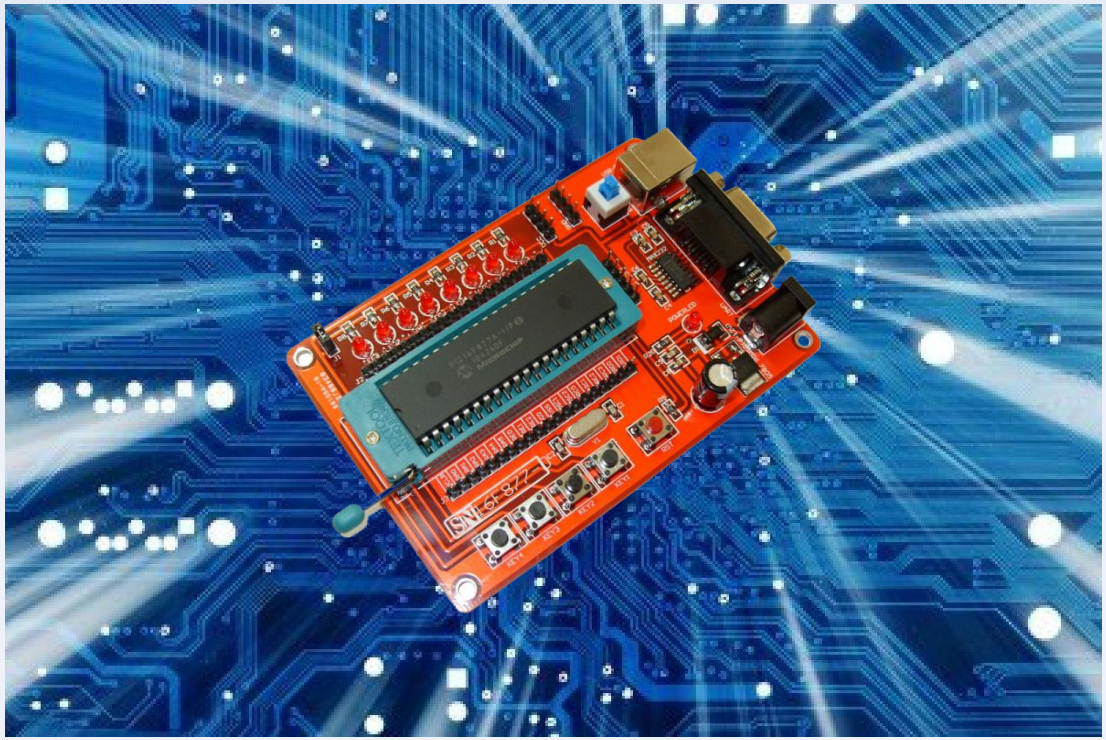
Informatique Industrielle – Niveau 1

Cours Licence 3 – Systèmes Embarqués



UNIVERSITÉ de Picardie Jules Verne

INSSET

INstitut Supérieur des Sciences Et Techniques SAINT-QUENTIN

Coordonnées

Cyrille DROMAS
Doctorant - CubeSat Picard (Contrôle d'attitude)
E-mail: cyrille.dromas+L3SET@gmail.com

INstitut Supérieur des Sciences et Techniques
Laboratoire des Technologies Innovantes, EA 3899
48 rue Raspail
CS 10422
02315 Saint-Quentin CEDEX
http://www.insset.u-picardie.fr

Organisation de l'enseignement

Contenu horaire :

● **6 heures** de Cours Magistraux + **6 heures** de Travaux Dirigés
- Présentation de l'informatique industrielle
- Présentation des systèmes micro-programmés (architecture, principes généraux)
- Étude d'un microcontrôleur PIC de Microchip
- Programmation en langage Assembleur

● **12 heures** de Travaux Pratiques (*3 x 4 heures*)
Mise en pratique des connaissances :
- Vérification des programmes sur le logiciel Proteus ISIS
- Implémentation sur le microcontrôleur Microchip PIC 18F67K22

Merci d'être à l'heure en cours et en TP !

Contrôle des connaissances

Note finale :

● **Travaux Pratiques - 10 points/20 :**

Les rapports de Travaux Pratiques seront notés sur plusieurs points :

- Clarté des rapports
- Rédaction d'un algorithme
- Les programmes écrits doivent être commentés
- Les programmes doivent être vérifié en simulation puis sur carte d'essai
- Les fonctionnalités doivent être validées par l'enseignant

● **Examen - 10 points/20 :**

Contrôle des connaissances avec documents, sans calculatrice, sur les notions vues en cours (TD inclus) ainsi qu'en TP.

La note finale sera déterminée de la manière suivante :

- 50% pour les TPs

- 50% pour l'examen

Sommaire

→ ***Présentation de l'informatique industrielle***
- Rappels sur les systèmes de numération
- Rappels sur la logique combinatoire et séquentielle
- Les différents systèmes programmables
- Architecture des microcontrôleurs
- Etude du microcontrôleur PIC 18F67K22
- Conception d'un système embarqué
- Programmation du PIC 18F67K22

L'informatique industrielle

« L'informatique industrielle est une branche technologique de l'informatique appliquée qui couvre l'ensemble des techniques de conception, d'analyse et de programmation de systèmes à base d'interfaçage de l'informatique avec de l'électronique, électrotechnique, mécanique, robotique, etc. à vocation industrielle, qui ne sont pas uniquement à base d'ordinateurs. »
(Source : Wikipédia)

Source: Distrimed

L'informatique industrielle

Domaines d'applications :

Alarme, automobile, aéronautique, ferroviaire, instrumentation, médical, téléphonie mobile, terminaux de paiement ...

L'informatique industrielle

Applications :

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

Source : Consomac

Source : MaterielMedical

Sommaire

- Présentation de l'informatique industrielle
→ ***Rappels sur les systèmes de numération***
- Rappels sur la logique combinatoire et séquentielle
- Les différents systèmes programmables
- Architecture des microcontrôleurs
- Etude du microcontrôleur PIC 18F67K22
- Conception d'un système embarqué
- Programmation du PIC 18F67K22

Les systèmes de numération

Binaire, octal, décimal et hexadécimal

Rappelons tout d'abord 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 notera avec en indice les lettres ‘b’, ‘o’, ‘d’ ou ‘h’ selon qu'il s'agira d'un codage binaire, octal, décimal ou hexadécimal. (Par exemple: 1001_b, $3F1_h$ ou 128_d)

Les systèmes de numération

Codes pondérés

Dans une base donnée, le nombre s'exprime comme une ***somme pondérée***.

Par exemple, le nombre 123 décimal (base 10) est constitué de 3 chiffres :

- Le chiffre 3 est affecté du poids de 1 (unités)
- Le chiffre 2 est affecté du poids de 10 (dizaines)
- Le chiffre 1 est affecté du poids de 100 (centaines)

Le nombre peut donc s'écrire :

$$1 \times 100 + 2 \times 10 + 3 \times 1 = 123_d$$

Chiffre *Poids*

Les systèmes de numération

Codes pondérés

Dans une base donnée, le nombre s'exprime comme une ***somme pondérée***.

Par exemple, le nombre 10 binaire (base 2) est constitué de 2 chiffres :

- Le chiffre 0 est affecté du poids de 1 (unités)
- Le chiffre 1 est affecté du poids de 10 (dizaines)

Le nombre peut donc s'écrire :

$$1 \times 10 + 0 \times 1 = 10_b$$

Chiffre *Poids*

Remarques :

- Le binaire est la base la plus simple : 2 chiffres (ou ***bits - binary digits***) « 0 » et « 1 ».
- Un ***byte*** (ou ***octet***) est une information de 8 bits.

Les systèmes de numération

Codes pondérés

Dans une base donnée, le nombre s'exprime comme une ***somme pondérée***.

Par exemple, le nombre 1AF hexadécimal (base 16) est constitué de 3 chiffres :

- Le chiffre F est affecté du poids de 1 (unités)
- Le chiffre A est affecté du poids de 10 (dizaines)
- Le chiffre 1 est affecté du poids de 100 (centaines)

Le nombre peut donc s'écrire :

$$1 \times 100 + A \times 10 + F \times 1 = 1AF_h$$

Chiffre *Poids*

Les systèmes de numération

Conversion

Conversion en décimal : *Développement en somme de puissances de la base.*

$$1 \quad 0 \quad 1 \quad 0_b$$

$$1 \times 2^3 + 0 \times 2^2 + 1 \times 2^1 + 0 \times 2^0 = 10_d$$

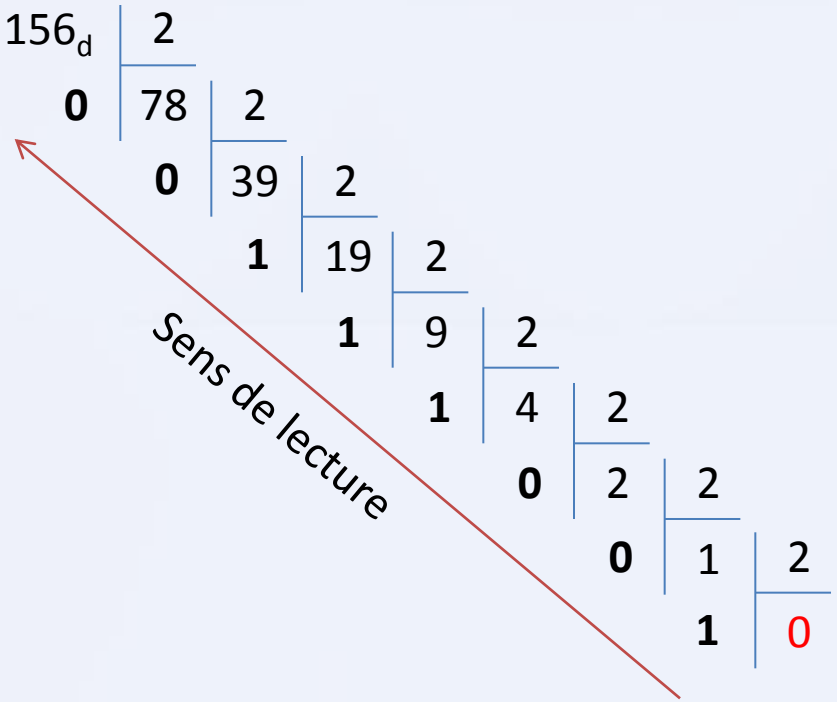
$$1 \quad A \quad F_h$$

$$1 \times 16^2 + 10 \times 16^1 + 15 \times 16^0 = 431_d$$

Les systèmes de numération

Conversion

Conversion décimal – binaire : *Divisions par 2 successives.*



$156_d = 1001\ 1100_b$

Les systèmes de numération

Conversion

Conversion décimal – hexadécimal : *Divisions par 16 successives.*

431_d	16		
F = 15	26	16	
A = 10	1	16	
	1	0	

Sens de lecture

$431_d = 1AF_h$

Les systèmes de numération

Conversion

Conversion binaire - hexadécimal :

Le codage hexadécimal a été créé afin de simplifier l'exploitation des nombres binaires. Il permet en particulier une conversion simple par regroupement des bits par 4 en partant de la droite. Chaque quartet (4 bits) étant alors simple à convertir.

Exemple : Conversion de « 0001 1010 1111_b » en hexadécimal

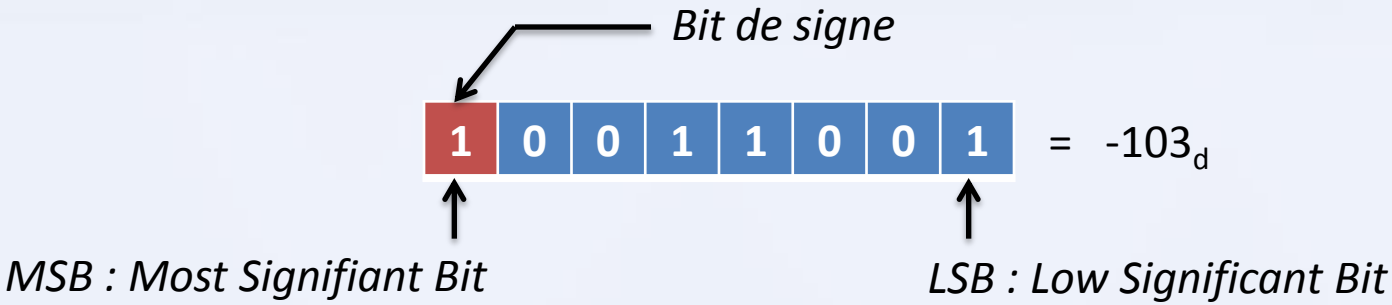
0001	1010	1111
1	A	F

$0001\ 1010\ 1111_b = 1AF_h$

Les systèmes de numération

Binaire signé

Dans certaines applications, il est nécessaire d'utiliser des nombres négatifs.



Pour rendre un nombre négatif, on procède en deux étapes :

1 – On inverse tout les bits du nombre

$103_d = 0110\ 0111_b \rightarrow 1001\ 1000_b$

2 – On ajoute 1

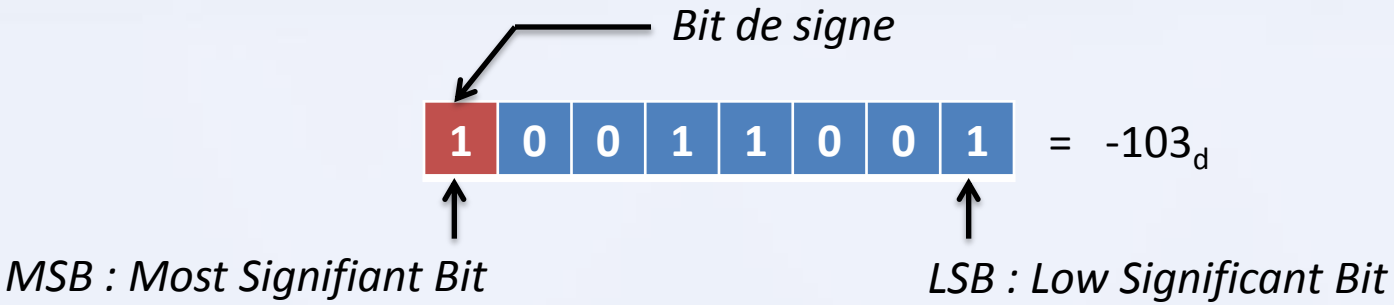
$1001\ 1000_b + 0000\ 0001_b = \mathbf{1001\ 1001_b}$

*Ce procédé se nomme le **complément à deux** du nombre*

Les systèmes de numération

Binaire signé

Dans certaines applications, il est nécessaire d'utiliser des nombres négatifs.



Pour faire la conversion inverse, on procède de la même manière :

1 – On inverse tout les bits du nombre

$-103_d = 1001\ 1001_b \rightarrow 0110\ 0110_b$

2 – On ajoute 1

$0110\ 0110_b + 0000\ 0001_b = \mathbf{0110\ 0111_b}$

Nous retrouvons bien notre 103_d de départ, ce qui est logique vu que $\mathbf{-(-103_d) = 103_d}$

Les systèmes de numération

Opérations arithmétiques binaires

Les techniques de calcul des opérations arithmétiques peuvent être transposées du décimal au binaire ainsi qu'à l'hexadécimal.

Addition : $V = A + B$

$$\begin{array}{r} {\scriptstyle 1\ \ 1\ \ \ \ \ \ \ \ } \\ 1001\ 0110_b \\ +\ 0101\ 0101_b \\ \hline = 1110\ 1011_b \end{array}$$

Les systèmes de numération

Opérations arithmétiques binaires

Les techniques de calcul des opérations arithmétiques peuvent être transposées du décimal au binaire ainsi qu'à l'hexadécimal.

Addition : $V = A + B$

$$\begin{array}{r} {\scriptstyle 1\ \ 1\ \ \ \ \ \ \ \ } \\ 1001\ 0110_b \\ +\ 0101\ 0101_b \\ \hline = 1110\ 1011_b \end{array}$$

Une addition de deux nombres de n bits donne un résultat sur n+1 bits !

$$\begin{array}{r} {\scriptstyle 1\ 1\ 1\ 1\ \ \ \ \ \ \ \ } \\ 1111_b \\ +\ \ 1111_b \\ \hline = 1\ 1110_b \end{array}$$

Les systèmes de numération

Opérations arithmétiques binaires

Les techniques de calcul des opérations arithmétiques peuvent être transposées du décimal au binaire ainsi qu'à l'hexadécimal.

Soustraction : $V = A - B = A + (-B)$

Pour calculer V, on calcule la somme entre A et le complément à deux de B

$1001\ 0110_b - 0101\ 0101_b$ ➔

$$\begin{array}{rr} & {\scriptstyle 1\quad 1\,1\,1\,1\,1} \\ & 1001\ 0110_b \\ + & 1010\ 1011_b \\ \hline = & 1\ 0100\ 0001_b \end{array}$$

➔

$$\begin{array}{rr} & 150_d \\ - & 85_d \\ \hline = & 65_d \end{array}$$

Les systèmes de numération

Opérations arithmétiques binaires

Les techniques de calcul des opérations arithmétiques peuvent être transposées du décimal au binaire ainsi qu'à l'hexadécimal.

Multiplication : $V = A \times B$

$$
\begin{array}{rl}
 & 1011_b \\
\times & 1010_b \\
\hline
 & {\scriptstyle 1}\ 1\,0110_b \\
+ & 101\ 1000_b \\
\hline
= & 110\ 1110_b
\end{array}
$$

Les systèmes de numération

Opérations arithmétiques binaires

Les techniques de calcul des opérations arithmétiques peuvent être transposées du décimal au binaire ainsi qu'à l'hexadécimal.

Multiplication : $V = A \times B$

$$
\begin{array}{rr}
 & 1011_b \\
\times & 1010_b \\
\hline
 & {\scriptstyle 1} \\
 & 1\,0110_b \\
+ & 101\,1000_b \\
\hline
= & 110\,1110_b
\end{array}
$$

Une multiplication de deux nombres de n bits donne un résultat sur 2 x n bits !

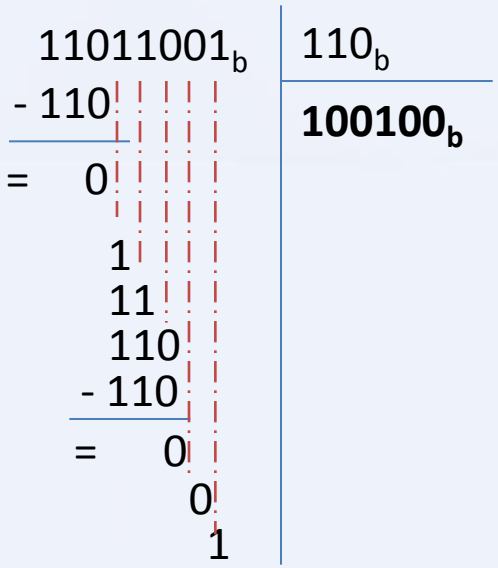
$$
\begin{array}{rr}
 & 11_b \\
\times & 11_b \\
\hline
 & 11_b \\
+ & 110_b \\
\hline
= & 1001_b
\end{array}
$$

Les systèmes de numération

Opérations arithmétiques binaires

Les techniques de calcul des opérations arithmétiques peuvent être transposées du décimal au binaire ainsi qu'à l'hexadécimal.

Division : *V = A / B*



Les systèmes de numération

Opérations arithmétiques binaires

Les techniques de calcul des opérations arithmétiques peuvent être transposées du décimal au binaire ainsi qu'à l'hexadécimal.

