_{(16)}$$

2.3.2. remarques

- En utilisant un mot de N chiffres hexadécimaux, on peut dénombrer 16^N termes. La valeur du dernier terme dans le système décimal est $16^N - 1$.
- Le nombre de chiffres hexadécimaux nécessaire pour représenter un nombre décimal D est $N = E(\frac{\log(D+1)}{\log(16)}) + 1$
- Chaque chiffre hexadécimal à comme équivalent binaire un groupe de 4 bits.

2.4. Les changements de base

De → vers ↓	Décimal	Binaire	Hexadécimal
Décimal		Cf. §2	Cf. §3
Binaire	 $7_{(10)} = 111_{(2)}$		$CAFE_{(16)} = 1100101011111110_{(2)}$ *chaque chiffre hexadécimal est transformé en son équivalent binaire à 4 chiffres*
Hexadécimal	 $2001_{(10)} = 7D1_{(16)}$	$1110101_{(2)} = 75_{(16)}$ *le nombre binaire est converti en groupe de 4 bits (de droite vers la gauche), puis les groupes sont converties en hexadécimal*	

2.5. Représentation des nombres entiers signés

2.5.1. La notation en complément à 2

- On note $\overline{A}$, le complément à 1 du nombre binaire A obtenue à partir de A en remplaçant les 1 de A par des 0 et les 0 de A par des 1.

exemple

$A = 1FAD_{16} = 1111110101101_2 = 8109_{10}$

$\overline{A} = 0000001010010_2 = 52_{16} = 82_{10}$

- Le complément à 2 d'un nombre binaire A et $\overline{A}+1$.

exemple

$A = 1FAD_{16}$

$\overline{A}+1 = 1010011_2 = 53_{16} = 83_{10}$

2.5.2. Ecriture des nombres (registre de n bits)

2.5.2.1.Nombre entier non signé

$n \in [0, 2^n - 1]$

2^{n-1}	2^{n-2}	…	2^i	…	2^1	2^0
MSB			***i^ème bit***			***LSB***

exemple

	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^9	2^8	2^7	2^6	2^5	2^4	2^3	2^2	2^1	2^0
$1FAD_{16}$	0	0	0	1	1	1	1	1	1	0	1	0	1	1	0	1

*2.5.2.2.***Nombre entier signé. Le bit de poids fort (MSB) devient le bit de *signe*.**

$n \in [-2^{n-1}, 2^{n-1} - 1]$.

2^{n-1}	2^{n-2}		2^i		2^1	2^0
Bit de signe	***MSB***		***i^ème bit***			***LSB***

Par convention, le bit de signe vaut 0 quand le nombre binaire est positif et 1 quand le nombre binaire est négatif.

exemple

	2^{15}	2^{14}	2^{13}	2^{12}	2^{11}	2^{10}	2^{9}	2^{8}	2^{7}	2^{6}	2^{5}	2^{4}	2^{3}	2^{2}	2^{1}	2^{0}
$1FAD_{16}$	**0**	0	0	1	1	1	1	1	1	0	1	0	1	1	0	1
$-1FAD_{16}$	**1**	1	1	0	0	0	0	0	0	1	0	1	0	0	1	1

Pour écrire les nombres binaires négatifs avec la notation en complément à 2, on attribue au bit de signe la valeur 1 et on transforme la grandeur binaire en son complément à 2.

Il est à remarquer que si on affecte –1 et non 1 à la valeur du bit de signe, nous retrouvons alors la valeur exacte du nombre

$$-1FAD_{16} = 1110000001010011_2 = -2^{15} + 2^{14} + 2^{13} + 2^6 + 2^4 + 2^1 + 2^0 = -8109$$

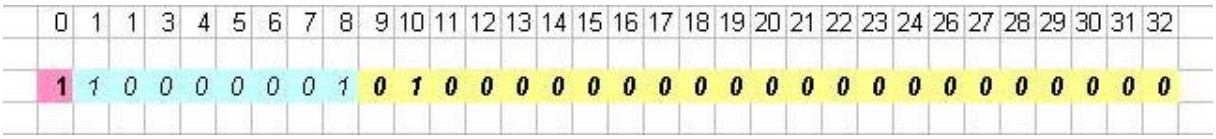
2.6. Norme IEEE754 (nombre réels sur 32 bits)

