PIC Microcontroleur

# **Microcontrôleurs**

# **INTRODUCTION GENERALE**

Un microcontrôleur est un circuit intégré qui réunisse 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. Les microcontrôleurs sont de véritables micro-ordinateurs intégrés sur une puce de silicium qui comportent une unité centrale, de la mémoire ou une interface à de la mémoire externe, des ports d’entrée-sortie, une interface pour lignes série (RS-232) ainsi qu’une unité de gestion de temps et d’événements...etc. Les microcontrôleurs se caractérisent par un plus haut degré d'intégration, une plus faible consommation électrique (quelques milliwatts en fonctionnement, quelques nano watts en veille), une vitesse de fonctionnement plus faible (quelques mégahertz à quelques centaines de mégahertz) et un coût réduit par rapport aux microprocesseurs utilisés dans les ordinateurs personnels. Les signaux d’entrée-sortie du microcontrôleur peuvent être facilement interfacés à des coupleurs optiques afin d’interfacer des capteurs et des actuateurs industriels.

Pratiquement tous les fabricants de microprocesseurs (Microship, Motorola, Intel, Hitachi, Texas Instrument, Toshiba, etc.) proposent une ou plusieurs gammes de microcontrôleurs. Les microcontrôleurs 4 bits servent essentiellement à des tâches simples. De tels microcontrôleurs sont par exemple utilisés au sein d’objets ménagers grand public, tels que des cuisinières, machines à laver ou aspirateurs. Les microcontrôleurs 8 bits sont capables de répondre à des exigences plus élevées et sont utilisés pour la commande de dispositifs informatiques tels que des joysticks, tablettes graphiques et modems. Ils sont également utilisés pour la programmation de petits robots ainsi que pour l’acquisition de données (convertisseurs A/D, etc.). Les microcontrôleurs 16/32 bits sont utilisés pour la commande de machines ou le contrôle de processus, lorsque les contraintes temps réel sont sévères ou lorsque les algorithmes de régulation nécessitent une puissance de calcul importante.

Dans ce support de cours nous allons définir le microcontrôleur PIC de Microship ainsi que les éléments contenus dans ce dernier. Afin de comprendre l’architecture d’un système à microcontrôleurs, le pic16f84 sera utilisé comme exemple dans la suite du chapitre. Nous allons aussi voir la manière de programmer un microcontrôleur avec un langage évolué (le langage C).

Afin de programmer un microcontrôleur il est indispensable de connaître sa structure interne: registres, mémoires, ports d’entrées sorties, et toutes leurs possibilités

# CHAPTER 1

##  Généralités notions fondamentales

### Généralités sur les PIC

Les PICs sont des microcontrôleurs à architecture RISC (Reduce Instructions Construction Set), ou encore composant à jeu d’instructions réduit. L'avantage est que plus on réduit le nombre d’instructions, plus leur décodage sera rapide ce qui augmente la vitesse de fonctionnement du microcontrôleur.

La famille des PICs est subdivisée en 3 grandes familles :

* La famille Base-Line, qui utilise des mots d’instructions de 12 bits,
* La famille Mid-Range, qui utilise des mots de 14 bits (et dont font partie les 16F84 et 16F876),
* La famille High-End, qui utilise des mots de 16 bits. Par la suite, d’autres familles sont apparues, comme la Enhanced family.

Nous nous limiterons dans ce support de cours à la famille Mid-Range et particulièrement au PIC16F876/877, sachant que si on a tout assimilé, on pourra facilement passer à une autre famille, et même à un autre microcontrôleur, sachant que la maitrise de cette dernière permet facilement le passage à une autre famille, et même à un autre microcontrôleur.

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **PIC** | **FLASH** | **RAM** | **EEPROM** | **I/O** | **A/D** | **Port //** | **Port Série** |
| 16F870 | 2K | 128 | 64 | 22 | 5 | NON | USART |
| 16F871 | 2K | 128 | 64 | 33 | 8 | PSP | USART |
| 16F872 | 2K | 128 | 64 | 22 | 5 | NON | MSSP |
| 16F873 | 4K | 192 | 128 | 22 | 5 | NON | USART/MSSP |
| 16F874 | 4K | 192 | 128 | 33 | 8 | PSP | USART/MSSP |
| 16F876 | 8K | 368 | 256 | 22 | 5 | NON | USART/MSSP |
| 16F877 | 8K | 368 | 256 | 33 | 8 | PSP | USART/MSSP |

Tableau 1.1 : différents circuit de la famille 16F87X

Les éléments essentiels du PIC 16F876 sont :

• Une mémoire programme de type EEPROM flash de 8K mots de 14 bits,

• Une RAM donnée de 368 octets,

• Une mémoire EEPROM de 256 octets,

• Trois ports d'entrée sortie, A (6 bits), B (8 bits), C (8 bits),

• Convertisseur Analogiques numériques 10 bits à 5 canaux,

• USART, Port série universel, mode asynchrone (RS232) et mode synchrone

• SSP, Port série synchrone supportant I2C

• Trois TIMERS avec leurs Prescalers, TMR0, TMR1, TMR2

• Deux modules de comparaison et Capture CCP1 et CCP2

• Un chien de garde,

• 13 sources d'interruption,

• Générateur d'horloge, à quartz (jusqu’ à 20 MHz) ou à Oscillateur RC

• Protection de code,

• Fonctionnement en mode sleep pour réduction de la consommation,

• Programmation par mode ICSP (In Circuit Serial Programming) 12V ou 5V,

• Possibilité aux applications utilisateur d’accéder à la mémoire programme,

• Tension de fonctionnement de 2 à 5V,

• Jeux de 35 instructions



Fig1.1. Brochage PIC16F877A



Fig1.2. Schéma block de l’architecture interne du PIC16F877A (voir datasheet page 6)