Multiplication / Division par 2_d :

Une multiplication par 2_d correspond à un décalage d'un bit vers la gauche :

$$\mathbf{1011\ 0010_b \times 10_b = 1\ 0110\ 0100_b}$$

Une division par 2_d correspond à un décalage d'un bit vers la droite :

$$\mathbf{1011\ 0010_b / 10_b = 0101\ 1001_b}$$

TD1 :
Les systèmes de numération

Sommaire

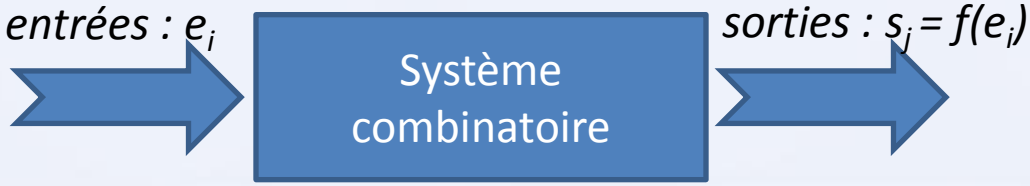
- Présentation de l'informatique industrielle
- Rappels sur les systèmes de numération
→ ***Rappels sur la logique combinatoire et séquentielle***
- Les différents systèmes programmables
- Architecture des microcontrôleurs
- Etude du microcontrôleur PIC 18F67K22
- Conception d'un système embarqué
- Programmation du PIC 18F67K22

Logique combinatoire et séquentielle

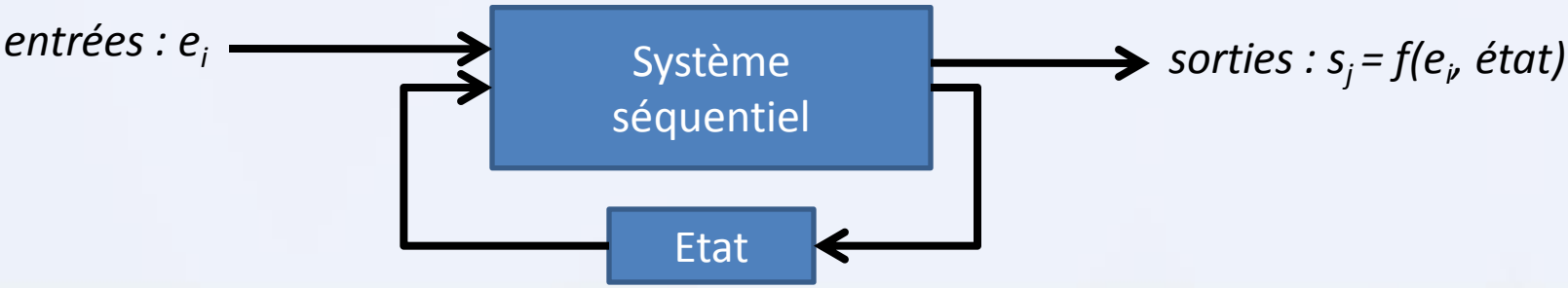
Définitions

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.



Logique combinatoire et séquentielle

Table de vérité

Définition :

Une **table de vérité** est un tableau définissant la valeur d'une fonction logique pour chacune des combinaisons possibles des entrées.

→ Une fonction logique de n variables admet 2^n combinaisons d'entrées possibles.
→ La table de vérité de cette fonction comporte donc 2^n lignes.

E_1	E_2	S
0	0	**X**
0	1	**X**
1	0	**X**
1	1	**X**

Notation :

→ Les états VRAI et FAUX sont codés respectivement 1 et 0.
→ La variable NON x est notée $\overline{x}$ et ce lit « x barre ».

Logique combinatoire et séquentielle

Opérateurs élémentaires

Parmi toutes les configurations possibles, on extrait typiquement 6 fonctions logiques d'intérêt que sont les opérateurs NON (une entrée) et ET, OU, ET-NON, OU-NON,et OU-EXCLUSIF (deux entrées).

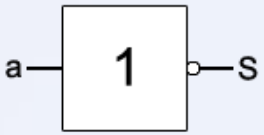
NON

Equation logique : $S = \overline{a}$

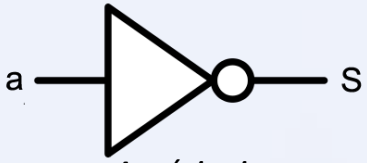
Table de vérité:

a	S
0	**1**
1	**0**

Symboles :



NFC03-212



Américain

Logique combinatoire et séquentielle

Opérateurs élémentaires

Parmi toutes les configurations possibles, on extrait typiquement 6 fonctions logiques d'intérêt que sont les opérateurs NON (une entrée) et ET, OU, ET-NON, OU-NON,et OU-EXCLUSIF (deux entrées).

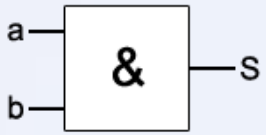
ET

Equation logique : S = a . b

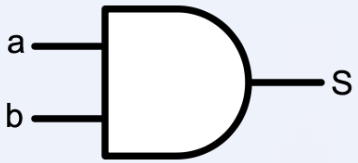
Table de vérité:

a	b	S
0	0	**0**
0	1	**0**
1	0	**0**
1	1	**1**

Symboles :



NFC03-212



Américain

Logique combinatoire et séquentielle

Opérateurs élémentaires

Parmi toutes les configurations possibles, on extrait typiquement 6 fonctions logiques d'intérêt que sont les opérateurs NON (une entrée) et ET, OU, ET-NON, OU-NON,et OU-EXCLUSIF (deux entrées).

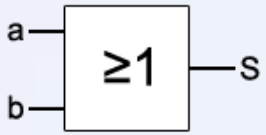
OU

Equation logique : S = a + b

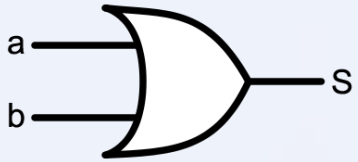
Table de vérité:

a	b	S
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**1**

Symboles :



NFC03-212



Américain

Logique combinatoire et séquentielle

Opérateurs élémentaires

Parmi toutes les configurations possibles, on extrait typiquement 6 fonctions logiques d'intérêt que sont les opérateurs NON (une entrée) et ET, OU, ET-NON, OU-NON,et OU-EXCLUSIF (deux entrées).

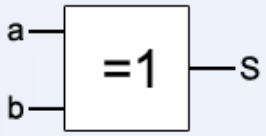
OU EXCLUSIF (XOR)

Equation logique : $S = a \oplus b = a.\overline{b} + \overline{a}.b$

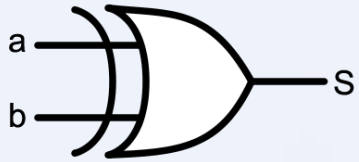
Table de vérité:

a	b	S
0	0	**0**
0	1	**1**
1	0	**1**
1	1	**0**

Symboles :



NFC03-212



Américain

Logique combinatoire et séquentielle

Opérateurs élémentaires

Parmi toutes les configurations possibles, on extrait typiquement 6 fonctions logiques d'intérêt que sont les opérateurs NON (une entrée) et ET, OU, ET-NON, OU-NON,et OU-EXCLUSIF (deux entrées).

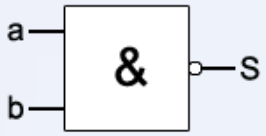
NON-ET (NAND)

Equation logique : $S = \overline{a \cdot b}$

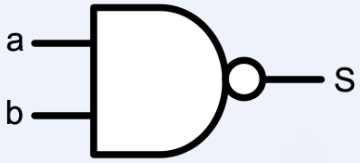
Table de vérité:

a	b	S
0	0	**1**
0	1	**1**
1	0	**1**
1	1	**0**

Symboles :



NFC03-212



Américain

Logique combinatoire et séquentielle

Opérateurs élémentaires

Parmi toutes les configurations possibles, on extrait typiquement 6 fonctions logiques d'intérêt que sont les opérateurs NON (une entrée) et ET, OU, ET-NON, OU-NON,et OU-EXCLUSIF (deux entrées).

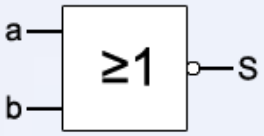
NON-OU (NOR)

Equation logique : $S = \overline{a + b}$

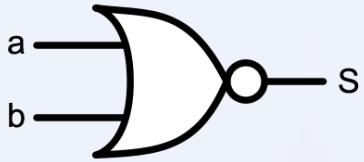
Table de vérité:

a	b	S
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**0**

Symboles :



NFC03-212



Américain

Logique combinatoire et séquentielle

Opérateurs élémentaires

Parmi toutes les configurations possibles, on extrait typiquement 6 fonctions logiques d'intérêt que sont les opérateurs NON (une entrée) et ET, OU, ET-NON, OU-NON,et OU-EXCLUSIF (deux entrées).

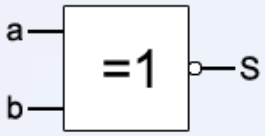
NON-OU EXCLUSIF (XNOR)

Equation logique : $S = \overline{a \oplus b}$

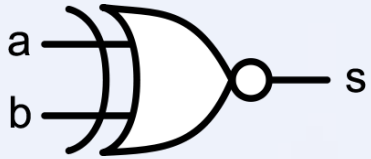
Table de vérité:

a	b	S
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

Symboles :



NFC03-212



Américain

Logique combinatoire et séquentielle

Opérateurs élémentaires

Parmi toutes les configurations possibles, on extrait typiquement 6 fonctions logiques d'intérêt que sont les opérateurs NON (une entrée) et ET, OU, ET-NON, OU-NON,et OU-EXCLUSIF (deux entrées).

Les opérateurs NON-ET et NON-OU 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.

Logique combinatoire et séquentielle

Algèbre de BOOLE

Les opérateurs logiques élémentaires permettent la construction d'une algèbre dite « 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	$\overline{A}$	NON-ET	$\overline{A \cdot B} = \overline{A} + \overline{B}$
ET	$A \cdot B$	NON-OU	$\overline{A + B} = \overline{A} \cdot \overline{B}$
OU	$A + B$	OU-EXCLUSIF	$A \oplus B$

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 :

$A \cdot B = B \cdot A$

$A + B = B + A$

$A \oplus B = B \oplus A$

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

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

Distributivité :

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

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

Lois de De Morgan :

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

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

Logique combinatoire et séquentielle

Algèbre de BOOLE

Les opérateurs logiques élémentaires permettent la construction d'une algèbre dite « 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	$\overline{A}$	NON-ET	$\overline{A \cdot B} = \overline{A} + \overline{B}$
ET	$A \cdot B$	NON-OU	$\overline{A + B} = \overline{A} \cdot \overline{B}$
OU	$A + B$	OU-EXCLUSIF	$A \oplus B$

Les différentes opérations bénéficient des propriétés suivantes :

Eléments neutres :

$A + 0 = A$

$A \cdot 1 = A$

Complémentation :

$A + \overline{A} = 1$

$A \cdot \overline{A} = 0$

Eléments absorbants :

$A \cdot 0 = 0$

$A + 1 = 1$

Idempotence :

$A + A = A$

$A \cdot A = A$

Logique combinatoire et séquentielle

Circuits intégrés logiques

En électronique, des C.I. spécialisés permettent de réaliser les fonctions logiques.
Il existe deux grandes familles de C.I. logiques : **TTL** et **CMOS**.

Famille TTL (Transistor-Transistor Logic)

Cette famille de C.I. est fabriquée avec des transistors bipolaires.

<u>En logique TTL-standard :</u>
Tension d'alimentation : Vcc = 5V (± 0,25V)
En entrée :

- 0 à 0,8 V : niveau logique 0
- 2 à 5 V : niveau logique 1

En sortie :

- 0 à 0,4 V : niveau logique 0
- 2,4 à 5 V : niveau logique 1

Logique combinatoire et séquentielle

Circuits intégrés logiques

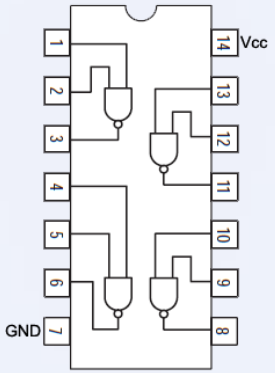
En électronique, des C.I. spécialisés permettent de réaliser les fonctions logiques.
Il existe deux grandes familles de C.I. logiques : **TTL** et **CMOS**.

Famille TTL (Transistor-Transistor Logic)

Cette famille de C.I. est fabriquée avec des transistors bipolaires.

Exemple : Circuit intégré 7400

Ce circuit dispose de quatre fonctions (ou portes) logiques NON-ET (NAND) à 2 entrées.



Brochage du 7400

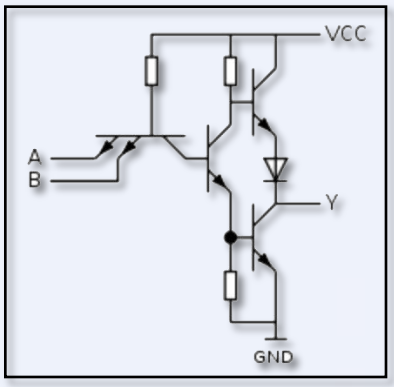


Schéma interne d'une porte NAND

Logique combinatoire et séquentielle

Circuits intégrés logiques

En électronique, des C.I. spécialisés permettent de réaliser les fonctions logiques.
Il existe deux grandes familles de C.I. logiques : **TTL** et **CMOS**.

Famille CMOS (Complementary Metal Oxide Semiconductor)

Cette famille de C.I. est fabriquée avec des transistors MOSFET.

<u>En logique CMOS-standard :</u>
Tension d'alimentation : Vcc = 3V à 18V
En entrée :

- 0.55 x Vcc à Vcc : niveau logique 0
- 0 à 0,45 x Vcc : niveau logique 1

En sortie :

- 0 à 0,05 x Vcc : niveau logique 0
- 0,95 x Vcc à Vcc : niveau logique 1

Logique combinatoire et séquentielle

Circuits intégrés logiques

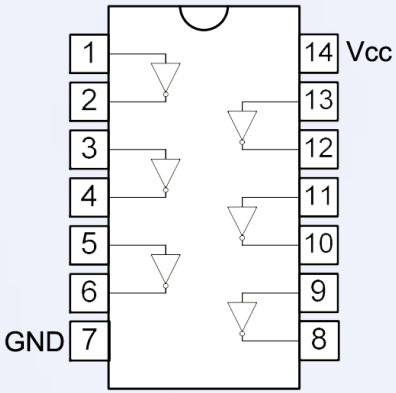
En électronique, des C.I. spécialisés permettent de réaliser les fonctions logiques.
Il existe deux grandes familles de C.I. logiques : **TTL** et **CMOS**.

Famille CMOS (Complementary Metal Oxide Semiconductor)

Cette famille de C.I. est fabriquée avec des transistors MOSFET.

Exemple : Circuit intégré 4069B

Ce circuit contient six portes inverseuses NON



Brochage du 4069B

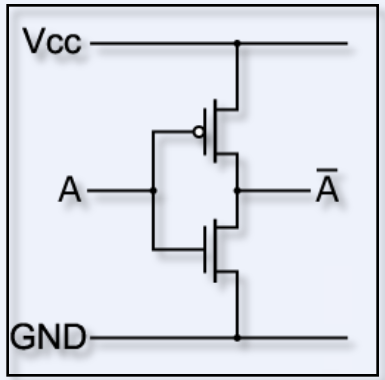


Schéma interne d'une porte NON

Logique combinatoire et séquentielle

Circuit combinatoire

Un circuit combinatoire est un circuit logique où chacune des sorties est une fonction logique des entrées.

Exemple : Synthèse d'un circuit combinatoire

On désire avoir un circuit logique à trois entrées et deux sorties dont la table de vérité est :

A	B	C	S_1	S_2
0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	0

Logique combinatoire et séquentielle

Circuit combinatoire

Un circuit combinatoire est un circuit logique où chacune des sorties est une fonction logique des entrées.

Exemple : Synthèse d'un circuit combinatoire

A	B	C	S_1	S_2
0	0	0	1	0
0	0	1	0	0
0	1	0	0	0
0	1	1	0	0
1	0	0	1	1
1	0	1	0	1
1	1	0	0	0
1	1	1	1	0

Equations logiques :

S_2 = 1 si (A = 1 et B = 0 et C = 0) ou (A = 1 et B = 0 et C = 1)

$S_2 = (A \cdot \overline{B} \cdot \overline{C}) + (A \cdot \overline{B} \cdot C)$

$S_1 = \overline{A}\,\overline{B}\,\overline{C} + A\,\overline{B}\,\overline{C} + A\,B\,C$

Simplification des équations logiques :

$S_2 = A\,\overline{B}\,\overline{C} + A\,\overline{B}\,C = A\overline{B}\,(\overline{C} + C) = A\overline{B}$

➔ S_2 = 1 si (A = 1 et B = 0)

$S_1 = (\overline{A} + A)\,\overline{B}\,\overline{C} + A\,B\,C = \overline{B}\,\overline{C} + A\,B\,C$

Logique combinatoire et séquentielle

Circuit combinatoire

Un circuit combinatoire est un circuit logique où chacune des sorties est une fonction logique des entrées.

Exemple : Synthèse d'un circuit combinatoire

Il est plus efficace d'utiliser la technique des « **tableaux de Karnaugh** » pour simplifier les équations logiques correspondant à une table de vérité :

S_1 :

A \ BC	00	01	11	10
0	1	0	0	0
1	1	0	1	0

$S_1 = \overline{B} \cdot \overline{C} + A \cdot B \cdot C$

S_2 :

A \ BC	00	01	11	10
0	0	0	0	0
1	1	1	0	0

$S_2 = A \cdot \overline{B}$

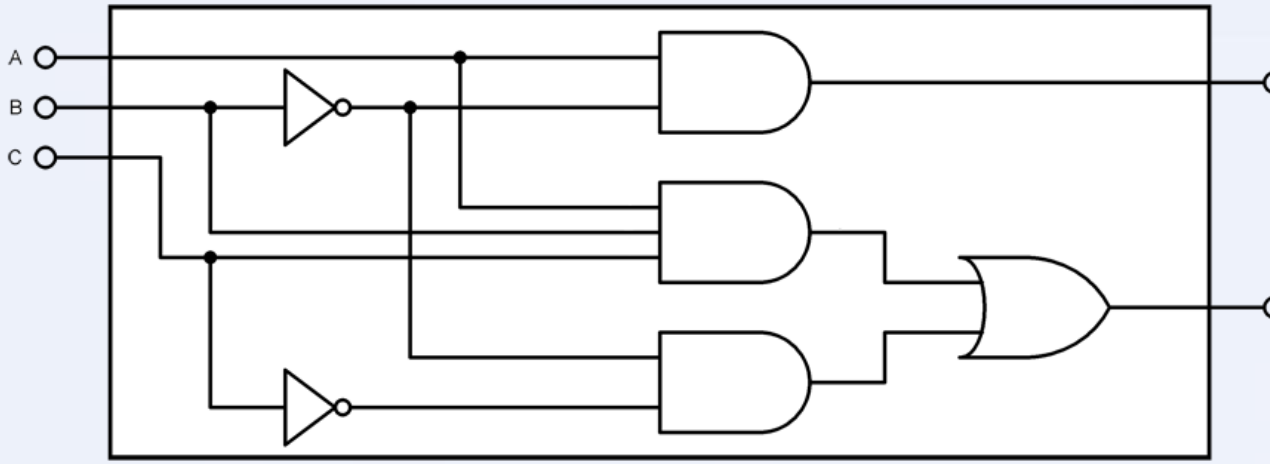
Logique combinatoire et séquentielle

Circuit combinatoire

Un circuit combinatoire est un circuit logique où chacune des sorties est une fonction logique des entrées.

Exemple : Synthèse d'un circuit combinatoire

Logigramme:



$S_2 = A \cdot \overline{B}$

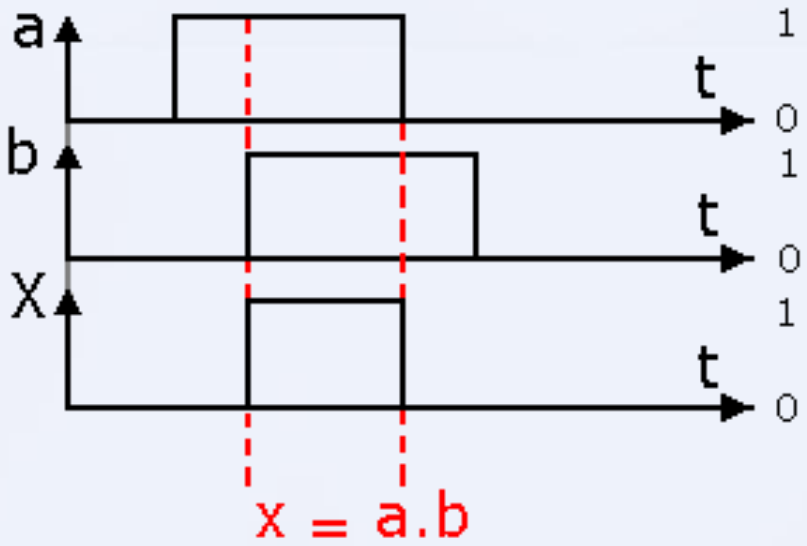
$S_1 = \overline{B} \cdot \overline{C} + A \cdot B \cdot C$

Logique combinatoire et séquentielle

Chronogramme

Le **chronogramme** est une représentation graphique de l'évolution temporelle d'un signal électrique ou d'un état.

