


# Objets Connectes : Programmation Microcontrôleur

Chapitre 2 : Concepts  
 Numération binaire  
 Architecture ALU  
 Programmation avec un microcontrôleur en langage C

1



## Annonces

- TD
  - Mise en ligne fin de semaine
  - Exercices a faire pour lundi, clôture dimanche soir
- Pour le TEA et les TP :
  - Pensez à consulter ressources en ligne Moodle
  - Pensez a venir avec smartphone chargé
  - Assurez-vous de pouvoir récupérer / transférer sur Moodle des photos prises avec votre smartphone :
    - Accès direct à Moodle avec le smartphone
    - Utilisation d'un stockage en ligne GoogleDocs, OneDrive, ...
    - Cable Usb – micro-USB ou USB-C
- Si vous souhaitez travailler avec votre propre PC, attendre instructions

2

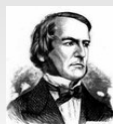
# Rappels & Concepts :

Numération binaire  
& logique combinatoire

3

## Binaire et logique combinatoire

- Principe de base de tous les systèmes informatiques
- Binaire et équations logiques :
  - Une variable logique peut prendre 2 états : vrai (true) ou faux (false). Ces 2 états possibles sont des **constantes logiques**
  - **Logique mathématique et calcul des propositions**
  - **Algèbre de Boole**



**George Boole** (1815 - 1864) est un logicien, mathématicien et philosophe britannique. Il est le créateur de la logique classique, fondée sur une structure algébrique définissant une sémantique, et que l'on appelle algèbre de Boole en son honneur.

4

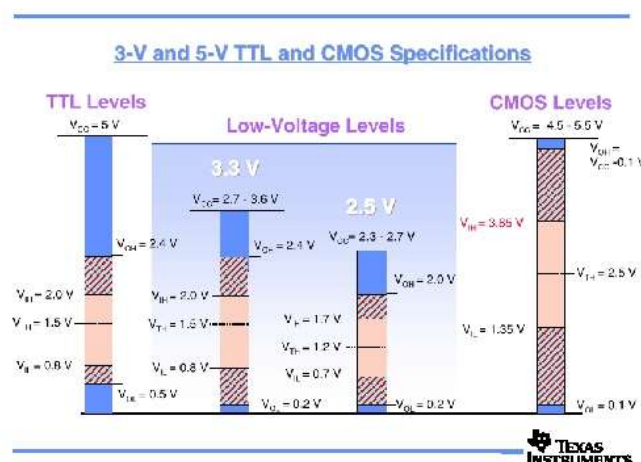
## Équivalents électriques

- Ces états sont aussi nommés :
  - état bas (low) ou état 0,
  - état haut (high) ou état 1
- Lorsque la logique combinatoire est concrétisée par des circuits électroniques :
  - Le niveau 0 correspond généralement au potentiel 0 (masse)
  - Le niveau 1 correspond :
    - A +5V pour la technologie TTL
    - A une tension nommée VDD, Vcc, V+ pour la logique CMOS, qui peut varier de 3 à 12V
  - Faible consommation (équipements portables, nomades), alors la tension correspondant au niveau 1 décroît : 3.3V, ... 1,2V

5

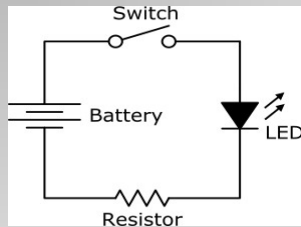
## Récapitulatif des tensions et seuils

### Voltage levels for different families



6

## Potentiel flottants

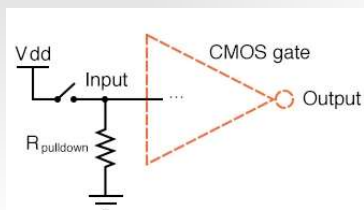
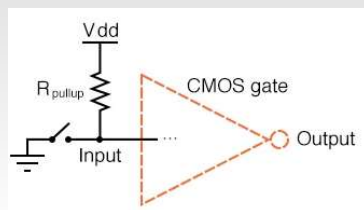
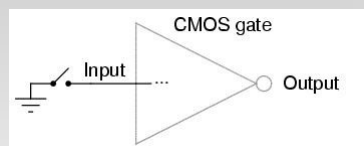


Que se passe t'il quand l'interrupteur est ouvert ?

Que se passe t'il quand l'interrupteur est fermé ?

