REPUBLIQUE ALGERIENNE DEMOCRATIQUE ET POPULAIRE
MINISTERE DE L'ENSEIGNEMENT SUPERIEUR
ET DE LA RECHERCHE SCIENTIFIQUE

UNIVERSITE ABOU BAKR BELKAID - TLEMCEN
FACULTE DE TECHNOLOGIE
DEPARTEMENT DE GENIE ELECTRIQUE ET ELECTRONIQUE
INSTRUMENTATION ELECTRONIQUE

Mémoire de Fin d'Etudes

En vue de l'obtention du diplôme:

MASTER

EN

INSTRUMENTATION ELECTRONIQUE

Présenté par :

- ✓ **MELLE : BENYETTOU AICHA**
- ✓ **MELLE : AMMOUR KHADIDJA**

THEME

ETUDE ET REALISATION D'UN FREQUENCEMETRE MULTI CALIBRE

Soutenu en JUIN 2015 devant le jury :

Président : MR/HAMMDOUN ABED ELKADER	*MC CLASSE A*
Examinateur: MR/BECHAR HASSANE	*MC CLASSE A*
Encadreur : MR/ NEMMICHE AHMED	*MC CLASSE A*

Année universitaire : 2014 - 2015

Remercîment

Avant tout, on rend grâce à DIEU tout puissant de nous avoir accordé la volonté et le courage

pour réaliser ce mémoire

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma reconnaissance.

J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé nos réflexions et ont accepté de nous rencontrer et répondre à nos questions durant notre recherches.

En exception MR_Bechar Hassan et MR-Hammdoun Abdel kader.

Au terme de ce travail je tiens tout d'abord à exprimer ma profonde gratitude à mon encadreur

Mr Nemiche Ahmed.

Mes remerciements s'adressent à tous les membres du jury pour l'honneur qu'ils nous ont fait en acceptant de juger ce travail.

Enfin, je remercie tous mes ami(e)s que j'aime tant Khadija-Khadîdja-Khadîdja-a ,Tema, Dalila et aussi notre nouvelle ami Boumediene.

Pour leur sincère amitié et confiance, et à qui je dois ma reconnaissance et mon attachement.

À tous ces intervenants, nous présentons nos remerciements, nos respects et notre gratitude.

Mercie a tous

Dédicace

GRACE A DIEU TOU PUISSONS J4AI DUE Réalisée mon rêve

Que je dédie à la mémoire de mes chers parents Ahmed et Yamina que dieu

les gardes dans son paradis.

A mes sœurs Souad et Salima et Tema

A mes frères Djilali et Hicham et Oussama

A mes nièces nour el iman, Anfal, Bouchera, Raimasse, Malek, Djihane, Inass,

Les 2 Amina et Meryem.

A mes neveux Ahmed et Akeram et Acheraf et Ahmed.

A mes ami(e) de l'instrumentation électronique <<2014-2015>>

Khadija 'a, khadidja_k, khadidja_s, Fatima, Dalila

A tousse qui mon aide.

AICHA BENYETTOU

Dédicace

Je ne saurais comment commencer cette dédicace sans heurter la sensibilité de certaines
personnes mais que tous ceux qui ne se reconnaîtront pas sachent que je les porte toujours
dans mon cœur.

Chaleureusement je dédie ce modeste travail :

***A** mes très chers parents en signe de ma profonde et affectueuse reconnaissance pour leur amour sans mesure, tous les sacrifices, les soutiens, les tolérances et les encouragements qu'ils ont bien voulu consentir pour moi. Tous les mots restent faibles pour leur exprimer mes sentiments et qu'ils acceptent seulement ces lignes en guise de témoignage à qui je souhaite la bonne santé et que dieu me les garde.*

***A** mes sœurs.*

***A** tous mes amis sans exception.*

***A** toute la promotion de master instrumentation électronique (2014_2015)*

***A**insi tous les étudiants de l'Université de Tlemcen.*

Khadidja

SOMMAIRE

SOMMAIRE

SOMMAIRE

SOMMAIRE

Chapitre2 : Compteur 4020

SOMMAIRE

Chapitre 3 : Fréquencemètre multi calibre

SOMMAIRE

Introduction générale

Les instruments électroniques permettent d'aider l'homme dans les tâches difficiles, répétitives ou pénibles. De plus ils constituent le rêve de substituer les instruments à l'homme dans ces tâches.

Les facultés de perception et de raisonnement des instruments électroniques progressent chaque jour sans cesse, ils sont appelés à jouer un rôle de plus en plus important dans notre vie.

Les instruments comportent de grands pôles d'intérêt.

Notre but dans ce mémoire est l'étude et la réalisation d'un fréquencemètre multi calibre programmé en utilisant le logiciel « Micro Pascal ». Notre instrument est constitué de plusieurs étages, les plus importants sont un microcontrôleur PIC16F876 A et un compteur d'impulsion 4020.

Le plan du mémoire est comme suit :

Dans le premier chapitre, nous présentons une étude approfondie sur le microcontrôleur PIC 16F876A utilisé pour commander notre instrument.

Le deuxième chapitre porte sur la description d'un compteur d'impulsion 4020 et d'une porte logique NAND 74hc00n.

Dans le troisième chapitre, nous faisons une étude théorique du montage, puis nous réalisons le montage du fréquencemètre multi calibre. Aussi, nous consacrons une grande partie aux logiciels de simulation utilisés pour la programmation. Nous présentons également les organigrammes utilisés par le microcontrôleur pour gérer notre circuit.

Enfin, nous terminons notre mémoire par une conclusion qui présente le bilan de ce travail et les perspectives envisagés.

I/1- Introduction :

Le microcontrôleur est un objet technique, intégrant de l'électronique, fait souvent apparaître des fonctions ayant pour rôle le traitement d'information : opérations arithmétiques (Addition, multiplication…) ou logiques (ET, OU…) entre plusieurs signaux d'entrée permettant de générer des signaux de sortie.

Ces fonctions peuvent être réalisées par des circuits analogiques ou logiques. Mais, lorsque l'objet technique devient complexe, et qu'il est alors nécessaire de réaliser un ensemble important de traitements d'informations, il devient plus simple de faire appel à une structure à base de microcontrôleur.

I/2- Définition de PIC :

Les microcontrôleurs sont aujourd'hui implantés dans la plupart des applications grand public ou professionnelles, il en existe plusieurs familles. La société Américaine Micro chip Technologie a mis au point dans les années 90 un microcontrôleur CMOS : le PIC (Periphirol Interface contrôler). Ce composant encore très utilisé à l'heure actuelle, est un compromis entre simplicité d'emploi, rapidité et prix.

Les PIC existent dans plusieurs versions :

- Les UVPROM qui sont effaçables par une source de rayonnement ultraviolet
- Les OTPROM programmables une seule fois
- Les EEPROM et flash EPROM qui sont effaçables électriquement.

I/3- Classification des Pics de Micro chip :

Actuellement les modèles micro chip sont classés en trois grandes familles, comportant chacune plusieurs références. Ces familles sont :

- Base –line : les instructions sont codées sur 12 bits.
- Mide –line : les instructions sont codées sur 14 bits.
- High –line : les instructions sont codées sur 16 bits.

Les Pics sont des composants statiques, Ils peuvent fonctionner avec des fréquences d'horloge allant du continu jusqu'à une fréquence maximale spécifique à chaque circuit. Dans notre application, nous avons choisi d'utiliser le PIC 16F876A qui contient un espace mémoire plus large que les autres Pics et qui est disponible dans le marché et très utilisé. [1]

I/4- Identification des pics:

Un PIC est généralement identifié par une référence de la forme suivante : xx(L) XXCFYY-ZZ [1].

xx : famille du composant, actuellement « 12, 14, 16,17 et 18 ».

L : tolérance plus importante de la plage de tension.

XX : type de programme.

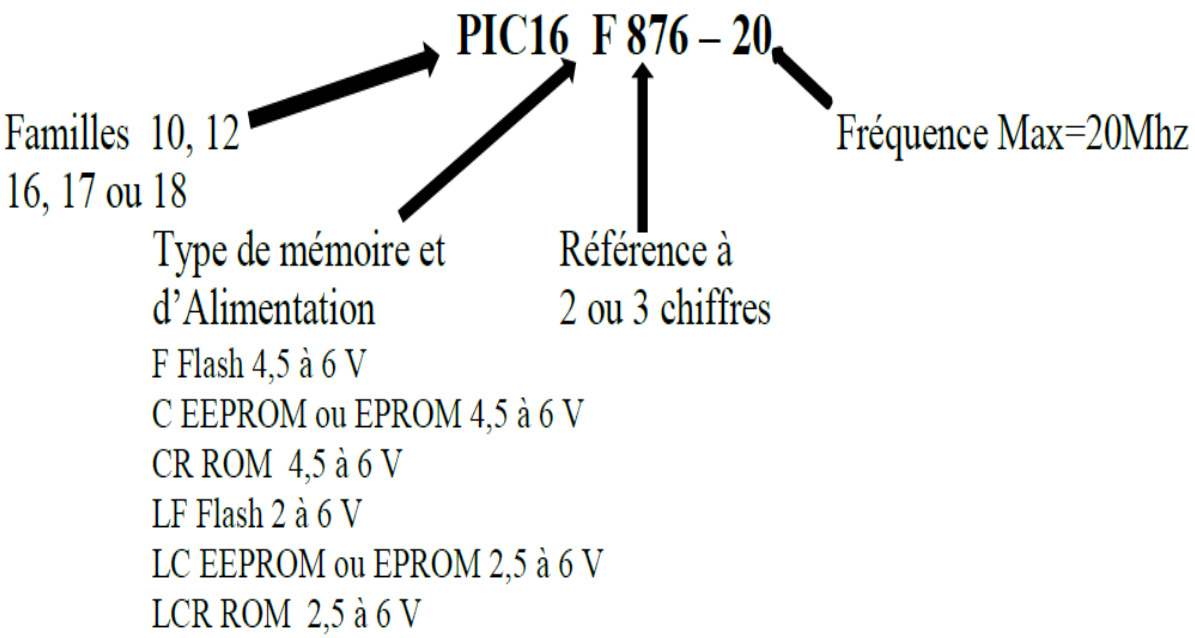
C : EPROM ou EEPROM.

F : flash.

YY : identificateur.

ZZ : vitesse maximale du quartz de pilotage.

Exemple :



I/5-Description :

- Consommation : moins de 2mA sous 5V à 4 MHz.
- Architecture RISC : 35 instructions de durée 1 ou 2 cycles.
- Durée du cycle : Période de l'oscillateur quartz divisée par 4 soit 200 ns pour un quartz de 20 MHz.
- Deux bus distincts pour le code programme et les data.

- Code instruction : mot de 14 bits et compteur programme (PC) sur 13 bits, ce qui permet d'adresser 8 K mots (de h'0000' à h'1FFF')
- Bus DATA sur 8 bits.
- 33 Ports Entrée-Sortie bidirectionnels pouvant produire 25 mA par sortie.

PORTA = 6 bits et PORTB PORTC et PORTD = 8bits PORTE = 3 bits pour le 16F877 et 22 I/O seulement pour le 16F876.

- **4 sources d'interruption :**
- Externe par la broche partagée avec le Port B : PB0
- Par changement d'état des bits du Port B: PB4 PB5 PB6 ou PB7
- Par un périphérique intégré dans le chip: écriture de Data en EEPROMterminée, conversion analogique terminée, réception USART ou I2C.
- Par débordement du Timer.

- 2 Compteurs 8 bits et 1 compteur 16 bits avec pré diviseur programmable.
- Convertisseur analogique 10 bits à 8 entrées pour le 16F877 et 4 entrées pour le 16F876.
- UART pour transmission série synchrone ou asynchrone.
- Interface I2C.

- 2 modules pour PWM avec une résolution de 10 bits.
- Interface avec un autre micro: 8 bits + 3 bits de contrôle pour R/W et CS.
- 368 Octets de RAM
- 256 Octets d'EEPROM Data.
- 8K mots de 14 bits en EEPROM Flash pour le programme (h'000' à h'1FFF').
- 1 registre de travail : W et un registre fichier : F permettant d'accéder à la RAM ou aux registres internes du PIC. Tous les deux sont des registres 8 bits.

PORTA : 6 entrées -sorties. 5 entrées du CAN. Entrée CLK du Timer 0.
PORTB : 8 entrées-sorties. 1 entrée interruption ext. Clk et Data pour prog.
PORTC : 8 entrées-sorties. Clk Timer1 et PWM1. USART. I2C.
PORTD : 8 entrées-sorties. Port interface micro-processeur (8 bits data). } N'existe pas sur le 16F876 (28broches)
PORTE : 3 entrées-sorties. 3 bits de contrôle interf micro. 3 entrées du CAN.

I/6- Organisation de l'espace programme :

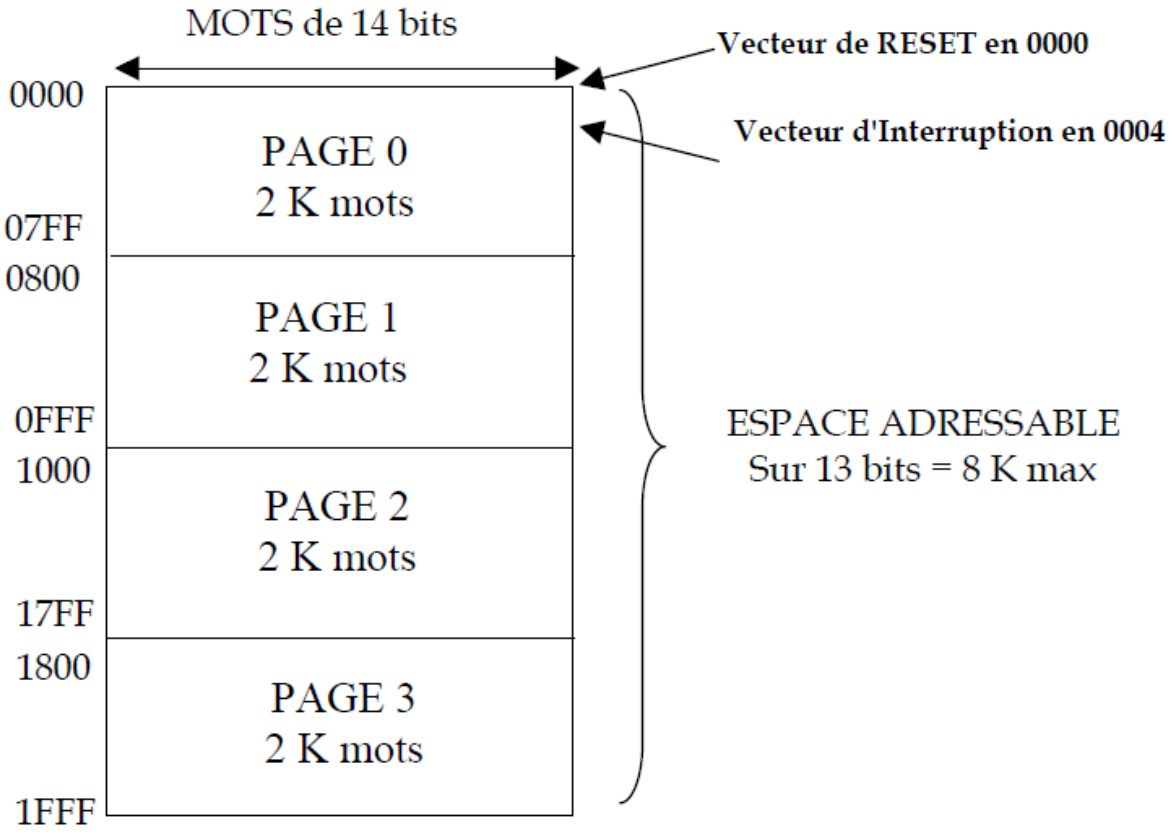


Figure I. 1- Schéma descriptif d'espace programme.

Les 2 bits MSB des 13 bits d'adresse (bits 11 et 12), viennent du registre.
PCLATH (bits 3 et 4) qui est à l'adresse : h'0A'.
Il faut impérativement les positionner pour la bonne page, avant d'utiliser les instructions:
CALL et GOTO.

I/6.1-Brochage : 16F876 (28 broches étroit) :

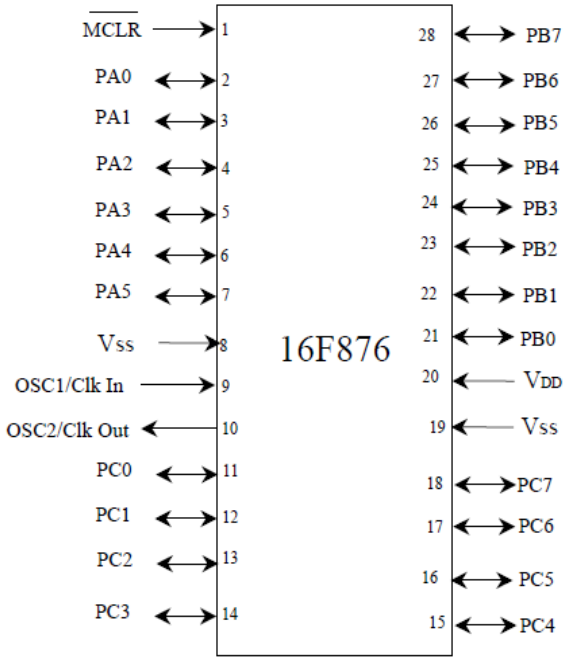


Figure I. 2- Brochage du pic 16F876.

I/6.2-Brochage : 16F877 (40 broches) :

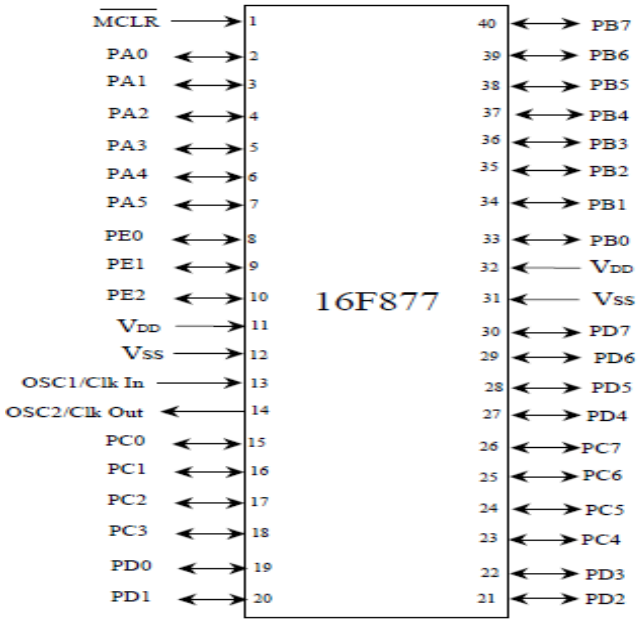
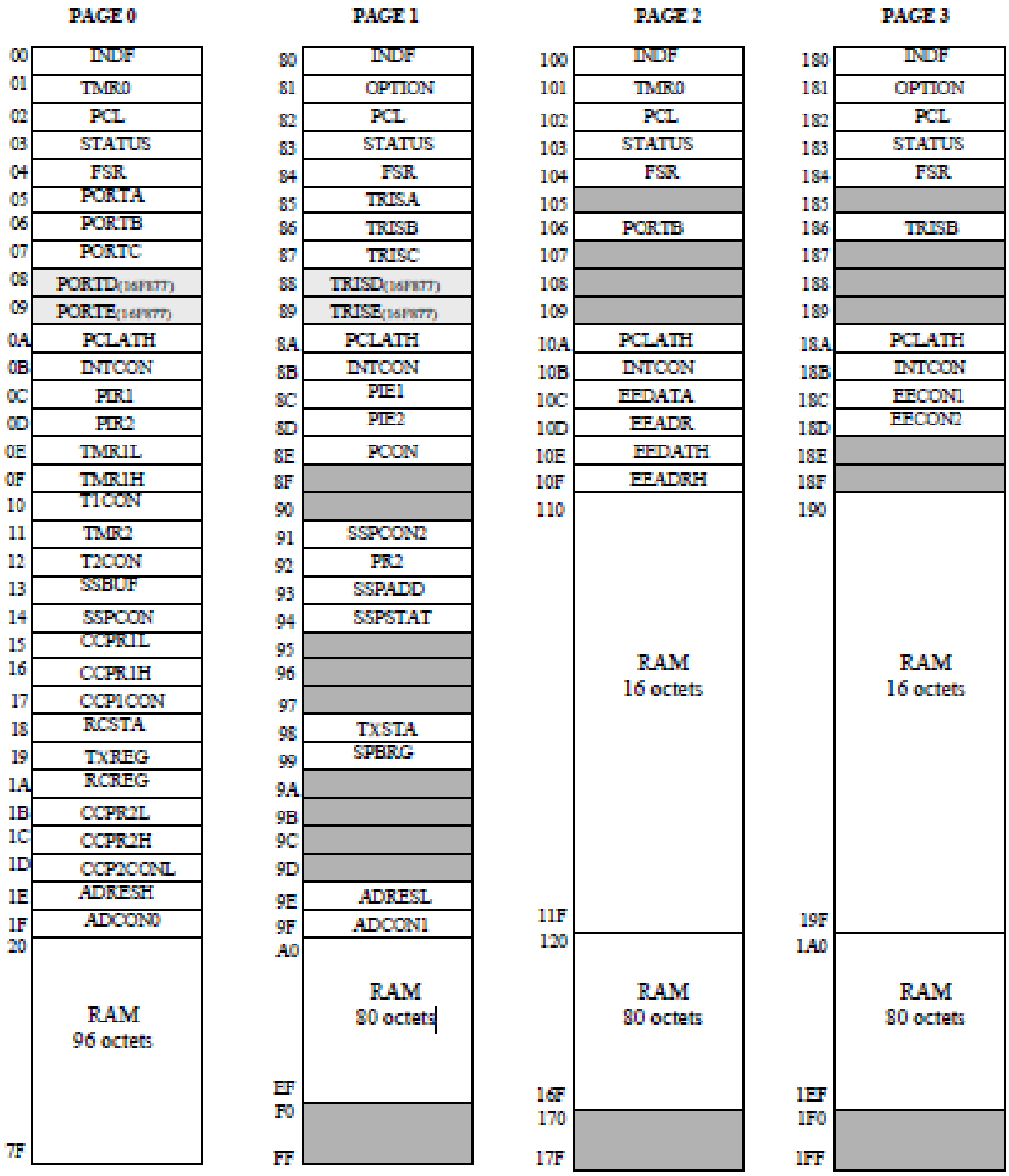


Figure I. 3-brochage pic16F877.



I/6.3-Registres internes:

I/6.3/1-Status register: (h'03' ou h'83' ou h'103 ou h'183') :

On accède indifféremment à ce registre par une quelconque de ces 4 adresses

Bit 7 Bit 0

IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C

Au reset : STATUS = 00011XXX

Bit 7 : **IRP** = permet la sélection des pages en adressage indirect.

Pour la PAGE 0 (de 00 à 7F) et la PAGE 1 (de 80 à FF) ce bit doit être laissé à "0". Mis à "1" il permettra d'atteindre la PAGE 3 (de 100 à17F) et la PAGE 4 (de 180 à 1FF).