Dans les microcontrôleurs, les états du système changent en fonction d'une base de temps qui est l'horloge. Ceci conduit naturellement à introduire les chronogrammes comme outil d'analyse des états logiques d'un système.



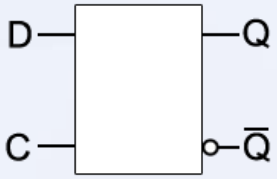
Logique combinatoire et séquentielle

Bascules asynchrones

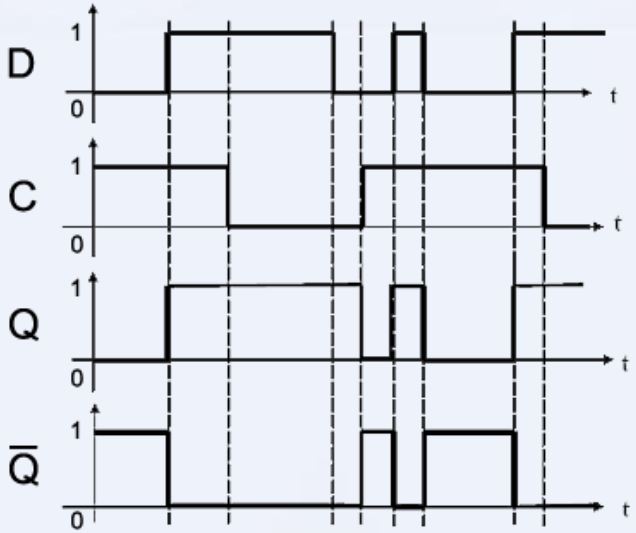
Une **bascule asynchrone** est une fonction « mémoire » qui est commandée. Ce type de fonction est notamment utilisé pour créer des registres du microcontrôleur.

Le verrou D (D Latch) :

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	Q_{n+1}	$\overline{Q_{n+1}}$
0	X	Q_n	$\overline{Q_n}$
1	0	0	1
1	1	1	0



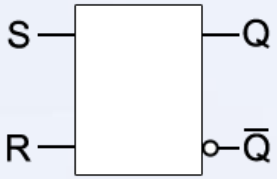
Logique combinatoire et séquentielle

Bascules asynchrones

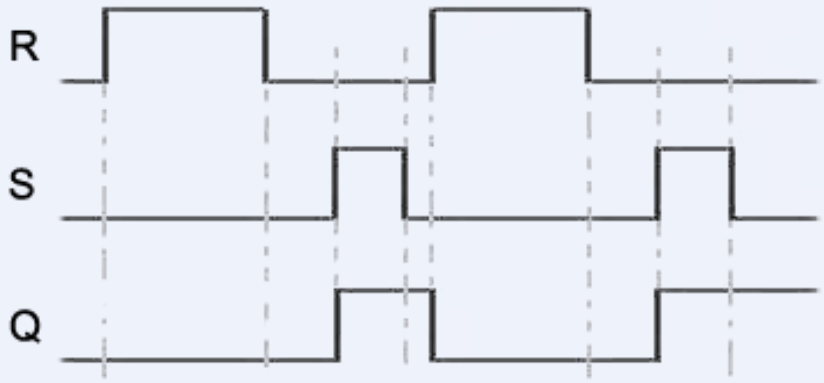
Une **bascule asynchrone** est une fonction « mémoire » qui est commandée. Ce type de fonction est notamment utilisé pour créer des registres du microcontrôleur.

Le verrou RS (RS Latch) :

Ce type de verrou peut être mis à « 0 » (Reset) ou à « 1 » (Set).
Si S et R sont à « 0 », la sortie précédente est mémorisée.



Entrées		Sorties	
S	**R**	**Q_{n+1}**	**Remarque**
0	0	Q_n	Mémorisation
0	1	0	Mise à zéro
1	0	1	Mise à un
1	1	X	Interdit !



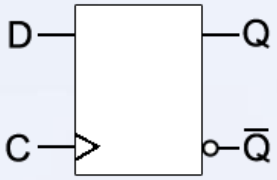
Logique combinatoire et séquentielle

Bascules synchrones

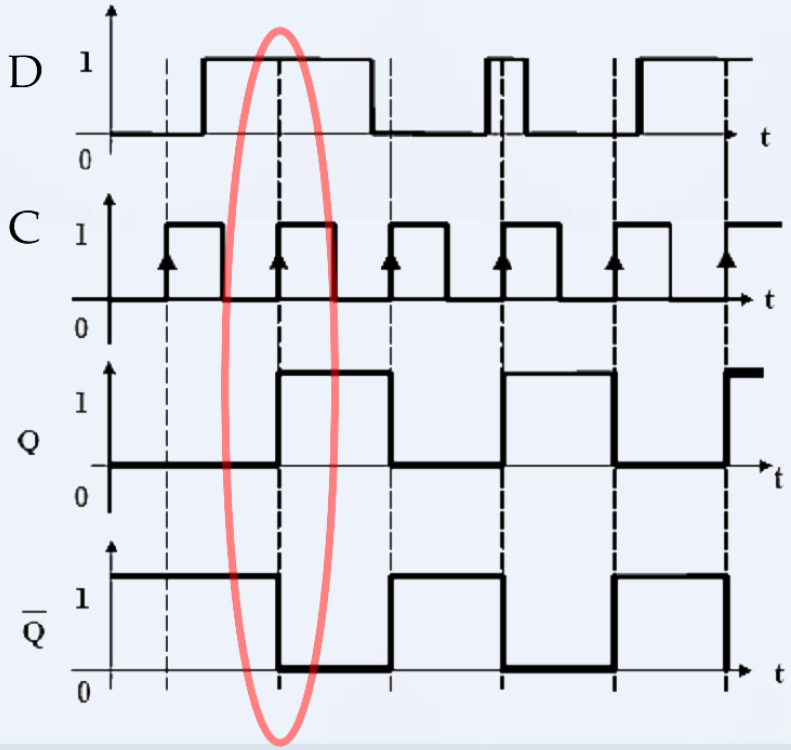
Une **bascule synchrone** est une bascule qui ne change d'état que sur front montant ou descendant appliqué sur son entrée de commande. Ce type de bascule est à la base du fonctionnement du microcontrôleur.

La bascule D (Flip-Flop D) :

C'est la version synchrone du verrou D !



Entrées		Sorties	
C	D	Q_{n+1}	$\overline{Q_{n+1}}$
0	X	Q_n	$\overline{Q_n}$
↑	0	0	1
↑	1	1	0



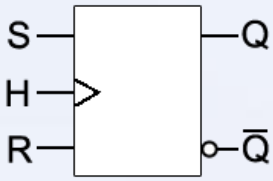
Logique combinatoire et séquentielle

Bascules synchrones

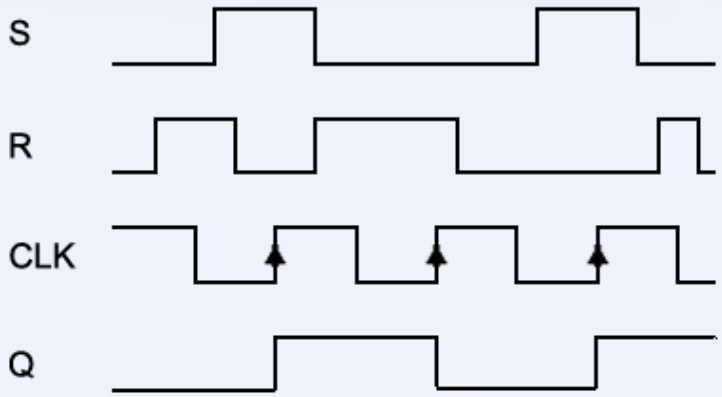
Une **bascule synchrone** est une bascule qui ne change d'état que sur front montant ou descendant appliqué sur son entrée de commande. Ce type de bascule est à la base du fonctionnement du microcontrôleur.

La bascule RSH (Flip-flop RSH) :

Le verrou RSH est un verrou RS possédant une troisième entrée : H (horloge).



Entrées			Sorties	
S	R	H	Q_{n+1}	Remarque
X	X	X	Q_n	Mémorisation
0	0	↑	Q_n	Mémorisation
0	1	↑	0	Mise à zéro
1	0	↑	1	Mise à un
1	1	↑	X	Interdit !

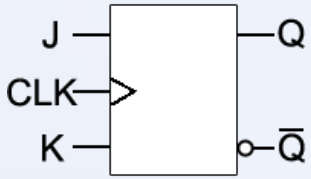


Logique combinatoire et séquentielle

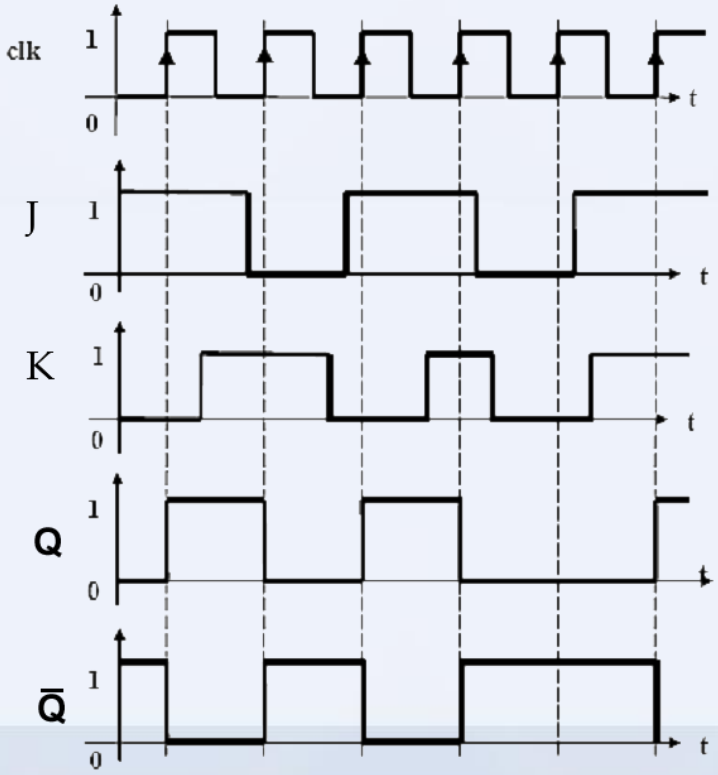
Bascules synchrones

Une **bascule synchrone** est une bascule qui ne change d'état que sur front montant ou descendant appliqué sur son entrée de commande. Ce type de bascule est à la base du fonctionnement du microcontrôleur.

La bascule JK (Jump Knock-out) :



Entrées			Sorties	
CLK	J	K	Q_{n+1}	$\overline{Q_{n+1}}$
0	X	X	Q_n	$\overline{Q_n}$
1	X	X	Q_n	$\overline{Q_n}$
↓	X	X	Q_n	$\overline{Q_n}$
↑	0	0	Q_n	$\overline{Q_n}$
↑	0	1	0	1
↑	1	0	1	0
↑	1	1	$\overline{Q_n}$	Q_n



Logique combinatoire et séquentielle

Utilité des C.I. logiques et bascules en informatique industrielle

➢ **Circuits arithmétiques :**

A partir de fonctions logiques, on peut créer les fonctions arithmétiques : addition, soustraction, multiplication, division, comparaison …

Exemple 1 : Comparateur 1 bit

Ce circuit doit réaliser la comparaison de deux nombres binaires de un bit : A, B.
Le résultat de la comparaison est donné par l'état de la sortie :

- S = 1 si A = B
- S = 0 si A ≠ B

Exercice :

1 - Donner la table de vérité du comparateur ci-dessus
2 - En déduire les équation booléenne des sorties
3 - Dessiner le logigramme correspondant

Logique combinatoire et séquentielle

Utilité des C.I. logiques et bascules en informatique industrielle

➢ **Circuits arithmétiques :**

A partir de fonctions logiques, on peut créer les fonctions arithmétiques : addition, soustraction, multiplication, division, comparaison …

Exemple 1 : Comparateur 1 bit

Ce circuit doit réaliser la comparaison de deux nombres binaires de un bit : A, B.
Le résultat de la comparaison est donné par l'état de la sortie :

- S = 1 si A = B
- S = 0 si A ≠ B

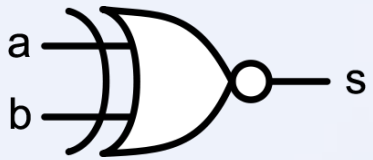
Table de vérité :

A	B	S
0	0	**1**
0	1	**0**
1	0	**0**
1	1	**1**

Equation booléenne des sorties :

$S = \overline{A} \cdot \overline{B} + A \cdot B = \overline{A \oplus B}$ ➔ NON-OU EXCLUSIF

Logigramme :



Logique combinatoire et séquentielle

Utilité des C.I. logiques et bascules en informatique industrielle

➢ **Circuits arithmétiques :**

A partir de fonctions logiques, on peut créer les fonctions arithmétiques : addition, soustraction, multiplication, division, comparaison …

Exemple 2 : Additionneur 1 bit

Ce circuit doit réaliser l'addition arithmétique de deux nombres binaires de un bit : A, B.

Le résultat de l'addition nécessite un nombre de deux bits (S0, S1)

Exercice :

1 - Donner la table de vérité de l'additionneur ci-dessus

2 - En déduire les équation booléenne des sorties

3 - Dessiner le logigramme correspondant

Logique combinatoire et séquentielle

Utilité des C.I. logiques et bascules en informatique industrielle

➢ **Circuits arithmétiques :**

A partir de fonctions logiques, on peut créer les fonctions arithmétiques : addition, soustraction, multiplication, division, comparaison …

Exemple 2 : Additionneur 1 bit

Ce circuit doit réaliser l'addition arithmétique de deux nombres binaires de un bit : A, B.

Le résultat de l'addition nécessite un nombre de deux bits (S0, S1)

Table de vérité :

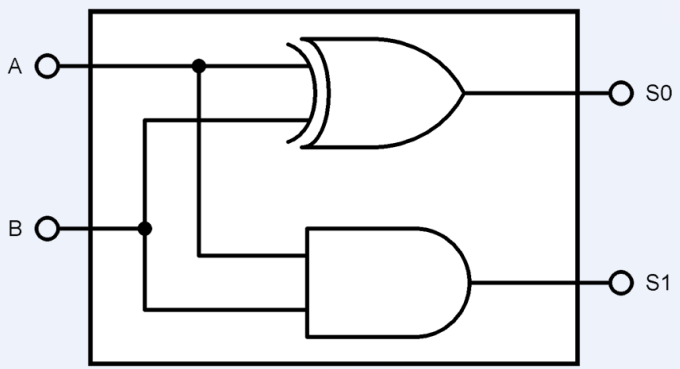
A	B	S_1 MSB	S_0 LSB
0	0	**0**	**0**
0	1	**0**	**1**
1	0	**0**	**1**
1	1	**1**	**0**

Equation booléenne des sorties :

$S_0 = A \oplus B$

$S_1 = A \cdot B$

Logigramme :



Logique combinatoire et séquentielle

Utilité des C.I. logiques et bascules en informatique industrielle

➢ **Circuits de multiplexage :**

Un multiplexeur (*MUX*) est un circuit permettant de concentrer différentes entrées sur une seule sortie.

Exercice :

1 - Donner la table de vérité du multiplexeur ci-dessus

2 - En déduire les équation booléenne des sorties

3 - Tracer le logigramme correspondant

Logique combinatoire et séquentielle

Utilité des C.I. logiques et bascules en informatique industrielle

➢ **Circuits de multiplexage :**

Un multiplexeur (*MUX*) est un circuit permettant de concentrer différentes entrées sur une seule sortie.

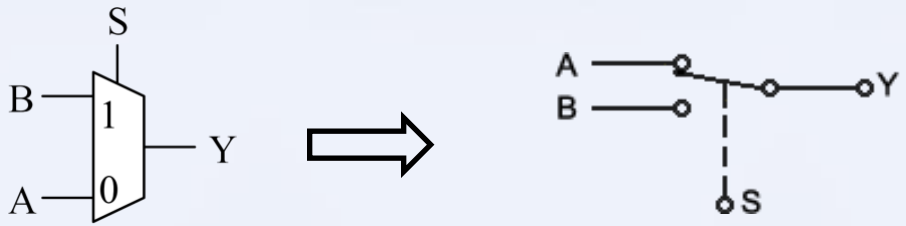


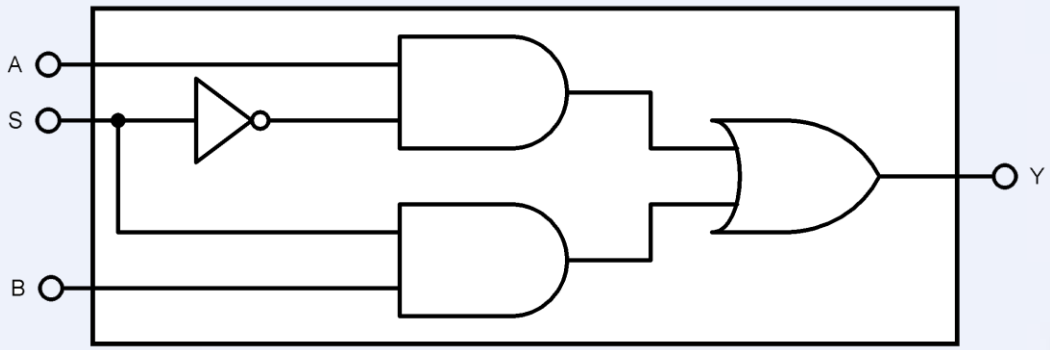
Table de vérité :

A	B	S	Y
0	0	0	0
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

Equation booléenne des sorties :

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

Logigramme :



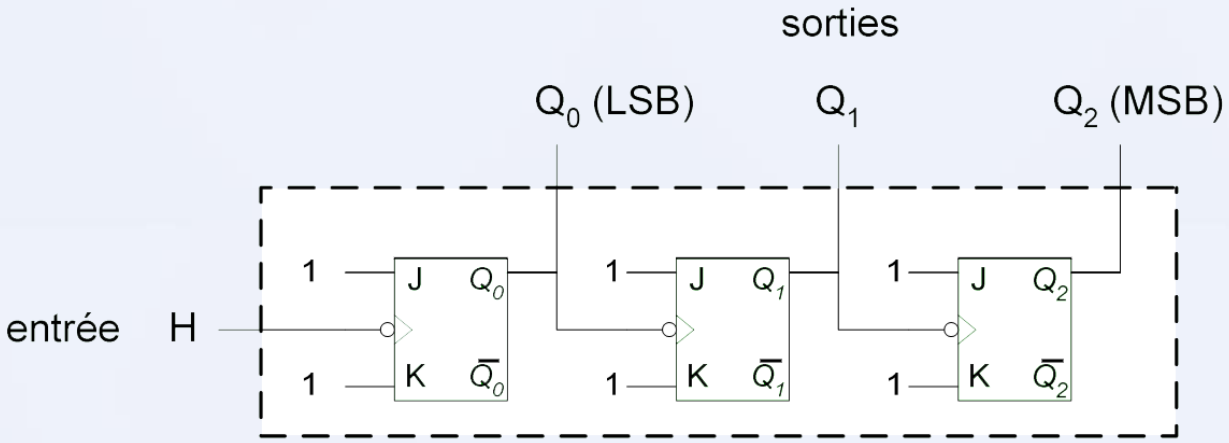
Logique combinatoire et séquentielle

Utilité des C.I. logiques et bascules en informatique industrielle

➢ **Fonction comptage : les compteurs numériques:**

Le comptage nécessite la fonction mémoire : on se sert donc de bascules.

Exemple : Compteur binaire asynchrone 3 bits



Ce circuit présente une entrée H et trois sorties.

Les sorties forment un nombre de 3 bits $(Q_2Q_1Q_0)_2$ qui donne le résultat du comptage.

