Université de Batna 2
Faculté de Technologie
Département de Génie Industriel

Support de cours

INFORMATIQUE INDUSTRIELLE

Option : *Mécatronique*

Année : Master I

Pr Hassen BOUZGOU

2019/2020

SOMMAIRE

FICHE TECHNIQUE

Objectif du cours

Le cours vise à donner aux étudiants un vue générale sur les notions de base de l'informatique industrielle, en mettant l'accent sur les systèmes micro-programmés, leurs techniques de programmation et les aspects pratiques de leur développement. L'objectif du cours est d'assurer une formation de base large et diversifiée, dans un esprit multidisciplinaire.

L'ensemble de la matière contient un cours, des travaux dirigés et des travaux pratiques de programmation des systèmes à base de PIC 16F84.

Pré requis

Electronique

Logique combinatoire

Programmation assembleur

Mode d'évaluation

Contrôle continu basé sur le travail personnel (40%), et examen écrit en fin du semestre sur l'ensemble de la matière vue au cours (60%).

CHAPITRE 1 : INTRODUCTION A L'INFORMATIQUE INDUSTRIELLE

1.1 Introduction

Un ordinateur est une machine de traitement de l'information. Il est capable d'acquérir de l'information, de la stocker, de la transformer en effectuant des traitements quelconques puis de la restituer sous une autre forme. La suite des instructions décrivant la façon dont l'ordinateur doit effectuer un certain travail est appelé programme. L'ensemble des instructions exécutables est appelé langage. Un ordinateur peut alors être considéré comme une hiérarchie de niveaux. A chaque niveau correspond une machine virtuelle et un langage associé.

L'informatique industrielle est une branche de l'informatique appliquée qui couvre l'ensemble des techniques de conception et de programmation, de systèmes informatisés à vocation industrielle, qui ne sont pas des ordinateurs.

1.2 Domaines d'applications :

- Automates, robotique,
- Mesures de grandeurs physiques,
- Systèmes temps-réel,
- Systèmes embarqués.

Figure 1.1 : Applications de l'informatique Industrielle

1.3 Les différents systèmes programmables

1.3.1 Les circuits spécialisés ou ASIC (Application Specific Integrated Circuit):

En français (circuit intégré développé pour un client). Les ASIC sont des circuits intégrés spécialisés optimisés pour une application (par exemple les cartes graphiques, les cartes son) et beaucoup plus rapides que les processeurs généralistes. C'est le cas par exemple des puces utilisées dans les téléphones portables.

Aujourd'hui, on développe un circuit électronique numérique en utilisant un langage de description de matériel[1] (VHDL, Verilog ou encore SystemC), qui est ensuite compilé par synthèse logique pour produire automatiquement le dessin du circuit.

Source : Texas Instruments

Source : NVidia

Figure 1.2 Quelques exemples des circuits ASIC

Avantages :

- Très rapide ;
- Consommation moindre ;
- Optimisé pour une application.

Inconvénients :

- Faible modularité (décomposition en modules) ;
- Possibilité d'évolution limitée ;
- Coût.

1.3.2 Les systèmes PLD

C'est des systèmes en logique programmée et/ou en logique programmable sont connus sous la désignation de PLD (Programmable Logic Device, circuit logique programmable). C'est un circuit intégré logique qui peut être reprogrammé après sa fabrication. Il est composé de nombreuses cellules logiques élémentaires pouvant être librement assemblé.

1 Un langage de description de matériel, ou du matériel (ou HDL pour hardware description language en anglais) est un langage informatique permettant la description d'un circuit électronique. Celui-ci peut décrire les fonctions réalisées par le circuit (description comportementale) ou les portes logiques utilisées par le circuit (description structurelle).

1.3.2.1 P.A.L. (Programmable Array Logic)

C'est à dire réseau logique programmable. La programmation de ces circuits s'effectue par destruction de fusibles. Une fois programmés on ne peut plus les effacer.

1.3.2.2 Les G.A.L. (Generic Array Logic) ou encore réseau logique générique. Leur fonctionnement est identique aux P.A.L. CMOS , ils sont programmables et effaçables électriquement.

1.3.2.3 C.P.L.D. (Complex Programmable Logic Device). Ces circuits sont composés de plusieurs P.A.L.s élémentaires (Par exemple l'équivalent de P.A.L.s 22V10) reliés entre-eux par une zone d'interconnexion. Ils permettent d'atteindre des vitesses de fonctionnement élevées (plusieurs centaines de Mhz).

1.3.2.4 Les L.C.A. & F.P.G.A. à anti-fusible.

- Les L.C.A. Ce qui signifie Logic Cell Array ou encore réseau de cellules logiques. Ces circuits sont composés de blocs logiques élémentaires de 2000 à 10000 portes que l'utilisateur peut interconnecter.

- Les F.P.G.A. à anti fusibles sont identiques aux L.C.A sauf qu'ils permettent une plus grande intégration de portes et ils ne sont pas effaçables électriquement. Le nom anti-fusible vient de la programmation des connexions qui s'effectue par fermeture de circuits, comparé aux fusibles où l'on ouvre les circuits.

Avantages :

- Forte modularité.
- Rapidité.

Inconvénients :

- Mise en œuvre plus complexe.
- Coûts de développement élevé.

1.4 Les systèmes micro-programmés :

Les micro-contrôleurs sont typiquement des systèmes micro-programmés. C'est un circuit intégré comprenant essentiellement un microprocesseur, ses mémoires, et des éléments personnalisés selon l'application.

Avantages :

- Mise en œuvre simple.
- Coûts de développement réduits.

Inconvénients :

- Plus lent.
- Utilisation sous optimale.

1.4.1 Types de processeurs

Un microprocesseur CISC à jeu d'instructions étendues, ou ***Complex Instruction Set Computer***, désigne un microprocesseur possédant un jeu d'instructions comprenant de très nombreuses instructions mixées à des modes d'adressages complexes.

- Grand nombre d'instructions,
- Type de processeur le plus répandu

Un processeur RISC à jeu d'instructions réduit (**Reduced Instruction Set Computer**) est un type particulier d'architecture matérielle de processeurs qui se caractérise par un jeu d'instructions réduit, aisé à décoder, uniquement composé d'instructions simples.

- Nombre d'instructions réduit.
- (sélection des instructions pour une exécution plus rapide).
- Décodage des instructions plus rapide.

1.5 Les différents bus d'un système micro-programmés

Définition : Un bus est un jeu de lignes partagées pour l'échange de mots numériques. Un bus permet de faire transiter (liaison série/parallèle) des informations codées en binaire entre deux points. Typiquement les informations sont regroupées en mots : octet (8 bits), word (16 bits) ou double word (32 bits).

Caractéristiques d'un bus:

- nombres de lignes,
- fréquence de transfert.

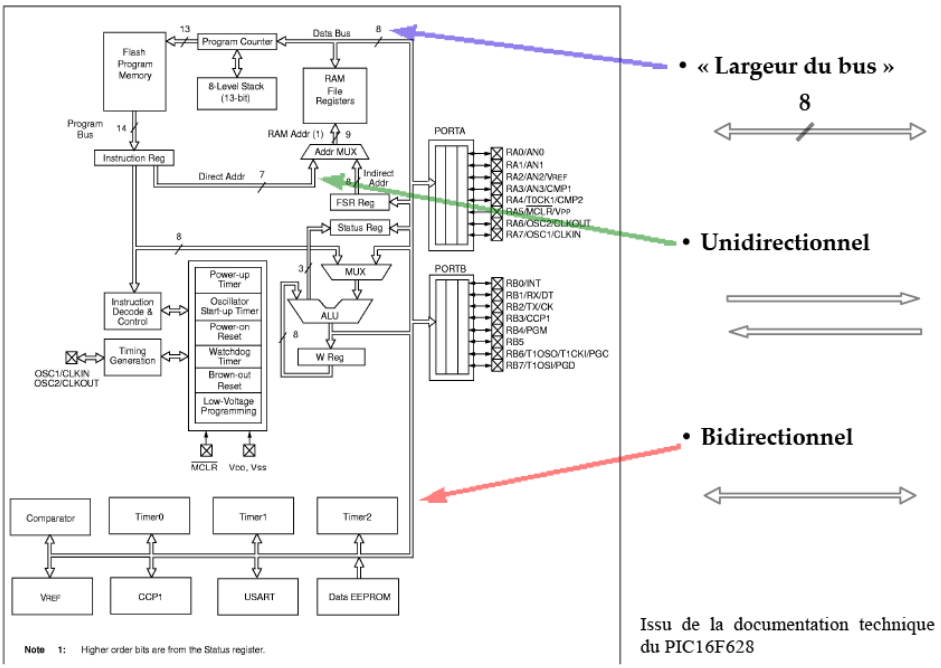


Figure 1.3. shéma fonctionel du PIC16F628

1.5.1 Types de bus :

● *Bus de données :* permet de transférer entre composants des données,

ex. : résultat d'une opération, valeur d'une variable, etc.

● *Bus d'adresses :* permet de transférer entre composants des adresses,

ex. : adresse d'une case mémoire, etc.

● *Bus de contrôle :* permet l'échange entre les composants d'informations de contrôle (bus rarement représenté sur les schémas).

ex. : périphérique prêt/occupé, erreur/exécution réussie, etc.

Définition : Une **adresse** est un nombre binaire qui indique un emplacement dans une zone mémoire.

1.6 Structures des systèmes micro-programmés

1.6.1 Structure de Von Neumann

À la base de l'architecture de presque tous les ordinateurs. La machine de Von Neumann est composée de cinq parties : la mémoire, l'unité arithmétique et logique, l'unité de contrôle, et les dispositifs d'entrées et de sorties.

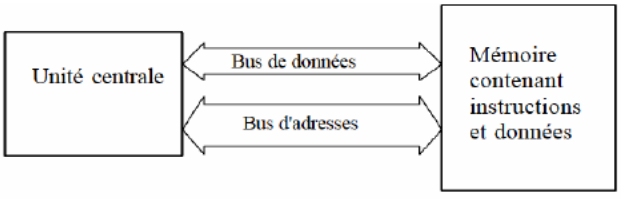


Figure 1.4 Structure de Von Neumann

1.6.2 Structure de Harvard

L'architecture de type Harvard est une conception des processeurs qui sépare physiquement la mémoire de données et la mémoire programme. L'accès à chacune des deux mémoires s'effectue via deux bus distincts.

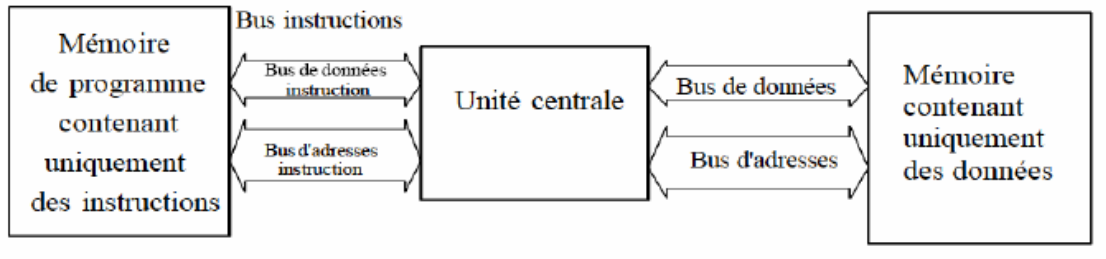


Figure 1.5 Structure de Harvard

La **différence** se situe au niveau de la séparation ou non des mémoires programmes et données. La structure de Harvard permet de transférer données et instruction simultanément, ce qui permet un gain de performances et de rapidité.

