BTS ATI

NOM

PRENOM

COURS / SYNTHESE
TD / TP
TEST / EVALUATION
NOTE D'INFORMATION

Automatique et Informatique Industrielle

NUMERATION

Sommaire :

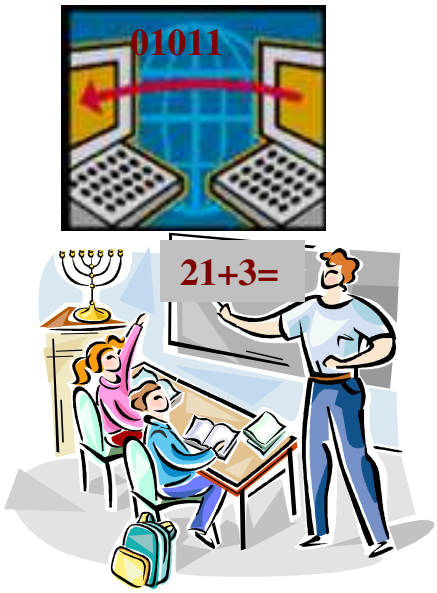
Automatique et Informatique Industrielle

I- Introduction

Nous avons vu précédemment que la fonction *traiter* traite la plupart du temps des informations de type binaires (voire numériques). Le traitement des signaux peut se faire en réalisant un certain nombre de fonctions (ET, OU, NON, calculs sur nombres, …).

Pour comprendre comment la fonction *traiter* réalise des calculs, il faut être capable de comprendre la correspondance entre la base 10 (base dans laquelle nous réalisons les calculs) et la base 2 (base dans laquelle calcule la fonction traiter).

Nous en profiterons pour parler de la base 16 qui est une base 2 améliorée.



II- Différentes bases :

Pour effectuer des calculs en nombre entiers (numérique), il est possible d'utiliser différentes bases arithmétiques qui sont « toutes » équivalentes du point de vue du résultat du calcul.

Base 10 ou ***décimal. 10*** symboles différents existent dans cette base.

C'est le système que l'on utilise pour compter (est issue des 10 doigts des mains).

0 ; 1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9

Si on veut aller plus loin dans le comptage, on combine ces codes. Exemple : ***$45_{(d)}$ ou $45_{(10)}$***

Base 2 ou ***binaire. 2***symboles différents existent dans cette base.

C'est le système que les machines utilisent pour compter. Le signal est présent ou absent dans un fil électrique.

0 ; 1

Si on veut aller plus loin dans le comptage, on combine ces codes. Exemple : ***$1011_{(b)}$ ou $1011_{(2)}$***

Base 16 ou ***hexadécimal. 16***symboles différents existent dans cette base.

C'est le système que l'on utilise pour programmer les machines puis qu'il s'agit d'un pseudo-binaire ou chaque code hexadécimal correspond à 4 codes binaires.

0 ; 1 ; 2 ; 3 ; 4 ; 5 ; 6 ; 7 ; 8 ; 9 ; A ; B ; C ; D ; E ; F

Si on veut aller plus loin dans le comptage, on combine ces codes. Exemple : ***$A12 ou $A12_{(h)}$ ou$A12_{(16)}$***

III- Définitions préliminaires :

Bit = unité la plus simple dans un système de numération qui ne peut prendre que 2 valeurs (souvent 0 ou 1).

Quartet= 4 bits (1 code hexadécimal).

Octet(Byte) = 8bits.

Mot(word) = 2 octets (soit 16 bits).

Long mot(Long) = 2 mots soit 32 bits.

M.S.B.= (Most Signifiant Bit) = bit de poids le plus fort dans un nombre codé en binaire.

L.S.B.= (Least Signifiant Bit) = bite de poids le plus faible dans un nombre codé en binaire.

On verra plus loin ce qu'est un poids.

IV- Correspondance entre les bases de numération :

Tableau de correspondance :

Tout comme en décimal, on définit un ordre de comptage pour les codes binaires et hexadécimaux.

Le code **Gray** (ou binaire réfléchi) a été développé pour fabriquer les tableaux de Karnaugh et pour éviter les aléas de fonctionnement des machines.