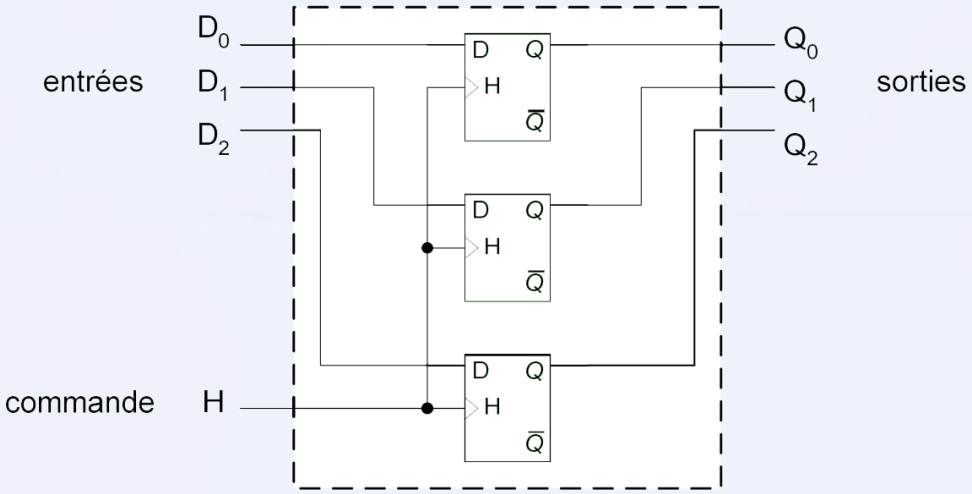
Logique combinatoire et séquentielle

Utilité des C.I. logiques et bascules en informatique industrielle

➢ **Registre mémoire :**

Les registres mémoire sont créé à partir de bascules.

Exemple : Registre mémoire 3 bits



Quand l'horloge devient active, les trois bits présents en entrée sont transférés en sortie. Les trois bits restent ensuite mémorisés en sortie aussi longtemps que l'horloge est inactive.

Logique combinatoire et séquentielle

Utilité des C.I. logiques et bascules en informatique industrielle

➢ **Registre mémoire :**

Les registres mémoire sont créé à partir de bascules.

Applications :

- Registres de microprocesseurs (8, 16, 32 bits)
- Mémoire SRAM : mémoire cache des ordinateurs …

Remarque : Taille mémoire

1 octet = 1 byte = 8 bits

1 « ko » = 2^{10} = 1024 octets

1 « Mo » = 1024 ko = 1 048 576 octets

1 « Go » = 1024 Mo

Une mémoire SRAM de 256ko nécessite donc :

256 × 1024 × 8 = 2 097 152 bascules

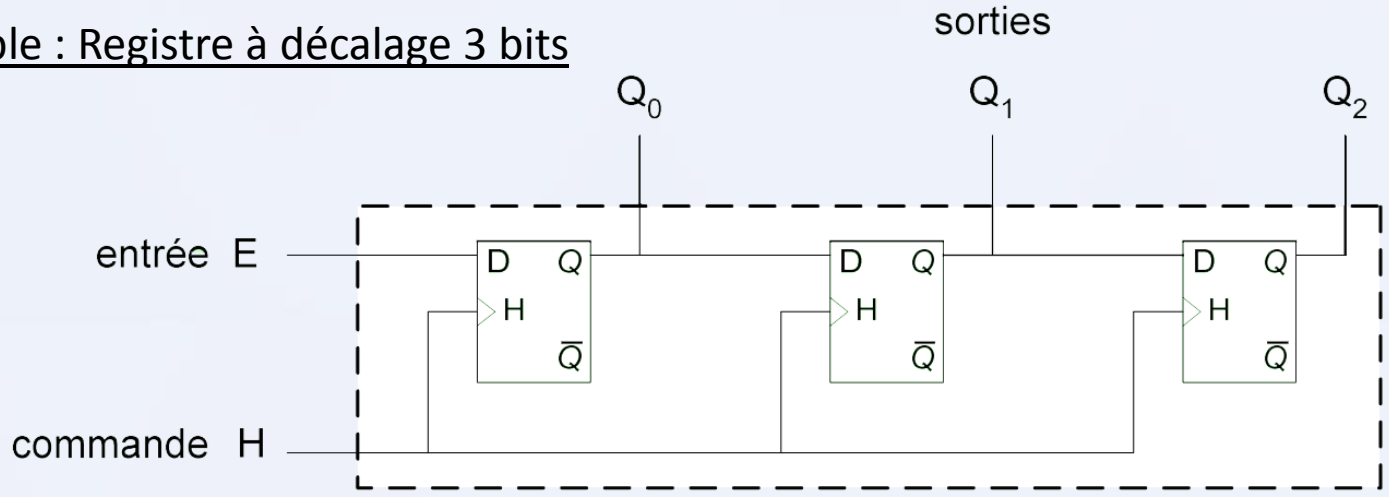
Logique combinatoire et séquentielle

Utilité des C.I. logiques et bascules en informatique industrielle

➢ **Registre à décalage :**

Les registres mémoire sont créé à partir de bascules.

Exemple : Registre à décalage 3 bits



A chaque front d'horloge, Q_0 prend la valeur de l'entrée E, Q_1 la valeur de Q_0 et Q_2 la valeur de Q_1.

Une application importante des registres à décalage est la transmission série de données logiques (RS232, CAN, SPI, USB …).

TD2 :
Logique combinatoire et séquentielle

Sommaire

- Présentation de l'informatique industrielle
- Rappels sur les systèmes de numération
- Rappels sur la logique combinatoire et séquentielle
→ ***Les différents systèmes programmables***
- Architecture des microcontrôleurs
- Etude du microcontrôleur PIC 18F67K22
- Conception d'un système embarqué
- Programmation du PIC 18F67K22

Les différents systèmes programmables

Circuits spécialisés ou ASIC (Application Specific Integrated Circuit)

Les circuits spécialisés sont des circuits spécialisés dès leur conception pour une application donnée.

Exemples : DSP (Digital Signal Processing), coprocesseur arithmétique, processeur 3D, contrôleur de bus, ...

Source : nVidia

Source : Astrium

Avantages	Inconvénients
• Très rapide • Consommation moindre • Optimisé pour une application	• Faible modularité • Possibilité d'évolution limité • Coût élevé

Les différents systèmes programmables

Systèmes en logique programmée et/ou en logique programmable

Exemples : FPGA (field-programmable gate array), PAL (programmable array logic), ...

Source : Xillinx

Source : Altera

« Un circuit logique programmable, ou réseau logique programmable, est un circuit intégré logique qui peut être reprogrammé après sa fabrication. Il est composé de nombreuses cellules logiques élémentaires pouvant être librement assemblées.» (Wikipédia)

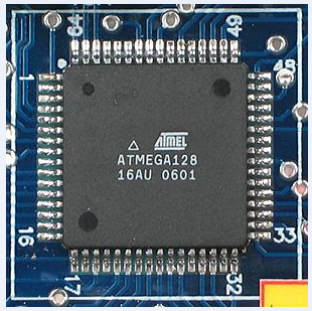
Avantages	Inconvénients
• Forte modularité • Rapidité	• Mise en œuvre plus complexe • Coût de développement élevé

Les différents systèmes programmables

Systèmes micro-programmés

Les microcontrôleurs sont typiquement des systèmes micro-programmés.

Source : Microchip



Source : Atmel

« Un microcontrôleur est un circuit intégré qui rassemble les éléments essentiels d'un ordinateur : processeur, mémoires (mémoire morte pour le programme, mémoire vive pour les données), unités périphériques et interfaces d'entrées-sorties.» (Wikipédia)

Avantages	Inconvénients
• Mise en œuvre simple • Coûts de développement réduits	• Plus lent • Utilisation sous optimale

Sommaire

- Présentation de l'informatique industrielle
- Rappels sur les systèmes de numération
- Rappels sur la logique combinatoire et séquentielle
- Les différents systèmes programmables

→ ***Architecture des microcontrôleurs***

- Etude du microcontrôleur PIC 18F67K22
- Conception d'un système embarqué
- Programmation du PIC 18F67K22

Architecture des microcontrôleurs

Les deux types de processeur

● **CISC :** *Complex Instruction Set Computer*
- Grand nombre d'instructions
- Type de processeur le plus répandu

● **RISC :** *Reduced Instruction Set Computer*
- Nombre d'instructions réduit (sélection des instructions pour une exécution plus rapide)
- Décodage des instructions plus rapide

Architecture des microcontrôleurs

Les différents bus d'un système micro-programmés

« Un bus est un jeu de lignes partagées pour l'échange de mots numériques.»
(Traité de l'électronique, Paul Horowitz & Winfield Hill)

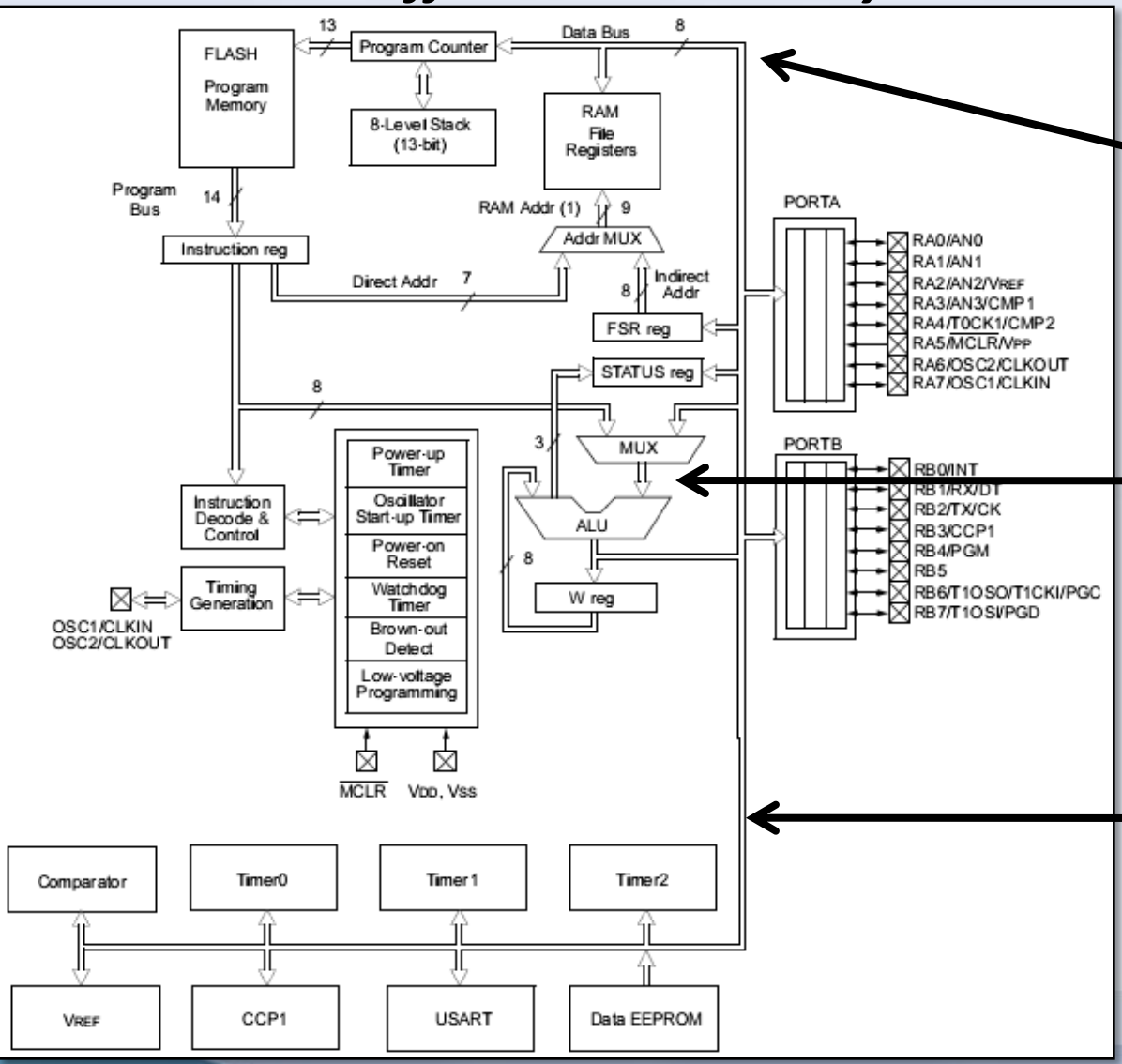
Définition : Un **bus** permet de faire transiter (liaison série / parallèle) des informations codées en binaire entre deux points. Typiquement les informations sont regroupés en mots : octet (8 bits), word (16 bits) ou double word (32 bits).

Caractéristiques d'un bus :

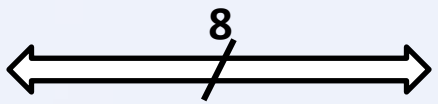
- Largeur du bus (nombre de lignes)
- Fréquence de transfert

Architecture des microcontrôleurs

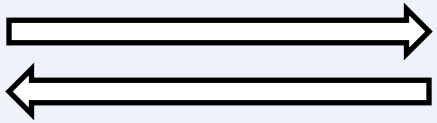
Les différents bus d'un système micro-programmés



« Largeur du bus »



« Unidirectionnel »



« Bidirectionnel »

Architecture des microcontrôleurs

Les différents bus d'un système micro-programmés

Il existe 3 types de bus :

- ***Bus de données :*** Permet de transférer des données entre composants,
  ex.: résultat d'une opération, valeur d'une variable, etc.

- ***Bus d'adresses :*** Permet de transférer des adresses entre composants,
  ex.: adresse d'une case mémoire, etc.

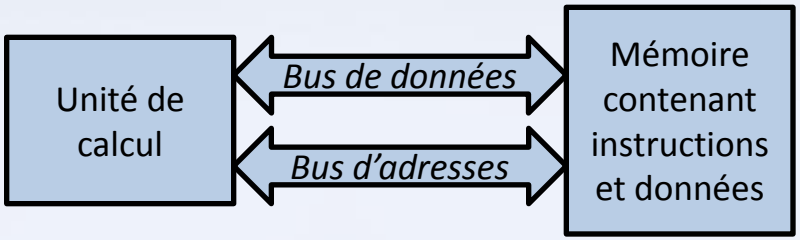
- ***Bus de contrôle :*** Permet l'échange entre les composants d'informations de contrôle [bus rarement représenté sur les schémas].
  ex.: périphérique prêt / occupé, erreur / exécution réussie, etc.

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

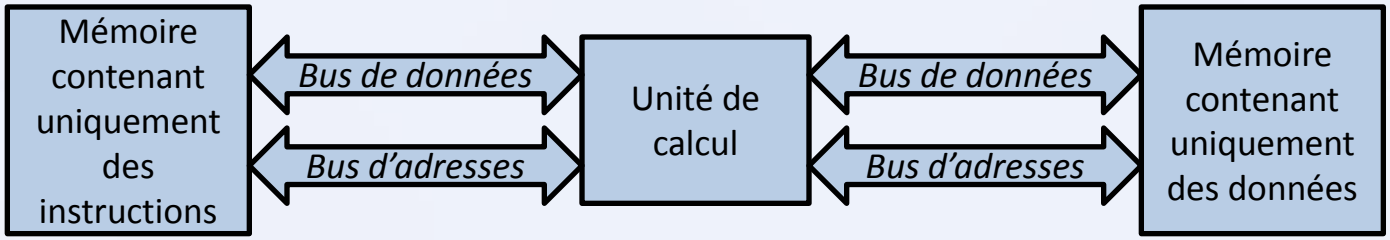
Architecture des microcontrôleurs

Les deux types d'architecture

- **Von Neumann :**



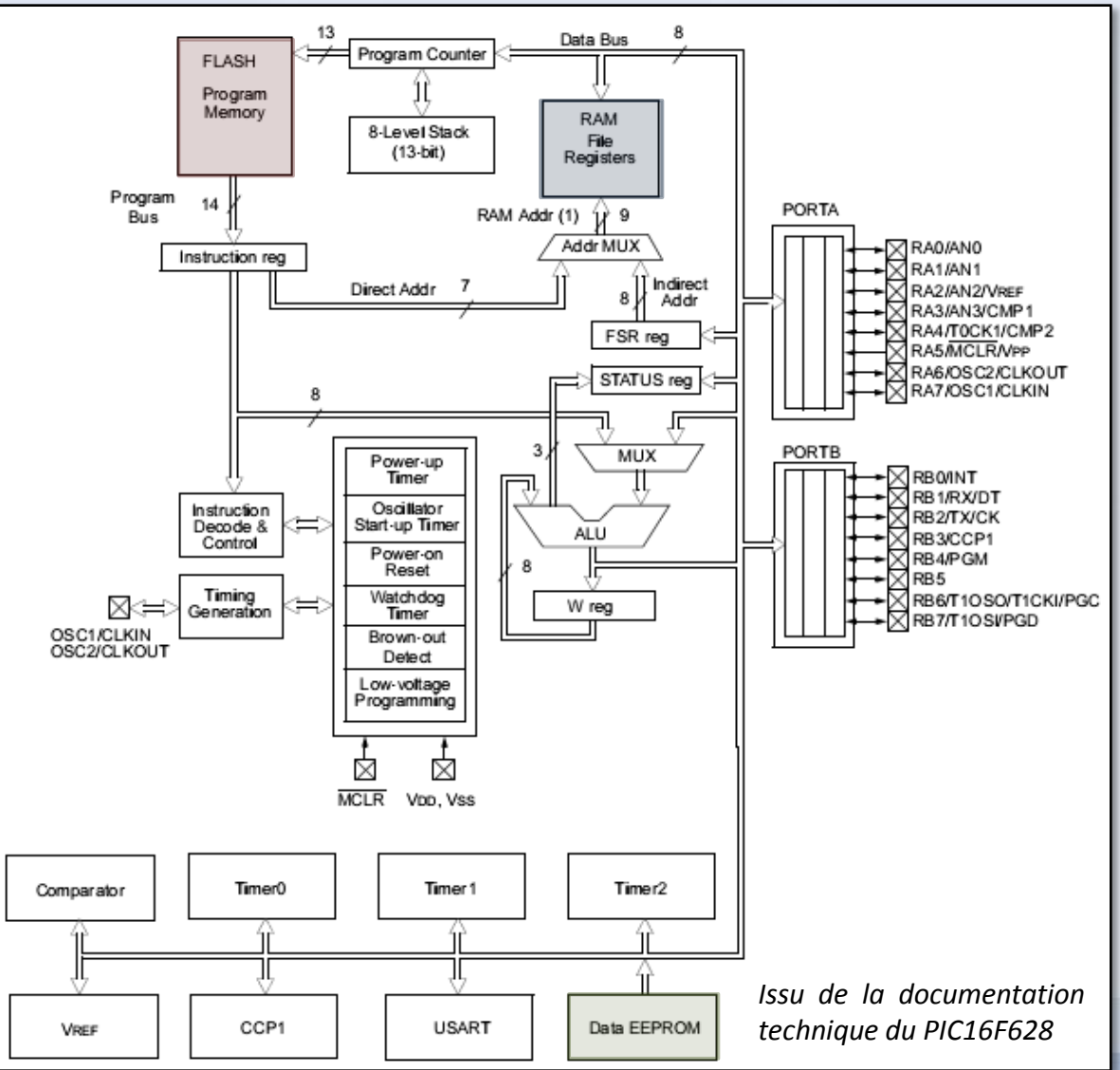
- **Harvard :**



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

Architecture des microcontrôleurs

Schéma bloc d'un microcontrôleur



Issu de la documentation technique du PIC16F628

Les mémoires :

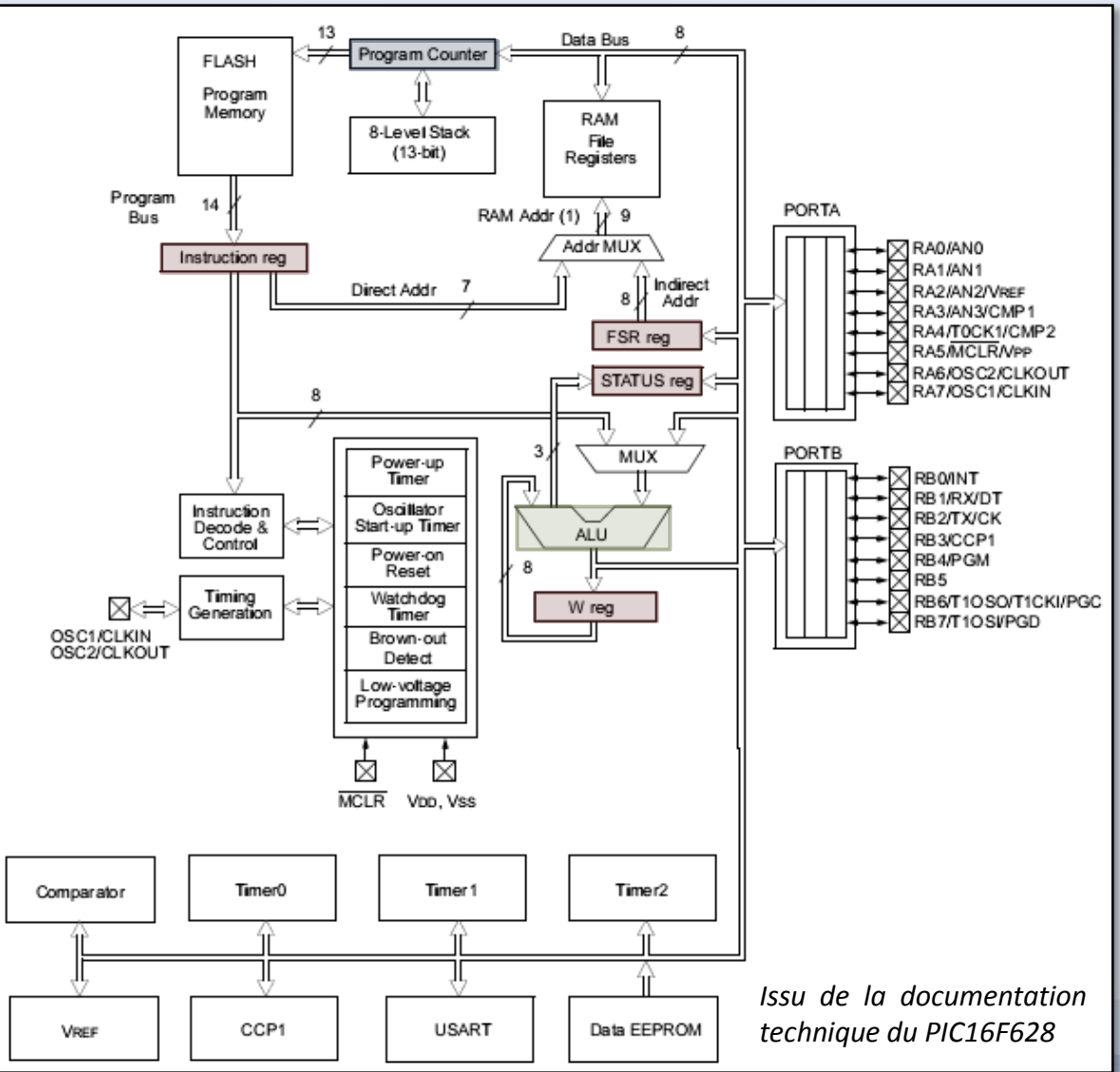
RAM (Random Access Mem.) :
Mémoire rapide permettant de stocker des données temporairement.

