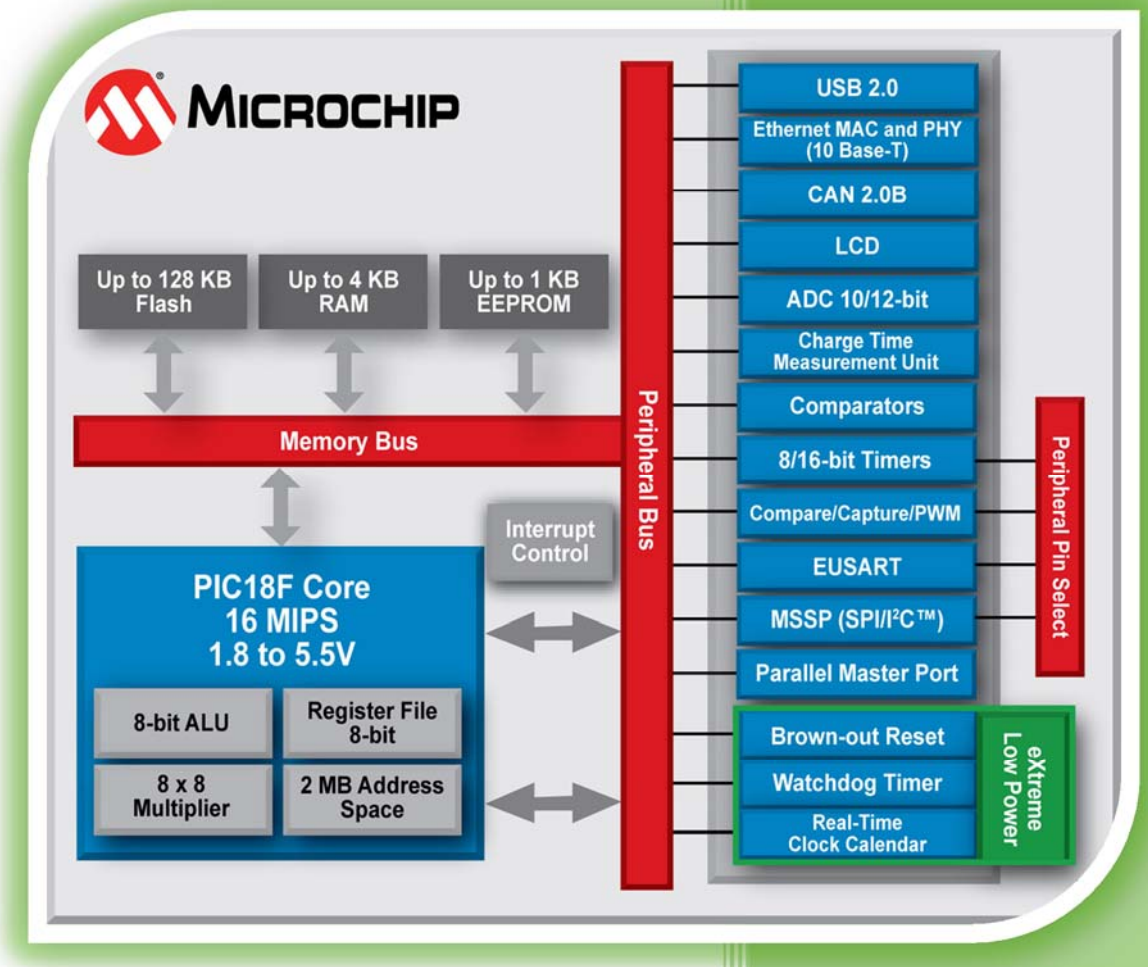
Institut Supérieur des Etudes Technologiques

de Sousse

2014

Support de cours microcontrôleur PIC18F4520



Ali Hmidene

I- Introduction aux microprocesseurs

1. Structure de base d'un calculateur

D'un point de vue matériel, un système informatique minimal est constitué d'un processeur, des mémoires et d'un ensemble des circuits d'interfaces.

Le microprocesseur (MPU, *Micro-Processing Unit)* élément de base, est un circuit intégré à très grande échelle d'intégration (VLSI) capable d'exécuter automatiquement des instructions (opérations élémentaires) qu'il ira chercher dans la mémoire. Il remplit donc les fonctions d'une unité centrale de traitement (CPU, *Central Processing Unit)* en un seul boîtier.

Toutes les informations qu'utilise le microprocesseur sont stockées dans des mémoires, en particulier le programme ; le fonctionnement du microprocesseur est entièrement conditionné par le contenu de celles-ci. Ces mémoires contiennent deux types d'informations : le **programme** et les **données** nécessaires pour la réalisation d'une tâche précise.

- Le programme (ensemble des ordres à exécuter) placé de façon définitive dans des mémoires qui ne pourront qu'être lues par le microprocesseur. Ce sont des mémoires non volatiles, de type **ROM** (Read Only Memory), **PROM, EPROM, Flash** …

- Les données ou OPERANDES proviennent le plus souvent, d'un calcul effectué par le microprocesseur ou d'un périphérique d'entrée, (clavier, disque, ...). Elles sont stockées dans des mémoires qui peuvent être lues et écrites appelées **RAM** (Random Acces Memory).

Dans un système à microprocesseur, l'interface d'entrée-sortie permet d'assurer la liaison entre l'unité centrale de traitement et l'environnement extérieur (périphériques). Le microprocesseur échange les informations avec les composants qui lui sont associés (mémoire et périphériques I/O) au moyen d'un ensemble des lignes de connexions appelé ***bus***.

Un bus est un ensemble de fils qui assure la transmission du même type d'information. On distingue trois types de bus véhiculant les informations dans un système de traitement à microprocesseur :

- **Le bus de données** : bidirectionnel, assure le transfert des informations entre le microprocesseur et son environnement. Le nombre de lignes du bus de données définie la capacité de traitement du microprocesseur ; selon le microprocesseur la largeur du bus peut être de 8 bits, 16 bits, 32 bits, 64 bits …

- **Le bus d'adresses :** unidirectionnel, permet la sélection de la case contenant l'information à traiter dans un *espace mémoire* (ou *espace adressable*). L'espace adressable peut avoir 2^n emplacements, avec *n* est le nombre de lignes du bus d'adresses.

- **le bus de contrôle** : constitué de quelques lignes, assure la synchronisation du flux d'informations sur les bus de données et d'adresses.

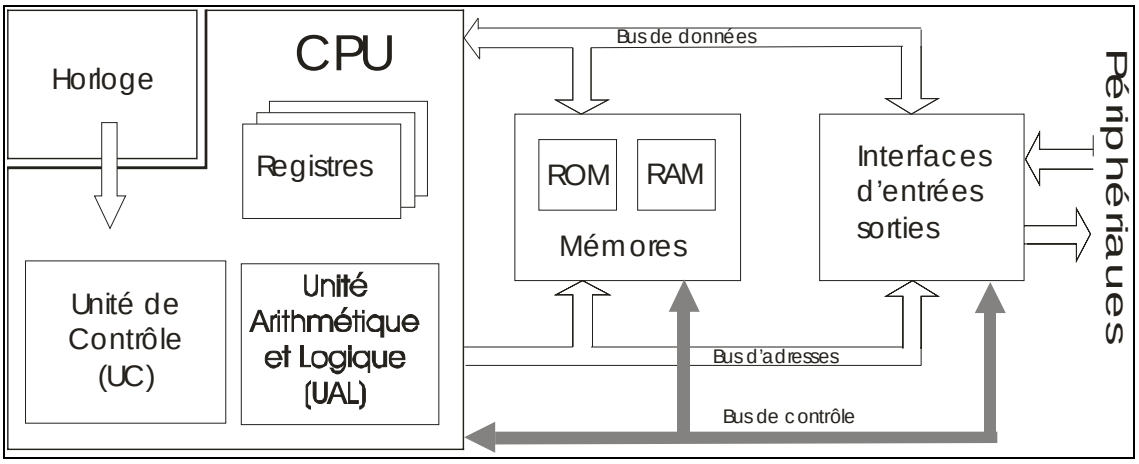


Figure I-1 : Schéma bloc d'un système à microprocesseur

2. Architecture d'un CPU

Le microprocesseur est constitué des unités fonctionnelles suivantes :

- Unité Arithmétique et Logique (UAL ou ALU en anglais),
- des registres,
- une unité de contrôle (CU).

2.1. Unité arithmétique et logique

Cet organe, interne au microprocesseur, permet la réalisation des opérations arithmétiques (Addition, Soustraction…) et logiques (AND, OR, XOR…). Outre les opérations arithmétiques et logiques, l'ALU réalise aussi les opérations de décalage et de rotation. Le registre d'état lié étroitement à l'ALU, met à la disposition du programmeur (à travers ces indicateurs : Flags) des renseignements supplémentaires sur le résultat de quelques opérations (résultat nul, négatif, dépassement…). Citons quelques indicateurs que nous retrouvons dans la plupart des microprocesseurs :

- bit de retenue (carry : **C**) : mis à 1 si le résultat d'une opération dépasse la taille du registre de destination.
- bit de retenue intermédiaire (Auxiliary-Carry : **AC**) : retenue sur le quatrième bit (utilisé pour le calcul en BCD).
- bit de signe (Sign : **S**) : mis à 1 si le résultat d'une opération arithmétique est négative.
- bit zéro (**Z**) **:** indique si le résultat d'une opération est nul.
- bit de débordement (overflow : **OV**) : mis à 1 si une opération arithmétique a écrasé le bit de signe.

Pour monter l'utilité de ces indicateurs, prenons l'exemple d'une instruction conditionnelle :

```
Si (A = B) alors
      Traitement 1
Sinon
      Traitement 2
```

La comparaison entre à **A** et **B** est traité par le microprocesseur de la manière suivante :

- Soustraire **B** de **A**
- Si l'indicateur **Z = 1** (c-à-d **A – B = 0**) alors exécution du **Traitement 1**
- Sinon (**Z = 0**) alors exécution du **Traitement 2**

2.2. Les registres

Il existe deux types de registres : les registres à usage général, et les registres d'adresses (ou pointeurs).

2.2.1. Les registres d'usage général (Registres de travail)

Ce sont des mémoires rapides à l'intérieur du microprocesseur ; ils permettent à l'UAL de manipuler des données à vitesse élevée.

L'adresse d'un registre est associée à son nom (on donne généralement comme nom une lettre **A**, **B**, **C**…).

2.2.2. Les registres d'adresses (pointeurs)

Ce sont des registres connectés sur le bus adresses. Ils sont utilisés pour l'adressage de la mémoire. On peut citer comme registres :

- Le compteur de programme **PC** (appelé aussi compteur ordinal) : le microprocesseur utilise ce registre pour repérer l'instruction à exécuter à un instant donné. Celui-ci contient toujours l'adresse de la prochaine instruction à exécuter.

- Le pointeur de pile (Stack Pointer **SP**), pointe toujours le sommet de la pile. La pile est une partie de la mémoire de données de type **LIFO** (Last In First Out) utilisée pour sauvegarder l'adresse de retour d'un sous-programme et/ou des variables utilisateurs.

- Les registres pointeurs de données ou d'index : utilisés pour l'adressage indirect de la mémoire de données.

2.3. L'unité de contrôle (UC)

Elle permet de séquencer le déroulement des instructions. Elle effectue la recherche en mémoire des instructions, le décodage et l'exécution de l'instruction recherchée. Elle est composée essentiellement :

- d'un *registre d'instruction* (RI), recevant le code de l'instruction à exécuter.

- d'un *décodeur d'instruction*, permettant de déterminer le type de l'instruction à exécuter.

- d'un *Bloc logique de contrôle (ou séquenceur)* : Il organise toutes les étapes d'exécution des instructions au rythme d'une horloge et élabore tous les signaux de synchronisation internes et externes du microprocesseur.

3. Fonctionnement d'un microprocesseur

Le programme est une suite d'instructions stockées dans la mémoire. Pour exécuter ces instructions de manière séquentielle, le microprocesseur utilise un registre appelé compteur de programme (appelé aussi compteur ordinal ou pointeur d'instruction). Ce registre contient l'adresse de l'instruction à exécuter.

3.1. Jeu d'instructions

Les microprocesseurs sont capables d'effectuer un certain nombre d'opérations élémentaires. Cet ensemble d'opérations élémentaires est appelé **jeu d'instructions**. Une instruction au niveau machine doit fournir à l'unité centrale toutes les informations nécessaires pour déclencher une telle opération élémentaire. Elle comporte en général plusieurs champs ; le premier champ contient le code de l'opération (***Code-Op*** ou ***Op-Code*** en anglais) ; les autres champs peuvent comporter des données ou l'identificateur des opérandes. Sur certaines machines toutes les instructions ont la même longueur, sur d'autres cette longueur est variable.

Format d'une instruction :

Code de l'opération	opérande1	,	opérande2

Exemple : Additionner le contenu du registre A avec le contenu de la case mémoire dont l'adresse est mentionnée dans le champ 3

ADDITIONNER	Registre A	,	@ Case mémoire

3.2. Cycle d'exécution d'une instruction

Lors de son exécution, une instruction est décomposée en mini-opérations élémentaires. Celles-ci sont généralement au nombre de trois : *Recherche de l'instruction*, *Décodage* et *Exécution*.

3.2.1. Recherche de l'instruction (Fetch)

Le contenu de PC (compteur de programme) est placé sur le bus des adresses (c'est l'unité de contrôle qui établit la connexion). L'unité de contrôle (UC) émet un ordre de lecture de la case mémoire dont le contenu sera ensuite acheminé à travers le bus de données au registre instruction (RI).

Remarque : Dès la mise sous ou après un RESET, le compteur de programme (PC) est initialisé par une valeur (adresse) fixée par le constructeur du microprocesseur. C'est l'adresse du début de programme.

3.2.2. Décodage (Decode)

Le registre d'instruction (RI) contient maintenant le **code opératoire**. L'unité de contrôle décode le contenu de RI pour savoir la nature de l'opération à effectuer (addition, rotation,...) et incrémente le compteur ordinal (PC) pour pointer sur l'instruction suivante.

3.2.3. Exécution (Execute)

Le cycle d'exécution varie en fonction et l'architecture et le type de l'instruction. Mais d'une manière générale c'est l'ALU qui exécute l'instruction en cours et positionne les indicateurs du registre d'état.

La Figure I-2 illustre la séquence de déroulement de ces étapes. Chaque instruction est équivalente à une suite de mini-opérations exécutées dans un ordre précis. C'est ainsi qu'un microprocesseur procède pour l'exécution de chaque instruction.

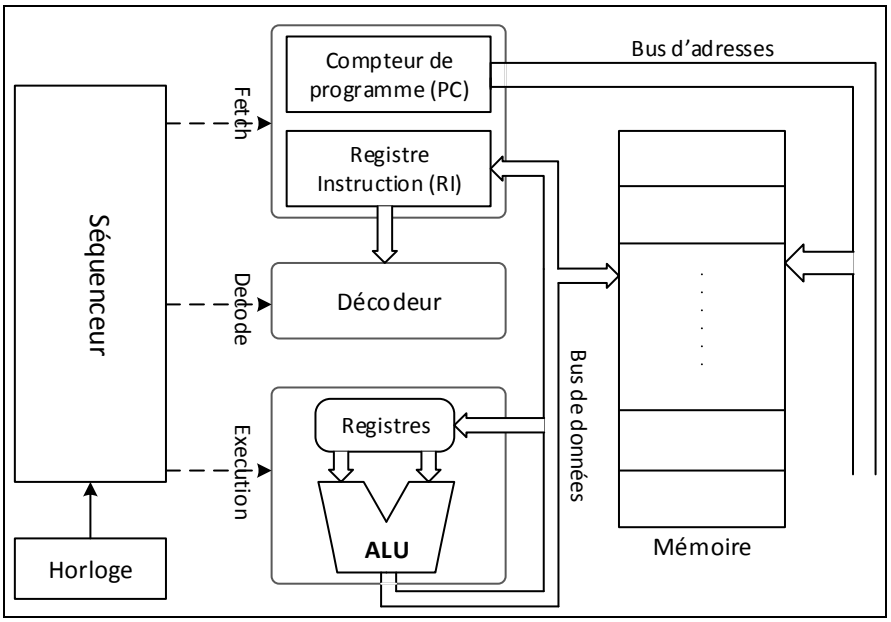


Figure I-2 : Etapes d'exécution d'une instruction

Ces micro-opérations (fetch, decode, execute) sont cadencées au rythme d'horloge qui pilote le séquenceur. La durée de traitement d'une instruction s'appelle cycle d'instruction ou cycle machine.

