

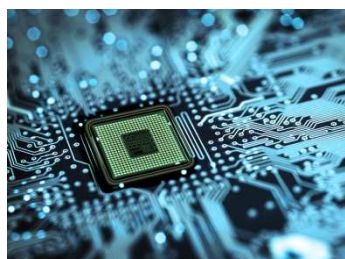
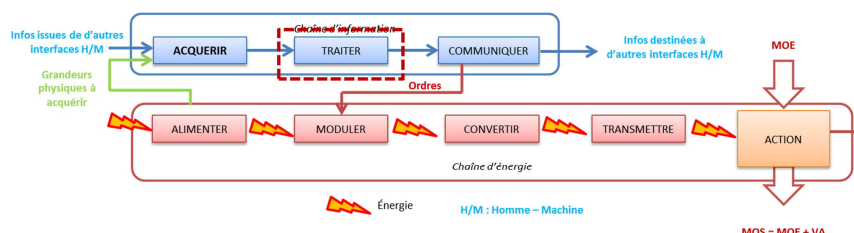
COURS

# CHAPITRE 1

## SYSTÈMES COMBINATOIRES

### Compétences Visées :

- A3-C8 : Description fonctionnelle des systèmes de traitement de l'information.
  - A3-C8.1 : Architecture générale de la chaîne d'information.
  - A3-C8 S2 : Identifier et décrire les composants associés au traitement de l'information.



Microprocesseur

<b>1</b>	<b>Unités de traitement dans les systèmes</b>	<b>2</b>
1.1	Fonction « Traiter »	2
1.2	Automate programmables industriels (API)	2
1.3	Cartes programmables – Carte Arduino	3
1.4	Traitement avec câblage électrique, pneumatique et hydraulique	3
<b>2</b>	<b>Codage des informations</b>	<b>4</b>
2.1	Codage en binaire et hexadécimal	4
2.2	Codage « Décimal codé binaire » DCB	4
2.3	Codage binaire réfléchi (Gray)	5
<b>3</b>	<b>Traitement des informations logiques – Algèbre de boole</b>	<b>5</b>
3.1	Système combinatoire	5
3.2	Table de vérité et chronogramme	6
3.3	Algèbre de Boole	6
3.3.1	Fonctions élémentaires	7
3.3.2	Propriétés et théorème de base	8
3.4	Établissement de fonction logique	8

## 1 UNITÉS DE TRAITEMENT DANS LES SYSTÈMES

### 1.1 Fonction « Traiter »

Lorsque l'information est acquise dans un système, l'unité de traitement réalise plusieurs opérations.

- ❑ Conversion analogique – numérique (CAN):
  - échantillonnage : cette opération consiste en un prélèvement de l'information à intervalle régulier. ;
  - blocage : pendant que le signal est converti, l'entrée est bloquée en l'état ;
  - codage : la valeur est codée en une information booléenne ou en information numérique. Suivant l'unité de traitement, le codage est limité à un certain nombre de bits, influant ainsi sur la valeur stockée.
- ❑ Stockage : une fois numérisée l'information est stockée en mémoire (mémoire flash, RAM, disque-dur...)
- ❑ Traitement : l'information peut alors être traitée à proprement parlé. Suivant les valeurs mesurées, l'unité de traitement pour alors modifier le comportement de la chaîne d'énergie.

Ces opérations vont être réalisées par un microcontrôleur ou un microprocesseur. Ces composants sont programmables.

#### Exemples de logiciels permettant de programmer des unités de traitement

Programmation par un langage  
« écrit »



Programmation par langage  
graphique



Programmation par diagramme  
d'état, programmation par  
schéma bloc



Programmation par logigramme :  
ISP Lever – Lattice Semiconductor



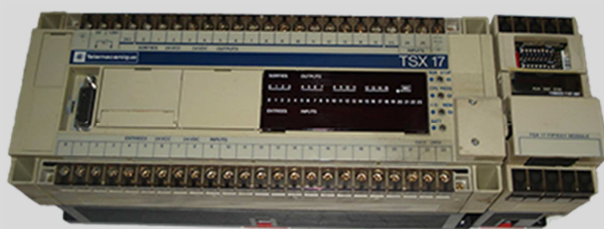
### 1.2 Automate programmables industriels (API)

Dans les systèmes industriels, le traitement de l'information est réalisé par un automate programmable industriel. Il s'agit d'un système électrique équipé d'entrées permettant de mesurer les états de capteurs ou de détecteurs et de sorties permettant de piloter des modulateurs d'énergie (relais électriques, distributeurs pneumatiques etc...)