ROM (Read Only Memory) :
Mémoire à lecture seule, programmée à vie.

EEPROM (Elec. Erasable Programmable Read Only Memory) :
Mémoire lente permettant de stocker des données même après coupure de l'alim.

Architecture des microcontrôleurs

Schéma bloc d'un microcontrôleur



Issu de la documentation technique du PIC16F628

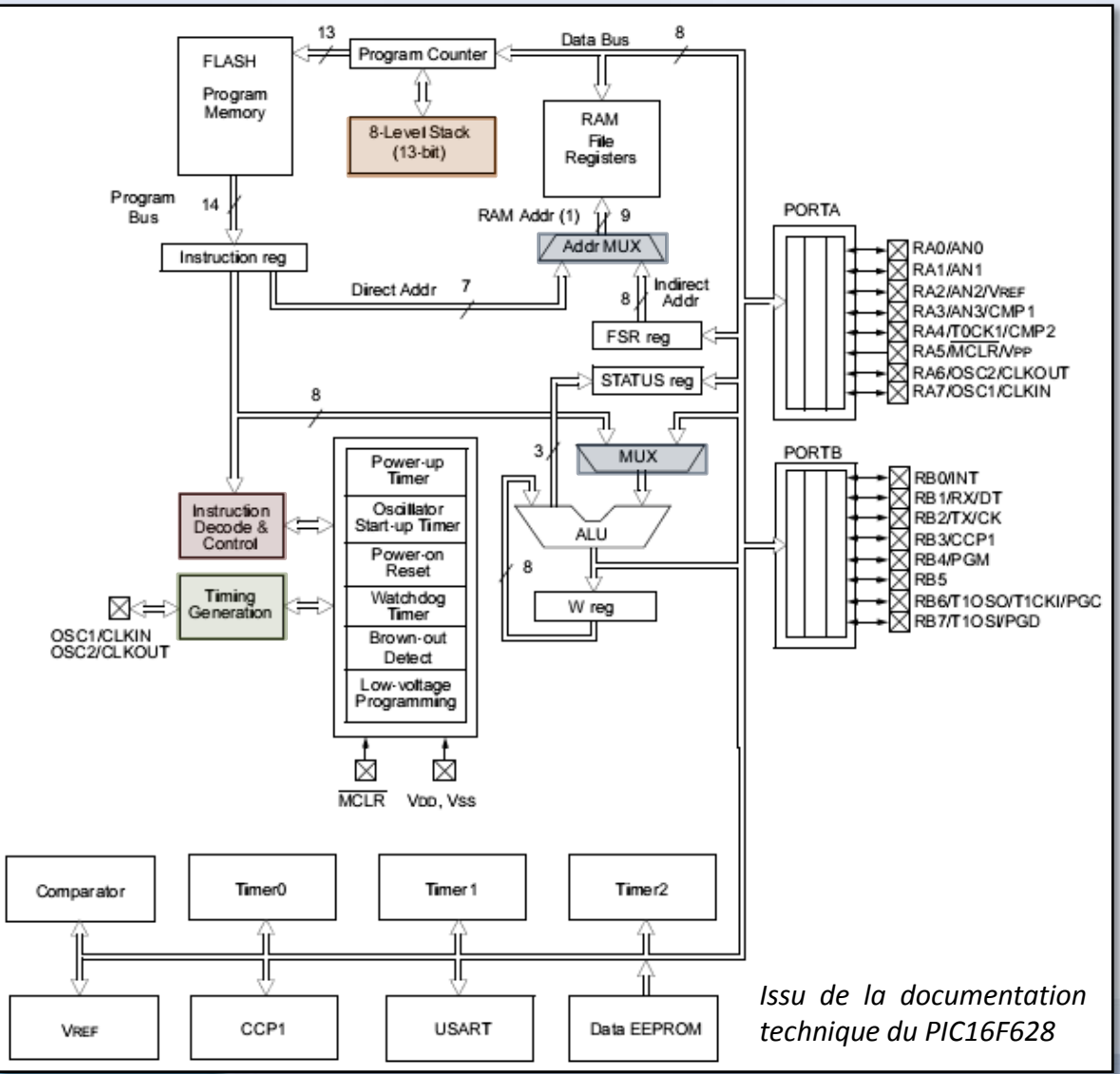
PC (Program Counter)

Registres (case mémoire)

ALU (Arithmetic and Logical Unit)

Architecture des microcontrôleurs

Schéma bloc d'un microcontrôleur



Issu de la documentation technique du PIC16F628

Multiplexeurs

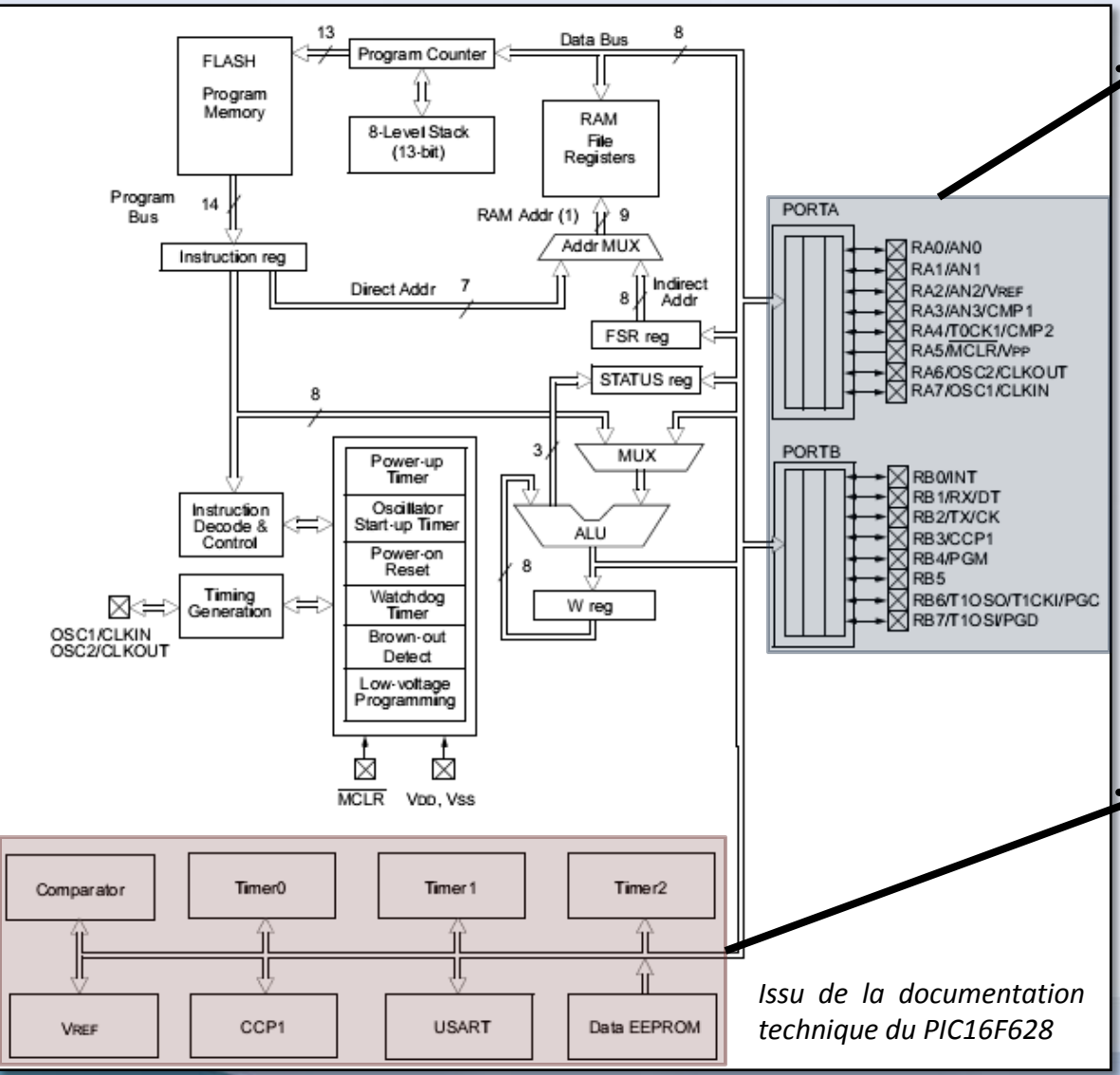
Décodeur d'instructions

Horloge

Stack (pile) :
LIFO (Last In First Out)
FIFO (First In First Out)

Architecture des microcontrôleurs

Schéma bloc d'un microcontrôleur



Issu de la documentation technique du PIC16F628

Ports d'entrées / sorties

- USART (Universal Sync. Async. Receiver Transmitter) Interface de communication série
- CCP (Capture/Compare/PWM) Modulation en largeur d'impulsions
- Timer
- Comparateur
- CAN/CNA
- Référence de tension
- *Module HF*
- *Liaison USB, ...*

Architecture des microcontrôleurs

Les éléments de choix

Architecture :
- ALU (8, 16, 32, 64 bits)
- Structure du processeur (Harvard, Von Neumann)
- Type de processeur (RISC, CISC)
- Taille des mémoires programme et données
- Nombre de ports d'entrées / sorties

Fonctionnalités :
- Fonctions analogiques : CAN, CNA, Comparateur, ...
- Fonctions de timing : Timer, Watchdog, ...
- Fonctions de communication : UART (Communication série), USB, I2C, ...
- Facilité de programmation : In-Circuit Serial Programming, Self Programming, ...

Architecture des microcontrôleurs

Les éléments de choix

Mise en œuvre, maintenance :

- Coût de développement : outils de développement, formation, ...
- Suivi du microcontrôleur : production suivie, disponibilité, composant obsolète, ...

Caractéristiques électriques :

- Fréquence d'horloge
- Tensions d'alimentation
- Consommation d'énergie, modes faible consommation d'énergie, ...

Caractéristiques physiques :

- Type de boîtier: DIL, PLCC, ...

Sommaire

- Présentation de l'informatique industrielle
- Rappels sur les systèmes de numération
- Rappels sur la logique combinatoire et séquentielle
- Les différents systèmes programmables
- Architecture des microcontrôleurs

→***Etude du microcontrôleur PIC 18F67K22***

- Conception d'un système embarqué
- Programmation du PIC 18F67K22

Etude du microcontrôleur PIC 18F67K22

Phase de démarrage du microcontrôleur

Suite à une opération de remise à zéro (RESET), le microcontrôleur effectue une phase de démarrage:

1. RESET : il peut être déclenché par la mise sous tension du microcontrôleur, la réception d'un signal sur la broche RESET du microcontrôleur, une instruction de RESET, ...

2. Initialisation du microcontrôleur : le microcontrôleur effectue une temporisation afin de garantir la stabilité des signaux d'horloge.

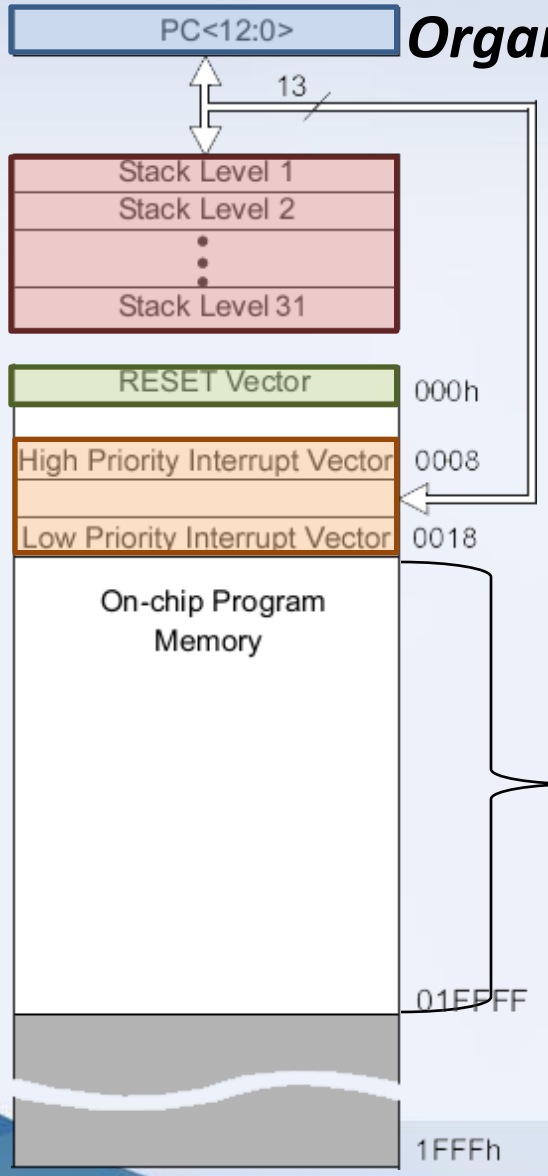
3. Effacement des registres : le microcontrôleur efface le contenu des registres (variable en fonction du « mode de RESET » effectué).

4. Lecture du vecteur RESET : Le microcontrôleur lit l'adresse du programme principal dans la mémoire programme.

5. Début de l'exécution du programme principal.

Etude du microcontrôleur PIC 18F67K22

Organisation de la mémoire programme



Compteur de programme (PC)

Pile (Stack)

Vecteur Reset : Pointeur vers l'adresse mémoire du début du programme principal.

Vecteurs d'interruption : Pointeur vers l'adresse mémoire du programme à exécuter en cas d'interruptions

Mémoire programme :
Zone mémoire réservée au stockage des programmes écrits par l'utilisateur.

Remarque : Un pointeur est une variable contenant une adresse mémoire.

Etude du microcontrôleur PIC 18F67K22

Fonctionnalités intégrées au microcontrôleur

Les microcontrôleurs intègrent des fonctionnalités qu'il est souvent utile de connaître pour gagner du temps de développement.

Par exemple, le PIC18F67K22 intègre les « modules » suivants :

- Différents modes de gestion de l'alimentation
- Un chien de garde *(Watchdog)*
- Des ports d'entrées / sorties parallèles
- Des compteurs *(Timer)*
- Des modules « Capture / Compare / PWM » *(CCP)*
- Des comparateurs *(Comparator)*
- Des modules de conversion analogique/numérique *(CAN/CNA)*
- Des modules de communication synchrone et asynchrone *(USART / SPI / I2C / …)*

Etude du microcontrôleur PIC 18F67K22

Fonctionnalités intégrées au microcontrôleur

Les différents modes de fonctionnement du microcontrôleur :

Les microcontrôleurs possèdent de nombreux modes de fonctionnement visant principalement à réduire la consommation d'énergie qui est une contrainte forte dans les systèmes embarqués.

• *Run mode* : mode de fonctionnement par défaut du microcontrôleur, toutes les fonctions sont activées, la consommation d'énergie est maximale.

• *Sleep mode :* le microcontrôleur est placé en mode sommeil, la consommation d'énergie est minimale, le microcontrôleur peut-être réveillé par une interruption.

• *IDLE mode :* le processeur du microcontrôleur est arrêté, plus aucune instruction n'est exécutée, l'utilisateur peut choisir de désactiver des fonctions du microcontrôleur afin de diminuer la consommation d'énergie. Les fonctions activées restantes fonctionnent normalement et peuvent réveiller le microcontrôleur par une interruption.

Etude du microcontrôleur PIC 18F67K22

Fonctionnalités intégrées au microcontrôleur

Le « chien de garde » (Watchdog) :

Un **watchdog** (WDT) est un dispositif de protection servant à éviter que le microcontrôleur ne se bloque. Un **watchdog** effectue un redémarrage du système (RESET) si une action définie n'est pas effectuée dans un délai donné.

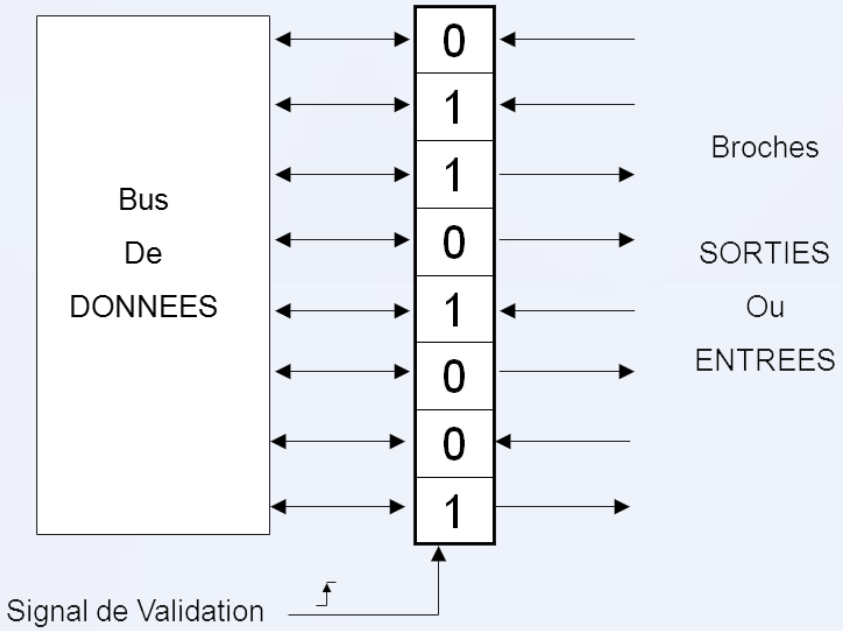
Concrètement, l'utilisateur affecte une valeur à un registre (Watchdog Postscaler), qui définit une durée temporelle (timeout). Périodiquement le microcontrôleur va incrémenter un registre (Watchdog counter). Si ce registre est plein (overflow), le microcontrôleur effectue un re-démarrage.

Pour que le microcontrôleur ne redémarre pas, le programme doit donc réinitialiser périodiquement le registre (Watchdog counter).