Bits 6 et 5 : **RP1 et RP0** = permettent la sélection des pages en adressage direct.

RP1	RP0	Page sélectée
0	0	PAGE 0 de 00 à 7F
0	1	PAGE 1 de 80 à FF
1	0	PAGE 2 de 100 à 17F
1	1	PAGE 3 de 180 à 1FF

Exemple:

```
PAGE0   BCF STATUS, 5 ; RP0=0
        BCF STATUS, 6; RP1=0

PAGE1   BSF STATUS, 5; RP0=1
        BCF STATUS, 6; RP1=0

PAGE2   BCF STATUS, 5; RP0=0
        BSF STATUS, 6; RP1=1

PAGE3   BSF STATUS, 5; RP0=1
        BSF STATUS, 6 ; RP1=1
```

A dressage direct:

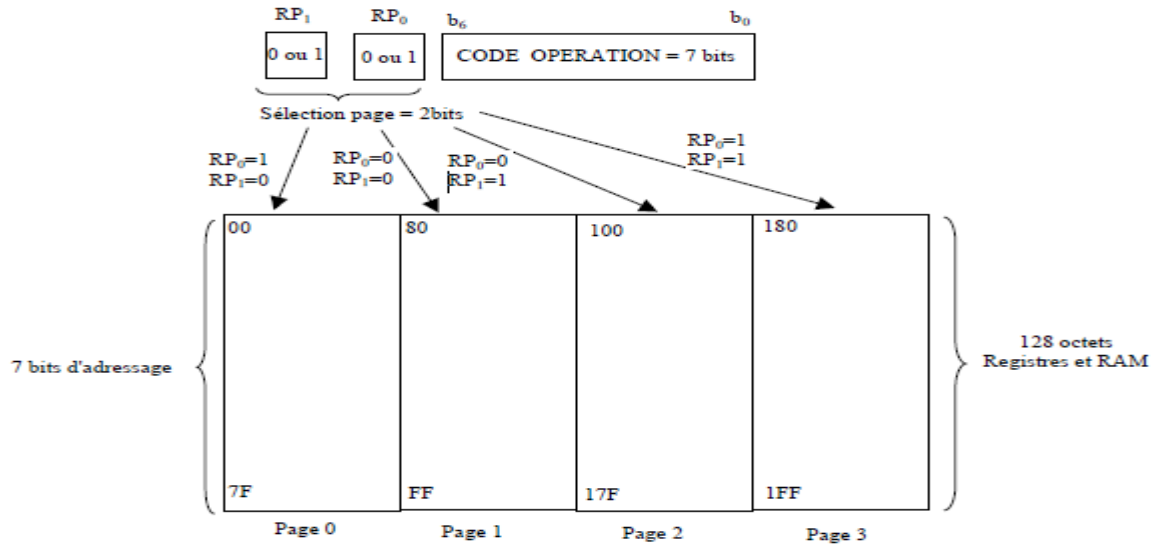


Figure I. 4- Schéma descriptif d'adressage direct.

A dressage indirect :

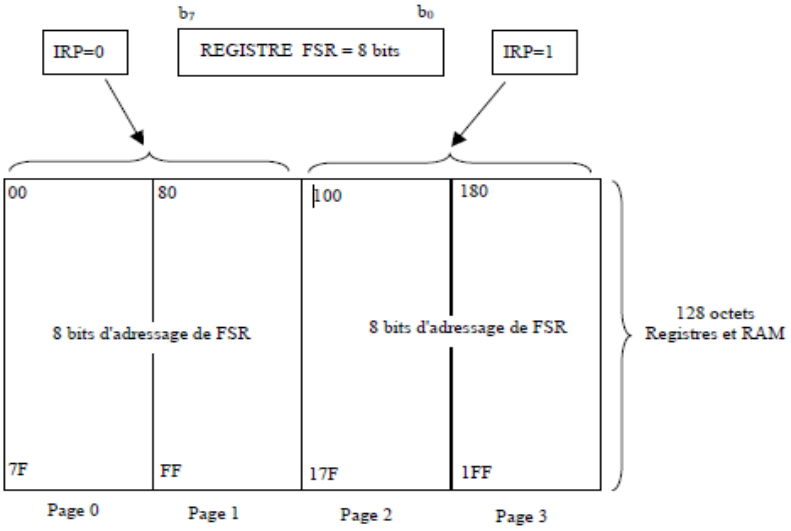


Figure I. 5 Schéma descriptif d'adressage indirect.

Bit 4: **TO** = Time Out bit.

Bit en lecture seulement.

1 = Après une mise sous tension, après une RAZ du Watch dog (CLRWDT) ou bien après l'instruction SLEEP.

0 = Signifie qu'un Time Out du timer de Watch dog est survenu.

Bit 3: **PD** = Power Down bit.

1 = Après une mise sous tension ou bien après une RAZ du Watch dog.

0 = Après l'instruction SLEEP.

Bit 2 : **Z** = Zéro bit.

1 = Le résultat d'une opération arithmétique ou logique est zéro.

0 = Le résultat d'une opération arithmétique ou logique est différent de zéro.

Bit 1: **DC** = Digit Carry bit.

1 = Une retenue sur le 4eme bit des poids faible est survenue après les instructions : ADDWF et ADDLW.

0= Carry bit = Pas de retenue sur le 4eme bit des poids faible.

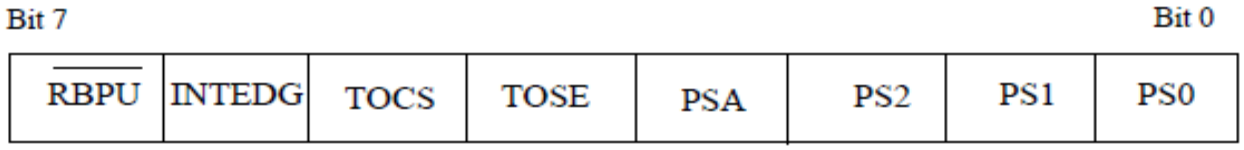
Bit 0 : **C**.

1 = Une retenue sur le bit MSB est survenue après les instructions ADDWF et ADDLW.

0 = Pas de retenue sur le bit MSB.

I/6.3/2-Option register: (h'81' ou h'181'):

Ce registre en lecture écriture permet de configurer les prés diviseurs du Timer et du Watch dog, la source du Timer, le front des interruptions et le choix du Pull up sur le Port B...



Bit 7: **RBPU** = Pull up Enable bit on Port B.

1 = Pull up désactivé sur le Port B.

0 = Pull up active.

Bit 6: **INTEDG** = Interrupt Edge select bit.

1 = Interruption si front montant sur la broche PB0/IRQ (pin 6).

0 = Interruption si front descendant Au reset : OPTION = 11111111 sur PB0/IRQ.

Bit 5: **TOCS** = Timer TMR0 Clock Source select bit.

1 = L'horloge du Timer est l'entrée PA4/Clk (pin 3).

0 = Le Timer utilise l'horloge interne du PIC.

Bit 4: **TOSE** = Timer TMR0 Source Edge select bit.

1 = Le Timer s'incrémente à chaque front montant de la broche PA4/Clk.

0 = Le Timer s'incrémente à chaque front descendant de la broche PA4/Clk.

Bit 3 : **PSA** = Prescaler Assignement bit.

1 = Le pré diviseur est affecté au Watch dog…

0 = Le pré diviseur est affecté au Timer TMR0.

Bits 2 à 0: **PS2 PS1 PS0** = Prescaler Rate Select bits.

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

Quand le pré_ diviseur est affecté au Watch dog (PSA=1), TMR0 est pré divisé par 1.

PCL REGISTER : (h'02' ou h'82' ou h'102' ou h'182').

PCLATH REGISTER: (h'0A' ou h'8A ou h'10A' ou h'18A').

Le compteur de programme est sur 13 bits. Les 8 bits de poids faible sont dans le registre PCL qui est en lecture/écriture.

Les 5 bits de poids forts ne sont pas lisibles mais on peut les écrire indirectement à travers le registre.

PCLATH.

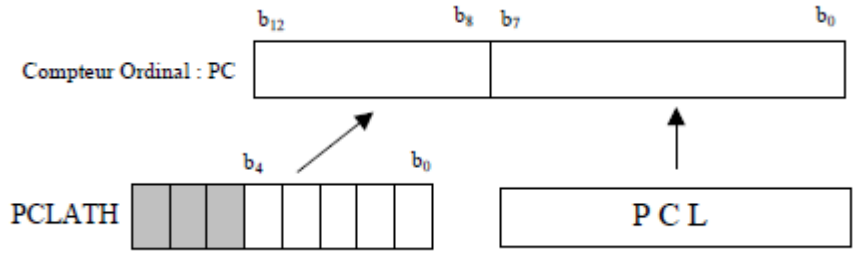


Figure I. 6-case mémoires de PCL.

I/6.3/3-Intcon register : (h'0B' ou h'8B' ou h'10B' ou h'18B') :

Ce registre en lecture écriture permet de configurer les différentes sources

Bit 7 Bit 0

GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF

Au reset : INTCON = 0000000X

Bit 7 : **GIE** = Global Interrup Enable bit

1 = Autorise toutes les interruptions non masquées.

0 = Désactive toutes les interruptions.

Bit 6 : **PEIE** = Peripheral Interrupt Enable bit.

1 = Autorise les interruptions causées par les périphériques.

0 = Désactive les interruptions causées par le périphériques.

Bit 5*:* **TOIE** = Timer TMR0 Overflow Interrup Enable bit.

1 = Autorise les interruptions du Timer TMR0.

0 = Désactive les interruptions du Timer TMR0.

Bit 4 : **INTE** = RB0/Int Interrup Enable bit.

1 = Autorise les interruptions sur la broche : PB0/IRQ (pin6).

0 = Désactive les interruptions sur la broche : PB0/IRQ (pin6).

Bit 3 : **RBIE** = RB Port Change Interrup Enable bit.

1 = Autorise les interruptions par changement d'état du Port B (PB4 à PB7).

0 = Désactive les interruptions par changement d'état du Port B (PB4 à PB7).

Bit 2*:* **TOIF** = Timer TMR0 Overflow Interrup Flag bit.

1 = Le Timer a débordé. Ce flag doit être remis à zéro par programme.

0 = Le Timer n'a pas débordé.

Bit 1*:* **INTF** = RB0/Int Interrup Flag bit.

1 = Une interruption sur la broche PB0/IRQ (pin 6) est survenue.

0 = Pas d'interruption sur la broche PB0/IRQ (pin 6).

Bit 0 : **RBIF** = RB Port Change Interrup Flag bit. Ce flag doit être remis à zéro par programme.

1 = Quand au moins une entrée du port B (de PB4 à PB7) a changé d'état.

0 = Aucune entrée de PB4 à PB7 n'a changé d'état.

I/6.3/4-PIE1 register: (h'8C': page 1):

Ce registre contient les bits individuels d'autorisation pour les Interruptions des périphériques. Le bit 6 d'INTCON (PEIE) doit être mis à "1" pour autoriser une quelconque IT de périphérique.

Bit 7 Bit 0

PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE

Au reset : PIE1 = 00000000

Bit 7 : **PSPIE** = Parallèle Slave Port Interrup Enable bit.

Ce bit n'existe pas pour un PIC 16F876 (28 pins). Toujours garder ce bit à "0".

1 = Autorise les interruptions R/W du port SSP.

0 = Désactive toutes ces interruptions.

Bit 6 : **ADIE** = A/D converter Interrup Enable bit.

1 = Autorise les interruptions du convertisseur analogique/digital.

0 = Désactive cette interruption.

Bit 5: **RCIE** = USART Receive Interrup Enable bit .

1 = Autorise les interruptions en réception de l'USART.

0 = Désactive cette interruption.

Bit 4: **TXIE** = USART Transmit Interrup Enable bit .

1 = Autorise les interruptions en émission de l'USART.

0 = Désactive cette interruption.

Bit 3 : **SSPIE** = Synchro nous Serial port Interrup Enable bit.

1 = Autorise les interruptions du module Synchrone (I2C).

0 = Désactive cette interruption.

Bit 2: **CCP1IE** = CCP1 Interrup Enable bit .

1 = Autorise les interruptions du CCP1.

0 = Désactive cette interruption.

Bit 1: **TMR2IE** = TMR2 Interrup Enable bit.

1 = Autorise les interruptions du Timer 2 TMR2.

0 = Désactive cette interruption.

Bit 0: **TMR1IE** = TMR1 over flow Interrup Enable bit.

1 = Autorise les interruptions de débordement du Timer 1 TMR1.

0 = Désactive cette interruption.

Bit 7 Bit 0

PSPIE ADIE RCIE TXIE SSPIE CCP1IE TMR2IE TMR1IE.

I/6.3/5-PIR1 register: (h'0C': page 0):

Ce registre contient les FLAG associés aux interruptions des périphériques.

Bit 7 Bit 0

PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF

Ces Flag passent à "1" quand une IT correspondante survient le bit et que d'autorisation est bien positionnée. Ces Flag doivent être remis à "0" par Soft.

Au reset : PIR1 = 00000000

Bit 7 : **PSPIE** = Parallèle Slave Port Interrup Flag bit.

1 = Une opération de R/W vient d'avoir lieu sur le port SSP.

0 = Il n'y a pas eu de R/W sur le port SSP.

Bit 6 : **ADIF** = A/D converter Interrup Flag bit.

1 = Une conversion A/D est terminée.

0 = la conversion A/D n'est pas terminée.

Bit 5: **RCIF** = USART Receive Interrup Flag bit .

1 = Le buffer de réception de l'USART est plein (donnée reçue).

0 = Le buffer de réception de l'USART est vide (rien de reçu).

Bit 4 : **TXIF** = USART Transmit Interrup Flag bit.

1 = Le buffer de transmission de l'USART est vide (on peut le remplir).

0 = Le buffer de réception de l'USART est plein (on ne peut pas le charger).

Bit 3 : **SSPIF** = Synchro nous Serial Port Interrup Flag bit.

1 = Une condition d'IT du module SSP est apparue.

0 = Aucune condition d'IT n'est apparue.

Bit 2: **CCP1IF** = CCP1 Interrup Flag bit.

1 = Une condition de Capture ou de Compare du Timer1 a fait une IT.

0 = Pas d'IT de capture ou de Compare du TIMER 1.

Bit 1: **TMR2IF** = TMR2 Interrup Flag bit.

1 = Le Timer2 a fait une IT.

0 = Pas d'IT du TIMER 2.

Bit: **TMR1IF** = TMR1 Overflow Interrup Flag bit.

1 = Le débordement Timer 1 a fait une IT.

0 = Pas de débordement du TIMER 2.

I/7- Ports entrée / sortie :

I/7.1-PORTA (h05) et TRISA (h85) :

Ce port bidirectionnel est constitué de 6 bits. Le registre de direction correspondant est TRISA.

Quand on écrit un "1" dans TRISA, le bit correspondant du PORTA est configuré en ENTREE, et le driver de sortie est placé en haute impédance.

Si on écrit un "0", le port devient une SORTIE, et le contenu du latch correspondant est chargé sur la broche sélectionnée.

Le bit 4 du Port peut également servir pour l'entrée horloge du timer TMR0.

Les autres bits du Port sont partagés avec le CAN.

N.B: **Après un reset le Port A est configuré en CAN. Il faut impérativement le configurer en I/O digitale pour l'utiliser comme tel. Il faut pour cela accéder au registre ADCON1 en h'9F'.**

Ce registre sera étudié dans le chapitre concernant le CAN.

On retiendra seulement que pour configurer les 5 bits du Port A en I/O digitales, il faut positionner les 4 bits PCFG à 0110.

Exemple :

```
 ADCON1 EQU h'9F'
PAGE1
MOVLW h'06'
MOVWF ADCON1
```

N.B: **Le Port PA4 qui est partagé avec l'entrée horloge du Timer 0 est un Drain ouvert. Il faut donc le relier au +Vcc par une résistance de 10 KΩ pour l'utiliser en tant que sortie.**

I/7.2- PortB (h06) et TrisB (h86) :

Il comporte 8 bits. Le registre de direction correspondant est TRISB.Si on écrit un "1" dans le registre TRISB, le driver de sortie correspondant passe en haute impédance. Si on écrit un "0", le contenu du Latch de sortie correspondant est recopié sur la broche de sortie.

Chaque broche du PORT B est munie d'un tirage au +VDD que l'on peut mettre ou non en service en mode entrée uniquement. On active cette fonction par la mise à "0" du bit 7 dans le registre OPTION en h'81'.
Au reset, le tirage est désactivé.
Il est inactif quand le port est configuré en sortie.
Les 4 broches PB7 PB6 PB5 et PB4 provoquent une interruption sur un changement d'état si elles sont configurées en ENTREE.
On doit remettre à zéro le Flag de cette interruption (bit 0 du registre INTCON en h'0B') dans le programme d'interruption.
Cette possibilité d'interruption sur un changement d'état associé à la fonction de tirage configurable sur ces 4 broches, permet l'interfaçage facile avec un clavier. Cela rend possible le réveil du PIC en mode SLEEP par un appui sur une touche du clavier.
Le bit 0 du PORT B peut également être utilisé comme entrée d'interruption externe. Le choix du front de déclenchement se fait en configurant le bit 6 du registre OPTION.
Au RESET: PORT A et PORT B configurés en ENTREE (TRISA et TRISB = 1) PORT B : Tirage désactivé.

I/7.3-PortC (h07) et TRISC (h87) :

Il s'agit d'un PORT 8 bits bidirectionnel.
Il est partagé avec le module de transmission synchrone I2C et l'USART.

I/7.3/1-USART ou SCI :

Elle utilise les pins 6 et 7 du PORT C: **PC6 = Tx DATA** et **PC7 = Rx DATA**
Les 5 registres utilisés sont :
Registre Emission : **TXREG** en h'19' page 0.
Registre Réception: **RCREG** en h'1A' page 0.
Registre d'état Emission : **TXSTA** en h'98' page 1.
Registre d'état Réception : **RCSTA** en h'18' page 0.
Registre du choix de la vitesse : **SPBRG** en h'99 page 1.

I/7.3/2-TXSTA register: (h'98': page 1):

Bit 7 Bit 0

CSRC	TX9	TXEN	SYNC		BRGH	TRMT	TX9D

Au reset : TXSTA = 00000010

Bit 7 : **CSRC**= Clock Source en synchrone. Sans importance en asynchrone.

Bit 6 : **TX9** = Autorisation d'émission sur 9 bits.

1 = Autorisé.

0 = Non autorisé.

Bit 5 : **TXEN** = Autorisation d'émission.

1 = Autorisé.

0 = Non autorisé.

Bit 4 : **SYNC** = Sélection mode Synchrone / Asynchrone.

1 = Mode synchrone.

0 = Mode asynchrone.

Bit 3 : **Non implémenté**

Bit 2 : **BRGH** = Sélection vitesse rapide en mode asynchrone.

1 = Vitesse haute sélectionnée.

0 = Vitesse basse sélectionnée.

Bit 1 : **TRMT** = bit d'état du registre à décalage Emission.

1 = Registre vide, donc émission terminée.

0 = Registre plein, donc émission en cours.

Bit 0 : **TX9D** = 9eme bit de Data transmise.

Ce bit peut être le bit de la parité.

I/7.3/3-SPBRG register : (h'99' : page 1).

Le Baud Rate Générateur est un registre 8 bits qui contient le facteur de division (N) de l'horloge interne qui permet d'obtenir la vitesse commune d'émission et de réception. En mode Asynchrone (bit SYNC = 0) suivant l'état du bit BRGH on aura le choix entre 2 vitesses : haute pour BRGH=1 et basse pour BRGH=0.

BRGH=0 VITESSES BASSES	BRGH=1 VITESSES HAUTES
$VITESSE = \frac{F_{oscill}}{64(N+1)}$	$VITESSE = \frac{F_{oscill}}{16(N+1)}$
$N = \frac{F_{oscill}}{64.Vitesse} - 1$	$N = \frac{F_{oscill}}{16.Vitesse} - 1$

Le nombre N est le nombre entier, arrondi de la valeur trouvée par les équations ci-dessus.

Il est recommandé d'utiliser si possible les vitesses hautes (BRGH=1), même pour des vitesses faibles, car dans ce cas on minimise l'erreur, en obtenant un nombre N plus grand.

Valeurs de N pour diverses vitesses avec un Quartz de 8 MHz :

VITESSES en Bits/sec ou BAUDS	VITESSES BASSES BRGH = 0	VITESSES HAUTES BRGH = 1
119200	x	4
57600	x	8
38400	x	12
19200	5	25
9600	12	51
4800	25	103
2400	51	207
1200	103	x

I/7.3/4-RCSTA register : (h'18' : page 0) :

Bit 7 Bit 0

SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

Au reset : RCSTA = 0000000X

Bit 7 : **SPEN**= Serial Port Enable. PC7 et PC6 configurés pour le port série.

1 = Port série en service.

0 = Port série désactivé.

Bit 6 : **RX9** = Autorisation de réception sur 9 bits.

1 = Autorisé.

0 = Non autorisé.

Bit 5: **SREN** = Single Receive Enable. Réservé pour mode Synchrone.

Non utilisé en mode Asynchrone.

Bit 4: **CREN** = Continous Receive Enable.

1 = Autorise la réception en continu.

0 = Désactive la réception en continu.

Bit 3: **ADDEN** = Adress Detect Enable. En mode Asynchrone 9 bits :

1 = Autorise la détection d'adresse, et charge la Data dans le registre de réception RCREG quand le 9eme bit du registre de dé sérialisation vaut "1".

0 = Déselecte la détection d'adresse. Tous les octets sont reçus et le 9eme bit peut servir de bit de parité.

Bit 2 : **FERR** = Framing Error.

1 = Une erreur de Framing est survenue.

0 = Pas' d'erreur de Fra ming.

Bit 1 : **OERR** = Overrun Error.

Un octet est reçu alors que le registre de réception n'a pas été vidé par lecture.

1 = Erreur Overrun.

0 = Pas d'erreur Overrun.

Bit 0 : **RX9D** = 9eme bit de Data reçue.

Ce bit peut être le bit de la parité.

EMISSION: Pin PC6 = TX DATA.

L'émission est autorisée par la mise à "1" du bit 5 de TXSTA: TXEN = 1.

La DATA à transmettre est mise dans le registre TXREG en h'19' page 0. Ce registre prévient qu'il est vide en mettant le flag TXIF à "1" (bit 4 de PIR1).
Ce flag passe à "0" dès que l'on charge un octet dans le registre TXREG. Il repasse à "1" par Hard quand le registre est vidé par transfert dans le registre de sérialisation : TSR.
Ce registre n'est pas accessible par l'utilisateur, il n'a pas d'adresse.
Si on charge alors un 2eme octet dans le registre TXREG le flag TXIF va passer à "0" et y rester tant que le registre TSR n'aura pas complètement sérialisé l'octet précédent à transmettre. Dès que le STOP de l'octet précédent a été transmis, le registre TXREG est transféré dans TSR et le flag TXIF repasse à"1" signalant ainsi que le registre de transmission TXREG est vide et peut donc recevoir un nouvel octet à transmettre.
Le bit TRMT (bit 1 de TXSTA) informe sur l'état du registre TSR. Quand le registre TSR n'a pas fini de sérialisé, TRMT=0. Ce flag repasse à "1" quand le registre est vide, c'est à dire quand le stop a été émis.
Le flag TXIF permet aussi de générer une interruption, à condition qu'elle soit autorisée par mise à "1" du bit 4 de PIE1 : TXIE = 1.
Il faut dans ce cas autoriser les interruptions des périphériques par mise à "1" du bit 6 de INTCON: PEIE = 1, et par la mise à "1" du bit 7 : GIE = 1.