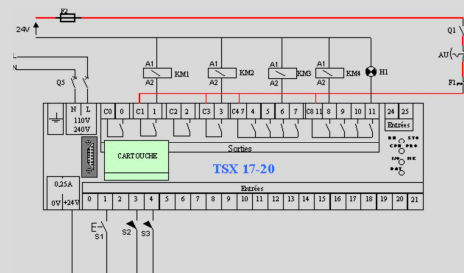
Le lien entre les entrées et les sorties se fait au moyen d'un programme (programme graphique, codage....).

#### Exemples de logiciels permettant de programmer des unités de traitement

Automate de la capsuleuse de bocaux



Automate TSX 17

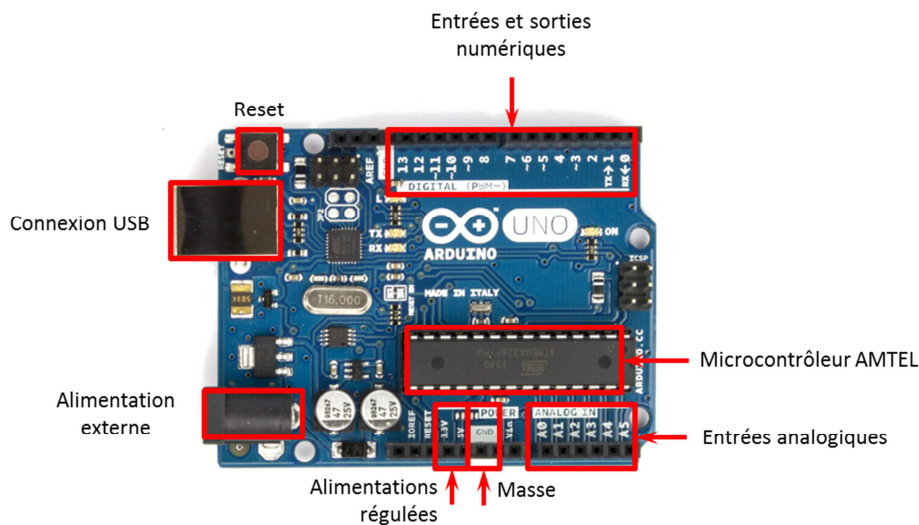


Représentation schématique des entrées et sorties

### 1.3 Cartes programmables – Carte Arduino

Les caractéristiques de la carte Arduino UNO sont les suivantes :

- ❑ Mémoire et microcontrôleur :
  - Microcontrôleur ATmega328 cadencé à 16 MHz
  - Mémoire Flash 32ko (dont 0,5 ko pour le système d'amorçage)
  - SRAM : 2ko
  - EEPROM 1ko
- ❑ Entrées sorties numériques
  - 14 dont 6 en MLI (PWM) indiquées ~ (40mA).
  - Ports Tx et Rx : reprise du port série
- ❑ Alimentation :
  - Alimentation par le port USB : 5V, 500mA
  - Alimentation externe en 7 à 12 V (2,1 mm)
  - Reprise de l'alimentation externe
  - Alimentation externe régulée en 5V/500mA et 3,3 V/50mA.
- ❑ Entrées analogiques :
  - 5 entrées analogiques

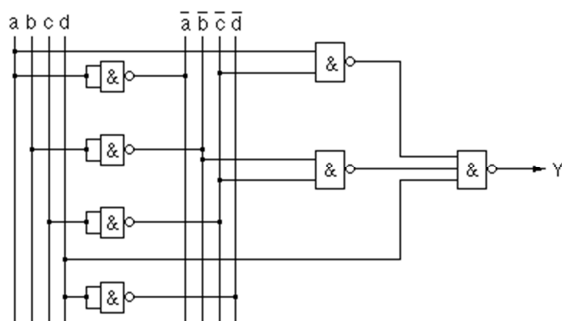


### 1.4 Traitement avec câblage électrique, pneumatique et hydraulique

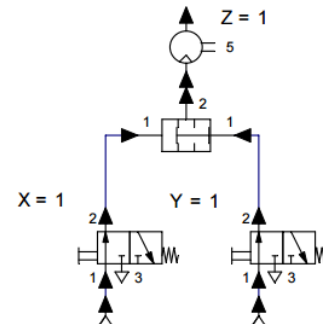
En utilisant des pré actionneurs électriques, pneumatiques ou hydrauliques il est possible de réaliser un traitement des informations en « logique câblée ».