Etude du microcontrôleur PIC 18F67K22

Fonctionnalités intégrées au microcontrôleur

Les ports d'entrées / sorties parallèles :

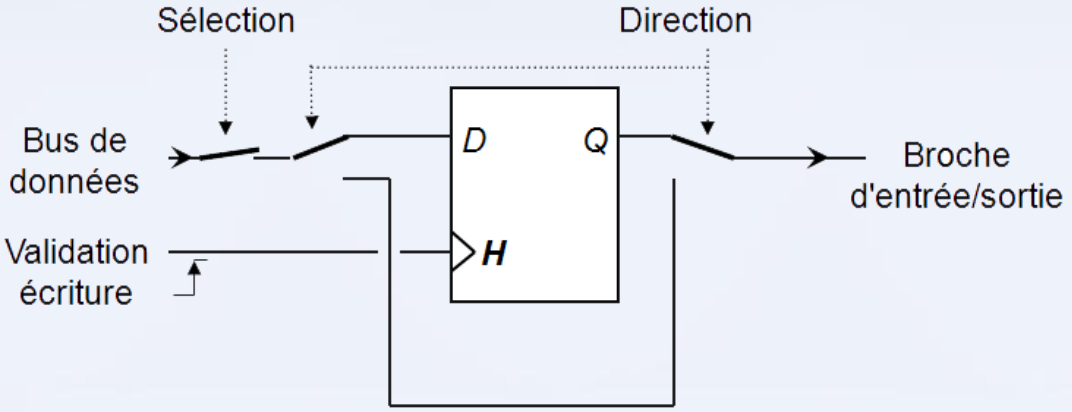


Etude du microcontrôleur PIC 18F67K22

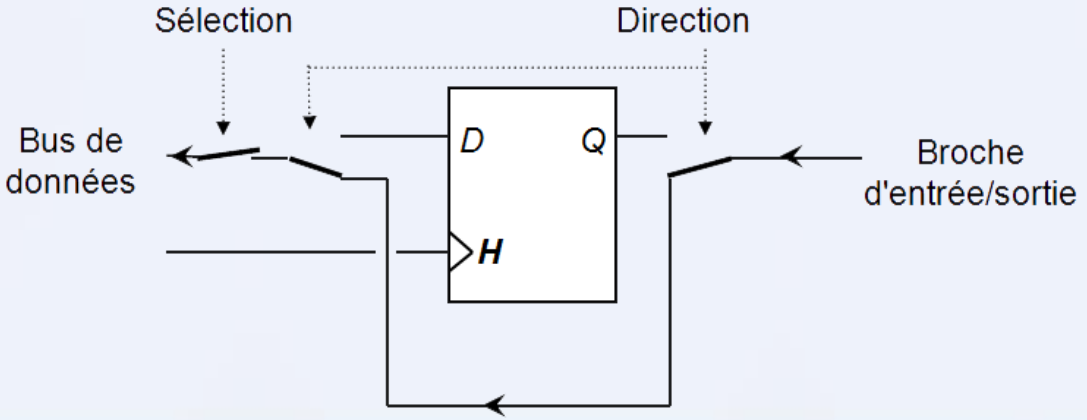
Fonctionnalités intégrées au microcontrôleur

Les ports d'entrées / sorties parallèles :

SORTIE



ENTREE



Etude du microcontrôleur PIC 18F67K22

Fonctionnalités intégrées au microcontrôleur

La fonctionnalité « Timer » :

Les **timers** sont des registres incrémentés à chaque réalisation d'un événement, la valeur de ces registres pouvant être pré-positionnée à une valeur initiale.

Les événements qui commandent l'incrémentation sont :

- un cycle d'horloge, c'est la fonction « timer »
- un front montant sur une broche en entrée, c'est la fonction « counter ».

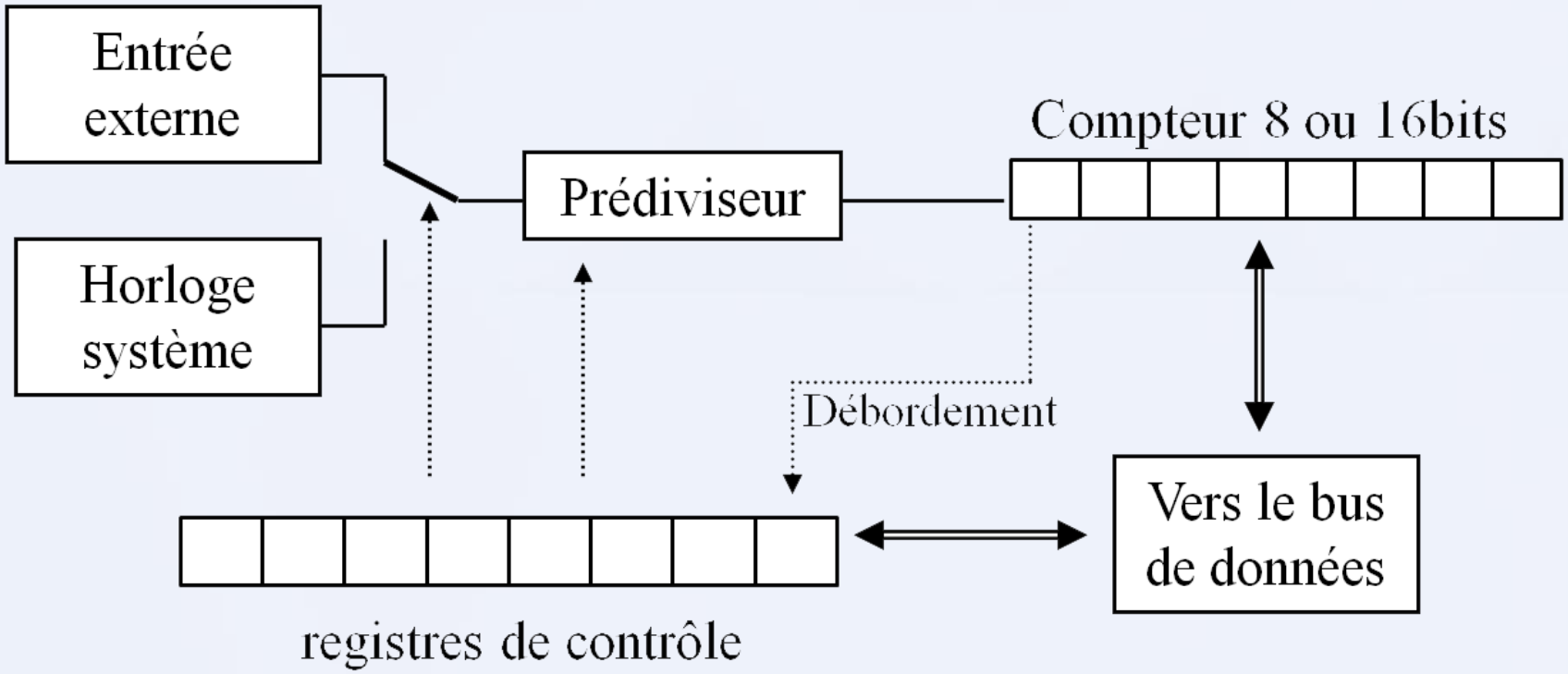
Il en découle que le module timer peut remplir les fonctions suivantes :

- Utilisation « timer » : Permet de fournir une référence temporelle à partir de l'horloge du microcontrôleur, notamment dans le cadre d'applications temps réel.
- Utilisation « counter » : Sert à compter un nombre d'événements asynchrones sur une broche d'entrée du microcontrôleur.

Etude du microcontrôleur PIC 18F67K22

Fonctionnalités intégrées au microcontrôleur

La fonctionnalité « Timer » :



Etude du microcontrôleur PIC 18F67K22

Fonctionnalités intégrées au microcontrôleur

Les modules « CCP » :

Ces modules possèdent trois modes de fonctionnement.

- ***Capture :***

Ce mode déclenche une action si un événement prédéterminé apparaît (ex : changement d'état sur une broche). Utilisé avec les timers, ce module peut compter les temps d'arrivées.

- ***Compare :***

Le mode compare effectue une comparaison permanente entre le contenu d'un timer et une valeur donnée pour déclencher une action si ces contenus sont égaux.

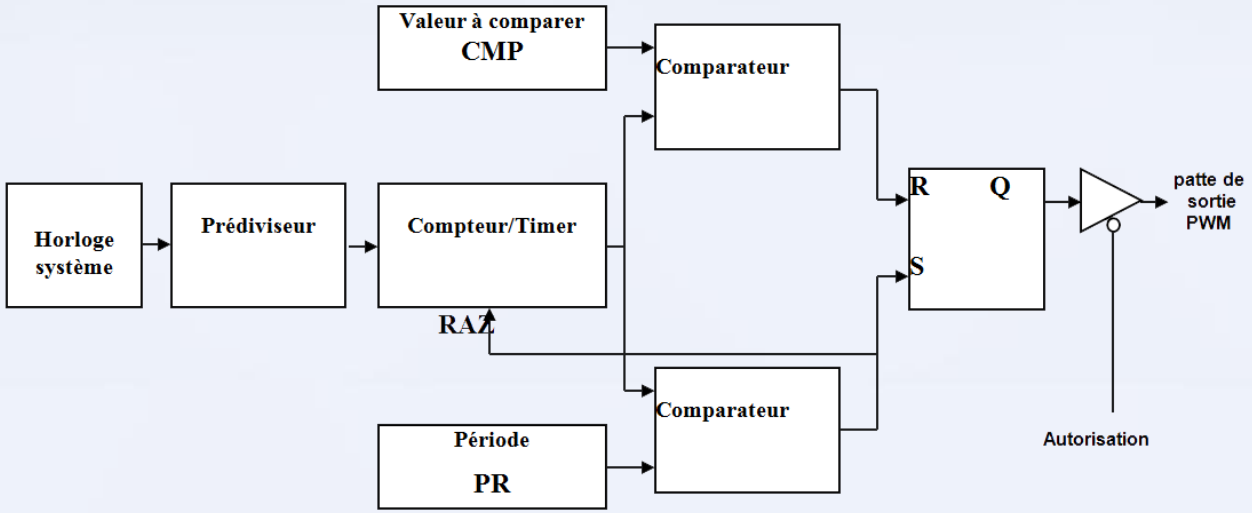
- ***Pulse width modulation (PWM) :***

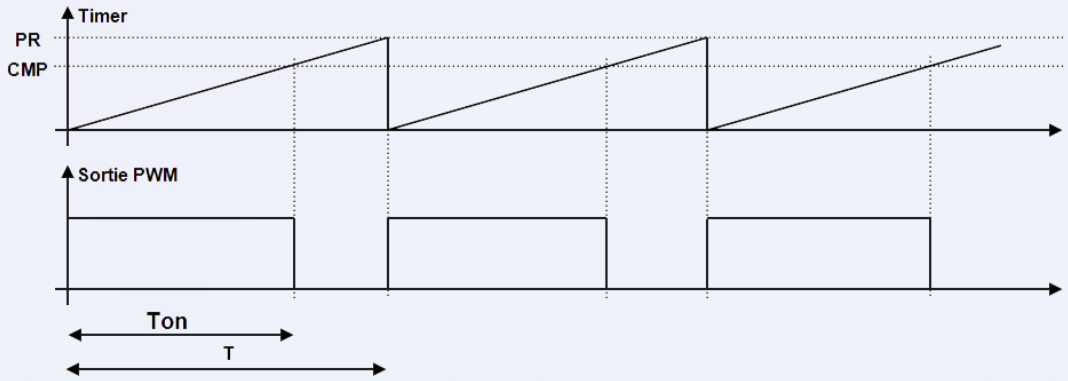
Le mode PWM génère un signal rectangulaire de fréquence et de rapport cyclique choisis par l'utilisateur.

Etude du microcontrôleur PIC 18F67K22

Fonctionnalités intégrées au microcontrôleur

Les modules « CCP » en mode « PWM » :





Etude du microcontrôleur PIC 18F67K22

Fonctionnalités intégrées au microcontrôleur

Les comparateurs :

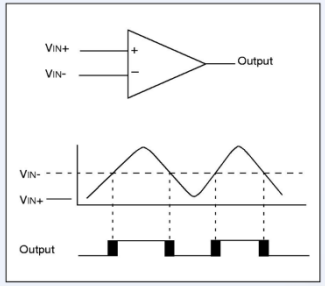
Les **comparateurs** permettent de comparer le signal analogique présent sur une broche du microcontrôleur à une valeur de référence.

Cette valeur de référence peut être :

- un signal analogique dont on fait l'acquisition sur une autre broche du microcontrôleur (convertisseur analogique - numérique)
- une tension de référence générée en interne par le microcontrôleur à l'aide du module de génération de tension de référence.

Ce principe de fonctionnement décrit ci-contre permet typiquement d'effectuer une commande de type tout-ou-rien (TOR).

FIGURE 20-2: SINGLE COMPARATOR



Etude du microcontrôleur PIC 18F67K22

Fonctionnalités intégrées au microcontrôleur

Le module de conversion analogique/numérique :

Le **module de conversion analogique/numérique** permet de convertir le signal analogique présent sur une broche du microcontrôleur en un signal numérique.

Les paramètres à prendre en compte pour la numérisation d'un signal sont :

- **la pleine échelle** du module de conversion A/N :

Indique la plage de tension admissible en entrée du module.

- **la dynamique** :

Indique le nombre de bits utilisés pour coder une valeur analogique en numérique.

- **la fréquence d'échantillonnage**

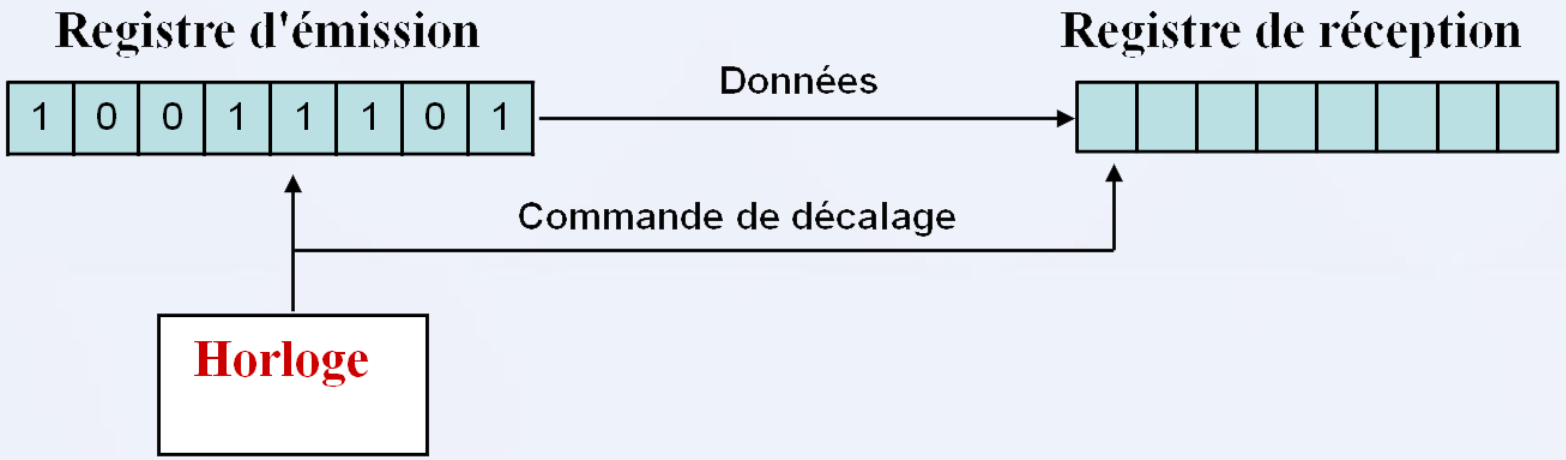
Remarque :

- La pleine échelle et la dynamique permettent de calculer la résolution en tension du module de conversion A/N
- La fréquence d'échantillonnage doit respecter le « théorème de Shannon »

Etude du microcontrôleur PIC 18F67K22

Fonctionnalités intégrées au microcontrôleur

Les modules de communication synchrone :

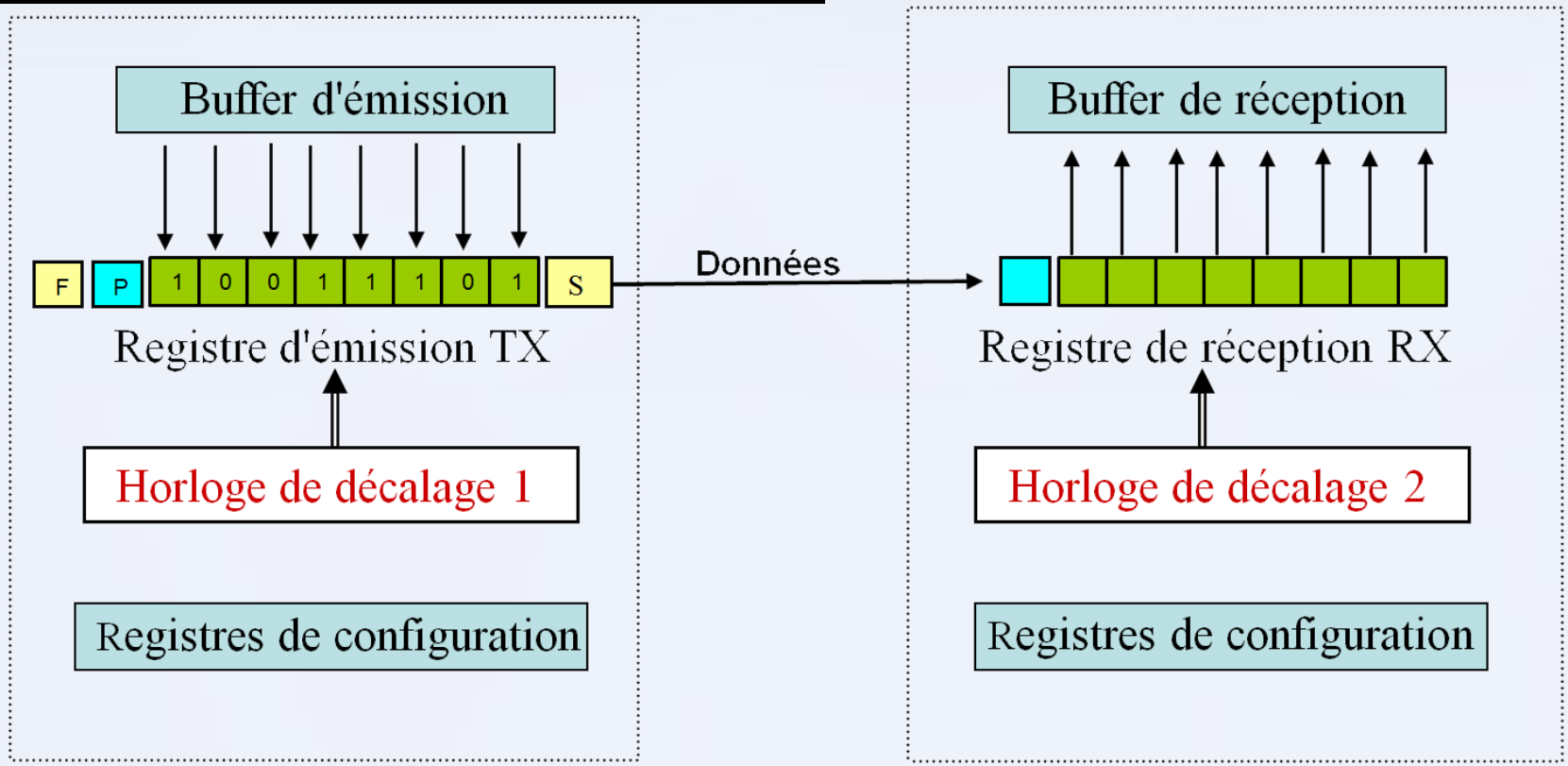


Un bit est envoyé à chaque front d'horloge.
Il faut donc 8 fronts d'horloge pour envoyer un octet.

Etude du microcontrôleur PIC 18F67K22

Fonctionnalités intégrées au microcontrôleur

Les modules de communication asynchrone :



L'horloge 2 se synchronise grâce aux bits de start « s » et de stop « f ».

Sommaire

- Présentation de l'informatique industrielle
- Rappels sur les systèmes de numération
- Rappels sur la logique combinatoire et séquentielle
- Les différents systèmes programmables
- Architecture des microcontrôleurs
- Etude du microcontrôleur PIC 18F67K22

→ ***Conception d'un système embarqué***

- Programmation du PIC 18F67K22

Conception d'un système embarqué

Développement en 3 étapes distinctes

- **Le développement matériel** s'appuie sur un *cahier des charges*, *c.à.d.* la définition des fonctionnalités et des performances du système. Cette étape doit permettre de spécifier les *caractéristiques* du microcontrôleur, de ses périphériques et de l'électronique associée.

