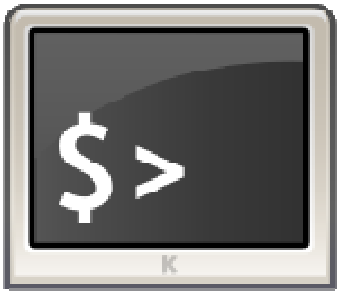
Cours

CHAPITRE 15

Génie Informatique et automatisme



CHAPITRE 15

Génie Informatique et automatisme

Annexes :

GENIE INFORMATIQUE ET AUTOMATISME

Logique : portes logiques

1

1 – PREAMBULE

Pour fonctionner de manière autonome, un système complexe doit prendre des décisions et envoyer des ordres en fonction des conditions présentes dans son environnement.

⇒ C'est la fonction « **TRAITER** » qui réalise cette tâche.

Exemple :

*Un store automatique doit se baisser **si** le soleil est présent et doit se remonter automatiquement **si** le vent se lève.*

Pour formuler des conditions répondant à un fonctionnement donné, les systèmes utilisent les opérateurs logiques. Un opérateur logique va effectuer une opération logique élémentaire entre des grandeurs binaires pour donner un résultat sous forme de grandeur binaire, c'est-à-dire valant 0 ou 1.

2 – OPERATEURS LOGIQUES DE LA LOGIQUE COMBINATOIRE

En logique combinatoire, nous avons affaire à 3 opérateurs logiques qui permettent la combinaison des variables binaires :

- Opérateur **complément** qui prend la variable et la complémente c'est-à-dire qu'un *0* devient un *1*
- Opérateur **ET** qui ne donne un 1 que lorsque l'ensemble des variables d'entrée vaut *1*.
- Opérateur **OU** qui donne un 1 lorsqu'au moins une des variables d'entrées vaut *1*.

La combinaison de ces opérateurs peut donner lieu à des fonctions plus ou moins complexes.

3 - PORTES / FONCTIONS LOGIQUES

Les principales fonctions logiques réalisées à partir des opérateurs cités précédemment sont données dans le tableau 1 de la page suivante. Dans ce tableau, on trouve :

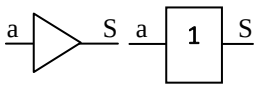
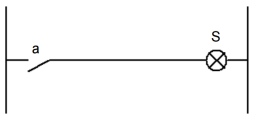
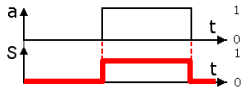
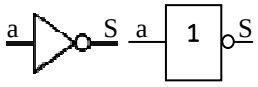
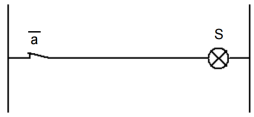
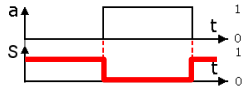
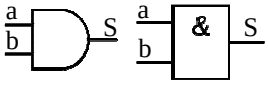
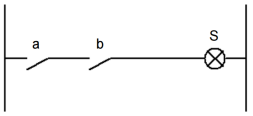
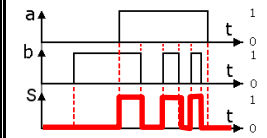
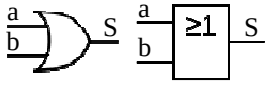
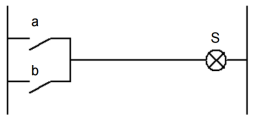
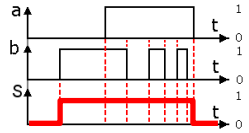
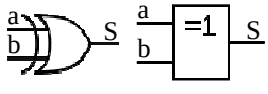
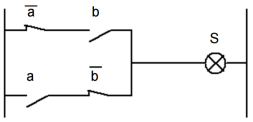
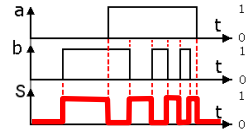
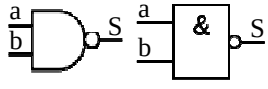
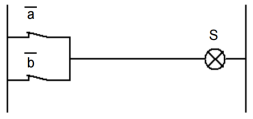
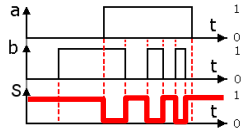
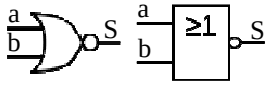
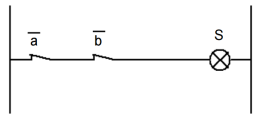
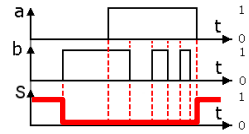
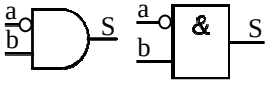
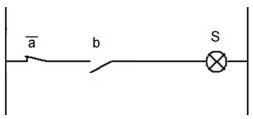
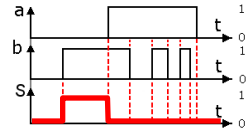
- leur nom ;
- leur modèle mathématique sous forme d'équation ;
- leur modèle graphique sous forme de logigramme ;
- leur modèle graphique sous forme de schéma à contact ;
- et leur comportement sous forme de chronogramme.

La représentation donnée utilise des portes à 2 entrées maximum. Il existe, bien entendu des portes logiques à plus de 2 entrées. Leur fonctionnement est « identique » à celui des portes à 2 entrées. Cette remarque est encore plus vraie de nos jours car la fonction traiter est de moins en moins réalisée par un câblage (eh oui, on peut créer des systèmes logiques à partir de circuits électroniques discrets reliés entre eux par des câbles ou des composants pneumatiques reliés entre eux par des tuyaux), mais plus à l'aide de composants programmés (microcontrôleurs, automates, FPGA, EPLD, PAL, …).

Portes universelles : Certaines portes sont dites universelles (comme les NOR et NAND). C'est-à-dire qu'à partir de ces portes logiques, on peut réaliser n'importe quelle autre porte.

Logigrammes homogènes : ils ne font appel qu'à un seul type de porte (que des NAND par exemple).

Tableau 1 : Récapitulatif des principales portes logiques

Fonction	Equation	Symboles logiques	Schéma à contacts	Table de vérité	Chronogramme
OUI	$S = a$			ENTREE / SORTIE a / S = a 0 / 0 1 / 1	
NON	$S = \overline{a}$			ENTREE / SORTIE a / S = $\overline{a}$ 0 / 1 1 / 0	
ET	$S = a \cdot b$			ENTREES / SORTIE a / b / S = a.b 0 / 0 / 0 0 / 1 / 0 1 / 0 / 0 1 / 1 / 1	
OU	$S = a + b$			ENTREES / SORTIE a / b / S = a+b 0 / 0 / 0 0 / 1 / 1 1 / 0 / 1 1 / 1 / 1	
OU Exclusif	$S = a \oplus b$			ENTREES / SORTIE a / b / S =a⊕b 0 / 0 / 0 0 / 1 / 1 1 / 0 / 1 1 / 1 / 0	
NAND ●	$S = \overline{a \cdot b}$			ENTREES / SORTIE a / b / S = a.b 0 / 0 / 1 0 / 1 / 1 1 / 0 / 1 1 / 1 / 0	
NOR ●	$S = \overline{a + b}$			ENTREES / SORTIE a / b / S = $\overline{a+b}$ 0 / 0 / 1 0 / 1 / 0 1 / 0 / 0 1 / 1 / 0	
INHIBITION ●	$S = \overline{a} \cdot b$			ENTREES / SORTIE a / b / S = $\overline{a}$.b 0 / 0 / 0 0 / 1 / 1 1 / 0 / 0 1 / 1 / 0	

● Fonctions universelles

Parfois, au lieu d'être représenté par un  , le complément est représenté par 

GENIE INFORMATIQUE ET AUTOMATISME

Logique : schémas logiques

2

PREAMBULE

Pour réaliser le fonctionnement d'un système automatisé, plusieurs solutions du **point de vue électrique** peuvent être utilisées :

- La logique à contacts : utilisation de contacts ;

- La logique électronique : utilisation de portes logiques (logigramme);

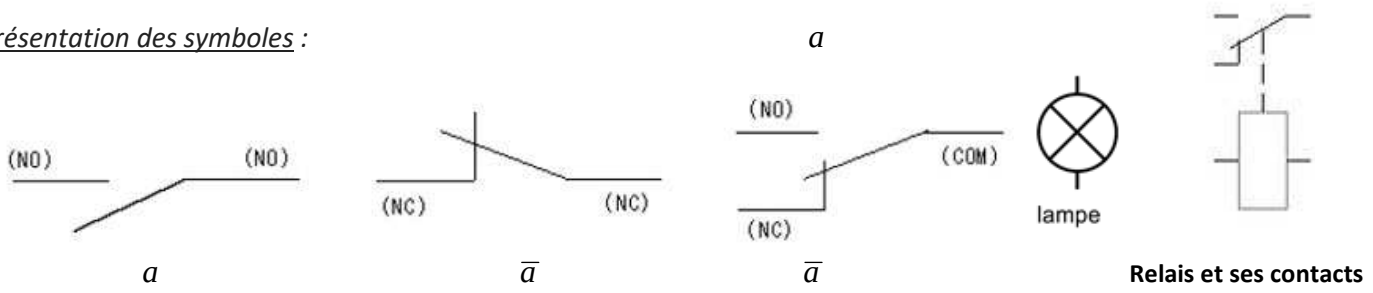
- La logique programmée : utilisation d'un automate programmable ou microcontrôleur ou autre ;

Pour n'importe quelle solution choisie, on réalise **un schéma logique** qui est destiné à faciliter l'étude et la compréhension du fonctionnement du système.

1 - LA LOGIQUE A CONTACTS

Elle utilise des contacts électriques de type Normalement Ouvert (ou normally open) NO ou de type Normalement Fermé (ou normally close) NF ou NC. Le fonctionnement du système peut être simulé par une lampe, un relais.

Représentation des symboles :



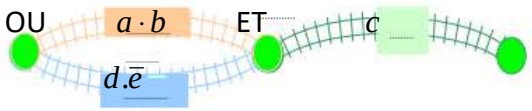
Relais et ses contacts

NOTA : cette représentation est destinée pour des schémas horizontaux. Il suffit de faire une rotation de $-\frac{\pi}{2}$ *des symboles pour une représentation d'un schéma vertical.*

Exemple : $S = [(a \cdot b) + (d \cdot \bar{e})].c$ (*prendre en compte en priorité les groupements entre les crochets*).

En faisant des groupements tels que $A = a \cdot b$; $B = d \cdot \bar{e}$ et $C = c$, cette équation peut alors s'écrire $S = [A + B].C$

Donc



Avec

$A = a \cdot b$ soit 

$B = d \cdot \bar{e}$ soit 

$C = c$ soit 

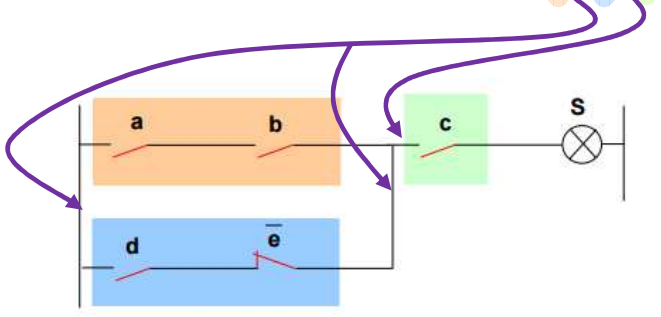


Schéma logique à contacts (ou schéma électrique)

2 - LA LOGIQUE ELECTRONIQUE AVEC PORTES LOGIQUES : LE LOGIGRAMME

Elle utilise un ensemble de portes logiques. On l'appelle le logigramme. Le fonctionnement du système peut être simulé par LED (diode électroluminescente).

Représentation des symboles : voir fiche 02 portes logiques

Exemple : $S = [(a \cdot b) + (d \cdot \bar{e})].c$

En faisant des groupements tels que $A = a \cdot b$; $B = d \cdot \bar{e}$ et $C = c$, cette équation peut alors s'écrire $S = [A + B].C$

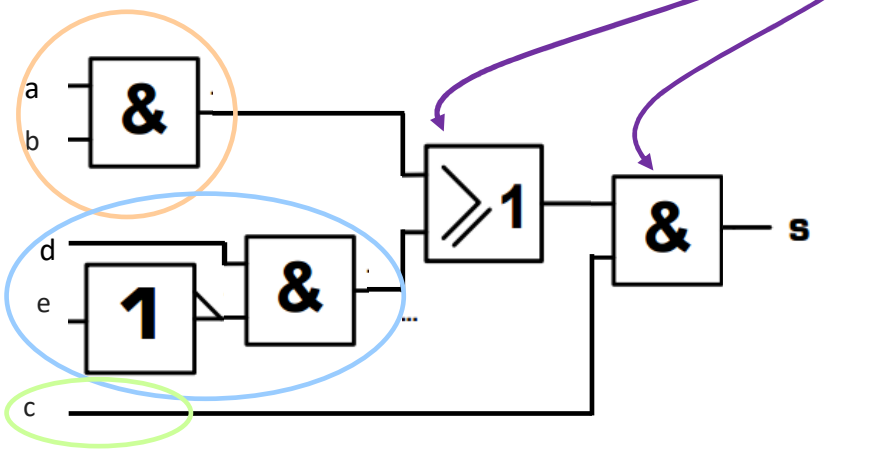
Donc

Avec

$A = a \cdot b$ soit a, b

$B = d \cdot \bar{e}$ soit d, e

$C = c$ soit c



Logigramme

3 - LA LOGIQUE PROGRAMMEE

Elle utilise un ensemble de contacts. Un schéma à contacts est un langage graphique très populaire auprès des automaticiens pour programmer les automates programmables industriels. On le nomme **schéma Ladder** (mot anglais pour échelle).

Représentation des principaux symboles :

- Les variables d'entrée a, b, c, etc sont appelées I_n comme INPUT

Variable d'entrée I_1 $\quad$ $\bar{I}_2$

- Les variables de sortie S, F,D, etc sont appelées O_n comme OUTPUT

Variable de sortie O_1

Exemple : $S = [(a \cdot b) + (d \cdot \bar{e})].c$ ce qui correspond en ladder à $O_1 = [(I_1 \cdot I_2) + (I_3 \cdot \bar{I}_4)].I_5$

En faisant des groupements tels que $A = I_1 \cdot I_2$; $B = I_3 \cdot \bar{I}_4$ et $C = I_5$, cette équation peut alors s'écrire $S = [A + B].C$

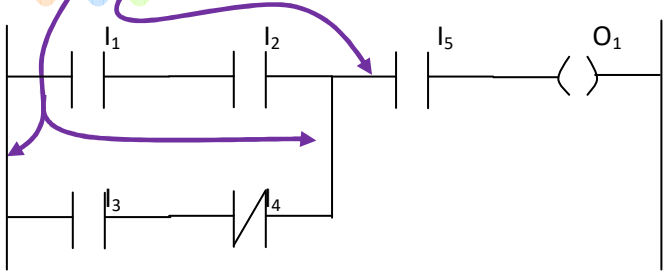


Schéma en LADDER

GENIE INFORMATIQUE ET AUTOMATISME

Logique : méthode de résolution

3

1 – MISE EN EQUATION DU BESOIN

Le fonctionnement d'un système répond nécessairement à un **cahier des charges**. Par exemple, on peut dire « Actionner les essuie-glaces automatiquement si il pleut » ; plus subtilement, on pourrait avoir « Actionner les essuie-glaces si la pluie est au moins égal à un débit donné et si la commande est en mode automatique (et non manuel) ».