I/8-Procedure pour émettre :

- Initialiser SPBRG pour la vitesse désirée et choix pour BRGH.
- Autoriser mode Asynchrone : SYN = 0 et SPEN = 1.
- Eventuellement faire TX9 = 1 si une émission sur 9 bits est désirée.
- Autoriser l'émission par TXEN = 1.
- Si une transmission 9 bits a été choisie, mettre le 9eme bit dans TX9D.
- Mettre l'octet à transmettre dans TXREG.
- Avant de remettre l'octet suivant à transmettre dans TXREG, il faut tester le flag TXIF qui est à "0" si le registre n'est pas disponible. Dès que le registre est vide ce flag passe à "1" et on peut alors charger TXREG par l'octet à transmettre.
- Pour savoir si le dernier octet a été émis, il suffit de tester le flag TRMT qui signale par son passage à "1" que le dernier bit du dernier octet et son STOP ont bien été sérialisés.
- On peut alors stopper le module émission de l'USART par TXEN=0.

I/9-Reception Pin PC7 = RX DATA:

La réception est autorisée par la mise à "1" du bit 4 de RCSTA: CREN = 1.
La DATA reçue est mise dans le registre RCREG en h'1A' page 0. Ce registre prévient qu'il est plein en mettant le flag RCIF à "1" (bit 5 de PIR1). On peut autoriser la génération d'une interruption quand RCIF = 1, c'est à dire quand une donnée valide est disponible dans RCREG par mise à "1" du bit 5de PIE1 : RCIE = 1. Le flag RCIF repasse à "0" par hard quand on vide le registre RCREG par sa lecture.
Si le STOP d'un 2eme octet survient alors que le registre RCREG n'a pas été vidé, une erreur OVERRUN se produit. Elle est signalée par le passage à "1"du bit 1 de RCSTA : OERR=1. L'octet dans le registre de dé sérialisation est alors perdu.
Le bit d'erreur OERR doit être remis à zéro par soft. Pour cela il faut stopper la réception par CREN=0 puis remettre en service la réception par CREN=1.
En fait le registre RCREG est un double registre FIFO. On peut donc recevoir2 octets et ne pas les lire avant qu'un 3eme octet ne fasse un OVERRUN.
On doit alors lire deux fois RCREG pour le vider les 2 octets reçus qui sont dans le FIFO.
Si un STOP est trouvé à "0" alors une ERROR FRAMING est générée par mis à "1" du bit 2 de RCSTA: FERR=1.

I/10-Procedure pour recevoir :

- Initialiser SPBRG pour la vitesse désirée et choix pour BRGH.
- Autoriser mode Asynchrone : SYN = 0 et SPEN = 1.
- Eventuellement faire RX9 = 1 si une réception sur 9 bits est désirée.
- Eventuellement faire RCIE = 1 si une réception par interruption est désirée.
- Autoriser l'émission par RCEN = 1.
- Test du Flag RCIF (ou attente IT) pour savoir si un octet a été reçu.
- Lire éventuellement le 9eme bit de Data dans RCSTA pour tester la parité.
- Lire les bits FERR et OERR pour déterminer les erreurs éventuelles.
- Si une erreur est survenue il faut faire CREN=0 puis CREN=1 pour RAZ.
- Lecture du registre RCREG pour récupérer l'octet reçu.

I/10.1-Mode 9 bits avec détection d'adresse :

- Faire toutes les initialisations précédentes: SPBRG, BRGH, SYN, SPEN.
- Faire RX9=1 ADDEN=1 et CREN=1

- Dès que le registre RCREG est signalé plein, il s'agit de l'adresse (9 eme bit =1).
- Le lire et le comparer avec l'adresse déterminée.

Si OK faire ADDEN=0 pour recevoir maintenant tous les octets, même si le 9eme bit n'est pas à "1".

Si pas OK rester avec ADDEN=1 pour ne recevoir que les octets filtrés d'adresse.

I/10.2-Module MSSP pour I2C:

Le module MSSP (Master Synchro nous Serial Port) permet de faire des transmissions au format I2C (Inter Integrated Circuits) en tant que Maître ou Esclave.

Il utilise les pins 3 et 4 du PORT C: **PC3 = SCL et PC4 = SDA.**

Les 5 registres utilisés sont :

Registre Buffer **SSPBUF** en h'13' page 0.

Registre d'adresse **SSPADD** en h'93' page 1.

Registre d'état **SSPSTAT** en h'94' page 1.

Registre de contrôle **SSPCON** en h'14' page 0.

Registre de contrôle n°2 **SSPCON2** en h'91' page 1.

I/10.3-SSPSTAT register : (h'94' : page 1) :

Bit 7 Bit 0

SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF

Au reset : SSPSTAT = 00000000

Bit 7 : **SMP**= bit d'échantillonnage.

Bit 6: **CKE** = Clock Edge Select. Ce bit à "0" : conforme aux spécifications I2C.

Bit 5: **D/A** = Data / Addressee bit.

1 = indique que le dernier octet reçu est une Data.

0 = indique que le dernier octet reçu est une adresse.

Bit 4 : **P** = STOP bit. Ce bit est remis à "0" quand le module SSP est désactivé.

1 = indique qu'un Stop a été détecté.

0 = indique qu'il n'a pas été détecté de Stop.

Bit 4 : **P** = STOP bit. Ce bit est remis à "0" quand le module SSP est désactivé.

1 = indique qu'un Stop a été détecté.

0 = indique qu'il n'a pas été détecté de Stop.

Bit 3 : **S** = START bit. Ce bit est remis à "0" quand le module SSP est désactivé.

1 = indique qu'un Start a été détecté.

0 = indique qu'il n'a pas été détecté de Start.

Bit 2 : **R/W** = Information sur le bit R/W. Donne la valeur de ce bit qui suit la dernière adresse. Ce bit est valable jusqu'à la réception du STOP.

- ***Mode esclave :***

1 = R.

0 = W.

- ***Mode maitre*** *:*

1 = Transmission en cours.

0 = Pas de transmission en cours.

Bit 1 : **UA** = mise à jour adresse. Utilisé en mode 10 bits seulement.

Bit 0 : **BF** = Buffer plein.

- ***Mode Réception I2C:***

1 = Registre plein : réception terminée.

0 = Registre vide : donc réception non terminée.

- **Mode Emission I2C:**

1 = Registre plein : donc émission en cours.

0 = Registre vide : donc émission terminée.

10.4-SSPCON register : (h'14' : page 0) :

Bit 7 Bit 0

WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0

Au reset : SSPCON = 00000000

Bit 7 : **WCOL**= détection de collision à l'écriture.

1 = Collision à l'écriture.

0 = Pas de collision.

Bit 6 : **SSPOV** = Over flow indicator en réception. Ne sert pas en émission Doit être remis à "0" par soft

1 = Un octet vient d'être reçu dans le buffer qui n'était pas vidé : Over flow.

0 = Pas d'Over flow.

Bit 5: **SSPEN** = Enable Module SSP.

1 = Module I2C activé et pin SCL et SDA sur PC3 et PC4 configurées.

0 = Module I2C désactivé. PC3 et PC4 libres pour usage I/O.

Bit 4 : **CKP** = Polarité du Clock. Inutilisé en mode maître.

En mode Esclave :

1 = Autorise horloge.

0 = Maintient CLK = 0.

Bit 3 à bit 0 : **SSPM3 / SSPM0** = Select Mode.

SSPM3	SSPM2	SSPM1	SSPM0	MODE
0	1	1	0	I²C esclave 7 bits.
0	1	1	1	I²C esclave 10 bits.
1	0	0	0	I²C maître CLK= F_{osc}/4(SSPADD+1).

I/10.5-SSPCON2 register : (h'91' : page 1) :

Bit 7 Bit 0

GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN

Au reset : SSPCON2 = 00000000

Bit 7 : **GCEN**= Appel général en mode I2C esclave seulement..

1 = Autorise IT quand une adresse appel général (h'0000) est reçue.

0 = Adresse appel général désactivée.

Bit 6 : **ACKSTAT** = Bit de ACK en mode I2C maître seulement.

1 = Pas de ACK reçu de l'esclave.

0 = ACK de l'esclave a été reçu.

Bit 5: **ACKDT** = ACK Data bit. En mode maître et réception : valeur qui sera transmise quand on va lancer une séquence de ACK en fin de réception.

1 = NO ACK.

0 = ACK.

Bit 4: **ACKEN** = ACK Sequence Enable. En mode maître et réception, lance une séquence d'ACK ou de NOACK suivant la valeur dans ACKDT.

1 = Lance la séquence de ACK et transmet ACKDT. Remis à "0" par hard.

0 = Pas de séquence d'ACK.

Bit 3: **RCEN** = Receive Enable Bit. En mode I2C maître seulement.

1 = Autorise mode réception.

0 = Pas de réception autorisée.

Bit 2: **PEN** = Stop Condition Enable. En mode I2C maître seulement.

1 = Lance la séquence de STOP sur SDA et SCL. Remis à "0" par hard.

0 = Pas de séquence de STOP.

Bit 1: **RSEN** = Repeated Start Condition. En mode I2C maître seulement.

1 = Séquence de répétition de START sur SDA et SCL. Remis à "0" par hard.

0 = Pas de séquence de répétition de START.

Bit 0: **SEN** = Start Condition Enable. En mode I2C maître seulement.

1 = Lance la séquence de START sur SDA et SCL. Remis à "0" par hard.

0 = Pas de séquence de START.

I/10.6-Mode maitre :

Les pins SDA et SCL sont manipulées par le Hard. Les événements qui causent le passage à "1" du flag SSPIF (bit 3 de PIR1 en h'0C') et éventuellement une IT si elles sont autorisées, sont :

- Condition de START.
- Condition de STOP.
- Répétition d'un START.
- ACK après un transfert.
- Transfert d'octet en Emission ou réception.

N.B: Le flag SSPIF doit être remis à "0" par soft.

I/10.6/1-Lancement d'un START:

- L'utilisateur doit mettre le bit SEN (de SSPCON2) à "1".
- Les pins SDA et SCL étant toutes les deux à "1", le module fait passer la pin SDA de "1" à "0" ce qui génère une condition de START.
- Le bit S (de SSPSTAT) passe à "1" pour signaler le START.

- A la fin du START, le bit SEN est remis à "0" par le hard.
- Dès que le flag SSPIF (de PIR1) passe à "1" pour signaler la fin du START, on peut charger le registre de transmission SSBUF avec l'octet à transmettre.

Ne pas oublier de remettre ce flag à "0" par le soft.

I/10.6/2-Transmission :

Dès que le registre SSBUF est chargé, le bit BF (de SSPSTAT) passe à "1"pour signaler que la transmission est en cours.

Au 8eme coup de CLK, la transmission est terminée et le bit BF repasse à "0".

I/10.6/3-Acquittement :

Quand l'esclave répond l'ACK, le bit ACKSTAT passe à "0".

La fin de l'ACK est signalée par le Flag SSPIF qui passe à "1". On doit alors remettre ce flag à "0" par soft.

I/10.6/4-Lancement d'un STOP :

L'utilisateur doit mettre le bit PEN (de SSPCON2) à "1". Le module fait alors passer SDA à "0" puis force SCL à"1". Quand SCL est à "1" il fait passer SDA de "0" à"1", ce qui génère une condition de STOP.

Le bit P (de SSPSTAT) passe à "1" pour signaler le STOP.

A la fin du STOP, le bit PEN est remis à "0" par le hard.

Le flag SSPIF (de PIR1) passe à "1" pour signaler la fin du STOP.

I/10.6/5-Réponse du maître à l'esclave :

Il s'agit soit d'un ACK soit d'un NOACK :

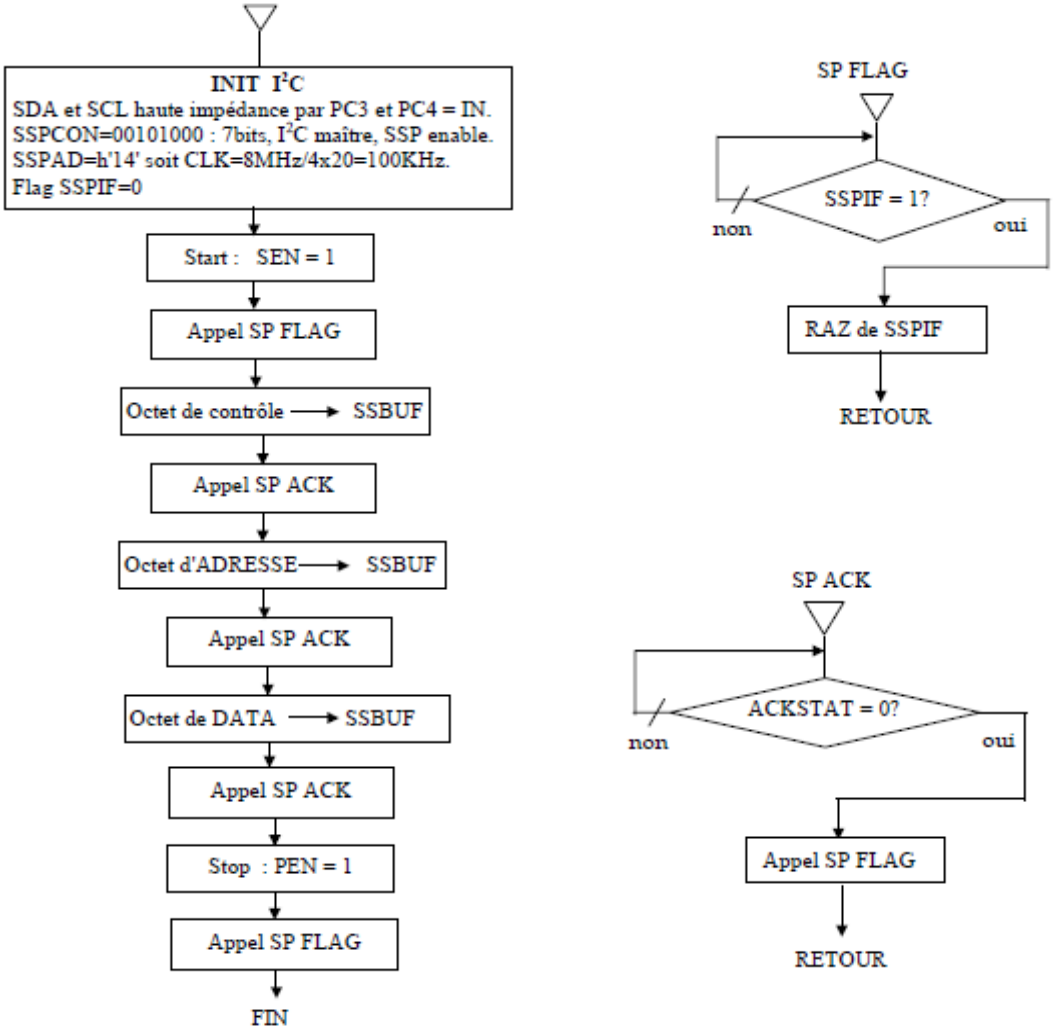
- Positionner le bit ACKDT (de SSPCON2) en fonction de la réponse à faire.
- Mettre le bit ACKEN (de SSPCON2) à "1".
- La fin de transmission de l'ACK ou du non ACK est signalée par le flag SSPIF qui passe à "1". Ne pas oublier de le remettre à "0" par soft.

I/10.6/6-Réception d'un octet par le maître :

Pour cela il faut mettre le module en réception en portant le bit RCEN à "1".

- La fin de réception est signalée par le passage à "1" du flag SSPIF, que l'on remettra à "0" par soft.
- On peut alors lire l'octet reçu dans SSBUF.

Exemple de Programme : Ecriture dans une EEPROM I2C type 24C16. [7]



I/11-Convertisseur A/D :

Il est constitué d'un module convertisseur à 5 entrées pour le boîtier 28broches (16F876) et à 8 entrées pour le boîtier 40 broches (16F877).

Les 5 premières entrées sont sur le Port A en PA0, PA1, PA2, PA3 et PA5.

Les 3 entrées supplémentaires du boîtier 40 pins sont en PE0, PE1 et PE2.

Le résultat de la conversion est codé sur 10 bits. C'est une valeur comprise entre h'000' et h'3FF'.

Les tensions de référence haute et basse peuvent être choisies par programmation parmi: VDD ou la broche PA3 pour VREF+ et VSS ou la brochePA2 pour VREF- .

Les 4 registres utilisés par le module convertisseur A/D sont :

- ADRESH en h'1E' page 0 : MSB des 10 bits du résultat.
- ADRESL en h'9E' page 1 : LSB des 10 bits du résultat.
- ADCON0 en h'1F' page 0 : registre de contrôle n°0 du convertisseur.
- ADCON1 en h'9F' page 1 : registre de contrôle n°1 du convertisseur.

I/11.1-ADCON0 : (h'1F' : page 0) :

Bit 7 Bit 0

ADSC1	ADSC0	CHS2	CHS1	CHS0	GO/$\overline{\text{Done}}$		ADON

Au reset : ADCON0 = 00000000

Bit 7 et bit 6 : **ADSC1 et ADSC0** = Clock Select bits.

Ces 2 bits permettent de choisir la vitesse de conversion :

00= Fosc/2.

01= Fosc/8.

10= Fosc/32.

11= Oscillateur RC interne.

Le temps de conversion d'un bit est TAD. Pour une conversion totale des 10bits il faut : 12.TAD.

Pour que la conversion soit correcte il faut que TAD soit au minimum de **1,6µs.**

Avec l'oscillateur interne RC on a : TAD = 4 µs typique (entre 2 et 6 µs).

Bit 5 bit4 et bit 3 : **CHS2 CHS1 et CHS0** = Channel Select bits.

Ces 3 bits permettent de choisir l'entrée qui va être convertie.

Canal	CHS2	CHS1	CHS0	PORT
0	0	0	0	PA_0
1	0	0	1	PA_1
2	0	1	0	PA_2
3	0	1	1	PA_3
4	1	0	0	PA_5
5	1	0	1	PE_0
6	1	1	0	PE_1
7	1	1	1	PE_2

Non disponible sur le 16F876 (28 pins).

Bit 2: **GO/DONE:** Status bit is ADON=1.

1 = Démarre la conversion A/D. Ce bit est remis à "0" par hard.

0 = La conversion A/D est terminée.

Bit 1 : **Bit non implanté.**

Bit 0 : **ADON :** A/D on bit.

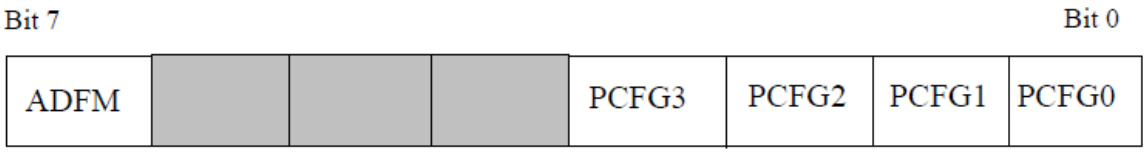
1= Convertisseur A/D en service.

0 = Convertisseur A/D à l'arrêt.

11.2-Temps de conversion TAD en fonction du Quartz et des bits du Clock select :

QUARTZ	CLOCK	T_{AD}	$12.T_{AD}$	Ne convient pas si T_{AD}<1,6µs
4 MHz	$F_{osc}/2$ = 2 MHz	0,5 µs	6 µs	Ne convient pas
	$F_{osc}/8$ = 500 KHz	2 µs	24 µs	OK
	$F_{osc}/32$ = 125 KHz	8 µs	96 µs	OK
8 MHz	$F_{osc}/2$ = 4 MHz	0,25 µs	3 µs	Ne convient pas
	$F_{osc}/8$ = 1 MHz	1 µs	12 µs	Ne convient pas
	$F_{osc}/32$ = 250 KHz	4 µs	48 µs	OK
12 MHz	$F_{osc}/2$ = 6 MHz	0,16µs	1,92 µs	Ne convient pas
	$F_{osc}/8$ = 1,5 MHz	0,66 µs	8 µs	Ne convient pas
	$F_{osc}/32$ = 375 KHz	2,6 µs	32 µs	OK
16 MHz	$F_{osc}/2$ = 8 MHz	0,125µs	1,5 µs	Ne convient pas
	$F_{osc}/8$ = 2 MHz	0,5 µs	6 µs	Ne convient pas
	$F_{osc}/32$ = 500 KHz	2 µs	24 µs	OK

11.3-ADCON1 : (h'9F' : page 1) :



Au reset : ADCON1 = 00000000

Bit 7 : **ADFM** = A/D Result format.

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

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

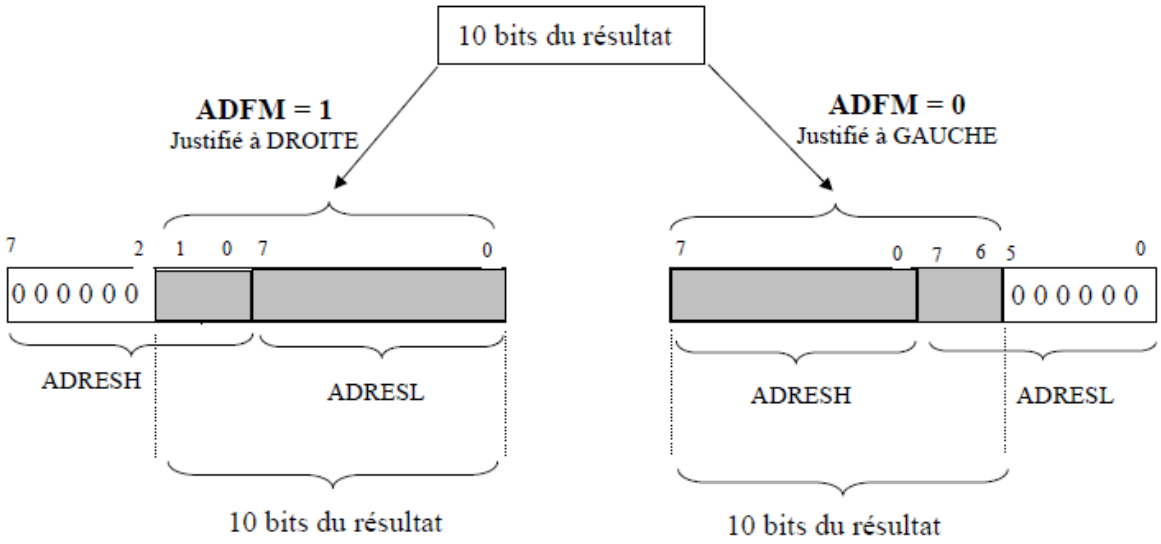


Figure I. 7-Schema descriptive ADCON1

Req : Bit 6 bit 5 et bit 4 : **Bits non implémentés.**

Bit 3 bit 2 bit 1 et bit 0 : **PCFG3 PCFG2 PCFG1 et PCFG0**

Bits de contrôle de la configuration des Ports.