Le code **BCD** (Binaire Codé Décimal) est un code qui reprend les chiffres décimaux 1 à 1 et qui les convertit en binaire (permet l'affichage sur les montres, les radios réveils, les afficheurs 7 segments, …)

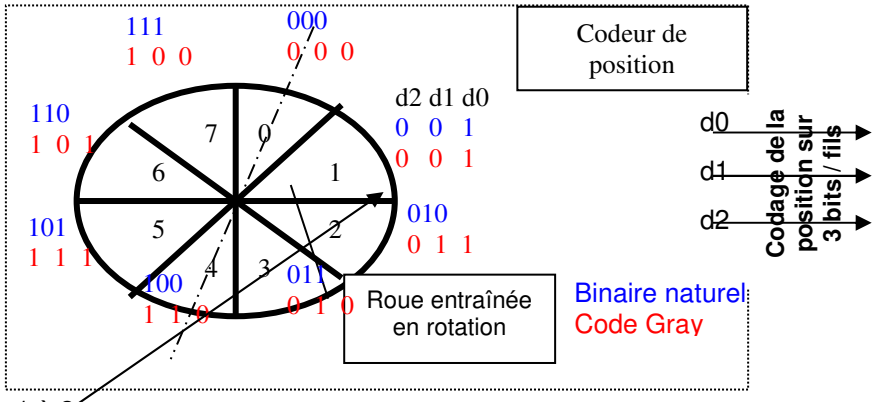
Décimal	Binaire	Hexadécimal	Binaire réfléchi ou code Gray	BCD
$00_{(10)}$	$0\ 0000_{(2)}$	$0\ 0_{(16)}$	$0\ 0000_{(Gray)}$	$00000000_{(BCD)}$
$01_{(10)}$	$0\ 0001_{(2)}$	$0\ 1_{(16)}$	$0\ 0001_{(Gray)}$	$00000001_{(BCD)}$
$0\ 2_{(10)}$	$0\ 0010_{(2)}$	$0\ 2_{(16)}$	0 0011	0000 0010
$0\ 3_{(10)}$	$0\ 0011_{(2)}$	$0\ 3_{(16)}$	0 0010	0000 0011
$0\ 4_{(10)}$	$0\ 0100_{(2)}$	$0\ 4_{(16)}$	0 0110	0000 0100
$0\ 5_{(10)}$	$0\ 0101_{(2)}$	$0\ 5_{(16)}$	0 0111	0000 0101
$0\ 6_{(10)}$	$0\ 0110_{(2)}$	$0\ 6_{(16)}$	0 0101	0000 0110
$0\ 7_{(10)}$	$0\ 0111_{(2)}$	$0\ 7_{(16)}$	0 0100	0000 0111
$0\ 8_{(10)}$	$0\ 1000_{(2)}$	$0\ 8_{(16)}$	0 1100	0000 1000
$0\ 9_{(10)}$	$0\ 1001_{(2)}$	$0\ 9_{(16)}$	0 1101	0000 1001
$1\ 0_{(10)}$	$0\ 1010_{(2)}$	$0\ A_{(16)}$	0 1111	0001 0000
$1\ 1_{(10)}$	$0\ 1011_{(2)}$	$0\ B_{(16)}$	0 1110	0001 0001
$1\ 2_{(10)}$	$0\ 1100_{(2)}$	$0\ C_{(16)}$	0 1010	0001 0010
$1\ 3_{(10)}$	$0\ 1101_{(2)}$	$0\ D_{(16)}$	0 1011	0001 0011
$1\ 4_{(10)}$	$0\ 1110_{(2)}$	$0\ E_{(16)}$	0 1001	0001 0100
$1\ 5_{(10)}$	$0\ 1111_{(2)}$	$0\ F_{(16)}$	0 1000	0001 0101
$1\ 6_{(10)}$	$1\ 0000_{(2)}$	$1\ 0_{(16)}$	1 1000	0001 0110

Aléa de fonctionnement :

Codage de la position d'un axe sur trois fils

Un aléa de fonctionnement c'est quand la technologie du capteur fait que le signal renvoyé par celui-ci risque de ne pas correspondre au signal attendu.

Exemple codeur absolu 3 bits (renvoie un signal sur 3 bits de la position d'un axe (en fonction du secteur))



De 1 à 2,
Or rien ne dit que ***techniquement on puisse les faire changer en même temps.***
=> On passera par ***un état faux 001-> 011 -> 010 ou 001 -> 000-> 010***
=> Pour éviter cela, on utilise ***le code Gray 1 seul fil à la fois changed'état.***

BTS ATI

NOM

PRENOM

COURS / SYNTHESE
TD / TP
TEST / EVALUATION
NOTE D'INFORMATION

Automatique et Informatique Industrielle

V- Techniques de conversion d'une base à une autre :

Comme on vient de le voir, une correspondance existe entre chaque base. Ainsi 5 en décimal donne 101 en binaire.