L'horloge peut être soit interne soit externe. L'horloge interne est constituée d'un oscillateur à quartz ou d'un oscillateur RC. Avec l'oscillateur à Quartz, on peut avoir des fréquences allant jusqu'à 20 MHz selon le type de µC. Le filtre passe bas (Rs, C1, C2) limite les harmoniques dus à l’écrêtage et Réduit l’amplitude de l’oscillation. Quel que soit l'oscillateur utilisé, l'horloge système dite aussi horloge instruction est obtenue en divisant la fréquence par 4. Avec un quartz de 4 MHz, on obtient une horloge instruction de 1 MHz, soit le temps pour exécuter une instruction de 1µs.



Fig 1.3. Horloge.

### L'ALU et l’accumulateur

L’unité ALU est une Unité Arithmétique et logique 8 Bits qui réalise les opérations arithmétiques et logique de base.

### Organisation de la mémoire RAM

L’espace mémoire RAM adressable est de 512 positions de 1 octet chacune :

• 96 positions sont réservées au SFR (Special Function Registers) qui
 sont les registres de configuration du PIC.

• Les 416 positions restantes constituent les registres GPR (General Propose Registers)
ou RAM utilisateur. Sur le 16F876 et 16F877, 3 blocs de 16 octets chacun ne sont pas implantés physiquement d’où une capacité de RAM utilisateur de 368 GPR.

### Les ports d’E/S

Le PIC 16F877 dispose de 33 broches d’entrée sortie regroupés dans 5 ports PORTA, PORTB, PORTC, PORTD et PORTE. Chaque broche d’un port peut être configurée soit en entrée soit en sortie à l’aide des registres de direction TRISA, TRISB, TRISC et TRISD et TRISE:

Bit k de TRISx = 0 broche k de PORTx = SORTIE

Bit k de TRISx = 1 broche k de PORTx = ENTRÉE

Certains ports ont quelques particularités que nous allons détailler ci-desous,

####  Le port d' E/S PORTA

Le port A désigné par PORTA est un port de 6 bits (RA0 à RA5). RA6 et RA7 ne sont pas accessibles. La configuration de direction se fait à l'aide du registre TRISA, positionner un bit de TRISA à 1 configure la broche correspondante de PORTA en entré et inversement. Au départ toutes les broches sont configurées en entrée

####  Pin RA4

En entrée, la pin RA4 peut être utilisée soit comme E/S numérique normale, soit comme En sortie, RA4 est une E/S à drain ouvert, pour l'utiliser comme sortie logique, il faut ajouter une résistance de pull-up externe. Le schéma (Fig.1.4.) illustre (pour les non électroniciens). Le principe d'une sortie drain ouvert (ou collecteur ouvert): si RA4 est positionnée à 0, l'interrupteur est fermé, la sortie est reliée à la masse, c'est un niveau bas. Si RA4 est placée résistance externe qui place la sortie au niveau haut.



Fig1.4. résistance de Pull-up

On peut utiliser le schéma. Il faut juste remarquer que la logique est inversée, si on envoie 0 sur RA4, l'interrupteur se ferme et la LED s'allume. Si on envoie 1, l'interrupteur s'ouvre et la LED s'éteint. Le schéma illustré sur Fig.IV.3 peut aussi être utilisé. La logique n'est pas inversée mais il demande une précaution particulière. Il ne faut pas positionner la sortie RA4 à l'aide d'une instruction qui réalise une opération sur l'état actuel du port. Si la sortie était niveau haut, l'interrupteur est ouvert, la LED est allumée et elle impose une tension de l'ordre de 1.5V qui sera considérée (à tort) comme un niveau bas lors de la lecture du port par les instructions précitées. La solution est d'utiliser des instructions qui positionnent le PORT sans tenir

####  Les autres broches de PORTA

Les autres broches (RA0, RA1, RA2, RA3 et RA5) peuvent être utilisée soit comme E/S numériques soit comme entrées analogiques. Au RESET, ces E/S sont configurées en entrées analogiques. Pour les utiliser en E/S numériques, il faut écrire '00000110' dans le registre ADCON1 (pour plus de détail, voir datasheet).

Pour utiliser PORTA en Numérique (normal), il faut placer 06h dans le registre
 ADCON1 (bank1)

Quel que soit le mode (Analogique ou Numérique), il faut utiliser le registre TRISA pour configurer la direction des E/S :

* Bit i de TRISA = 0 bit i de PORTA configuré en sortie
* Bit i de TRISA = 1 bit i de PORTA configuré en entrée

####  Le port d'E/S PORTB

* Le port B désigné par PORTB est un port bidirectionnel de 8 bits (RB0 à RB7). Toutes les broches sont compatibles TTL.
* La configuration de direction se fait à l'aide du registre TRISB, positionner un bit de TRISB à 1 configure la broche correspondante de PORTB en entré et inversement. Au départ toutes les broches sont configurées en entrée.
* En entrée, la ligne RB0 appelée aussi INT peut déclencher l’interruption externe INT.
* En entrée, une quelconque des lignes RB4 à RB7 peut déclencher l'interruption RBI. Nous reviendrons là-dessus dans le paragraphe réservé aux interruptions.