CHAPITRE 2 : LE MICROCONTROLEUR

2.1 Introduction

Un microcontrôleur (en notation abrégée µc) est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte pour le programme ROM, mémoire vive pour les données RAM), unités périphériques et interfaces d'entrées-sorties. Les microcontrôleurs se caractérisent par un plus haut degré d'intégration, une plus faible consommation électrique, une vitesse de fonctionnement plus faible (de quelques mégahertz jusqu'à plus d'un gigahertz1) et un coût réduit par rapport aux microprocesseurs polyvalents utilisés dans les ordinateurs personnels.

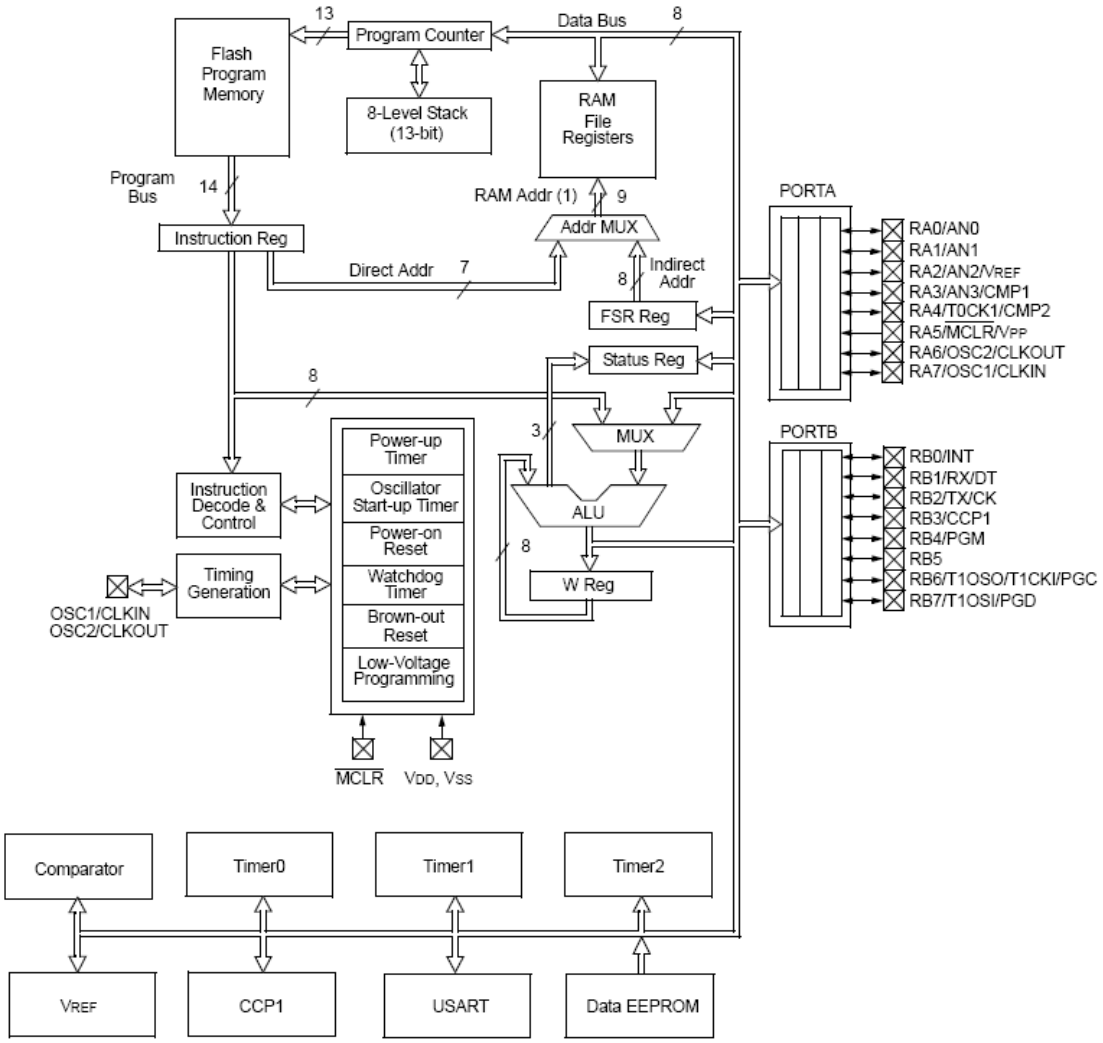


Figure 2.1 Schéma bloc d'un microcontrôleur PIC16F628

2.2 Composants d'un Microcontrôleur

2.2.1 Les mémoires :

RAM (Random Access Memory)

Mémoire rapide qui permet de stocker temporairement des données.

ROM (Read Only Memory)

Mémoire à lecture seule, programmée à vie.

EEPROM (Elec. Erasable Programmable Read Only Memory)

Mémoire lente qui permet de stocker des données même après coupure de l'alimentation.

2.2.2 PC (Program Counter)

Dans un processeur, le compteur ordinal ou pointeur d'instruction, est le registre (souvent nommé PC) qui contient l'adresse mémoire de l'instruction en cours d'exécution ou prochainement exécutée (cela dépend de l'architecture). Une fois l'instruction chargée, il est automatiquement incrémenté pour pointer l'instruction suivante.

2.2.3 Registres

Se sont des cases mémoires, chaque instruction à exécuter est chargée dans le registre d'instructions qui le tient pendant qu'il est décodé, préparé et finalement exécuté, ce qui peut prendre plusieurs étapes. (Instruction Reg, FSR Reg, Status Reg, W Reg).

2.2.4 ALU (L'unité arithmétique et logique)

C'est l'organe de l'ordinateur chargé d'effectuer les calculs. Le plus souvent, l'UAL est incluse dans l'unité centrale ou le microprocesseur.

2.2.5 Multiplexeur (MUX)

C'est un circuit permettant de concentrer sur une même voie de transmissions différentes types de liaisons.

2.2.6 Décodeur d'instructions (Instruction decode and control)

Circuit de l'UC qui reçoit le code opération d'une instruction et le transforme en signaux de contrôle nécessaires à l'exécution de l'instruction.

2.2.7 horloge (Timing generation)

Circuit qui génère un signal d'horloge numérique pour la synchronisation.

2.2.8 Pile (Stack)

Elle correspond alors à une zone de la mémoire, et le processeur retient l'adresse du dernier élément. ***Deux principes:*** *(LIFO (Last In First Out), FIFO (First In First Out)).*

2.2.9 Ports d'entrées/sorties

C'est les périphériques d'où viennent les échanges d'informations entre le processeur et les périphériques qui lui sont associés.

2.2.10 Interfaces de communication série USART

(Universal Synchronous Asynch.ReceiverTransmitter), est un organe permettant des échanges de données série (un bit à la fois) entre un microprocesseur et un périphérique.

2.2.11 Modulation en largeur d'impulsions CCP

(Capture/Compare/PWM), est une technique couramment utilisée pour synthétiser des signaux continus à l'aide de circuits à états discrets (binaires)[1].

2.2.12 Timer

Son rôle est de permettre la synchronisation des opérations que le microcontrôleur est chargé d'effectuer.

2.2.13 Comparateur

Il n'indique pas une mesure absolue mais une mesure relative par rapport à un point de référence.

2.2.14 Convertisseur analogique-numérique (CAN/CNA)

C'est un montage électronique dont la fonction est de traduire une grandeur analogique en une valeur numérique et vice versa.

2.2.15 Référence de tension.

Pour comparer les différents signaux d'entrée.

1 Le principe général est qu'en appliquant une succession d'états discrets pendant des durées bien choisies, on peut obtenir en moyenne sur une certaine durée n'importe quelle valeur intermédiaire.

CHAPITRE 3 : RAPPELS SUR LES NOMBRES BINAIRES

3.1 Introduction

Les différentes bases qui nous seront utiles :

- le binaire (base 2) est constitué de 2 chiffres : 0, 1
- l'octal (base 8), est constitué de 8 chiffres : 0, 1, 2, 3, 4, 5, 6, 7
- le décimal (base 10), est constitué de 10 chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
- l'hexadécimal (base 16), est constitué de 16 chiffres : 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F

Remarque : pour connaître la base associée à un nombre, on le note entre parenthèse avec en indice une lettre b,o,d ou h selon qu'il s'agit d'un codage binaire, octal, décimal ou hexadécimal. Par exemple, $(1001)_b$, $(3F1)_h$ ou $(128)_d$.

3.2 Codes pondérés

Dans une base donnée, le nombre s'exprime comme une somme pondérée. Par exemple, le nombre 128 décimale (base 10) est constitué de 3 chiffres :

- le chiffre 8 est affecté du poids de 1 (unités)
- le chiffre 2 est affecté du poids de 10 (dizaines) « Un Zéro »
- le chiffre 1 est affecté du poids de 100 (centaines) « Un Zéro Zéro»

Le nombre peut donc s'écrire

$$1 \times 100 + 2 \times 10 + 8 \times 1 = (\mathbf{128})_d$$

Chiffre — Poids

Le nombre 110 binaire (base 2) est constitué de 3 chiffres :

- le chiffre 0 est affecté du poids de 1 ;
- le chiffre 1 est affecté du poids de 2 ;
- le chiffre 1 est affecté du poids de 3.

Dans une base donnée, le nombre s'exprime comme une somme pondérée. Par exemple, le nombre 1D7 hexadécimal (base 16) est constitué de 3 chiffres:

- le chiffre 7 est affecté du poids de 1 (unités)
- le chiffre D est affecté du poids de 10 (dizaines)
- le chiffre 1 est affecté du poids de 100 (centaines)

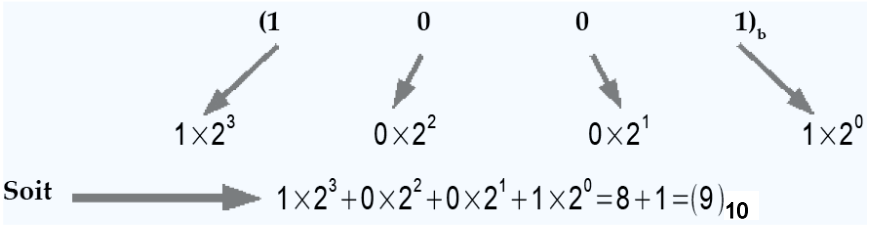
Le nombre peut donc s'écrire

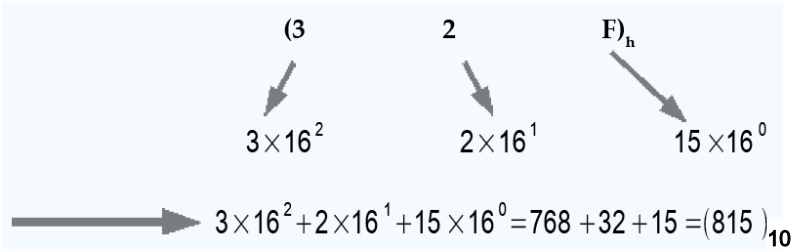
$$1 \times 100 + D \times 10 + 7 \times 1 = (1D7)_h$$