OK quand l'interrupteur est fermé (0V = 0 logique)

Mais quel est le niveau logique quand l'interrupteur est ouvert ?



7

## Numération binaire

- Le système de numération décimale, ou de base 10, est utilisé pour compter.  
10 symboles : 0, 1, 2, 3, 4, 5, 6, 7, 8 et 9.
- Le système de numération binaire, ou de base 2, ne peut utiliser deux symboles (2 états) : 0 et 1

$10^4$	$10^3$	$10^2$	$10^1$	$10^0$
10000	1000	100	10	1
2	3	6	0	5

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

- Le binaire naturel** = système de numération à poids positionnels

$$x = p_n p_{n-1} \dots p_2 p_1 p_0 = p_n 2^n + \dots + p_2 2^2 + p_1 2^1 + p_0 2^0$$

par exemple:

$$x = 0110 = 0x2^3 + 1x2^2 + 1x2^1 + 0x2^0 = 4 + 2 = 6$$

8

## Numération hexadécimale

- Système de numération en base 16, avec donc 16 symboles : 0...9, A, B, C, D, E, F
- Intérêt principal : Ecriture de valeurs binaires dans le code

La conversion de binaire en hexadécimal se fait en regroupant les chiffres (les *bits*) quatre par quatre, ou inversement en remplaçant chaque chiffre hexadécimal par 4 chiffres binaires :

binaire	1.0101.1010.1010.1100.1111.0111						
regroupé par 4	1	0101	1010	1010	1100	1111	0111
regroupé en hexadécimal	1	5	A	A	C	F	7
hexadécimal	15AACF7						
(Décimal)	22719735						

```
int a = 0x15FA;
int b = a & 0xFFFE;
```

## Opérations arithmétiques

- Entiers non-signés : un mot de N bits peut représenter  $2^N$  valeurs :  

$$0 \leq x \leq 2^N - 1 \quad (\text{p.ex. } 0 \leq x \leq 2^4 - 1 = 15)$$

- Pour les nombres entiers **non-signés**, l'addition binaire est identique à l'addition décimale:

<sup>1</sup>				<sup>1</sup>	<sup>1</sup>
3	+			00011	+
27	=			11011	=
30				11110	

- Mots de taille fixe... si une opération engendre un débordement, ceci doit être traité.

11	+	1011	+
10	=	1010	=
21		10101	

### Codage binaire des nombres signés, tentative I

- Signe et valeur absolue :

$$00110 = (+) + 0x2^3 + 1x2^2 + 1x2^1 + 0x2^0 = 6$$

$$10110 = (-) + 0x2^3 + 1x2^2 + 1x2^1 + 0x2^0 = -6$$

- Précision (N bits):

$$-2^{N-1}-1 \leq x \leq 2^{N-1}-1 \quad \{\text{p.ex. } -7 = -(2^{4-1}-1) \leq x \leq 2^{4-1}-1 = 7\}$$

- Zéro:

$$000...00 = 0 \quad 100...00 = -0$$

### Codage binaire des nombres signés, tentative II

- Complément à un :

$$00110 = + 0x2^4 + 0x2^3 + 1x2^2 + 1x2^1 + 0x2^0 = 6$$

$$11001 = - 1x2^4 - 1x2^3 - 0x2^2 - 0x2^1 - 1x2^0 = -6$$

- Précision (N bits):

$$-2^{N-1}-1 \leq x \leq 2^{N-1}-1 \quad \{\text{p.ex. } -7 = -(2^{4-1}-1) \leq x \leq 2^{4-1}-1 = 7\}$$

- Zéro :

$$000...00 = 0 \quad 111...11 = -0$$

## Codage binaire des nombres signés, tentative III

- Complément à deux :

$$00110 = + 0 \times 2^4 + 0 \times 2^3 + 1 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 6$$

$$11010 = - 1 \times 2^4 - 1 \times 2^3 - 0 \times 2^2 - 1 \times 2^1 - 0 \times 2^0 - 1 = -6$$

- Précision (N bits):

$$-2^{N-1} \leq x \leq 2^{N-1}-1 \quad \{\text{p.ex. } -8 = -(2^{4-1}) \leq x \leq 2^{4-1}-1 = 7\}$$

- Zéro :