Signe s (1 bit b_0)	Exposant e (8bits $b_1 \ldots b_8$)	Mantisse m (23 bits $b_9 \ldots b_{31}$)
C'est le MSB 1 si négatif, 0 si positif	$e \in [1,254]$ si e=0 et m=0 alors x=0 si e=255 et m=0 alors x=∝	$m = \sum_{i=9}^{31} 2^{8-i} b_i$

- En simple précision, un mot de 32 bits représente le nombre réel

$$(-1)^s .2^{(e-127)}.(1,m) = b_0 b_1 \ldots b_{31} = (-1)^{b_0} .2^{b_1 b_2 \ldots b_8}.2^{-127}.(1, b_9 b_{10} b_{11} \ldots b_{31})_2$$

Exemple



s=1
e=129
m=0,25

Soit x = -5

3. Les codes

3.1. Le code DCB

Si on représente chaque chiffre d'un nombre décimal par son équivalent binaire, on obtient le code dit Décimal Codé Binaire (DCB).

Les groupes 1010, 1011, 1100, 1101, 1110, 1111 sont inadmissibles.

Le code DCB n'est pas un système de numération. Un nombre codé en DCB n'est pas un nombre binaire pur. Le principal avantage du code DCB provient de la facilité relative avec laquelle on passe d'un nombre codé DCB à un nombre décimal. Mais ce code est gourmant en bits et les calculs arithmétiques sont complexes ce qui fait qu'il n'est pas utilisé sur les ordinateurs.

exemple

2001 codé en DCB devient 0010 0000 0000 0001. Mais stocké dans le registre 16 bits d'une Unité de traitement (UT) on obtient la valeur $2001_{16} = 8193_{10}$!

3.2. Le code GRAY

Le code GRAY appartient à la catégorie des codes dit à distance minimale (une représentation codée ne diffère de celle qui la précède que par un bit). Le code GRAY est le principe de codage des tableaux de Karnaugh.

construction

	Syst. Numération binaire				CODE GRAY			
	2^3	2^2	2^1	2^0	G_3	G_2	G_1	G_0
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

2^i zéro
2^{i+1} un
2^{i+1} zéro
…

$G_n = B_n$
$G_{n-1} = B_{n-1} \oplus B_n$
…
$G_i = B_i \oplus B_{i+1}$
…
$G_0 = B_0 \oplus B_1$

3.3. Le code ASCII

contrôle des liaisons séries

0 NUL
1 SOH
2 STX
3 ETX
4 EOT
5 ENQ
6 ACK
7 BEL
8 BS
9 HT
10 LF
11 VT
12 NP
13 CR
14 SO
15 SI
16 DLE
17 DC1
18 DC2
19 DC3
20 DC4
21 NAK
22 SYN
23 ETB
24 CAN
25 EM
26 SUB
27 ESC
28 FS
29 GS
30 RS
31 US

de 128 à 160, caractères semi-graphiques...

0	32		64	@	96	‘	128	160		192	À	224	à
1	33	!	65	A	97	a	129	161	¡	193	Á	225	á
2	34	"	66	B	98	b	130	162	¢	194	Â	226	â
3	35	#	67	C	99	c	131	163	£	195	Ã	227	ã
4	36	$	68	D	100	d	132	164	¤	196	Ä	228	ä
5	37	%	69	E	101	e	133	165	¥	197	Å	229	å
6	38	&	70	F	102	f	134	166	¦	198	Æ	230	æ
7	39	’	71	G	103	g	135	167	§	199	Ç	231	ç
8	40	(	72	H	104	h	136	168	¨	200	È	232	è
9	41	)	73	I	105	i	137	169	©	201	É	233	é
10	42	*	74	J	106	j	138	170	ª	202	Ê	234	ê
11	43	+	75	K	107	k	139	171	«	203	Ë	235	ë
12	44	,	76	L	108	l	140	172	¬	204	Ì	236	ì
13	45	-	77	M	109	m	141	173	-	205	Í	237	í
14	46	.	78	N	110	n	142	174	®	206	Î	238	î
15	47	/	79	O	111	o	143	175	¯	207	Ï	239	ï
16	48	0	80	P	112	p	144	176	°	208	Đ	240	ð
17	49	1	81	Q	113	q	145	177	±	209	Ñ	241	ñ
18	50	2	82	R	114	r	146	178	²	210	Ò	242	ò
19	51	3	83	S	115	s	147	179	³	211	Ó	243	ó
20	52	4	84	T	116	t	148	180	´	212	Ô	244	ô
21	53	5	85	U	117	u	149	181	µ	213	Õ	245	õ
22	54	6	86	V	118	v	150	182	¶	214	Ö	246	ö
23	55	7	87	W	119	w	151	183	·	215	×	247	÷
24	56	8	88	X	120	x	152	184	¸	216	Ø	248	ø
25	57	9	89	Y	121	y	153	185	¹	217	Ù	249	ù
26	58	:	90	Z	122	z	154	186	º	218	Ú	250	ú
27	59	;	91	[	123	{	155	187	»	219	Û	251	û
28	60	<	92	\	124	|	156	188	¼	220	Ü	252	ü
29	61	=	93	]	125	}	157	189	½	221	Ý	253	ý
30	62	>	94	^	126	~	158	190	¾	222	Þ	254	þ
31	63	?	95	_	127		159	191	¿	223	ß	255	ÿ