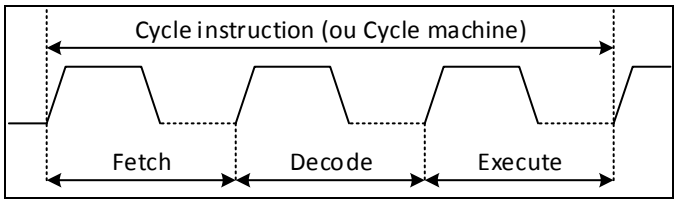


Figure I-3 : Cycle d'exécution d'une instruction

Le nombre de périodes d'horloge nécessaires à l'exécution d'une instruction dépend de l'architecture du processeur et du mode d'adressage. Le microcontrôleur PIC qu'on va étudier, exécute toutes les instructions (hormis les instructions de saut) sur quatre périodes d'horloge.

4. Différentes architecture

Pour l'organisation des différentes unités, il existe deux architectures :

- **l'architecture Von Neuman** (du nom d'un des savants qui a contribué à la mise au point des premiers ordinateurs). La mémoire programme, la mémoire de données et les périphériques d'entrées/sorties partagent le même bus s'adresses et de données.
- **l'architecture Harvard**, sépare systématiquement la mémoire de programme de la mémoire de données : l'adressage de ces mémoires est indépendant. Ce type d'architecture favorise l'accès simultané aux mémoires de programme et de données.

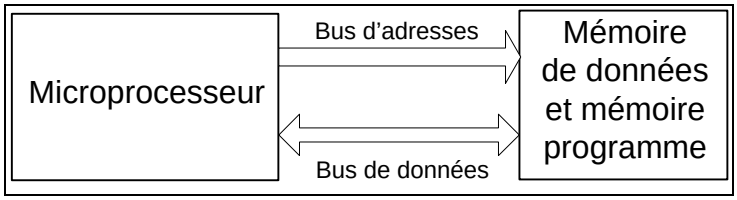


Figure I-4 : Architecture Von Neuman

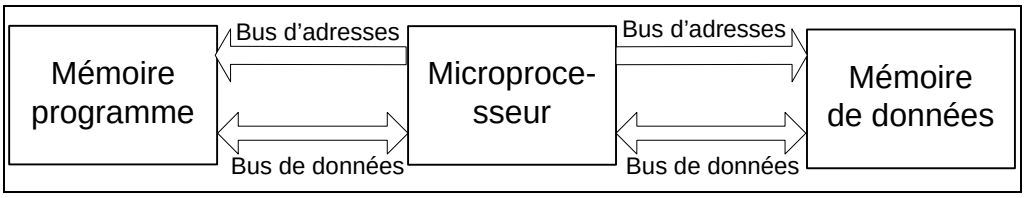


Figure I-5 : Architecture Harvard

Au niveau de jeu d'instructions, les microprocesseurs se réparties en deux grandes catégories appelées CISC et RISC :

- **Architecture CISC** (Complex Instruction Set Computer) : une instruction peut effectuer plusieurs opérations élémentaires (faire par exemple une opération arithmétique avec chargement du résultat dans la mémoire). La longueur de l'instruction et le temps d'exécution varie d'une instruction à l'autre.

- **Architecture RISC** (Reduced Instruction Set Computer) : les processeurs RISC possèdent un jeu d'instruction réduit où chaque instruction effectue une opération élémentaire. Seules les instructions *Load* et *Store* accèdent à la mémoire. La plupart des instructions ont la même taille et s'exécute sur un seul cycle d'horloge.

5. Les processeurs spécialisés

5.1. Les microcontrôleurs

Au début de la commercialisation des microprocesseurs, un système minimum était obligatoirement constitué de plusieurs circuits intégrés.

Les microcontrôleurs sont apparus suite au progrès considérable de l'intégration des composants. Un microcontrôleur regroupe dans un même boîtier, un CPU, une RAM, une ROM et des périphériques (PORTS, ADC, Timers ...).

Les microcontrôleurs sont dédiés aux applications qui ne nécessitent pas une grande quantité de calculs complexes, mais qui demandent beaucoup de manipulations des entrées/sorties. C'est le cas de contrôle des processus.

5.2. Les processeurs de traitement de signal (DSP : Digital Signal Processor)

Les DSP sont des processeurs optimisés pour exécuter des applications de traitement numérique de signal (filtrage numérique, convolution, transformé de Fourrier rapide…). Les DSP sont employés dans les modems, les téléphones mobiles, les appareils multimédia …).

6. Les mémoires

Les mémoires sont des circuits intégrés à grande échelle d'intégration, capables de sauvegarder des informations *binaires* de façon permanente ou temporaire. Elles sont liées étroitement aux microprocesseurs puisqu'elles constituent l'élément de stockage de premier niveau.

On distingué deux type de mémoires :

- Les mémoires vives (RAM : Random Access Memoy) : Ce sont des mémoires volatiles, le maintien de l'information dépend de la présente de l'alimentation. Toute coupure de l'alimentation provoque la perte des informations. Elles sont utilisées pour stocker généralement les données temporaires.

- Les mémoires mortes (ROM : Read Only Memory) : gardent les informations même en absence d'alimentation. Ces mémoires contiennent des informations figées (souvent des programmes) et que l'accès ne se fait qu'on lecture seule.

6.1. Schéma synoptique d'une mémoire

Une mémoire est accessible via, un bus d'adresses (qui définit sa capacité), un bus de données et les signaux de lecture et d'écriture.

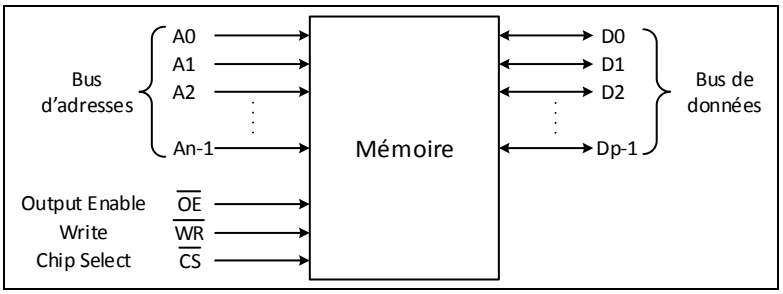


Figure I-6

La capacité de la mémoire est donnée par la relation $C = 2^n$ avec *n* est le nombre de lignes d'adresses. Elle est exprimée en KiloBits ou KiloOctets (1K = 1024).

6.2. Cycles de Lecture / Ecriture

Cycle de lecture : Le cycle de lecture commence à l'instant où le microprocesseur met l'adresse de la case mémoire sur le bus, cette adresse doit rester stable pendant la durée totale du cycle t_{RC} (Read Cycle Time). Le circuit de décodage met la ligne $\overline{OE}$ à l'état bas indiquant la présence d'une adresse stable sur le bus. Le mémoire répond après une durée appelée temps d'accès t_{AA} (Adress Acces time) en plaçant le contenu de la case sur le bus de données. Le paramètre t_{OE} (Output Enable) définit le temps nécessaire à la mémoire pour mettre une donnée stable sur le bus. A partir de cet instant la donnée peut être récupérer. Après le retour de de la ligne $\overline{OE}$ à l'état haut, la mémoire met un temps t_{OD} pour mettre le bus à l'état haute impédance.

Cycle d'écriture : Le cycle d'écriture est similaire à celui de lecture. Le microprocesseur fournit l'adresse et la donnée à mémoriser. Après la stabilisation de l'adresse, la ligne $\overline{WR}$ est forcée à l'état bas. Pendant la durée de cette impulsion t_{PW}, l'adresse et la donnée doivent maintenues stables.

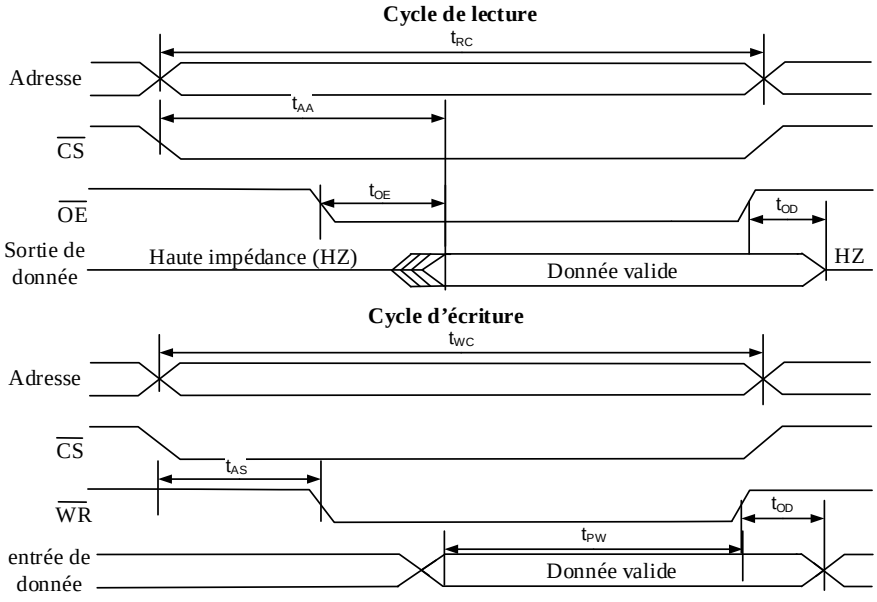


Figure I-7

II- MICROCONTROLEUR PIC18F4520

1. Présentation

Les microcontrôleurs PICs sont des composants dits RISC (Reduced Instructions Set Computer), ou encore composants à jeu d'instructions réduit. La famille PIC18 présente des différences assez notables avec la famille précédente. Compte tenu de leur richesse en ressources matérielles internes et de capacité très importante en mémoires, ces circuits sont bien adaptés à la programmation en langages évolués (Basic, C, Pascal…).

Les microcontrôleurs PIC18 font partie de la famille «High-Performance », leurs instructions sont codées sur 16bits, excepté de quelques-unes codées sur 32bits.

Nous allons dans ce document étudier le microcontrôleur PIC18F4520.

2. Caractéristiques du PIC18F4520

2.1. Caractéristique de la CPU

- CPU à architecture RISC (8 bits)
- Adressage de la mémoire de programme sur 21 bits (32 KOctets de mémoire flash intégré)
- Adressage de la mémoire de données sur 12 bits (SRAM de 1536 Octets intégré)
- EEPROM de données de 256 Octets,
- Interruptions à 2 niveaux de priorité,
- Pile de 31 niveaux,
- Vitesse d'exécution jusqu'à 10MIPS,
- Instructions codées sur 16bits
- Multiplieur 8x8 bits
- Chien de garde (WatchDog),

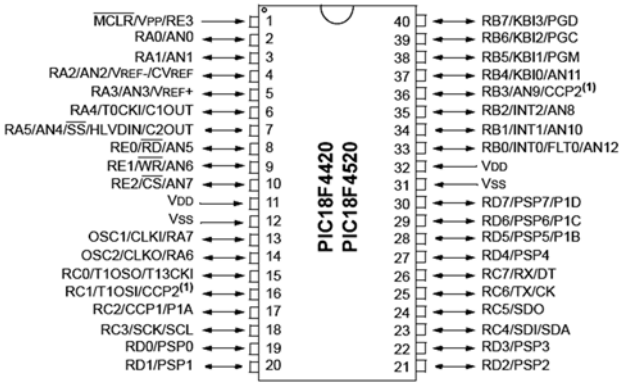


Figure II-1: brochage du PIC18F4520

2.2. Caractéristiques des périphériques

- 5 ports d'entrées/sorties bidirectionnels (A, B, C, D et E),
- 4 Timers
- Deux modules CCP « Capture, Compare et PWM »
- ADC à 13 entrées, avec une résolution de 10 bits,
- Deux comparateurs analogiques,
- Détection de niveau de tension (HLVD)
- Master Synchronous Serial Port (MSSP) travaillant en mode SPI et en mode I2C,
- Enhanced Universel Synchronous Asynchronous Receiver Transmitter (EUSART) avec un 9ème bit pour la détection d'adresse,
- Parallel Slave Port (PSP) de 8 bits avec signaux de contrôle externes RD\, RW\ et CS\.

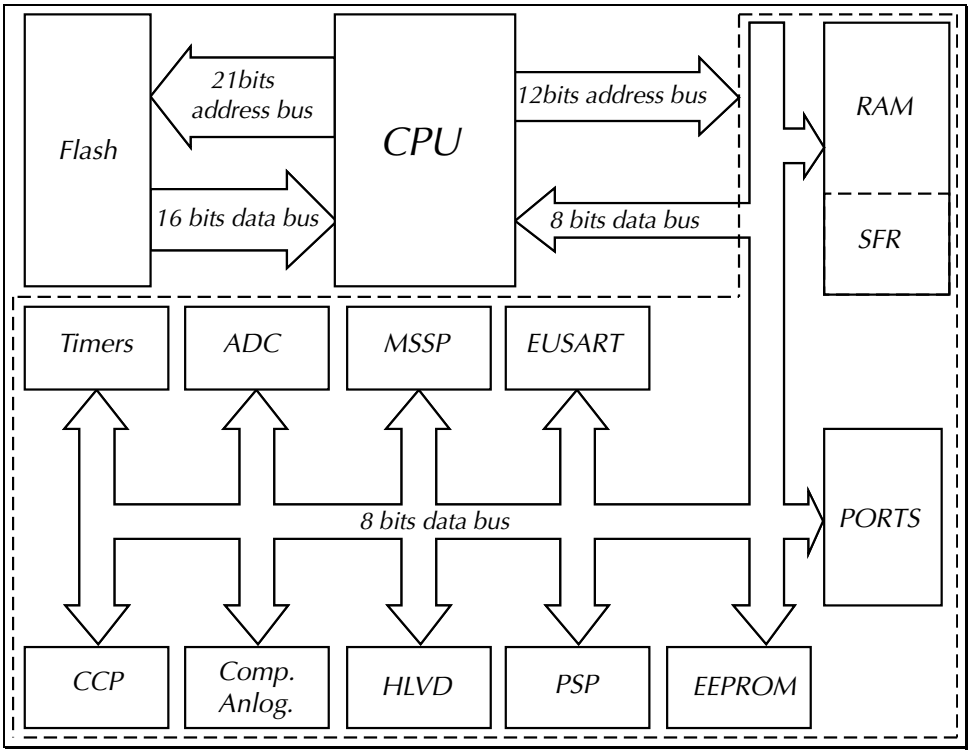


Figure II-2 : Schéma bloc du microcontrôleur PIC18F4520

3. Architecture interne des microcontrôleurs PIC18

Les microcontrôleurs de la famille PIC18 se basent sur une architecture Harvard, cette architecture permet une accessibilité séparée de la mémoire programme et celle de données. Le microcontrôleur PIC comporte un ensemble de registres répartis en deux catégories : des registres rattachés à l'unité centrale et les registres associés aux périphériques. Tous les registres sont implantés dans la mémoire de données. Celle-ci est organisée en octets, par conséquent, tous les registres sont de huit bits. Quelques registres sont concaténés pour former des registres 16 bits ou plus.