Conversion binaire-hexadécimal : le codage hexadecimal a été créé afin d'alléger l'exploitation des nombres binaires. Il permet en particulier une conversion simple par regroupement des bits par 4 en partant de la droite, chaque paquet étant alors simple à convertir :

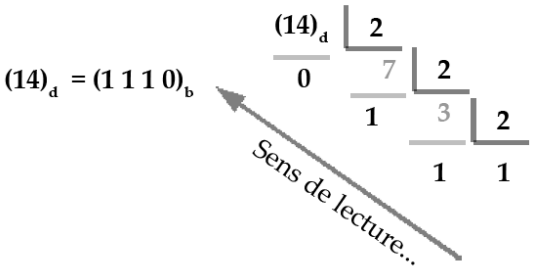
$$0001 \quad 1101 \quad 0111 = (1D7)_h$$

3.2.1 Conversion en décimal : développement en somme de puissances de la base :

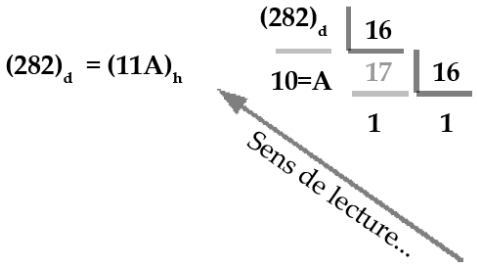




3.2.2 Conversion décimal-binaire : division par 2 successives...

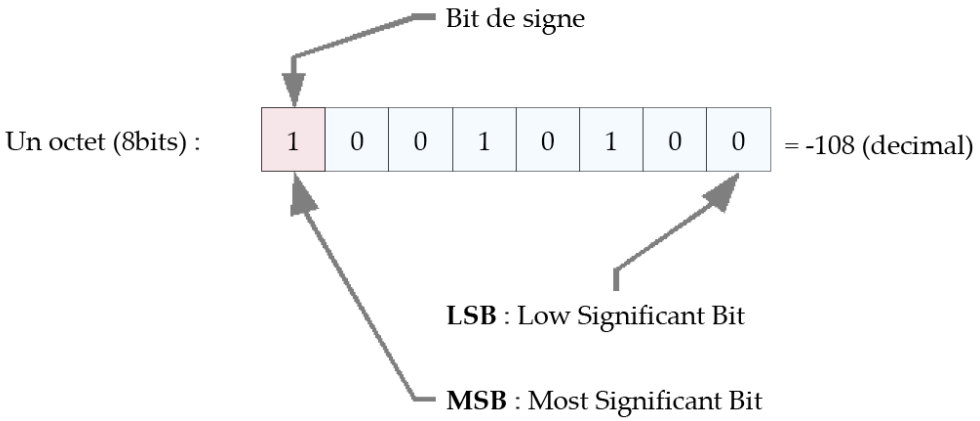


3.2.3 Conversion décimal - hexadécimal : division par 16 successives...



Pour indiquer le signe d'un nombre binaire, on ajoute un bit en tête du nombre.

On peut ainsi coder les entiers relatifs et les nombres réels.



Un octet (8 bits)

Un mot ou word (16 bits)

Un double mot ou double word (32 bits)

3.3 Opérations arithmétiques binaires

Les techniques de calcul des opérations arithmétiques peuvent être transposées du décimal au binaire.

Addition : V = A + B, Exemple : $(0110)_b + (0101)_b = (1011)_b$

0+0=0, 0+1=1, 1+1=0 (garder 1)

Soustraction : 0-0=0, 0-1=1(on retient 1), 1-1=0.

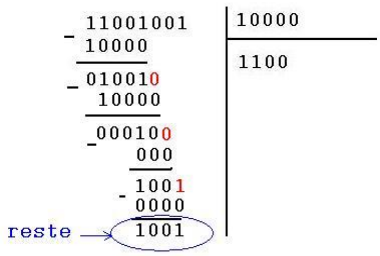
Multiplication :

La multiplication binaire s'effectue selon le principe des multiplications décimal, on multiplie donc le multiplicande par chacun des bits du multiplicateur. On décale les résultats intermédiaires obtenus et on effectue ensuite l'addition de ses résultats partiels.

V = A x B, Exemple : $(0110)_b . (0101)_b = (011110)_b$

La division :

La division va être basée sur une succession de soustraction et s'emploi de la même façon qu'une division décimal ordinaire.



CHAPITRE 4 : LES INSTRUCTIONS

4.1 Introduction

Un jeu d'instruction est un ensemble d'opérations directement réalisables sur un système micro-programmé.

> **Exemple :** le PIC18F4520 (RISC) à un jeu d'instructions composé de 75 instructions. L'exécution d'une instruction peut demander un ou plusieurs cycles d'horloges suivant la complexité de l'instruction.

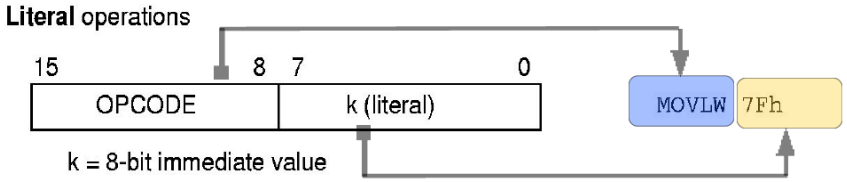
Cycle d'horloge : correspond à une période de l'horloge (signal de référence temporelle). La fréquence d'horloge est le nombre de cycles effectués par une horloge en une seconde.

4.2 Définition d'une instruction

Une instruction est composée au minimum de deux parties:

Instruction = OPCODE + opérande(s)

OPCODE (Operation CODE): partie d'une instruction qui précise quelle opération doit être réalisée.



4.3 Pipeline et flot d'instructions

3 étapes pour l'exécution d'une instruction :

- ✓ Lecture de l'instruction (1)
- ✓ Décodage de l'instruction (2)
- ✓ Exécution de l'instruction (3)

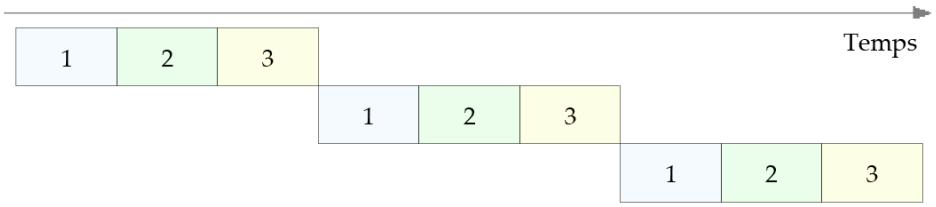


Figure 4.1 Instruction sans pipline

Un pipeline (ou chaîne de traitement), est l'élément d'un processeur dans lequel l'exécution des instructions est découpée en plusieurs étapes. Avec un pipeline, le processeur peut commencer à exécuter une nouvelle instruction sans attendre que la précédente soit terminée. Chacune des étapes du pipeline est implémentée par un circuit intégré indépendant, appelé étage. Le nombre d'étages d'un pipeline est appelé sa profondeur.

Création d'un pipeline => permet une exécution plus rapide des instructions.

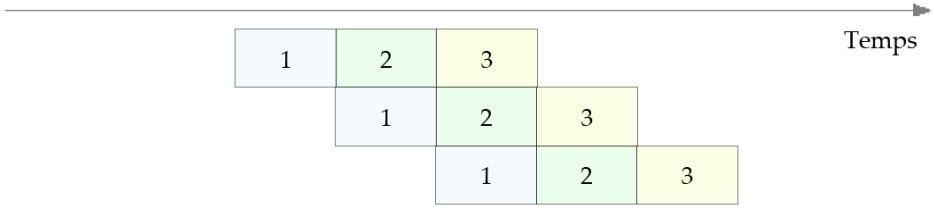


Figure 4.2 Instructions avec pipline

4.4 Les modes d'adressage

La nature et le nombre d'opérandes qui constituent une instruction déterminent le mode d'adressage de l'instruction. On distingue 4 modes d'adressage principaux.

4.4.1 L'adressage inhérent: il n'y a pas d'opérande !

ex : NOP, RESET, CLRWDT ;

RESET **Reset**

Syntax:	RESET
Operands:	None
Operation:	Reset all registers and flags that are affected by a MCLR Reset.
Status Affected:	All
Encoding:	0000 0000 1111 1111
Description:	This instruction provides a way to execute a MCLR Reset in software.
Words:	1
Cycles:	1

Q Cycle Activity:

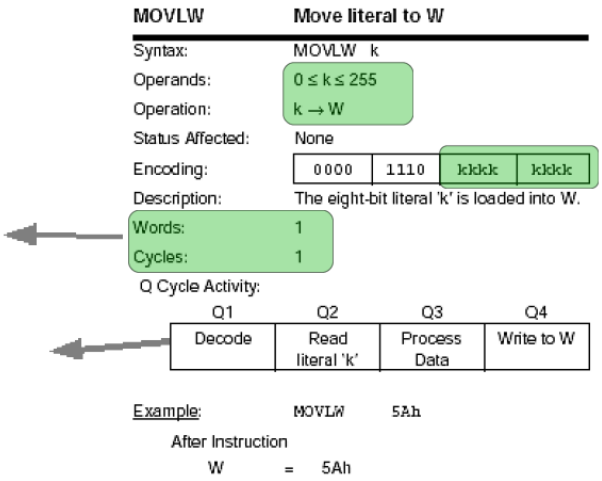
Q1	Q2	Q3	Q4
Decode	Start Reset	No operation	No operation

Example: RESET

After Instruction
Registers = Reset Value
Flags* = Reset Value

4.4.2 L'adressage immédiat : l'opérande à une valeur

ex : MOVLW 5Ah ;



Nombres de cycles nécessaires à l'exécution

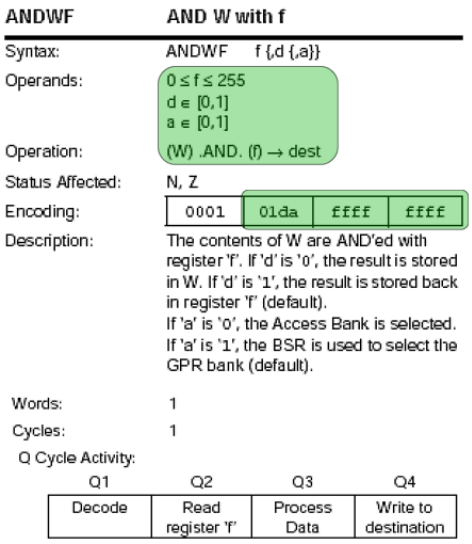
Exécution de l'instruction (pipeline à 4 niveaux)

4.4.3 L'adressage direct (étendu) : l'opérande est l'adresse (bits de poids faibles de l'adresse complète) de la donnée dans la page mémoire active.

ex : ADDWF 000Fh,

En mode direct étendu : on transmet l'adresse complète

```
Example:        ANDWF   REG, 0, 0
    Before Instruction
        W       =   17h
        REG     =   C2h
    After Instruction
        W       =   02h
        REG     =   C2h
```



4.4.4 L'adressage indirect (indexé):

L'opérande est l'adresse d'un registre qui contient l'adresse de la donnée.

En ***mode indirect indexé***, on ajoute un décalage par rapport à l'adresse.