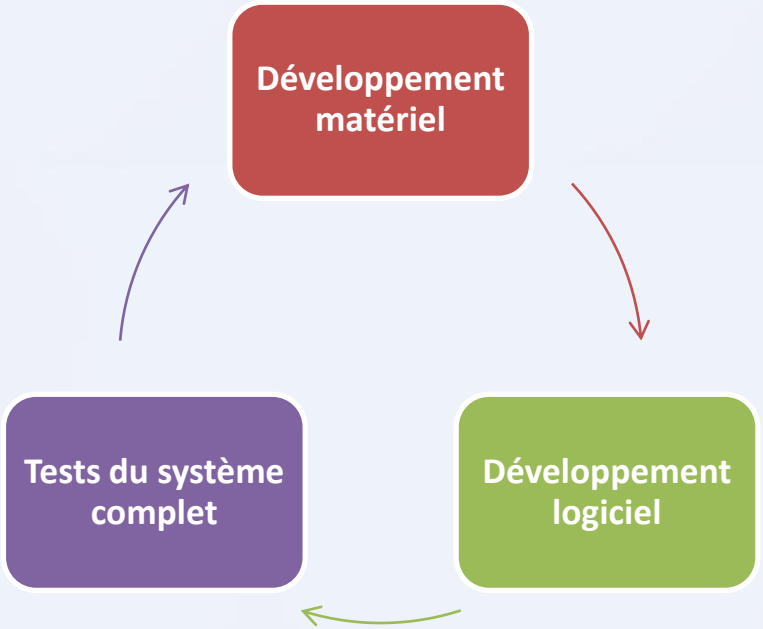
- **Le développement logiciel** s'appuie sur l'étape précédente pour construire un algorigramme, puis le code qui va être testé. Cette étape requiert que vous choisissiez le langage (assembleur et/ou évolué) que vous utiliserez sur des bases *objectives*, par exemple de manière à optimiser le temps de développement, la facilité de maintenance, le nombre d'opérations, *etc.*

- **La phase de test** doit être menée pour vérifier que le cahier des charges initial est bien rempli. Cette phase de test « finale » n'empêche pas d'avoir mené des tests séparés lors des phases de développement matériel et logiciel.

Conception d'un système embarqué

Cycle de conception

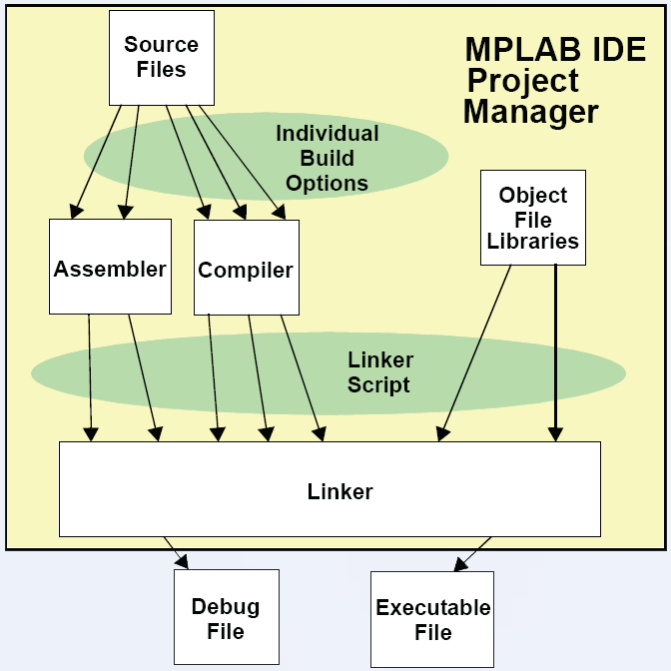
Chacune de ces trois étapes précédentes peut être relativement complexe et mobiliser des moyens financiers et humains conséquents. Par ailleurs, le test à une étape peut remettre en cause les choix fait à une étape précédente : en pratique, on est donc plutôt confronté à un **cycle de conception** plutôt qu'à un enchaînement parfaitement séquentiel !



Conception d'un système embarqué

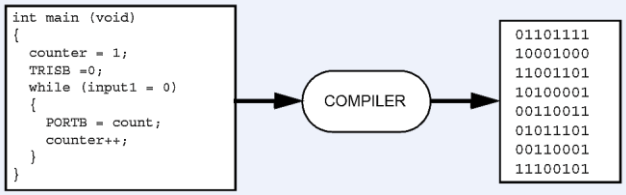
Développement logiciel

La construction d'un code machine exécutable s'appuie sur un certain nombre de composantes (fichiers sources, librairies) qui suivent le diagramme organisationnel ci-dessous.



- **Les fichiers sources** écrits dans un langage assembleur et/ou évolué doivent permettent au système embarqué d'effectuer les tâches requises.

- **Le compilateur et/ou l'assembleur** a pour rôle de convertir les instructions des sources en langage machine.



- **L'éditeur de lien** permet de construire un exécutable à partir des objets issus soit des sources soit de librairies préexistantes.

Sommaire

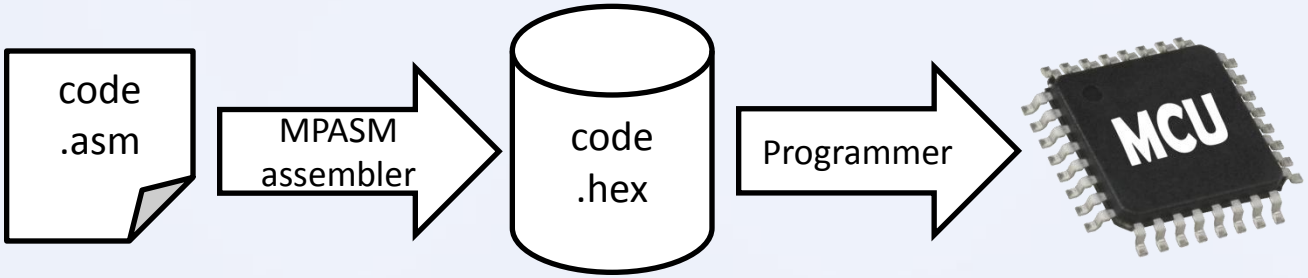
- Présentation de l'informatique industrielle
- Rappels sur les systèmes de numération
- Rappels sur la logique combinatoire et séquentielle
- Les différents systèmes programmables
- Architecture des microcontrôleurs
- Etude du microcontrôleur PIC 18F67K22
- Conception d'un système embarqué

→ ***Programmation du PIC 18F67K22***

Programmation du PIC 18F67K22

La programmation en Assembleur

Le langage Assembleur (abrégé ASM) **est un langage de programmation de bas-niveau**, qui fait la correspondance entre des instructions en langage machine (mots binaires) et des symboles appelés *mnémoniques* plus simples à utiliser.



Le langage Assembleur est un langage compilé, c'est à dire :

1. L'utilisateur écrit son programme en langage Assembleur. Ce fichier est ensuite assemblé pour traduire le programme en langage machine (avec éventuellement des améliorations).

2. Le programme en langage machine est alors utilisé pour programmer le microcontrôleur, c.à.d. qu'il est transféré dans la mémoire (programme) pour être exécuté.

Programmation du PIC 18F67K22

Jeu d'instructions

Un jeu d'instructions est un ensemble d'opérations directement réalisables sur un système microprogrammé donné.

Par exemple : Le PIC18F67K22 (RISC) possède un jeu d'instructions composé de 75 instructions. L'exécution d'une instruction peut nécessiter un ou plusieurs cycles d'horloges suivant la complexité de l'instruction.

Remarque : Un 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.

Programmation du PIC 18F67K22

Les types d'instructions en Assembleur

A. Les instructions propres au microcontrôleur :

- Les instructions de transfert : ***movlw***, ***movf***, ...
- Les instructions arithmétiques : ***decf***, ***addwf***, ...
- Les instructions logiques : ***xorlw***, ***andlw***, ...
- Les instructions de branchement : ***bz*** (*branch if zero*), ***bra*** (*branch always*), ...

B. Les instructions préprocesseur permettent au programmeur de donner des indications au compilateur, elles sont destinées au PC et non pas au microcontrôleur !

Il existe différents types d'instruction préprocesseur :

- les instructions de contrôle : ***org*** = début du programme, ***end*** = fin du programme, ...
- les instructions conditionnelles : ***if***, ***else***, ***endif***, …
- les instructions relatives aux données : ***res*** = réservation d'espace mémoire, …
- les instructions pour les macros

Programmation du PIC 18F67K22

Les différentes opérandes

Les opérandes utilisées par les instructions ASM peuvent être de plusieurs types :

W : registre de travail
f : adresse mémoire de registres (register file address)

d : sélection de destination : d=0 vers W, d=1 vers f
k : champ littéral (8 bits)
b : numéro du bit dans un octet (sur 3 bits)

PC : compteur programme

Programmation du PIC 18F67K22

Le registre STATUS

REGISTER 6-2: STATUS REGISTER

U-0	U-0	U-0	R/W-x	R/W-x	R/W-x	R/W-x	R/W-x
—	—	—	N	OV	Z	DC(1)	C(2)
bit 7							bit 0

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

bit 7-5 **Unimplemented:** Read as '0'

bit 4 **N:** Negative bit

This bit is used for signed arithmetic (2's complement). It indicates whether the result was negative (ALU MSB = 1).

1 = Result was negative
0 = Result was positive

bit 3 **OV:** Overflow bit

This bit is used for signed arithmetic (2's complement). It indicates an overflow of the seven-bit magnitude which causes the sign bit (bit 7) to change state.

1 = Overflow occurred for signed arithmetic (in this arithmetic operation)
0 = No overflow occurred

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/$\overline{\text{Borrow}}$ bit(1)

For ADDWF, ADDLW, SUBLW and SUBWF instructions:

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/$\overline{\text{Borrow}}$ bit(2)

For ADDWF, ADDLW, SUBLW and SUBWF instructions:

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

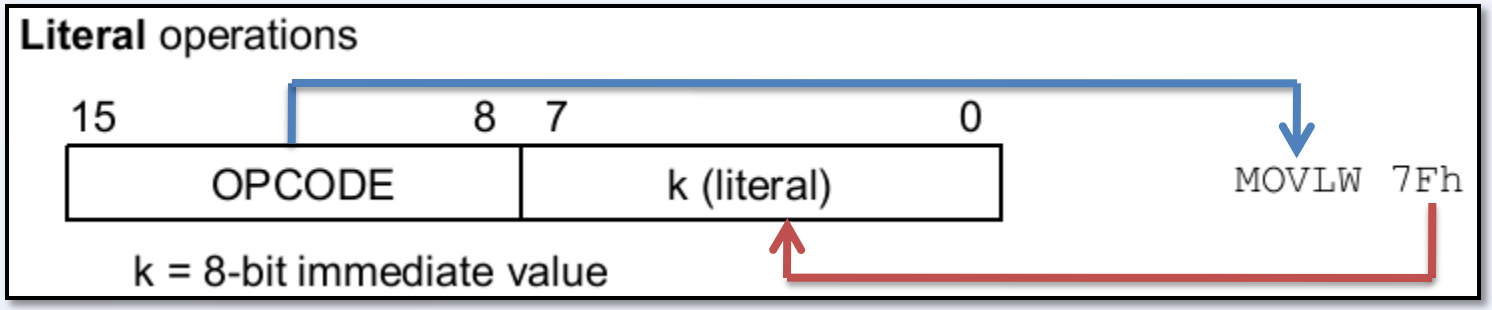
Programmation du PIC 18F67K22

Structure d'une instructions

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.



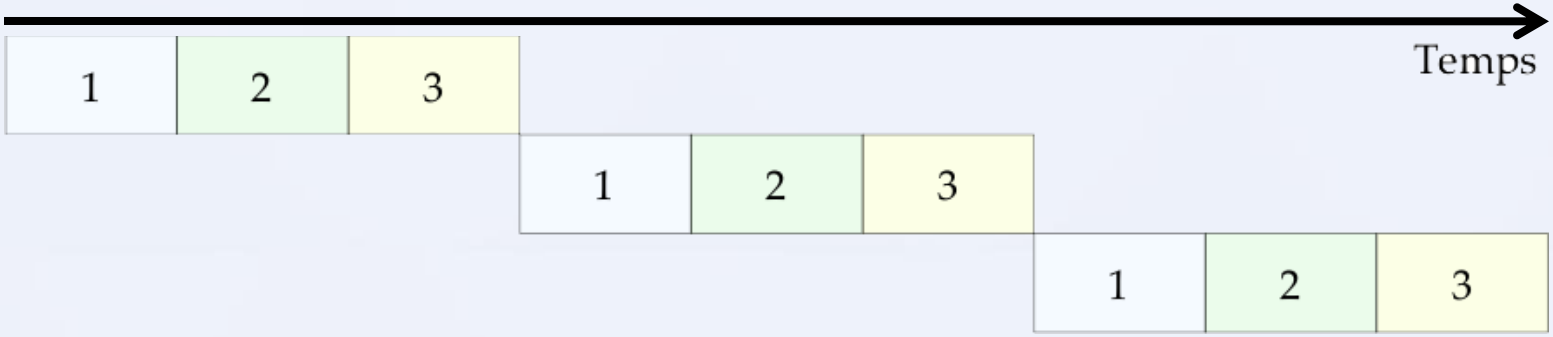
Extrait du datasheet (documentation technique) du PIC18F67K22

Programmation du PIC 18F67K22

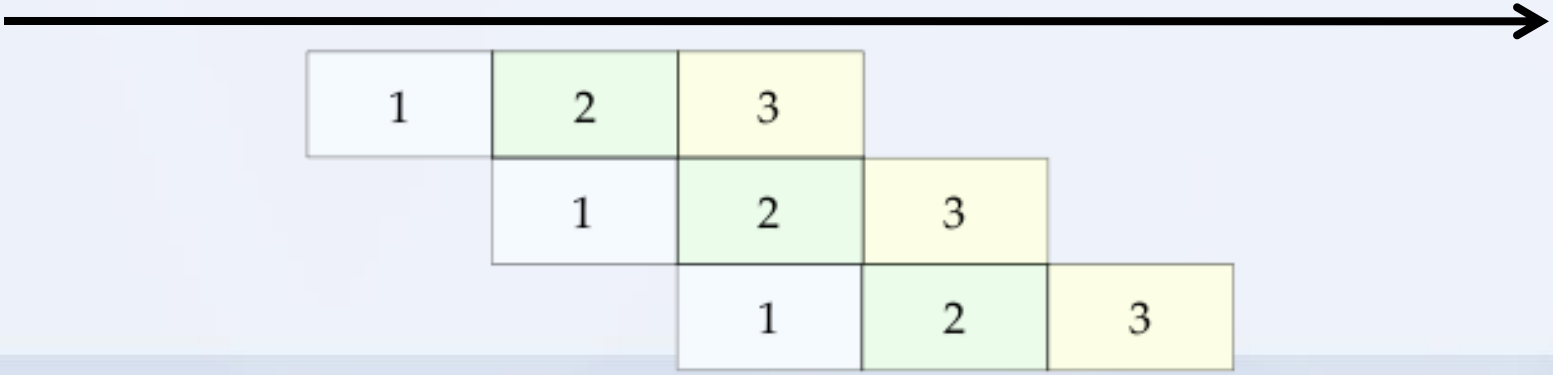
Pipeline et flot d'instructions

3 étapes pour l'exécution d'une instruction :

1) Lecture de l'instruction
2) Décodage de l'instruction
3) Exécution de l'instruction



Création d'un pipeline: Permet une exécution plus rapide des instructions



Programmation du PIC 18F67K22

Les différents 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.*

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

ex : NOP, RESET, CLRWDT ;

Description de l'instruction RESET extraite de la notice technique (le « datasheet ») du PIC 18F67K22.

Remarque : Un nombre important d'information utiles figure sur ces fiches.

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

Programmation du PIC 18F67K22

Les différents 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.*

L'adressage immédiat : l'opérande est une valeur

ex : MOVLW 5Ah ;

Nombres de cycles nécessaires à l'exécution

Exécution de l'instruction (pipeline à 4 niveaux)

MOVLW	Move Literal to W			
Syntax:	MOVLW k			
Operands:	$0 \le k \le 255$			
Operation:	$k \rightarrow W$			
Status Affected:	None			
Encoding:	0000	1110	kkkk	kkkk
Description:	The eight-bit literal 'k' is loaded into W.			
Words:	1			
Cycles:	1			

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read literal 'k'	Process Data	Write to W

Example: MOVLW 5Ah

After Instruction
W = 5Ah

Programmation du PIC 18F67K22

Les différents 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.*

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 : ANDWF REG, W, 0

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

ANDWF — **AND W with f**

Syntax: ANDWF f {,d {,a}}

Operands: $0 \le f \le 255$, $d \in [0,1]$, $a \in [0,1]$

Operation: (W) .AND. (f) → dest

Status Affected: N, Z

Encoding:

0001	01da	ffff	ffff

Description: The contents of W are ANDed with register 'f'. If 'd' is '0', the result is stored in W. If 'd' is '1', the result is stored back in register 'f'.

If 'a' is '0', the Access Bank is selected. If 'a' is '1', the BSR is used to select the GPR bank.

If 'a' is '0' and the extended instruction set is enabled, this instruction operates in Indexed Literal Offset Addressing mode whenever $f \le 95$ (5Fh). See **Section 29.2.3 "Byte-Oriented and Bit-Oriented Instructions in Indexed Literal Offset Mode"** for details.

Words: 1

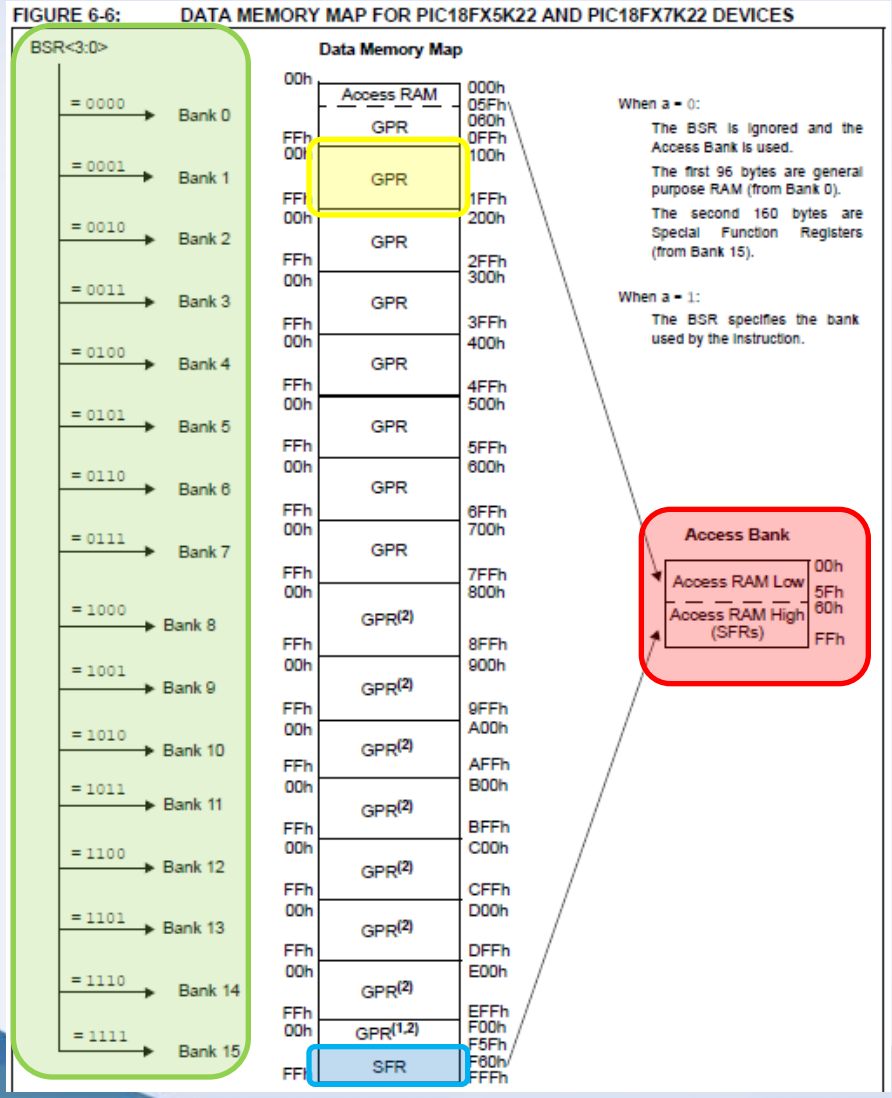
Cycles: 1

Q Cycle Activity:

Q1	Q2	Q3	Q4
Decode	Read register 'f'	Process Data	Write to destination

Programmation du PIC 18F67K22

Les différents modes d'adressage



Organisation de la mémoire données

BSR (*Bank Select Register*)
Permet de présélectionner la page pour un accès mémoire plus rapide.
=> **Pagination de la mémoire**

GPR (*General Purpose Registers*)
Espaces mémoires qui permet le stockage de données temporaires (variable, ...)