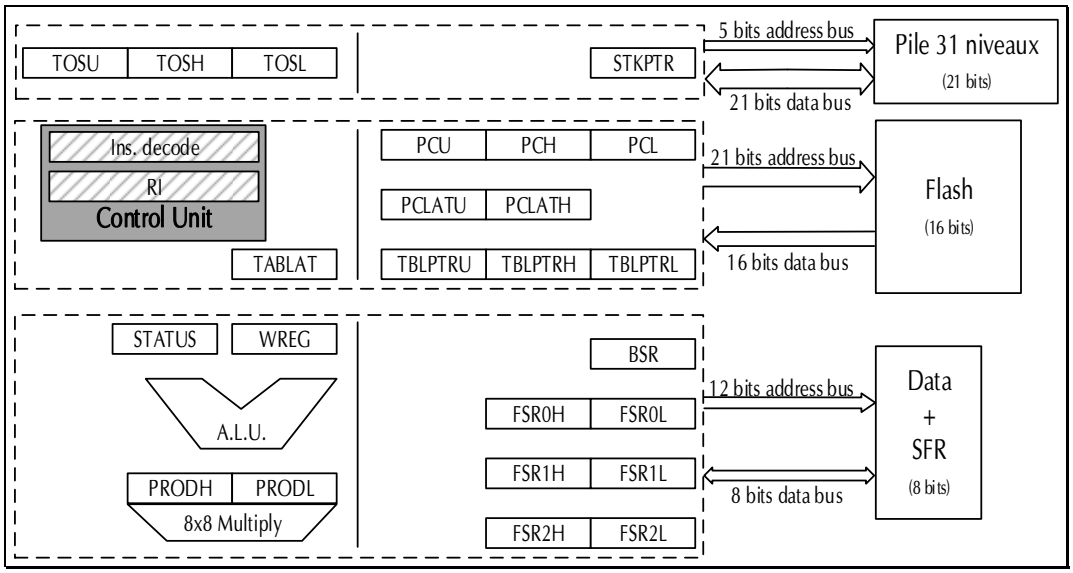


Figure II-3 : Organisation interne du PIC18F4520

Les registres de l'unité centrale sont organisés en trois groupes :

Registres pointeurs de la mémoire programme :

- Compteur de programme PC, contient l'adresse de la prochaine instruction à exécuter. La mémoire programme est adressée sur 21 bits, le compteur de programme est formé donc de trois registres PCL, PCH et PCU.
- Table Pointer TBLPTR, il est souvent utile de sauvegarder des constantes dans la mémoire programme (tel que les messages d'invites…). Le registre TBLPTR (formé par TBLPTRL, TBLPTRH et TBLPTRU) est utilisé pour adresser les données dans la mémoire programme, et TBALAT comme registre da donnée.

Registres de gestion de la pile

- Pointeur de pile STKPTR, est un registre de 5 bits, utilisé pour l'adressage d'une pile de 31 niveaux. Cette pile dispose de son propre espace mémoire, destinée principalement pour la sauvegarde des adresses de retour des sous-programmes.
- Le registre TOS (composé de trois registre TOSL, TOSH, TOSU) sert à charger (ou récupérer) les données utilisateur dans (ou à partir de) la pile. Ce procédé permet l'implémentation d'une pile logicielle.

Registres rattachés à l'unité d'exécution

- BSR (Bank Select Register), est utilisé avec la partie basse de l'instruction pour l'adressage direct de la mémoire de données.
- FSR0, FSR1 et FSR2 sont de registres de 12 bits utilisés comme pointeur de la mémoire de données.
- Le registre WREG : le seul registre à usage générale, il sert pour le traitement temporaire des données.
- Le registre PROD (PRODL, PRODH), ce registre reçoit le résultat du multiplieur câblé.
- Le registre STATUS : décrit l'état du microprocesseur. Il renseigne à travers de bits (nommés (Flags) de l'état de quelques instructions. La fonction de ces indicateurs est donnée comme suit :

Figure II-4 : Registre STATUS du microcontrôleur PIC18

C : indicateur de retenue (carry), la polarité de ce bit est inversée en cas d'un emprunt.

DC : indicateur de retenue auxiliaire, la polarité de ce bit est inversée en cas d'un emprunt.

Z : indicateur d'un résultat nul.

OV : indicateur de dépassement, ce bit est utilisé par les opérations arithmétiques signées.

N : indicateur de signe.

Registres de contrôle et de configuration

Le microcontrôleur PIC18F4520 comporte un ensemble de registres de contrôle et de configuration. Nous citons à titre indicatif les registres RCON, WDTCON, les registres CONFIGx, les registres DEVID et les registres ID.

4. Horloge du microcontrôleur PIC18

Le microcontrôleur PIC18F4520 peut opérer avec 10 sources d'horloges différentes. Les 4 bits du registre **FOSC** permettent la sélection de l'un des modes d'horloges.

- LP : faible consommation (32Khz)

- XT : Quartz/Résonateur (1 à 4Mhz)

- HS : Haute Vitesse, Quartz/Résonateur (4 à 25Mhz)

- HSPLL : Haute Vitesse, Quartz/Résonateur avec PLL (Multiplication de fréquence par 4). Dans ce cas la fréquence du quartz ne doit pas dépasser 10 Mhz.

- RC : Circuit RC externe, FOSC/4 est fournie sur la broche RA6

- RCIO : Circuit RC externe, la broche OSC2/RA6 peut être utilisée en I/O

- INTIO1 : Oscillateur interne, FOSC/4 est fournie sur la broche RA6 et RA7 peut être utilisée en I/O

- INTIO2 : Oscillateur interne, les broches RA6 et RA7 peuvent être utilisées en I/O

- EC : Horloge externe appliquée à l'entrée OSC1/RA7, FOSC/4 est fournie sur la broche OSC2/RA6

- ECIO : Horloge externe appliquée à l'entrée OSC1, la broche OSC2/RA6 peut être utilisée en I/O

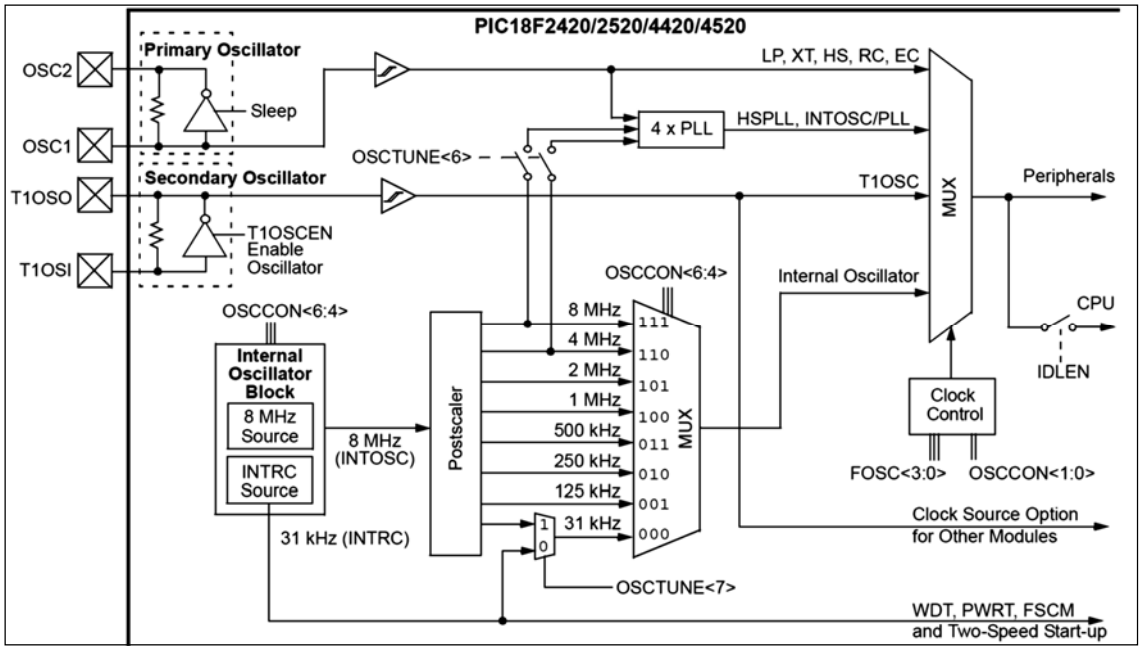
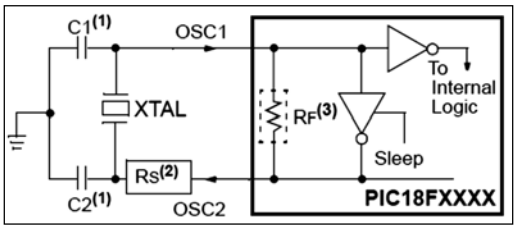
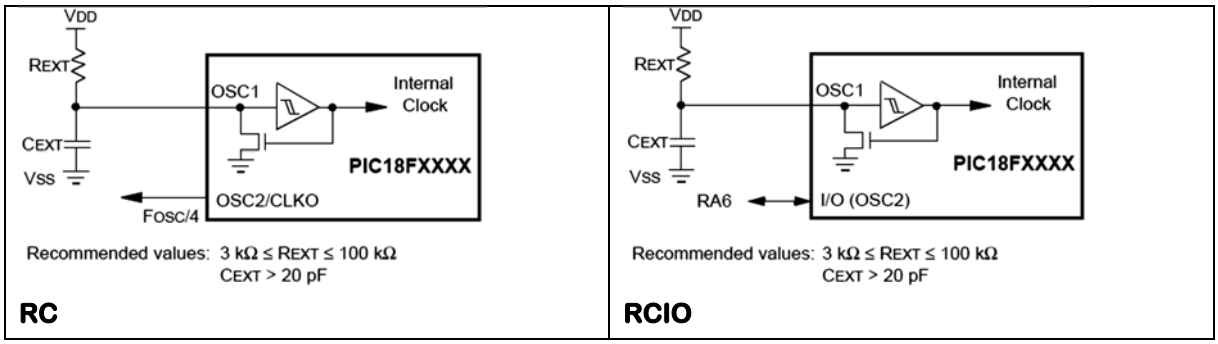


Figure II-5 : circuiterie d'horloge

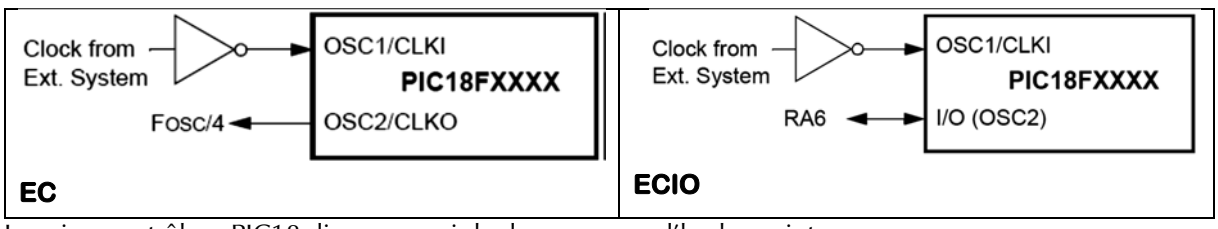
LP, XT, HS, HSPLL : ces différents modes s'appliquent à des oscillateurs à quartz ou des résonateur céramiques connectés entre les broches OSC1 et OSC2.



Les modes RC ou RCIO sont obtenus par l'ajout d'un circuit RC externe.



EC et ECIO sont obtenus par l'application d'un signal d'horloge à l'entrée OSC1



Le microcontrôleur PIC18 dispose aussi de deux sources d'horloges internes :

- un oscillateur interne délivrant une fréquence de 8 MHz, cette fréquence peut être utilisée telle quelle ou être réduite après passage par un diviseur programmable délivrant ainsi des fréquences allant de 31Khz à 8 Mhz.
- Un oscillateur interne de type RC, fonctionnant à une fréquence fixe de 31Khz. Cette fréquence est utilisée lorsqu'on veut faire fonctionner le circuit en mode faible consommation ; de même elle est utilisée par le *Watchdog*, par le circuit *Power on timer* (PWRT) au démarrage ou par le circuit de détection de défaillance de l'horloge principale (FSCM : *Fail Safe Clock Monitor*).

5. Circuit RESET

Les microcontrôleurs PIC18 disposent de huit sources de RESET distinctes :

- Une RESET à la mise sous tension appelé **POR** (Power On Reset)
- Une RESET par la mise de la ligne $\overline{MCLR}$ à la masse pendant le fonctionnement normal.
- Une RESET par la mise de la ligne $\overline{MCLR}$ à la masse pendant le fonctionnement en mode sommeil.
- Une RESET par débordement du timer Watchdog
- Une RESET par la détection d'une chute de la tension d'alimentation **BOR** (Brown Out Reset)
- Une RESET logicielle, suite à l'exécution de l'instruction **RESET**.
- Une RESET suite à une opération d'empilement alors que la pile est pleine.
- Une RESET suite à une opération de dépilement alors que la pile est vide.

La Figure II-6 représente le schéma interne du circuiterie de RESET.

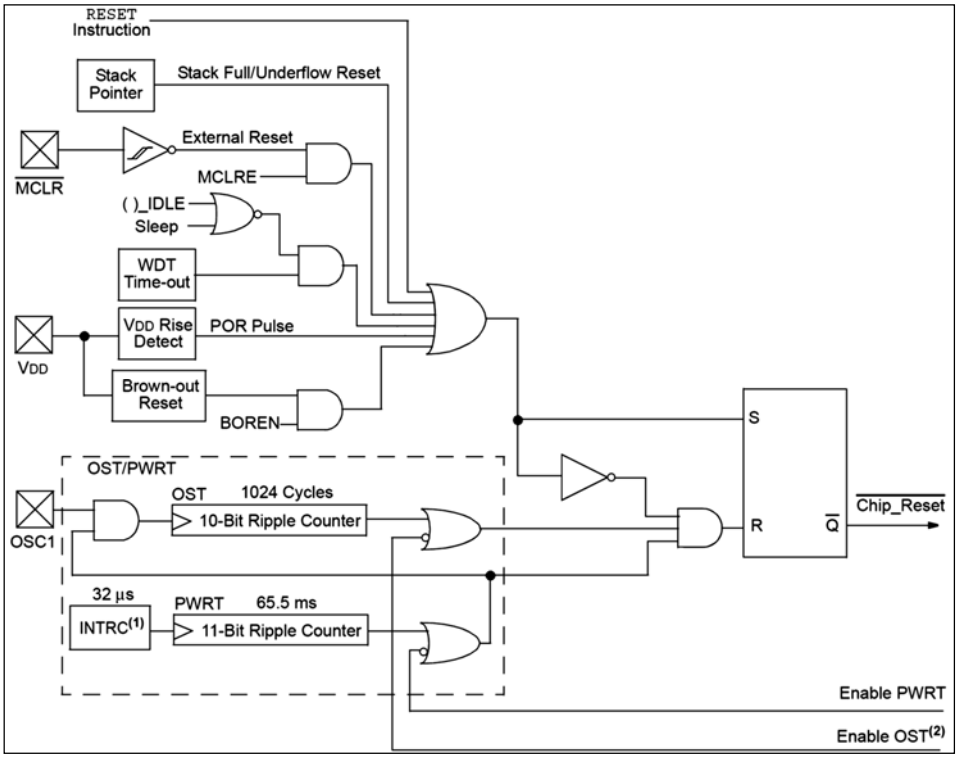


Figure II-6 : Circuiterie RESET

La Figure II-7 représente le chronogramme type d'une séquence de RESET à la mise sous tension (POR).

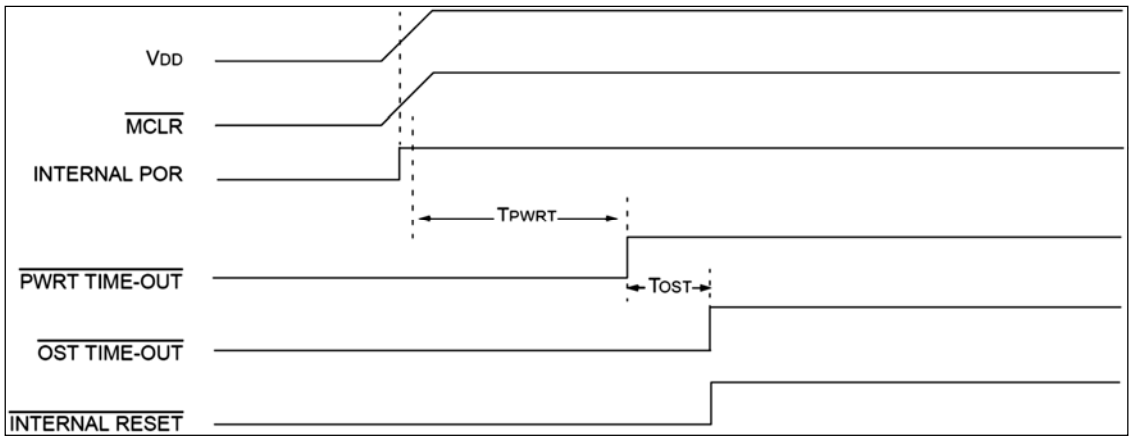


Figure II-7 : Chronogramme type d'une séquence de RESET

6. Organisation de la mémoire

Les microcontrôleurs PIC18 disposent de trois types de mémoires :

- Mémoire de programme (Flash)
- Mémoire de données (RAM)
- EEPROM de données

La mémoire données contient les registres des ressources internes et les variables du programme proprement dites, de même cette mémoire fait appel un mécanisme de pagination qui complique un peu son accès.

La mémoire de programme quant à elle s'étend linéairement de l'adresse la plus basse à l'adresse la plus haute. Contrairement aux microcontrôleurs PIC16, les microcontrôleurs PIC18 n'utilisent aucun mécanisme de pagination de la mémoire Flash.