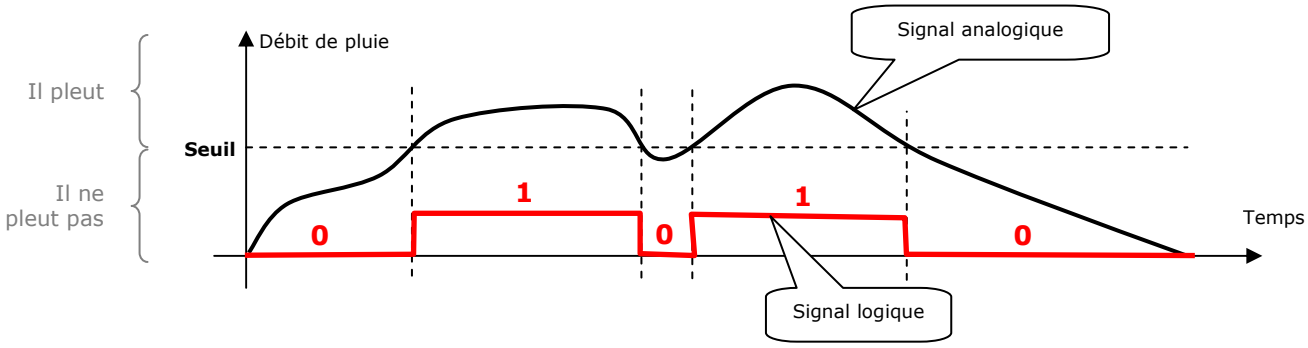
On voit bien dans cet exemple qu'une **variable de sortie** (l'actionnement des essuie-glaces) dépend de l'état d'un certain nombre de **variables d'entrée** (le débit de pluie et le mode enclenché, manuel ou automatique).

Ceci peut se mettre sous forme d'une équation dite logique ; en appelant « S » (majuscule) la sortie et « a » et « b » (minuscule) les variables d'entrée (débit de pluie et mode automatique), on pourrait avoir l'équation logique suivante :

$$S = a \; ET \; b \text{, c'est-à-dire } S = a \cdot b$$

Telle qu'écrite, cette équation suppose deux choses :

- tout d'abord, que le débit de pluie est logique, c'est-à-dire qu'il vaut « 0 » ou « 1 ». Or, un débit de pluie est généralement variable ! On peut certes dire « il pleut » ou « il pleut pas », de façon binaire, mais faut-il pour autant mettre les essuie-glaces en marche dès qu'une petite goutte est tombée ? Probablement que non. Ce faisant, il faut définir **un seuil** qui va permettre de déclarer si – oui ou non – il pleut suffisamment :



- Ensuite, cette équation suppose que la variable « b » est associé au mode automatique car il faut bien avoir $b = 1$ pour que l'action soit effective de façon automatique. Mais la variable « b » aurait très bien pu être associée au mode manuel, auquel cas il faudrait avoir $b = 0$ (c'est dire « ne pas être en mode manuel ») pour que l'action se réalise. Dans ce cas, pour dire que $S = 1$ si $a = 1$ ET $b = 0$, l'équation s'écrit : $S = a \cdot \overline{b}$.

Les équations logiques font référence à une algèbre spécifique appelée algèbre de BOOLE. Elle possède ses propres règles de calcul et de simplification.

Traiter un problème consiste donc à trouver les équations logiques qui régissent le fonctionnement attendu du système au regard du cahier des charges qu'on s'est donné.

2 – METHODE PRATIQUE POUR TRAITER UN PROBLEME DE A à Z

1) **Lire** le cahier des charges du système : *quelles actions sont attendues en fonction de quelles variables ?*

2) **Identifier** et **nommer** les variables d'entrée (en minuscule) et celles de sortie (en majuscule) si les noms ne sont pas déjà donnés.

3) **Réaliser** la table de vérité pour chaque action attendue (c'est-à-dire pour chaque sortie).

4) **Ecrire** l'équation logique à partir de la table de vérité en ne gardant que les lignes pour lesquelles la sortie est à l'état logique « 1 ».

5) **Simplifier** l'équation logique (méthode algébrique ou bien semi-graphique avec le tableau de KARNAUGH).

6) **Représenter** l'équation à l'aide d'un schéma électrique (si demandé).

7) **Représenter** l'équation à l'aide d'un schéma logique hétérogène (« OUI », « NON », « ET », « OU ») ou homogène (« NAND », « NOR ») (si demandé).

8) **Programmer** l'équation (pour le microcontrôleur ou directement dans *LabView* ou encore pour la carte *Arduino* ou tout autre matériel choisi ou imposé).

9) **Tester** le fonctionnement et vérifier s'il correspond au cahier des charges.

10) **Valider** ou **modifier** le programme si nécessaire.

GENIE INFORMATIQUE ET AUTOMATISME

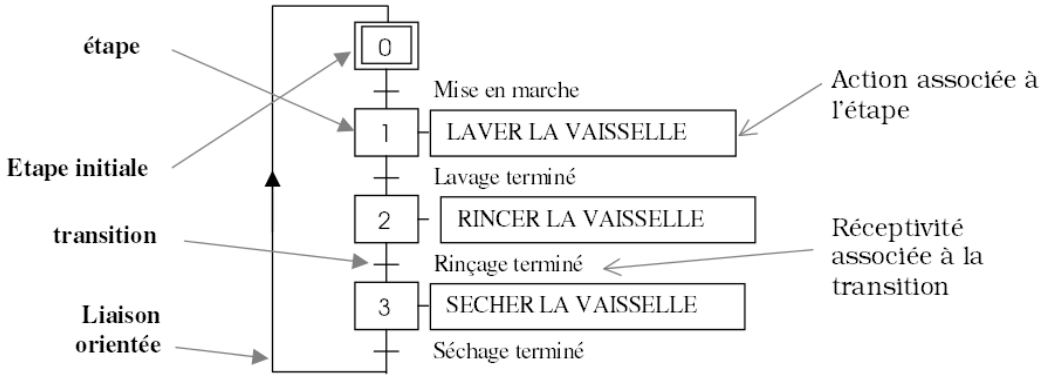
Langage « Grafcet »

5

1 – PREAMBULE

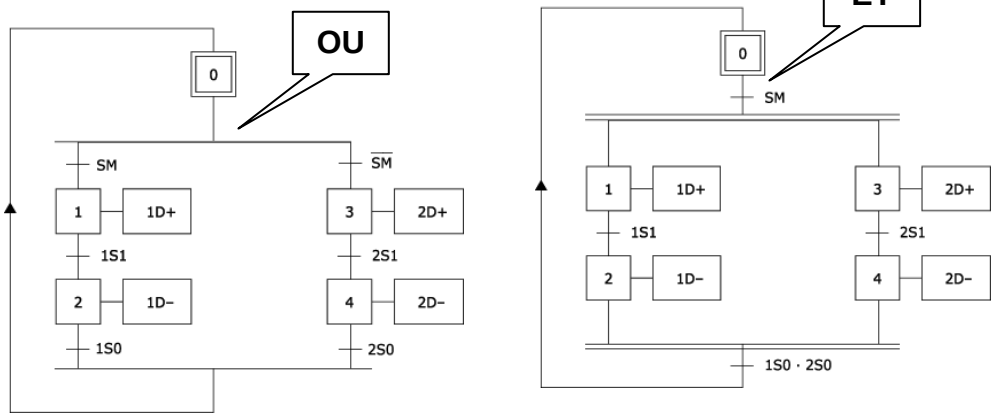
Les **systèmes industriels** sont pilotés à l'aide d'un « **Automate Programmable Industriel** » (API) dont le rôle est de gérer l'état des sorties (Output) en fonction de l'état des entrées (Input). Ce traitement est fait par un programme appelé **GRAFCET**.

2 – STRUCTURE GENERALE D'UN GRAFCET (GRAPHE FONCTIONNEL DE COMMANDE ETAPE/TRANSITION)



- Un Grafcet se lit de haut en bas.
- Deux étapes doivent être séparées par une transition
- Deux transitions doivent être séparées par une étape.
- Une étape est soit active, soit inactive. Si une étape est active, les actions associées sont réalisées.
- Une réceptivité est une condition logique qui est soit vraie, soit fausse. Si une réceptivité est vraie, alors la transition à laquelle elle est associée est franchissable et obligatoirement franchie.
- Lorsqu'une transition est franchie, les étapes immédiatement précédentes deviennent inactives et les étapes immédiatement suivantes deviennent actives.

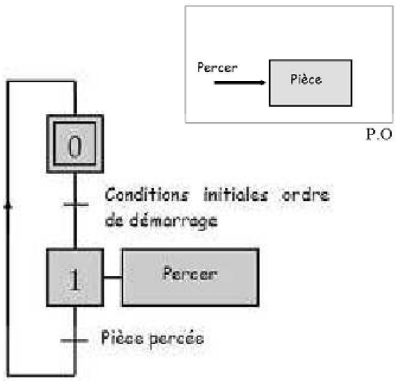
Divergence/Convergence



3 – LES DIFFERENTS POINTS DE VUE

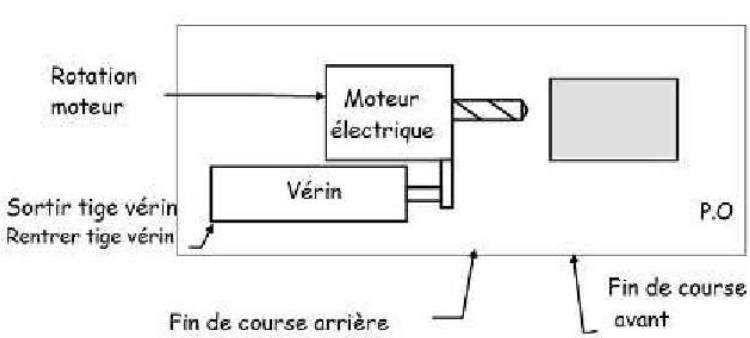
Grafcet « point de vue système »

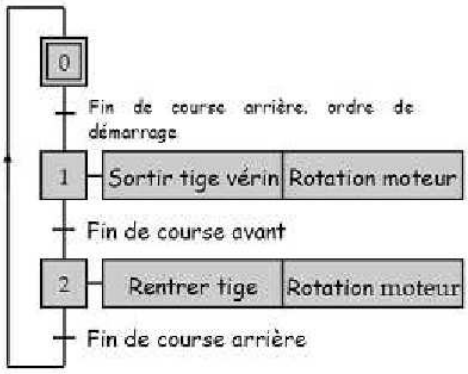
Le Grafcet point de vue système décrit les effets attendus sur la matière d'œuvre **sans préjuger des moyens techniques** pour effectuer ces opérations. Les actions sont définies à l'aide d'un verbe à l'infinitif et éventuellement d'un complément.



Grafcet « point de vue PO »

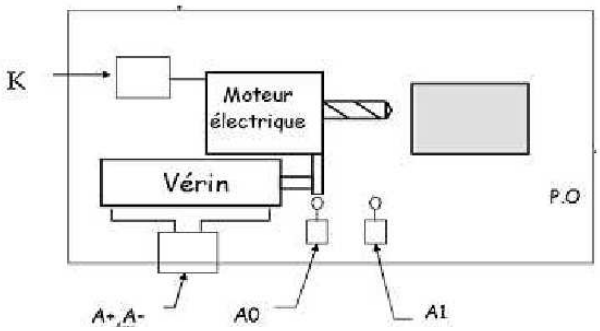
Le grafcet point de vue partie opérative décrit le fonctionnement attendu des actionneurs qui ont été préalablement définis (moteur, vérins).

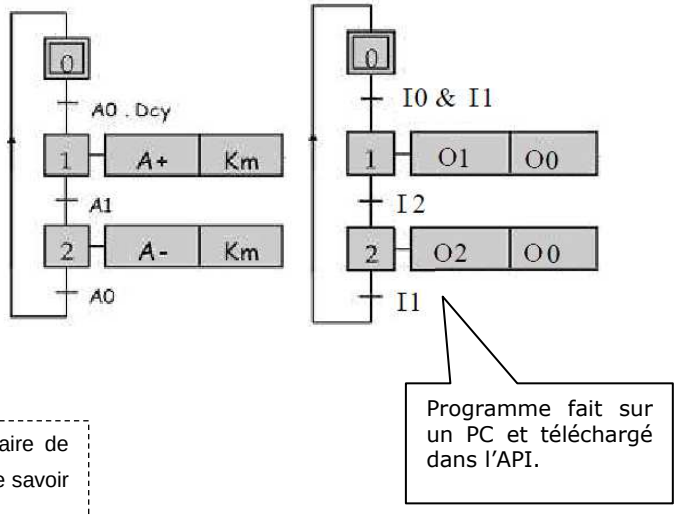




Grafcet « point de vue PC » (ou automate)

Le Grafcet point de vue partie commande décrit les ordres à donner aux préactionneurs (câblés sur les sorties de l'API) en fonction des informations délivrées par les capteurs (câblés sur les entrées de l'API).
Pour ce Grafcet tous les composants doivent donc être définis (actionneurs, préactionneurs, capteurs, IHM).





Remarque : une fois que tout est câblé sur l'API, il est nécessaire de disposer d'une **« *table d'affectation des Entrées/Sorties* »** afin de savoir qui est câblé où.
Cette table se trouve généralement dans le dossier technique du système.

GENIE INFORMATIQUE ET AUTOMATISME

Systèmes à base de microprocesseurs

6

1 – PREAMBULE

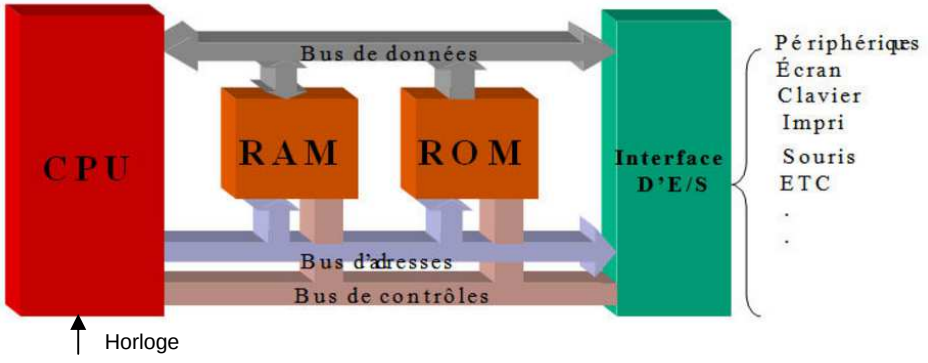
Pour fonctionner de manière autonome, un système complexe doit prendre des décisions en fonction des conditions présentes dans son environnement. C'est la fonction « **TRAITER** » qui réalise cette tâche.
Ce type fonction peut être réalisé avec des systèmes à base de microprocesseurs.
Leur fonctionnement fait appel à la notion de **variables numériques** ou de **variables booléennes**.

2 – ARCHITECTURE DE BASE D'UN SYSTEME A BASE DE MICROPROCESSEUR

Pour fonctionner un système à base de microprocesseur doit contenir (voir figure suivante) :

- Un microprocesseur (**UNITE CENTRALE OU CPU**) ;
- Un ou des composant(s) permettant de stocker des données (résultats de calculs) ou des programmes (**RAM / ROM / EPROM / PROM / …**) ;
- Un ou des composant(s) permettant de communiquer avec l'extérieur (**INTERFACE D'ENTREES / SORTIES**) ;
- Un ensemble de fils permettant de faire communiquer les informations d'un composant à l'autre (**BUS DE DONNEES**)
- Un ensemble de fils permettant de sélectionner le composant auquel le microprocesseur souhaite s'adresser (**BUS D'ADRESSES**) ;
- Un ensemble de fils permettant de contrôler le sens d'échange entre le CPU et les composants annexes (**BUS DE CONTROLES**) ;
- Des composants supplémentaires (décodage d'adresses, horloge, …).