Un acronyme pour American Standard Code for Information Interchange. Ce code à 7-bits représente les lettres de bases de l'alphabet Romain, les nombres ainsi que d'autres caractères utilisés en informatique. Les ordinateurs ne peuvent pas comprendre les langages humains tels que l'anglais. Ils communiquent dans un langage qui leur est propre appelé binaire, qui est composé de "0" et de "1".
Les humains peuvent communiquer avec les ordinateurs en utilisant un ensemble de caractères appelés ASCII. Chaque caractère de l'ensemble ASCII est composé de 7 bits d'information, vues par l'ordinateur comme une combinaison de "0" et de "1". Ceci nous permet d'écrire des caractères et des nombres, qui ont l'apparence de l'anglais à nos yeux, mais qui peuvent être lus, enregistrés et manipulés par l'ordinateur. Les fichiers ASCII sont également appelés fichiers texte.

Le langage ASCII a été étendu à un code à 8 bits dans le but de supporter des caractères additionnels utilisés par des langues autres que l'anglais, par exemple, le français et l'allemand. Les navigateurs les plus populaires sur le Web supportent maintenant les codes ASCII à 8 bits, vous permettant de voir les caractères é, à, ô et ç entre autres.

3.3.1. Ascii et internet

3.3.1.1.dictionnaire des smiley

:-)	Sourire	:)	Mini smiley	+O:-)	le Pape
:-D	Rire	:}	Smiley moustachu	5:-)	Elvis Presley
:-*	Bisou !	:>	Sarcastique	:/7)	Cyrano de Bergerac
;-)	Complice	:(	Mécontent	>:*)	Bozo le Clown
:-X	Motus & bouche cousue	:@	J'ai encore mes	8:-)	Mickey
:-P	Bavard [Langue qui dépasse]	amygdales !		:-*f*	Muet
:-|	Indifférent	:{	Bof !	:-§	Mince !!
:-(	Pas content	:D	Rire	:-…	Monstre sévère
:'(	Une larme	:[	Petit vampire	:-~)	j'ai un rhuBe !
:-o	Oh!	:I	Hmmm...	:-†	j'ai abusé du whisky
0:-)	je suis innocent !	:<	Déçu		
>:->	Taquin	:O	Bâillement		
:->	Sourire sarcastique	:,(	Une larme		
>;->	Taquin & complice	|I	Endormi		
:-\	Heu...	|^o	Ronfler		
		:*	Bises		
:-/	C'est pas de ma faute !				

3.3.1.2.exemple de l'Art ASCII

```
                    o   o
                   \|||/
                   (o o)
-------------------------oOO--(_)--OOo-----------------------
```

3.4. Détection de code erroné

- Une transformation inopportune de la valeur binaire 1 en 0, ou de 0 en 1, dans un dispositif électronique est une erreur souvent fréquente. Il est nécessaire de **détecter** d'éventuelles erreurs. La seule possibilité est d'utiliser un code redondant, c'est à dire un ensemble de mots dont seule une partie est déclarée valide. Il existe plusieurs codage permettant de détecter une erreur éventuelle.