####  Le port d' E/S PORTC

* Le port C désigné par PORTC est un port bidirectionnel de 8 bits (RC0 à RC7). Toutes les broches sont compatibles TTL.
* La configuration de direction se fait à l'aide du registre TRISC, positionner un bit de TRISC à 1 configure la broche correspondante de PORTC en entré et inversement. Au départ toutes les broches sont configurées en entrée.
* Toutes les broches du port C peuvent être utilisées soit comme E/S normales soit comme broches d'accès à différents modules comme le timer 1, les modules de comparaison et de capture CCP1/2, le timer 2, le port I2C ou le port série
* Pour l’utilisation d’une broche du port C comme E/S normale, il faut s’assurer qu’elle n’a pas été affectée à un de ces modules. Par exemple, si TIMER1 est validé, il peut utiliser les broches RC0 et RC1 selon sa configuration.

####  Le port d'E/S PORTD

* Le port D désigné par PORTD est un port bidirectionnel de 8 bits (RD0 à RD7). Toutes les broches sont compatibles TTL et ont la fonction trigger de Schmitt en entrée.
* Chaque broche et configurable en entrée ou en sortie à l’aide du registre TRISD. Pour configurer une broche en entrée, on positionne le bit correspondant dans TRISD à 1 et inversement.
* PORTD n’est pas implémenté sur tous les processeurs 16F87X, il est disponible sur le 16F877, le 16F874 et le 16F871
* PORTD peut être utilisé dans un mode particulier appelé parallel slave port, pour cela il faut placer le bit PSPMODE (bit 4) de TRISE à 1. Dans ce cas les 3 bits de PORTE deviennent les entrées de control de ce port (RE, WE et CS)

Pour utiliser PORTD en mode normal, il faut placer le bit PSPMODE de TRISE à 0

####  Le port d'E/S PORTE

* PORTE contient seulement 3 bits RE0, RE1 et RE2. Les 3 sont configurables en entrée ou en sortie à l’aide des bits 0, 1 ou 2 du registre TRISE.
* PORTE n’est pas implémenté sur tous les processeurs 16F87X, il est disponible sur le 16F877, le 16F874 et le 16F871
* Les 3 bits de PORTE peuvent être utilisés soit comme E/S numérique soit comme entrées analogiques du CAN. La configuration se fait à l’aide du registre ADCON1.
* Si le bit PSPMODE de TRISE est placé à 1, Les trois bits de PORTE deviennent les entrées de control du PORTD qui (dans ce cas) fonctionne en mode parallel Slave mode
* A la mise sous tension (RESET), les 3 broches de PORTE sont configurées comme entrées analogiques.

Pour utiliser les broches de PORTE en E/S numériques normales :

- Placer 06h dans ADCON1

### Les mémoires permanentes

Le PIC 16F877 dispose de 2 mémoires permanentes. La mémoire EEPROM PROGRAMME de capacité 8k mots de 14 bits et la mémoire EEPROM DE DONNÉES de capacité 56 octets. On dispose de deux méthodes pour écrire dans ces mémoires.

La première consiste à flasher le PIC avec l'exécutable (.hex). Les instructions sont flashés dans la EEPROM programme et les donnée sont flashés dans la EEPROM de données. Le flashage se fait à l’aide d’un programmeur et le soft qui va avec.

La deuxième méthode consiste à accéder aux mémoires EEPROM à partir du programme durant la phase d'exécution de ce dernier. C’est ce que nous allons détailler dans les paragraphes ci-dessous.

####  La mémoire EEPROM de données

Le PIC 16F876/877 dispose de 256 octets de mémoire EEPROM de donnée. Son implantation physique commence à la position d’adresse absolue 2100h. Mais pour y accéder à partir des programmes utilisateur on utilise l’adressage relatif par rapport à la première position. La première position aura l’adresse 0, la deuxième aura l’adresse 1 et la dernière aura l’adresse 255. Pour accéder à la EEPROM, on utilise 4 registres particuliers :

• EEADR : registre d’adresse (relative) (bank 2)

• EEDATA : registre de donnée (bank 2)

• EECON1 : registre de control (bank 3)

• EECON2 : 2ème registre de control (bank 3)

Le registre EECON1 :

EEPGD : Accès à la mémoire EEPROP ou à la mémoire Programme

* 0 : EEPROM de données
* 1 : Mémoire programme (flash)

WRERR : Erreur d'écriture (indicateur)

* 0 : Pas d'erreur
* 1 : Une erreur s'est produite

WREN : Validation de l'écriture dans l'EEPROM

* 0 : Ecriture interdite
* 1 : Ecriture autorisée

WR : Write Enable. Ce bit doit être mis à 1 pour démarrer l'écriture d'un octet. Il est remis
à zéro automatiquement à la fin de l'écriture. Ce bit ne peut pas être mis à zéro par une instruction.