Figure 1 : architecture de base

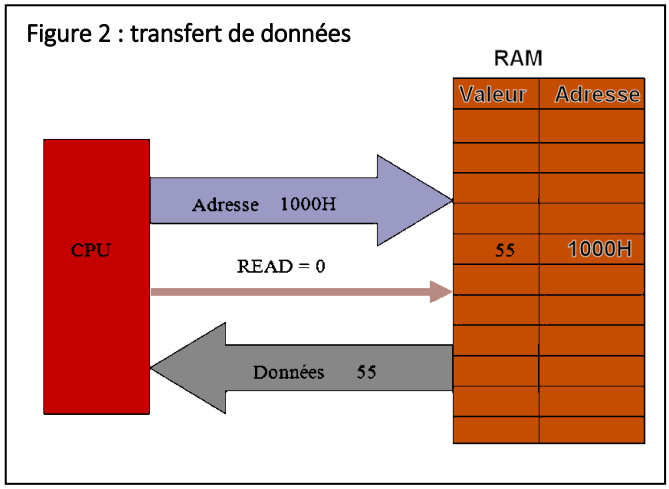


Un bus est un ensemble de fils qui sont mis à 1 ou à 0 en fonction des données qui doivent circuler dessus.

Exemple de transfert de données (la CPU veut lire une information contenue en RAM) :

1/ la CPU positionne le bus d'adresses pour s'adresser à une zone mémoire de la RAM (exemple adresse 1000_h) ;
2/ ensuite il indique sur le bus de contrôles s'il veut écrire ou lire (pour l'exemple lire) ;
3/ la RAM, qui comprend qu'on s'adresse à elle et qu'on veut lire la valeur contenue dans l'adresse 1000_h dans une zone particulière, met sur le bus de données la valeur qui est contenue dans cette adresse (par exemple la valeur 55) ;
4/ la CPU lit ce qui est contenu sur le bus de données.

Figure 2 : transfert de données



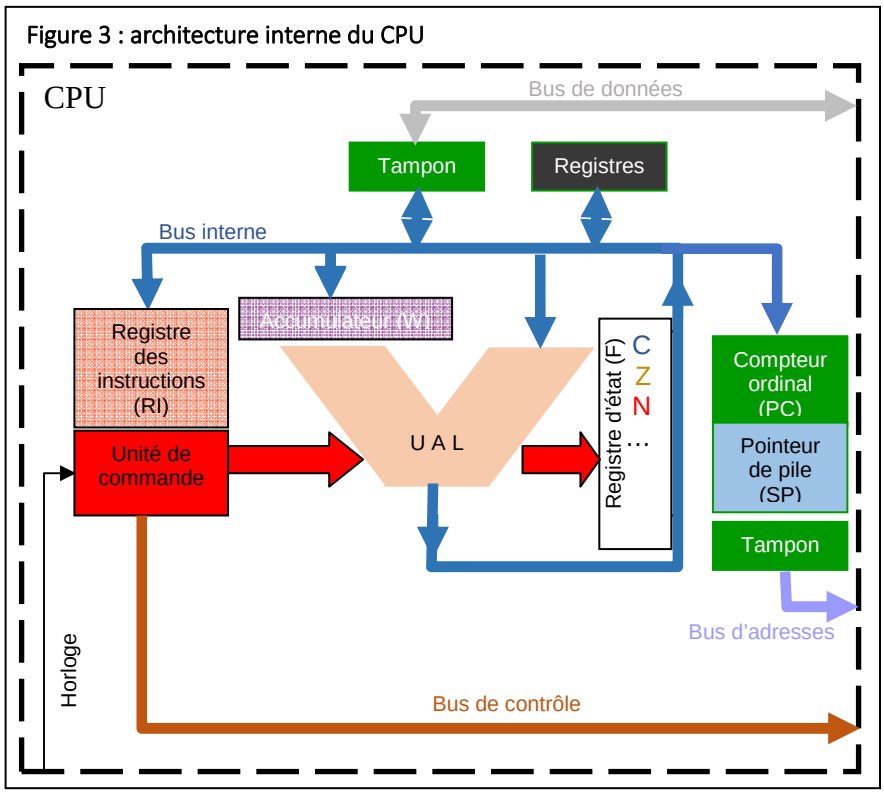
3 – MICROPROCESSEUR

Les microprocesseurs possèdent différents éléments (voir figure 3) :

- **une Unité Arithmétique et Logique** (U.A.L.) (qui réalise des calculs de base (addition, soustractions, … selon le jeu d'instructions du processeur) ;
- **des registres** pour stocker des données (utiles au calcul) dont un particulier appelé accumulateur ;
- **des registres particuliers** qui permettent de pointer sur les adresses mémoire en fonction du programme ;
- **un registre d'état** qui permet de donner des informations sur le résultat du calcul (s'il vaut zéro, si on a dépassé les capacités de l'U.A.L., …)
- …

Le fil d'horloge permet de cadencer le fonctionnement du processeur (c'est-à-dire traiter les instructions du programme les unes derrière les autres en suivant un rythme particulier.

Figure 3 : architecture interne du CPU



GENIE INFORMATIQUE ET AUTOMATISME

Microcontrôleurs

7

1 – PREAMBULE

Un microcontrôleur est un système à base de microprocesseur (voir la fiche génie informatique 1 sur les systèmes à bases de microprocesseur).
Un microcontrôleur contient l'ensemble des composants cités dans l'architecture d'un système à base de microprocesseur dans un seul boitier.

Dans la suite de cette fiche, nous parlerons du ATMEGA 328 qui est le microcontrôleur contenu dans la carte ARDUINO UNO.

2 – LE MICROCONTROLEUR ATMEGA 328

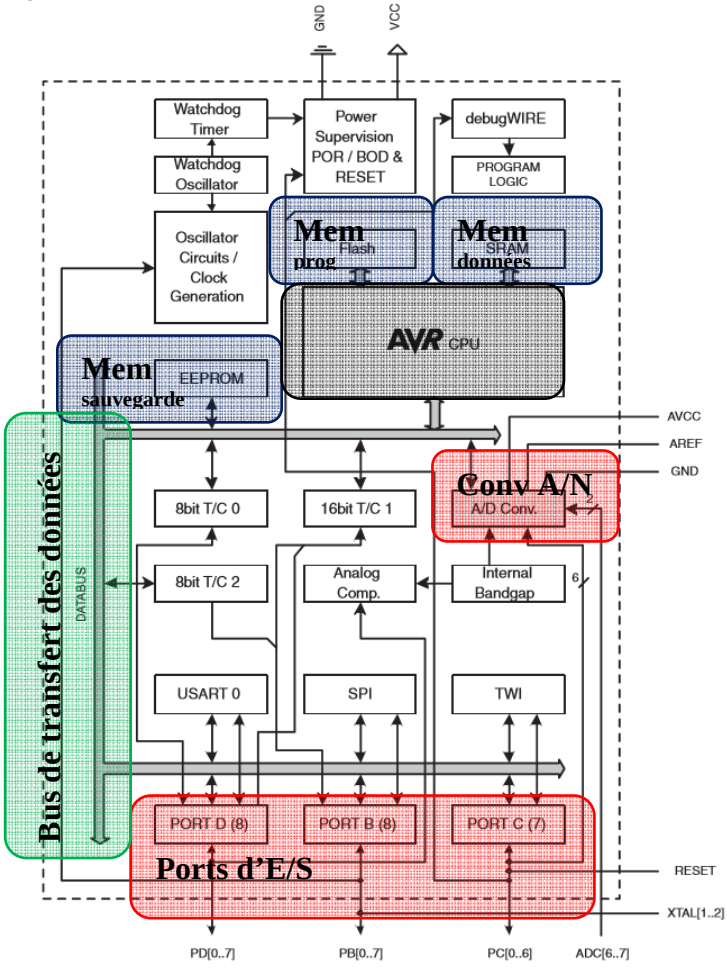
Le microcontrôleur ATMEGA 328 est basé sur une architecture de type R.I.S.C. (Reduced Instruction Set Computer). Son jeu d'instruction de base est simple ce qui lui confère une rapidité d'exécution de ces instructions élevée.

Il contient l'ensemble des éléments utiles au fonctionnement d'un microprocesseur, avec des composants supplémentaires permettant :

- de piloter des composants par des commandes Tout Ou Rien ;
- de piloter des hacheurs par MLI (Modulation de Largeur d'Impulsion) (grâce à ses broches PWM (Pulse Width Modulation)) ;
- de gérer des conversions analogiques/numériques sur broches d'entrée (broches analogiques) ;
- de gérer les transferts de données par bus série RS232, I2C, … ;
- …

Son architecture est la suivante (fig. 1) :

Figure 1 : Architecture du ATMEGA 328

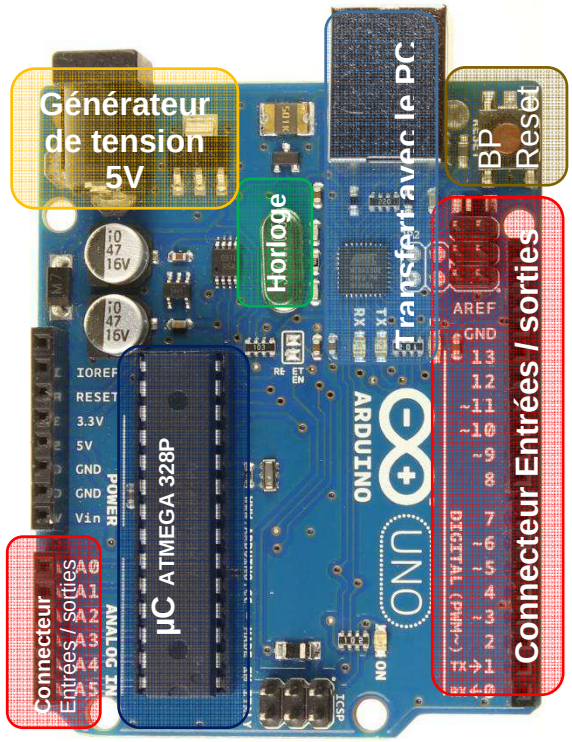


3 – CARTE ARDUINO UNO

La carte UNO possède l'ensemble des composants nécessaires au fonctionnement du microcontrôleur (fig. 2) :

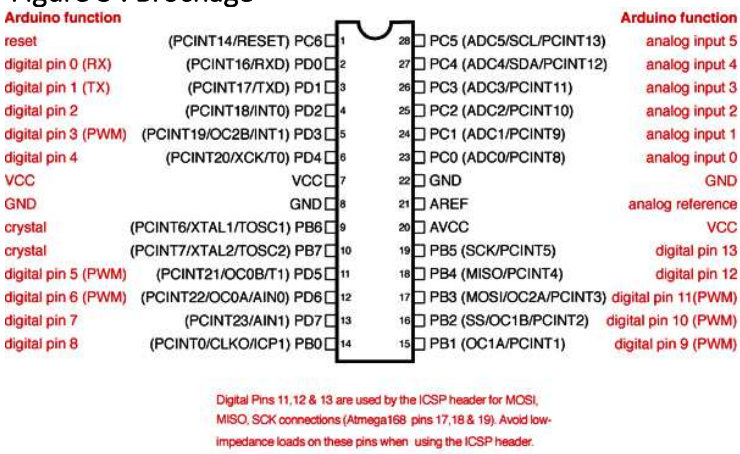
- générateur de tension stabilisée ;
- horloge ;
- connecteurs et composants nécessaires au transfert ;
- LED ;
- BP ;
- Connecteurs d'E/S ;
- …

Figure 2 : Carte UNO



Le brochage est le suivant (fig. 3) :

Figure 3 : Brochage



Les caractéristiques de la carte sont données dans le tableau ci-dessous :

Microcontrôleur	ATmega328P
Tension de fonctionnement	5V
Tension recommandée d'alimentation de la carte	7-12V
Tension limite d'alimentation	6-20V
Pattes d'entrée sortie	14 (dont 6 en sortie MLI)
Entrées analogiques	6
Courant de sortie max. par broche de sortie	20 mA
Courant de sortie max. sous tension de 3,3V	50 mA
Mémoire Flash	32 kO dont 0,5 kO utilisés pour transfert PC
SRAM / EEPROM	2 kO / 1 kO
Vitesse d'horloge	16 MHz
Dimensions	68.6 mm x 53.4 mm
Masse	25 g

4 – CHOIX D'UN MICROCONTROLEUR

La méthode ci-dessous n'est peut-être pas exhaustive, mais permet de faciliter le choix.

Il faut se poser quelques questions :

1/ Combien d'entrées doit posséder le µC et de quel type (TOR / Num / Ana) ?

2/ Combien de sortie doit posséder le µC et de quel type (TOR / Num / PWM) ?

3/ Comment alimenter le µC ?

4/ Comment adapter les niveaux de tension et de courant entre le µC et ses interfaces ?

Chapitre 15

GENIE INFORMATIQUE ET AUTOMATISME

Algorithmie

8

1 - PREAMBULE

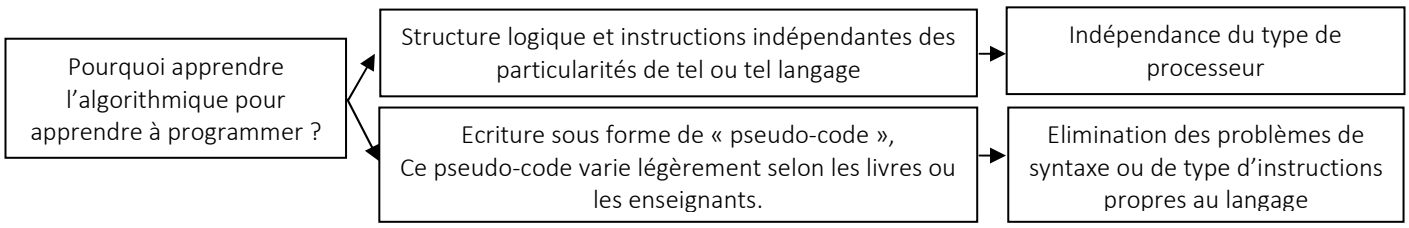
Le microcontrôleur n'est pas capable de faire autre chose que de suivre des instructions les unes derrière les autres (un peu à l'image d'une recette de cuisine ou d'un mode d'emploi d'un appareil).

Le meilleur moyen pour décrire son comportement est de travailler sous forme d'algorithmes.

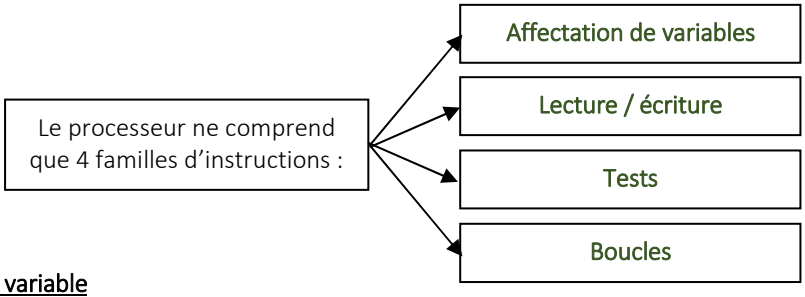
La maîtrise de l'algorithmique requiert deux qualités, très complémentaires :

- Il faut avoir une certaine intuition, car aucune recette ne permet de savoir a priori quelles instructions permettront d'obtenir le résultat voulu. Ce qu'on appelle l'intuition n'est finalement que de l'expérience tellement répétée que le raisonnement, au départ laborieux, finit par devenir « spontané » ;
- Il faut être méthodique et rigoureux. En effet, chaque fois qu'on écrit une série d'instructions qu'on croit justes, il faut systématiquement se mettre mentalement à la place de la machine qui va les exécuter, armé d'un papier et d'un crayon, afin de vérifier si le résultat obtenu est bien celui que l'on voulait. Cette opération ne requiert pas la moindre once d'intelligence. Mais elle reste néanmoins indispensable, si l'on ne veut pas écrire à l'aveuglette.