Les câblages électriques avec des portes logiques (NON, ET, OU, NON ET, NON OU ...) ainsi que le câblage avec des pré actionneurs pneumatiques ou hydrauliques afin de traiter des équations booléennes ne se font quasiment plus.

En revanche, pour des raison de sécurité, il existe encore des résolutions d'équations logiques avec relais et interrupteurs dans les circuits électriques.



Plan de câblage en porte NAND



Plan de câblage logique câblée pneumatique

## 2 CODAGE DES INFORMATIONS

### 2.1 Codage en binaire et hexadécimal

Les conversions de nombres entiers (ou réels) vers les bases 2 et les bases 16 ont été vues en informatique. Nous ne reviendrons pas dessus.

Base 10	Base 2	Base 16
0	0 0 0 0	0
1	0 0 0 1	1
2	0 0 1 0	2
3	0 0 1 1	3
4	0 1 0 0	4
5	0 1 0 1	5
6	0 1 1 0	6
7	0 1 1 1	7

Base 10	Base 2	Base 16
8	1 0 0 0	8
9	1 0 0 1	9
10	1 0 1 0	A
11	1 0 1 1	B
12	1 1 0 0	C
13	1 1 0 1	D
14	1 1 1 0	E
15	1 1 1 1	F

**Exemple de conversion :**

$$(179)_{10} = (10110011)_2 = (B3)_{16}$$

### 2.2 Codage « Décimal codé binaire » DCB

C'est un code pondéré basé sur le code binaire naturel mais qui est adapté à la représentation des nombres en base 10. En effet le code binaire pur n'associe pas des bits spécifiques aux unités, dizaines, centaines, ... La propriété du code DCB est d'associer 4 bits différents à chaque puissance de 10.

**Exemple:**

$$(179)_{10} = (0001\ 0111\ 1001)_{DCB}$$

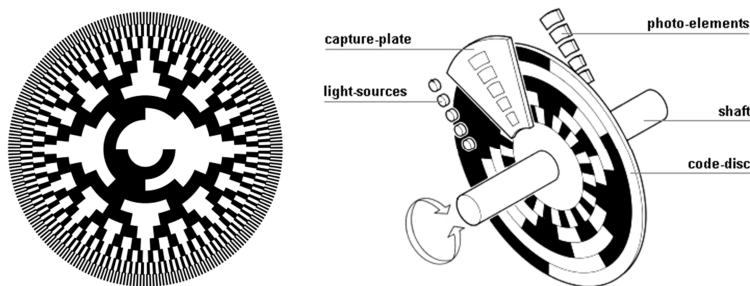
Ce code est utilisé pour les afficheurs 7 segments. Chaque afficheur reçoit le chiffre codé en binaire sur 4 bits. Cette représentation adaptée à la représentation binaire des nombres décimaux utilise un nombre de bits supérieur à celui du binaire naturel et donc une place plus importante en mémoire de l'ordinateur.



## 2.3 Codage binaire réfléchi (Gray)

Ce code non pondéré est un arrangement du système binaire. Le passage d'un nombre à l'autre se fait en changeant l'état d'un seul bit. Il est très utilisé pour décrire des automatismes (un changement d'état d'un composant correspond à un bit qui change), en particulier dans **les codeurs de position absolue**.

Avantage : il apporte une garantie d'interprétation avec une erreur maximale d'incréméntation.



Disque codage Gray

Décimal	Binaire pur	Binaire réfléchi
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

Pour obtenir le code Gray, il faut faire toutes les  $2^1, 2^2, 2^3 \dots$  lignes, une symétrie en commençant par le bit de droite et changer la valeur du bit de gauche.

## 3 TRAITEMENT DES INFORMATIONS LOGIQUES – ALGÈBRE DE BOOLE

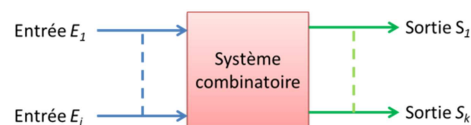
### 3.1 Système combinatoire

#### Définition :

Une variable  $a$  est binaire si et seulement si elle peut prendre, à chaque instant, qu'une seule valeur parmi un ensemble de 2 valeurs possibles.