Comment convertir des nombres plus élevés sans avoir à faire un tableau très long ?

Binaire → décimal : Multiplications successive par 2 dans un tableau

Rang du bit	7	6	5	4	3	2	1	0
Mot Binaire bit à bit	0	0	1	0	0	0	1	1
Poids du bit = 2^{rang}	2^7 =128	2^6 =64	2^5 =32	2^4 =16	2^3 =8	2^2 =4	2^1 =2	2^0 =1
Produit = poids * bit	0*128=0	0*64=0	1*32=32	0*16=0	0*8=0	0*4=0	1*2=2	1*1=1
Somme des produits	0*128 + 0*64 + 1*32 + 0*16 + 0*8 + 0*4 + 1*2 + 1*1 = 35							

Nota : Ce tableau peut être utilisé pour le convertir du décimal en binaire

Décimal → binaire : Divisions euclidiennes successives par 2

35 : 2 = 17 r 1
17 : 2 = 8 r 1
8 : 2 = 4 r 0
4 : 2 = 2 r 0
2 : 2 = 1 r 0
1 : 2 = 0 r 1

$35_{(10)} = 10\ 0011_{(2)}$

Hexadécimal → binaire :

Conversion à l'aide du tableau page 3 de chaque symbole hexadécimal en 1 quartet

A 1 2 (16)
= 1010 0001 0010 (2)

Binaire → hexadécimal :

Regroupement des bits par quartets puis chaque quartet est converti en hexadécimal à l'aide du tableau page 3

1010 0001 0010 (2)
= A 1 2 (16)

VI- Nombre de combinaisons possibles en binaire naturel

Connaissant le nombre de bits n, le nombre de combinaisons N possible est $N = 2^n$.

On peut donc compter de 0 à $2^n - 1$

VII- Nombres négatifs en binaire :

Dans tous les cas, il faut préciser le nombre de bits utilisés. En effet, le bit le plus à gauche du nombre binaire correspond au signe (0 : signe positif – 1 signe négatif).

Plusieurs techniques existent pour coder un nombre binaire en négatif :

- Signe – magnitude

  Le bit 1 bit de signe + autant de bits que nécessaires pour donner la valeur binaire positive.

- Complément à 1

  On prend chaque bit du nombre binaire positif et on complémente tous les bits un à un ($\overline{0} = 1$ et $\overline{1} = 0$)).

- Complément à 2

  Seule cette méthode permet le calcul arithmétique.

  On prend chaque bit du nombre binaire positif et on complémente tous les bits un à un ($\overline{0} = 1$ et $\overline{1} = 0$)).

  Ensuite on ajoute la valeur 1 au mot ainsi obtenu (voir ci-dessous).

VIII- Exercices

Addition binaire : 1 + 1 égal 0 plus une retenue :

retenue	1		
		0	1
+		0	1
		1	0

1a - Que vaut 12 décimal en binaire sur 8 bits ?

1b- Détaillez le calcul pour convertir 125 décimal en binaire sur 8 bits.

1c- Effectuez l'opération décimale suivante en détaillant le calcul et les retenues :

	1	2	5
+	0	1	2
	1	3	7

1d- De la même façon, effectuez l'opération binaire suivante qui correspond à 125 + 12 en détaillant les retenues et en n'oubliant pas que vous n'avez que 2 signes différents (0 et 1) à votre disposition :

	0	1	1	1	1	1	0	1
+	0	0	0	0	1	1	0	0
	1	0	0	0	1	0	0	1

1e- Convertissez en décimal le résultat de cette opération et vérifiez qu'il est bien égal à $125+12$.

$2^7\ 2^6\ 2^5\ 2^4\ 2^3\ 2^2\ 2^1\ 2^0$

$1\ 0\ 0\ 0\ 1\ 0\ 0\ 1_{(2)} = 137_{(10)} = 125_{(10)} + 12_{(10)}$

2- Retrancher 126 à 312 en faisant l'opération en binaire sur 16 bits

$12_{(10)}$ = 0000 0001 0011 1000$_{(2)}$ $126_{(10)}$ = 0000 0000 0111 1110$_{(2)}$ -$126_{(10)}$ = 1111 1111 1000 0010 $_{(2)}$

$312_{(10)} + (-\ 126_{(10)}) = 186_{(10)}$ = 0000 0000 1011 1010$_{(2)}$