2 - ALGORITHME OU PROGRAMME ?



3 - QUATRE FAMILLES D'INSTRUCTIONS, ET C'EST TOUT !



Les différents types de variable

Une variable est une zone mémoire réservée pour stocker le résultat d'une opération

Type		Plage	Espace utilisé en mémoire
Booléen	Boolean (TOR)	VRAI (TRUE – « 1 ») ou FAUX (FALSE – « 0 »)	1 bit
Numérique	Byte (octet)	0 à 255 ou -128 à +127	1 octet
	Entier simple	-32 768 à 32 767	2 octets
	Entier long	-2 147 483 648 à 2 147 483 647	4 octets
	Réel simple	$-3{,}40\times10^{38}$ à $-1{,}40\times10^{-45}$ pour les valeurs négatives $1{,}40\times10^{-45}$ à $3{,}40\times10^{38}$ pour les valeurs positives	4 octets
	Réel double	$-1{,}79\times10^{308}$ à $-4{,}94\times10^{-324}$ pour les valeurs négatives $4{,}94\times10^{-324}$ à $1{,}79\times10^{308}$ pour les valeurs positives	8 octets
Alphanumérique	Caractère	Tout type de caractère (lettres, signes de ponctuation, espaces, chiffres), code ASCII	1 octet
	Chaine	Ensemble de caractères les uns derrière les autres En pseudo-code, une chaîne de caractères est toujours notée entre guillemets	fonction du nombre de caractères contenus dans la chaine
« Assemblage »	On peut assembler des types de variable de base pour se créer son propre type (date, …)		

L'affectation

La seule chose qu'on puisse faire avec une variable, c'est **l'affecter,** c'est-à-dire **lui attribuer une valeur**.

En pseudo-code, l'instruction d'affectation se note avec le signe ←

Ainsi :	Variable Toto	Variable Tutu
Toto ← 24	24	xx
Tutu ← Toto + 4	24	28

Une instruction d'affectation ne modifie que ce qui est situé à gauche de la flèche.

Opérateurs possibles dans les expressions :

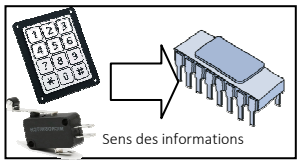
Variable	Opérateurs
Booléen	ET, OU, NON, XOR
Numériques	+, -, /, *, (,),^
Chaine	& (*concaténer*)

La lecture / l'écriture.

Le processeur peut avoir besoin de communiquer avec l'extérieur

LECTURE

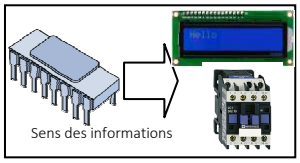
Le processeur effectue une lecture si on y rentre des informations (au clavier ou avec un capteur, …) pour qu'elles soient utilisées par le programme. *Titi ← LIRECLAVIER*



ECRITURE

Le processeur effectue une écriture si celui-ci envoie une information (sur un écran, un préactionneur, …)

ECRIREECRAN (Toto)



Les tests.

Il n'y a que deux formes possibles pour un test ; la première est la plus simple, la seconde la plus complexe.

```
SI <condition> ALORS
    action_alors
FINSI
```

```
SI <condition> ALORS
    action_alors
SINON
    Action_sinon
FINSI
```

```
SELONQUE
<condition_1> FAIRE action_1
…
<condition_n> FAIRE action_n
FINSELONQUE
```

Une **condition** est une **expression** dont la valeur est VRAIE ou FAUSSE. Cela peut donc être :

- une **variable** (ou une expression) de type booléen ;
- une **comparaison** réalisée à partir des opérateurs : =, ≠, >, <, ≥, ≤).

Les boucles.

On les appelle aussi structures itératives ou structures répétitives.

Boucle Tant que

```
TantQue <condition>
    actions
FinTantQue
```

Boucle Répéter jusqu'à

```
Répeter
    actions
Jusqu'à <condition>
```

Boucle Pour - Compter en bouclant

```
Pour compteur ← val_inf à val_sup Pas val_pas
    actions_à_répéter
Compteur suivant
```

4 - METHODE DE TRAVAIL

Pour faciliter le travail d'écriture et de débogage il faut,

1- Décomposer le problème en sous problèmes ;
2- Répertorier les variables et leur type utiles pour chaque sous problème ;
3- Créer un algorithme pour chaque sous problème (mettre des commentaires dans les lignes de code + prendre des noms de variables logiques pour tout le monde) ;
4- Tester le fonctionnement de chaque algorithme de sous problème ;
5- Créer un algorithme général gérant les sous algorithmes ;
6- Tester le fonctionnement général ;
7- Traduire chaque sous problème dans le langage de programmation désiré ;
8- Tester chaque sous-programme ;
9- Traduire l'algorithme général dans le langage de programmation ;
10- Tester le fonctionnement général.

GENIE INFORMATIQUE ET AUTOMATISME

Algorigrammes

9

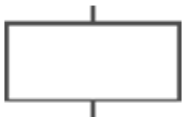
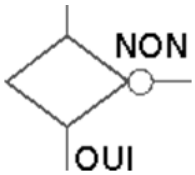
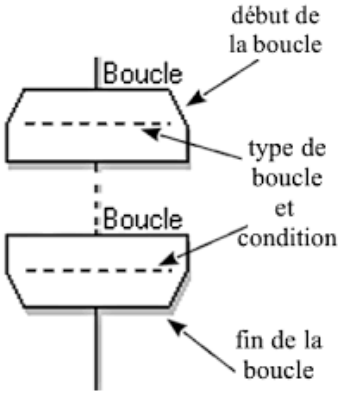
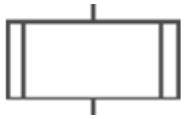
1 - PREAMBULE

Les algorigrammes ne sont que des moyens de représenter les algorithmes sous forme graphique. Ils sont utilisés pour des algorithmes simples.

Pour se remémorer ce qu'il faut connaitre sur les algorithmes, se référer à **la fiche n°8 sur l'algorithmie**.

2 - LES SYMBOLES GRAPHIQUES

Les symboles graphiques sont des symboles normalisés (ISO 5807). En voici une partie.

	Marque le début et la fin d'un bloc.
	Description d'une opération d'entrée ou de sortie (externe).
	Description d'une action d'affectation (interne).
	Exécution d'un test, selon le résultat, le programme se poursuit vers une branche ou vers l'autre.
	Réalisation d'une boucle : - Tant que … - Répéter, jusqu'à …
	Appel d'une macro (sous-programme)
	Commentaire (pour faciliter la lecture)

3 - LES STRUCTURES DE BASE

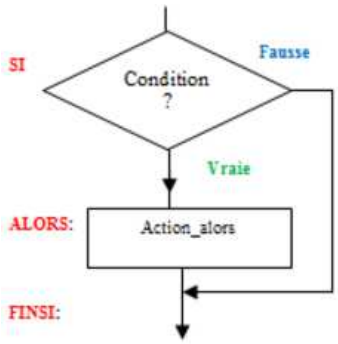
Les tests :

SI ALORS

SI <condition> ALORS

action_alors

FINSI



SI ALORS SINON

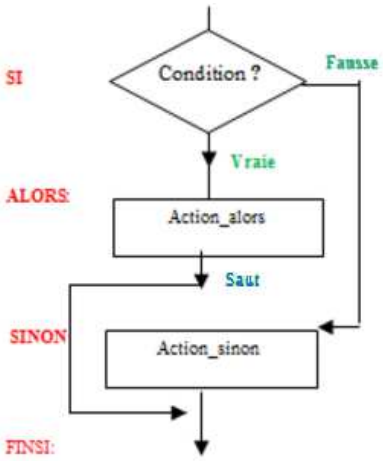
SI <condition> ALORS

action_alors

SINON

Action_sinon

FINSI



SELON QUE

SELONQUE est un raccourci d'écriture avec des SI imbriqués

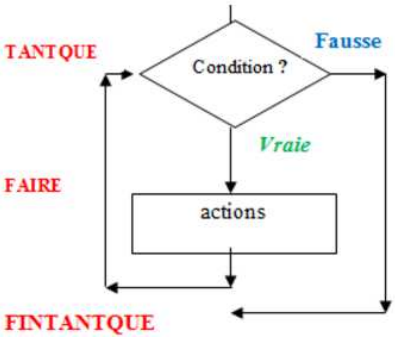
Les boucles :

BOUCLE TANT QUE

TantQue <condition>

actions

FinTantQue

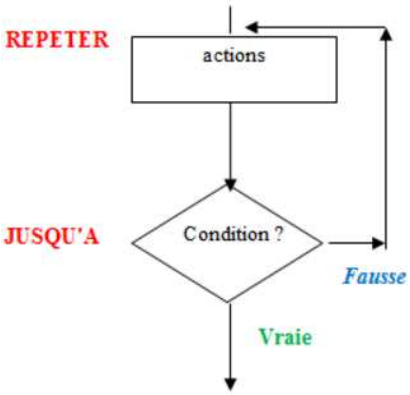


BOUCLE REPETER JUSQU'A

Répéter

actions

Jusqu'à <condition>

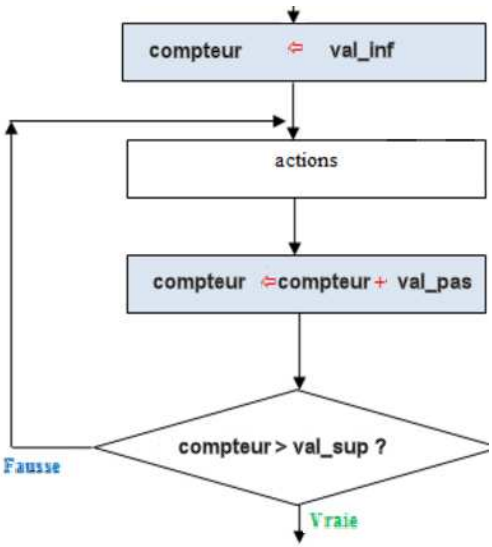


BOUCLE POUR - COMPTER EN BOUCLANT

***Pour** compteur ← val_inf **à** val_sup **Pas** val_pas*

actions_à_répéter

Compteur suivant



GENIE INFORMATIQUE ET AUTOMATISME

Transmissions de données

10

1- PREAMBULE

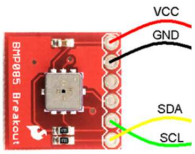
Les équipements électroniques sont, de nos jours, de plus en plus équipés de microprocesseurs. On parle d'équipements informatiques.

Leur fonctionnement est basé sur le traitement de nombres (numériques) qui signifient quelque chose pour le système. Des exemples sont traités figure 1.

Figure 1 : Différents matériels numériques

Variateur de vitesse Altivar

La valeur $100_{(10)}$ stockée dans l'adresse mémoire 250 du variateur obligera le variateur à envoyer des courants de fréquence 10,0 Hz au moteur asynchrone lorsque le variateur sera réglé en grande vitesse.

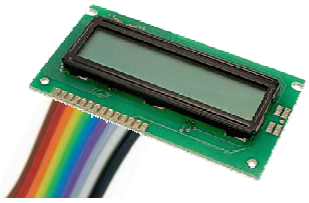


Capteur barométrique I2C

La pression atmosphérique mesurée est renvoyée par le capteur sous forme de 2 octets successifs transmis à l'aide des fils SDA et SCL

Souris PS2

3 octets d'informations sont transmis sur un mode série. Un clic gauche provoque la mise à 1 d'un des bits du premier octet.



Afficheur LCD alphanumérique parallèle

Sous certaines conditions, l'envoi d'un octet contenant la valeur $65_{(10)}$ sur le port parallèle de l'afficheur fera afficher un caractère « A » (suit le code ASCII).

2- TRANSMISSION SERIE / TRANSMISSION PARALLELE

Pour transmettre une information binaire (en bande de base, c'est-à-dire sans modulation), il n'y a que 2 solutions qui vont conditionner l'architecture de la transmission :

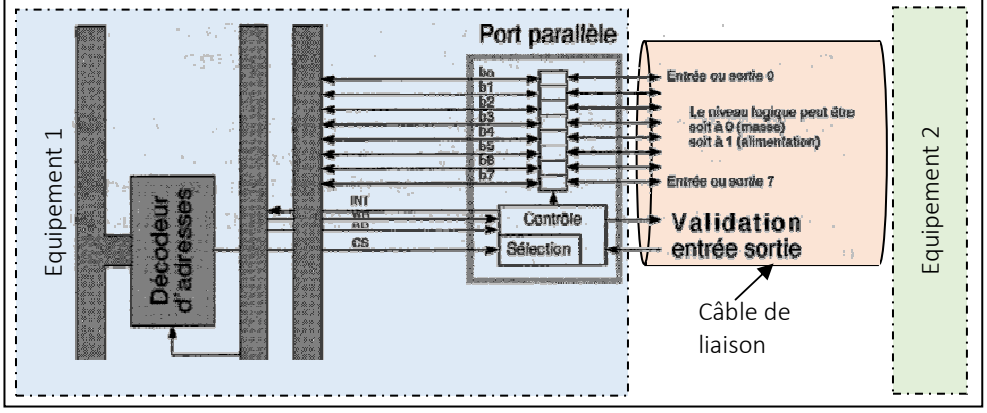
- la transmission des bits en parallèle ;
- la transmission des bits en série.

Transmission parallèle (fig. 2) :

Les données à transmettre sont **envoyées par paquets en même temps sur plusieurs fils en parallèle** (bus parallèle), souvent 8 fils de données (octet) accompagnés de fils de contrôle supplémentaires pour gérer le flux de la transmission (vitesse et sens).

Ce mode de transmission est utilisé dans les cartes mères des ordinateurs et dans les microcontrôleurs.

Figure 2 : Transmission parallèle entre 2 équipements



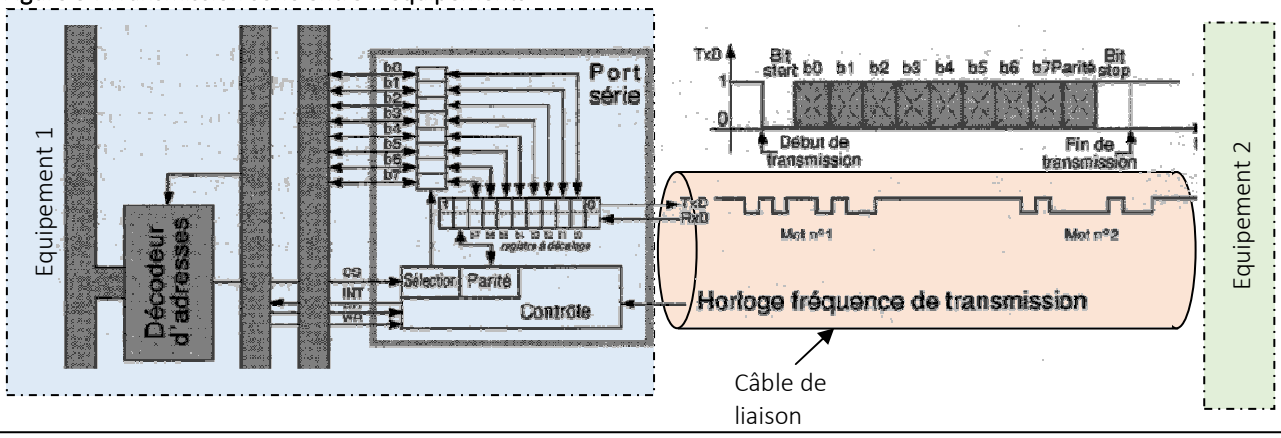
Les caractéristiques de cette transmission sont :

- une **grande vitesse de transmission** (8 bits transmis en même temps) ;
- un **nombre de fils important** (au moins 8 pour transmettre un octet).
- une **longueur des fils de liaisons courte** (un risque d'interférence d'un premier signal porté sur un fil avec un second (diaphonie) causé par des phénomènes d'induction électromagnétique).