Ces bits permettent de choisir le partage entre entrées analogiques et digitales sur les PORTS A et E.

Ils permettent également de choisir pour VREF+ entre VDD et PA3 et pour et VREF - entre VSS et PA2.

I/11.4-Configuration des PORTS en fonctions des 4 bits PCFG:

A = Entrée Analogique.

D = I/O Digitale.

4 Bits PCFG	N°7 PE_2	N°6 PE_1	N°5 PE_0	N°4 PA_5	N°3 PA_3	N°2 PA_2	N°1 PA_1	N°0 PA_0	V_{REF^+}	V_{REF^-}
0 0 0 0	A	A	A	A	A	A	A	A	V_{DD}	V_{SS}
0 0 0 1	A	A	A	A	V_{REF^+}	A	A	A	PA_3	V_{SS}
0 0 1 0	D	D	D	A	A	A	A	A	V_{DD}	V_{SS}
0 0 1 1	D	D	D	A	V_{REF^+}	A	A	A	PA_3	V_{SS}
0 1 0 0	D	D	D	D	A	D	A	A	V_{DD}	V_{SS}
0 1 0 1	D	D	D	D	V_{REF^+}	D	A	A	PA_3	V_{SS}
0 1 1 x	D	D	D	D	D	D	D	D	V_{DD}	V_{SS}
1 0 0 0	A	A	A	A	V_{REF^+}	V_{REF^-}	A	A	PA_3	PA_2
1 0 0 1	D	D	A	A	A	A	A	A	V_{DD}	V_{SS}
1 0 1 0	D	D	A	A	V_{REF^+}	A	A	A	PA_3	V_{SS}
1 0 1 1	D	D	A	A	V_{REF^+}	V_{REF^-}	A	A	PA_3	PA_2
1 1 0 0	D	D	D	A	V_{REF^+}	V_{REF^-}	A	A	PA_3	PA_2
1 1 0 1	D	D	D	D	V_{REF^+}	V_{REF^-}	A	A	PA_3	PA_2
1 1 1 0	D	D	D	D	D	D	D	A	V_{DD}	V_{SS}
1 1 1 1	D	D	D	D	V_{REF^+}	V_{REF^-}	D	A	PA_3	PA_2

Ces entrées n'existent pas sur le 16F876 (boîtier 28 pins).

Remarque :

Au reset le registre ADCON1 est initialisé à h'00'. Cela signifie que les 5 bits du Port A et les 3 bits du Port E sont configurés en entrées analogiques.

Pour récupérer le 5 bits du Port A et les 3 bits de Port E en tant que I/O digitales il faut écrire la valeur h'06' dans ADCON1.

I/12- Mémoire EEPROM :

Il y a 2 zones mémoire EEPROM accessible par l'utilisateur. La zone DATA et la mémoire Flash de Programme.

- La zone DATA est organisée en mot de 8 bits. Les 256 octets se programment et se lisent comme avec un PIC 16F84. Elle est positionnée dans le chip entre les adresses h'2100' et h'21FF'.

On l'adressera (de h'00' à h'FF') par le registre spécifique: EEADR en h'10D'(page 2). Les données transiteront par le registre: EEDATA en h'10C' (page 2).

- La zone EEPROM de programme qui est organisée en mots de 14 bits, est également accessible par l'utilisateur. Pour adresser les 8 K mots il faut 13bits d'adresse. Un registre

supplémentaire est disponible: EEADRH en h'10F'(page 2). Les 8 bits LSB de l'adresse seront passés par EEADR et les 5 bits MSB par EEADRH.

Les données sur 14 bits seront échangées à travers le registre EEDATA pour les 8 bits LSB et grâce à un registre supplémentaire : EEDATH en h'10E'(page 2) pour les 6 bits MSB.

I/12.1-Registres utilisés par l'EEPROM sont :

EEDATA : en h'10C' (page 2).

EEADR : en h'10D' (page 2).

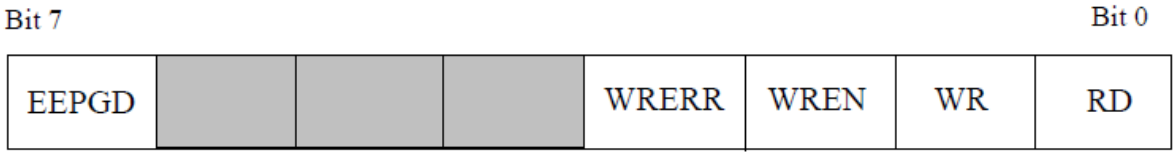
EEDATH : en h'10E' (page 2).

EEADRH : en h'10F' (page 2).

EECON1 : en h'18C' (page 3).

EECON2 : en h'18D' (page 3).

EECON1 : (h'18C' : page 3).



Au reset : EECON1 = X000X000

Bit 7 : **EEPGD** =Program/Data EEPROM select bit : choix de la zone EEPROM.

1= Accès à la zone EEPROM de programme.

0= Accès à la zone EEPROM de DATA.

Ce bit ne doit pas être changé pendant une opération de lecture ou d'écriture.

Bit 3: **WRERR** =EEPROM error flag.

1= Opération d'écriture prématurément terminée (causée par reset).

0= Opération d'écriture achevée normalement.

Bit 2 : **WREN** =EEPROM write enable bit.

1= Autorise cycle d'écriture en EEPROM.

0= Interdit l'écriture en EEPROM.

Bit 1 : **WR** = Write Control bit.

1= Lance le cycle écriture de l'EEPROM. Ce bit est remis à "0" par hard.

0= Le cycle d'écriture est terminé.

Bit 0: **RD** = Read Control bit.

1= Lance un cycle de lecture de l'EEPROM. Ce bit est remis à "0" par hard.

0= Pas de cycle lecture.

EECON2 : (h'18D' : page 3).

Ce registre n'est pas un registre de configuration physique. Il n'est utilisé que pendant les séquences d'écriture en EEPROM.

Remarque: **Pour pouvoir écrire en EEPROM programme, il faut configurer correctement certains bits du REGISTRE de CONFIG en h'2007'.**
Le bit WRT (Flash Program Memory enable) doit être forcé à "1". Les bits CP0 et CP1 doivent être tous les deux à "1" (Code Prote ct OFF), ou bien ne protéger qu'une zone particulière dans laquelle on ne voudra pas écrire.
Les procédures de lecture et d'écriture en zone DATA sont identiques à celles décrites pour le PIC 16F84.
Pour la zone flash du programme, elles sont particulières pour les PIC 16F876et 16F877.

I/12.2-Lecture en zone EEPROM programme :

1 - Mettre le MSB de l'adresse dans EEADRH.
2 - Mettre le LSB de l'adresse dans EEADR
3 - Mettre le bit EEPGD à "1" pour pointer la zone programme.
4 - Mettre le bit RD à "1" pour lancer le cycle de lecture.
5 - Attendre 2 cycles par deux NOP.
6 - Lire le MSB dans EEDATH et le LSB dans EEDATA.

I/12.3-Ecriture en zone eeprom programme :

1 - Mettre le MSB de l'adresse dans EEADRH.
2 - Mettre le LSB de l'adresse dans EEADR.
3 - Mettre le MSB de la DATA dans EEDATH.
4 - Mettre le LSB de la DATA dans EEDATA.
5 - Mettre le bit EEPGD à "1" pour pointer la zone programme.
6 - Mettre le bit WREN à "1" pour autoriser l'écriture.
7 - Inhiber les interruptions par mise à "0" du bit GIE de INTCON.
8 - Lancer la séquence spécifique suivante :
- ✓ Ecrire h'55' dans EECON2.
- ✓ Ecrire h'AA' dans EECON2.
- ✓ Lancer le cycle d'écriture par WR=1.

- ✓ On attend 2 cycles par deux NOP.
- ✓ Le PIC ignore maintenant toutes les instructions.
- ✓ A la fin de l'écriture, le PIC exécute l'instruction suivante.

9 - On autorise les interruptions par GIE=1.

10 - On interdit l'écriture en EEPROM par WREN = 0.

Exemple de programme :

```
;*******************************************************************************
;***  Ecriture en h'03FF' de la Data h'3439' (2 caractères ASCII :"4" et "9")  ***
;*******************************************************************************

            PAGE2           ;accès page 2
            MOVLW h'03'
            MOVWF EEADRH    ;MSB de l'adresse=03
            MOVLW h'FF'
            MOVWF EEADR     ;LSB de l'adresse=FF
            MOVLW h'34'
            MOVWF EEDATH    ;MSB de la DATA=34
            MOVLW h'39'
            MOVWF EEDATA    ;LSB de la DATA=39
            PAGE3           ;accès page 3
            BSF   EECON1,7  ;bit EEPGD=1 : accès mémoire programme
            BSF   EECON1,2  ;bit WREN=1 : autorise écriture en EEPROM
            BCF   INTCON,7  ;bit GIE=0 : masque les IT
Séquence    MOVLW h'55'
spécifique  MOVWF EECON2
            MOVLW h'AA'
            MOVWF EECON2
            BSF   EECON1,1  ;bit W=1 : démarre cycle écriture
            NOP             ;2 cycles d'attente
            NOP             ;Le micro attend la fin de l'écriture
            BSF   INTCON,7  ;bit GIE=1 : autorise les IT
            BCF   EECON1,2  ;bit WREN=0 : interdit écriture en EEPROM
```

I/13- Timers:

I/13.1-Module timer 0:

Ce module est le même que dans le PIC 16F84.

Le compteur/Timer TMR0 a les caractéristiques suivantes :

- Compteur sur 8 bits.
- Lecture / écriture de TMR0.
- Pré diviseur 8 bits programmable.
- Choix de l'horloge : interne en Timer et externe en compteur.
- Interruption au débordement (passage de FF à 00).
- Choix du front de l'horloge en mode horloge externe.

Tous les bits de configuration de TMR0 sont dans le registre OPTION en h'81'page 1 ou en h'181' page 3.

Le registre TMR0 est à l'adresse h'01' page 0 ou en h'101' page 2.

I/13.2-Mode Timer :

Le choix de ce mode se fait par : TOCS = 0 (b5 de OPTION).

TMR0 est incrémenté à chaque cycle instruction (Fosc/4), en considérant le pré diviseur avec un rapport de 1.

I/13.2/1-Mode compteur :

Ce mode est sélectionné si TOCS = 1. TMR0 est alors incrémenté à chaque front montant ou descendant sur la broche PA4/CLK

(pin3). Le choix du front est fait par le bit TOSE (b4 d'OPTION).

Si TOSE = 0 le compteur s'incrémente à chaque front montant.

Si TOSE = 1 c'est le front descendant qui incrémente le compteur.

I/13.2/2-Le prediviseur :

Il est partagé entre le Watch dog et TMR0.

L'affectation se fait par le bit PSA (b3 d'OPTION).

Si PSA = 0 le pré diviseur est affecté à TMR0. Le choix du rapport de division se fait avec les bits PS2, PS1 et PS0 (b2, b1 et b0 d'OPTION).

Si PSA = 1 le pré diviseur est affecté au Watch dog et le rapport de division pour TMR0 est fixé à 1.

I/13.2/3-Interruption :

Elle est générée quand TMR0 passe de la valeur FF à 00.

Le Flag TOIF (b2 d'INTCON) passe à "1". On peut masquer la génération de l'interruption en mettant le bit TOIE (b5 de INTCON) à "0".

Le Flag TOIF dit être remis à zéro par soft dans le sous-programme d'interruption, avant de ré-autoriser cette interruption.

I/13.3-Module Timer 1 :

Le Timer 1 est un compteur sur 16 bits constitué de 2 registres 8 bits TMR1Hen h'0F' page 0 et TMR1L en h'0E' page 0 également, que l'on peut lire ou écrire.

Le registre TMR1 (constitué de TMR1H et TMR1L) s'incrémente de h'0000'jusqu'à h'FFFF' et repasse ensuite à h'0000' pour continuer le comptage.

Quand il y a débordement, une interruption peut être générée si on l'autorisée par TMR1IE =1 (bit 0 de PIE1) et le Flag TMR1IF (bit 0 de PIR1) passe à "1".

Ce module peut fonctionner en mode TIMER, quand il s'incrémente à chaque cycle instruction (Fosc/4 avec le pré diviseur considéré à "1") ou en mode compteur, quand il s'incrémente à chaque front montant de l'horloge externe appliquée sur le Port C0.

L'horloge externe peut également être l'oscillateur interne, dont la fréquence est fixée par un quartz externe branché entre la broche Port C0 et la broche Port C1.

Le contrôle du TIMER 1 se fait par le registre T1CON en h'10' page 0.

I/13.3/1-T1CON : (h'10' : page 0) :

Bit 7 Bit 0

		T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON

Au reset : T1CON = 00000000

Bit 7 et bit 6 : **bits non implémentés.**

Bit 5 et bit 4 : **T1CKPS** = Sélection du pré diviseur placé avant le TIMER.

T1CKPS1	T1CKPS0	PRE DIV
0	0	1
0	1	2
1	0	4
1	1	8

Bit 3 : **T1OSCEN :** Bit d'autorisation de l'oscillateur du Timer 1.

1 = oscillateur autorisé

0 = oscillateur stoppé.

Bit 2 : **T1SYNC :** Bit de contrôle de la synchronisation du CLK externe.

1 = Pas de synchronisation de l'horloge externe.

0 = Synchronisation de l'horloge externe.

Bit1 : **TMR1CS :** Bit de sélection de la source horloge.

1 = Mode Compteur: Clk externe sur la broche PC0 ou Quartz entre PC0 et PC1

0 = Mode Timer: Clk interne = Fosc/4.

Bit 0 : **TMR1ON :** Bit d'autorisation du Timer 1.

1 = Timer 1 en service.

0 = Timer 1 stoppé.

I/13.4-Oscillateur interne du Timer 1 :

Un oscillateur à quartz a été embarqué sur le chip. Il est branché entre les broches PC0 (oscillateur out) et PC1 (oscillateur in). Il est mis en service par la mise à "1" du bit T1OSCEN. Cet oscillateur à faible consommation est limité à200 KHz. Il continu à osciller en mode SLEEP du PIC.

Il est principalement destiné pour générer un événement temps réel toutes les secondes par utilisation d'un quartz 32,768 KHz.

I/13.5-Schéma synoptique du Timer 1 :

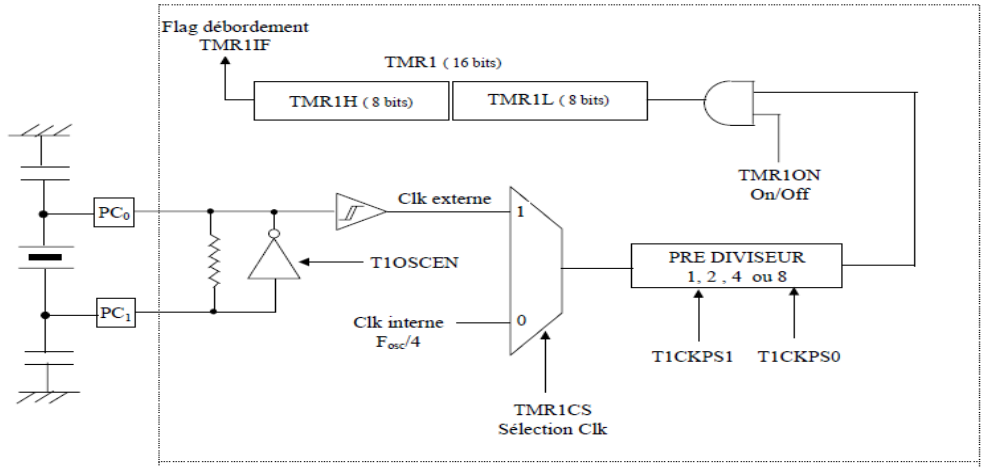


Figure I.8- Schéma bloc du Timer.

I/13.6-Module Timer 2 :

Le module Timer 2 est un compteur 8 bits avec pré diviseur et post diviseur.

Ce compteur TMR2 en h'11 ' page 0 est un registre en lecture ou écriture. Il possède un registre 8 bits pour la période : PR2 en h'92' page 1.

Le compteur s'incrémente de h'00' jusqu'à la valeur contenue par PR2 et repasse ensuite à "0" pour continuer le comptage. Au reset PR2 est initialisé à "FF".

L'entrée du compteur est l'horloge cycle interne : Fosc/4 qui passe à travers un pré diviseur programmable par 1, 4 ou 16.

La sortie du compteur passe dans un post diviseur programmable sur 4 bits entre 1 et 16.

Quand la sortie du compteur passe par la valeur programmée dans PR2, il y a génération d'une interruption (si elle a été autorisée par TMR2IE=1) et le flag
TMR2IF est positionné à "1". Ceci bien entendu en considérant le post diviseur programmé à "1".
Le contrôle du Timer 2 se fait par le registre T2CON en h'12' page 0.

I/13.6/1-T2CON : (h'12' : page 0) :

Bit 7 Bit 0

	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0

Au reset : T2CON = 00000000
Bit 7 : **bit non implémenté.**
Bit 6 à bit 3 : **TOUTPS :** Programmation du Post diviseur.
0 0 0 0 = post divise par 1.
0 0 0 1 = post divise par 2.
0 0 1 0 = post divise par 3.
. . . .
. . . .
1 1 1 1 = post divise par 16.
Bit 2 : **TMR2ON :** mise en service du Timer 2.
1= Timer 2 : On.
0= Timer 2 : Off.
Bit 1 et bit0 : **T2CKPS :** Programmation du pré diviseur.
0 0= pré_ diviser par 1.
0 1= pré_ diviser par 4.
1 X= pré_ diviser par 16.

I/13.6/2-Module ccp : capture compare et PWM :

Il y a deux modules identiques CCP1 et CCP2 composés chacun d'un registre16 bits. Ils peuvent opérer soit comme un registre 16 bits de capture, soit comme un registre 16 bits de comparaison, soit enfin comme un registre 8 bits pour générer du PWM.
Le module CCP1 est constitué de deux registres de 8 bits : CCPR1L en h'15'page 0 et CCPR1H en h'16' page 0. Ce module est contrôlé par le registreCCP1CON en h'17' page 0. La

sortie en mode COMPARE ou mode PWM et l'entrée en mode CAPTURE se font par la broche PC2.

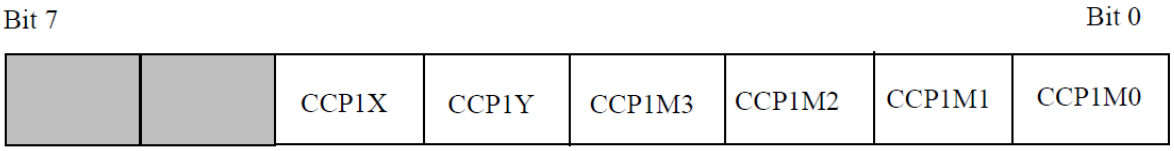
Le module CCP2 est constitué de deux registres de 8 bits : CCPR2L en h'1B'page 0 et CCPR2H en h'1C' page 0. Ce module est contrôlé par le registreCCP2CON en h'1D' page 0.

La sortie en mode COMPARE ou mode PWM et l'entrée en mode CAPTURE se font par la broche PC1.

En mode COMPARE ou CAPTURE, les modules utilisent le TIMER 1. En mode PWM, ils utilisent le TIMER 2.

Les registres de contrôles CCP1CON et CCP2CON sont identiques. On ne décrira que CCP1CON.

I/13.6/2.1-CCP1CON : (h'17' : page 0). (et CCP2CON en h'1D' : page 0.)



Bit 4 : **CCP1X et CCP1Y :**

Bits non utilisés en modes Compare et Capture.

Ce sont les 2 bits LSB pour le Duty cycle en mode PWM. Les 8 bits MSB sont dans le registre CCPR1L en h'15' page 0.

Bit 3 à bit 0 : **CCP1M3 à CCP1M0 :** bits de mode Au reset : CCP1CON = 00000000

Bit 7 et bit 6 : **bits non implémentés.**

Bit 5 et sélection du

0 0 0 0 = Module CCP stoppé.

0 1 0 0 = Mode Capture à chaque front descendant.

0 1 0 1 = Mode Capture à chaque front montant.

0 1 1 0 = Mode Capture tous les 4 fronts montants.

0 1 1 1 = Mode Capture tous les 16 fronts montants.

1 0 0 0 = Mode Compare. Pin de sortie mise à "1" et Flag CCP1IF = 1 à l'égalité.

1 0 0 1 = Mode Compare. Pin de sortie mise à "0" et Flag CCP1IF = 1 à l'égalité.

1 0 1 0 = Mode Compare. Génération d'une Interrup. et Flag CCP1IF = 1 à l'égalité.

1 0 1 1 = Mode Compare. Evénement spécial généré et Flag CCP1IF = 1 à l'égalité.

1 1 x x = Mode PWM.

I/13.6/2.2-Mode comparaison :

Les deux modules CCP étant identiques on ne décrira que le module 1.

Les 16 bits des registres CCPR1 (CCPR1H et CCPR1L) sont constamment comparés avec le valeur sur 16 bits des registres du Timer 1 (TMR1H etTMR1L).

Quand il y a égalité, la broche préalablement programmée en sortiePC2, passe soit à "1" soit à "0" suivant la configuration des 4 bits CCP1M du registre CCP1CON.

Au même instant le Flag CCP1IF est mis à "1".

En mode Compare, les événements spéciaux générés quand il y a égalité sont:

- Pour CCP1: reset du Timer 1.
- Pour CCP2 : reset du Timer 1 et démarrage d'une conversion A/D.

Dans ce cas la broche de sortie n'est pas affectée, mais le Flag CCP1IF est mis à "1".

Il est rappelé que ce Flag doit être remis à "0" par soft.

I/13.6/2.3-Mode capture :

Quand un événement extérieur apparaît sur la broche préalablement programmée en entrée PC2, la valeur du 16 bits des registres du Timer 1 (TMR1L et TMR1H) est recopiée dans les registres CCPR1 (CCPRIH etCCPR1L). Cet événement est programmable par les 4 bits CCP1M du registreCCP1CON. La capture peut avoir lieu à chaque front descendant, à chaque front montant, tous les 4 ou tous les 16 fronts montants.

Quand la capture a eu lieu, le flag CCP1IF est mis à "1".

Ce bit doit être remis à "0" par soft.