#### Définition :

Un système logique combinatoire est un système binaire pour lequel à un état des variables d'entrée  $E_i$  correspond un unique état des variables de sortie  $S_j$ . (La réciproque n'est pas vraie.)



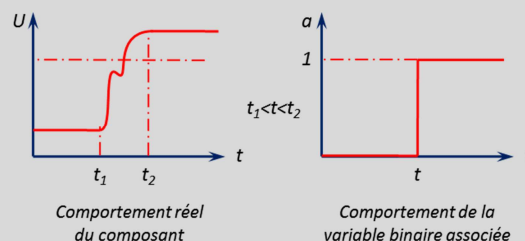
#### Exemple de systèmes ayant un fonctionnement combinatoire :

Digicode, codeur absolu.

#### Remarque:

Le comportement tout ou rien (TOR) ne correspond qu'au comportement normalement prévu en régime stabilisé et en l'absence de tout dysfonctionnement.

L'association d'une variable binaire à un composant ne peut pas rendre compte des états transitoires apparaissent entre deux états stables. C'est donc une simplification du comportement réel.



### 3.2 Table de vérité et chronogramme

#### Définition : Table de vérité

Une table de vérité exprime l'état de la variable de sortie en fonction des combinaisons des variables d'entrées. Chaque combinaison écrite sous forme binaire correspond à un état précis de variable sortie. Une table de vérité s'utilise principalement en logique combinatoire. Elle est représentée sous la forme d'un tableau dont le nombre de colonnes est égal au nombre d'entrée et de sortie, le nombre de ligne est égal à  $2^{(\text{le nombre d'entrée})}$ .

#### Définition : Chronogramme

Un chronogramme est la représentation graphique de variables binaires en fonction de temps :

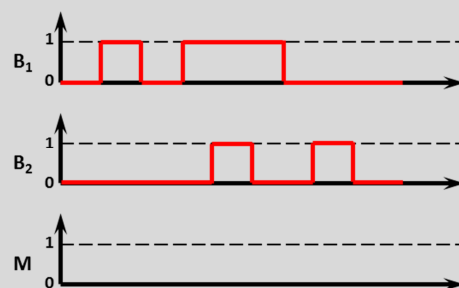
- ❑ comme une variable binaire ne peut prendre que deux états 0 ou 1, elle sera représentée par deux niveaux : bas (0) ou haut (1) ;
- ❑ pour analyser l'état des variables entre elles, l'échelle et l'origine des temps sont communes.

#### Exemple mise en marche d'un moteur avec bouton de sécurité

Pour des contraintes de sécurité, on souhaite qu'un moteur **M** soit mise en marche uniquement lorsque deux boutons **B1** et **B2** sont enclenchés.

1. Donner la table de vérité.
2. Compléter le chronogramme.

$B_1$	$B_2$	$M$
0	0	
0	1	
1	0	
1	1	



#### Définition : Front montant – front descendant

On souhaite parfois détecter le passage d'un état à l'autre :

- ❑ on appelle front montant le passage de l'état bas à l'état haut et on note  $\uparrow E$  ;
- ❑ on appelle front descendant le passage de l'état haut à l'état bas et on note  $\downarrow E$ .

### 3.3 Algèbre de Boole

C'est un algèbre de propositions logiques mise au point par un mathématicien anglais, Georges Boole (1815 – 1864).

#### Définition :

Un ensemble  $E$  a une structure d'algèbre Boole si on a défini dans cet ensemble :