RD : Read Enable. Ce bit doit être mis à 1 pour démarrer la lecture d'un octet. Il est remis
à zéro automatiquement à la fin de la lecture. Ce bit ne peut pas être mis à zéro par une instruction

####  Procédure de lecture dans l'EEPROM

Pour lire le contenu d’une position de la mémoire EEPROM, on place l’adresse dans le registre

EEADR, on lance l’opération de lecture à l’aide du bit RD du registre EECON1, la donnée désirée

est tout de suite disponible dans le registre EEDATA :

1) Mettre le bit EEPGD à 0 pour pointer sur l'EEPROM de donnée

2) Placer l’adresse relative de la position à lire dans EEADR

3) Mettre le bit RD à 1 pour démarrer la lecture. Ce bit revient à 0 automatiquement tout de suite après le transfert de la donnée vers EEDATA, (moins d'un cycle)

4) Traiter la donnée disponible dans EEDATA

5) Recommencer au point 2 si on a d'autres données à lire,

* Procédure d'écriture dans l'EEPROM

Pour écrire une donnée dans une position de la mémoire EEPROM, on place l’adresse dans le registre EEADR, la donnée dans le registre EEDATA, on lance l’opération d’écriture à l’aide du bit WR de EECON1 et du registre EECON2. La donnée présente dans EEDATA est alors copiée dans le EEPROM mais cette opération prend 10 ms. A la fin de l'écriture le bit WR revient à zéro automatiquement, le drapeau EEIF est levé ce qui peut déclencher l'interruption EEI si elle a été validée auparavant :

1) Interdire les interruptions (si elles ont été validées avant : bit INTCON.GIE)

2) Mettre le bit EEPGD à 0 pour pointer sur l'EEPROM de donnée

3) Positionner le bit WREN pour valider l'écriture dans l'EEPROM

4) Placer l’adresse relative de la position à écrire dans EEADR

5) Placer la donnée à écrire dans le registre EEDATA

6) Ecrire 55h dans EECON2 (commande de process hardwares)

Ecrire AAh dans EECON2 (commandes de process hardwares) Positionner le bit WR pour démarrer l'opération d'écriture, et attendre qu’il
revienne à 0

7) Recommencer au point (4) si on a d'autres données à écrire,

Remarque : Les bits WREN et WR ne peuvent être positionné dans la même instruction. WR ne peut être positionné que si le bit WREN a été positionné avant.

### Les interruptions

Une interruption provoque l’arrêt du programme principal pour aller exécuter une procédure d'interruption. A la fin de cette procédure, le microcontrôleur reprend le programme principal à l’endroit où il l’a laissé. A chaque interruption sont associés deux bits, un bit de validation et un drapeau. Le premier permet d'autoriser ou non l'interruption, le second permet au programmeur de savoir de quelle interruption il s'agit. Sur le 16F876/877, l'es interruptions sont classées en deux catégories, les interruptions primaires et les interruptions périphériques. Elles sont gérées par les registres :

INTCON

PIE1 (bk1)

PIR1 (bk0)

PIE2 (bk0)

PIR2 (bk1)

OPTION\_REG(bk1)

• Toutes les interruptions peuvent être validées/interdites par le bit INTCON.GIE

• Toutes les interruptions périphériques peuvent être validées/interdites par le bit INTCON.PEIE

• Chaque interruption peut être validée/interdite par son bit de validation individuel

En résumé, pour valider une interruption périphérique (par exemple), il faut positionner 3 bits, GIE, PEIE et le bit individuel de l’interruption.

####  Déroulement d'une interruption

Lorsque l'événement déclencheur d'une interruption intervient, alors son drapeau est positionné à 1. Si l'interruption a été validée (bits de validations = 1), elle est alors déclenchée: le programme arrête ce qu'il est en train de faire et va exécuter la procédure d'interruption qui se trouve à l'adresse 4 en exécutant les étapes suivantes :

• l'adresse contenue dans le PC (Program Counter) est sauvegardée dans la pile, puis remplacée par la valeur 0004 (adresse de la routine d'interruption).

• Le bit GIE est placé "0" pour inhiber toutes les interruptions (afin que le PIC ne soit pas dérangé pendant l'exécution de la procédure d'interruption).

• A la fin de la procédure d'interruption (instruction RETFIE) :

• le bit GIE est remis à 1 (autorisant ainsi un autre événement)

• le contenu du PC est rechargé à partir de la pile ce qui permet au programme de reprendre là où il s'est arrêté

Il est très important de noter:

Le drapeau reste à l’état haut même après le traitement de l’interruption. Par conséquent, il faut toujours le remettre à "0" à la fin de la routine d'interruption sinon l'interruption sera déclenchée de nouveau juste après l'instruction RETFIE. Seul le PC est empilé automatiquement. Si cela est nécessaire, les registres W et STATUS doivent être sauvegardés en RAM puis restaurés à la fin de la routine pour que le microcontrôleur puisse reprendre le programme principal dans les mêmes conditions où il l'a laissé.

####  Les sources d'interruption

|  |  |  |  |
| --- | --- | --- | --- |
| **Interruption** : Source d’interruption | Validation | Flag | PEIE |
| **T0I** : Débordement Timer 0 | INTCON,T0IE | INTCON,T0IF | non |
| **INT** : Front sur RB0/INT | INTCON,INTE | INTCON,INTF | non |
| **RBI** : Front sur RB4-RB7 | INTCON,RBIE | INTCON,RBIF | non |
| **ADI** : Fin de conversion A/N | PIE1,ADIE | PIR1,ADIF | oui |
| **RCI** : Un Octet est reçu sur l'USART | PIE1,RCIE | PIR1,RCIF | oui |
| **TXI** : Fin transmission d'un octet sur l'USART | PIE1,TXIE | PIR1,TXIF | oui |
| **SSPI** : Caractère émis/reçu sur port série synchrone | PIE1,SSPIE | PIR1,SSPIF | oui |
| **TMR1I** : Débordement de Timer 1 | PIE1,TMR1IE | PIR1,TMR1IF | oui |
| **TMR2I** :Timer 2 a atteint la valeur programmée | PIE1,TMR2IE | PIR1,TMR2IF | oui |
| **PSPI** : Lecture/écriture terminée sur Port parallèle (16F877) | PIE1,PSPIE | PIR1,PSPIF | oui |
| **CCP1I** : Capture/comparaison de TMR1 avec module CCP1 | PIE1,CCP1IE | PIR1,CCP1IF | oui |
| **CCP2I** : Capture/comparaison de TMR1 avec module CCP2 | PIE2,CCP2IE | PIR2,CCP2IF | oui |
| **EEI** : Fin d'écriture en EEPROM | PIE2,EEIE | PIR2,EEIF | oui |
| **BCLI** : Collision sur bus SSP en mode I2C | PIE2,BCLIE | PIR2.BCLIF | oui |