Access Bank
Pointeurs vers des zones mémoires

SFR (*Special Function Registers*)
Registres de contrôle et d'état pour les périphériques (notamment...)

Programmation du PIC 18F67K22

Les différents modes d'adressage

Pagination de la mémoire :

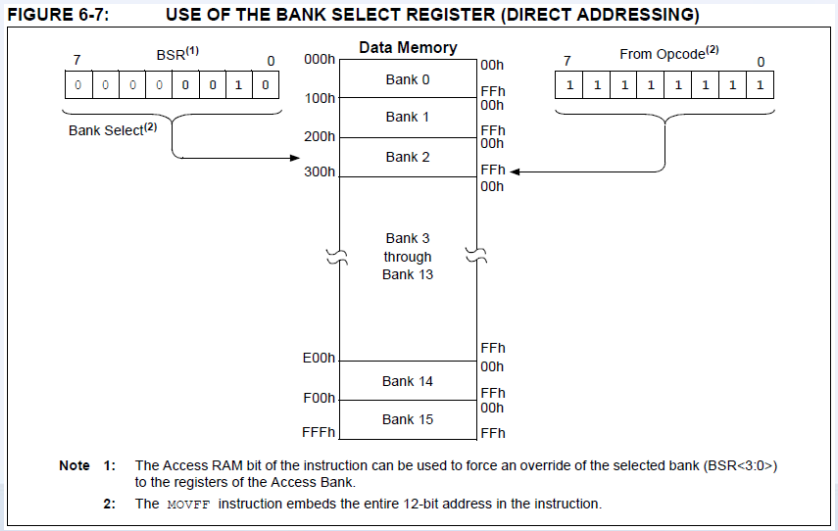
« La pagination de la mémoire consiste à diviser la mémoire en blocs (pages) de longueur fixe. » (Source : Comment Ça Marche)

Une adresse mémoire est alors divisée en deux parties :

Partie haute	Partie basse

Dans le cas d'une instruction avec adressage direct, on transmet seulement la partie basse de l'adresse. Le microcontrôleur utilise le registre BSR pour compléter l'adresse.

Attention ! En adressage direct, on doit s'assurer que l'on travaille **dans la bonne page mémoire**.



Programmation du PIC 18F67K22

Les différents 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.*

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.

Remarque : Il existe de nombreux autres modes d'adressage (ex. implicite, inhérent, relatif). Leur nombre varie en fonction du constructeur et du microcontrôleur !

Programmation du PIC 18F67K22

Les différents modes d'adressage

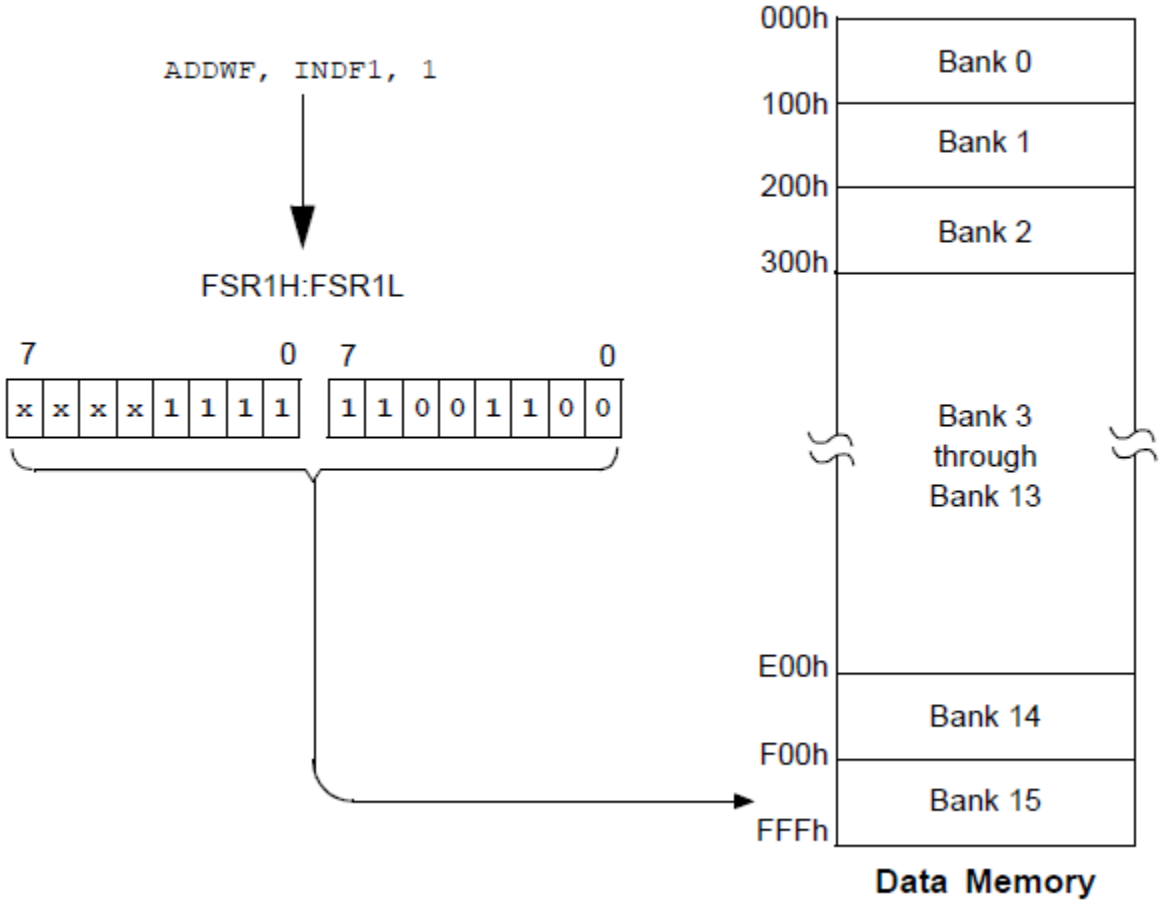
FIGURE 6-8: INDIRECT ADDRESSING

Using an instruction with one of the Indirect Addressing registers as the operand....

...uses the 12-bit address stored in the FSR pair associated with that register....

...to determine the data memory location to be used in that operation.

In this case, the FSR1 pair contains FCCh. This means the contents of location, FCCh, will be added to that of the W register and stored back in FCCh.

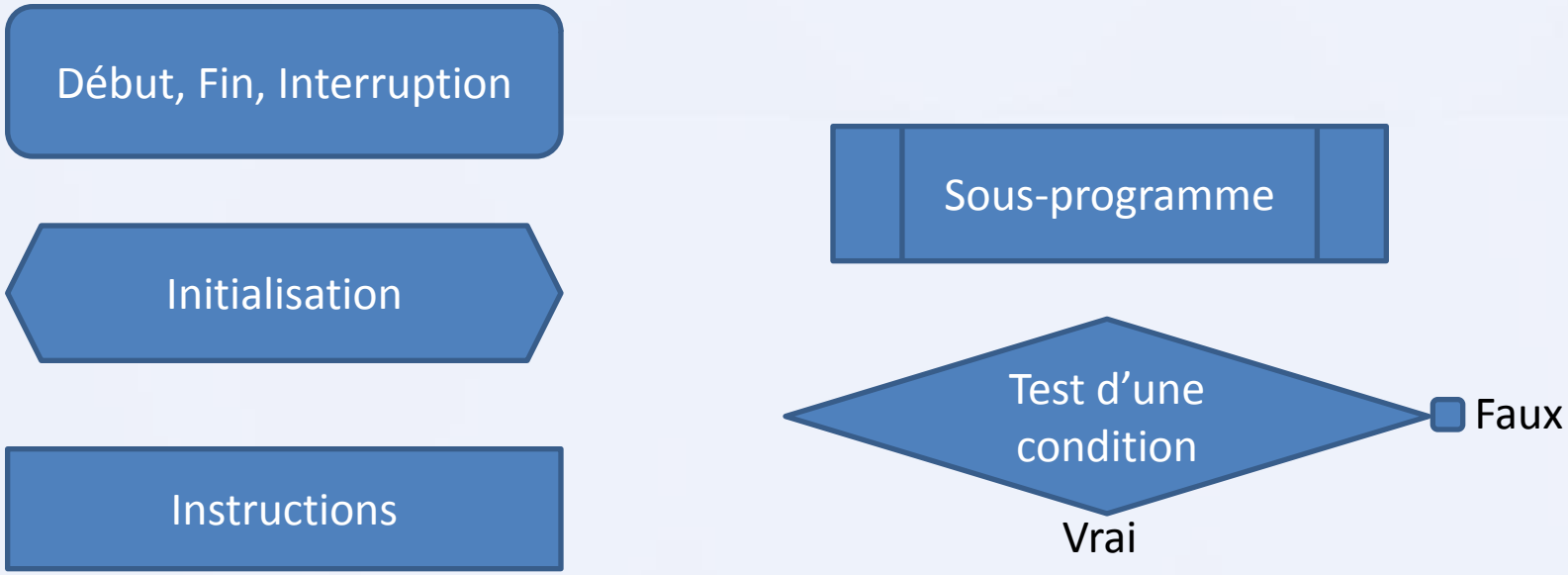


Programmation du PIC 18F67K22

Algorigrammes

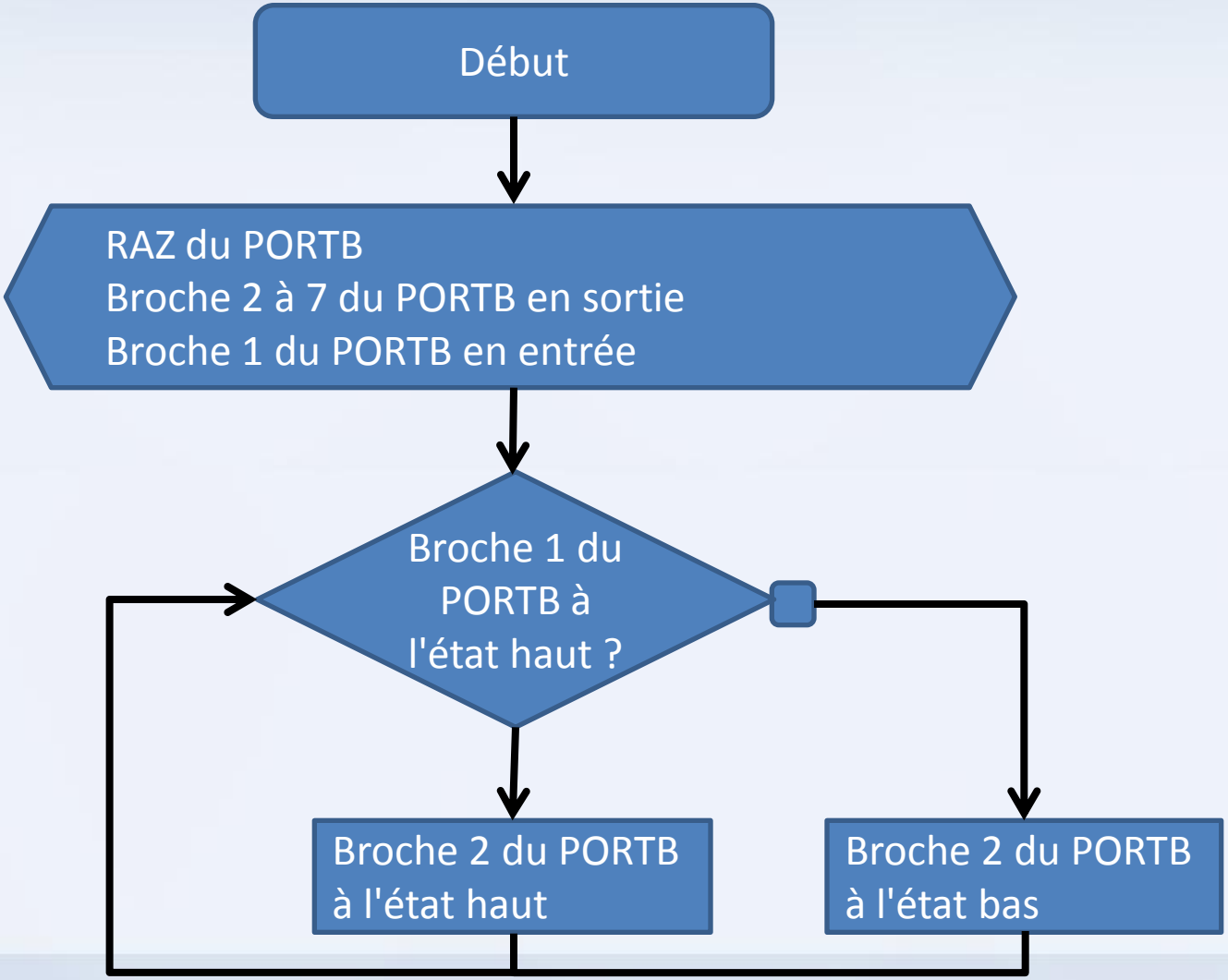
La description du programme par un algorigramme permet de :

- ✓ **gagner en efficacité** lors de la phase de codage du programme
- ✓ **optimiser la structure** du programme
- ✓ **clarifier le fonctionnement** du programme
- ✓ **rendre compréhensible** le programme à une personne extérieure



Programmation du PIC 18F67K22

Premier programme en assembleur



Programmation du PIC 18F67K22

Premier programme en assembleur

Dans un programme en assembleur, on peut distinguer une **partie préliminaire** qui est systématique, c.à.d. qui ne change pas d'un programme à l'autre.

```
;
; Filename : premier_programme.asm
;
; Description : Recopie de l'état de la broche 1 du PORTB sur la broche 2 du PORTB
;
; Author: Cyrille DROMAS
; Company: Université de Picardie Jules Vernes
; Revision: 1.00
; Date: 2012/11

; Définition du microcontrôleur utilisé
list p=18f67k22

; Définitions des emplacements mémoires des registres (définie par MPLAB - logiciel de développement Microchip)
#include <p18f67k22.inc>
```

Programmation du PIC 18F67K22

Premier programme en assembleur

La première partie concerne l'en-tête qui définit, le plus clairement possible, la fonction du programme ainsi que divers informations permettant de gérer l'historique du code (auteur, date d'écritures et de modifications, numéro de version du fichier, *etc.*)

```
;
; Filename : premier_programme.asm
;
; Description : Recopie de l'état de la broche 1 du PORTB sur la broche 2 du PORTB
;
; Author: Cyrille DROMAS
; Company: Université de Picardie Jules Vernes
; Revision: 1.00
; Date: 2012/11
```

```
; Définition du microcontrôleur utilisé
list p=18f67k22

; Définitions des emplacements mémoires des registres (définie par MPLAB - logiciel de développement Microchip)
#include <p18f67k22.inc>
```

Programmation du PIC 18F67K22

Premier programme en assembleur

La déclaration du microcontrôleur permet au compilateur de générer un code machine qui soit compréhensible pour le microcontrôleur que vous souhaitez programmer.

```
;
; Filename : premier_programme.asm
;
; Description : Recopie de l'état de la broche 1 du PORTB sur la broche 2 du PORTB
;
; Author: Cyrille DROMAS
; Company: Université de Picardie Jules Vernes
; Revision: 1.00
; Date: 2012/11
```

```
; Définition du microcontrôleur utilisé
list p=18f67k22
```

```
; Définitions des emplacements mémoires des registres (définie par MPLAB - logiciel de développement Microchip)
#include <p18f67k22.inc>
```

Programmation du PIC 18F67K22

Premier programme en assembleur

Une directive au pré-processeur demande l'inclusion d'un fichier de définition spécifique au microcontrôleur qui définit certaines configurations matérielles par défaut et permet de simplifier l'écriture des programmes, cf. transparent suivant.

```
;
; Filename : premier_programme.asm
;
; Description : Recopie de l'état de la broche 1 du PORTB sur la broche 2 du PORTB
;
; Author: Cyrille DROMAS
; Company: Université de Picardie Jules Vernes
; Revision: 1.00
; Date: 2012/11

; Définition du microcontrôleur utilisé
list p=18f67k22
```

```
; Définitions des emplacements mémoires des registres (définie par MPLAB - logiciel de développement Microchip)
#include <p18f67k22.inc>
```

Programmation du PIC 18F67K22

Premier programme en assembleur

Une directive au pré-processeur demande l'inclusion d'un fichier de définition spécifique au microcontrôleur qui définit certaines configurations matérielles par défaut et permet de simplifier l'écriture des programmes, cf. transparent suivant.

Extrait du fichier p18f67k22.inc de définitions propre au microcontrôleur :

```
;----- Register Files
PORTA                EQU  H'0F80'
PORTB                EQU  H'0F81'
PORTC                EQU  H'0F82'
PORTD                EQU  H'0F83'
PORTE                EQU  H'0F84'
PORTF                EQU  H'0F85'
PORTG                EQU  H'0F86'
LATA                 EQU  H'0F89'
LATB                 EQU  H'0F8A'
LATC                 EQU  H'0F8B'
```

Programmation du PIC 18F67K22

Premier programme en assembleur

On peut distinguer ensuite une **seconde partie** qui correspond à la configuration des éléments du microcontrôleur qui entrent *directement* en jeu dans la fonction réalisée.

La première opération consiste systématiquement à **initialiser le vecteur RESET**.

RAZ du PORTB
Broche 2 à 7 du PORTB en sortie
Broche 1 du PORTB en entrée

```
; Début du programme (vecteur RESET)
org  0x0000
goto init
```

```
init
        clrf  PORTB
        movlw b'00000001' ; (Broche 1 en entrée - Broches 2 à 8 en sortie)
        movwf TRISB ; Configuration de la direction du PORTB
```

Programmation du PIC 18F67K22

Premier programme en assembleur

La seconde opération correspond à la **configuration du PORT B** telle que décrite par l'algorigramme. La démarche à suivre est directement décrite dans le *Datasheet* du PIC 18F67K22.

RAZ du PORTB
Broche 2 à 7 du PORTB en sortie
Broche 1 du PORTB en entrée

```
; Début du programme (vecteur RESET)
org  0x0000
goto init
```

```
init
        clrf  PORTB
        movlw b'00000001' ; (Broche 1 en entrée - Broches 2 à 8 en sortie)
        movwf TRISB ; Configuration de la direction du PORTB
```

Programmation du PIC 18F67K22

Premier programme en assembleur

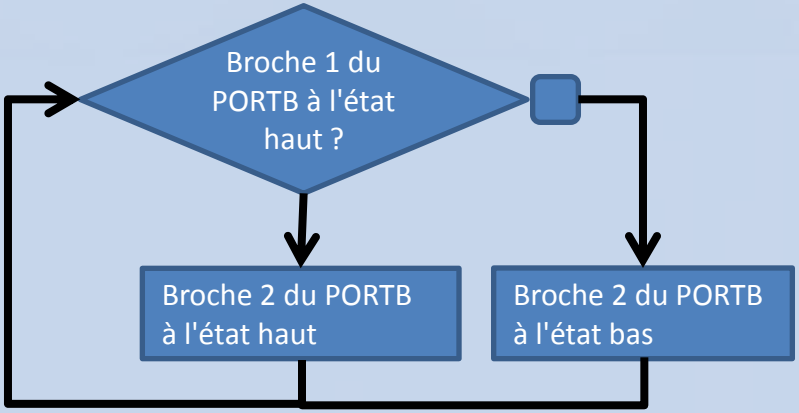
La troisième partie du programme est dédiée à la réalisation de la fonction principale, c.à.d. la boucle et le test.

```
boucle
        ; Broche 1 du PORTB à l'état haut ? Si oui, saute l'instruction suivante
        btfss PORTB, 0
        goto allumer

eteindre
        ; Broche 2 du PORTB à l'état haut
        bcf  PORTB, 1
        goto boucle

allumer
        ; Broche 2 du PORTB à l'état haut
        bsf  PORTB, 1
        goto boucle

end
```



TD3 :

Prise en main des logiciels ISIS et MPLAB

Sources

Bigonoff :
http://www.abcelectronique.com/bigonoff/

Jacky Senlis :
http://jgsenlis.free.fr/

Marc Allain & Julien Marot :
http://www.lsis.org/master/ancien_site/documents/micro/Cours.pdf