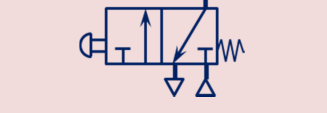
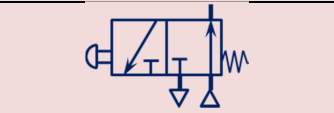
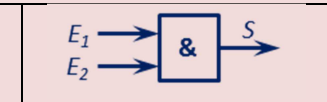
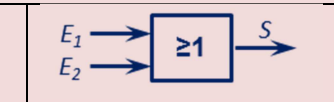
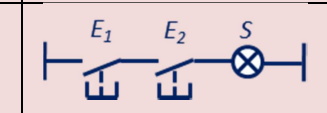
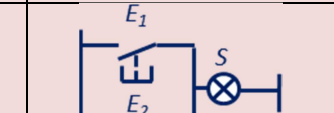
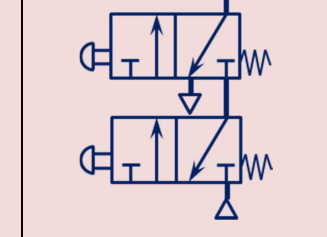
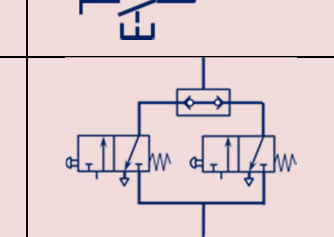
- ❑ une relation d'équivalence notée  $=$  ;
- ❑ deux lois de composition interne  $+$  (addition booléenne) et  $\cdot$  (multiplication booléenne) ;
- ❑ une loi appelée complémentation ( $\bar{a}$  complément de  $a$ ).

Une algèbre binaire est une algèbre de Boole dont les éléments  $B$  ne peuvent prendre que deux valeurs notées 0 ou 1 :

$$B = [0,1].$$

### 3.3.1 Fonctions élémentaires

#### Fonctions élémentaires:

Fonction OUI			Fonction NON – Complément		
Table de vérité		Équation logique	Table de vérité		Équation logique
Entrée $E$	Sortie $S$	$S = E$	Entrée $E$	Sortie $S$	$S = \bar{E}$
0	0		0	1	
1	1		1	0	
Logigramme			Logigramme		
Schéma électrique			Schéma électrique		
Schéma pneumatique			Schéma pneumatique		
Fonction ET – Produit booléen			Fonction OU – Addition booléenne		
Table de vérité		Équation logique	Table de vérité		Équation logique
$E_1$	$E_2$	$S$	$E_1$	$E_2$	$S$
0	0	0	0	0	0
0	1	0	0	1	1
1	0	0	1	0	1
1	1	1	1	1	1
Logigramme			Logigramme		
Schéma électrique			Schéma électrique		
Schéma pneumatique			Schéma pneumatique		

### 3.3.2 Propriétés et théorème de base

#### Propriétés :

<input type="checkbox"/> Commutativité	$a + b = b + a$	$a \cdot b = b \cdot a$	
<input type="checkbox"/> Distributivité	$a \cdot (b + c) = a \cdot b + a \cdot c$	$a + (b \cdot c) = (a + b) \cdot (a + c)$	
<input type="checkbox"/> Associativité	$a + (b + c) = (a + b) + c = a + b + c$	$a \cdot (b \cdot c) = (a \cdot b) \cdot c = a \cdot b \cdot c$	
<input type="checkbox"/> Élément neutre	$a + 0 = a$	$a \cdot 1 = a$	
<input type="checkbox"/> Élément absorbant	$a + 1 = 1$	$a \cdot 0 = 0$	
<input type="checkbox"/> Complémentarité	$a + \bar{a} = 1$	$a \cdot \bar{a} = 0$	$\bar{\bar{a}} = a$
<input type="checkbox"/> Idem potence	$a + a = a$	$a \cdot a = a$	
	<b>Absorption :</b>	<b>Inclusion :</b>	
<input type="checkbox"/> Identité remarquable	$a + a \cdot b = a$	$a + \bar{a} \cdot b = a + b$	

#### Théorème de de Morgan :

$$\overline{\sum_{i=1}^n a_i} = \prod_{i=1}^n \bar{a}_i$$

$$\overline{\prod_{i=1}^n a_i} = \sum_{i=1}^n \bar{a}_i$$

### 3.4 Établissement de fonction logique

#### Méthode

Pour établir une fonction logique, il est nécessaire d'établir la table de vérité.

#### Somme canonique

Pour établir une équation logique :

- ☐ on exprime chacune des combinaisons (en fonction de toutes les entrées) pour lesquelles la sortie vaut 1 ;
- ☐ on fait la somme logique (ET) de toutes les combinaisons.

#### Produit canonique

Dans cette méthode, on va exprimer le complément de la sortie :

- ☐ on exprime chacune des combinaisons (en fonction de toutes les entrées) pour lesquelles la sortie vaut 0 ;
- ☐ on fait la somme logique (ET) de toutes les combinaisons ;
- ☐ on complémente l'expression.