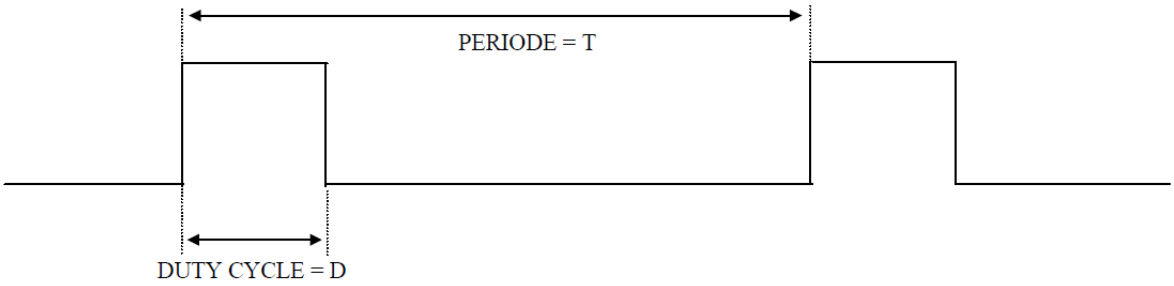
Si une nouvelle capture survient alors que la valeur dans CCPR1 n'a pas étélue, l'ancienne valeur est perdue.

Les fonctions de Capture et de Compare sur le Timer 1 par les modules CCP1et CCP2peuvent générer une interruption quand le Flag CCP1IF passe à "1"

Si le bit d'autorisation CCP1IE du registre PIE1 est mis à "1".

I/13.6/2.4-Mode PWM :

Un signal PWM est caractérisé par une période, et un temps de travail ou le signal est à "1". Ce temps est appelé DUTY CYCLE.



La broche PC2 doit être configurée comme une sortie en mettant le bit 2 de TRISC à "0".

Le signal PWM est fabriqué à partir du Timer 2. Le signal interne Fosc/4 passe à travers le pré diviseur programmable par 1, 4 ou 16 et fait compter TMR2.

Quand ce registre atteint la valeur écrite dans le registre PR2,

Trois événements se produisent :

- RAZ du registre TMR2.
- La broche de sortie PC2 est mise à "1", sauf si le Duty cycle vaut 0.
- Chargement de la valeur du registre CCPR1L dans le registre de Duty.

Quand le registre TMR2 atteint la valeur inscrite dans le registre interne de Duty, c'est à dire la valeur qui avait été inscrite dans CCPR1L, la broche de sortie PC2 est remise à "0".

La période est déterminée par la relation suivante : [7]

$$T = \left[PR_2 + 1\right] \cdot 4 \cdot T_{osc} \cdot \text{valeur du pré diviseur}$$

La durée du Duty cycle est la valeur écrite dans le registre 8 bits : CCPR1L.

La durée du signal au niveau "1" est donnée par la relation :

$$D = \left[CCPR1L\right] \cdot 4 \cdot T_{osc} \cdot \text{ valeur du pré diviseur}$$

On peut avoir une meilleure résolution sur le Duty cycle en utilisant les 2 bits de LSB réservés à cet effet dans le registre CCP1CON.

Il s'agit des bits 4 et 5 : CCP1X et CCP1Y. La résolution est dans ce cas divisée par 4. Si on appelle X le registre sur 10 bits, constitué de ces 2 bits en LSB et des 8 bits deCCPR1Lcomme MSB, la durée à "1" est donnée par la relation :

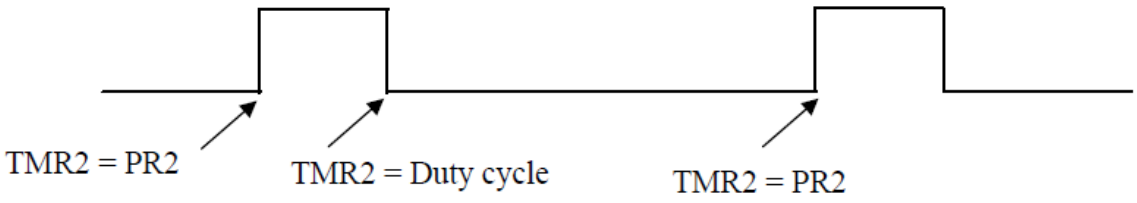
$$D = X \cdot T_{osc} \cdot \text{ valeur du pré diviseur}$$

Dans les 2 cas, quand on incrémente le registre CCPR1L de une unité, le Duty cycle augmente d'une durée égale à : 4.Tosc.valeur du pré diviseur.

Quand on passe à "1" le bit 5 la durée augmente de 2.Tosc.pré div.

Quand on passe à "1" le bit 4 de CCP1CON, la durée augmente de Tosc.pré div

Si la durée du Duty cycle est supérieure à la période, le signal n'est pas remis à "0".



I/13.6/2.5-Marche à suivre pour faire du PWM :

- Ecrire dans PR2 la valeur permettant d'obtenir la Période désirée.
- Ecrire dans CCPR1L la valeur donnant la durée du Duty cycle.
- Configurer la broche d'utilisation en sortie (bit = 0 dans TRISC).
- Initialiser le Timer 2 par T2CON (prédiv, TMR2 on).
- Initialiser le CCP par CCP1CON (mode PWM, 2 bits LSB Duty cycle).

I/13.7-Recapitulatif sur les TIMER et les modules CCP

TIMER 0 **Broche utilisée : PA4**	**TIMER 1** **Broches utilisées : PC0 et PC1**	**TIMER 2** **Broches utilisées : néant**
- 8 bits - Pré div de 2^0 à 2^8 - IT et Flag au débordement - Clk int ($F_{osc}/4$) ou ext (PA4)	- 16 bits - Pré div de 2^0 à 2^3 - IT et Flag au débordement - Clk int ($F_{osc}/4$) ou ext (PC0) - Oscil quartz ext PC_0/PC_1 - Sert en Capture et Compare	- 8 bits - Période programmable dans PR2 - Pré div par 1 , 4 ou 16 - Post div de 1 à 16. - IT et Flag quand Timer 2= PR2 - Clk int ($F_{osc}/4$) - Sert en PWM

CCP1 Broche utilisée : PC2	CCP2 Broche utilisée : PC1
- **Capture :** entrée sur broche PC2. *Recopie des 16 bits du timer 1 dans les 2 registres CCPR1H et CCPR1L quand survient un événement extérieur sur la broche PC2 tel que :* **ou ou 4 ou 16** - **Compare :** sortie sur broche PC2. *Comparaison des 16 bits du timer 1 et du contenu des 2 registres CCPR1H et CCPR1L et passage soit à"1" soit à "0" de la broche PC2.* - **PWM :** sortie sur broche PC2. *Période du signal donnée par Timer 2 et valeur dans PR2. Durée du Duty cycle dans CCPR1L.*	- **Capture :** entrée sur broche PC1. *Recopie des 16 bits du timer 1 dans les 2 registres CCPR2H et CCPR2L quand survient un événement extérieur sur la broche PC1 tel que :* **ou ou 4 ou 16** - **Compare :** sortie sur broche PC1. *Comparaison des 16 bits du timer 1 et du contenu des 2 registres CCPR2H et CCPR2L et passage soit à"1" soit à "0" de la broche PC1.* - **PWM :** sortie sur broche PC1. *Période du signal donnée par Timer 2 et valeur dans PR2. Durée du Duty cycle dans CCPR2L.*

I.14- 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.

Le microcontrôleur dispose de plusieurs sources d'interruptions.

• Une interruption externe, action sur la broche **INT/RB0**.

• Débordement du **TIMER0**.

• Changement d'état logique sur une des broches du **PORTB (RB4 à RB7)**.

• Une interruption d'un des périphériques (**PEIE**).

• Fin de programmation d'une case mémoire de l'**EEPROM**.

• Changement d'état sur le **PORTD** (**PSPIE**).

• Fin de conversion analogique numérique (**ADIE**).

• Réception d'une information sur la liaison série (**RCIE**).

• Fin d'émission d'une information sur la liaison série (**TXIE**).

• Interruption **SPI** ou **I2C** du module **MSSP** (**SSPIE**).
• Interruption du registre de capture et/ou de comparaison 1 (**CCPI1E**).
• Interruption du registre de capture et/ou de comparaison 2 (**CCPI2E**).
• Débordement du **TIMER1** (**TMR1E**).
• Débordement du **TIMER2** (**TMR2E**).
• Collision de BUS (BCLIE)

❖ *Mécanisme général d'une interruption :*

Nous pouvons dire, sans beaucoup nous tromper, qu'une routine d'interruption est un sous-programme particulier, déclenché par l'apparition d'un événement spécifique mais qui est très simple.

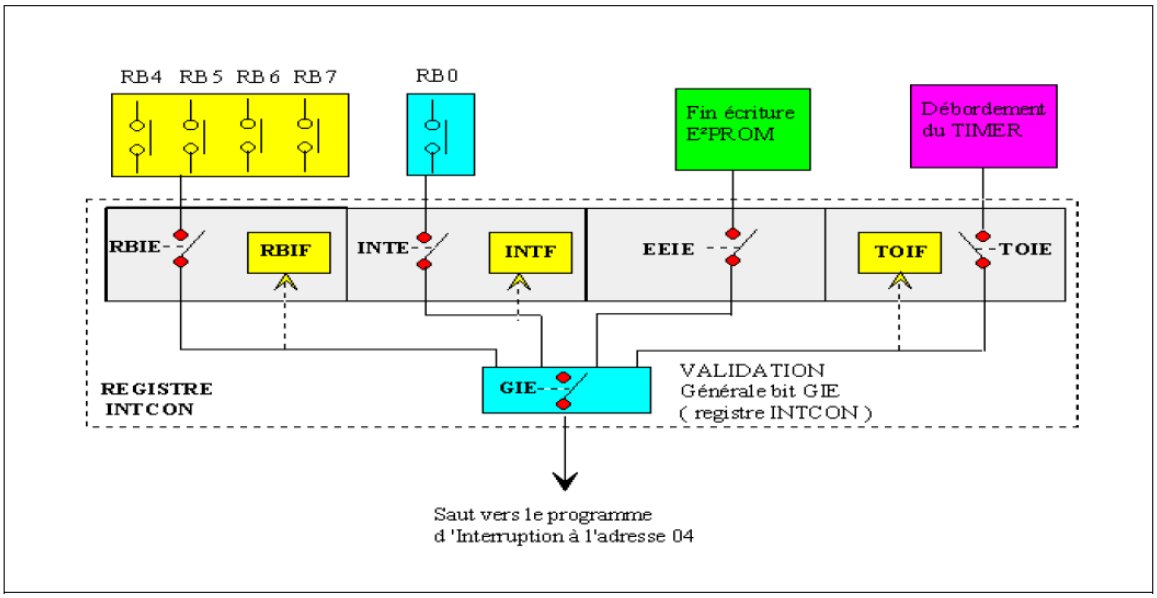


Figure I.9- Schéma de mécanisme d'interruption.

I.15- Conclusion :

Partant d'une présentation générale sur les microcontrôleurs, nous avons ensuite défini la famille des Pics et plus particulièrement le 16F876A.

En conclusion dans ce chapitre nous pouvons dire que le microcontrôleur peut bien jouer le rôle d'une unité de commande pour notre système.

Pour faire fonctionner cette unité de commande, il faut la programmer et l'adapter à un compilateur de programmation.

II/1-Introduction :

Le compteur binaire de type MC14020B 14 est construit avec des transistors MOSFET P-canal ou N-canal dans une structure monolithique simple.

La présente partie est conçue avec une onde d'entrée formant le circuit et 14 étapes d'ondulation-portent le compteur binaire.

Chaîne de tension d'alimentation de coupure (C.C) de VDD – 0,5 à +18,0 V.

II/2- Fonction :

Un compteur binaire est utilisé :

-pour compter un certain nombre d'évènements binaires.

-pour diviser la fréquence d'un signal logique par 2m.

Il restitue donc des signaux logiques à partir de signaux logiques d'entrée.

II/3-Principe d'un compteur binaire :

Un compteur est un dispositif **séquentiel** composé essentiellement de bascules associées les unes derrière les autres tels que le montre la figure suivante, et utilisées en diviseur de fréquence par 2:

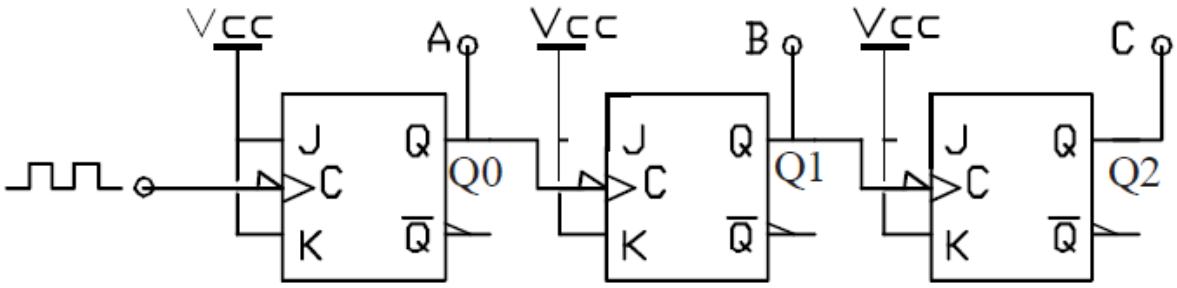


Figure II. 1- Structure interne de compteur.

Le branchement des bascules JK fait en sorte qu'à chaque front actif d'horloge (front descendant ici) la sortie change d'état.

La sortie de la première bascule fait office de signal d'horloge pour la deuxième etc...

La sortie de chaque bascule est appelée **Q** suivie d'un numéro qui correspond à la sortie de la première bascule (exemple : **Q0**).

En considérant que le signal d'entrée **Horloge** est un signal impulsionnel périodique (TTL), chaque sortie (pour un compteur binaire) divise la fréquence précédente par 2.

En reconstituant le mot binaire formé par **Q0** à **Q3** (où Q3 est le bit de poids fort), on constate que la structure compte le nombre de fronts actifs de l'horloge. Si elle comporte m bascules, elle est capable de compter 2m fronts actifs.

Par ailleurs, la fréquence de la sortie numéro n sera égale à la fréquence du signal d'horloge divisé par 2n+1

$$f_n = \frac{f_{HORLOGE}}{2^{n+1}}$$

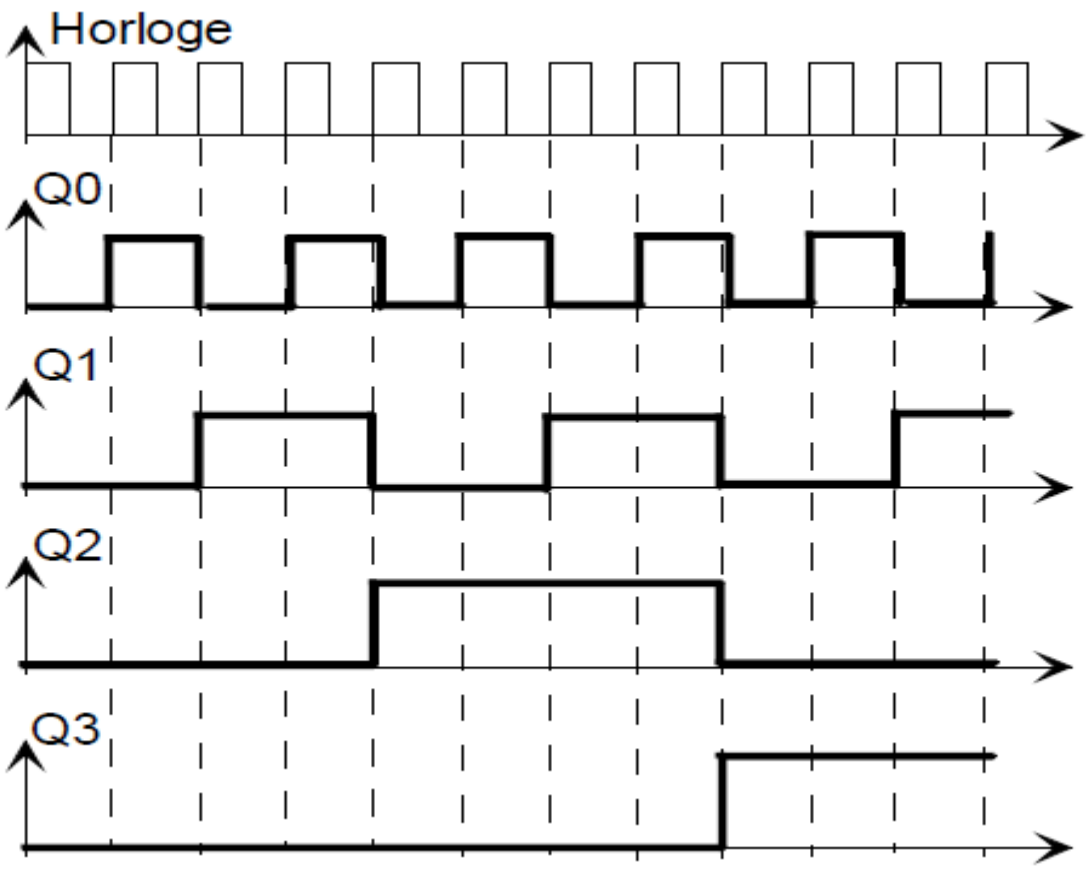


Figure II/ 2 -Chronogramme de compteur séquentiel.

II/*4-Principe d'un décompteur :*

Un décompteur possède les mêmes caractéristiques qu'un compteur à un point près qu'à chaque nouvel événement le code de sortie est décrémenté de 1.La structure interne simplifiée est la suivante :

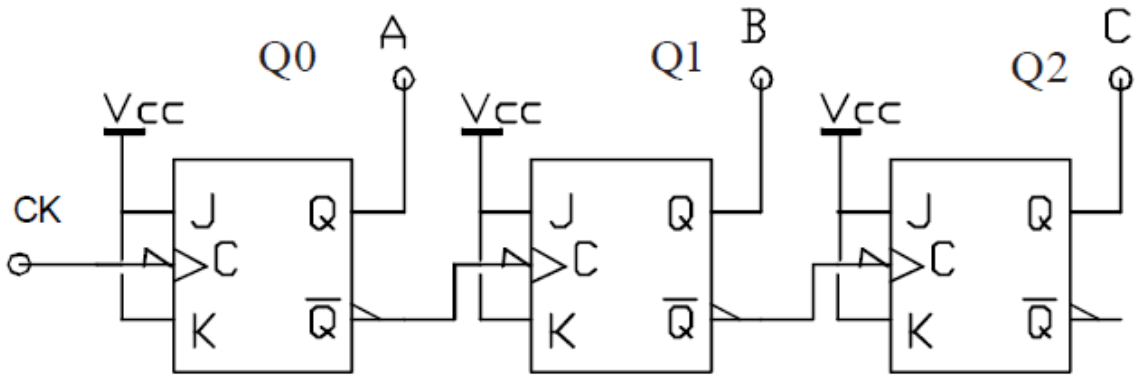


Figure II. 3 –Structure interne d'un décompteur.

Les chronogrammes de sortie deviennent :

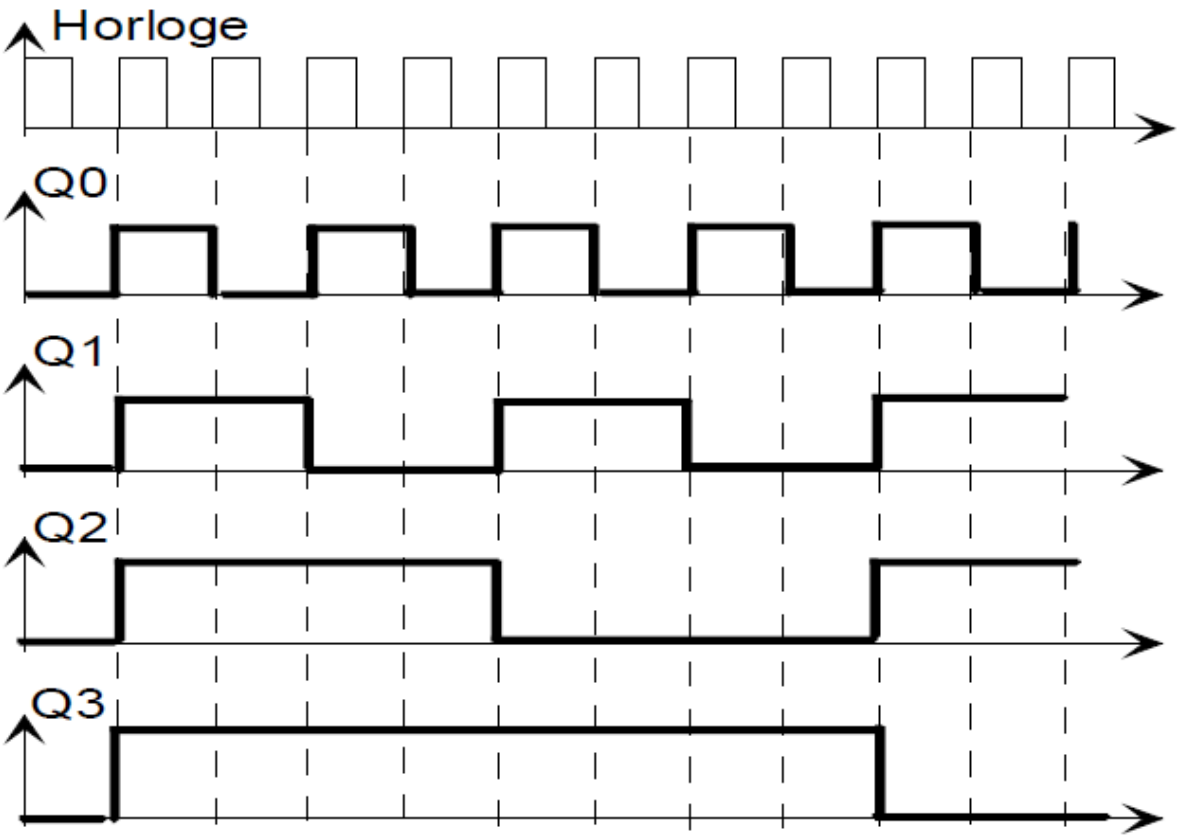


Figure II.4- Chronogramme de sortie de décompteur.

II/5-*Compteur synchrone et asynchrone :*

Pour les structures vues précédemment toutes les sorties ne changent pas d'état en même temps en raison du temps de propagation propre à chaque bascule. On dit que ces compteurs sont des **compteurs asynchrones.**

A fréquence élevée, cet asynchronisme peut entraîner des dysfonctionnements. Aussi a-t-on mis au point des **compteurs synchrones** pour lesquels toutes les sorties changent d'état simultanément.