- **Bit de parité (paire ou impaire)**

  On suppose que l'on utilise des mots de n bits pour coder de façon significative 2^n informations. On utilise alors un $n+1^{ième}$ bit, appelé bit de parité afin d'obtenir un mot ayant un nombre pair (ou impair) de 1. Pour savoir si un code est valide, il suffit alors de compter le nombre de bits à 1 et vérifier si ce nombre est pair (ou impair). Le rôle du bit de parité est de détecter les erreurs « monobit ».

  exemple 10010010 →1 10010010 si parité paire

- **Code 2 parmi 5**
  Dans ce code, chaque mot de 5 bits contient seulement 2 bits à 1. C'est un code non pondéré (un code pondéré étant un code où chaque chiffre binaire est doté d'un poids).

caractère	B_4 bit de parité	B_3	B_2	B_1	B_0

0	0	1	1	0	0
1	1	0	0	0	1
2	1	0	0	1	0
3	0	0	0	1	1
4	1	0	1	0	0
5	0	0	1	0	1
6	0	0	1	1	0
7	0	1	0	0	1
8	1	1	0	0	0
9	0	0	0	1	1
DL			0	1	1
FL			1	0	1

3.5. Etude du code autocorrecteur de Hamming

3.5.1. Définitions

3.5.1.1.distance

c'est le nombre de positions où les symboles sont différents entre deux mots

exemple D(101111,111110) = 2

3.5.1.2.distance minimum d_{min}

c'est la plus petite distance d que l'on peut trouver entre tout les mots différents d'un code pris deux à deux..

3.5.1.3.décodage par maximum de vraisemblance

Dans le canal binaire symétrique, la probabilité pour qu'un bit reçu soit erroné est p. La probabilité de recevoir un mot de longueur n sans erreur est $(1-p)^n$. La probabilité de recevoir un mot avec une erreur dans une position donné est $p(1-p)^{n-1}$.

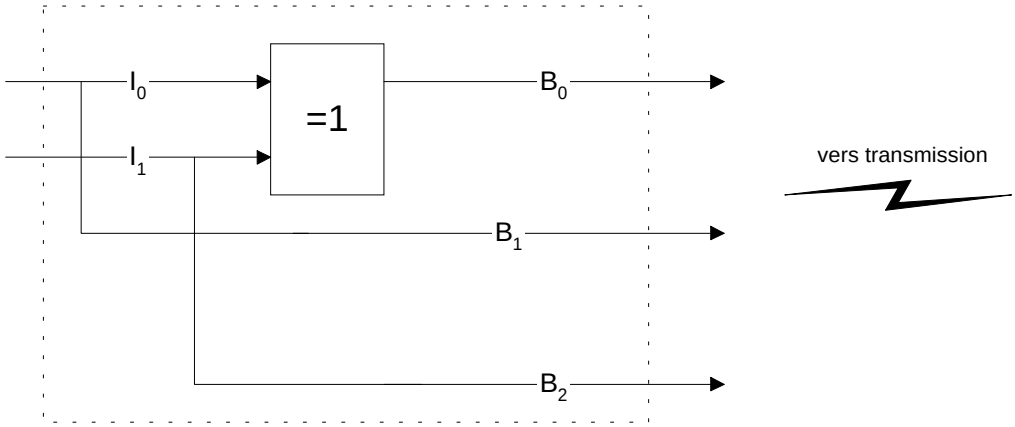
Puisque p<1/2, l'occurence d'une seule erreur est beaucoup plus probable que celle de deux erreurs dans le même mot et ainsi de suite...

Il paraît donc raisonnable d'accepter que le mot erroné est une déformation du mot de code le plus proche. C'est le principe de décodage par maximum de vraisemblance.

pour détecter t erreurs, il faut utiliser un code dont $d_{\min} \geq t+1$

Pour corriger t erreurs, il faut utiliser un code dont $d_{\min} \geq 2t+1$

exemple de codeur

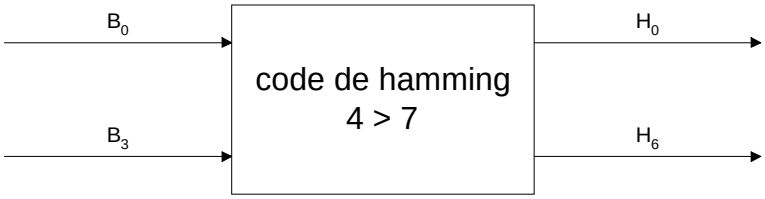


I_1I_0	$B_2B_1B_0$
00	*000*
01	*011*
10	*101*
11	*110*

distance	*000*	*011*	*101*	*110*
000		*2*	*2*	*2*
011	*2*		*2*	*2*
101	*2*	*2*		*2*
110	*2*	*2*	*2*	