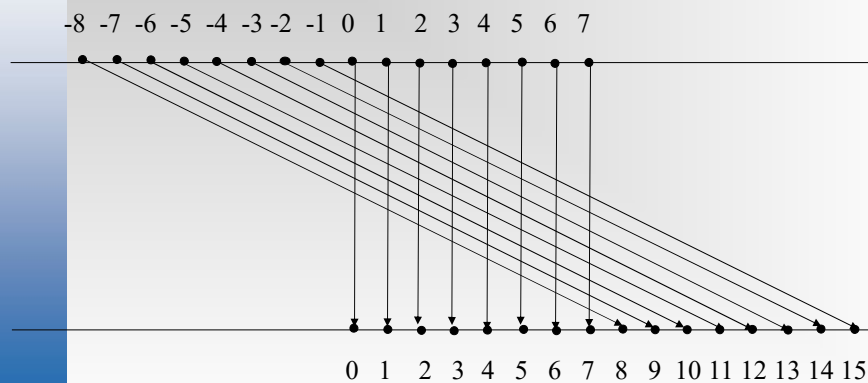
$$000\dots00 = 0 \quad 100\dots00 = ???$$

13

## Codage binaire - Nombres signés

$X > 0$	0	$X < 0$
$X$	0	$X + 2^k = \bar{X} + 1$

$k=4$



14

## Addition binaire - Nombres signés

- Pour le complément à deux, une soustraction peut être traitée par des addition.
- Les débordements doivent être traités différemment par rapport aux nombres non-signés.

13 -	01101 -	-13 -	10011 -
7 =	00111 =	7 =	00111 =
13 +	01101 +	-13 +	10011 +
- 7 =	11001 =	- 7 =	11001 =
6	01101 +	-20	10011 +
	11000 +		11000 +
	1 =		1 =
	100110		101100

Débordement?  
**No!**

Débordement?  
**Oui!**

15

## Nombre signés: Extension du signe

- Une solution pour éviter les débordements existe: on peut utiliser des mots plus grands (p.ex. 16 bits) pour stocker le résultat d'une opération sur des mots plus petits (p.ex. 8 bits). Mais dans le cas des nombres signés, une extension du signe devient nécessaire pour obtenir un résultat correct.

5 +	0101 +	-13 +	10011 +
7 =	0111 =	- 7 =	11001 =
12	1100 (-4)	-20	101100 (Err)
0000	1100 (12)	111	101100 (-20)

- Donc, attention au typage des données !
  - `short` différent de `unsigned short` !

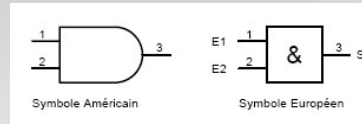
16



## Fonctions logiques : ET logique (AND)

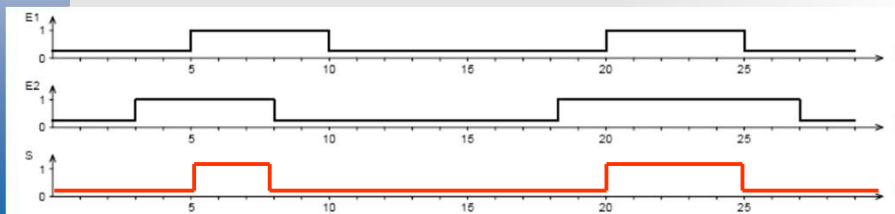
- Conjonction logique
- S est vrai si et seulement si e1 est vrai et e2 est vrai
- Montage en série

e1	e2	S
0	0	0
0	1	0
1	0	0
1	1	1



$$S = e1 \cdot e2$$

$$S = e1 \cap e2$$



17

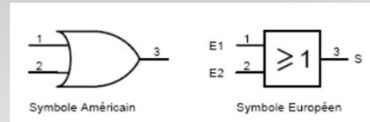
## Fonctions logiques : ET logique (AND)

- Propriétés de la fonction ET (AND)
  - Élément neutre est la constant logique 1 :  
 $A \cdot 1 = A$
  - Élément absorbant est la constante logique 0 :  
 $A \cdot 0 = 0$
  - L'opérateur ET est commutatif :  
 $A \cdot B = B \cdot A$
  - L'opérateur ET est associatif :  
 $A \cdot (B \cdot C) = (A \cdot B) \cdot C$
  - $A \cdot A = A$  (Idempotence)
  - $A \cdot \overline{A} = 0$  (Contradiction)