Transmission série (figure 3) :

Les données à transmettre sont séparées bits par bits et **les bits sont envoyés les un après les autres sur un seul fil**. Ces données sont encadrées par des bits supplémentaires transmis pour contrôler le flux et la non corruption de la transmission. Le cadencement est généré par une horloge (qui peut être transmise ou non sur un fil supplémentaire). La vitesse de transmission dépend de cette horloge.

Figure 3 : Transmission série entre 2 équipements



Ce mode de transmission est utilisé pour la mise en réseau des matériels informatique (RS232, I2C, Ethernet, …).

Les caractéristiques de cette transmission sont :

- une **grande longueur des fils de liaisons** (les risques de diaphonie sont limité car le nombre de fils en parallèle est limité).
- un **nombre de fils de transmission limité** (2 fils minimum)
- une **« relative » lenteur de transmission** (bits transmis les uns à la suite) ;

Nota : Afin de minimiser la diaphonie, on utilise souvent des paires torsadées dans les câbles servant aux transmissions de données dans les réseaux.

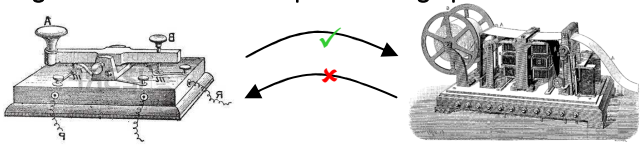
3- TRANSMISSION SIMPLEX / HALF-DUPLEX / FULL DUPLEX

Transmission simplex (figure 4) :

Un canal de communication simplex ne permet la transmission de l'information que dans **une direction seulement**.

La borne d'émission d'un des équipements (TX) est reliée à la borne de réception de l'autre (RX).

Figure 4 : transmission simplex du télégraphe

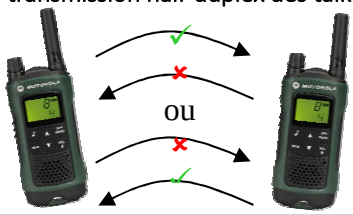


Transmission half-duplex (figure 5) :

Le mode de communication half-duplex est réalisé lorsqu'un canal de communication unique (simplex) permet aux informations de circuler alternativement dans des **directions opposées**.

La même borne d'un équipement sert à l'émission et la réception (RX/TX).

Figure 5 : transmission half-duplex des talkies-walkies

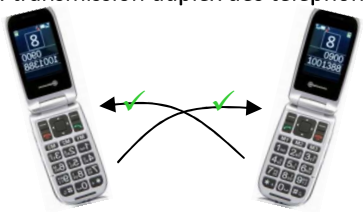


Transmission full-duplex (figure 6) :

On parle de full-duplex si la transmission de l'information peut se faire **simultanément dans les 2 sens**.

Lorsque la transmission est câblée, on relie la borne d'émission (TX) d'un équipement à la borne de réception (RX) de l'autre équipement par un câble croisé.

Figure 6 : transmission duplex des téléphones



GENIE INFORMATIQUE ET AUTOMATISME

Liaison série

11

1 - PREAMBULE

Pour établir une liaisons série entre 2 équipements, il faut définir :

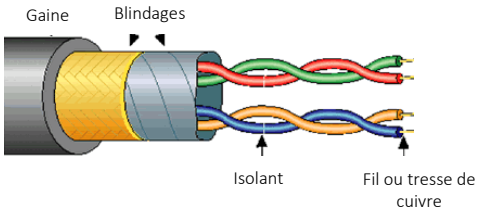
- Les points d'accès aux équipements (connectique et média utilisés) ;
- Le type de transmission (asynchrone / synchrone) ;
- Les niveaux de tension admis acceptables pour reconnaitre les bits transmis.

Ces points correspondent à ce qui est fait dans la couche 1 du modèle OSI (fiche de génie informatique 9).

2 - LES POINTS D'ACCES AUX EQUIPEMENTS (CONNECTIQUE ET MEDIA UTILISES)

Pour transmettre des informations d'un équipement à un autre, il faut définir le support de transmission, ce que l'on appelle le Média.
Il peut être de différents types :

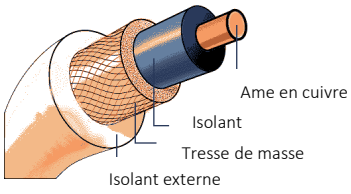
- ***La paire torsadée***



On utilise 2 à 4 paires de fils *de cuivre enroulées* (fil de téléphone).

Chaque extrémité du câble est branchée à une fiche *(RJ45, Sub D9, SUB D25, …) selon le standard de transmission utilisé*.

- **Le câble *coaxial***

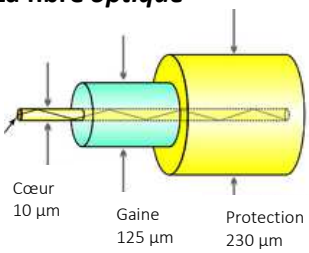


Il est constitué d'un *fil de cuivre rigide au centre et d'une tresse de cuivre qui le recouvre*.

Chaque extrémité est reliée à une fiche *BNC*.

Les liaisons sur chaque poste se font à l'aide d'un *T*.

- **La fibre *optique***



Elle permet de transmettre sous forme de *signaux lumineux des informations à grand débit*.

Elle est *immunisée aux perturbations électromagnétiques*. On l'utilise également dans les cas de *confidentialité car elle est plus difficile à écouter*.

Par contre, son utilisation reste sensible à la qualité des terminaisons (*problèmes de réfraction*).

- **La liaison *sans fil***

Elle peut être soit *infrarouge (Irda)*, soit *hertzienne (WiFi ou Bluetooth)*, soit *satellitaire.* Elle permet des transmissions à *haut débit peu sécurisées*.

3 - LE TYPE DE TRANSMISSION (ASYNCHRONE / SYNCHRONE)

La transmission série nécessite une synchronisation entre l'émetteur et le récepteur. Cette synchronisation peut être réalisée par un fil d'horloge entre l'émetteur et le récepteur ou sans fil d'horloge (synchronisation en envoyant dans la donnée une information d'horloge).

*Dans le cas d'une transmission **synchrone**, l'envoi de plusieurs données successives est cadencé par l'horloge et la durée entre donnée est identique.*

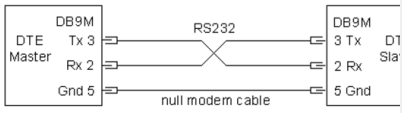
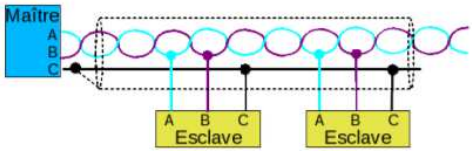
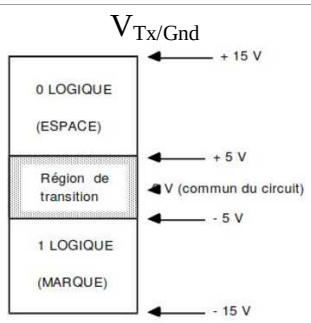
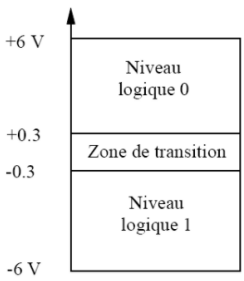
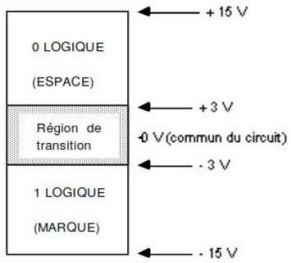
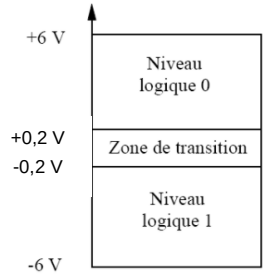
*Pour une transmission **asynchrone**, le rythme d'envoi de plusieurs données successives peut être aléatoire.*

4- LES NIVEAUX DE TENSION AUTORISES DANS LES CABLES (POUR LES LIAISONS FILAIRES)

Pour permettre à l'équipement récepteur de comprendre parfaitement l'information que lui envoie l'émetteur, il faut que l'émetteur mette des niveaux de tension acceptables (suffisants pour que le récepteur comprenne, pas trop élevés pour que les équipements ne grillent pas).

5- COMPARAISON ENTRE 2 STANDARDS (RS232 / RS485)

Il s'agit de standards qui définissent les points cités précédemment.

SPECIFICATIONS	RS232 – V28 (utilisé sur les vieux PC)	RS485 (utilisé dans les réseaux locaux industriels)
Sens des échanges	Full duplex	Half ou full duplex
Mode de fonctionnement	Asymétrique référencé	Différentiel
Immunité aux perturbations	Faible due au mode asymétrique (1 seul fil de transmission => parasite capté et transmis)	Forte grâce au mode différentiel (le fil A et le B voient le même parasite, comme on soustrait VA à VB => le parasite disparait).
Nombre maximal d'émetteur/récepteur	1 émetteur 1 récepteur	32 émetteurs 32 récepteurs
Topologie de liaisons	Liaisons point à point	Bus multipoints
Câblages typiques		
Longueur du câble maximum (m)	15	1200
Débit maximum (bit/s) (peut être assimilé a des Bauds lorsque l'on ne module pas)	19,2k pour 15m	10M pour 100m 100k pour 1200m
Niveaux logiques de sortie		
Niveaux logique d'entrée		

Impédance d'entrée d'un récepteur (Ω)	3k to 7k	>=12k

GENIE INFORMATIQUE ET AUTOMATISME

Liaison série asynchrone

12

1 - PREAMBULE

La transmission série asynchrone définit des règles de transmission des données pour gérer le bon fonctionnement.

Ce point correspond à la définition de la couche 2 du modèle OSI.

2 - PARAMETRES DU PROTOCOLE DE TRANSMISSION (FIGURE 1)

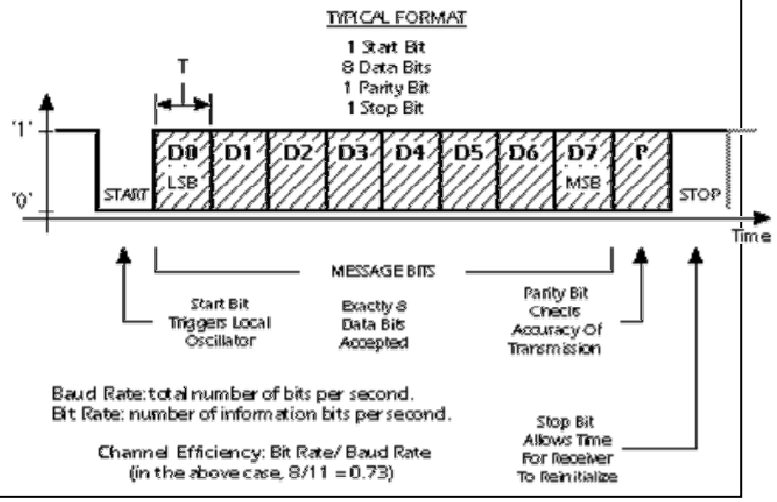
- La vitesse de transmission est définie par le **débit** de la ligne (bit.s^{-1}). Lorsque l'on fait une transmission sans modulation, le signal sur la ligne ne peut prendre que 2 valeurs (« 0 » ou « 1 »). Dans ce cas, on assimile les **bit.s^{-1}** à des **Baud**. Cette vitesse définit la durée T entre 2 bits transmis.

Lorsqu'aucun caractère n'est transmis, le niveau logique de la ligne de transmission est à « 1 ». Ceci permet de dissocier le bruit d'un changement d'état.

Pour transmettre des données sur le Média, il est nécessaire d'envelopper les données à transmettre de bits supplémentaires qui permettent de gérer le flux et « contrôler » l'intégrité de la donnée transmise :

- 1 **bit de START** qui passe la ligne au « 0 » permet de réveiller les équipements endormis (puisque les données peuvent être transmises à un rythme aléatoire) ;
- Les bits utiles sont ensuite transmis (donnée). Le nombre de bits de donnée peut être de 5, 6, 7 ou 8 bits (on envoie le LSB en premier) ;
- On intègre souvent à la transmission de la donnée un bit de vérification de l'intégrité de la transmission de la donnée (**bit de parité**[1] : paire, impaire, ou sans) ;
- On ajoute un ou plusieurs **bit(s) de STOP** qui passe(nt) la ligne au « 1 » pour indiquer au récepteur la fin de la transmission de la donnée.

Figure 1 : transmission série asynchrone (train de bits)



[1] **Parité paire** : bit de parité mis à 1 ou 0 pour assurer un nombre total de bits à 1 (bits de donnée + bit de parité) pair (exemple : donnée à transmettre 1001 0111 : il y a 5 bits à 1 dans la donnée. 5 c'est impair donc on met le bit de parité à 1 pour avoir un nombre pair de bits transmis et on transmettra 1 1001 0111).
Parité impaire : bit de parité mis à 1 ou 0 pour assurer un nombre total de bits à 1 impair (exemple donnée à transmettre 1001 0111 : il y a 5 bits à 1 dans la donnée. 5 c'est impair donc on met le bit de parité à 0 pour avoir un nombre impair de bits transmis et on transmettra 0 1001 0111).

3 - RELATION ENTRE BITS TRANSMIS ET SIGNAL ELECTRIQUE SUR LA LIGNE (FIGURE 2 ET 3)

Si on analyse le signal sur une ligne de transmission RS 232 (voir fiche Génie informatique 6), le niveau de tension est inversé par rapport au niveau logique.

La figure 2 montre les bits (0 / 1) à envoyer sur la ligne pour transmettre la lettre @ codée en ASCII 7 bits. Les paramètres de la transmission sont : un bit de START, 1 bit de parité paire et 2 bits de STOP.

La figure 3 montre le signal émis sur la ligne de transmission.