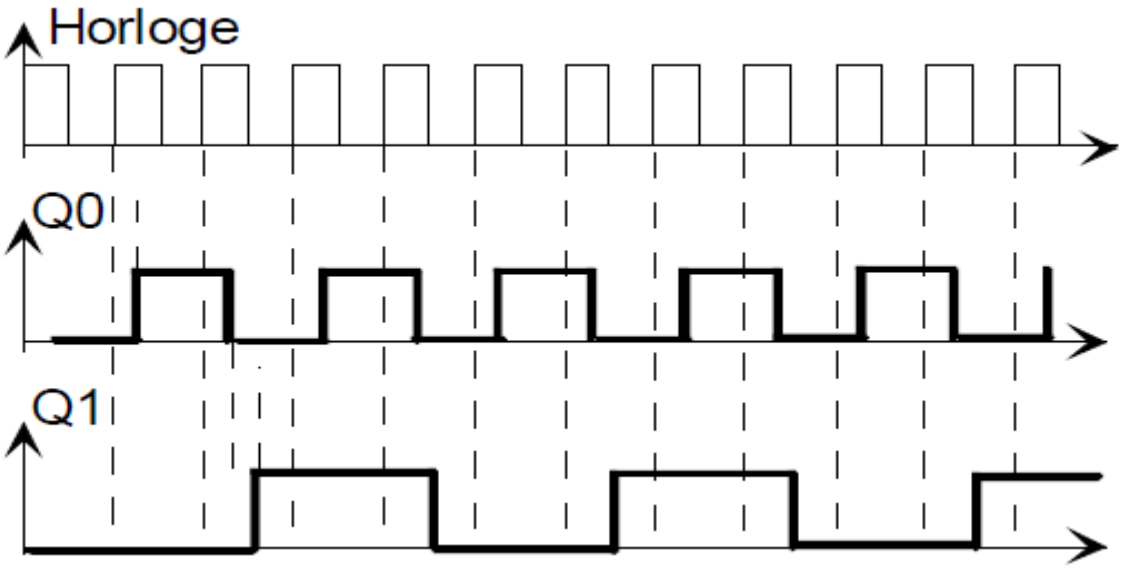


Figure II.5-Chronogramme du compteur synchrone.

Le temps de propagation n'est visible qu'entre le front d'horloge et le changement des sorties.

II/6- Les compteurs intégrés :

Compteur binaire simple : exemple du CD4020.

Ce compteur asynchrone en technologie CMOS intègre14 bascules dont toutes les sorties ne sont pas disponibles. Il est presque toujours utilisé en diviseur de fréquence assurant une division allant de 2(21) à 16384 (214).

Son entrée dynamique d'horloge est active sur front descendant. Comme tous les compteurs, il dispose d'une entrée prioritaire d'initialisation (ici de remise à zéro) permettant l'initialisation lors de la mise sous tension.

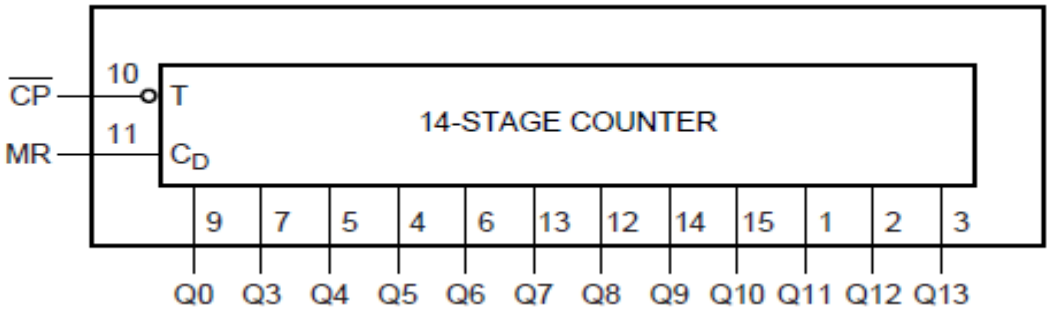


Figure II. 6-Schema interne de compteur BC4020.

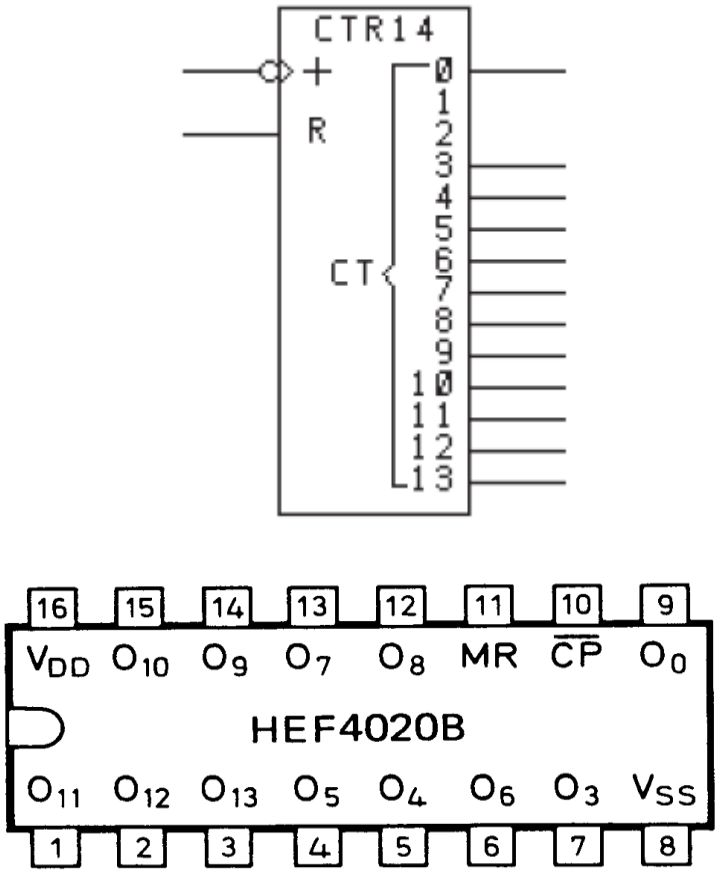
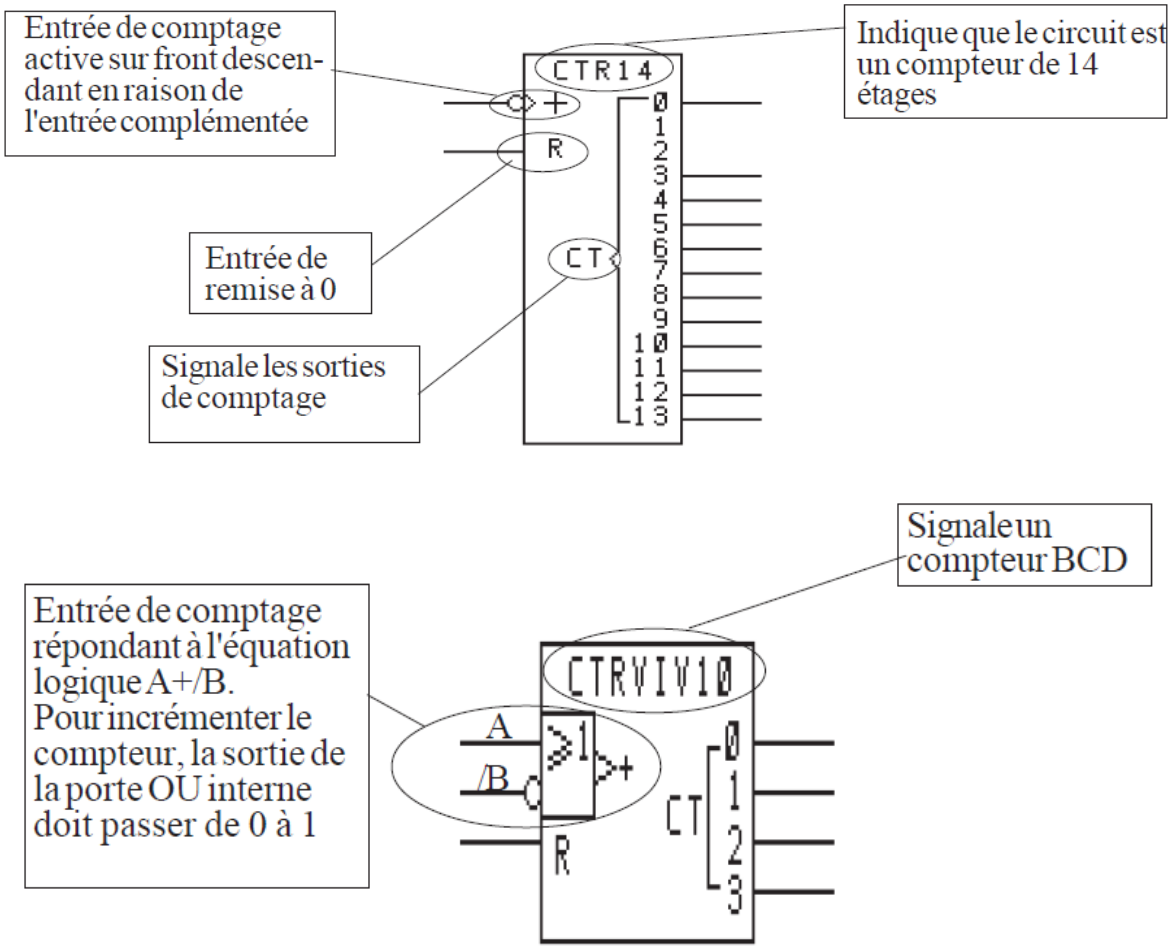


Figure II. 7- Brochage de compteur BC4020

II/7-Symbolisation normalisée :



II/8-Diagramme de logique.

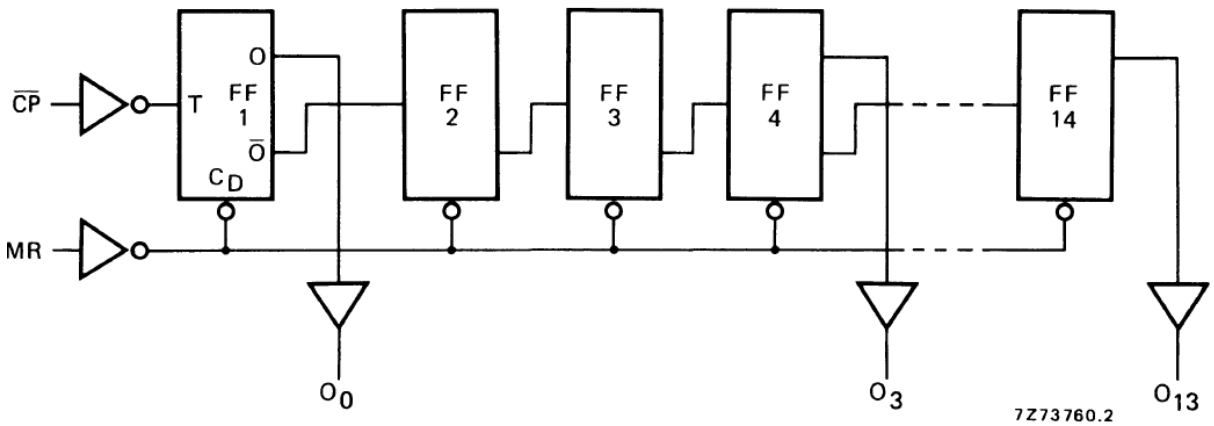


Figure II. 8-Diagramme de logique.

II/9- Les caractéristiques à C.A :

VSS = 0 V ; Tamb = °C 25 ; CL = 50 PF ; £ 20 NS de temps de transition d'entrée.

	V_{DD} V	SYMBOL	MIN.	TYP.	MAX.		TYPICAL EXTRAPOLATION FORMULA
Propagation delays							
$\overline{CP} \rightarrow O_0$	5			105	210	ns	78 ns + (0,55 ns/pF) C_L
HIGH to LOW	10	t_{PHL}		45	90	ns	34 ns + (0,23 ns/pF) C_L
	15			30	65	ns	22 ns + (0,16 ns/pF) C_L
	5			105	210	ns	78 ns + (0,55 ns/pF) C_L
LOW to HIGH	10	t_{PLH}		50	95	ns	39 ns + (0,23 ns/pF) C_L
	15			35	70	ns	27 ns + (0,16 ns/pF) C_L
$O_n \rightarrow O_n + 1$	5			80	160	ns	53 ns + (0,55 ns/pF) C_L
HIGH to LOW	10	t_{PHL}		30	60	ns	19 ns + (0,23 ns/pF) C_L
	15			20	40	ns	12 ns + (0,16 ns/pF) C_L
	5			70	140	ns	43 ns + (0,55 ns/pF) C_L
LOW to HIGH	10	t_{PLH}		25	50	ns	14 ns + (0,23 ns/pF) C_L
	15			20	40	ns	12 ns + (0,16 ns/pF) C_L
$MR \rightarrow O_n$	5			180	360	ns	153 ns + (0,55 ns/pF) C_L
HIGH to LOW	10	t_{PHL}		90	180	ns	79 ns + (0,23 ns/pF) C_L
	15			70	140	ns	62 ns + (0,16 ns/pF) C_L
Output transition times	5			60	120	ns	10 ns + (1,0 ns/pF) C_L
HIGH to LOW	10	t_{THL}		30	60	ns	9 ns + (0,42 ns/pF) C_L
	15			20	40	ns	6 ns + (0,28 ns/pF) C_L
	5			60	120	ns	10 ns + (1,0 ns/pF) C_L
LOW to HIGH	10	t_{TLH}		30	60	ns	9 ns + (0,42 ns/pF) C_L
	15			20	40	ns	6 ns + (0,28 ns/pF) C_L
Minimum clock	5		50	25		ns	
pulse width; HIGH	10	t_{WCPH}	25	15		ns	
	15		20	10		ns	
Minimum MR	5		130	65		ns	
pulse width; HIGH	10	t_{WMRH}	95	50		ns	
	15		90	45		ns	
Recovery time	5		115	60		ns	
for MR	10	t_{RMR}	65	35		ns	
	15		55	25		ns	
Maximum clock	5		5	10		MHz	
pulse frequency	10	f_{max}	13	25		MHz	
	15		18	35		MHz	

	V_{DD} V	TYPICAL FORMULA FOR P (µW)	
Dynamic power dissipation per package (P)	5	$600\ f_i + \sum (f_o\ C_L) \times V_{DD}^2$	where f_i = input freq. (MHz) f_o = output freq. (MHz) C_L = load cap. (pF) $\sum (f_o C_L)$ = sum of outputs V_{DD} = supply voltage (V)
	10	$2\ 800\ f_i + \sum (f_o\ C_L) \times V_{DD}^2$	
	15	$8\ 200\ f_i + \sum (f_o\ C_L) \times V_{DD}^2$	

II/10-Diviseur fixe à bascule D ou bascule JK :

Ce type de diviseur divis toujours par deux, c'est à dire que l'on retrouve sur la sortie Out, un signal dont la fréquence est exactement la moitié de celle du signal appliqué sur l'entrée In. Si le signal d'entrée est à 1 KHz, le signal de sortie est à 500 Hz. Le schéma de gauche est un diviseur par deux réalisé avec une bascule "D" (une moitié d'un CD4013), et le schéma de droite est un diviseur par deux réalisé avec une bascule "JK" (une moitié d'un CD4027).

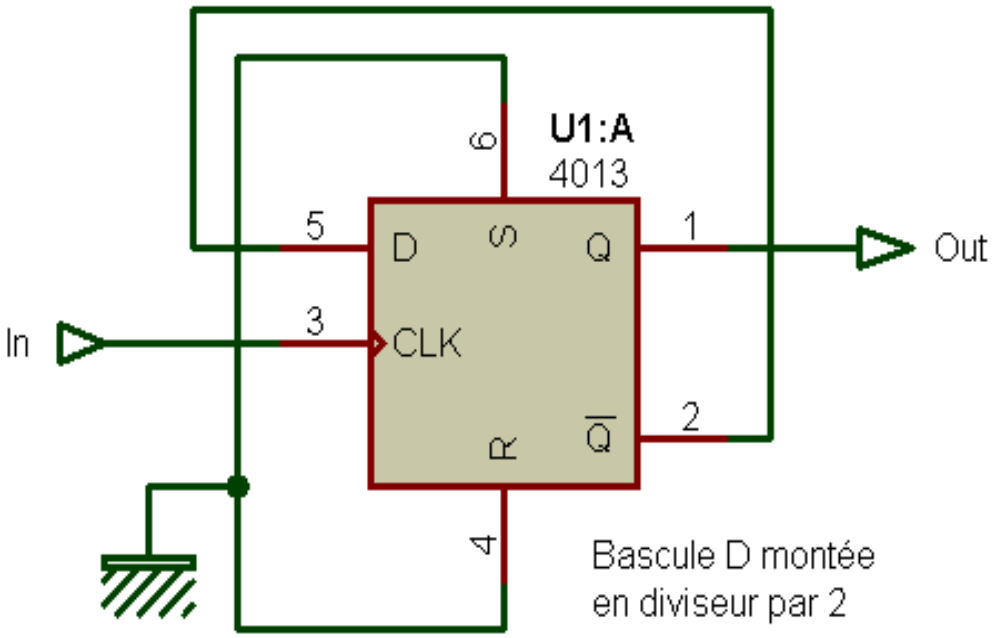


Figure II.9 –Schéma interne de bascule D

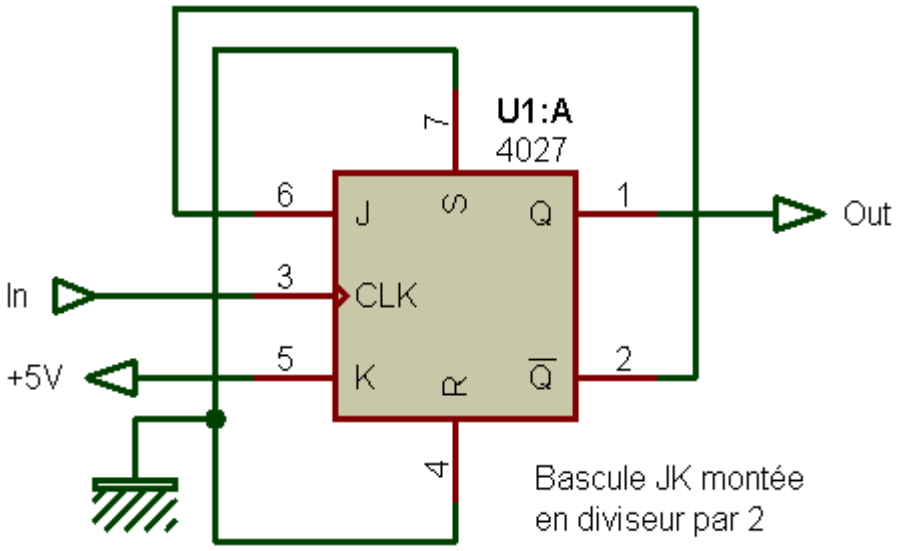


Figure II.10 –Schéma interne de bascule JK

En mettant plusieurs bascules de ce type en cascade, on peut obtenir un facteur de division de 2 puissances n, n étant le nombre de bascules. Le schéma suivant montre que l'on peut obtenir un facteur de division de 16 (2 puissance 4) avec quatre bascules D. On pourrait bien sûr faire de même avec les bascules JK.

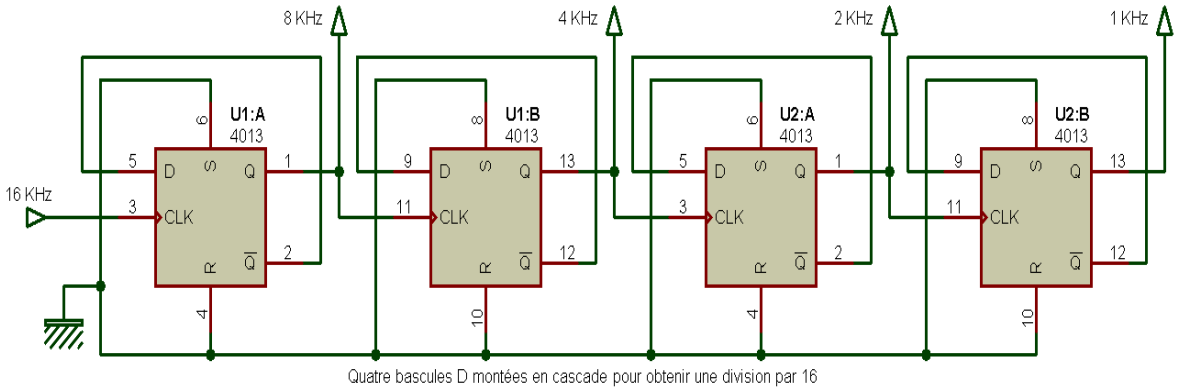


Figure II.11-Quatre bascules montées en cascade pour obtenir un diviseur.

En pratique, et quand le taux de division souhaité est important, il faut mettre en cascade un grand nombre de bascule ; cela prendrait une grande place. Si l'on souhaitait obtenir un facteur de division de 1024 ; en pratique, il est bien souvent préférable d'utiliser un circuit intégré spécialisé dans le comptage, qui intègre déjà un grand nombre de bascules, mettant ainsi à disposition une ou plusieurs sorties divisant successivement par 2 (2e1), par 4

(2e2), par 8 (2e3), par 16 (2e4), par 32 (2e5), etc. C'est le cas par exemple des circuits intégrés logiques CMOS CD4020, CD4040 ou CD4060.

II/12-Diviseur fixe avec un compteur binaire :

Division par un nombre égal à une puissance de deux : Les compteurs binaires à plusieurs étages (plusieurs bascules), tels les circuits CMOS CD4020, CD4040 ou CD4060, permettent d'obtenir un facteur de division de 2 puissance n, n dépendant de la sortie du compteur utilisé. Le schéma qui suit montre trois compteurs différents mais de même famille dont les entrées d'horloges sont reliées entre elles. Cette façon de faire n'est bien entendu pas obligatoire, c'est juste pour montrer sur un seul schéma, que l'on peut employer l'un ou l'autre des compteurs, selon le ou les taux de division souhaités.

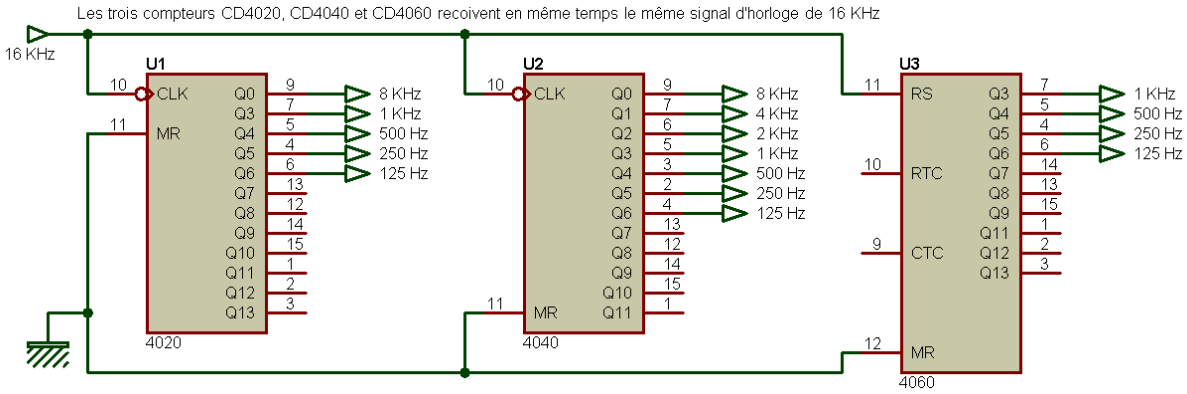


Figure II.12-Schéma des trois compteurs de la même famille reçoivent le même signal d'horloge.

II/13-Conclusion :

Certains compteurs ne mettent pas à disposition certaines sorties, sautant par exemple directement de la sortie Q0 (division par 2) à la sortie Q3 (division par 16). Si une des sorties manque cruellement pour une application, c'est qu'un seul circuit ne suffit pas, et il faut alors en ajouter un.