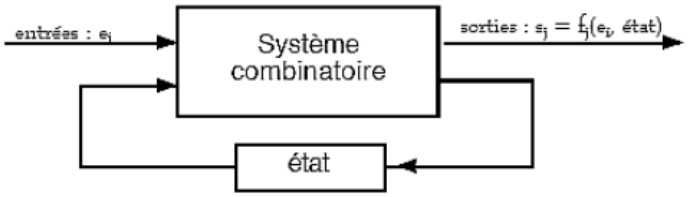
CHAPITRE 5 : LOGIQUE COMBINATOIRE ET SEQUENTIELLE

5.1 Introduction

La compréhension du fonctionnement d'un microcontrôleur s'appuie sur des connaissances élémentaires de logique combinatoire et séquentielle.

- **un système est dit combinatoire** si l'état (logique) des sorties ne dépend que de l'état (logique) présent appliqué à ses entrées.

- **un système est dit séquentiel** si l'état (logique) de la sortie du système à l'instant t dépend de l'état (logique) présent appliqué aux entrées et des états de la sortie dans le passé.



5.2 Table de vérité

Considérons tout d'abord le cas de la logique combinatoire à 1 sortie (le cas à plusieurs sorties n'est pas très différent). Pour connaître l'état du système aux divers combinaisons logiques des entrées on construit la table de vérité qui exprime la valeur de la sortie *s* en fonction de toutes les configurations possible des entrées binaires (E_i),

Table de vérité à 2 entrées :

E_1	E_2	s
0	0	x
0	1	x
1	0	x
1	1	x

Table de vérité à 3 entrées :

E_1	E_2	E_3	s
0	0	0	x
0	0	1	x
0	1	0	x
0	1	1	x
1	0	0	x
1	0	1	x
1	1	0	x
1	1	1	x

On notera que pour une fonction logique à une seule variable d'entrée, il existe 2^2=4 combinaisons de sorties.

E_1	F0	F1	F2	F3
0	0	1	0	1
1	0	0	1	1

De même, pour deux variables d'entrées, il existe 2^4=16 combinaisons de sorties.

E_1	E_2	F0	F1	F2	F3	F4	F5	F6	F7	F8	F9	F10	F11	F12	F13	F14	F15
0	0	0	0	0	0	0	0	0	0	1	1	1	1	1	1	1	1
0	1	0	0	0	0	1	1	1	1	0	0	0	0	1	1	1	1
1	0	0	0	1	1	0	0	1	1	0	0	1	1	0	0	1	1
1	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1	0	1

5.3 Opérateurs élémentaires

Il existe 6 fonctions logiques de base que sont les opérateurs NON (une entrée), ET, OU, ET-NON, OU-NON, et OU-EXCLUSIF (deux entrées).

Table de vérité :

E	S
0	1
1	0

NON

Symboles :

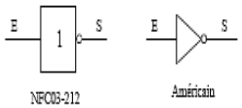


Table de vérité :

E_1	E_2	S
0	0	0
0	1	1
1	0	1
1	1	1

OU

Symboles :

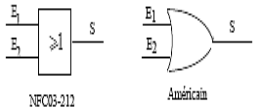


Table de vérité :

E_1	E_2	S
0	0	0
0	1	0
1	0	0
1	1	1

ET

Symboles :

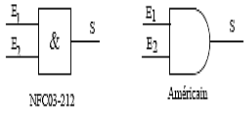


Table de vérité :

E_1	E_2	S
0	0	0
0	1	1
1	0	1
1	1	0

OU-EX

Symbole :

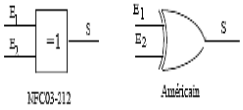


Table de vérité :

E_1	E_2	S
0	0	1
0	1	1
1	0	1
1	1	0

ET-NON

Symboles :

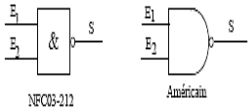
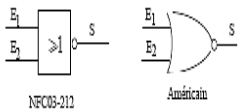


Table de vérité :

E_1	E_2	S
0	0	1
0	1	0
1	0	0
1	1	0

OU-NON

Symboles :



Les opérateurs ET-NON et OU-NON forment un groupe complet, c.à.d. que toute fonction logique complexe peut être construite sur la base de l'une de ces fonctions élémentaires.

5.4 Algèbre de BOOLE

Les opérateurs logiques élémentaires permettent la construction de « l'**algèbre de Boole** ». Ainsi, si on considère deux entrées binaires A et B, on adopte alors la convention suivante pour construire des équations logiques:

NON (not)	$\bar{A}$	ET-NON (nand)	$\overline{A.B} = \bar{A} + \bar{B}$
ET (and)	A.B	OU-NON (nor)	$\overline{A+B} = \bar{A} . \bar{B}$
OU (or)	A+B	OU-EXCLUSIF (xor)	$A \oplus B$

Les différentes opérations bénéficient des propriétés suivantes

Associativité :

$$(A\cdot B)\cdot C = A\cdot B\cdot C$$
$$(A+B)+C = A+B+C$$
$$(A\oplus B)\oplus C = A\oplus B\oplus C$$

Commutativité :

$$A\cdot B = B\cdot A \qquad \overline{A\cdot B} = \overline{B\cdot A}$$
$$A+B = B+A \qquad \overline{A+B} = \overline{B+A}$$
$$A\oplus B = B\oplus A$$

Distributivité :

$$A+(B\cdot C) = (A+B)\cdot(A+C)$$
$$A\cdot(B+C) = A\cdot B + A\cdot C$$

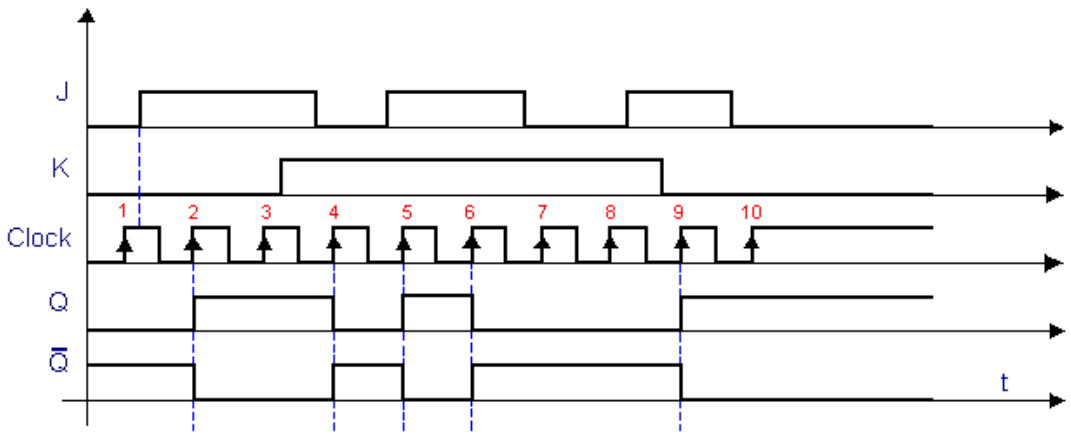
Lois de De Morgan :

$$\overline{A\cdot B} = \overline{A}+\overline{B}$$
$$\overline{A+B} = \overline{A}\cdot\overline{B}$$

5.5 Le chronogramme

Dans les microcontrôleurs, les états du système changent en fonction d'une base de temps qui est l'horloge. Les chronogrammes sont des outils d'analyse des états logiques d'un système.

Le chronogramme a pour objet de tracer l'état binaire des sorties en fonction de l'évolution temporelle de l'état des entrées.

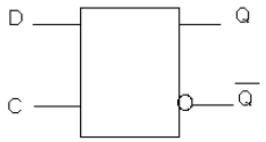


5.6 Les bascules asynchrones

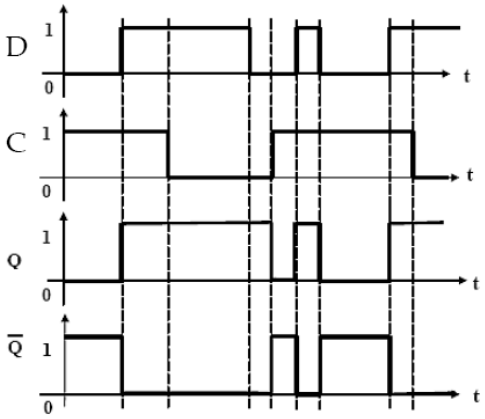
Une bascule asynchrone est une fonction « mémoire » qui est commandée. Ce type de fonction est notamment utilisé pour créer des registres du microcontrôleur.

5.6.1 Le verrou D (Latch D)

Le verrou D (ou bascule D asynchrone) copie en sortie l'état de l'entrée D uniquement si sa commande C est active; dans le cas contraire, l'état en sortie Q est celui précédent.



Entrées		Sorties	
C	D	Qn+1	$\overline{Qn+1}$
0	X	Qn	$\overline{Qn}$
1	0	0	1
1	1	1	0

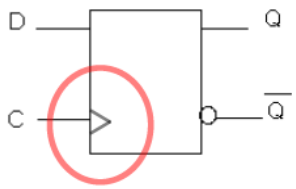


5.7 Les bascules synchrones

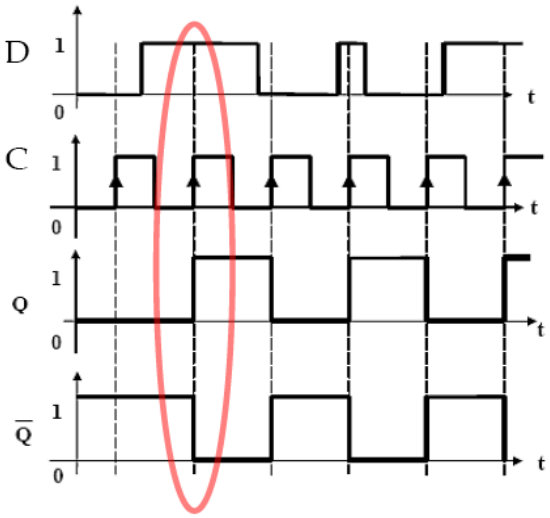
5.7.1 La Bascule D (Flip-Flop D)

Une bascule synchrone est une bascule ***qui ne change d'état que sur front montant ou descendant*** appliqué sur son entrée de commande. C'est la version synchrone du verrou D.

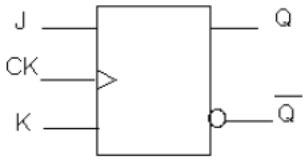
- quand H est à 0, la sortie maintient son état, quel que soit le niveau appliqué à D.
- quand H est à 1, la sortie Q recopie l'état de D.