### L'interruption INT (Entrée RB0 du port B)

Cette interruption est provoquée par un changement d'état sur l'entrée RB0 du port B quand elle est programmée en entrée. En plus de son bit de validation INTE et son drapeau INTF, elle est gérée aussi par le bit INTEDG (OPTION\_REG) qui détermine le front sur lequel l'interruption se déclenche, 1=montant, 0=descendant

### L'interruption RBI (RB4 A RB7 du port B)

Cette interruption est provoquée par un changement d'état sur l'une des entrées RB4 à RB7 du port B, Le front n'a pas d'importance. Les bits associés sont RBIE (validation) et RBIF (drapeau).

**ATTENTION** : Le drapeau RBIF ne peut être remis à zéro sans la lecture préalable de PORTB (MOVF PORTB,w). Si on ne veut pas modifier le contenu de W, on peut copier PORTB sur lui- même (MOVF PORTB,f).

### Les Timers

####  Le Timer TMR0

C’est un compteur 8 bits ayant les caractéristiques suivantes :

• Il est incrémenté en permanence soit par l’horloge interne Fosc/4 (mode timer) soit par une horloge externe appliquée à la broche RA4 du port A (mode compteur). Le choix de l'horloge se fait à l'aide du bit T0CS du registre OPTION\_REG

• T0CS = 0 horloge interne

• T0CS = 1 horloge externe appliquée à RA4

• Dans le cas de l'horloge externe, Le bit T0SE du registre OPTION\_REG permet de choisir le front sur lequel le TIMER s'incrémente.

• T0SE = 0 incrémentation sur fronts montants

• T0SE = 1 incrémentation sur fronts descendants

• Quel que soit l'horloge choisie, on peut la passer dans un diviseur de fréquence programmable (prescaler) dont le rapport DIV est fixés par les bits PS0, PS1 et PS2 du registre OPTION\_REG (tableau ci-contre).

|  |  |  |  |
| --- | --- | --- | --- |
| **PS2** | **PS1** | **PS0** | **Div** |
| 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 |

 L'affectation ou non du prédiviseur se fait à l'aide du bit PSA du registre OPTION\_REG

• PSA = 0 on utilise le prédiviseur

• PSA = 1 pas de prédiviseur (affecté au chien de garde)

• Le contenu du timer TMR0 est accessible par le registre qui porte le même nom. Il peut être lu ou écrit à n'importe quel moment. Après une écriture, le timer ne compte pas pendant deux cycles machine.

• **Au débordement de TMR0 (FF = 00), le drapeau INTCON.T0IF est placé à 1. Ceci peut déclencher l'interruption T0I si celle-ci est validée**

|  |  |
| --- | --- |
| OPTION\_REG |  |



#### Le Watchdog Timer WDT (Chien de garde)

C’est un compteur 8 bits incrémenté en permanence (même si le µC est en mode sleep) par une horloge RC intégrée indépendante de l'horloge système. Lorsqu’il déborde, (WDT TimeOut), deux situations sont possibles :

• Si le µC est en fonctionnement normal, le WDT time-out provoque un RESET. Ceci permet d’éviter de rester planté en cas de blocage du microcontrôleur par un processus indésirable non contrôlé

• Si le µC est en mode SLEEP, le WDT time-out provoque un WAKE-UP, l'exécution du programme continue normalement là où elle s'est arrêtée avant de rentrer en mode SLEEP. Cette situation est souvent exploitée pour réaliser des temporisations

L'horloge du WDT a une période voisine de 70 µs ce donne un Time-Out toutes les 18 ms. Il est cependant possible d'augmenter cette durée en faisant passé le signal Time-Out dans un prédiviseur programmable (partagé avec le timer TMR0). L'affectation se fait à l'aide du bit PSA du registre OPTION\_REG

• PSA = 1 on utilise le prédiviseur

• PSA = 0 pas de prédiviseur (affecté à TMR0)

|  |  |  |  |
| --- | --- | --- | --- |
| PS2 | PS1 | PS0 | Div |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 2 |
| 0 | 1 | 0 | 4 |
| 0 | 1 | 1 | 8 |
| 1 | 0 | 0 | 16 |
| 1 | 0 | 1 | 32 |
| 1 | 1 | 0 | 64 |
| 1 | 1 | 1 | 128 |

Le rapport du prédiviseur est fixé par les bits PS0, PS1 et PS2 du registre OPTION\_REG (voir tableau ci-contre)

L'utilisation du WDT doit se faire avec précaution pour éviter la réinitialisation (inattendue) répétée du programme. Pour éviter un WDT time Out lors de l'exécution d'un programme, on a deux possibilités :

• Inhiber le WDT d'une façon permanente en mettant à 0 le bit WDTE dans l'EEPROM de configuration

• Remettre le WDT à 0 périodiquement dans le programme à l'aide de l'instruction CLRWDT pour éviter qu'il ne déborde



#### Le Timer TMR1

TMR1 est un Timer/Compteur 16 bits accessible en lecture/écriture par l'intermédiaire des registres 8 bits TMR1H (bank0) et TMR1L (bank0) qui constituent sa partie haute et sa partie basse. On le configure à l'aide du registre T1CON (bank0).