« @ » en code ASCII 100 0000

Figure 2 : Train de bits à transmettre pour transmettre la lettre @ codée en ASCII

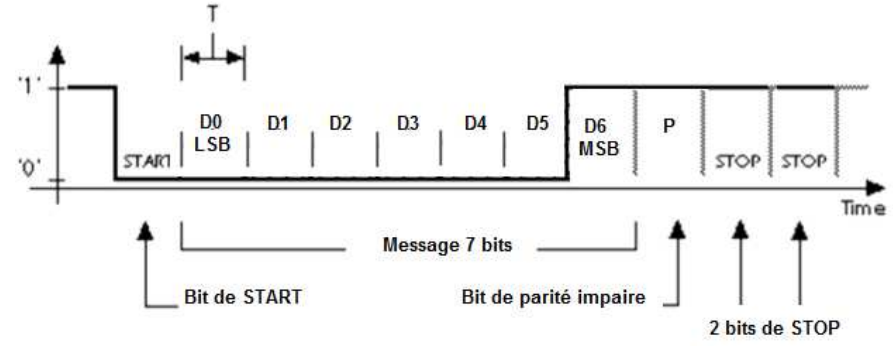
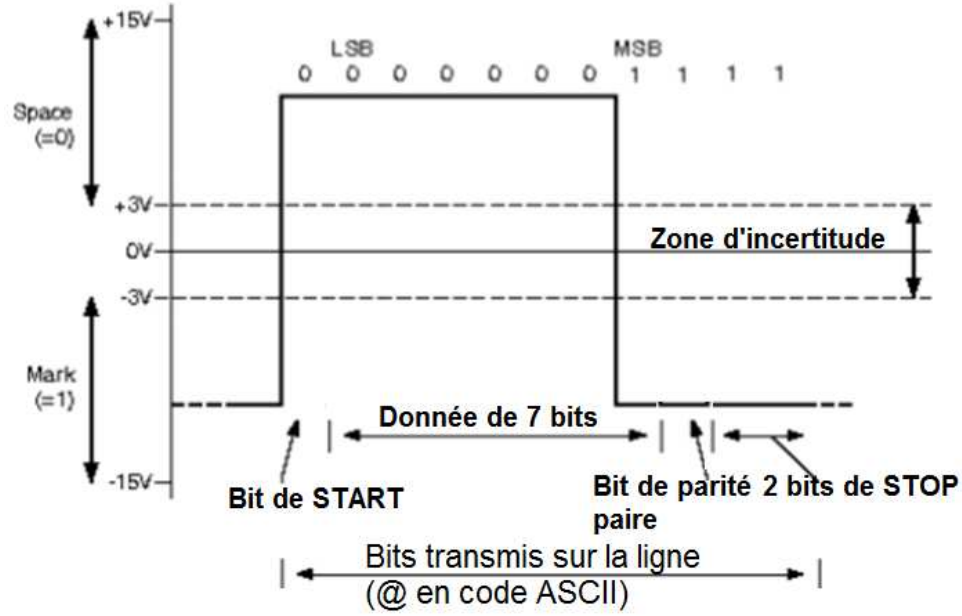


Figure 3 : exemple de signal sur la ligne de transmission lors de la transmission de la lettre A codée en code ASCII



4 - EFFICACITE DE LA TRANSMISSION

L'**efficacité** d'une telle transmission peut être déterminée : $E = \dfrac{\text{Nombre de bits de donnée}}{\text{Nombre de bits réellement transmis}}$

Dans le cas ci-dessus $E = \dfrac{7}{11} = 63,7\%$. Cela signifie que l'on prend 36,3% du temps pour transmettre autre chose que l'information proprement dite.

GENIE INFORMATIQUE

Protocole de communication MODBUS

13

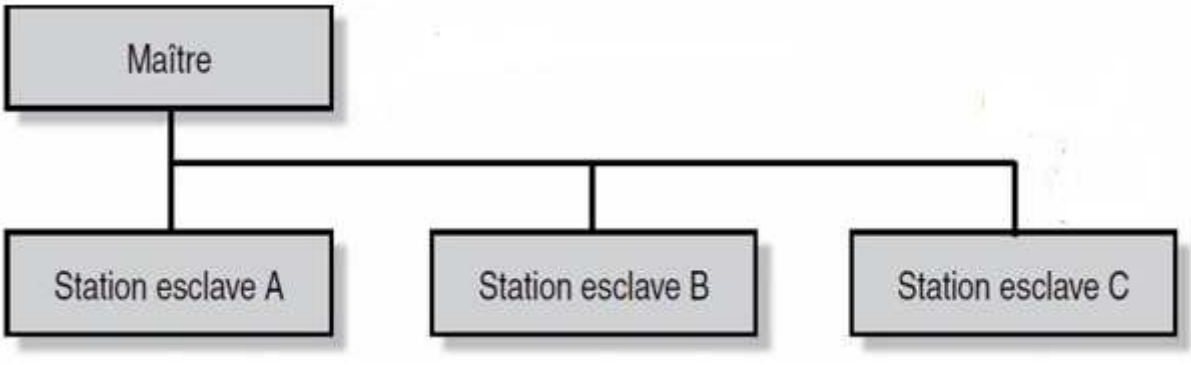
1 – PREAMBULE

Bus de terrain MODBUS :

Dans les réseaux informatiques et les télécommunications, un protocole de communication est un ensemble de règles qui régissent un type de communication particulier.

Le protocole MODBUS est un protocole de communication (de dialogue) basé sur une structure hiérarchisée entre un maître et plusieurs esclaves (stations). Il permet au maître de lire et d'écrire certaines valeurs dans les stations esclaves.

Figure 1 : Bus de terrain MODBUS



Le maître peut questionner l'ensemble des esclaves sur son bus (broadcast), ou interroger un esclave et attendre sa réponse.

Sur un réseau formé par une liaison RS 485, on ne peut avoir qu'un seul maître et jusqu'à 31 esclaves différents.

Définitions :

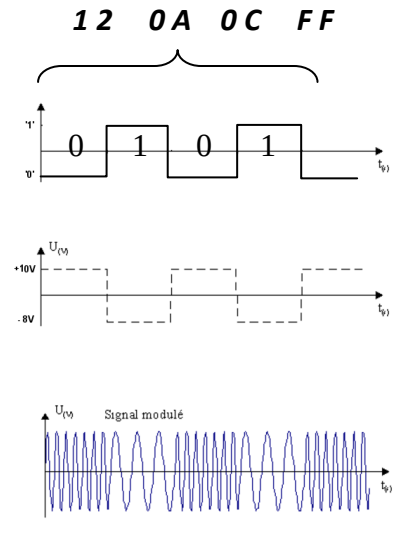
Exemples

Trame : Ensemble des informations transmises sous forme hexadécimale (ne tient pas compte des modes de transmission (série, parallèle, bit de start, …).

Train de bits : Ensemble des '0' et '1' successifs envoyés dans une liaison série (bits de start, parité, stop, compris).

Signal : grandeur portée par le média pour transmettre les 0 ou 1 (souvent, pour du filaire, il s'agit d'une tension) :

- le signal peut être en bande de base, cela signifie que le signal transporte les 0 et 1 sous forme de niveau de tension
- Le signal peut être modulé, en fréquence par exemple, cela signifie que le signal est sinusoïdal et la fréquence varie en fonction du bit à transmettre. Dans cette situation, on peut transmettre plusieurs bits en même temps.



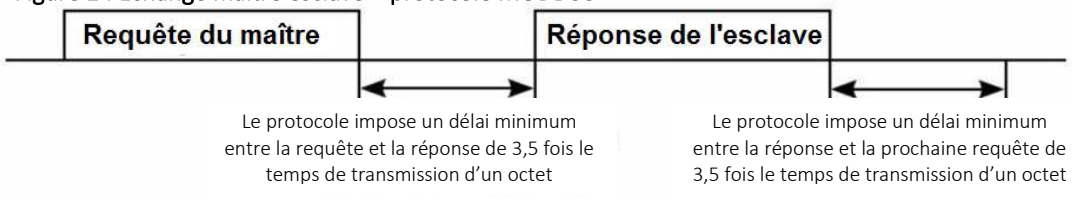
2 – TRAME D'ECHANGE

Echange maître esclave :

Chaque intervenant sur le réseau possède un identifiant (adresse). Cette adresse est formée de 8 bits et doit être unique sur le réseau.

Le maître envoie une requête à un esclave et attend une réponse de celui-ci. L'échange doit suivre le format ci-dessous :

Figure 2 : Echange maître esclave – protocole MODBUS



Requête du maitre à l'esclave :

La requête peut être de différents types. Selon le besoin, on peut écrire une donnée (plus ou moins complexe) dans une zone mémoire de l'esclave ou lire une donnée.

En plus d'envoyer l'adresse de l'esclave auquel s'adresse le maître, la requête contient la fonction que l'on souhaite réaliser (voir l'annexe 1). Cette fonction sera codée sur 8 bits.

La donnée à transmettre est toujours un multiple de 8 bits.

A la fin de la transmission, le maître ajoute un mot de contrôle qui permet de vérifier l'intégrité du message transmis (Code de Redondance Cyclique). Ce mot de contrôle est formé sur 2 octets.

La trame ainsi formée (couche 2 du modèle OSI) aura l'allure suivante (les 3 dernières lignes sont un exemple d'échange avec l'Altivar de la barrière DECmapark) :

	Adresse de l'esclave		Code fonction de la requête		Données (informations spécifiques concernant la demande)						Mot de contrôle (CRC16)			
	1 octet		1 octet		n octets[1]						2 octets			
Trame (16)	0	7	0	6	0	0	F	B	A	1	1	3	C	0
Trame (2)	0000	0111	0000	0110	0000	0000	1111	1011	1010	0001	0001	0011	1100	0000
Train de bits transmis	0 11100001		0 01100001		0 00000001		0 11011111 1		0 10000101 1		0 11001000 1		0 00000011 1	

START LSB adresse MSB STOP

Paramètres de transmission avec l'altivar de la barrière DEC
1 bit START, 8 bits de données, Pas de contrôle de parité, 1 bit de stop
Débit : 19 200 bit.s^{-1}

Une transmission suivant le protocole MODBUS RTU utilise la transmission de données en série asynchrone (voir fiche Génie Informatique 7) half-duplex (voir fiche Génie Informatique 5).

A chacun des octets présentés dans l'exemple, on ajoutera un bit de START, (éventuellement un bit de parité), et un bit de STOP comme le montre la dernière ligne du tableau ci-dessus.

Réponse de l'esclave :

La réponse sera constituée de l'adresse de l'esclave qui répond, du code fonction demandée, de la réponse à la fonction puis du CRC16.

Adresse de l'esclave	Code fonction de la requête	Données (informations spécifiques concernant la demande	Mot de contrôle (CRC16)
1 octet	1 octet	m octets[2]	2 octets

[1] n dépend de la fonction réalisée
[2] m dépend de la fonction réalisée

GENIE INFORMATIQUE ET AUTOMATISME

Modèle OSI

14

1 - PREAMBULE

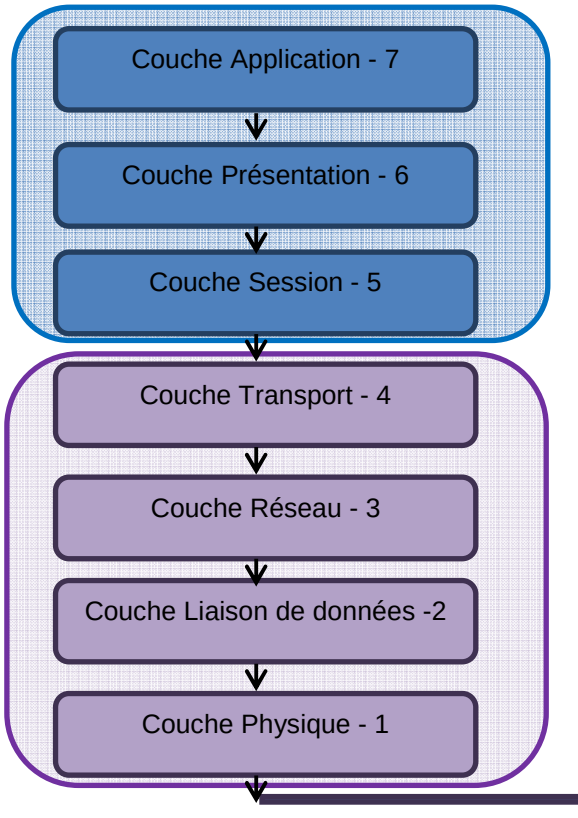
« Les constructeurs informatiques ont proposé des architectures réseaux propres à leurs équipements. Par exemple, IBM a proposé SNA, DEC a proposé DNA... Ces architectures ont toutes le même défaut : du fait de leur caractère propriétaire, il n'est pas facile de les interconnecter, à moins d'un accord entre constructeurs. Aussi, pour éviter la multiplication des solutions d'interconnexion d'architectures hétérogènes, l'ISO (International Standards Organisation), organisme dépendant de l'ONU et composé de 140 organismes nationaux de normalisation, a développé un modèle de référence appelé modèle OSI (Open Systems Interconnection). Ce modèle décrit les concepts utilisés et la démarche suivie pour normaliser l'interconnexion de systèmes ouverts (un réseau est composé de systèmes ouverts lorsque la modification, l'adjonction ou la suppression d'un de ces systèmes ne modifie pas le comportement global du réseau).

L'objectif de cette norme est de spécifier un cadre général pour la création de normes ultérieures cohérentes. Le modèle lui-même ne définit pas de service particulier et encore moins de protocole. ... »[1]

2 - LES 7 COUCHES DU MODELE

Le modèle OSI est un modèle en 7 couches successives dans lesquelles les informations à transmettre vont successivement passer et être complétées ou analysés par une machine souhaitant transmettre des informations à une autre machine.

Les 7 couches sont les suivantes :



- Fournit une interface avec les applications
- Fournit l'accès réseau aux applications

Ex. : navigateur web : transforme les données brutes pour les envoyer sur le réseau (http, ftp, smtp, ...)

- Assure la mise en forme des données avant transmission

Ex. : permet de s'affranchir du type de système et de machine pour permettre l'interopérabilité

- Initie et termine les sessions de transmission
- Assure la séparation des paquets

Ex. : permet de différentier les informations envoyées et reçues / application

- Définit comment les données sont transmises sur le réseau
- Fournit aux couches supérieures les outils de transport fiable

Ex. port (protocole TCP ou UDP)

- Fournit l'adressage logique (@IP)
- Détermine le bon chemin vers le destinataire
- Fournit les services d'adressage physique (@MAC) aux couches supérieures
- Assure le contrôle de validité des trames de données

- Fournit l'accès au média (câble, wifi, bluetooth, …)
  Média : support de transmission

Train binaire : trame (signal électrique, ...)