Ici, la distance du codeB est $d_{\min} = 2 \Rightarrow t = 1$. *On pourrait détecter la présence d'une erreur lors de la réception (exemple la réception de 001).*

3.5.2. Etude du code de Hamming « code autocorrecteur à 7 bits ».



$$H_0 = B_0$$
$$H_1 = B_1$$
$$H_2 = B_2$$
$$H_3 = B_2 \oplus B_1 \oplus B_0$$
$$H_4 = B_3$$
$$H_5 = B_3 \oplus B_1 \oplus B_0$$
$$H_6 = B_3 \oplus B_2 \oplus B_0$$

Etablir la table de transcodage

H_6	H_5	H_4	H_3	H_2	H_1	H_0	b_3	b_2	b_1	b_0
							0	0	0	0
							0	0	0	1
							0	0	1	1

							0	0	1	0
							0	1	1	0
							0	1	1	1
							0	1	0	1
							0	1	0	0
							1	1	0	0
							1	1	0	1
							1	1	1	1
							1	1	1	0
							1	0	1	0
							1	0	1	1
							1	0	0	1
							1	0	0	0

Calculer la distance d_{min} du code.

En **déduire** que ce code est autocorrecteur sur erreur mono-bit.

Soit le mot de contrôle MC = $C_2C_1C_0$ défini sur 3 bits.

$$C_2 = H_3 \oplus H_2 \oplus H_1 \oplus H_0$$
$$C_1 = H_5 \oplus H_4 \oplus H_1 \oplus H_0$$
$$C_0 = H_6 \oplus H_4 \oplus H_2 \oplus H_0$$

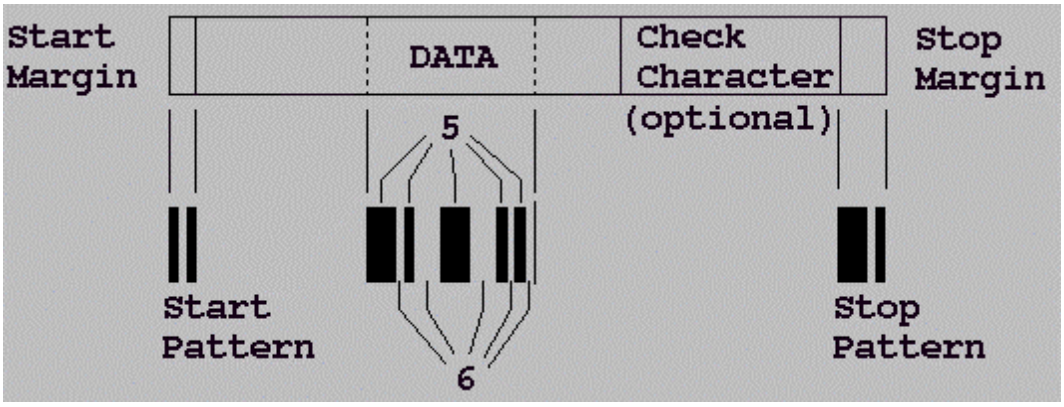
Déterminer la valeur de MC en l'absence d'erreur.

Vérifier que le mot de contrôle indique le rang R du bit erroné en cas d'erreur mono-bit.

3.6.

3.6.1. CODE 2 PARMI 5 ENTRELACE

Code numérique très dense, mais dont la moins bonne fiabilité intrinsèque oblige à l'utiliser soit en longueur fixe, soit avec une clé de contrôle (voir annexe). Le code 2 parmi 5 entrelacé utilise la même codification des caractères que le code 2 parmi 5, mais en entrelaçant les caractères deux par deux. Le premier caractère est codé avec les barres, tandis que le deuxième utilise les espaces de la même zone, et ainsi de suite. Les chiffres de rang impair sont donc codés avec les barres, tandis que les chiffres de rang pair sont codés avec les espaces. La conséquence est que le code 2 parmi 5 entrelacé encode toujours un nombre pair de caractères. Ce code utilise pour chaque caractère cinq éléments, dont 2 sont larges, d'où son nom.