18

**Fonctions logiques : OU logique (OR)**

- Pour que la sortie soit à 1 :  
Il suffit qu'une entrée e1 OU e2 soit à 1
- Pour que la sortie soit à 0 :  
Il faut que toutes les entrées soient à 0
- Montage en parallèle

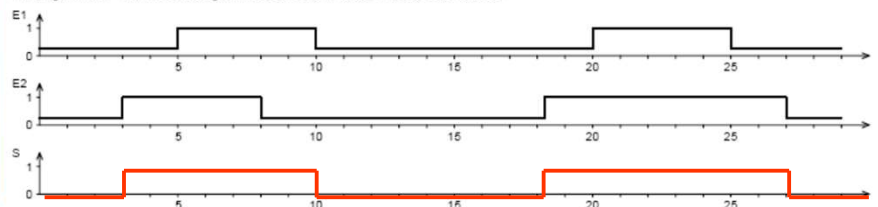


e1	e2	S
0	0	0
0	1	1
1	0	1
1	1	1

$$S = e1 + e2$$

$$S = e1 \cup e2$$

Chronogrammes : Établir le chronogramme de la sortie en fonction de celui des entrées.



19

**Fonctions logiques : OU logique (OR)**

- Propriétés de la fonction OU (OR)
  - Élément neutre est la constant logique 0 :  
 $A + 0 = A$
  - Élément absorbant est la constante logique 1 :  
 $A + 1 = 1$
  - L'opérateur OU est commutatif :  
 $A + B = B + A$
  - L'opérateur OU est associatif :  
 $A + (B + C) = (A + B) + C$
  - $A + A = A$  (Idempotence)
  - $A + \overline{A} = 1$  (Tautologie)

20

**Monotonie et complétude**

- Les opérateurs ET et OU sont monotones.

e1	e2	S
0	0	0
0	1	0
1	0	0
1	1	1

e1	e2	S
0	0	0
0	1	1
1	0	1
1	1	1

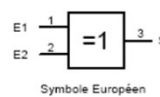
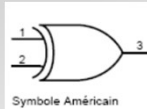
- On ne peut donc pas exprimer toute équation logique sans recourir à l'inverseur.
- Ensemble complet d'opérateur (ET, OU, NOT)
- Trouver un opérateur unique **complet** ?
  - NAND (Not AND)
  - NOR (Not OR)

21

**Fonctions logiques : OU exclusif (XOR)**

- OU Exclusif (eXclusive OR)

- Pour que la sortie soit à 1 :  
Il faut que e1 OU e2 soit à 1  
mais pas les 2
- Pour que la sortie soit à 0 :  
Il faut que  
les entrées soient  
au même niveau  
logique

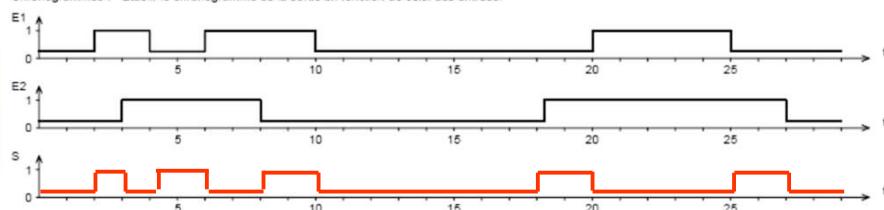


e1	e2	S
0	0	0
0	1	1
1	0	1
1	1	0

$$S = e1 \oplus e2$$

$$S = \bar{e1}.e2 + e1.\bar{e2}$$

Chronogrammes : Établir le chronogramme de la sortie en fonction de celui des entrées.



22

# Concepts :

Programmation des microcontrôleurs en  
langage C  
Environnement de développement

23

## Une programmation « proche du hardware »

- Contraintes matérielles fortes :
  - Place mémoire / Taille des données
  - Opérateurs (multiplication, div, modulo) non implémenté de façon native (petits microcontrôleurs)
  - L'accès à certaines opérations non prévue en langage C : interruptions, mise en veille, modification du registre d'état
  - Bibliothèques minimalistes / implémentation partielle (par ex: `printf`)
  - Attention à l'optimisation lors de la phase de compilation

```
while (PIN == OldPIN)
{ // wait for change in PIN
  ;
  // delay(50);
}
```

- Compilation croisée, téléversement, debug...