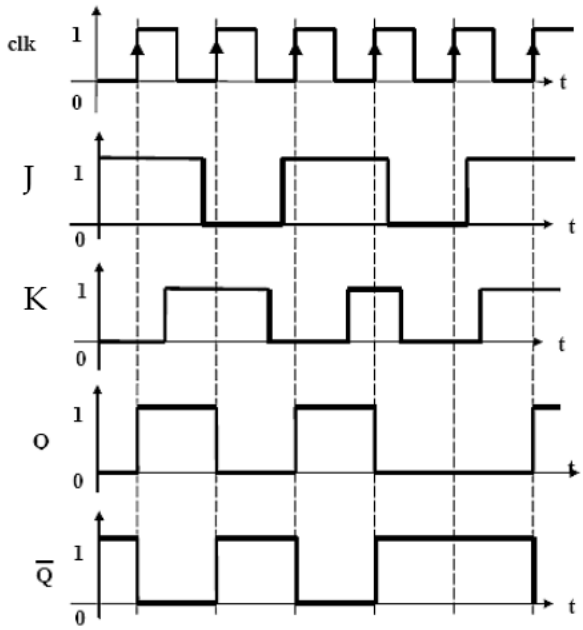
Entrées		Sorties	
C	D	Qn+1	$\overline{Qn+1}$
0	X	Qn	$\overline{Qn}$
↑	0	0	1
↑	1	1	0



5.7.2 La Bascule JK (Jump-Knock out)



Entrées			Sorties	
CK	J	K	Qn+1	$\overline{Qn+1}$
0	X	X	Qn	$\overline{Qn}$
1	X	X	Qn	$\overline{Qn}$
↓	X	X	Qn	$\overline{Qn}$
↑	0	0	Qn	$\overline{Qn}$
↑	0	1	0	1
↑	1	0	1	0
↑	1	1	$\overline{Qn}$	Qn



Cette bascule comporte deux entrées de contrôle : J (Jack) et K (King). Le fonctionnement est synchrone à une entrée d'horloge H, c'est-à-dire que la valeur de sortie ne peut changer qu'au moment d'un front d'horloge, montant ou descendant selon les modèles.

- Pour J = K = 0, il y a conservation du dernier état logique Q_{n-1} indépendamment de l'horloge : état mémoire.
- Pour J = K = 1, le système bascule à chaque front d'horloge.
- Pour J différent de K, la sortie Q recopie l'entrée J et la sortie $\bar{Q}$ recopie l'entrée K à chaque front d'horloge.

CHAPITRE 6 : ÉTUDE DU FONCTIONNEMENT D'UN MICROCONTROLEUR PIC

6.1 Introduction

Un PIC est un microcontrôleur de Microchip. Ses caractéristiques principales sont:

- Séparation des mémoires de programme et de données (architecture Harvard) ;
- Communication avec l'extérieur seulement par des ports : il ne possède pas de bus d'adresses, de bus de données et de bus de contrôle comme la plupart des microprocesseurs ;
- Utilisation d'un jeu d'instructions réduit (Type RISC) : Les instructions sont ainsi codées sur un nombre réduit de bits, ce qui accélère l'exécution (1 cycle machine par instruction sauf pour les sauts qui requirent 2 cycles).

6.1.1 Principales caractéristiques du PIC 16F84:

- 35 instructions.
- Instructions codées sur 14 bits.
- Données sur 8 bits.
- 1 cycle machine par instruction, sauf pour les sauts (2 cycles machine).
- Vitesse maximum 10 MHz soit une instruction en 400 ns (1 cycle machine = 4 cycles d'horloge).
- 4 sources d'interruption.
- 1000 cycles d'effacement/écriture pour la mémoire flash, 10.000.000 pour la mémoire de donnée EEPROM.

6.2 Brochage et fonction des pattes

Les fonctions des pattes sont les suivantes:

- V_{SS}, V_{DD} : Alimentation *(V_{DD}=12 ou 5v)*
- OSC1,2 : Horloge
- RA0-4 : Port A
- RB0-7 : Port B
- T0CKL : Entrée de comptage
- INT : Entrée d'interruption
- MCLR : Reset : 0V
- Choix du mode programmation : 12V - 14V
- exécution : 4.5V - 5.5V

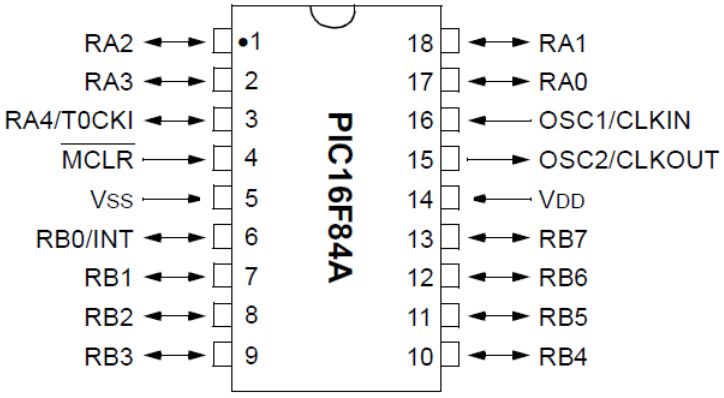


Figure 6.1 : Brochage du circuit.

6.3 Architecture générale

Il est constitué des éléments suivants :

- un système d'initialisation à la mise sous tension (power-up timer, …) ;
    - *Module utilisé par certains microcontrôleurs pour maintenir le système remis à zéro jusqu'à ce que l'oscillateur à cristaux liquides soit stable*
- un système de génération d'horloge à partir du quartz externe (timing génération) ;
- une unité arithmétique et logique (ALU) ;
- une mémoire flash de programme de 1k "mots" de 14 bits;
- un compteur de programme (program counter) et une pile (stack) ;
- un bus spécifique pour le programme (program bus) ;
- un registre contenant le code de l'instruction à exécuter ;
- un bus spécifique pour les données (data bus) ;
- une mémoire RAM contenant ;
    - les SFR ;
    - 68 octets de données ;
- une mémoire EEPROM de 64 octets de données ;
- 2 ports d'entrées/sorties ;
- un compteur (timer) ;
    - *permettre la synchronisation des opérations que le microcontrôleur est chargé d'effectuer*
- un chien de garde (watchdog) ;
    - *un circuit électronique ou un logiciel utilisé en électronique numérique pour s'assurer qu'un automate ou un ordinateur ne reste pas bloqué à une étape particulière du traitement qu'il effectue. C'est une protection destinée généralement à redémarrer le système, si une action définie n'est pas exécutée dans un délai imparti*

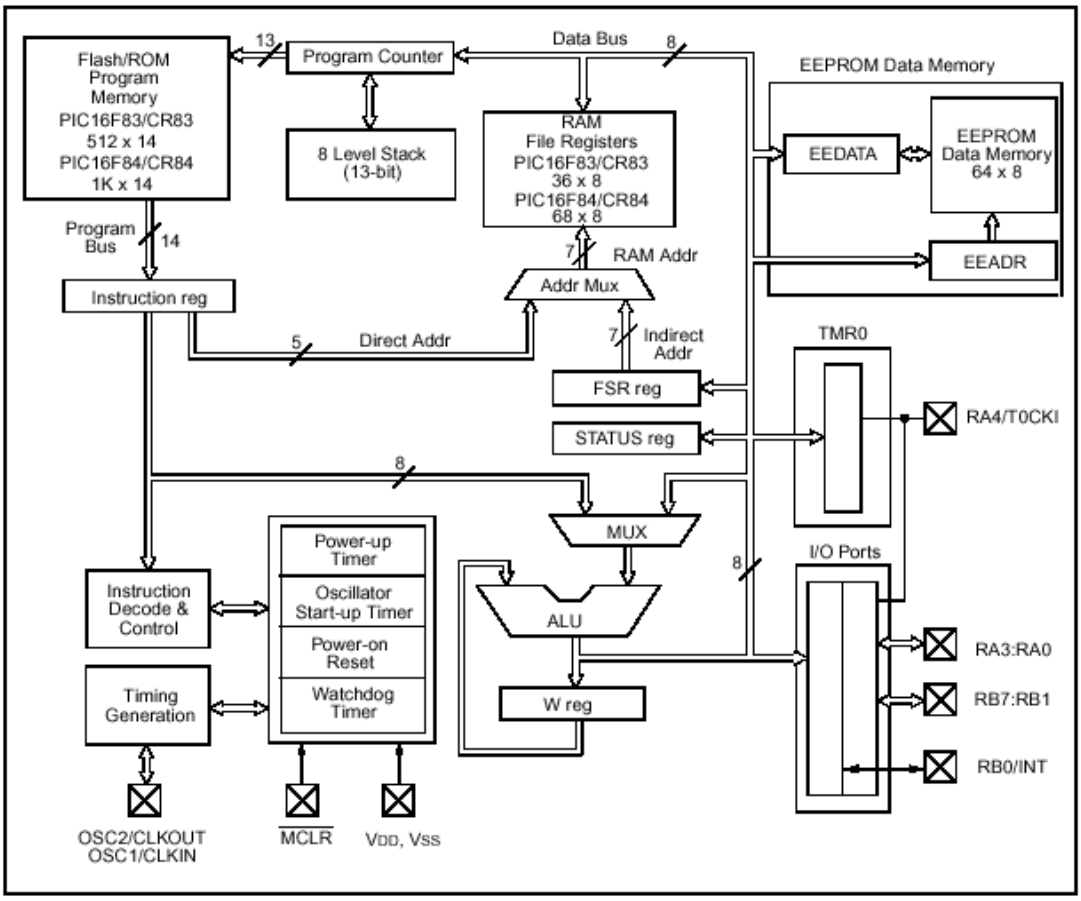


Figure 6.2: Architecture générale du PIC 16F8X.

6.4 Organisation de la mémoire

Avec une structure Harvard, Le PIC contient la mémoire programme et la mémoire données avec un accès séparé à chacune pendant le même cycle machine.

6.4.1 Mémoire de programme

Elle contient le programme à exécuter qui est téléchargé par liaison série. Elle contient 1k "mots" de 14 bits dans le cas du PIC 16F84, même si le compteur de programme (PC) de 13 bits peut en adresser 8k.

- L'adresse 0000h contient le vecteur du reset ;
- l'adresse 0004h l'unique vecteur d'interruption du PIC.

La pile contient 8 valeurs. Comme le compteur de programme, elle n'a pas d'adresse dans la plage de mémoire. Ce sont des zones réservées par le système.

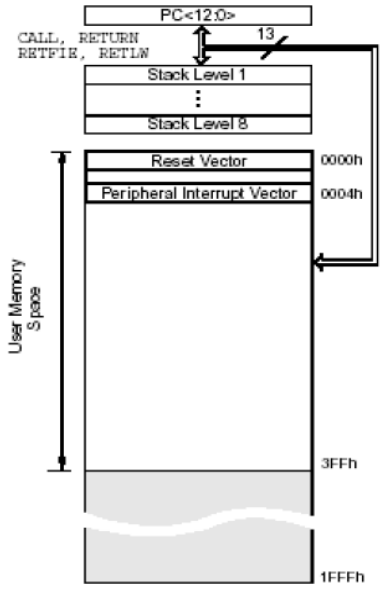


Figure 6.3: Organisation de la mémoire de programme et de la pile.

6.4.2 Mémoire de données

Elle est composée en deux parties de RAM et une zone EEPROM. La première contient les SFRs (Special Function Registers) qui permettent de contrôler les opérations sur le circuit. La seconde contient des registres généraux, libres pour l'utilisateur. La dernière contient 64 octets.

Registres généraux

Ils sont accessibles soit directement soit indirectement à travers les registres FSR et INDF.

Registres spéciaux - SFRs

Ils permettent la gestion du circuit. Certains ont une fonction générale, d'autres une fonction spécifique attachée à un périphérique donné.

Mémoire EEPROM

Le PIC possède une zone EEPROM de 64 octets accessibles en lecture et en écriture par le programme. Elle sauvegarde des valeurs même si l'alimentation est éteinte, et les récupérer lors de la mise sous tension. Leur accès est spécifique et requiert l'utilisation de registres dédiés.