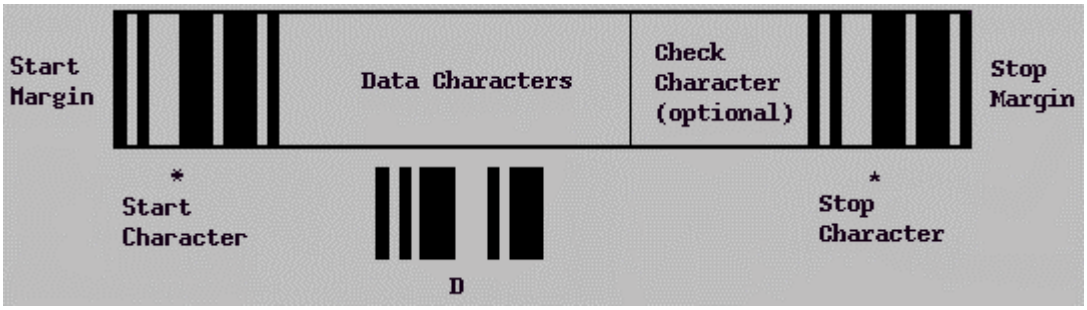
ASCII Character	Binary Word	Check Character value
1	10001	1
2	01001	2
3	11000	3
4	00101	4
5	10100	5
6	01100	6
7	00011	7
8	10010	8
9	01010	9
0	00110	0
Start	0000	
Stop	100	

3.6.1.1.Check Character Example

Message:	2632534
Characters:	2 6 3 2 5 3 4
Position:	E O E O E O E
Sum of odd :	6 + 2 + 3 = 11
Sum of even :	2 + 3 + 5 + 4 = 14
Sum of even x 3:	14 x 3 = 42
Sum of even and odd:	11 + 42 = 53
Determine the smallest number which, when added to the sum in step four, will result in a multiple of ten. This number is the value of the check character.	7
Check character:	7
	Final message: 2632534 7

3.6.2. CODE 39

Code alphanumérique autocontrôlé, présentant de très bonnes qualités de fiabilité, de facilité de lecture et d'impression qui en font le code le plus utilisé en applications industrielles. Le code 39 est un code alphanumérique permettant de coder dans sa version d'origine 43 caractères, c'est-à-dire les chiffres de 0 à 9, les lettres de A à Z, 6 symboles y compris l'espace, plus un caractère particulier de début et fin de message. La dénomination "code 39" provient de sa structure qui est de "3 parmi 9". En effet, chaque caractère du jeu de base est représenté par 9 éléments (5 barres et 4 espaces) parmi lesquels 3 sont larges (1 binaire) et 6 sont étroits (0 binaire). Les espaces entre caractères ne sont pas significatifs.



ASCII Character	Binary Word	Check Bars	Character spaces	Value
0	000110100	00110	0100	0
1	100100001	10001	0100	1
2	001100001	01001	0100	2
3	101100000	11000	0100	3
4	000110001	00101	0100	4
5	100110000	10100	0100	5
6	001110000	01100	0100	6
7	000100101	00011	0100	7
8	100100100	10010	0100	8
9	001100100	01010	0100	9
A	100001001	10001	0010	10
B	001001001	01001	0010	11
C	101001000	11000	0010	12
D	000011001	00101	0010	13
E	100011000	10100	0010	14
F	001011000	01100	0010	15
G	000001101	00011	0010	16
H	100001100	10010	0010	17
I	001001100	01010	0010	18
J	000011100	00110	0010	19
K	100000011	10001	0001	20
L	001000011	01001	0001	21
M	101000010	11000	0001	22
N	000010011	00101	0001	23
O	100010010	10100	0001	24
P	001010010	01100	0001	25
Q	000000111	00011	0001	26
R	100000110	10010	0001	27
S	001000110	01010	0001	28
T	000010110	00110	0001	29
U	110000001	10001	1000	30
V	011000001	01001	1000	31
W	111000000	11000	1000	32
X	010010001	00101	1000	33
Y	110010000	10100	1000	34
Z	011010000	01100	1000	35
-	010000101	00011	1000	36
.	110000100	10010	1000	37
SPACE	011000100	01010	1000	38
*	010010100	00110	1000	-
$	010101000	00000	1110	39
/	010100010	00000	1101	40
+	010001010	00000	1011	41
%	000101010	00000	0111	42

3.6.2.1.Check Character Example

Message	CODE 39
Characters	C O D E 3 9
Value	12 24 13 14 38 3 9
Sum of values	113
Modulo 43	113 / 43 = 2 remainder 27
check character	R
	Final message: CODE 39R