24

- Les notions basiques de la programmation en C doivent être acquises... si non : travail personnel
- Cours EPFL : rappel programmation avec l'utilisation du framework Arduino
  - <https://www.youtube.com/watch?v=Ec5CDRh8jlA>
  - <https://www.youtube.com/watch?v=Ec5CDRh8jlA>
- Cours EPFL : Programmation et types de données
  - <https://www.youtube.com/watch?v=0ycIEdpUf8o>
  - <https://www.youtube.com/watch?v=0ycIEdpUf8o>

- boolean (8 bits) - true/false
- char (8 bits) signed number from -128 to 127.
- byte (8 bits) unsigned number from 0-255 = unsigned char
- short (16 bits) signed number from -32768 to 32767.
- word (16 bits) unsigned number from 0-65535 = unsigned short
- int / unsigned int (16 or 32 bits)
- En general taille des mots manipulés par l'ALU
- long (32 bits) signed number from -2147483648 to 2147483647
- unsigned long (32 bits) unsigned number from 0 to 4294967295.
- float (32 bits) signed number from  $-3.4028235 \cdot 10^{38}$  to  $3.4028235 \cdot 10^{38}$
- Ne peuvent être manipulés tels quels par l'ALU -> appel de routines.

- Les opérateurs logiques & arithmétiques
  - Opérateurs à un paramètre:
    - - change le signe de la variable
    - ~complément à 1
    - ++/-- incrémentation/décrémentation
  - Opérateurs arithmétiques:
    - \*,/,+,-
    - % modulo (reste de la division entière)
  - Opérateurs sur bits:
    - <<,>> décalage à gauche ou à droite  
Exemple : `status = byte << 4;`
    - Décalage de 1 bit correspond à une division ou une multiplication par 2  
Attention, le résultat reste un nombre entier.

27

- ET logique, OU logique, XOR (ou exclusif), NOT
 

```
c = a & b;
c = a | b;
c = a ^ b;
c = ~a;
```
- Ne pas confondre avec les opérations booléennes
 

```
if ((a > 0) && !(a < 15))
```

**en C, le type booléen n'existe pas !**
- Il est souvent nécessaire de pouvoir mettre à 1 ou remettre à 0 un bit **particulier** dans un mot binaire:
 Opération binaire impliquant un **masque**
  - Comment mettre à 1 le 3<sup>ème</sup> bit du mot sans modifier les autres bits du mot ?
 

```
var = var | 0x04; // le C ne connaît que l'hexadécimal
```
- Que font les instructions suivantes :

```
c &= 0x01;
c = a | ~a;
```

```
c &= ~a;
c = a ^ a;
```

28

## Classe de stockage *volatile*

- Cette classe indique qu'une variable peut être modifiée en arrière-plan par un autre programme (par exemple par une interruption, par un thread, ou plus spécifiquement pour les microcontrôleurs lors de la lecture d'un port d'E/S)
  - La variable doit être « rafraichie » à chaque fois qu'on y fait référence dans un registre du processeur (relecture, même si la variable est dans un registre)
  - Interdiction de supprimer (optimisation, car nulle par dans le code il y a affectation).

```
void read_keyboard()
{
    volatile char keyboard_ch; /* a volatile variable */ . .
}

while (P1IN == OldP1IN)
{ // wait for change in P1IN ...
}
```

29

## Classe de stockage *const*

- Contenu d'une variable rendu non modifiable : **variable en lecture seule** sauf lors de la déclaration avec initialisation. Si const volatile, modification possible par une entité extérieure au programme.
- Attention :

```
const char str[] = "A string constant";
str[0] = 'a'; // n'est plus autorisé
```

- Une variable qui est une constante !
  - Meilleure écriture, permet de faire du code plus sûr ;
  - Permet de « piloter » le placement en mémoire

```
const char message[] = "Salut les gens !" ;
... //du code
monlcd.print(message) ;
```

30

- Une **fonction intrinsèque** (*intrinsics*) est une fonction disponible dans un langage de programmation donné dont l'implémentation est assurée par le compilateur même.
- Le langage C n'a pas été prévu pour microcontrôleur :
  - Gestion des interruptions
  - Mise en veille, sélection des modes de basse consommation
  - Certaines opérations arithmétiques ou de manipulation de bits :
    - Additions / soustraction en BCD
    - Permutation d'octets / nibbles
- Exemples : 

```
__disable_interrupt();  
__set_SP_register(unsigned short);
```