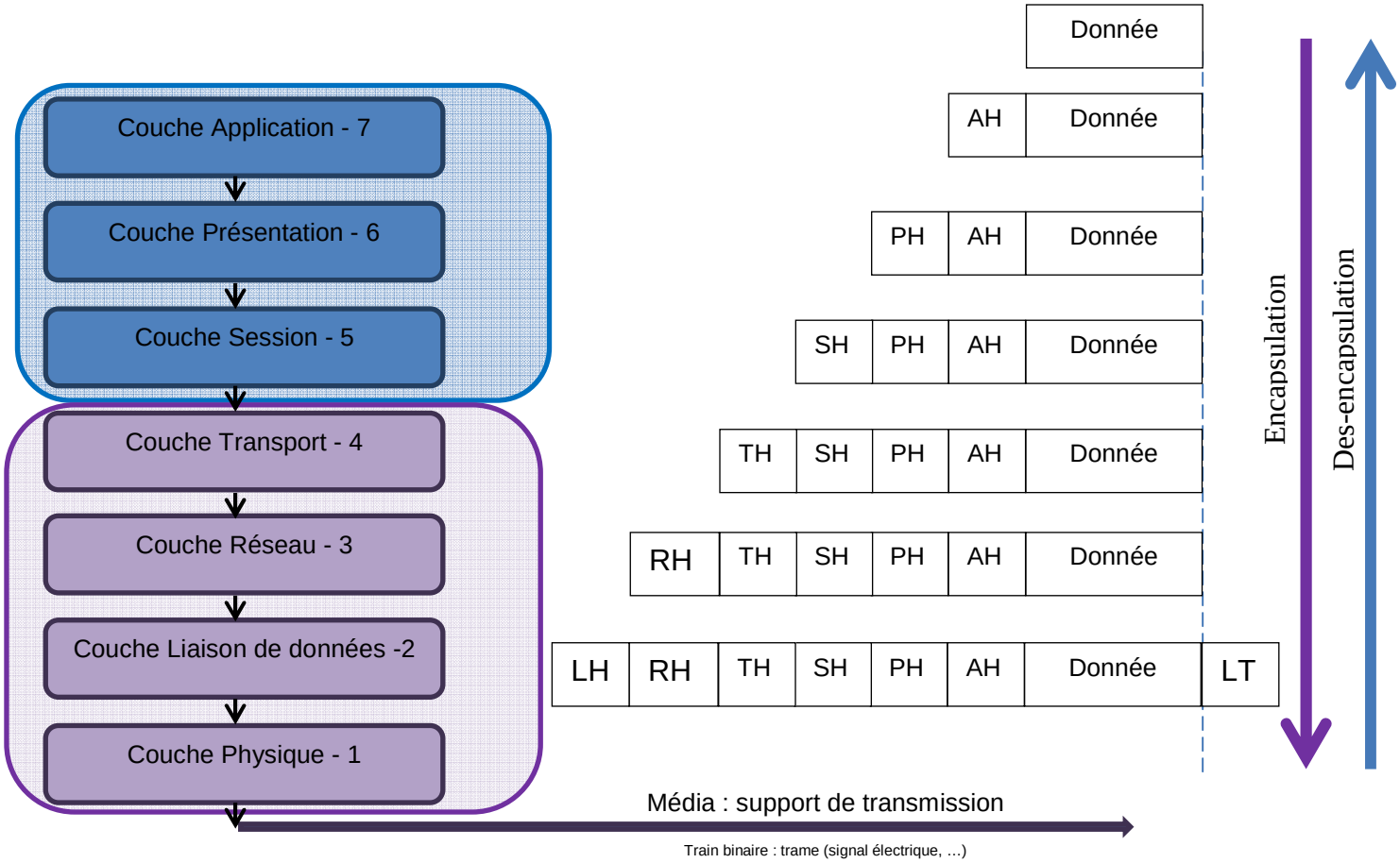
[1] Tiré de : http://www.frameip.com/osi/

3 - CREATION DE LA TRAME : PROCESSUS D'ENCAPSULATION/DES-ENCAPSULATION

Comme nous venons de le voir, le modèle OSI fonctionne en 7 couches. Chacune à une fonction spécifique. Les données effectivement transmises sur le réseau sont le résultat du traitement de la donnée initiale par chaque couche.

Lorsque l'on met une donnée sur le réseau, le fait de passer dans les 7 couches s'appelle l'encapsulation.

Lorsque l'on décrypte une donnée issue du réseau pour l'exploiter avec le logiciel, le fait de passer dans les 7 couches s'appelle le dés-encapsulation.



GENIE INFORMATIQUE ET AUTOMATISME

Réseaux informatiques

15

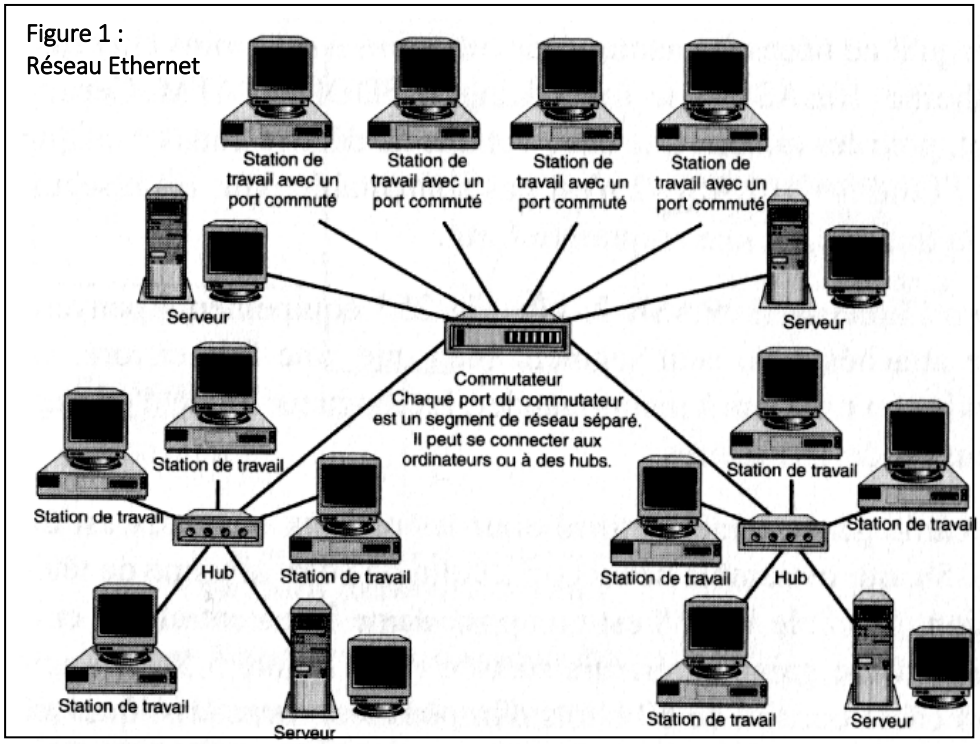
1 - PREAMBULE

A l'origine, les réseaux reliaient des terminaux passifs à de gros ordinateurs centraux.

A l'heure actuelle, ils permettent de connecter tous types d'ordinateurs ou éléments informatiques. Ils sont utilisés partout afin de partager des informations (échange de courrier, échange de fichiers, …), mais aussi pour partager des ressources (imprimante, logiciel, …).

Les systèmes minimums, pour pouvoir communiquer, entre eux ont besoin de matériels spécifiques : le média - les cartes réseau, ...

Figure 1 :
Réseau Ethernet



2 - TAILLES DE RESEAU

Selon la taille du réseau, il portera une dénomination différente :

Bus de carte : liaison entre le microprocesseur et ses extensions (mémoire, ports d(E/S) ;

Réseau de terrain : liaison entre un microcontrôleur à des capteurs déportés ;

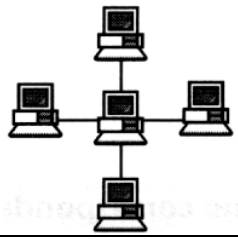
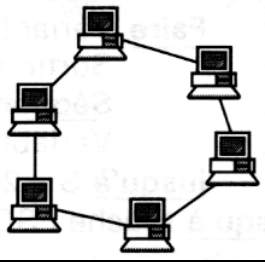
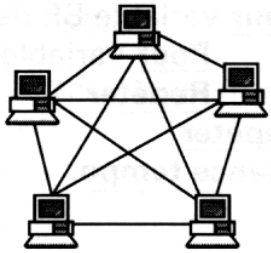
LAN (Local Area Network) : liaison entre ordinateurs situés dans une petite aire géographique (réseau privé (par exemple le lycée)).

MAN (Metropolitan Area Network) : interconnexion de plusieurs LAN géographiquement proches. Il utilise des commutateurs ou des routeurs interconnectés par des liens hauts débits. Il s'agit encore d'un réseau privé mais qui peut utiliser les interconnexions de communication publiques.

WAN (Wide Area Network) : interconnexion de plusieurs LAN à travers de grandes distances géographiques. Il fonctionne grâce à des routeurs qui choisissent le trajet pour acheminer les données. Internet est un WAN (les transmissions pouvant être terrestres ou spatiales).

3 - MODES DE LIAISONS ET TOPOLOGIE

Il existe 2 modes de liaison (figure 2) :

Le mode de diffusion	Le mode point à point
Principe : Tous les équipements partagent le même support de transport. Chaque message envoyé par un équipement connecté au réseau est reçu par tous les autres. C'est l'adressage spécifique qui est placé dans le message qui permet à chaque équipement de déterminer si le message le concerne ou pas.	**Principe :** Une liaison ne relie que 2 équipements entre eux. Quand 2 équipements non connectés entre eux veulent communiquer, ils le font par l'intermédiaire d'un/des autre(s) équipement(s). Dans le cas de l'étoile, le nœud central reçoit et envoie tous les messages.
Avantage : La défaillance d'un des équipements ne provoque pas la panne du réseau.	**Avantage :** Le fonctionnement est simple, et très fiable pour le maillage régulier.
Inconvénient : La rupture du support provoque l'arrêt du réseau.	**Inconvénients :** La panne du nœud central (étoile) ou d'un des équipements (boucle) paralyse tout le réseau. Le maillage régulier est couteux en câbles.
Bus 	Etoile 
Anneau 	Boucle 
Satellite 	Maillage 

GENIE INFORMATIQUE ET AUTOMATISME

Adresse MAC – Adresse IP

16

1 - PREAMBULE

Les cartes réseau sont chargées mettre les informations à échanger sur le média pour la connexion au réseau IP. On les trouve notamment dans : les PC – les cartes d'extensions pour microcontrôleur, les extensions pour automates, …

2 - CARTE RESEAU ETHERNET

Les cartes Réseau Ethernet (figure 1) utilisent un câble multipaire comme média. Ce câble possède 1 paire de fils pour l'émission depuis la carte (Tx+ et Tx-) et 1 paire de fils pour la réception sur la carte (Rx+ et Rx-) (un peu à l'image de la liaison série RS485). Selon la technologie Ethernet (10 base 2, 100 base 10, …), on ajoute ou pas encore 2 autres paires de fils.

Figure 1 : Cartes réseau

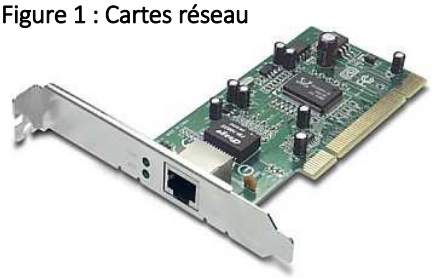
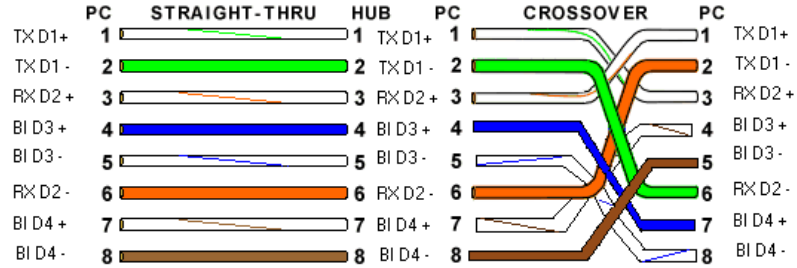


Figure 2 : Connectique des câbles réseau Ethernet



La figure 2 représente la connectique à appliquer selon le cas :

- 2 ordinateurs connectés directement => **câble *croisé*** (TX PC1 sur RX PC2 et inversement) ;
- un concentrateur (HUB ou SWITCH) est utilisé => **câble *droit*** (le concentrateur réalise le croisement).

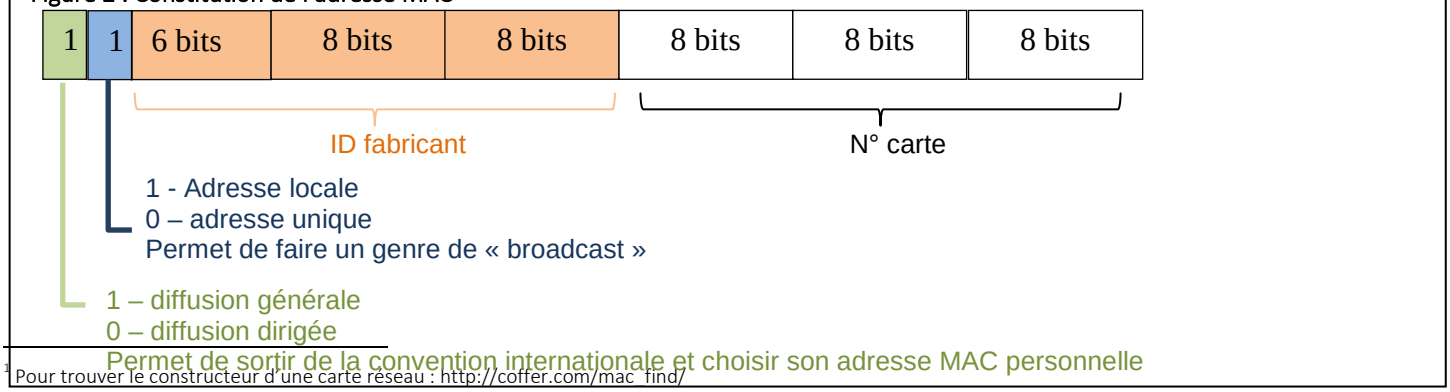
3 - ADRESSE MAC – ADRESSE PHYSIQUE

Afin de permettre l'échange d'informations entre 2 équipements distincts d'un même réseau, qui peut contenir d'autres équipements, il est nécessaire de nommer chaque carte réseau de façon unique (un peu à l'image du nom de famille et du prénom que vos parents vous ont donné). Cela permet de s'adresser de façon univoque à chaque équipement.

Le nom ainsi donné correspond à **l'adresse MAC (Medium Access Card)**.

Il s'agit d'un numéro donné à la carte par le constructeur de celle-ci, comportant **6 octets (48 bits)** ($18{,}4.10^{18}$ n° de cartes possibles) (exemple d'adresse MAC[1] : 00:11:43:00:FB:01).
Ce numéro suit une convention internationale (Organizationally Unique Identifier) gérée par l'IEEE qui est décrite à la figure 2 :

Figure 2 : Constitution de l'adresse MAC



[1] Pour trouver le constructeur d'une carte réseau : http://coffer.com/mac_find/

4 - ADRESSES IP – ADRESSE LOGIQUE

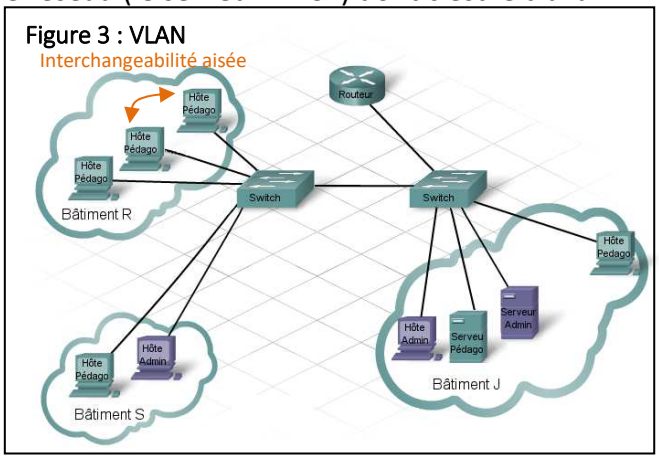
Les adresses logiques (**adresse IP**) sont des adresses distribuées :

- de façon permanente, en dur par le concepteur du réseau ;
- ou automatiquement, par une machine spéciale sur le réseau (le serveur DHCP) dont c'est le travail.

Les adresses IP permettent :

- une interchangeabilité aisée des machines sur le réseau ;
- une création de réseaux logiques différents sur le même réseau physique (exemple : le VLAN Pédago et le VLAN Admin du lycée).

Figure 3 : VLAN
Interchangeabilité aisée



Les adresses IP_{v4} sont des adresses formées de ***32 bits*** (***4 octets (écrits en décimal)***) soit ***4,295 milliards d'adresses possibles***. Ainsi l'adresse IP du serveur du rectorat est 185.75.143.25.