6.1. Mémoire de programme

Les microcontrôleurs PIC18 disposent d'un compteur de programme (PC) de 21bits, ce qui leurs permet d'adresser un espace maximal de 2Moctets (000000h à 1FFFFFH). La mémoire flash intégrée ne dépasse pas 128ko ; soit de 32Ko de mémoire flash pour le PIC18F4520, l'espace restant est lu comme zéro. Les instructions sont codées sur 16 bits, alors que la mémoire programme est organisée en octets, ce qui explique la parité d'adressage de la mémoire ; par suite le bit LSB du compteur de programme (PC) est toujours nul.

Il est possible aussi de lire ou d'écrire de données dans la mémoire programme en cours d'exécution en utilisant les instructions TBLRD et TBLWT. Dans ce cas le registre TBLPTR est utilisé comme pointeur d'adresse et le registre TABLAT comme registre de donnée. Le registre TBLPTR comporte 22 bits, il est possible donc d'accéder à des emplacements au-delàs de 2Mo. Le 22ème bit permet l'accès aux registres « device ID », « user ID » et « configuration bits »

La Figure II-8 montre, qu'au niveau des instructions, la mémoire est considérée comme étant une mémoire formée des mots de 16 bits (double octets) ; le compteur de programme s'incrémente toujours de deux unités. Le vecteur RESET occupe l'adresse 0000H. Les vecteurs d'interruptions démarrent à l'adresse 0008H ou 0018H, selon type de priorité (haute priorité ou basse priorité).

Lorsque la mémoire est pointée par le registre TBLPTR « Table Pointer », la mémoire est considérée comme une succession des cases de 8 bits. Le registre TBLPTR permet l'accès à la totalité de la mémoire de 000000H à 3FFFFFH.

La zone réservée à « ID », permet à l'utilisateur d'écrire des informations personnelles, tel que le code du produit… Cette zone peut être lue même lorsque la mémoire est protégée.

La zone « configuration bits » est modifiée uniquement lors de la programmation. Nous pouvons configurer à travers ces bits : le mode d'horloge (LP, XT, HS, HSPLL,….), l'activation du WatchDog et son prédiviseur, la protection de la mémoire programme, la validation de la broche MCLR, l'activation de mode debug,…

Ces bits sont configurés à travers la directive « config ».

Exemple de configuration : Oscillateur HS, Watchdog désactivé, programmation faible tension désactivé

En langage assembleur :

```
CONFIG    OSC = HS
CONFIG    WDT = OFF
CONFIG    LVP  = OFF
```

En langage XC8

```
#pragma config OSC = HS, WDT= OFF, LVP = OFF
```

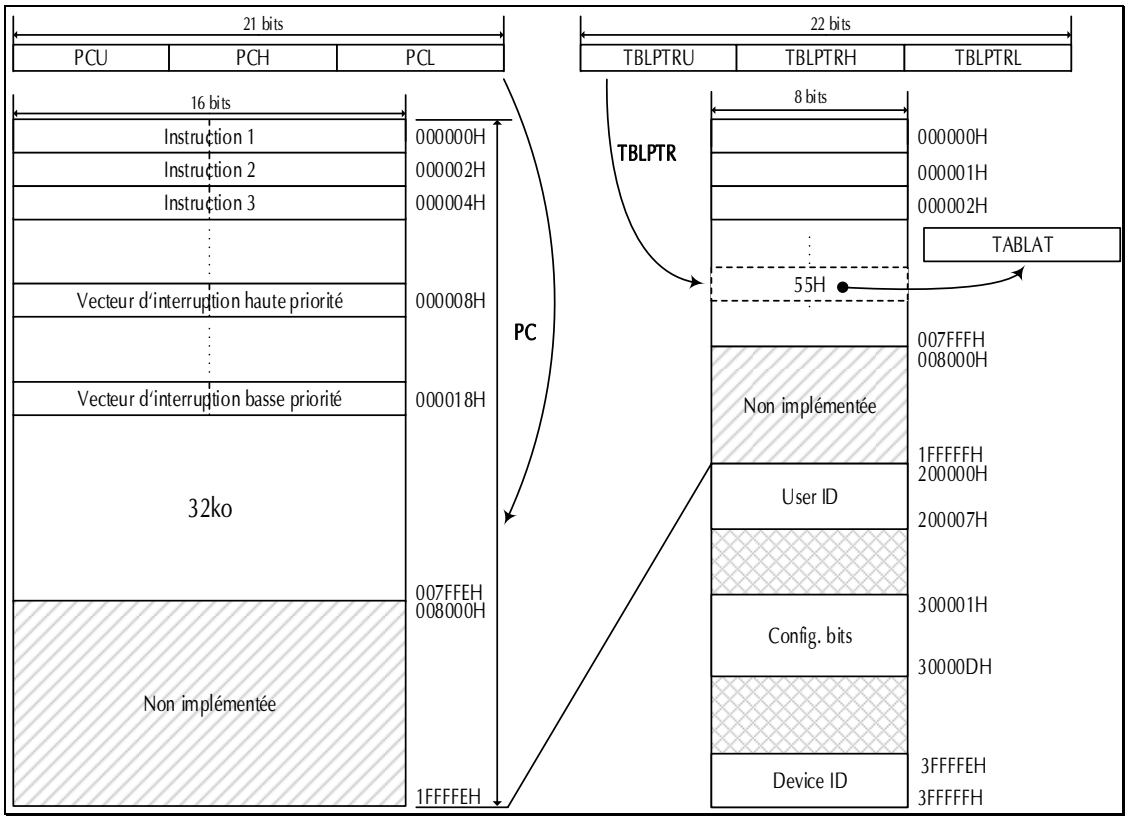


Figure II-8 : organisation de la mémoire programme

6.2. Mémoire de données

La mémoire de données (**F**ile Registers) contient les registres **SFR** (Special Function Registers), et les cases mémoires à usage général appelées **GPR** (General Purpose Register). La zone des registres **SFR** comporte les registres de la CPU et celles des périphériques, tandis que la partie restante **GPR**, constitue la mémoire de stockage des données utilisateur. Elle utilise un mécanisme de pagination (nommé ***bank*** par Microchip) plus ou moins hérité des versions précédentes, mais beaucoup plus souple d'emploi.

La mémoire « File register » est organisée en octets, elle comporte 16 banques de 256 octets chacun, soit un espace maximal de 4Ko (1536 octets pour le microcontrôleur PIC18F4520). La sélection des banques est obtenue par les 4 bits du registre **BSR** (Bank Select Register).

Il est important de savoir que tous les registres **SFR** sont regroupés dans la partie haute de la banque 15 (de F80H à FFFH), mais il est possible d'accéder à ces registres à partir de la banque 0 en utilisant le mode « ***Access bank*** ».

Trois modes possibles sont utilisés pour l'accès à la mémoire :

- « Access Bank », la banque 0 devient une combinaison de 128 octets de la partie basse de la mémoire (00H à 7FH) et 128 octets de la partie haute de la banque 15 (registres **SFR**).
- Accès direct, l'adresse d'une case mémoire est la concaténation de 4 bits du registre **BSR** et les 8 bits provenant de l'instruction, pour former une adresse de 12 bits. Il est possible en utilisant

l'instruction **movff** d'accéder de manière linéaire à la totalité de la mémoire, mais cette instruction est codée sur 32bits.

- Accès indirect en utilisant l'un des registres **FSR0**, **FSR1** ou **FSR2**. Chaque registre est formé de deux registres de 8 bits (FSR0H, FSR0L pour FSR0). L'accès à la totalité de la mémoire se fait d'une manière linéaire et sans aucune contrainte de pagination.

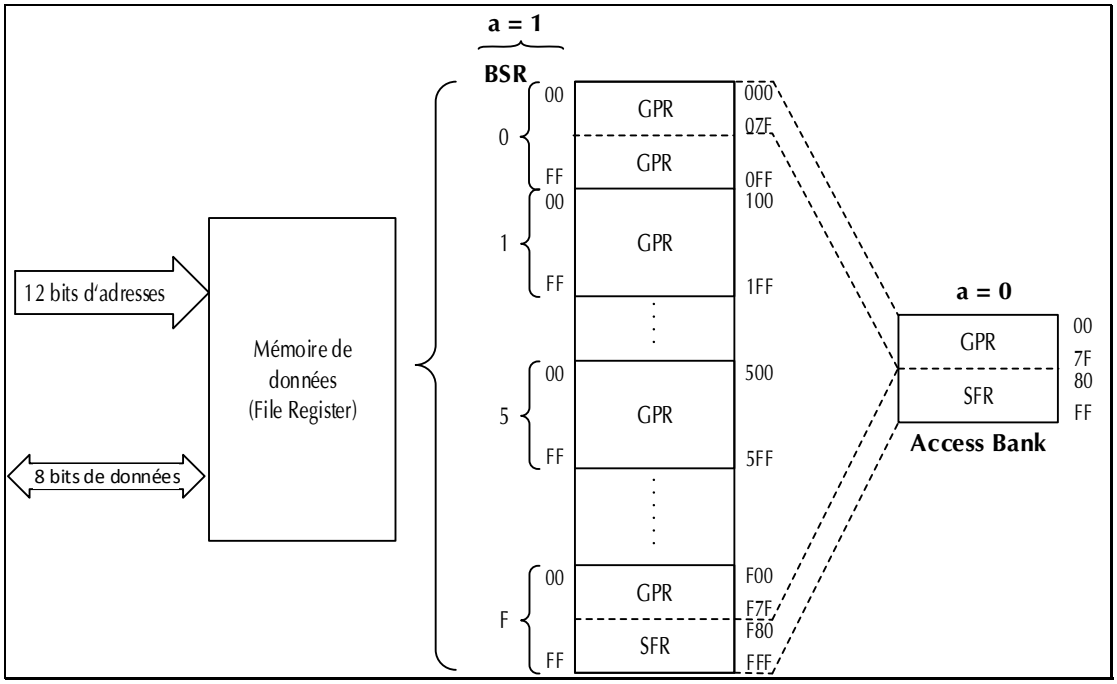


Figure II-9 : Mémoire de données

6.3. Registres SFR du microcontrôleur PIC18F4520

SFR (Special Function Registres) est l'ensemble de tous les registres utilisés par la CPU et par les périphériques. Ces registres sont implémentés dans la RAM à partir de l'adresse F80H jusqu'à FFFH. En mode Accès Bank cette zone est translatée à l'adresse 080H. Nous donnons à Figure II-10 la liste de tous les registres avec leurs emplacements mémoire. Nous remarquons que le registre WREG fait partie des SFR. Les cases grisées sont non implémentées et les cases hachurés représentent des registres virtuels (qui n'existent pas physiquement).

FFFh	TOSU	FDFh	INDF2	FBFh	CCPR1H	F9Fh	IPR1
FFEh	TOSH	FDEh	POSTINC2	FBEh	CCPR1L	F9Eh	PIR1
FFDh	TOSL	FDDh	POSTDEC2	FBDh	CCP1CON	F9Dh	PIE1
FFCh	STKPTR	FDCh	PREINC2	FBCh	CCPR2H	F9Ch	
FFBh	PCLATU	FDBh	PLUSW2	FBBh	CCPR2L	F9Bh	OSCTUNE
FFAh	PCLATH	FDAh	FSR2H	FBAh	CCP2CON	F9Ah	
FF9h	PCL	FD9h	FSR2L	FB9h		F99h	
FF8h	TBLPTRU	FD8h	STATUS	FB8h	BAUDCON	F98h	
FF7h	TBLPTRH	FD7h	TMR0H	FB7h	PWM1CON	F97h	
FF6h	TBLPTRL	FD6h	TMR0L	FB6h	ECCP1AS	F96h	TRISE
FF5h	TABLAT	FD5h	T0CON	FB5h	CVRCON	F95h	TRISD
FF4h	PRODH	FD4h		FB4h	CMCON	F94h	TRISC
FF3h	PRODL	FD3h	OSCCON	FB3h	TMR3H	F93h	TRISB
FF2h	INTCON	FD2h	HLVDCON	FB2h	TMR3L	F92h	TRISA
FF1h	INTCON2	FD1h	WDTCON	FB1h	T3CON	F91h	
FF0h	INTCON3	FD0h	RCON	FB0h	SPBRGH	F90h	
FEFh	INDF0	FCFh	TMR1H	FAFh	SPBRG	F8Fh	
FEEh	POSTINC0	FCEh	TMR1L	FAEh	RCREG	F8Eh	

FEDh	POSTDEC0	FCDh	T1CON	FADh	TXREG	F8Dh	LATE
FECh	PREINC0	FCCh	TMR2	FACh	TXSTA	F8Ch	LATD
FEBh	PLUSW0	FCBh	PR2	FABh	RCSTA	F8Bh	LATC
FEAh	FSR0H	FCAh	T2CON	FAAh		F8Ah	LATB
FE9h	FSR0L	FC9h	SSPBUF	FA9h	EEADR	F89h	LATA
FE8h	WREG	FC8h	SSPADD	FA8h	EEDATA	F88h	
FE7h	INDF1	FC7h	SSPSTAT	FA7h	EECON2	F87h	
FE6h	POSTINC1	FC6h	SSPCON1	FA6h	EECON1	F86h	
FE5h	POSTDEC1	FC5h	SSPCON2	FA5h		F85h	
FE4h	PREINC1	FC4h	ADRESH	FA4h		F84h	PORTE
FE3h	PLUSW1	FC3h	ADRESL	FA3h		F83h	PORTD
FE2h	FSR1H	FC2h	ADCON0	FA2h	IPR2	F82h	PORTC
FE1h	FSR1L	FC1h	ADCON1	FA1h	PIR2	F81h	PORTB
FE0h	BSR	FC0h	ADCON2	FA0h	PIE2	F80h	PORTA

Figure II-10 : Registres spéciaux (SFR) du PIC18f4520

7. Les instructions du microcontrôleur PIC18F4520

Chaque microcontrôleur reconnait un ensemble d'instructions fixé par le constructeur, appelé jeu d'instructions. Les microcontrôleurs PIC18 disposent de 75 instructions ; la plupart de ces instructions sont codées sur 16 bits, à l'exception de quelques-unes codées sur 32bits.

Chaque instruction comporte le code opératoire « ***Code-op*** », qui spécifie le type de l'opération et des **opérandes**. Un opérande peut être : une valeur, ou une adresse, ou signifie qu'on utilise un registre comme intermédiaire ou rien du tout ; Ceci dépend du *mode d'adressage*.

7.1. Modes d'adressage

On peut définir le mode d'adressage, comme étant la façon utilisée par l'instruction pour accéder à l'information à manipuler. Pour mieux comprendre le mode d'adressage, prenons un exemple concret : *Nous voulons charger une information dans le registre* ***WREG***. Il y a plusieurs façons pour parvenir à faire cette opération :

1. L'information à charger dans WREG est une constante dont on connait la valeur (supposons que cette valeur est égale à 20). Dans ce cas, cette valeur constitue l'opérande qui suit le code-op de l'opération de transfert (mov). Ce mode d'adressage est appelé, *adressage immédiat* (nommé « literal » par Microchip). L'instruction s'écrit donc :

   movlw 20 ; (WREG) ⇦ 20

Cette écriture est facile à interpréter : « **mov** » : opération de transfert, «**l**» : literal, l'information se trouve dans l'instruction et **W** : la destination.

La Figure II-11 illustre les étapes d'exécution de l'instruction. Le microcontrôleur charge l'instruction dans le registre **RI** (Fetch) pour le décodage. Dans la phase de l'exécution, la valeur qui se trouve dans RI (20 dans notre exemple) sera transférer à WREG.

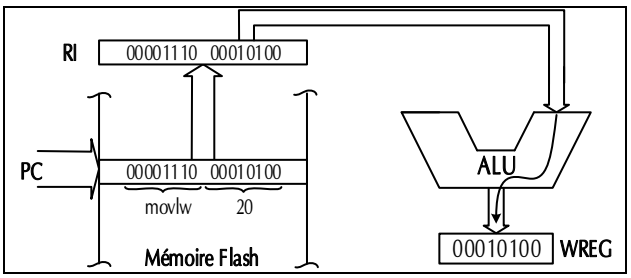


Figure II-11 : Adressage Immédiat "literal"

2. L'information à charger dans WREG se trouve dans une case mémoire dont on connait son adresse. Dans ce cas, l'opérande est interpréter comme étant une adresse. Ce mode d'adressage est appelé *adressage direct*.
   Supposons que la valeur à charger se trouve à l'adresse 0x550 (550 en héxa).
   L'adresse de la case mémoire est la concaténation de 4 bits du registre BSR et le 8 bits provenant de l'instruction.

   movf 0x50, w ; 0x50 c'est l'octet le plus faible de l'adresse

« **mov** » : opération de transfert, «**f**» : l'information se trouve dans la mémoire de données (**F**ile registers), 0x50 : c'est la partie basse de l'adresse de la case mémoire et **W** : la destination.