• TMR1 peut fonctionner dans 3 modes différents :

* + Timer Synchrone (horloge interne)
	+ Compteur Synchrone (horloge externe)
	+ Compteur Asynchrone (horloge externe)

Le bit de contrôle TMR1CS détermine le fonctionnement en Timer ou en Compteur et le bit T1SYNC détermine le mode de fonctionnement en synchrone ou en asynchrone

• TMR1 peut être arrêté/démarré à l'aide du bit TMR1ON

• TMR1 peut être RAZ à l'aide du module de capture/comparaison CCP

• TMR1 peut être précédé d'un prédiviseur (Prescaler) qui peut diviser la fréquence par 1, 2, 4 ou 8 selon la valeur des bits T1CKPS1 et T1CKPS0

• A son débordement (FFFFh = 0000h) le drapeau PIR1.TMR1IF (bank0) est positionné ce qui peut déclencher l'interruption périphérique TMR1I si elle est validée par son bit de validation PIE1.TMR1IE (bank1).

* Le mode Timer

Dans ce mode, TMR1 est incrémenté par l’horloge système Fosc/4 éventuellement prédivisée. Le bit de synchronisation n'a pas d'effet car l'horloge Fosc/4 est toujours synchronisée sur l'horloge système.

* Le mode Compteur

Dans ce mode, TMR1 est incrémenté à chaque front montant de l'horloge externe T1CKI (RC0) ou l'horloge dédiée générée par l’oscillateur T1OSC à condition de positionner le bit T1OSCEN à 1 et de brancher un quartz entre les broche RC0 et RC1.

**En mode compteur, RC0 et RC1 sont automatiquement configurées en entrée, on n’a pas besoin de configurer les bits TRISC, 0 et TRISC 1**

• En fonctionnement Synchrone, l'horloge externe (éventuellement prédivisée) n'incrémente pas directement le timer mais elle est synchronisée sur l'horloge système ce qui peut entraîner un délai de l'ordre de 1 cycle machine. Dans cette configuration

• En fonctionnement Asynchrone, l'horloge externe (éventuellement prédivisée) incrémente le timer indépendamment de l'horloge système.

**En mode Compteur Asynchrone, on ne peut pas utiliser les modules CCP1 et CCP2 pour faire des captures ou des comparaisons sur TMR1**

* Le registre de control de T1CON



T1CKPS1,T1CKPS0 : Control du prescaler

00 : division par 1

01 : division par 2

10 : division par 4

11 : division par 8

T1OSCEN : Validation de l'Oscillateur associé à TMR1

0 : Oscillateur arrêté

1 : Oscillateur activé

T1SYNC : Synchronisation de l'horloge externe (ignoré en mode timer)

0 : Synchronisation

1 : pas de synchronisation

TMR1CS : Choix de l'horloge du Timer

0 : horloge système (Fosc/4) : mode timer

1 : Horloge externe : mode compteur

TMR1ON : Démarrer arrêter le timer

0 : Timer stoppé

1 : Timer en fonctionnement

Le temps qui sépare 2 levés de drapeau TMR1IF est : TF1 = 65536 x T1 = 65536 x (DIV x TH) Si on ajoute un compteur N dans le programme on obtient un temps = N x TF1

####  Les module de Comparaison/Capture CCP1 et CCP2

Chacun des modules CCP1 et CCP2 permet :

* Soit de CAPTURER en un seul coup le contenu du double registre TMR1
* Soit de COMPARER en permanence son contenu avec un registre 16 bits et de déclencher un événement au moment de l’égalité.

Ces modules ne fonctionnent pas si TMR1 est configuré en mode Compteur non synchronisé

#### Le module CCP1

Ce module est constitué de :

* Un registre 16 bits CCPR1 utilisé pour la capture ou la comparaison de TMR1. Il est accessible par sa partie basse CCP1L et sa partie haute CCP1H
* Un registre de contrôle 8 bits CCP1CON.
* Un prédiviseur permettant de filtrer les événements déclencheurs de capture venant de la broche RC2
* Le mode Capture



Dans ce mode le contenu de TMR1 est copié dans CCPR1 chaque fois qu'un événement intervient sur la broche RC2. Le choix de l'événement déclencheur se fait en programmant le prescaler à l’aide des bits 0 à 3 du registre de contrôle CCP1CON. On a le choix parmi les événements suivants :

* A chaque front descendant
* A chaque front montant (prescaler 1:1)
* A chaque 4ème front montant (prescaler 1:4)
* A chaque 16ème front montant (prescaler 1:16)

A la fin de la capture, le drapeau CCP1IF est positionné, l'interruption périphérique associée est déclenchée si elle a été validée. Plusieurs aspects sont à noter comme :