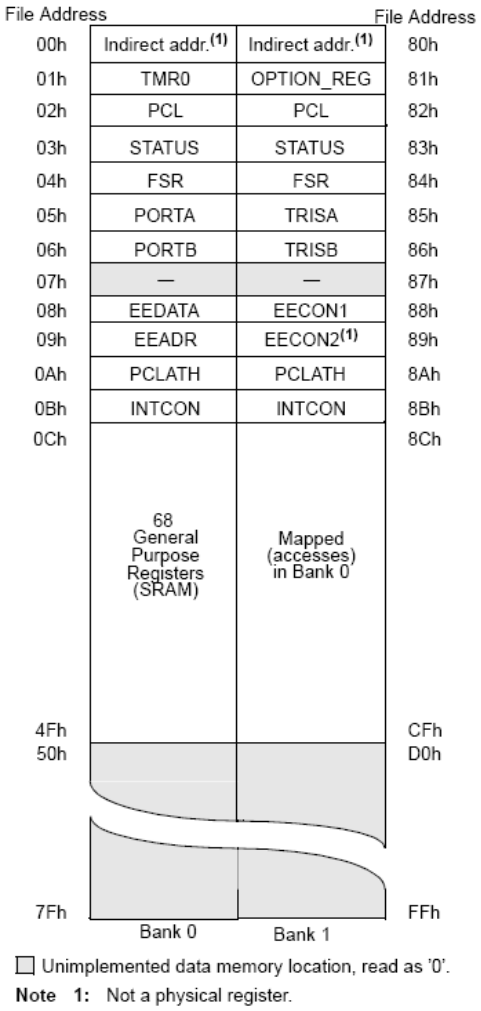


Figure 6.4: Organisation de la mémoire de données.

INDF (00h - 80h) : Utilise le contenu de FSR pour l'accès indirect à la mémoire.

TMR0 (01h) : Registre lié au compteur. PCL (02h - 82h) : Contient les poids faibles du compteur de programmes (PC). Le registre PCLATH (0Ah-8Ah) contient les poids forts.

STATUS (03h - 83h) : Il contient l'état de l'unité arithmétique et logique ainsi que les bits de sélection des banques.

FSR (04h - 84h) : Permet l'adressage indirect.

PORTA (05h) : Donne accès en lecture ou écriture au port A.

PORTB (06h) : Donne accès en lecture ou écriture au port B.

EEDATA (08h) : Permet l'accès aux données dans la mémoire EEPROM.

EEADR (09h) : Permet l'accès aux adresses de la mémoire EEPROM.

PCLATCH (0Ah - 8Ah) : Donne accès en écriture aux bits de poids forts du compteur de programme.

INTCON (0Bh - 8Bh) : Masque d'interruptions.

OPTION_REG (81h) : Contient des bits de configuration pour divers périphériques.

TRISA (85h) : Indique la direction (entrée ou sortie) du port A.

TRISB (86h) : Indique la direction (entrée ou sortie) du port B.

EECON1 (88h) : Permet le contrôle d'accès à la mémoire EEPROM.

EECON2 (89h) : Permet le contrôle d'accès à la mémoire EEPROM.

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C
bit 7							bit 0

bit 7-6 **Unimplemented:** Maintain as '0'

bit 5 **RP0**: Register Bank Select bits (used for direct addressing)
01 = Bank 1 (80h - FFh)
00 = Bank 0 (00h - 7Fh)

bit 4 $\overline{TO}$: Time-out bit
1 = After power-up, CLRWDT instruction, or SLEEP instruction
0 = A WDT time-out occurred

bit 3 $\overline{PD}$: Power-down bit
1 = After power-up or by the CLRWDT instruction
0 = By execution of the SLEEP instruction

bit 2 **Z**: Zero bit
1 = The result of an arithmetic or logic operation is zero
0 = The result of an arithmetic or logic operation is not zero

bit 1 **DC**: Digit carry/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{borrow}$, the polarity is reversed)
1 = A carry-out from the 4th low order bit of the result occurred
0 = No carry-out from the 4th low order bit of the result

bit 0 **C**: Carry/$\overline{borrow}$ bit (ADDWF, ADDLW, SUBLW, SUBWF instructions) (for $\overline{borrow}$, the polarity is reversed)
1 = A carry-out from the Most Significant bit of the result occurred
0 = No carry-out from the Most Significant bit of the result occurred

Note: A subtraction is executed by adding the two's complement of the second operand. For rotate (RRF, RLF) instructions, this bit is loaded with either the high or low order bit of the source register.

Legend:			
R = Readable bit	W = Writable bit	U = Unimplemented bit, read as '0'	
- n = Value at POR	'1' = Bit is set	'0' = Bit is cleared	x = Bit is unknown

Figure 6.5: Registre d'étai du PIC - STATUS..

6.5 Exécution d'un programme – notion de pipe-line

L'enchaînement des instructions et fait tous les 4 cycles d'horloge. Pendant un premier cycle machine, l'instruction à exécuter est stockée en mémoire RAM. Le cycle suivant, elle est exécutée. Chaque instruction dure donc 2 cycles machine.

La notion de pipeline permet de réduire ce temps à un seul cycle machine. L'idée est d'exécuter l'instruction n-1 pendant que l'instruction n est chargée en mémoire RAM. Ainsi, une fois le système enclenché, pendant chaque cycle machine une instruction est chargée et une autre exécutée.

On a donc l'équivalent d'une instruction par cycle machine. Notons que l'instruction CALL dure 2 cycles machine comme toutes les instructions de branchement.

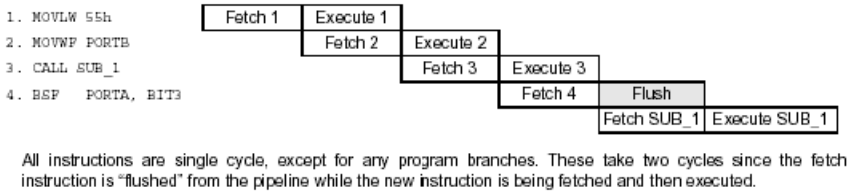


Figure 6.5 : Pipline du PIC.

6.6 Ports d'entrées/Sorties

Le PIC 16F84 est doté de deux ports d'entrées/Sorties appelés PortA et PortB.

6.6.1 Port A

Il comporte 5 pattes d'entrée/sortie bi-directionnelles, notées RAx avec x={0,1,2,3,4}. Le registre PORTA, d'adresse 05h dans la banque 0, permet d'y accéder en lecture ou en écriture. Le registre TRISA, d'adresse 85h dans la banque 1, permet de choisir le sens de chaque patte (entrée ou sortie) : un bit à 1 positionne le port en entrée, un bit à 0 positionne le port en sortie.

6.6.2 Port B

Il comporte 8 pattes d'entrée/sortie bi-directionnelles, notées RBx avec x={0,1,2,3,4,5,6,7}. Le registre PORTB, d'adresse 06h dans la banque 0, permet d'y accéder en lecture ou en écriture. Le registre TRISB, d'adresse 86h dans la banque 1, permet de choisir le sens de chaque patte (entrée ou sortie) : un bit à 1 positionne le port en entrée, un bit à 0 positionne le port en sortie.

6.7 Interruptions

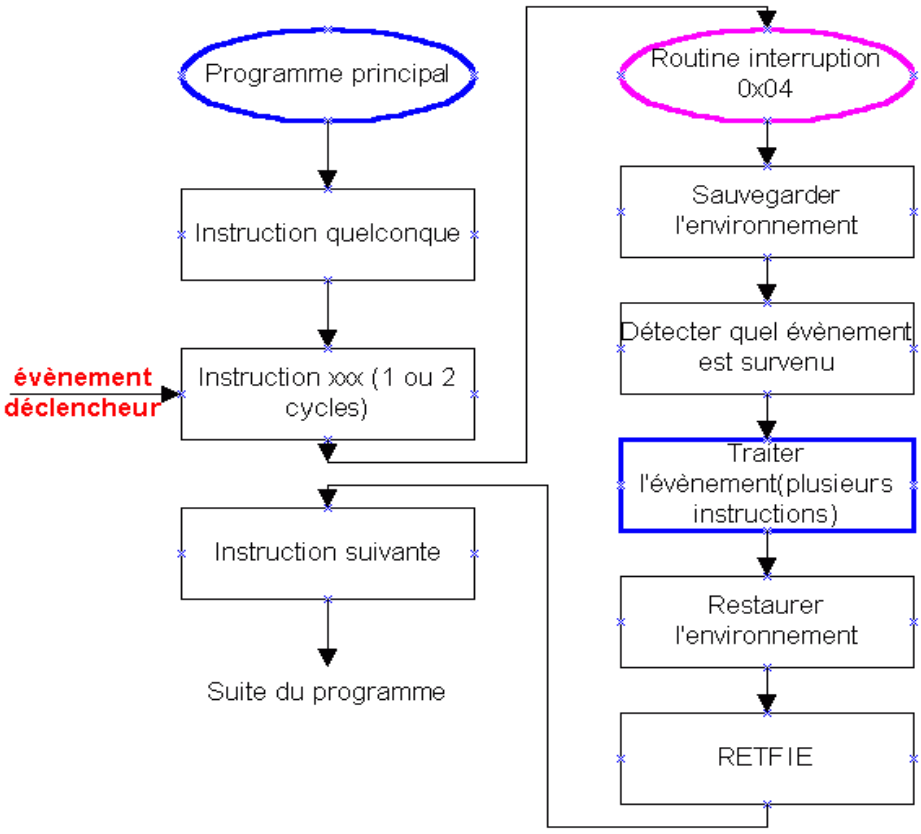


Figure 6.6 : Déroulement d'un programme lors d'une interruption.

L'interruption est un mécanisme fondamental de tout processeur. Il permet de prendre en compte des événements extérieurs au processeur et de leur associer un traitement spécifique. Il faut noter que l'exécution d'une instruction n'est jamais interrompue ; c'est à la fin de l'instruction en cours lors de l'arrivée de l'événement que le sous-programme d'interruption est exécuté.

La séquence classique de fonctionnement d'une interruption est la suivante :

1- Détection de l'événement déclencheur.
2- Fin de l'instruction en cours.
3- Sauvegarde de l'adresse de retour.
4- Déroutement vers la routine d'interruption.
5- Sauvegarde du contexte.
6- Identification de l'événement survenu.
7- Traitement de l'interruption correspondante.
8- Restauration du contexte.
9- Retour au programme initial.

6.8 Chien de garde

C'est un système de protection contre un blocage du programme. Par exemple, si le programme attend le résultat d'un système extérieur (conversion analogique numérique par exemple) et qu'il n'y a pas de réponse, il peut rester bloquer. Pour en sortir on utilise un chien de garde. Il s'agit d'un compteur qui, lorsqu'il arrive en fin de comptage, permet de redémarrer le programme.

6.9 Schémas de base

Un système minimum peut être le suivant, avec simplement une alimentation 5Vdc, un quartz 4MHz de type, deux condensateurs 27pF céramique et un PIC 16F84 4MHz.

Dans ce cas, l'entrée de reset MCLR\ est connectée à l'alimentation positive +5V pour un reset de type POR (Power On Reset : reset à la mise sous tension) interne au PIC.