(WREG) ⇦ (0x550) : les parenthèses désigne, le contenu de la case mémoire d'adresse 0x550.

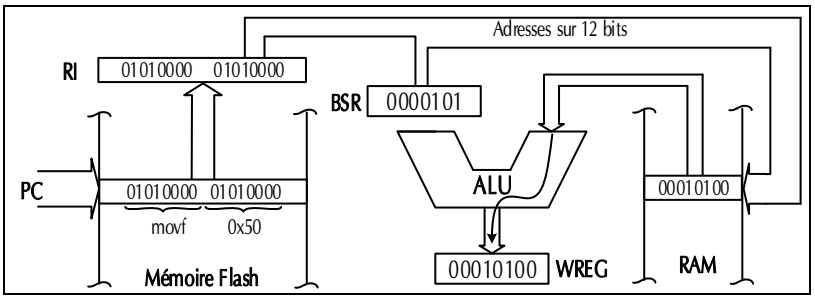


Figure II-12 : Adressage direct

3. L'information à charger dans WREG se trouve dans une case mémoire, mais l'adresse se trouve aussi dans un registre particulier (appelé pointeur de données). Le microcontrôleur doit charger dans WREG le contenu de la case mémoire dont l'adresse se trouve dans le registre pointeur de données. On appelle ce mode : *adressage indirect*.
   Le microcontrôleur PIC18 dispose de trois registres pointeur de données **FSR0**, **FSR1** et **FSR2**. Supposons que l'adresse de la case mémoire (0x550 dans notre exemple) est chargée dans FSR0.
   movf INDF0, w ; WREG ⇦ ((FSR0))

« **mov** » : opération de transfert, «**f**» : l'information se trouve dans la mémoire de données (**F**ile register), **INDF0** (INDirecf File) : c'est un registre virtuel utilisé pour accéder à **FSR0**, et **W** : la destination.

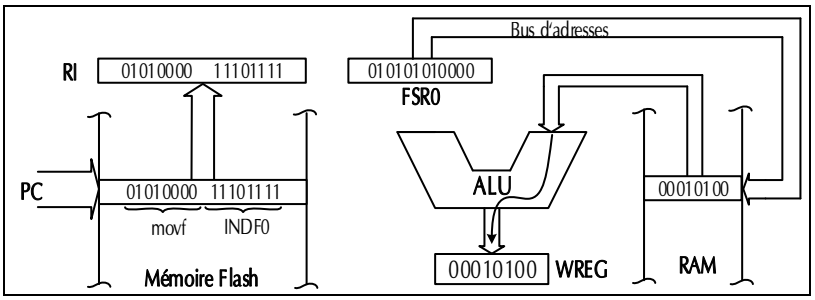


Figure II-13 : Adressage indirect

4. Pour terminer, il reste un dernier mode d'adressage qui ne s'applique pas à notre exemple : *l'adressage inhérent*. Dans ce mode, l'instruction n'a pas besoin d'opérande ; elle utilise uniquement le code-op.

Exemple d'instructions à adressage inhérent :

- RESET : initialise le microcontrôleur,
- SLEEP : met le microcontrôleur en mode veille
- CLRWDT : réinitialise le compteur du Watchdog
- PUSH : charge une valeur dans la pile ; la valeur se trouve par défaut dans le registre TOS
- POP : l'inverse de PUSH, restaure une valeur de la pile
- NOP : not opération ; le microcontrôleur passe un cycle machine sans rien faire.

7.2. Le jeu d'instructions

Microchip a organisé les instructions du microcontrôleur PIC18 en quatre groupes :

- Les opérations immédiates « litéral »
- Les opérations orientées octets,
- Les opérations orientées bits,
- Les opérations de contrôle,

7.2.1. Les opérations immédiates « literal »

L'adressage immédiat ne s'applique qu'au registre WREG (qu'on note W dans les instructions). L'instruction est codée sur 16 bits (8 bits pour le code-op et 8 bits pour l'opérante) ; l'opérande dans ce mode d'adresse est une constante qu'on note « **k** ». Traitons ici quelques instructions :

L'instruction MOVLW

Syntaxe : movlw k ; (WREG) ⇦ k, avec k compris en 0 et 255
; charger dans WREG la constante k

Code machine : 00001110 kkkkkkkk

L'instruction ADDLW

Syntaxe : addlw k ; (WREG) ⇦ (WREG) + k
; Ajouter à WREG la valeur k

Code machine : 00001111 kkkkkkkk

Registre STATUS : cette instruction affecte tous les flags : N, OV, Z, DC et C

L'instruction SUBLW

Syntaxe : sublw k ; (WREG) ⇦ k – (WREG)
; Soustraire de k le contenu de WREG

Code machine : 00001000 kkkkkkkk

Registre STATUS : cette instruction affecte tous les flags : N, OV, Z, DC et C

L'instruction ANDLW

Syntaxe : andlw k ; (WREG) ⇦ (WREG) and k
; effectue un AND bit à bit

Code machine : 00001011 kkkkkkkk

Registre STATUS : cette instruction affecte les flags : N et Z ; le flag N n'a aucun intérêt.

L'instruction MULLW

Syntaxe : mullw k ; (PRODH,PRODL) ⇦ (WREG) x k
; multiplie (WREG) par k, le résultat est chargé dans le registre PROD.

Code machine : 00001101 kkkkkkkk

Registre STATUS : cette instruction n'affecte aucun flag.

Il y deux instructions spéciale à adressage immédiat : movlb et lfsr

L'instruction MOVLB

Syntaxe : movlb k ; (BSR) ⇦ k, avec k compris entre 0 et 15
; Charge dans BSR le numéro de la banque.

Code machine : 00000001 kkkkkkkk

Registre STATUS : cette instruction n'affecte aucun flag.

L'instruction LFSR

Syntaxe : lfsr f,k ; (FSRx) ⇦ k , avec k compris entre 0 et 4095

Cette instruction charge une adresse dans l'un de trois charges (FSR0, FSR1 ou FSR2) ; vous remarquez la présence du paramètre « **f** » qui définit le numéro du registre FSR (f = 0 ou 1 ou 2). « **k** » représente l'adresse de la case mémoire codée sur 12 bits.

Code machine : 11101110 00ff kkkk
11110000 kkkkkkkk

Registre STATUS : cette instruction n'affecte aucun flag.

7.2.2. Les opérations orientées octets

Ce sont des instructions qui manipulent des données sous formes d'octets ; les informations se trouvent dans la mémoire de données (File registers) ou bien dans le registre WREG. La plupart des opérations sont effectuées entre le registre WREG et une case mémoire. L'accès à une case dépend du mode d'adressage utilisé.

Notations :

- Au niveau du mnémonique : lorsqu'on trouve le lettre **W**, c'est que l'un des opérandes se trouve dans WREG ; si on trouve le lettre « **f** », l'opérande se trouve dans une case mémoire.
- Au niveau des opérandes : on désigne par la lettre « **f** » l'adresse de la case mémoire et par **W** le registre WREG.
- On désigne par la lettre « **a** » le mode d'accès à la RAM
  - a = 0 : Acces Bank, l'adresse est codée sur 8 bits, le contenu de BSR est ignoré
  - a = 1 : le numéro de la banque est spécifié par BSR
- le bit de destination « **d** » : le résultat d'une opération est placé soit dans la RAM « **f** » si **d** = 1, soit dans « **W** » si **d** = 0.

L'instruction ADDWF

Syntaxe : **addwf f,d,a** ; (dest) ⇦ (WREG) + (f) , *dest* peut être une case mémoire ou WREG.
; effectue une addition sur 8 bits entre le registre désigné par « f » et
; WREG, le résultat est placé soit dans WREG soit dans f.

Code machine : 001001da ffffffff

Registre STATUS : cette instruction affecte tous les flags : N, OV, Z, DC et C

Exemple : Additionner le contenu de WREG avec celui du PORTC, le résultat est placé dans le PORTC. L'adresse du PORTC est égale à 0xF82 (voir tableau Figure II-10).

```
movlb  0x0F        ; Sélectionner la banque des SFR, c'est le quartet haut de l'adresse du PORTC
addwf  0x82,1,1    ; 0x82 : partie restante de l'adresse du PORTC, 1 : PORTC la destination
                   ; 1 : adresse de la banque dans BSR. On peut utiliser une écriture plus lisible
                   ; « addwf PORTC, f », le dernier paramètre est par défaut 1.
```

On peut également utiliser mode « Acces Bank », dans ce cas le contenu de BSR est ignoré :

```
addwf PORTC, f ,a  ; « f » : destination File Register et « a » : Acces Bank
```

L'instruction ADDWFC

Syntaxe : **addwfc f,d,a** ; (dest) ⇦ (WREG) + (f) +C, *dest* peut être une case mémoire ou WREG
; additionner le contenu du registre désigné par « f » et WREG et le flag
; C , le résultat est placé soit dans WREG soit dans f.

Code machine : 001000da ffffffff

Registre STATUS : cette instruction affecte tous les flags : N, OV, Z, DC et C

L'instruction CPFSEQ

Syntaxe : **cpfseq f, a** ; compare le contenu de WREG avec le registre désigné par « f » et
; saute l'instruction suivante (skip) s'ils sont égaux.

Code machine : 0110001a ffffffff

Registre STATUS : cette instruction n'affecte aucun flag.

L'instruction CPFSGT

Syntaxe : **cpfsgt f, a** ; compare le contenu du registre désigné par « f » et WREG
; saute l'instruction suivante (skip) si (f) > WREG.

Code machine : 0110010a ffffffff

Registre STATUS : cette instruction n'affecte aucun flag.

L'instruction CPFSLT

Syntaxe : **cpfslt f, a** ; compare le contenu du registre désigné par « **f** » et WREG
; saute l'instruction suivante (skip) si (**f**) < WREG.

Code machine : 0110000a ffffffff

Registre STATUS : cette instruction n'affecte aucun flag.

L'instruction SUBWF

Syntaxe : **subwf f,d,a** ; (dest) ⇦ (f) – (WREG), *dest* peut être une case mémoire ou WREG
; soustrait le contenu WREG du registre désigné par « **f** », le résultat est
; placé soit dans WREG soit dans **f**.

Code machine : 010111da ffffffff

Registre STATUS : cette instruction affecte tous les flags : N, OV, Z, DC et C

L'instruction SUBWFB

Syntaxe : **subwfb f,d,a** ; (dest) ⇦ (f) – (WREG) – C, dest peut être une case mémoire ou WREG
; soustrait le contenu WREG et l'emprunt du registre désigné par « f »,
; le résultat est placé soit dans WREG soit dans **f**.

Code machine : 010110da ffffffff

Registre STATUS : cette instruction affecte tous les flags : N, OV, Z, DC et C

L'instruction MOVWF

Syntaxe **: movwf f, a** ; (f) ⇦(WREG), copier le contenu de WREG dans le registre d'adresse « **f** »,

Code machine : 0110111a ffffffff

Registre STATUS : cette instruction n'affecte aucun flag.

L'instruction MOVF

Syntaxe : **movf f,d,a** ; (dest) ⇦ (f), copier le registre d'adresse **f** dans WREG ou « f ».
; copier **f** dans **f** parait n'a aucun sens, mais cette instruction a la
; particularité de modifier la flag **Z**. on peut dans ce cas tester si le
; contenu d'une case mémoire est nul sans utiliser d'autres ressources.

Code machine : 010100da ffffffff

Registre STATUS : cette instruction affecte les flags : Z et N

L'instruction MOVFF

Syntaxe : **movff fs,fd** ; (fd) ⇦ (fs), copier le contenu d'un registre dans un autre registre, sans
; restriction de banque. Les adresses fs et fd sont codées sur 12 bits.

Code machine : 1100 ffffffffffff_s

1111 ffffffffffff_d

Registre STATUS : cette instruction n'affecte aucun flag.

7.2.3. Opérations orientées bits

Ces instructions permettent de modifier ou tester un bit dans un registre. Pour ces instructions on doit fournir comme arguments, l'adresse du registre « **f** » et le numéro du bit « **b** ».

Ces instructions d'affectent pas les flags.

L'instruction BCF

Syntaxe : **bcf f,b,a** ; forcer le bit « **b** » du registre « **f** » à 0. « **b** » compris entre 0 et 7.

Code machine : 1001bbba ffffffff

L'instruction BSF

Syntaxe **: bsf f,b,a** ; forcer le bit « **b** » du registre « **f** » à 1. « **b** » compris entre 0 et 7.

Code machine : 1000bbba ffffffff

L'instruction BTG

Syntaxe : **btg f,b,a** ; inverse l'état du bit « **b** » du registre « **f** ». « **b** » compris entre 0 et 7.

Code machine : 0111bbba ffffffff

L'instruction BTFSC

Syntaxe : **btfsc f,b,a** ; saute l'instruction suivante si le bit « **b** » du registre « **f** » est à 0.

Code machine : 1011bbba ffffffff

L'instruction BTFSS

Syntaxe : **btfss f,b,a** ; saute l'instruction suivante si le bit « **b** » du registre « **f** » est à 1.

Code machine : 1010bbba ffffffff

7.2.4. Les opérations de contrôle

On retrouve ici les instructions à adressage inhérent, les instructions de branchement et les instructions de manipulation des données placées dans la mémoire de programme.

Les instructions de branchement conditionnel

BC/ BNC : sauter si C = 1/ C = 0

BN/ BNN : sauter si N = 1/ N = 0

BZ/ BNZ : sauter si Z = 1/ Z = 0

BOV/ BNOV : sauter si OV = 1/ OV = 0

Syntaxe : **BC n** ; saut relatif court si C = 1, **n** est compris entre –128 et +127. Le compteur
; de programme est augmenté de 2n + 2. (PC) ⇦ (PC) +2 +2n

Code machine : 11100010 nnnnnnnn

Les instructions de branchement inconditionnel

BRA : saut relatif

GOTO : saut absolu

Syntaxe : **bra n** ; saut relatif, **n** est compris entre –1024 et +1023. Le compteur
; de programme est augmenté de 2n + 2. (PC) ⇦ (PC) +2 +2n

Code machine : 11010nnn nnnnnnnn

Syntaxe : **goto k** ; saut absolu, PC<20 :1> ⇦ k. **k** est codé sur 20 bits alors que PC compte
; 21 bits, ceci est logique car le bit LSB est toujours nul. Pour trouver
; l'adresse réelle il faut décaler k à gauche de un bit.

Code machine : 1110 1111kkkkkkkk
1111 kkkkkkkkkkkk

Les instructions de saut vers un sous-programme,

RCALL/CALL : branchement relatif/absolu

Syntaxe : **rcall n** ; saut relatif, **n** est compris entre –1024 et +1023. Le compteur
; de programme est augmenté de 2n + 2. (PC) ⇦ (PC) +2 +2n ; avant le
; branchement (PC) +2 est sauvegardé dans la pile.

Code machine : 11011nnn nnnnnnnn

Syntaxe : **call k,{s}** ; saut absolu, PC<20 :1> ⇦ k. avant de sauter (PC) +2 est sauvegardé
; dans la pile. Si le paramètre optionnel **s** = 1, les registre WREG,
; STATUS et BSR sont sauvegardés dans WS, STATUSS et BSRS

Code machine : 1110 110skkkkkkkk
1111 kkkkkkkkkkkk

RETURN/ RETLW/RETFIE

Syntaxe : **return {s}** ; fin d'un sous-programme, PC est rechargé depuis la pile. Si le paramètre
; optionnel **s** = 1, les registres WREG, STATUS et BSR sont rechargés
; depuis les registres WS, STATUSS et BSRS

Code machine : 000000000001001s

Syntaxe : **retlw k** ; fin d'un sous-programme, PC est rechargé depuis la pile et WREG est
; chargé par la valeur k.

Code machine : 00001100kkkkkkkk

Syntaxe : **retfie {s}** ; fin d'un sous-programme d'interruption, PC est rechargé depuis la pile.
; Si le paramètre optionnel **s** = 1, les registres WREG, STATUS et BSR
; sont rechargés depuis WS, STATUSS et BSRS. Le bit GIE/GIEH
; ou PEIE/GIEL est positionné.