* Si on veut déclencher la capture par un signal externe, celui-ci doit être appliqué sur la pin RC2 qui doit être configurée en entrée par le bit TRISC,2
* Si on veut déclencher la capture par programme en changeant la valeur du bit RC2, celui-ci

doit être configuré en sortie par le bit TRISC,2

* Lors de la modification du mode de capture, une interruption indésirable peut intervenir. L'utilisateur doit veiller à masquer l'interruption CCP1I avant de procéder à cette modification et de baisser le drapeau CCP1IF après la modification.
* En mode Sleep, le prescaler et le drapeau CCP1IF restent opérationnels. le positionnement du drapeau peut déclencher un Wake-up si le bit de validation CCP1IE a été validé auparavant
* Le registre de configuration CCP1CON

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| — | — | DC1B1 | DC1B0 | CCP1M3 | CCP1M2 | CCP1M1 | CCP1M0 |

**DC1B1, DC1B0 :** utilisés en PWM mode Ce sont les 2 bits de poids faible des 10 bits MWM duty cycle. Les 8 autres bits sont dans le registre CCPR1L

**CCP1M3 à CCP1M0** : mode de fonctionnement du module CCP1

0000 : Module arrêté (reset module)

0100 : Capture sur chaque front descendant

0101 : Capture sur chaque front montant

0110 : Capture tous les 4 fronts montants

0111 : capture tous les 16 fronts montants

1000 : Mode comparaison (drapeau CCP1IF + broche RC2 0 = 1)

1001 : Mode comparaison (drapeau CCP1IF + broche RC2 1 = 0)

1010 : Mode comparaison (drapeau CCP1IF seulement)

1011 : Mode comparaison (drapeau CCP1IF + RAZ TMR1)

10xx : PWM mode (modulation de largeur d’impulsion)

* Le mode Comparaison

Dans ce mode le registre CCPR1 est comparé en permanence à TMR1. Quand l'égalité intervient, le drapeau CCP1IF passe à 1 et différentes actions sont accomplies selon le mode défini par les bits de configuration CCP1M3:CCP1M0



**mode 1000 :**

Au moment de l'égalité, le drapeau CCP1IF est le bit RC2 passent à 1. C'est à l'utilisateur de les remettre à 0 pour une prochaine utilisation. RC2 doit être configuré en sortie.

**mode 1001 :**

Au moment de l'égalité, le drapeau CCP1IF passe à 1 et le bit RC2 passe à 0. C'est à l'utilisateur de les remettre à leur état d’origine pour une prochaine utilisation. RC2 doit être configuré en sortie.

**mode 1010 :**

A l'égalité le drapeau CCP1IF passe à 1. La broche RC2 n'est pas utilisée.

**mode 1011 :**

A l'égalité, le drapeau CCP1IF passe à 1 et le timer TMR1 est remis à 0

**VII.4.2 Le module CCP2**

Le module CCP2 est identique au module CCP1, il suffit d’interchanger 1 et 2 et de constater les points suivants :

* CCP2 est associé à la broche RC1
* Il est géré par le registre de control CCP2CON
* Son registre de capture/comparaison est CCPR2 = CCPR2H:CCPR2L
* En mode comparaison 1011, A l’égalité :

 o le drapeau CCP2IF passe à 1

o RAZ de TMR1

o envoi d’un GO vers le convertisseur analogique numérique

### Le Timer TMR2



TMR2 est un timer 8 bits accessible en lecture écriture constitué de :

* + un registre de control T2CON (bank0)
	+ un prédiviseur (1,4,16)
	+ un registre de période PR2 (bank1) accessible en lecture/écriture
	+ un comparateur,
	+ un postdiviseur (1 à 16)
	+ TMR2 est incrémenté par l'horloge interne Fosc/4. Il commence à 0 et quant il atteint la valeur du registre PR2, le comparateur génère un signal qui :
		- Remet TMR2 à 0
		- incrémente le postscaler qui fonctionne comme un diviseur de fréquence

Comme le comptage commence à 0, si PR2=N, alors le comparateur annonce une égalité tous les N+1 coups d’horloge

* + Au débordement du postscaler, le drapeau PIR1.TMR2IF est positionné, l'interruption correspondante et déclenchée si elle est validée
	+ TMR2 est remis à zéro à chaque RESET
	+ Le prescaler et le postscaler sont initialisés à chaque écriture dans TMR2 ou dans T2CON et au RESET du processeur
	+ Le fonctionnement de TMR2 est configuré à l'aide du registre de control T2CON :

En résumé, Si on note :

DIV1 : rapport du prédiviseur

DIV2 : Rapport du postdiviseur

P : Valeur placée dans le registre PR2

Tsy : période de l'horloge système,

TF2 = DIV1 \* (P+1) \* DIV2 \* Tsy

### Le module de conversion A/N



Ce module est constitué d'un convertisseur Analogique Numérique 10 bits dont l'entrée analogique peut être connectée sur l'une des 8 (5 pour 16F876) entrées analogiques externes. On dit qu'on a un CAN à 8 canaux. Les entrées analogiques doivent être configurées en entrée à l'aide des registres TRISA et/ou TRISE. L’échantillonneur bloqueur est intégré, il est constitué d’un interrupteur d’échantillonnage et d’une capacité de blocage de 120 pF. Les tensions de références permettant de fixer la dynamique du convertisseur. Elles peuvent être choisies parmi Vdd, Vss, Vr+ ou Vr-.