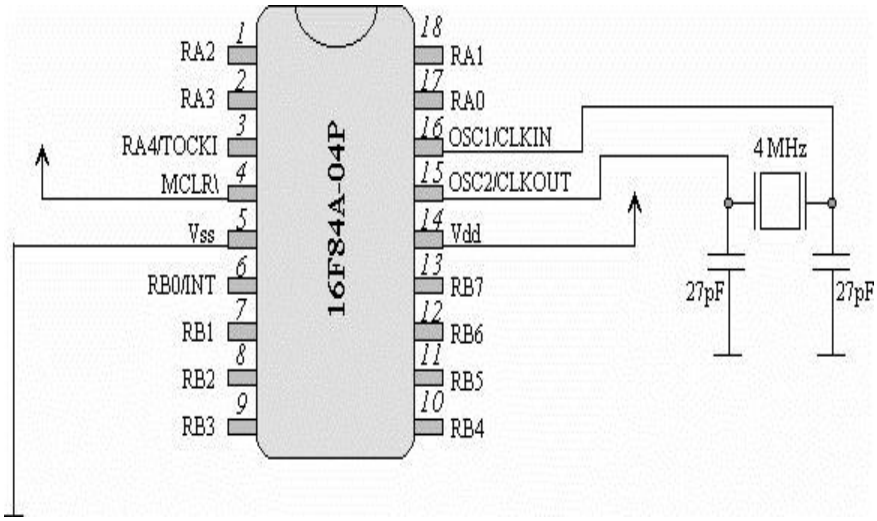


Figure 6.7 : Polarisation d'un PIC.

6.10 Jeu d'instructions

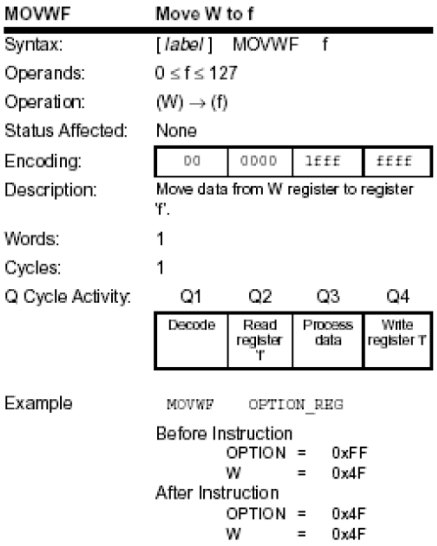
Les PICs sont conçus selon une architecture RISC. Programmer avec un nombre d'instructions réduit permet de limiter la taille de leur codage et donc de la place mémoire et du temps d'exécution.

Toutes les instructions sont codées sur 14 bits. Elles sont regroupées en trois grands types :

- Instructions orientées octets
- Instructions orientées bits
- Instructions de contrôle

Le registre de travail W joue un rôle particulier dans un grand nombre d'instructions.

Exemple d'instruction – le transfert



MOVWF	Move W to f			
Syntax:	[*label*] MOVWF f			
Operands:	0 ≤ f ≤ 127			
Operation:	(W) → (f)			
Status Affected:	None			
Encoding:	00	0000	1fff	ffff
Description:	Move data from W register to register 'f'.			
Words:	1			
Cycles:	1			

Q Cycle Activity:	Q1	Q2	Q3	Q4
	Decode	Read register 'f'	Process data	Write register T

Example

```
MOVWF   OPTION_REG
Before Instruction
        OPTION  =   0xFF
        W       =   0x4F
After Instruction
        OPTION  =   0x4F
        W       =   0x4F
```

Elle permet de transférer le contenu du registre W dans un registre f.

W : régistre de travail (accumulateur), taille 8 bits

k : valeur littérale, taille 8 bits

Mnémonique , opérande	Description	bit du registre STATUS affecté	nombre de cycles
MOVLW k	k (8 bits) est chargé dans (W)	-	1
ADDLW k	Additionne k (8 bits) et (W) et place le résultat dans (W)	C, DC , Z	1
SUBLW k	Soustrait W de k (8 bits) et place le résultat dans (W) k - (W) -> (W)	C, DC , Z	1
ANDLW k	Réalise un ET logique entre k (8 bits) et (W), et place le résultat dans (W)	Z	1
IORLW k	Réalise un OU logique (inclusif) entre k (8 bits) et (W), et place le résultat dans (W)	Z	1
XORLW k	Réalise un OU exclusif entre k (8 bits) et (W), et place le résultat dans (W)	Z	1

L : label (étiquette)

Mnémonique , opérande	Description	bit du registre STATUS affecté	nombre de cycles
GOTO L	Branchement à l'adresse L	-	2
CALL L	Appelle un sous-programme (subroutine) situé à l'adresse L	-	2
RETURN	Retour de sous-programme	-	2
RETLW k	Retour de sous-programme, avec chargement de la valeur littérale k (8 bits) dans (W)	-	2
RETFIE	Retour de sous-programme d'interruption	-	2
CLRWDT	Efface le Watchdog	/TO, /PD	1
SLEEP	Place le microcontrôleur en mode sommeil	/TO, /PD	1

f : registre (spécial ou d'usage général)

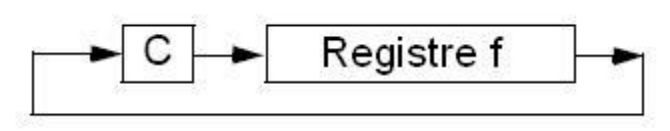
b : position du bit (0 à 7)

Mnémonique , opérande	Description	bit du registre STATUS affecté	nombre de cycles
BCF f , b	Mise à 0 du b ème bit du registre f	-	1
BSF f , b	Mise à 1 du b ème bit du registre f	-	1
BTFSC f , b	Si le b ème bit du registre f est égal à 0, alors l'instruction suivante est ignorée, et une instruction NOP est exécutée à la place (soit 2 cycles)	-	1 ou 2
BTFSS f , b	Si le b ème bit du registre f est égal à 1, alors l'instruction suivante est ignorée, et une instruction NOP est exécutée à la place (soit 2 cycles)	-	1 ou 2

f : registre (spécial ou d'usage général)

d : registre de destination (on peut choisir entre le *registre de travail* **W** et le *registre* **f**).

Mnémonique , opérande	Description	bit du registre STATUS affecté	nombre de cycles
MOVWF f	(W) est chargé dans (f)	-	1
MOVF f , d	(f) (8 bits) est chargé dans (destination)	Z	1
ADDWF f , d	Additionne le contenu du registre f (8 bits) et (W), et place le résultat dans (destination)	C, DC , Z	1
SUBWF f , d	Soustrait (W) de (f) (8 bits) et place le résultat dans (destination). (f) - (W) ->(destination)	C, DC , Z	1
ANDWF f , d	Réalise un ET logique entre (f) (8 bits) et (W), et place le résultat dans (destination)	Z	1
IORWF f , d	Réalise un OU logique (inclusif) entre (f) (8 bits) et (W), et place le résultat dans (destination)	Z	1
XORWF f , d	Réalise un OU exclusif entre (f) (8 bits) et (W), et place le résultat dans (destination)	Z	1
COMF f , d	Réalise le complément logique de (f) (8 bits), et place le résultat dans (destination)	Z	1
DECF f , d	Décrémente (f) et place le résultat dans (destination). (f) - 1 -> (destination)	Z	1
DECFSZ f , d	Décrémente (f) et place le résultat dans (destination). Si le résultat est 0, alors l'instruction suivante est ignorée, et une instruction NOP est exécutée à la place (soit 2 cycles)	-	1 ou 2
INCF f , d	Incrémente (f) et place le résultat dans (destination). (f) + 1 -> (destination)	Z	1
INCFSZ f , d	Incrémente (f) et place le résultat dans (destination).	-	1 ou 2

	Si le résultat est 0, alors l'instruction suivante est ignorée, et une instruction NOP est exécutée à la place (soit 2 cycles)		
CLRF f	Efface le contenu du registre (f). Remarque : le bit Z est donc mis à 1.	Z	1
CLRW	Efface le contenu de l'accumulateur (W). Remarque : le bit Z est donc mis à 1.	Z	1
RLF f , d	Réalise une rotation circulaire à gauche :  Le résultat est placé dans (destination).	C	1
RRF f , d	Réalise une rotation circulaire à droite :  Le résultat est placé dans (destination).	C	1
SWAPF f , d	Les 4 bits de poids forts et les 4 bits de poids faibles de (f) sont échangés. Le résultat est placé dans (destination).	-	1
NOP	Cette instruction ne fait rien (durée 1 cycle).	-	1

CHAPITRE 7 : ACQUISITION ET TRAITEMENT NUMERIQUE DES DONNEES

7.1 Introduction

Chaîne d'acquisition et de restitution de données

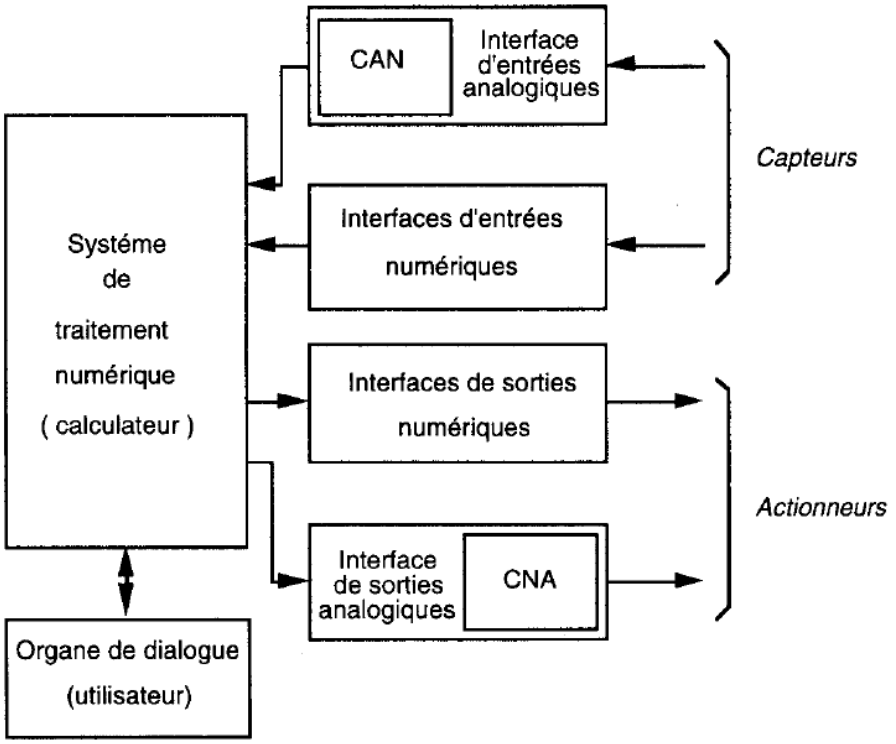


Figure 7.1 : Exemple typique d'un système de contrôle-commande industriel.