Actuellement le protocole des adresses est **IP_{v4}**, mais nous allons passer en IP_{v6} (soit des adresses sur 16 octets au lieu de 4 actuellement).

Afin de séparer les réseaux logiques sur le même réseau physique, on organise les adresses IP comme suit :

Les adresses de classes A, pour 128 réseaux permettent d'adresser 16777216 « équipements adressables ». Elles sont du type X. N. N. N où X peut être compris entre 1 et 126 et N compris entre 0 et 255.

	Division d'une adresse IP				
	Classe A	adressage du réseau	adressage des hôtes		
Adresse IP (2)	0	000 1100	1000 0000	0000 1111	0101 0110
Adresse IP (10)	12		128	15	86
Masque de sous réseau[2]	255		0	0	0

Les adresses de classes B, pour ***16384*** réseaux permettent d'adresser ***65534*** « équipements adressables ». Elles vont des adresses réseau 128.1.0.0 à 191.254.0.0.

	Division d'une adresse IP				
	Classe B	adressage du réseau		adressage des hôtes	
Adresse IP (2)	10	00 0010	1111 1110	0000 1111	0101 0110
Adresse IP (10)	130		254	27	106
Masque de sous réseau	255		255	0	0

Les adresses de classes C, pour ***2097152*** réseaux permettent d'adresser ***254*** « équipements adressables ». Elles vont des adresses réseau 192.0.1.0 à 223.255.254.0.

	Division d'une adresse IP				
	Classe C	adressage du réseau			adressage des hôtes
Adresse IP (2)	110	1 0010	1110 1000	0011 1010	0101 1100
Adresse IP (10)	210		232	58	92
Masque de sous réseau	255		255	255	0

Les classes D et E sont des classes particulières.

[2] Le masque de sous réseau permet en faisant un ET bit à bit avec l'adresse IP de retrouver le réseau auquel appartient la machine et son numéro

GENIE INFORMATIQUE ET AUTOMATISME

Asservissement et régulation - Paramètres

17

1- SCHEMA BLOC TEMPOREL

La figure 1 représente un système asservi sous forme de blocs. Se référer à la fiche 12 sur la structure pour comprendre la structure et son intérêt.

Pour la suite, nous considèrerons que le capteur à une fonction de transfert de 1 ce qui signifie que x(t) = s(t).

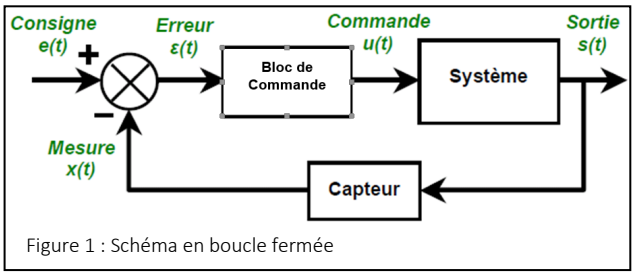


Figure 1 : Schéma en boucle fermée

2- PERFORMANCE DES SYSTEMES ASSERVIS

La performance d'un système asservi dépend de plusieurs critères dont voici les principaux :

1. La stabilité (figure 2)

Un système physique est stable s'il retourne spontanément vers son état d'équilibre lorsqu'il en est écarté.

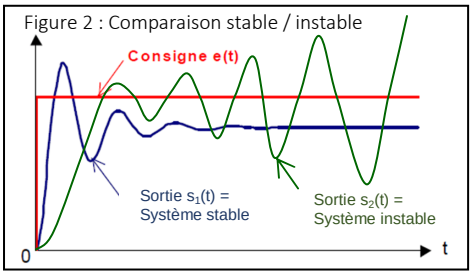


Figure 2 : Comparaison stable / instable

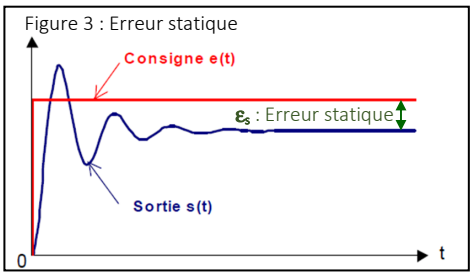


Figure 3 : Erreur statique

2. L'erreur statique (figure 3)

Lorsque la consigne est du type **échelon**, l'erreur pour t infini (ε(∞)) est appelée erreur statique ou erreur de position (ε_s).

$$\varepsilon_s = \varepsilon(\infty) = e(\infty) - s(\infty)$$

3. L'erreur dynamique ou de trainage ou de poursuite (figure 4)

Lorsque la consigne est du type **rampe**, l'erreur pour t infini (ε(∞)) est appelée erreur statique ou erreur de position (ε_d).

$$\varepsilon_d = \varepsilon(\infty) = e(\infty) - s(\infty)$$

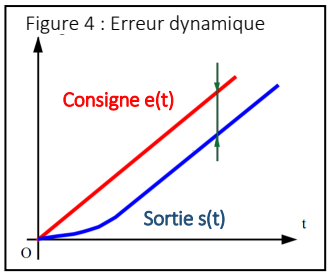


Figure 4 : Erreur dynamique

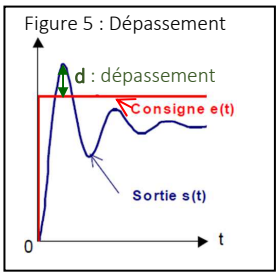


Figure 5 : Dépassement

4. Le dépassement (figure 5)

Lorsque le système a une réponse à la sollicitation trop forte et que cela l'amène à dépasser la consigne, on dit qu'il a un dépassement.

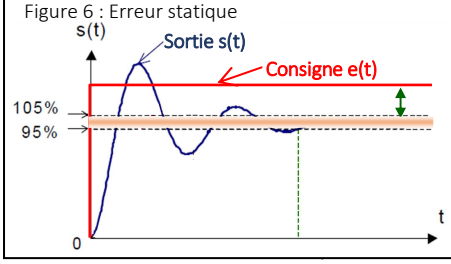
Le premier dépassement est souvent exprimé en % d'erreur par rapport à la consigne.

$$d\% = d / e(\infty) \times 100$$

5. Le temps de réponse (figure 6)

Le temps de réponse à 5% d'un système est le temps mis pour que sa sortie atteigne et reste dans l'intervalle [95% ; 105%] de la valeur finale stabilisée lorsque la consigne est un échelon.

Figure 6 : Erreur statique



3- CORRECTION

1. Défauts d'un système asservi

- Imprécision : l'erreur est trop grande ;
  le dépassement est trop grand.
- Lenteur : le système a un temps de réponse trop long.
- Instabilité : la sortie peut devenir oscillatoire peu amortie voir même instable (on dit qu'il pompe).

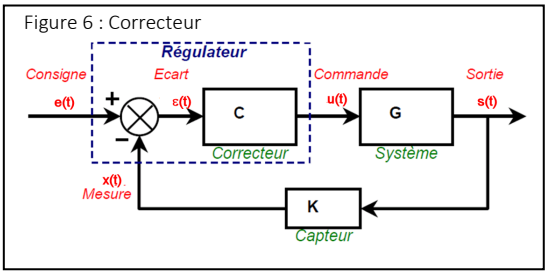
Un système asservi rapide et précis risque d'être instable.

Il y a donc un dilemme entre la précision et la stabilité, la rapidité.

2. Correcteur (figure 7)

On place un correcteur entre le bloc comparateur et le système pour corriger les défauts de l'asservissement :

Figure 6 : Correcteur



3. Types de correcteurs (étude qualitative)

Correcteur proportionnel **P**

- Il **augmente** le gain du système et donc sa **rapidité** et sa **précision**.
- Il peut rendre le système asservi **instable**.

Correcteur proportionnel intégral **PI**

- Il augmente le gain en basse fréquence sans déstabiliser le système asservi, il améliore donc la précision.
- Il peut même **annuler l'erreur statique**.

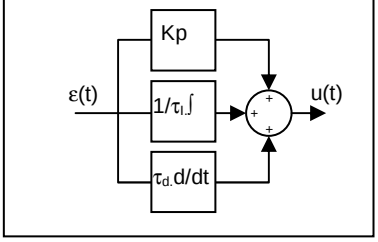
Correcteur proportionnel dérivé **PD**

- Il augmente la marge de phase et stabilise le système asservi.
- Il peut aussi augmenter **la rapidité**.

Correcteur proportionnel intégral et dérivé **PID** (figure 8)

- Il combine l'action des correcteurs précédents pour améliorer les performances globales du système asservi, mais les gains proportionnel, intégrale et dérivé sont délicats à déterminer.

Figure 8 : Correcteur PID parallèle



GENIE INFORMATIQUE ET AUTOMATISME

Asservissement et régulation - Structure

18

1- PREAMBULE (FIGURE 1)

Un système est un ensemble de processus physiques-chimiques en évolution réalisé dans le but d'obtenir des objectifs en fonction d'une action sur les entrées de commande.

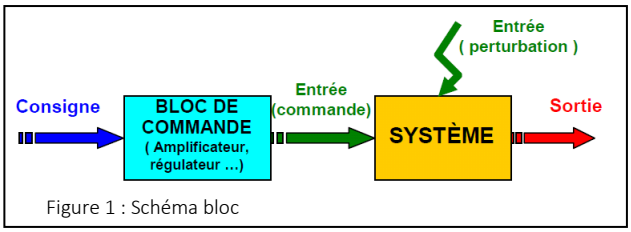


Figure 1 : Schéma bloc

Pour pouvoir commander sereinement le système (donner une consigne) on ajoute à celui-ci un bloc de commande qui va traduire la consigne en commande.

Les **signaux** relatifs à un système sont de deux types :

- Signaux d'entrées : ils sont indépendants du système et peuvent être « commandables » (consignes) ou non « commandables » (perturbations).
- Signaux de sorties : ils sont dépendants du système et des signaux d'entrées. Pour évaluer les objectifs, ces signaux doivent être observables par utilisation de capteurs.

La consigne peut être de 2 types :

- Signal analogique : par exemple la tension de sortie d'un potentiomètre.
- Information numérique : contenu d'une variable informatique, par exemple la variable *position* dans le cas d'une commande de position angulaire d'une antenne.

2- BOUCLE OUVERTE

Un système est en boucle ouverte lorsqu'on n'a aucune information sur la sortie (on ne contrôle pas l'objectif que l'on se fixe).

La moindre perturbation ne sera pas corrigée puisqu'on ne contrôle pas l'état de la sortie.

La figure 2 représente un four dont l'opérateur ne peut que régler le débit de gaz (en agissant sur la consigne). L'ouverture du four, une température extérieure différente de celle habituelle, … perturbent la température intérieure du four (celle qu'on aimerait précisément contrôler) sans que cela ne puisse être corrigé par le système. Seul l'opérateur peut agir (au petit bonheur la chance) en augmentant ou réduisant la consigne.

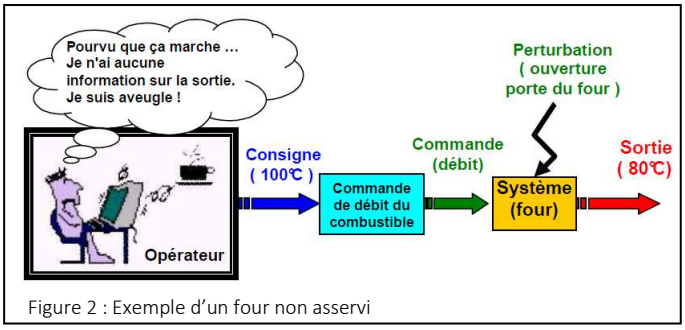


Figure 2 : Exemple d'un four non asservi

3- BOUCLE FERMEE (FIGURES 3 ET 4)

Pour remédier au problème de la boucle ouverte et contrôler parfaitement la grandeur de sortie, on ferme la boucle (figures 3 et 4). Cela signifie qu'on mesure la grandeur de sortie grâce à un capteur et on la compare à la consigne. En fonction du signe et de l'importance de la différence entre consigne et mesure, on agit plus ou moins fortement sur le bloc de commande.

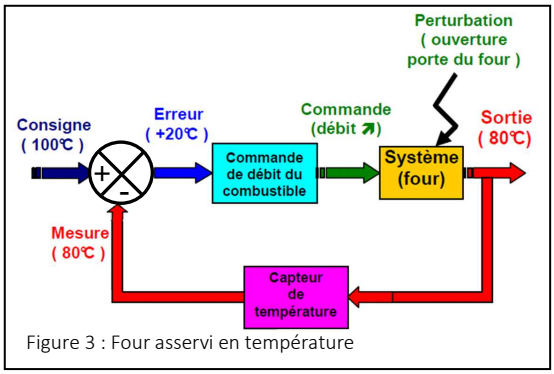


Figure 3 : Four asservi en température

4- A RETENIR (FIGURE 4)

Un système asservi (figure 4 à connaitre) :

- cherche à faire en sorte que la sortie (s(t)) suive la consigne (e(t)) ;
- compare la consigne (e(t)) à l'image de la sortie (mesure) (x(t)) et agit sur le système pour contrer les perturbations ;
- contient un bloc de commande (plus ou moins sophistiqué et un comparateur (soustracteur).

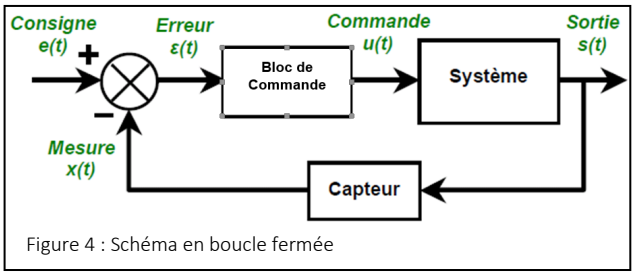


Figure 4 : Schéma en boucle fermée

La différence entre un asservissement et une régulation :

- Un asservissement : la consigne peut varier et on cherche à la suivre en permanence (asservissement de position, …) ;
- Une régulation : c'est un asservissement dont la consigne est fixe (régulation de température, de vitesse, …)

GENIE INFORMATIQUE ET AUTOMATISME

Protocole de communication MODBUS

A1

Fonctions possibles

MODBUS offre 19 fonctions différentes. Elles se caractérisent par un code fonction sur un octet (en hexadécimal).

Tous les équipements ne supportent pas tous les codes fonction.

Code	Nature de la fonction Modbus	Données liées à la requête	Supporté par l'Altivar
01	Lecture de n bits de sortie consécutifs		
02	Lecture de n bits d'entrée consécutifs		
03	Lecture de n mots de sortie consécutifs	adresse (2octets), longueur n (2octets)	X
04	Lecture de n mots d'entrée consécutifs		
05	Ecriture de 1 bit de sortie		
06	Ecriture de 1 mot de sortie	adresse (2octets), valeur (2octets)	X
07	Lecture du statut d'exception		
08	Accès aux concepteurs de diagnostic		
09	Téléchargement, télé-déchargement et modes de marche		
0A	Demande de compte-rendu de fonctionnement		
0B	Lecture du compteur d'événements		
0C	Lecture d'événements de connexion		
0D	Téléchargement, télé-déchargement et modes de marche		
0E	Demande de compte-rendu de fonctionnement		
0F	Ecriture de n bits de sortie		
10	Ecriture de n mots de sortie	Adresse de base, valeurs des mots écrits (2 * n octets)	X
11	Lecture identification		
12	Téléchargement, télé-déchargement et modes de marche		
13	Reset de l'esclave après erreur non recouverte		