Code machine : 000000000001000s

7.2.5. Table des instructions

Les instructions orientées octets

ADDWF	f, d, a	Add WREG and f	1	0010	01da	ffff	ffff	C, DC, Z, OV, N
ADDWFC	f, d, a	Add WREG and Carry bit to f	1	0010	00da	ffff	ffff	C, DC, Z, OV, N
ANDWF	f, d, a	AND WREG with f	1	0001	01da	ffff	ffff	Z, N
CLRF	f, a	Clear f	1	0110	101a	ffff	ffff	Z
COMF	f, d, a	Complement f	1	0001	11da	ffff	ffff	Z, N
CPFSEQ	f, a	Compare f with WREG, Skip =	1 (2 or 3)	0110	001a	ffff	ffff	None
CPFSGT	f, a	Compare f with WREG, Skip >	1 (2 or 3)	0110	010a	ffff	ffff	None
CPFSLT	f, a	Compare f with WREG, Skip <	1 (2 or 3)	0110	000a	ffff	ffff	None
DECF	f, d, a	Decrement f	1	0000	01da	ffff	ffff	C, DC, Z, OV, N
DECFSZ	f, d, a	Decrement f, Skip if 0	1 (2 or 3)	0010	11da	ffff	ffff	None
DCFSNZ	f, d, a	Decrement f, Skip if Not 0	1 (2 or 3)	0100	11da	ffff	ffff	None
INCF	f, d, a	Increment f	1	0010	10da	ffff	ffff	C, DC, Z, OV, N
INCFSZ	f, d, a	Increment f, Skip if 0	1 (2 or 3)	0011	11da	ffff	ffff	None
INFSNZ	f, d, a	Increment f, Skip if Not 0	1 (2 or 3)	0100	10da	ffff	ffff	None
IORWF	f, d, a	Inclusive OR WREG with f	1	0001	00da	ffff	ffff	Z, N
MOVF	f, d, a	Move f	1	0101	00da	ffff	ffff	Z, N
MOVFF	fs, fd	Move fs (source) to 1st word	2	1100	ffff	ffff	ffff	None
		fd (destination) 2nd word		1111	ffff	ffff	ffff	
MOVWF	f, a	Move WREG to f	1	0110	111a	ffff	ffff	None
MULWF	f, a	Multiply WREG with f	1	0000	001a	ffff	ffff	None
NEGF	f, a	Negate f	1	0110	110a	ffff	ffff	C, DC, Z, OV, N
RLCF	f, d, a	Rotate Left f through Carry	1	0011	01da	ffff	ffff	C, Z, N
RLNCF	f, d, a	Rotate Left f (No Carry)	1	0100	01da	ffff	ffff	Z, N
RRCF	f, d, a	Rotate Right f through Carry	1	0011	00da	ffff	ffff	C, Z, N
RRNCF	f, d, a	Rotate Right f (No Carry)	1	0100	00da	ffff	ffff	Z, N
SETF	f, a	Set f	1	0110	100a	ffff	ffff	None
SUBFWB	f, d, a	Subtract f from WREG with Borrow	1	0101	01da	ffff	ffff	C, DC, Z, OV, N
SUBWF	f, d, a	Subtract WREG from f	1	0101	11da	ffff	ffff	C, DC, Z, OV, N
SUBWFB	f, d, a	Subtract WREG from f with Borrow	1	0101	10da	ffff	ffff	C, DC, Z, OV, N
SWAPF	f, d, a	Swap Nibbles in f	1	0011	10da	ffff	ffff	None
TSTFSZ	f, a	Test f, Skip if 0	1 (2 or 3)	0110	011a	ffff	ffff	None
XORWF	f, d, a	Exclusive OR WREG with f	1	0001	10da	ffff	ffff	Z, N

Les instructions orientées bit

BCF	f, b, a	Bit Clear f	1	1001	bbba	ffff	ffff	None
BSF	f, b, a	Bit Set f	1	1000	bbba	ffff	ffff	None
BTFSC	f, b, a	Bit Test f, Skip if Clear	1 (2 or 3)	1011	bbba	ffff	ffff	None
BTFSS	f, b, a	Bit Test f, Skip if Set	1 (2 or 3)	1010	bbba	ffff	ffff	None
BTG	f, d, a	Bit Toggle f	1	0111	bbba	ffff	ffff	None

Les instructions de contrôle

BC	n	Branch if Carry	1 (2)	1110	0010	nnnn	nnnn	None
BN	n	Branch if Negative	1 (2)	1110	110	nnnn	nnnn	None
BNC	n	Branch if Not Carry	1 (2)	1110	0011	nnnn	nnnn	None
BNN	n	Branch if Not Negative	1 (2)	1110	111	nnnn	nnnn	None
BNOV	n	Branch if Not Overflow	1 (2)	1110	101	nnnn	nnnn	None
BNZ	n	Branch if Not Zero	1 (2)	1110	0001	nnnn	nnnn	None
BOV	n	Branch if Overflow	1 (2)	1110	100	nnnn	nnnn	None
BRA	n	Branch Unconditionally	2	1101	0nnn	nnnn	nnnn	None
BZ	n	Branch if Zero	1 (2)	1110	0000	nnnn	nnnn	None
CALL	n, s	Call Subroutine 1st word	2	1110	110s	kkkk	kkkk	None
		2nd word		1111	kkkk	kkkk	kkkk	
CLRWDT	—	Clear Watchdog Timer	1	0000	0000	0000	100	TO, PD
DAW	—	Decimal Adjust WREG	1	0000	0000	0000	111	C
GOTO	n	Go to Address 1st word	2	1110	1111	kkkk	kkkk	None
		2nd word		1111	kkkk	kkkk	kkkk	
NOP	—	No Operation	1	0000	0000	0000	0000	None
NOP	—	No Operation	1	1111	xxxx	xxxx	xxxx	None
POP	—	Pop Top of Return Stack (TOS)	1	0000	0000	0000	110	None
PUSH	—	Push Top of Return Stack (TOS)	1	0000	0000	0000	101	None
RCALL	n	Relative Call	2	1101	1nnn	nnnn	nnnn	None
RESET		Software Device Reset	1	0000	0000	1111	1111	All
RETFIE	s	Return from Interrupt Enable	2	0000	0000	0001	000s	GIE/GIEH, PEIE/GIEL
RETLW	k	Return with Literal in WREG	2	0000	1100	kkkk	kkkk	None
RETURN	s	Return from Subroutine	2	0000	0000	0001	001s	None
SLEEP	—	Go into Standby mode	1	0000	0000	0000	0011	TO, PD

Les instructions immédiates « literal »

ADDLW	k	Add Literal and WREG	1	0000	1111	kkkk	kkkk	C, DC, Z, OV, N
ANDLW	k	AND Literal with WREG	1	0000	1011	kkkk	kkkk	Z, N
IORLW	k	Inclusive OR Literal with WREG	1	0000	1001	kkkk	kkkk	Z, N
LFSR	f, k	Move Literal (12-bit) 2nd word	2	1110	1110	00ff	kkkk	None
		to FSR(f) 1st word		1111	0000	kkkk	kkkk	
MOVLB	k	Move Literal to BSR<3:0>	1	0000	0001	0000	kkkk	None
MOVLW	k	Move Literal to WREG	1	0000	1110	kkkk	kkkk	None
MULLW	k	Multiply Literal with WREG	1	0000	1101	kkkk	kkkk	None
RETLW	k	Return with Literal in WREG	2	0000	1100	kkkk	kkkk	None
SUBLW	k	Subtract WREG from Literal	1	0000	1000	kkkk	kkkk	C, DC, Z, OV, N
XORLW	k	Exclusive OR Literal with WREG	1	0000	1010	kkkk	kkkk	Z, N

Les instructions des données de la mémoire de programme

TBLRD*	Table Read	2	0000	0000	0000	1000	None
TBLRD*+	Table Read with Post-Increment		0000	0000	0000	1001	None
TBLRD*-	Table Read with Post-Decrement		0000	0000	0000	1010	None
TBLRD+*	Table Read with Pre-Increment		0000	0000	0000	1011	None
TBLWT*	Table Write	2	0000	0000	0000	1100	None
TBLWT*+	Table Write with Post-Increment		0000	0000	0000	1101	None
TBLWT*-	Table Write with Post-Decrement		0000	0000	0000	1110	None
TBLWT+*	Table Write with Pre-Increment		0000	0000	0000	1111	None

III-LES PORTS D'ENTREES/SORTIES

Le microcontrôleur PIC18F4520 dispose de 36 lignes d'entrées/sorties réparties sur 5 ports parallèles bidirectionnels :

8 lignes sur le PORTA : RA0,RA1,RA2,RA3,RA4,RA5,RA6,RA7

8 lignes sur le PORTB : RB0,RB1,RB2,RB3,RB4,RB5,RB6,RB7

8 lignes sur le PORTC : RC0,RC1,RC2,RC3,RC4,RC5,RC6,RC7

8 lignes sur le PORTD : RD0,RD1,RD2,RD3,RD4,RD5,RD6,RD7

4 lignes sur le PORTE : RE0,RE1,RE2,RE3

Chaque port est contrôlé par trois registres :

- un registre appelé **PORTx**, c'est le registre de donnée utilisé pour lire ou modifier l'état des lignes correspondantes (x ∈{A, B, C, D, E}).

- un registre de direction des données **TRISx** (TRansfert Input Set) ; il permet de définir individuellement le sens de chaque ligne du port en entrée ou en sortie. La mise à 1 ou à 0 d'un bit du registre TRISx, configure la broche correspondante du PORTx en entrée ou en sortie.

- un registre **LATx**, ce registre est utilisé lors de l'opération «read-modify-write» afin de ne modifier que l'état d'une ligne du PORTx. Il est vivement conseillé lors d'une opération de modification des bits d'utiliser ce registre au lieu du PORTx correspondant.

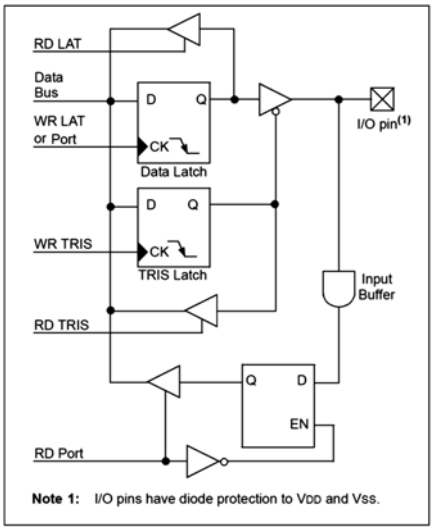


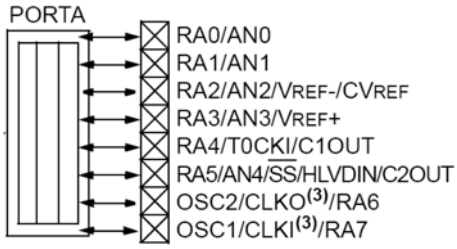
Figure III-1

La plupart des broches des PORTs sont partagées avec d'autres périphériques. En général si une broche est utilisée par un périphérique, celle-ci ne peut pas être utilisée comme broche d'entrée/sortie. Après un RESET, tous les ports sont configurés en entées.

1.1. PORTA

Le PORTA est un port de 8 bits, mais du fait du partage de certaines de ces lignes avec d'autres ressources indispensables, le nombre de lignes réellement disponible peut se trouver réellement réduit. Les lignes RA6 et RA7 du port PORTA sont multiplexées avec les lignes de l'horloge principale OSCI et OSCO, donc elles sont disponibles que dans le cas où le microcontrôleur fonctionnement avec une horloge interne.

La broche RA4 est multiplexé avec l'entrée d'horloge externe du timer0 (T0CKI) ou la sortie du comparateur analogique. Les autres broches sont multiplexées avec les entrées du convertisseur analogique-numérique ou le comparateur analogique (AN0 .. AN5).



La broche RA5 peut être utilisée par le module SPI ou par le module de la détection de la variation de tension HLVD.

La broche RA2 peut être utilisée comme sortie de référence (CVref), le bit CVROE du registre CVRCON permet de configurer la ligne RA2 pour la génération d'une tension de référence.

Après un reset les broches du PORTA sont configurée en entées analogiques. Si vous voulez utiliser ses broches en entrées/sorties TOR, il faut charger les registres ADCON1 et CMCON par la valeur Binaire '00000111'.

L'exemple suivant configure les trois broches du PORTA RA0, RA1 et RA2 en sortie et les autres en entrée :

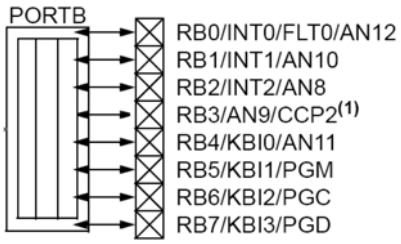
```
movlw 07h               ; Configure les broches en entrées-sorties
movwf ADCON1            ; numériques
movlw 07h               ; Le comparateur analogique n'utilise pas
movwf CMCON             ; les broches du PORTA
movlw F8h               ; RA<0:2> en sorties et RA<3:7> en entrées
movwf TRISA             ;
```

En langage C

```
ADCON1      = 0x07 ;
CMCON       = 0x07 ;
TRISA       = 0xF8 ;
```

1.2. PORTB

Toutes les broches du PORTB possèdent des résistances de tirage (pull-ups). Ces résistances sont mises en œuvre par la mise à 0 du bit RBPU\ du registre INTCON2 ; elles sont automatiquement désactivées quand le port est configuré en sortie.



Les broches RB7/PGD, RB6/PGC et RB5/PGM sont utilisé pour la programmation du circuit.

Les broches RB4:RB7 peuvent affecter le flag RBIF en cas de changement d'état de l'une de 4 broches, et éventuellement générer une interruption si, celle-ci est autorisée. Il faut bien noter, que seules les broches qui sont configurées en entrées peuvent déclencher l'interruption.

La broche RB3 peut être utilisée par le module CCP2 ou l'entrée analogique AN9.

La broche RB0, RB1 et RB2 peuvent être utilisé comme entrées d'interruptions externes ou entrées analogiques. Ces interruptions peuvent être générées, sur front montant ou descendant selon l'état des bits INTEDGx du registre INTCON2. La broche RB0/FLT0 peut être utilisée comme entrée de détection de défaut du module CCP1 lorsqu'il est configuré en mode PWM.

1.3. PORTC

Le PORTC est multiplexé avec plusieurs périphériques ; on se limite ici à donner les fonctions des différentes broches.

RC0/T1OSO/T13CKI : Input/output port pin or Timer1 oscillator output/Timer1 or 3 clock input

RC1/T1OSI/CCP2 : Input/output port pin or Timer1 oscillator input or Capture2 input/Compare2 output/PWM2 output

RC2/CCP1/P1A : Input/output port pin or Capture1 input/Compare1 output/PWM1output
RC3/SCK/SCL : RC3 can also be the synchronous serial clock for both SPI and I2C modes.
RC4/SDI/SDA : RC4 can also be the SPI Data In (SPI mode) or data I/O (I2C mode).
RC5/SDO : Input/output port pin or Synchronous Serial Port data output (SPI mode)
RC6/TX/CK : Input/output port pin or Asynchronous Transmit or Synchronous Clock
RC7/RX/DT : Input/output port pin or Asynchronous Receive or Synchronous Data

1.4. PORTD et PORTE

En plus de leur utilisation comme PORTS E/S; les ports D et E, permettent au microcontrôleur de travailler en mode PSP (Parallel Slave Port) c'est-à-dire, qu'il peut être interfacé avec un autre microprocesseur. Dans ce cas le PORTD représente le bus de données et le PORTE les signaux de contrôle (RD\, WR\ et CS\). Le registre TRISE dispose de certains bits supplémentaires pour la gestion du mode PSP. Les lignes RD5, RD6 et RD7 sont partagées avec le module ECCP1.

Le PORTE peut être aussi, configuré en mode analogique pour former avec le PORTA les 8 entrées du convertisseur analogique numérique.

Par défaut, le PORTE est configuré comme port analogique, et donc, comme pour le PORTA, vous devrez placer la valeur "00001111" dans ADCON1 pour pouvoir l'utiliser comme port E/S numérique. Cette remarque concerne aussi le PORTB.