C'est le cas classique de la base de temps "1seconde" (1 Hz) que l'on peut obtenir avec un quartz 32,768 KHz monté avec un compteur CD4060, mais dont la sortie de rang le plus élevé du compteur délivre un signal de fréquence 2 Hz, et nécessite donc un diviseur supplémentaire pour obtenir le signal de 1 Hz souhaité.

III/I-Etude théorique :

III/I/1-Introduction :

Un fréquencemètre est un instrument de mesure destiné à afficher la fréquence d'un signal périodique.

Dans un fréquencemètre numérique, on compte le nombre de cycles qu'il y a pendant un intervalle de temps. Si on compte le nombre de cycle en 1 seconde, on aura par définition la fréquence en Hertz. On part généralement d'une référence à 10 MHz qui fournit les signaux pour l'intervalle de temps et pour les différentes impulsions nécessaires à la mise en mémoire du résultat et à l'affichage.

Dans les fréquencemètres bon marché la référence a une précision de 10^{-5} à 10^{-6}. Il s'agit en général d'un oscillateur à quartz avec une faible dérive et un faible coefficient de température. Pour une meilleure précision (10^{-7} à 10^{-8}) on place le quartz dans une enceinte où la température est maintenue constante grâce à un thermostat. Pour une meilleure précision encore on synchronise l'oscillateur de référence à 10 MHz avec une source radio (DCF 77 ...).

Application: Un fréquencemètre ordinaire avec une précision de $5\ 10^{-5}$ indique 430,067897 MHz. Quelle est la fréquence exacte ?

Solution : la fréquence exacte est située entre 430,067897 MHz - (430 MHz x $5\ 10^{-5}$) et 430,067897 MHz + (430 MHz x $5\ 10^{-5}$) soit 430 067 897 Hz $\pm$ 21500 Hz soit 430,046397 et 430,089397 MHz.

Conclusion : sur un fréquencemètre avec une précision "ordinaire" il n'est pas nécessaire d'avoir beaucoup de digits et on en notant les résultats, dans un rapport par exemple, on se limitera aussi aux chiffres significatifs. Par exemple, si la précision est à 10^{-n}, on ne donne que (n+1) chiffres.

Un fréquencemètre numérique est donc caractérisé par sa précision, celle-ci dépend aussi de la température. Mais la précision peut aussi être influencée par le temps. En anglais on parle alors d' "**aging**".

III/I/2-Fonctionnement :

L'appareil est principalement un compteur d'occurrences d'une transition caractéristique du signal entrant.

1. Un étage de mise en forme transforme le signal d'entrée en impulsions de même fréquence.
2. Un oscillateur aussi stable que possible appelé base de temps fournit la référence à laquelle comparer les fréquences.
3. La mesure peut se faire :

- Soit en comptant les impulsions issues de l'entrée dans un temps donné (correspondant à un nombre déterminé de périodes de la base de temps). On obtient directement la fréquence.
- Soit en comptant le nombre de périodes de la base de temps dans l'intervalle entre un nombre déterminé d'impulsions issues du signal d'entrée. On obtient un multiple de la période du signal à mesurer, à partir duquel il faut calculer la fréquence.
- Soit, indirectement, en mélangeant un signal dérivé des transitions caractéristiques à un autre, de fréquence proche, constitué à partir de la base de temps, et en mesurant ensuite, par l'un ou l'autre des moyens précédents, la fréquence des battements qui s'ensuivent.

L'affichage est généralement numérique. Il donne une fréquence moyenne et parfois une indication de la déviation du signal par rapport à cette fréquence.

III/I/3- Caractéristiques du fréquencemètre :

Un fréquencemètre est caractérisé par :

- la plage de fréquences.
- la précision, c'est-à-dire la précision de sa référence en fréquence.
- sa résolution, c'est-à-dire le nombre de digits affichés.
- sa sensibilité, c'est-à-dire la tension (dans le cas d'une entrée haute impédance) ou la puissance (dans le cas d'une entrée à 50Ω) minimale requise.
- par la tension maximum qu'on peut lui appliquer sans endommager le circuit d'entrée ("burn-out").

III/I/4-Porte logique NAND /74hc00n :

Les principales caractéristiques des circuits intégrés les plus utilisés par les amateurs d'électronique, constituent essentiellement les groupes suivants : Circuits TTL (Transistor-Transistor-Logique) de la famille 74.

***Figure III.1:** Photo de porte logique 74hc00n.*

III/I/4.1-Brochage et fonctionnement :

Les trois lettres sont suivies de 4 chiffres. Ce nombre peut être celui de Pro-Electron ou un numéro propre au constructeur. Mais dans tous les cas ce nombre indique qu'un autre circuit ayant le même numéro aura la même fonction. Hors le cas de diverses températures de fonctionnement les circuits portant les mêmes numéros sont interchangeables.

C'est ainsi que la fonction des circuits TTL HC 7400 et MIC 7400 est exactement la même, et le brochage est également identique sur les deux modèles

- 7400.
- Quatre portes NAND à 2 entrées.
- Durée de retard d'impulsions : 10 ns (9,5 ns, **3** ns).
- Consommation : 40 mW (8 mW, 76 mW).

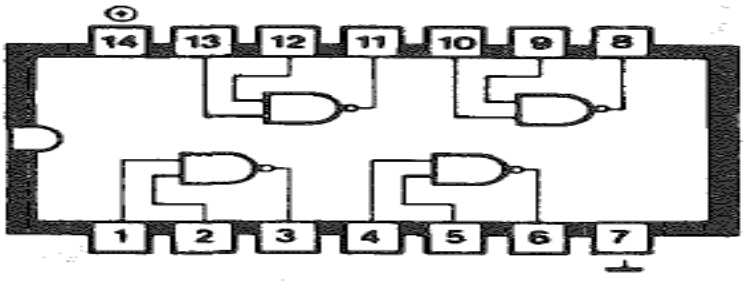


Figure III.2: Schéma interne de porte logique 74hc00n.

III/I/5-Ecran LCD série de jhd162a : CONTENU D'AFFICHAGE : 16 CHAR X 2ROW.

L'afficheur LCD c'est un module compact intelligent, nécessite peu de composants externes pour un bon fonctionnement. Il consomme relativement de 1 à 5mA .Plusieurs afficheurs sont disponibles sur le marché.

Ils sont très utilisés dans les montages à base de microcontrôleurs. Ils peuvent aussi être utilisés lors de la phase de développement d'un programme, pour faciliter l'affichage des valeurs de différentes variables.

Figure III.3-photo d'écran LCD

Figure III.3: Ecran LCD (vue de face et vue de dos.

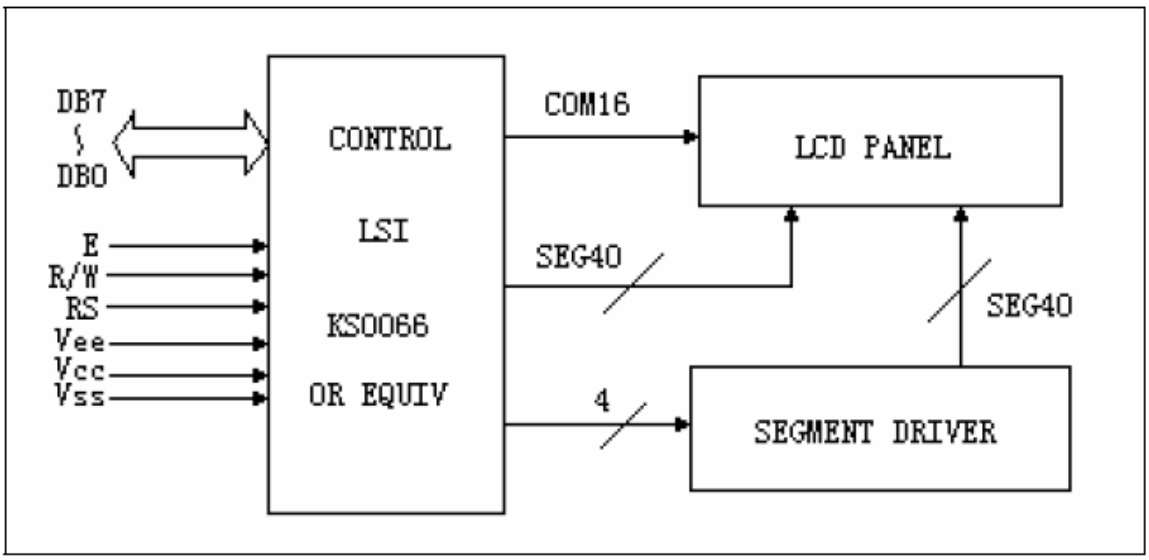
Figure III.4: Brochage d'écran LCD.

III/I/5.1-Characteristics:

- CHAR. POINTS : 5 x 8.
- ENTRAÎNEMENT DU MODE : 1/16D
- TYPES DISPONIBLES
- CONTRE-JOUR EL/100VAC, LED/4.2VDC.
- PARAMÈTRE (VDD=5.0V±10%, VSS=0V, Ta=25℃).

Parameter	Symbol	Testing Criteria	Standard Values			Unit
			Min.	Typ.	Max	
Supply voltage	V_{DD}-V_{SS}	-	4.5	5.0	5.5	V
Input high voltage	V_{IH}	-	2.2	-	V_{DD}	V
Input low voltage	V_{IL}	-	-0.3	-	0.6	V
Output high voltage	V_{OH}	$-I_{OH}$=02mA	2.4	-	-	V
Output low voltage	V_{OL}	I_{OL}=1.2mA	-	-	0.4	V
Operating voltage	I_{DD}	V_{DD}=5.0V	-	1.5	3.0	mA

III/I/5.2-Circuit d'application :



III/I/5.3-Contenu de dimensions/display :

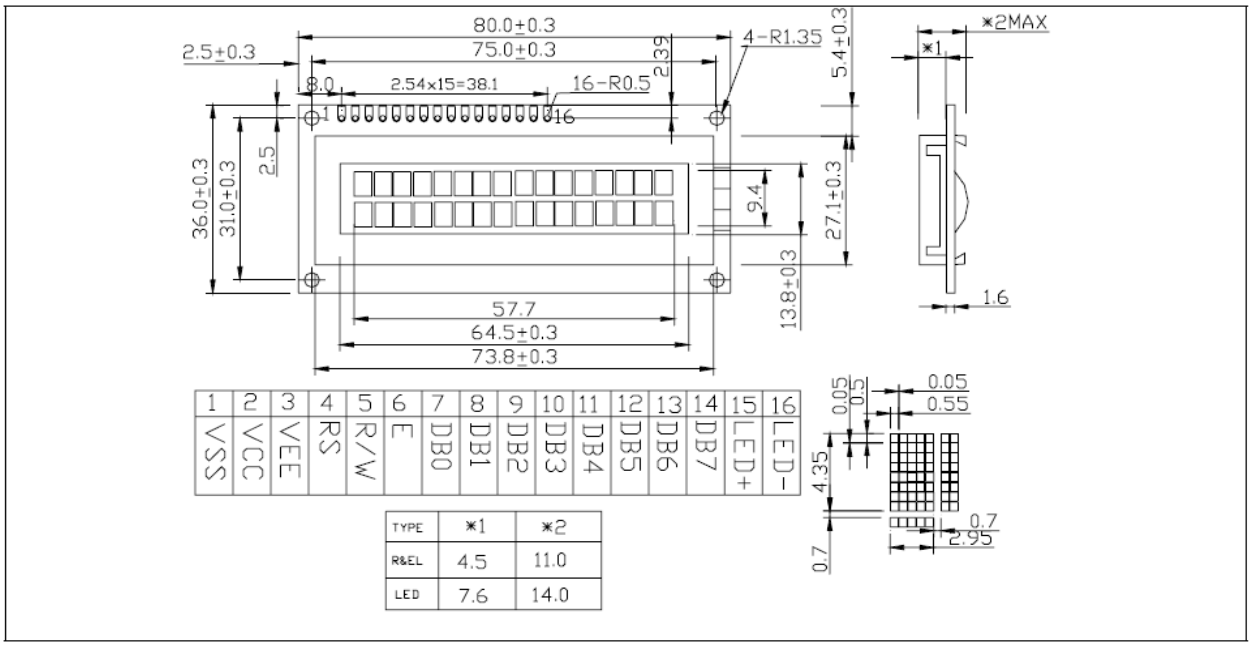


Figure II. 5 : Dimensions /display.

III/I/5.4-Brochage :

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
VSS	Vcc	VEE	RS	R/W	E	DB0	DB1	DB2	DB3	DB4	DB5	DB6	DB7	LED+	LED-

AC Characteristics (V_{DD} = 4.5V ~ 5.5V, Ta = -30 ~ +85°C)

Mode	Characteristic	Symbol	Min.	Typ.	Max.	Unit
Write Mode (Refer to Fig-6)	E Cycle Time	tc	500	-	-	ns
	E Rise / Fall Time	t_R,t_F	-	-	20	
	E Pulse Width (High, Low)	tw	230	-	-	
	R/W and RS Setup Time	tsu1	40	-	-	
	R/W and RS Hold Time	t_{H1}	10	-	-	
	Data Setup Time	tsu2	80	-	-	
	Data Hold Time	t_{H2}	10	-	-	
Read Mode (Refer to Fig-7)	E Cycle Time	tc	500	-	-	ns
	E Rise / Fall Time	t_R,t_F	-	-	20	
	E Pulse Width (High, Low)	tw	230	-	-	
	R/W and RS Setup Time	tsu	40	-	-	
	R/W and RS Hold Time	t_H	10	-	-	
	Data Output Delay Time	t_D	-	-	120	
	Data Hold Time	t_{DH}	5	-	-	

AC Characteristics (V_{DD} =2.7V ~ 4.5V, Ta = -30 ~ +85°C)

Mode	Characteristic	Symbol	Min.	Typ.	Max.	Unit
Write Mode (Refer to Fig-6)	E Cycle Time	tc	1000	-	-	ns
	E Rise / Fall Time	$t_R t_F$	-	-	25	
	E Pulse Width (High, Low)	tw	450	-	-	
	R/W and RS Setup Time	tsu1	60	-	-	
	R/W and RS Hold Time	t_{H1}	20	-	-	
	Data Setup Time	tsu2	195	-	-	
	Data Hold Time	t_{H2}	10	-	-	
Read Mode (Refer to Fig-7)	E Cycle Time	tc	1000	-	-	ns
	E Rise / Fall Time	t_R,t_F	-	-	25	
	E Pulse Width (High, Low)	tw	450	-	-	
	R/W and RS Setup Time	tsu	60	-	-	
	R/W and RS Hold Time	t_H	20	-	-	
	Data Output Delay Time	t_D	-	-	360	
	Data Hold Time	t_{DH}	5	-	-	

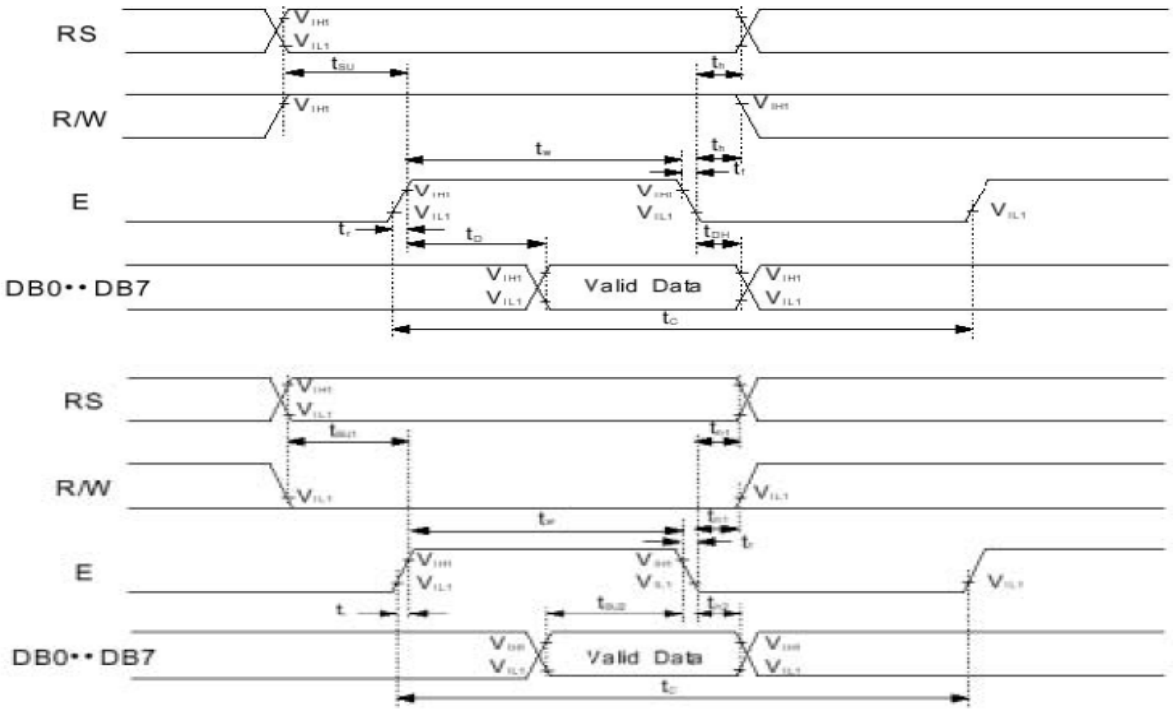


Figure III. 6 -Diagramme de synchronisation.

On écrit le diagramme de synchronisation de mode Synchronisation interface avec 8-bitMPU ; en connectant la longueur de données soyez à 8 bits le transfert est exécuté à la fois par 8 que le port de l'exemple db0todb7 de l'ordre de synchronisation est montré ci-dessous.

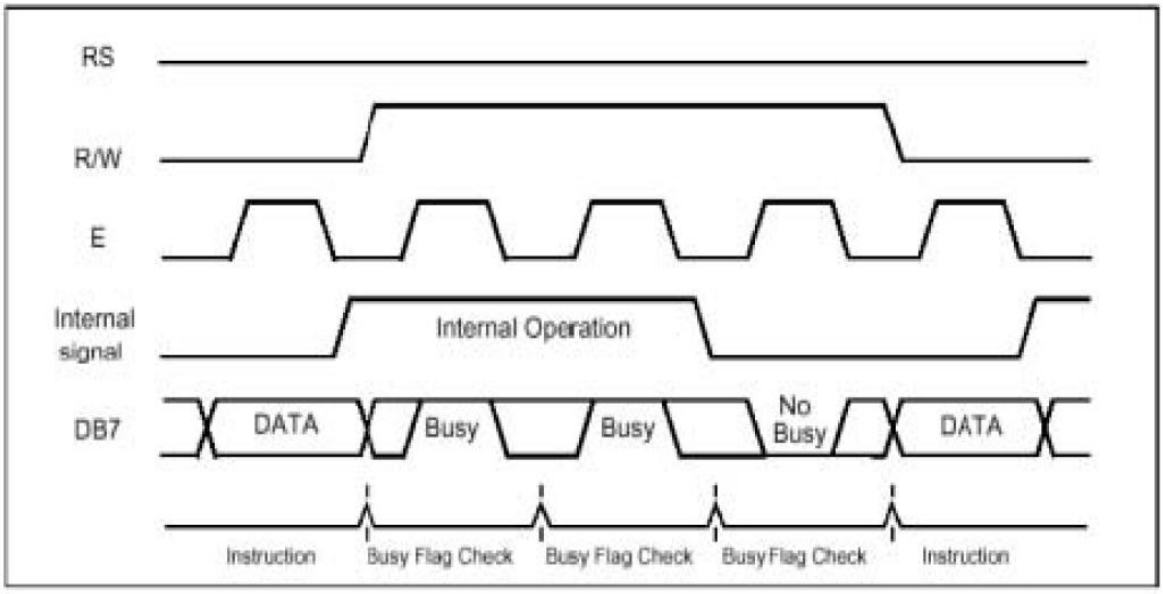


Figure III. 7 : Schéma descriptif de diagramme de synchronisation.

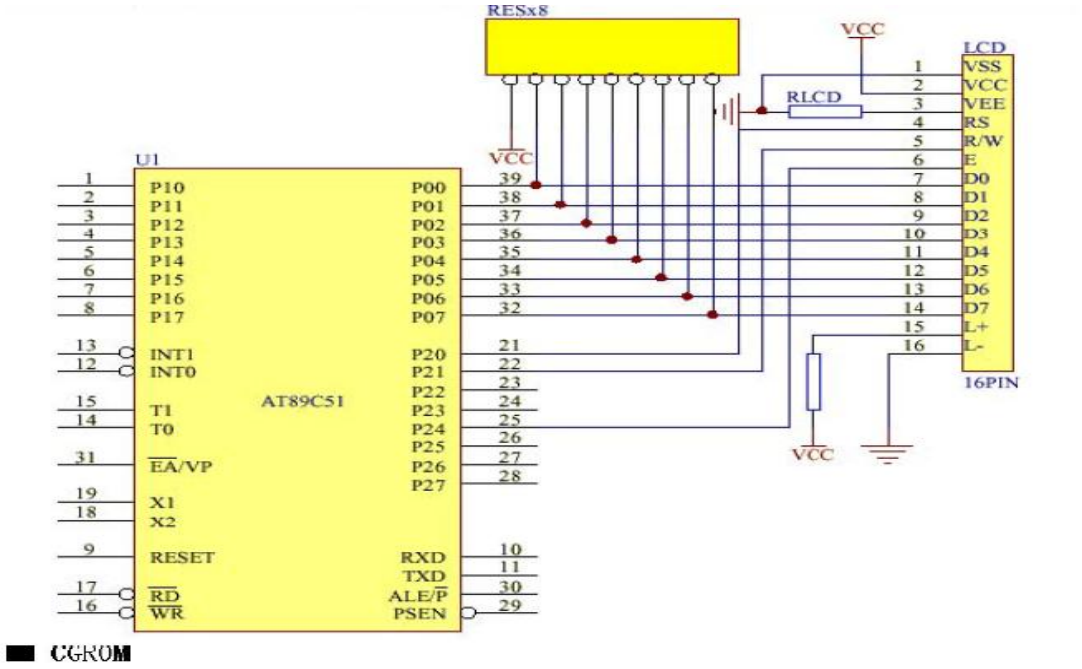


Figure III.8 : Schéma descriptif de brochage d'écran LCD.

III/II-Etude pratique :

III/II/1-Introduction :

L'idée principale de ce projet été de réaliser un fréquencemètre multi calibre. Ce qui fait que ce travail a été réalisé sur un circuit imprimé,

La réalisation de cet instrument requiert diverses notions dans plusieurs domaines : électronique, informatique.

La commande de cet instrument sera assurée principalement par un microcontrôleur PIC, et un compteur d'impulsion ; et comme signal d'entrée, un signal guidé par une porte logique NAND et l'affichage sera sur écran LCD.

II/2- Schéma bloc:

La figure suivante représente les étages qui constituent notre fréquencemètre multi calibre.