Le control du module se fait par les deux registres ADCON0 et ADCON1

**ADCON0**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ADCS1 | ADCS0 | CHS2 | CHS1 | CHS0 | GO/DONE | — | ADON |

**ADCS1:ADCS0** : Choix de l'horloge de conversion donc du temps de

* 00 : Fosc/2
* 01 : Fosc/8
* 10 : Fosc/32
* 11 : Oscillateur RC dédié au CAN

**CHS2:CHS0 :** choix de l'entrée analogique

* 000 = channel 0, (RA0)
* 001 = channel 1, (RA1)
* 010 = channel 2, (RA2)
* 011 = channel 3, (RA3)
* 100 = channel 4, (RA5)
* 101 = channel 5, (RE0)
* 110 = channel 6, (RE1)
* 111 = channel 7, (RE2)

**GO/DONE :** Une conversion démarre quand on place ce bit à 1. A la fin de la conversion, il est remis automatiquement à zéro. Ce bit peut aussi être positionné automatiquement par le module CCP2.

**ADON** : Ce bit permet de mettre le module AN en service

**ADCON1**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| ADFM | — | — | — | PCFG3 | PCFG2 | PCFG1 | PCFG0 |

**ADFM :** justification à droite ou à gauche du résultat dans les registre ADRESH et ADRESL

 ADRESH ADRESL

1 : justifié à droite 000000XX XXXXXXXX

0 : justifié à gauche XXXXXXXX XX000000

**PCFG3:PCFG0** : configuration des E/S et des tensions de références. Les 5 broches de PORTA et les 3 de PORTE peuvent être configurés soit en E/S digitales, soit en entrées analogiques. RA2 et RA3 peuvent aussi être configurées en entrée de référence.

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| PCFG3:PCFG0 | RE2 | RE1 | RE0 | RA5 | RA3 | RA2 | RA1 | RA0 | VREF+ | VREF- | A/R/N |
| 0000 | A | A | A | A | A | A | A | A | VDD | VSS | 8/0/0 |
| 0001 | A | A | A | A | VREF+ | A | A | A | RA3 | VSS | 7/1/0 |
| 0010 | N | N | N | A | A | A | A | A | VDD | VSS | 5/0/3 |
| 0011 | N | N | N | A | VREF+ | A | A | A | RA3 | VSS | 4/1/3 |
| 0100 | N | N | N | N | A | N | A | A | VDD | VSS | 3/0/5 |
| 0101 | N | N | N | N | VREF+ | N | A | A | RA3 | VSS | 2/1/5 |
| 011x | N | N | N | N | N | N | N | N | VDD | VSS | 0/0/8 |
| 1000 | A | A | A | A | VREF+ | VREF- | A | A | RA3 | RA2 | 6/2/0 |
| 1001 | N | N | A | A | A | A | A | A | VDD | VSS | 6/0/2 |
| 1010 | N | N | A | A | VREF+ | A | A | A | RA3 | VSS | 5/1/2 |
| 1011 | N | N | A | A | VREF+ | VREF- | A | A | RA3 | RA2 | 4/2/2 |
| 1100 | N | N | N | A | VREF+ | VREF- | A | A | RA3 | RA2 | 3/2/3 |
| 1101 | N | N | N | N | VREF+ | VREF- | A | A | RA3 | RA2 | 2/2/4 |
| 1110 | N | N | N | N | N | N | N | A | VDD | VSS | 1/0/7 |
| 1111 | N | N | N | N | VREF+ | VREF- | N | A | RA3 | RA2 | 1/2/5 |

### L'USART

L'USART (Universal Synchronous Asynchronous Receiver Transmitter) est l'un des deux modules de communication série dont dispose le PIC 16F876/877. L'USART peut être configuré comme système de communication asynchrone full duplex ou comme système synchrone half duplex (non étudié).

La communication se fait sur les deux broches RC6/TX et RC7/RX qui doivent être configurés toutes les deux en ENTREE par TRISC.

#### Mode Asynchrone

Si on place le bit SYNC du registre TXSTAT à 0, l'USART fonctionne dans le mode asynchrone standard, 10 (ou 11) bits sont transmis ou reçus dans l'ordre ci-dessous :

* 1 bit de START (toujours 0)
* 8 ou 9 bits de donnée (LSB d'abord)
* 1 bits de STOP (toujours 1)
* La transmission se fait sur la broche RC6/TX et la réception sur la broche RC7/RX
* La configuration et le control du port se fait par les registres TXSTA et RCSTA
* La vitesse de communication est fixée par le registre SPBRG et le bit TXSTA.BRGH
* La parité n'est pas gérée d'une façon matérielle, elle peut être gérée par soft si son utilisation est nécessaire.
* L'accès au port en lecture ou écriture se fait par les registres tampon RCREG et TXREG. La transmission et la réception se font par deux registres à décalage, un pour la transmission (TSR) et un pour la réception (RSR). L'accès au registres tampon peut se faire alors que les registre à décalage sont en train de transmettre/recevoir une donnée.
* Les drapeaux PIR1.RCIF et PIR1.TXIF sont très utiles pour gérer la lecture/écriture dans le port. RCIF est positionné quand le port a terminé une réception et TXIF est positionné quand le buffer de transmission TXREG est "vide".



Le contrôle de la transmission se fait par le registre **TXSTA**



**CSRC** : non utilisé en mode asynchrone

**TX9** et **TX9D** : Pour utiliser le mode 9 bits il faut positionner le bit TX9. Le 9ème bit doit être écrit dans TX9D avant d'écrire les 8 bits de données dans TXREG

**TXEN** : permet de valider ou interdire la transmission

**SYNC** : 0  mode asynchrone, 1  mode synchrone

**BRGH** : sélectionne le mode haut débit du générateur de baud rate

**TRMT** : Indicateur de l’activité du registre à décalage de transmission TSR