IV- CONVERTISSEUR ANALOGIQUE NUMERIQUE

1. Présentation

Le convertisseur analogique numérique des microcontrôleurs PIC fonctionne selon le principe des approximations successives avec une résolution de 10 bits. Le microcontrôleur PIC184520 dispose de 13 canaux d'entrées analogiques reparties sur les ports A, B et E. Les tensions de référence V_{ref+} et V_{ref-} du module ADC peuvent être choisies de manière logicielle par combinaison des lignes VDD, VSS, RA2 ou RA3.

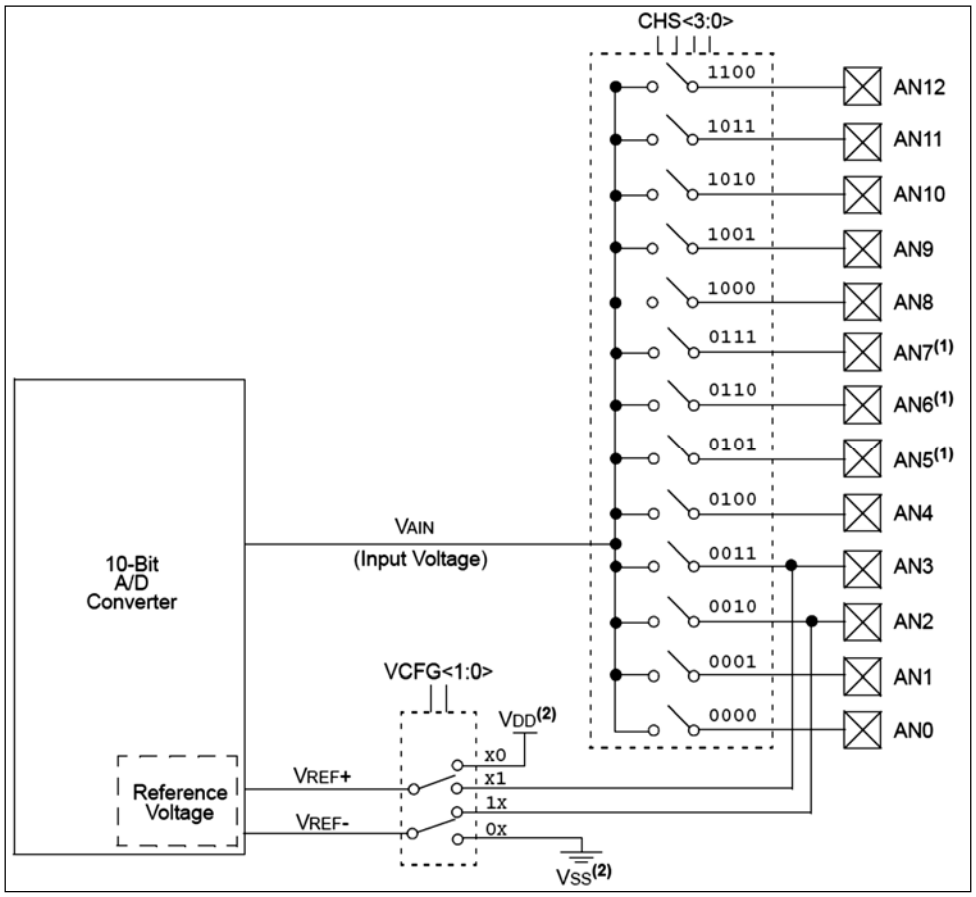


Figure IV-1 : schéma bloc de l'ADC

Le module ADC dispose de cinq registres :

- ADRESL : A/D RESult Low register
- ADRESH : A/D RESult High register
- ADCON0 : A/D CONtrol register 0
- ADCON1 : A/D CONtrol register 1
- ADCON1 : A/D CONtrol register 2

1.1. Registre ADCON0

On contrôle à travers ce registre la sélection du canal (entrée à convertir), l'ordre de début de conversion et la mise en service du module ADC.

U-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
-	-	CHS3	CHS2	CHS1	CHS0	$G0/\overline{DONE}$	ADON
b7	b6	b5	b4	b3	b2	b1	b0

Bit 7 ,6 : **Non implémentés**.

Bit 5..2 : **CHS2..CHS0 : Analog Channel Select bits.**

CHS3	CHS2	CHS1	CHS0	Canal	Broche
0	0	0	0	0	RA0/AN0
0	0	0	1	1	RA1/AN1
0	0	1	0	2	RA2/AN2
0	0	1	1	3	RA3/AN3
0	1	0	0	4	RA5/AN4
0	1	0	1	5	RE0/AN5
0	1	1	0	6	RE1/AN6
0	1	1	1	7	RE2/AN7
1	0	0	0	8	RB2/AN8
1	0	0	1	9	RB3/AN9
1	0	1	0	10	RB1/AN10
1	0	1	1	11	RB4/AN11
1	1	0	0	12	RB0/AN12
1	1	0	1	Non implémentés	
1	1	1	0		
1	1	1	1		

bit 1 : GO/$\overline{DONE}$: A/D Conversion Status bit.

GO/$\overline{DONE}$ = 1 : la conversion progresse (la mise à 1 de ce bit démarre la conversion)

GO/$\overline{DONE}$ = 0 : Conversion terminée (ce bit est remis automatiquement à 0 la fin de conversion).

La fin de conversion est signalée par le retour à zéro du bit GO/$\overline{DONE}$ et la mise à 1 du bit ADIF du registre PIR1.

Bit 0 : ADON : A/D On bit

ADON = 1 : Convertisseur A/D en service

ADON = 0 : Convertisseur A/D hors service

1.2. Registre ADCON1

Ce registre permet essentiellement de définir la fonction des broches qui sont multiplexées avec le convertisseur analogique/numérique (RA2 et RA3 peuvent être utilisées pour les tensions de références).

U - 0	U - 0	R/W - 0	R/W - 0	R/W - 0	R/W-q(1)	R/W-q(1)	R/W-q(1)
-	-	VCFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0
b7	b6	b5	b4	b3	b2	b1	b0

Bits 7,6 : **Non implémentés**

Bit 5 : **VCFG1: Voltage Reference Configuration bit (VREF- source)**

VCFG1 = 1 : Vref- sur AN2

VCFG1 = 0 : Vref- = VSS

Bit 4 : **VCFG0: Voltage Reference Configuration bit (VREF+ source)**

VCFG0 = 1 : Vref+ sur AN3

VCFG0 = 0 : Vref+ = VDD

bits 3..0 : **PCFG3..PCFG0** : **A/D Port ConFiGuration control bit**s. : le tableau suivant récapitule la configuration des broches selon les combinaisons des bits PCFG3..PCFG0.

PCFG 3 .. 0	AN12 RB0	AN11 RB4	AN10 RB1	AN9 RB3	AN8 RB2	AN7 RE2	AN6 RE1	AN5 RE0	AN4 RA5	AN3 RA3	AN2 RA2	AN1 RA1	AN0 RA0
0000	A	A	A	A	A	A	A	A	A	A	A	A	A
0001	A	A	A	A	A	A	A	A	A	A	A	A	A
0010	A	A	A	A	A	A	A	A	A	A	A	A	A
0011	D	A	A	A	A	A	A	A	A	A	A	A	A
0100	D	D	A	A	A	A	A	A	A	A	A	A	A
0101	D	D	D	A	A	A	A	A	A	A	A	A	A
0110	D	D	D	D	A	A	A	A	A	A	A	A	A
0111	D	D	D	D	D	A	A	A	A	A	A	A	A
1000	D	D	D	D	D	D	A	A	A	A	A	A	A
1001	D	D	D	D	D	D	D	A	A	A	A	A	A
1010	D	D	D	D	D	D	D	D	A	A	A	A	A
1011	D	D	D	D	D	D	D	D	D	A	A	A	A
1100	D	D	D	D	D	D	D	D	D	D	A	A	A
1101	D	D	D	D	D	D	D	D	D	D	D	A	A
1110	D	D	D	D	D	D	D	D	D	D	D	D	A
1111	D	D	D	D	D	D	D	D	D	D	D	D	D

1.3. Registre ADCON2

La fréquence d'horloge du convertisseur A/D est dérivée de la fréquence principale du microcontrôleur Fosc. Pour des raisons électroniques, la période d'horloge de l'ADC, $\mathbf{T_{AD}}$ ne doit être supérieure à 0,7µs. Donc en fonction de la fréquence du quartz, il faudra choisir le diviseur le plus approprié pour avoir un $T_{AD} \geq 0{,}7\mu s$; pour cela Microchip a divisé la plage de fréquence en 6 intervalles afin de garantir un temps Tad ≥ 0,7µs. Dans le cas d'une horloge à base du circuit RC, T_{AD} devrait être entre 1,2 et 2,5µs. Le facteur de division est fixé par le trois bits ADCS2..ADCS0 du registre ADCON2. De plus on peut programmer le temps d'acquisition qui sera pris automatiquement lorsqu'on lance le début de conversion

R/W-0	U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W -0	R/W-0
ADFM	-	ACQT2	ACQT1	ACQT0	ADCS2	ADCS1	ADCS0
b7	b6	b5	b4	b3	b2	b1	b0

Bit 7 : **ADFM** : **A/D Result format select.** Définie la façon de charger le résultat de conversion dans les registres ADRESL et ADRESH (Figure IV-2).

ADFM = 1 : Justifié à droite. ADRESH ne contient que les 2 bits MSB du résultat. Les 6 MSB bits de ce registre sont lus comme des "0".

ADFM = 0 : Justifié à gauche. ADRESL ne contient que les 2 bits LSB du résultat. Les 6 LSB bits de ce registre sont lus comme des "0".

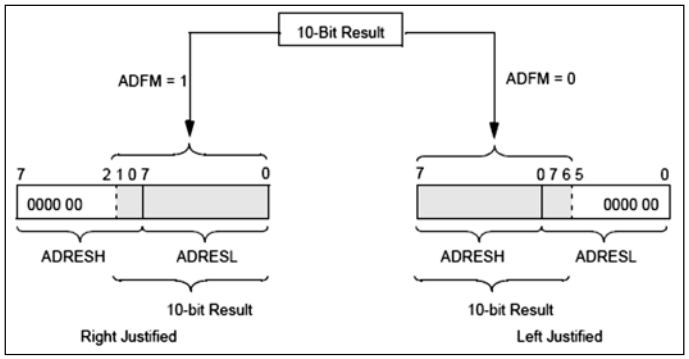


Figure IV-2 : format de donnée

Bit 6 : Non implémenté

Bits 5..3 : **ACQT<2:0>:** A/D Acquisition Time Select bits

ACQT2	ACQT1	ACQT0	Temps d'acquisition (T_{ACQ})
1	1	1	20 T_{AD}
1	1	0	16 T_{AD}
1	0	1	12 T_{AD}
1	0	0	8 T_{AD}
0	1	1	6 T_{AD}
0	1	0	4 T_{AD}
0	0	1	2 T_{AD}
0	0	0	0 T_{AD}

Bits 2..0 : **ADCS<2:0>:** A/D Conversion Clock Select bits

ADCS2	ADCS1	ADCS0	Période d'horloge T_{AD}	Fréquence Maximale
0	0	0	2 T_{OSC}	2,86 Mhz
1	0	0	4 T_{OSC}	5,71 Mhz
0	0	1	8 T_{OSC}	11,43 Mhz
1	0	1	16 T_{OSC}	22,86 Mhz
0	1	0	32 T_{OSC}	40 Mhz
1	1	0	64 T_{OSC}	40 Mhz
x	1	1	RC	1 Mhz

2. Conversion A/N

La conversion d'un signal analogique en équivalent numérique passe par deux phases :

- l'échantillonnage blocage (sample and hold). Cette opération consiste à connecter l'entrée à convertir à un condensateur interne, qui va se charger à travers une résistance interne jusqu'à la tension appliquée. Ce temps est appelé temps d'acquisition **T_{ACQ}**.

- Une fois le condensateur est chargé, l'ordre de début de conversion (**GO/$\overline{DONE}$** = 1) permet de déconnecter la tension appliquée, pour procéder à la phase ce conversion.

2.1. Temps d'acquisition

C'est le temps nécessaire pour que le condensateur interne atteigne une tension proche de la tension à convertir. Cette charge s'effectue à travers une résistance interne et la résistance de la source connectée à l'entrée de l'ADC. Ce temps est augmenté du temps de réaction des circuits et d'un temps qui dépend de la variation de la température.

Tacq = Tamp + Tc + Tcoff

Tamp : temps de réaction des circuits

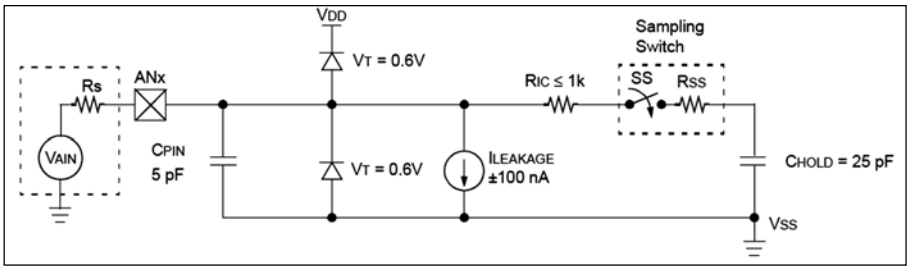
Tc : temps de charge du condensateur

Tcoff : temps qui dépend du coefficient de température.

Le temps de réaction est fixé à 0,2µs ; de même Tcoff est limité à une valeur maximale de 1,2µs.

La résistance interne totale (composée de RIC et RSS) varie de 1Kohms sous 6V pour arriver à 4Kohms sous 3V, en passant à 2Kohms sous 5V. De plus Microchip recommande que la résistance de la source soit inférieure à 2,5 Kohms.

Tc = CHOLD (RIC + RSS + RS) ln(1/2047)



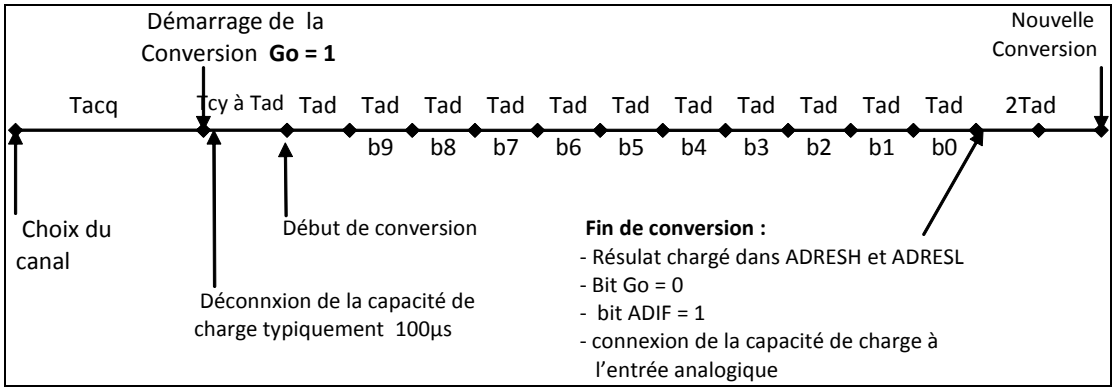
Dans les conditions normale et pour une tension de 5V, Tc = 1,05µs. Le temps d'acquisition $\mathbf{T_{ACQ}}$ = 2,5µs.

2.2. Temps de conversion

C'est le temps nécessaire pour convertir le signal analogique en équivalent numérique. Il dépend de l'horloge interne $\mathbf{T_{AD}}$ (Typiquement 0,7µs) .

Le microcontrôleur PIC nécessite un temps T_{AD} avant le démarrage effectif de la conversion, et un temps supplémentaire T_{AD} à la fin de la conversion. Soit au total 12 Tad. Donc dans les meilleurs conditions **Tconv** = 12*0,7µs = 8,4µs.

Il faut bien noter qu'un temps équivalent à 2 * T_{AD} est nécessaire avant de pouvoir effectuer une nouvelle conversion.



Pour les microcontrôleurs PIC18, le temps d'acquisition peut être programmer automatiquement à travers les bits ACQT2..0 du registre ADCON2. Dans ce cas, après la mise à 1 du bit **GO/$\overline{DONE}$**, le microcontrôleur attend un temps T_{ACQ} avant de démarrer la conversion.