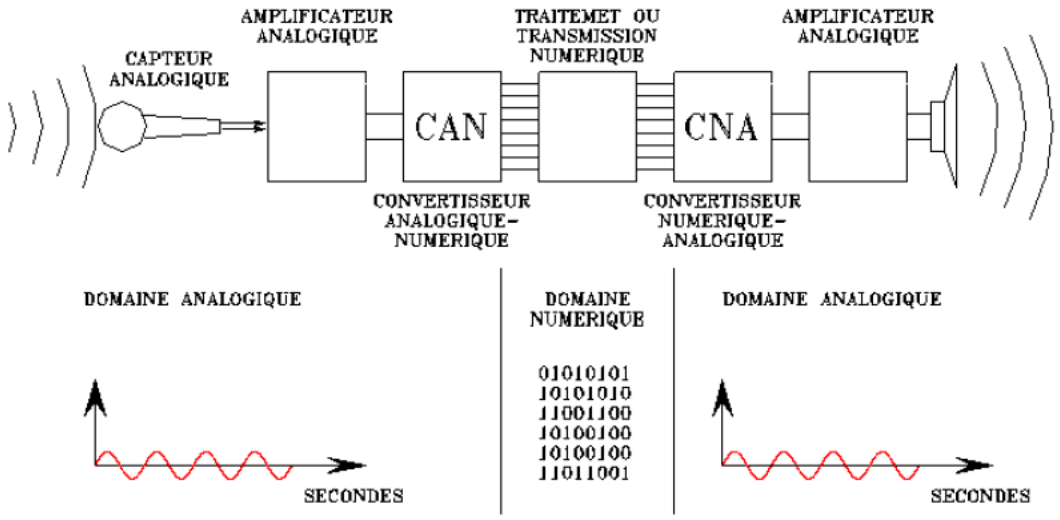


Figure 7.2 : Chaîne classique d'acquisition de données.

7.2 Le théorème de l'échantillonnage (théorème de Shannon)

Les signaux d'informations sont presque toujours de type analogique. Si l'on veut traiter un signal numériquement, il faut le représenter par une suite de valeurs ponctuelles prélevées régulièrement (à pas constant) ou irrégulièrement (à pas variable) échantillonnage.

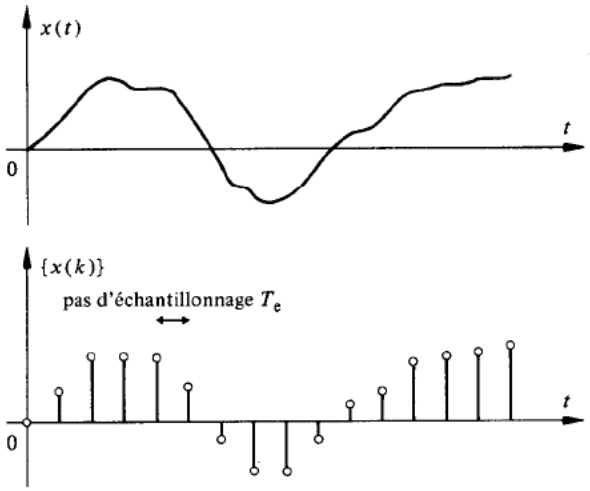


Figure 7.3. Echantillonnage d'un signal analogique

On sait que l'opérateur idéal d'échantillonnage est assimilable à un opérateur de modulation d'amplitude travaillant avec une porteuse constituée par une suite périodique d'impulsion de Dirac.

On appelle période d'échantillonnage T la durée entre la prise de 2 échantillons successifs. Son inverse est appelé cadence d'échantillonnage ou fréquence d'échantillonnage et sera noté

$$f_e = 1/T.$$

Notons par $x_a(t)$ le signal analogique de départ. La constitution du signal échantillonné $x_e(t)$ est obtenu par multiplication de ce signal analogique par un train d'impulsions de Dirac d'amplitude unité et équi-espacées de T. Ce train est noté p(t) et est défini par

$$p(t) = \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

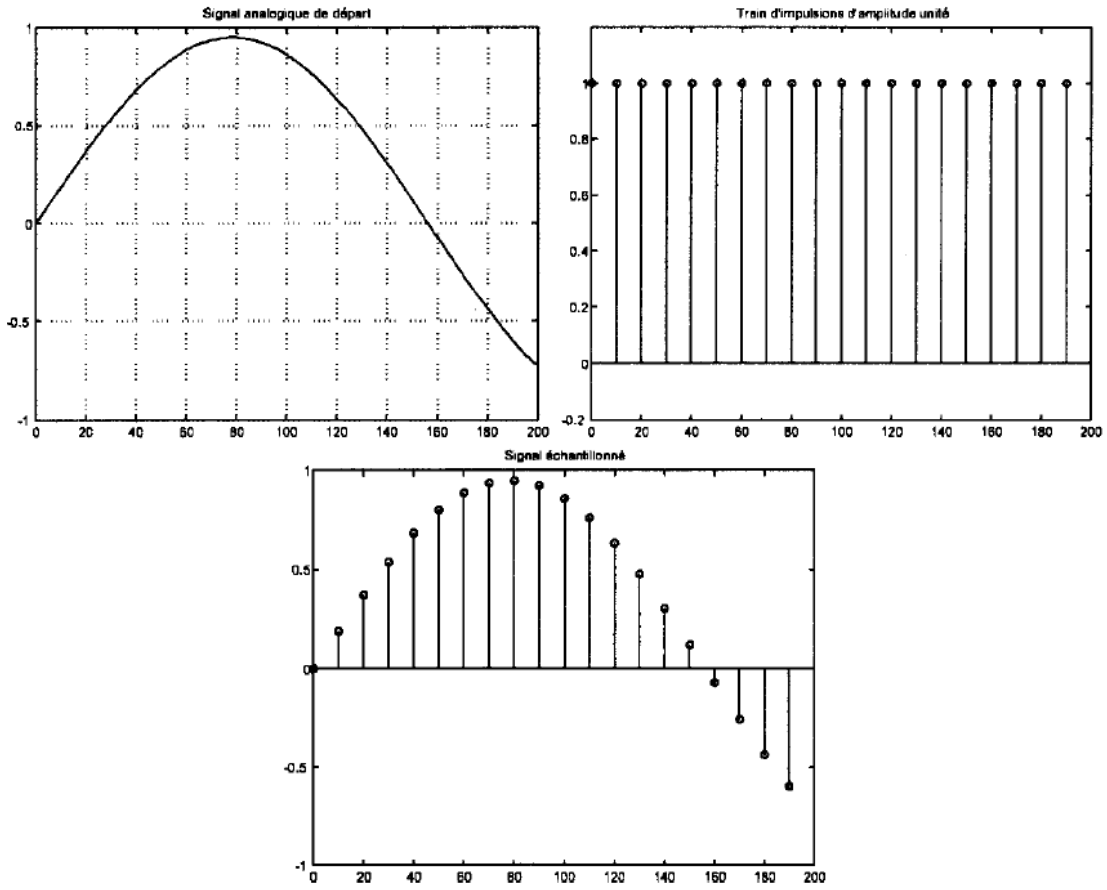


Figure 7.4. Opération d'échantillonnage : multiplication par un train d'impulsions

Le signal échantillonné correspond donc à:

$$x_e(t) = x_a(t).p(t) = x_a(t) \sum_{n=-\infty}^{\infty} \delta(t - nT)$$

On a f(t)δ(t) = f(0)δ(t), donc

$$x_e(t) = \sum_{-\infty}^{\infty} x_a(nT)\delta(t - nT)$$

7.3 Conversions Analogique/Numérique

les Convertisseurs Analogique Numérique (CAN, ADC en anglais, pour analog to digital converter), qui vont transformer les tensions analogiques en signaux logiques aptes à être traités par microprocesseur (numérisation des signaux).

Quand on voudra numériser un signal analogique (donc continu), il va falloir le discrétiser sur deux dimensions : le temps et l'amplitude.

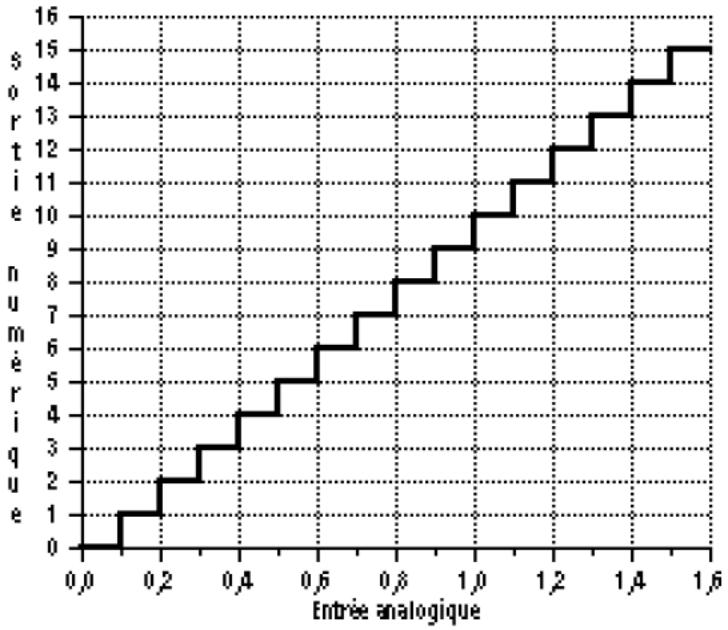


Figure 7.5. Fonction de transfert d'un CAN.

- Vlog = 0 pour 0 ≤ Ve < 0,1V
- V log = 1 pour 0,1≤ Ve < 0,2V
- V log = 6 pour 0,6 ≤ Ve < 0,7V ...

Plage de conversion

C'est la gamme de tension analogique d'entrée bornée : c'est la plage de conversion (ou tension de pleine échelle) du convertisseur. Cette plage de conversion sera couramment de 0-5V, 0-10V, ou encore ±5V ou ±10V.

Résolution

Le signal numérisé sera d'autant plus riche en information que l'intervalle de tension qui sera codé par le même nombre binaire sera petit.

Dynamique

La dynamique d'un signal est le rapport entre la tension maxi et la tension mini que pourra prendre ce signal.

7.4 Conversions Numérique /Analogique

les Convertisseurs Numérique Analogique (CNA, DAC en anglais, pour digital to analog converter) qui vont convertir les signaux logiques en tension analogique.

La fonction de transfert sera la même que celle de la figure 1 mais inversée. La tension de sortie aura une forme d'escalier.

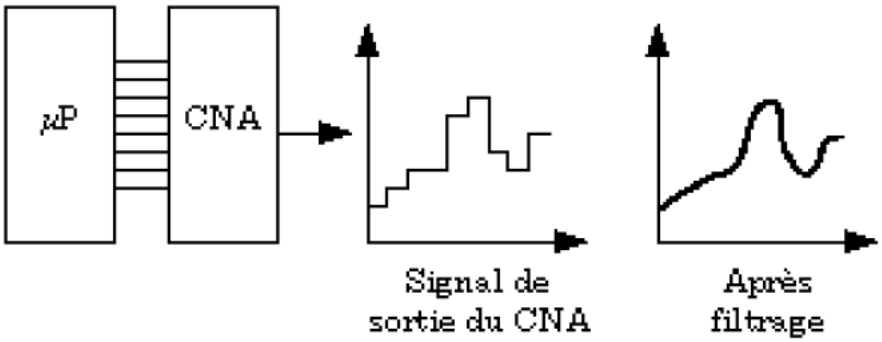


Figure 7.6. Conversion numérique analogique.

Résolution

La résolution du CNA sera la variation de tension de sortie correspondant à la variation d'une unité du nombre binaire en entrée. La définition est équivalente à celle du CAN.

Plage de conversion.

Il y a ici une petite différence avec le CAN (voir figure 8) : la plage de conversion numérique va de 0 à 2^N-1, N étant le nombre de bits du convertisseur, et à chaque valeur numérique correspond une valeur analogique de sortie et une seule. Par rapport à celle du CAN, la plage de conversion s'arrêtera donc un LSB plus tôt (sur l'échelle analogique du CAN, ceci correspond à la dernière transition numérique).