V- LES TIMERS

1. Introduction

Les Timers/compteurs sont des périphériques de gestion de temps. Ils permettent de réaliser les fonctions suivantes :

- comptage des évènements
- synchronisation des signaux
- fixer le débit d'une liaison série synchrone et asynchrone
- génération des événements périodiques (échantillonnage des signaux analogiques, rafraichissement des afficheurs multiplexés, régulation numérique …)
- génération des signaux périodiques (carré, MLI …)
- mesure de temps…

2. Fonctionnement des Timers

Les timers sont des compteurs formés généralement d'un **pré-diviseur** suivi d'un **registre compteur** de 8 ou 16 bits. L'entrée d'horloge peut être interne (*mode timer*) ou externe (*mode compteur d'événements*). Lorsque le registre compteur atteint sa valeur maximale et repasse à 0, un bit indicateur de débordement (*flag*) sera positionné et une interruption pourra être générée, informant ainsi la CPU du débordement du timer. Il faut bien noter que le programmeur devra remettre à zéro cet indicateur après chaque débordement.

Le microcontrôleur PIC18F4520 dispose de quatre timers appelés Timer0, Timer1, Timer2 et Timer3.

3. Timer 0

Ce Timer est implémenté dans toutes les versions des microcontrôleurs de Microchip, son ancienne appellation était **RTC** (**R**eal **T**ime **C**lock). Il est formé d'un pré-diviseur programmable 8 bits (Programmable Prescaler) suivi d'un registre compteur 8 bits (**TMR0**), étendue pour les PIC18 à 16bits. Ce Timer travaille en deux modes : le mode 8 bits (Pour assurer la compatibilité avec les anciennes versions) et le mode 16 bits.

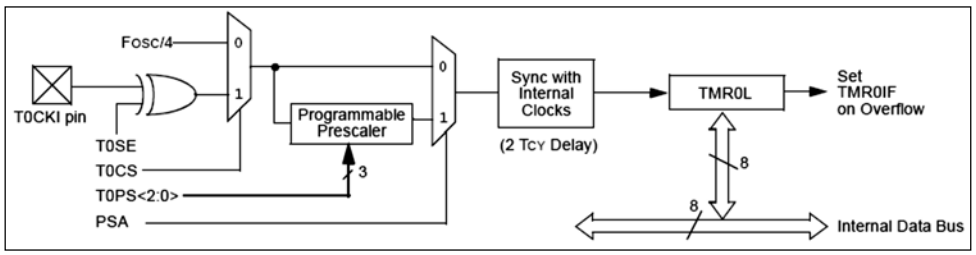


Figure V-1 : Timer 0 en mode 8 bits

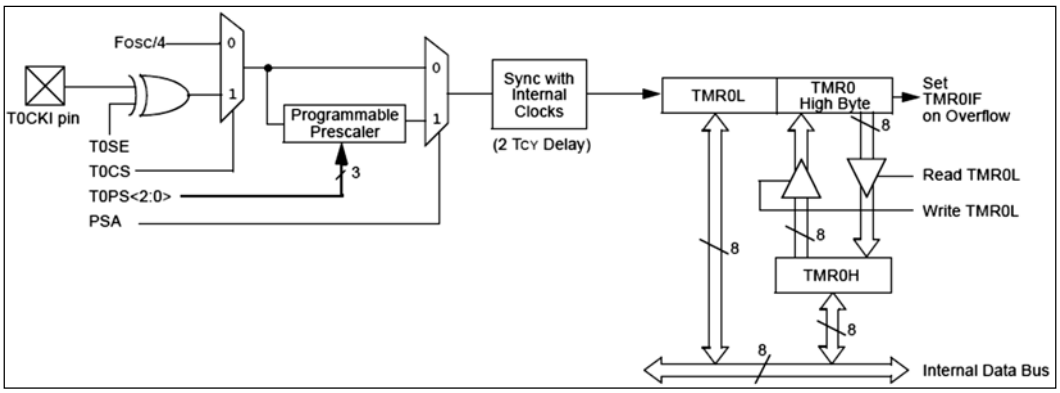


Figure V-2 : Timer 0 en mode 16 bits

Le bit **T0CS** permet de choisir l'horloge, interne (Fosc/4) ou externe RA4/T0CKI. Dans ce dernier cas l'incrémentation du timer 0 peut se faire soit sur front montant ou descendant suivant la valeur du bit **T0SE**. Le bit **PSA** permet l'insertion du prédivisieur dans la chaîne de comptage ou non (pas de prédivision). Les bits **PS2**, **PS1** et **PS0** sont utilisés pour fixer la valeur de prédivision à 2,4, 8, 16, 32, 64, 128 ou 256.

Quand le contenu du **TMR0** passe de FF à 00 (ou de FFFF à 0000 en mode 16 bits) le bit **TMR0IF** du registre **INTCON** passe à 1 pour signaler un débordement et, une interruption pourra être déclenchée, si le bit **TMR0IE** est positionné.

En mode 16 bits, la lecture du registre **TMR0L** charge automatiquement le contenu du registre **TMR0** dans le registre buffer **TMR0H**. De même, l'écriture dans **TMR0L** charge automatiquement le contenu du registre **TMR0H** dans le registre **TMR0**.

Tous les bits de configuration du Timer0 sont implémentés dans le registre **T0CON**.

Registre T0CON

R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1	R/W-1
TMR0ON	T08BIT	T0CS	T0SE	PSA	PS2	PS1	PS0
b7	b6	b5	b4	b3	b2	b1	b0

Bit 7 : **TMR0ON: Timer0 On/Off Control bit**

TMR0ON = 1 : validation du Timer 0.

TMR0ON = 0 : Timer 0 stopé

Bit 6 : **T08BIT: Timer0 8-Bit/16-Bit Control bit**

T08BIT = 1 : mode 8 bit (par défaut).

T08BIT = 0 : mode 16 bits.

Bit 5 : **T0CS** : **TMR0 Clock Source Select bit**

T0CS = 1 : fonctionnement en mode compteur (horloge externe)

T0CS = 0 : fonctionnement en mode Timer (horloge interne Fosc/4)

bit 4 : **T0SE** TMR0 Source Edge Select bit.

T0SE = 1 : incrémentation sur transition positive de l'horloge externe

T0SE = 0 : incrémentation sur transition négative de l'horloge externe

Bit 3 : **PSA** : Prescaler Assignment bit.

PSA = 1 : Prédiviseur non assigné

PSA = 0 : Prédiviseur assigné au Timer0

Bit 2..0 : **PS2, PS1, PS0** : Prescaler Rate Select bits.

PS2	PS1	PS0	Prédiv Timer0
0	0	0	2
0	0	1	4
0	1	0	8
0	1	1	16
1	0	0	32
1	0	1	64
1	1	0	128
1	1	1	256

4. Timer 1

L'architecture du Timer 1 est semblable à celle du Timer 0 :

- Il est formé d'une paire de registres de 8 bits TMR1 et TMR1L montés en cascade formant ainsi un registre compteur 16 bits.
- Le prédiviseur programmable permet la division de fréquence par 1, 2, 4 ou 8.
- Le Timer 1 peut, comme le Timer 0, fonctionner en mode Timer ou en mode compteur avec la possibilité de connecter un Quartz entre les broches RC0 et RC1.
- Fonctionnement en mode synchrone ou asynchrone. Le mode asynchrone permet au Timer 1 de compter en mode veille (Sleep).
- Possibilité de bloquer le comptage.

Tous les bits de configuration associés au Timer1 se trouvent dans le registre **T1CON.** Le bit indicateur de débordement ***TMR1IF*** et le bit de validation de l'interruption ***TMR1IE*** se trouvent respectivement dans les registres **PIR1** et **PIE1** ; le bit du choix de niveau de priorité ***TMR1IP*** se trouve dans le registre **IPR1**.

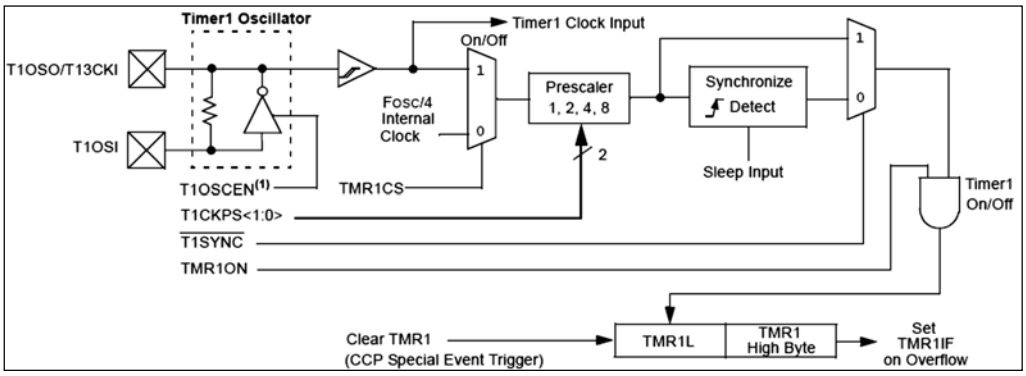


Figure V-3

Registre T1CON

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T1RUN	T1CKPS1	T1CKPS0	T1OSCEN	T1SYNC\	TMR1CS	TMR1ON
b7	b6	b5	b4	b3	b2	b1	b0

Bit 7 : **RD16 :** 16-Bit Read/Write Mode Enable bit

RD16 = 1 : valide la lecture ou l'écriture du timer en mode 16 bits

RD16 = 0 : valide la lecture ou l'écriture du timer en double octets

Bit 6 : **T1RUN: Timer1 System Clock Status bit**

T1RUN = 1 : indique que l'horloge est délivrée à partir de l'oscillateur

T1RUN = 0 : indique que l'horloge est délivrée par une autre source

Bit 5..4 : **T1CKPS1:T1CKPS0: Timer1 Input Clock Prescale Select bits**

T1CKPS1	T1CKPS0	Prédivision
0	0	1
0	1	2
1	0	4
1	1	8

Bit 3 : **T1OSCEN: Timer1 Oscillator Enable Control bit**

T1OSCEN = 1 : oscillateur autorisé

T1OSCEN = 0 : oscillateur stoppé

bit 2 : **T1SYNC\: Timer1 External Clock Input Synchronization Control bit**

T1SYNC\ = 1 : Pas de synchronisation de l'horloge externe

T1SYNC\ = 0 : Synchronisation de l'horloge externe

Bit 1 : **TMR1CS: Timer1 Clock Source Select bit**

TMR1CS = 1 : horloge externe

TMR1CS = 0 : horloge interne

Bit 0 : **TMR1ON: Timer1 On bit**

TMR1ON = 1 : comptage progresse

TMR1ON = 0 : comptage bloqué

5. Timer 2

Le Timer 2 comporte un registre compteur 8 bits (**TMR2**) avec un pré-divideur et un post-diviseur. Ce timer admet uniquement une horloge interne (Fosc/4). Le pré-diviseur peut être paramétré par l'une de trois valeurs : 1, 4 ou 16, tandis que le post-diviseur permet des divisions de 1 à 16 : 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15 ou 16.

Le principe de fonctionnement du Timer 2 est différent à ces précédents. Le débordement du compteur n'aura pas lieu lorsque celui-ci dépasse sa valeur maximale (FFH), mais lorsqu'il dépasse une valeur prédéfinie mémorisée dans le registre **PR2**. La période totale de débordement est donnée par la formule suivante : $T = T_{osc} \times 4 \times \text{Pré}div \times (PR2 + 1) \times Postdiv$.

Chaque débordement entraine le positionnement de l'indicateur ***TMR2IF*** et pourra générer une interruption si le bit ***TMR2IE*** est positionné ; le niveau de priorité est fixé par le bit ***TMR2IP***.

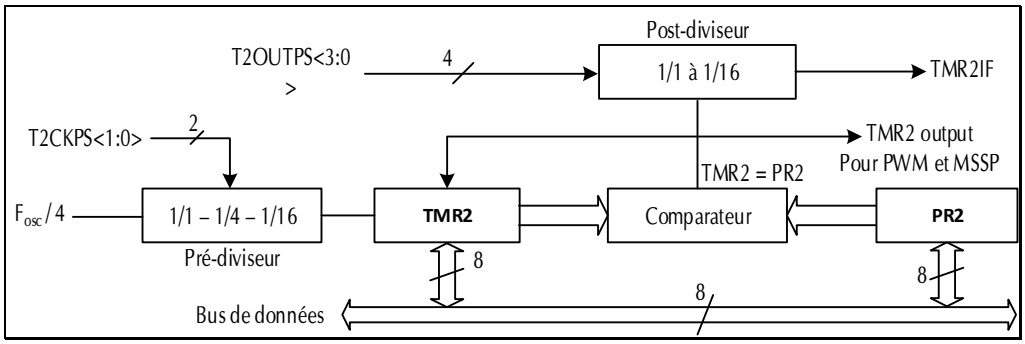


Figure V-4

Le registre T2CON permet la configuration du timer 2.

Registre T2CON

U-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
-	TOUTPS3	TOUTPS 2	TOUTPS 1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0
b7	b6	b5	b4	b3	b2	b1	b0

Bit 7 : **Non implementé .**

Bit 6..3 : **TOUTPS3..TOUTPS0: Timer2 Output Postscale Select bits**

TOUTPS3	TOUTPS 2	TOUTPS 1	TOUTPS0	Postdivision
0	0	0	0	1
0	0	0	1	2
0	0	1	0	3
		:		:
		:		:
1	1	1	1	16

Bit 2 : **TMR2ON: Timer2 On bit**

TMR2ON = 1 : Timer 2 ON (mise en service)

TMR2ON = 0 : Timer 2 OFF (mise hors service)

bit 1..0 : **T2CKPS1:T2CKPS0**: Timer2 Clock Prescale Select bits

T2CKPS1	T2CKPS0	Prédivision
0	0	1
0	1	4
1	x	16

6. Timer 3

Le Timer 3 est pratiquement identique au timer 1, avec une légère différence.

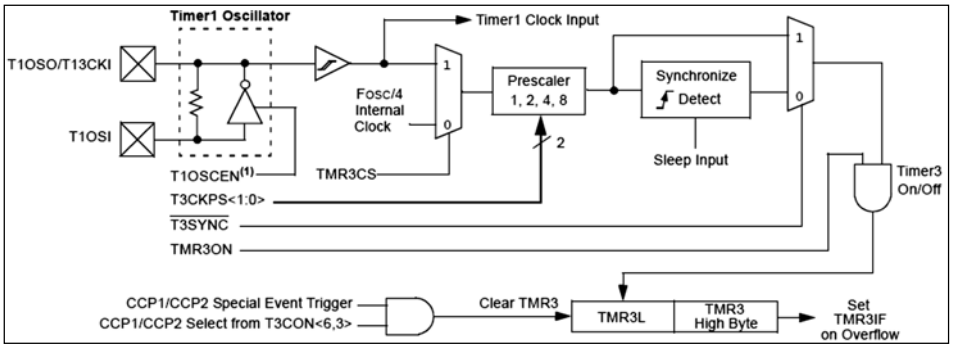


Figure V-5

Registre T3CON

R/W-0	R-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0	R/W-0
RD16	T3CCP2	T3CKPS1	T3CKPS0	T3CCP1	T3SYNC\	TMR3CS	TMR3ON
b7	b6	b5	b4	b3	b2	b1	b0

Bit 7 : **RD16 :** 16-Bit Read/Write Mode Enable bit

RD16 = 1 : valide la lecture ou l'écriture du timer en mode 16 bits

RD16 = 0 : valide la lecture ou l'écriture du timer en double octets

Bit 6..3 : **T3CCP<2:1>: Timer3 and Timer1 to CCPx Enable bits**

T3CCP2	T3CCP1	fonction
0	0	Timer1 utilisé par les modules CCP en mode capture/compare
0	1	Timer1 utilisé par le module CCP1 en mode capture/compare Timer3 utilisé par le module CCP2 en mode capture/compare
1	X	Timer3 utilisé par les modules CCP en mode capture/compare

Bit 5..4 : **T3CKPS1:T3CKPS0: Timer3 Input Clock Prescale Select bits**

T3CKPS1	T3CKPS0	Prédivision
0	0	1
0	1	2
1	0	4
1	1	8

bit 2 : **T3SYNC\: Timer3 External Clock Input Synchronization Control bit**

T3SYNC\ = 1 : Pas de synchronisation de l'horloge externe

T3SYNC\ = 0 : Synchronisation de l'horloge externe

Bit 1 : **TMR3CS: Timer1 Clock Source Select bit**

TMR3CS = 1 : horloge externe

TMR3CS = 0 : horloge interne

Bit 0 : **TMR3ON: Timer1 On bit**

TMR3ON = 1 : Timer 1 débloqué

TMR3ON = 0 : Timer 1 bloqué