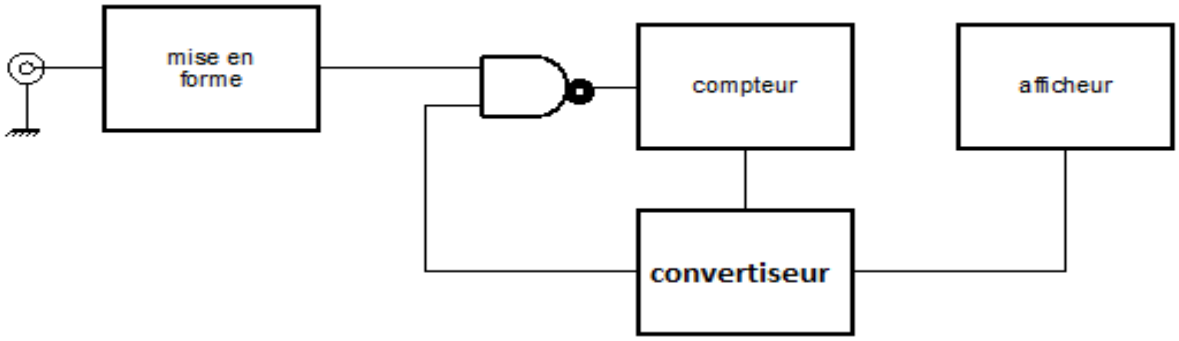
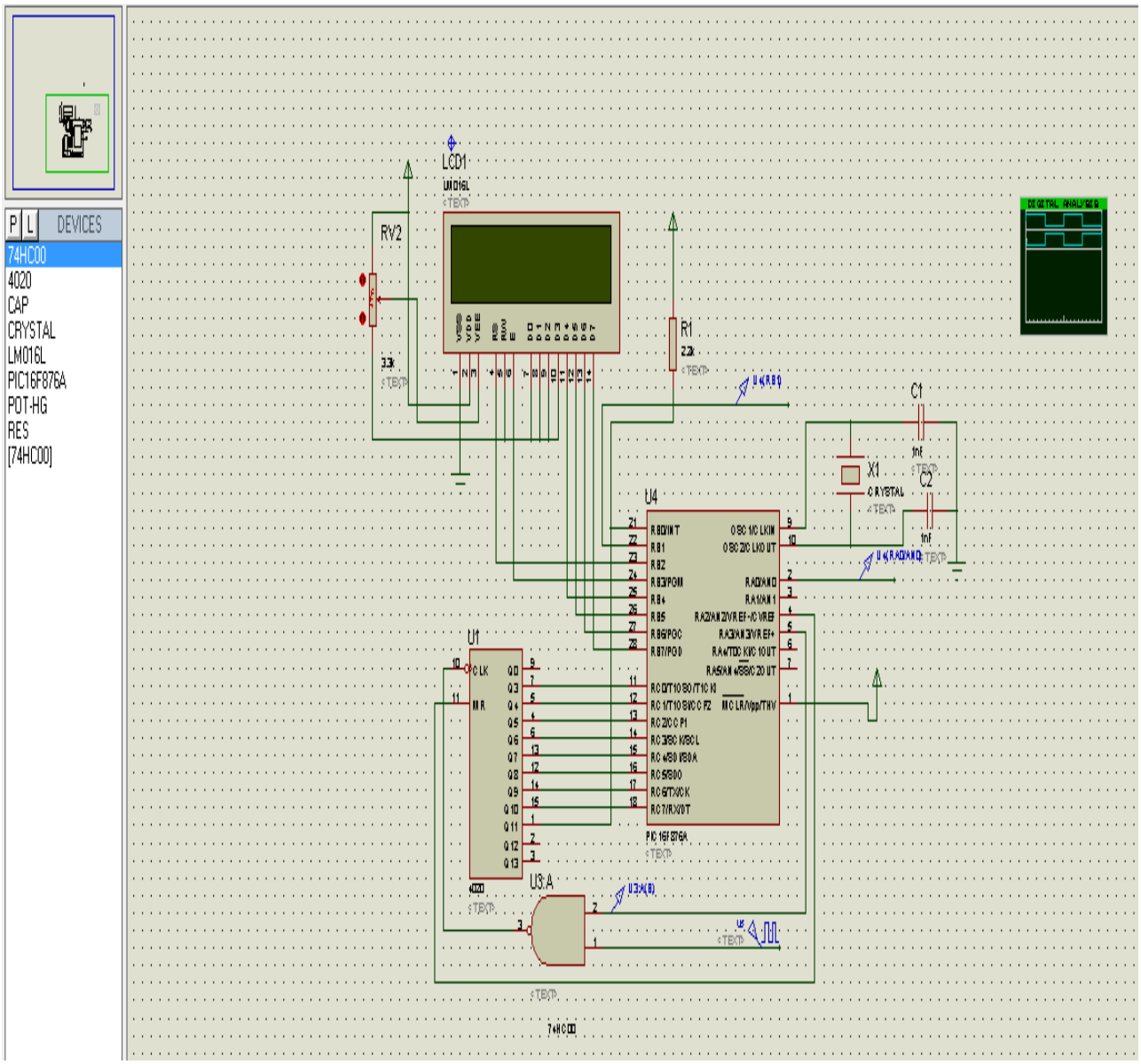


Figure III.9 : Schéma synoptique.

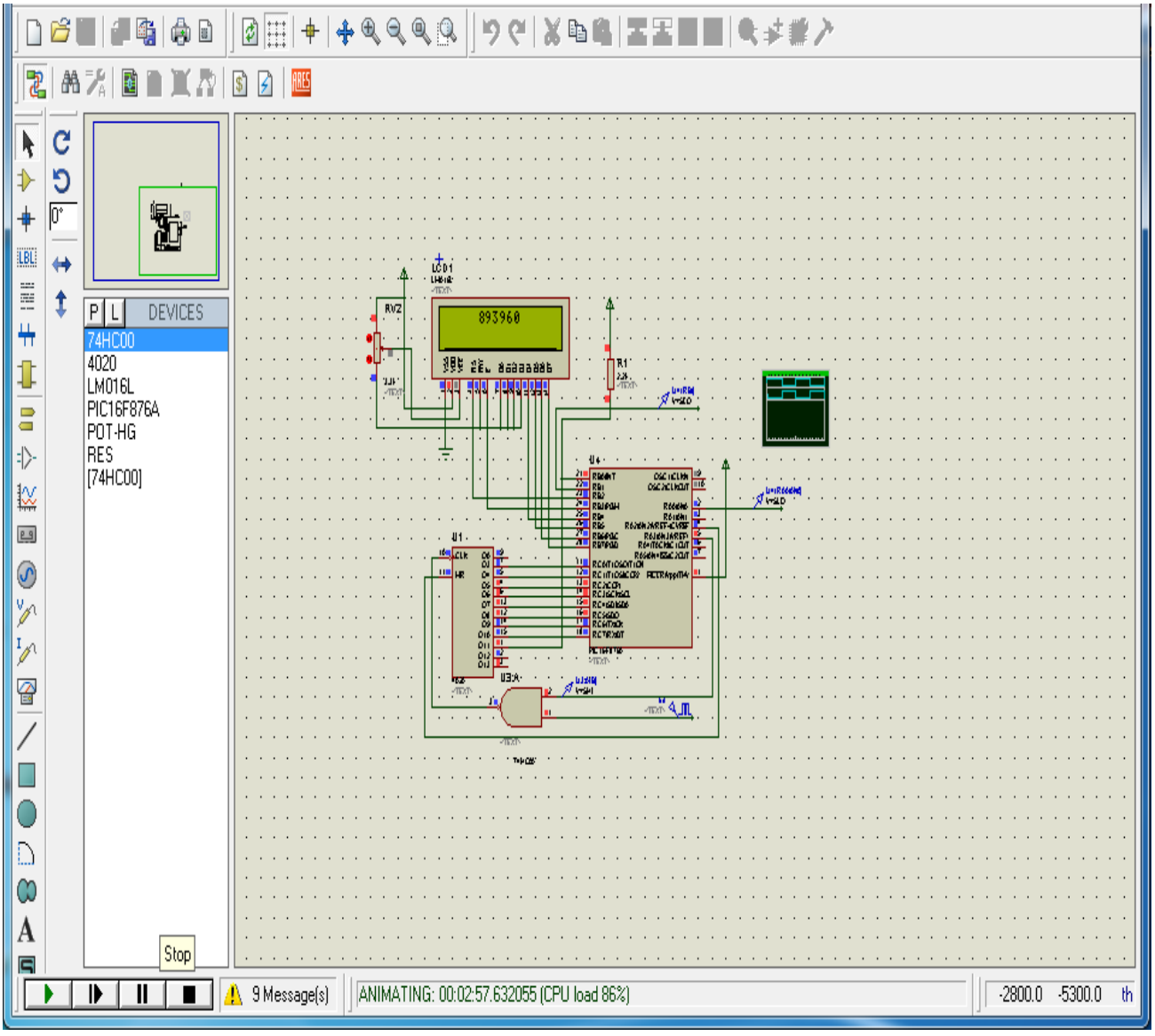
III/II/3-Schéma électrique du fréquencemètre multi calibre :



III/II/3.1-liste des composants utilises :

- Le microcontrôleur pic 16F876A.
- Le compteur BC4020.
- Porte logique NAND 47HC00N
- Deux condensateurs 27 pico farad.
- Quartz 12 méga hertz.
- Connecteur 16 pin.
- Trois résistances : 300 Ω, 553 Ω, 1k Ω.
- Ecran LCD 16 caractères 2 lignes.

II/3.Schéma électrique de fréquencemètre multi calibre sous simulation ISIS :



II/4-Realisation :

Dans les figures suivantes on peut voir le circuit imprimé du fréquencemètre et sa visualisation en trois dimensions (3D) de composants, réalisée par le logiciel de simulation PROTEUSE.

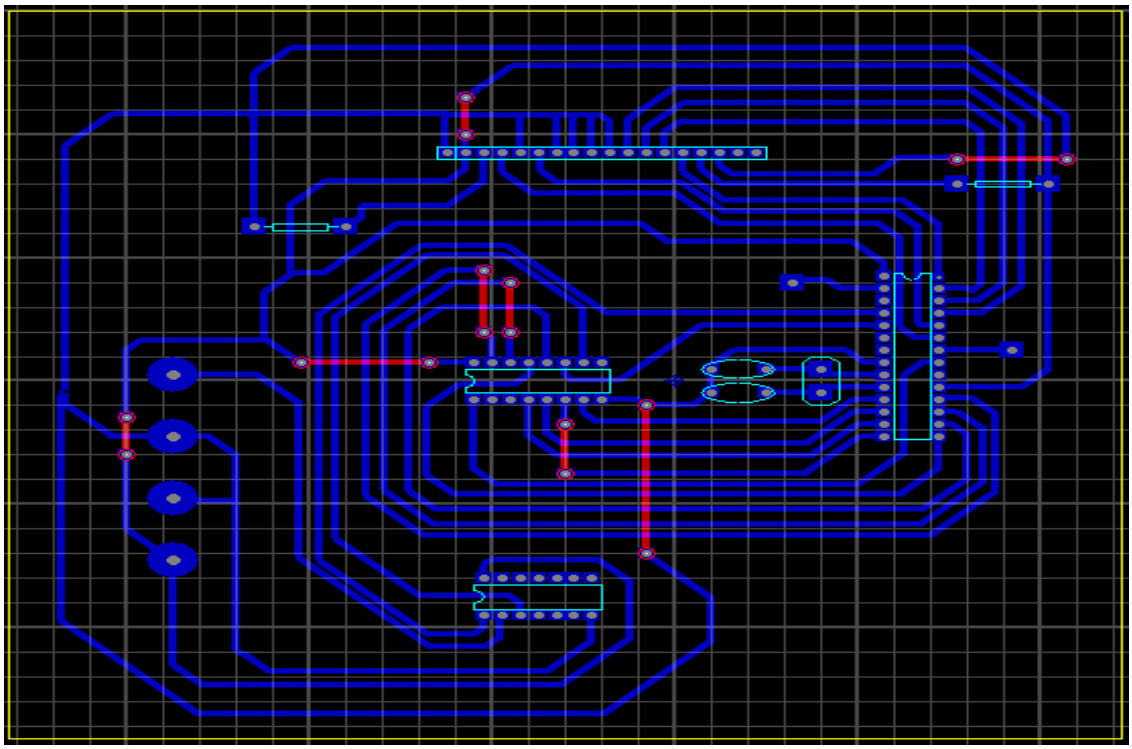


Figure III.10: Circuit imprimé du fréquencemètre multi calibre (avec composants).

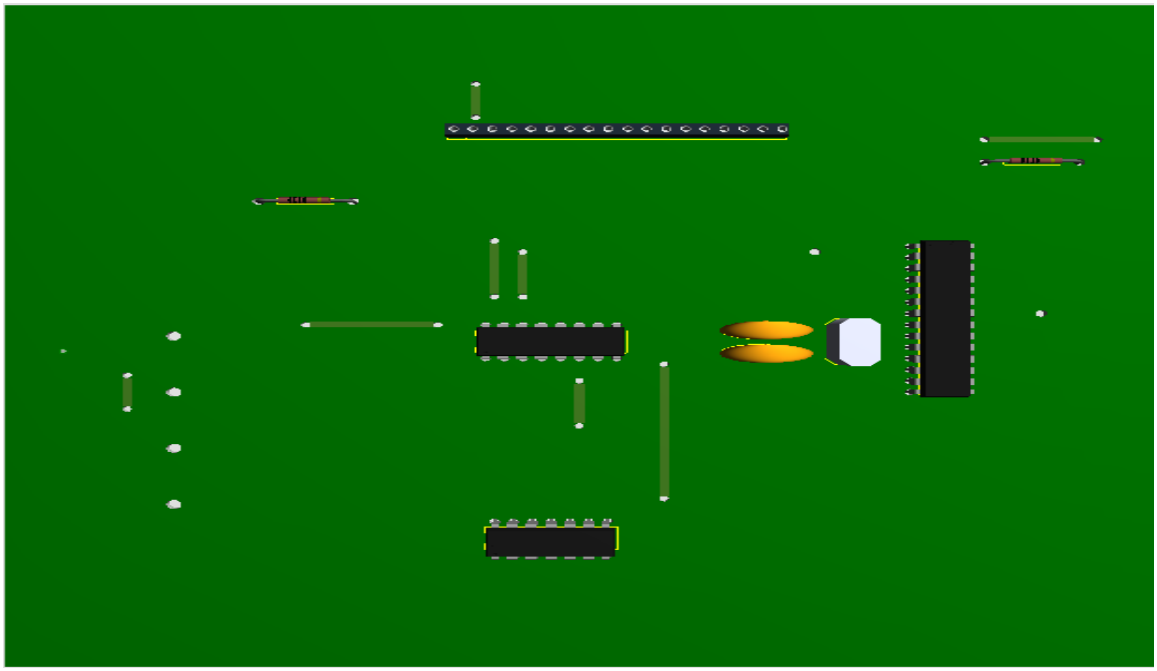


Figure11: Vue 3D du circuit réalisé

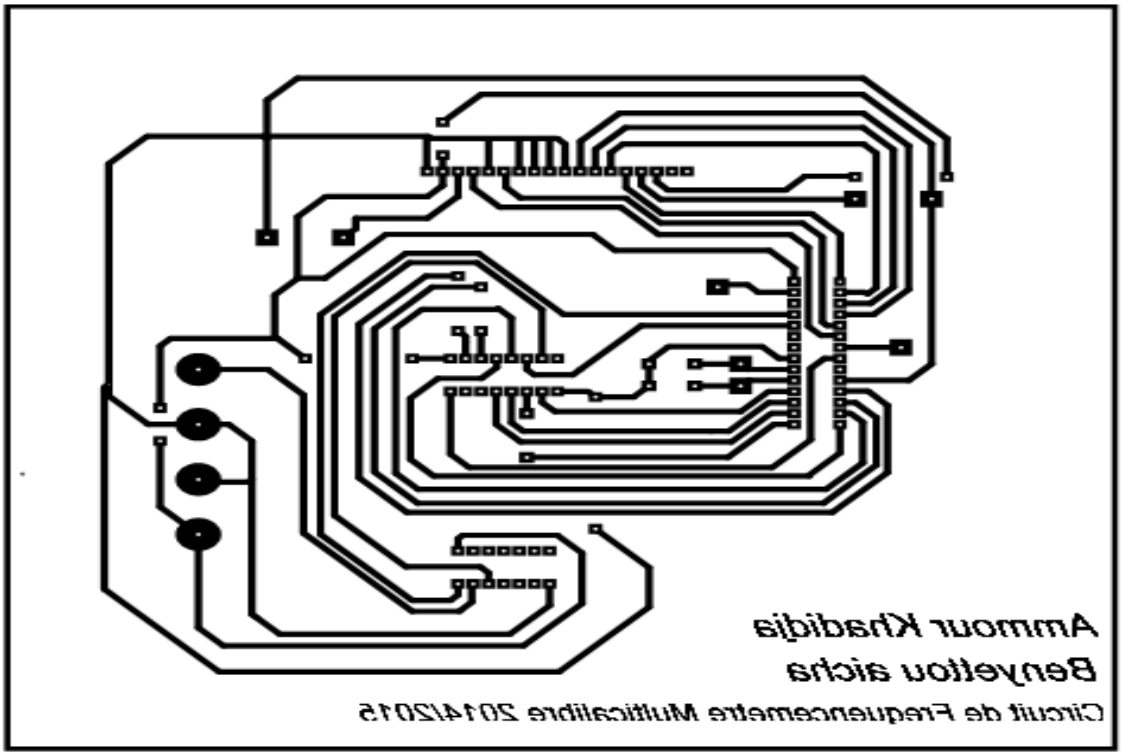


Figure III.12: Circuit imprimé côté cuivre.

III/II/5-Partie logiciels et programmation:

Nous avons utilisé plusieurs logiciels pour réaliser ce travail ; nous allons les présenter dans ce qui suit.

II/5.1- Logiciel de simulation PROTEUS :

Le Logiciel de simulation des circuits électronique est un logiciel qui dessine un circuit par les composant et les circuits intégrés et permet de voir les résultats de la réalisation pratique, il existe plusieurs simulateurs : Work bench, Multi Sim, PROTEUS, Tina…etc. Pour notre cas on a utilisé PROTEUS.

Le logiciel PROTEUS se compose de deux parties, le logiciel ISIS pour la simulation des circuits électroniques, et le logiciel ARES pour dessiner les circuits imprimés.

Le but du logiciel ISIS est de dessiner, simuler des circuits électroniques et tracer les courbes (Voir ANNEXE A).

La deuxième partie le logiciel ARES ; est un logiciel permettant le routage de cartes électroniques en mode automatique ou manuel. Il est possible d'utiliser ARES sans créer du schéma électronique dans l'ISIS. Cette fonctionnalité permet de réaliser très rapidement des circuits de faible complexité en plaçant les composants et traçant les pistes directement dans ARES. Une fois les connections établies il est possible d'effectuer un routage automatique des pistes (Voir ANNEXE A).

III/II/5.2- Logiciel de programmation **IC-P**

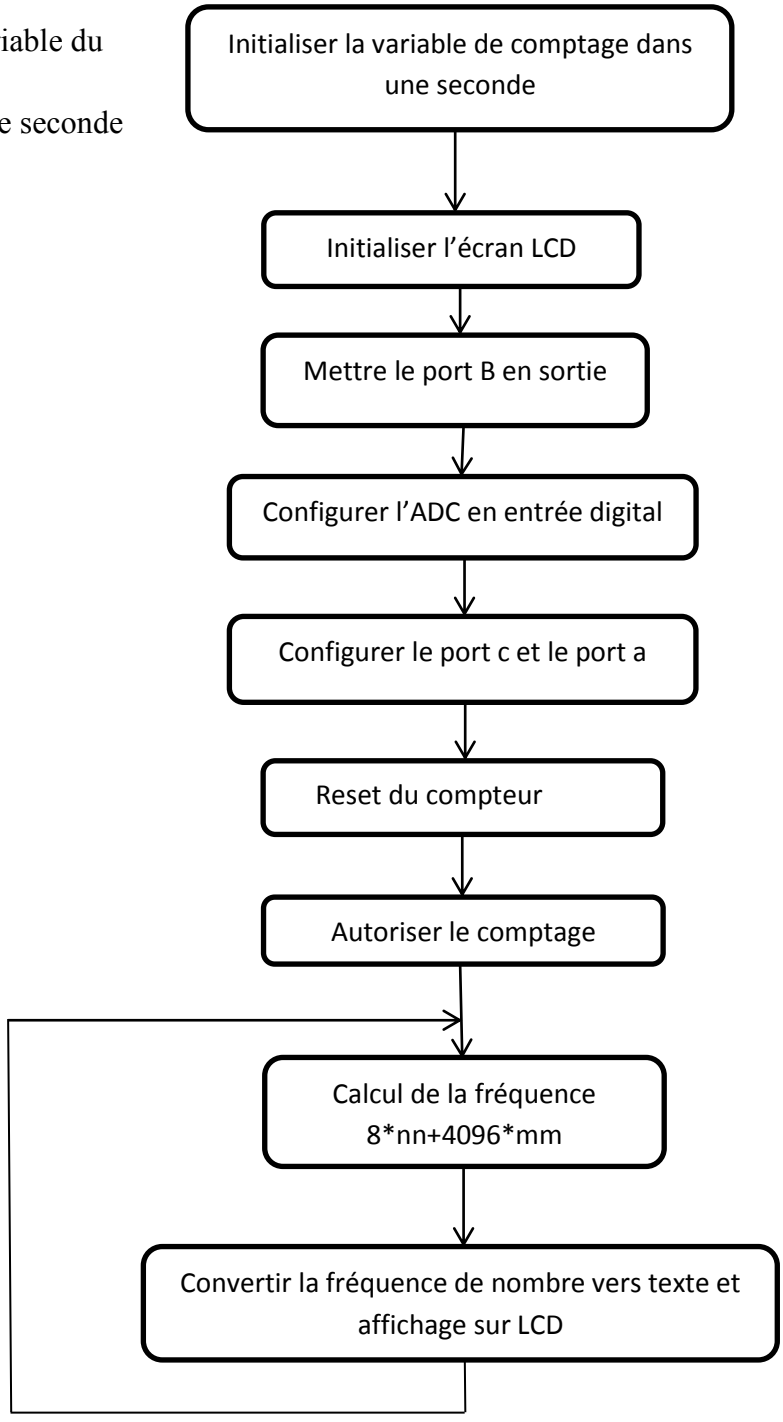
Nous allons programmer notre PIC pour commander le fréquencemètre multi calibre à servir l'utilité de l'utilisateur, il y a plusieurs logiciels pour programmer le PIC comme : MPLAB, micro-c, CCS Compile, flow code**,** IC-PROG,…etc. pour notre application on a choisi d'utiliser le logiciel IC-PROG.

Après la compilation du programme et bien évidemment après sa simulation, on passe à une phase très importante c'est le transfert du code source vers le PIC. En effet il suffit d'insérer le PIC 16F876A sur le support du programmateur, puis lancer le programme IC-PROG (voir annexe B).

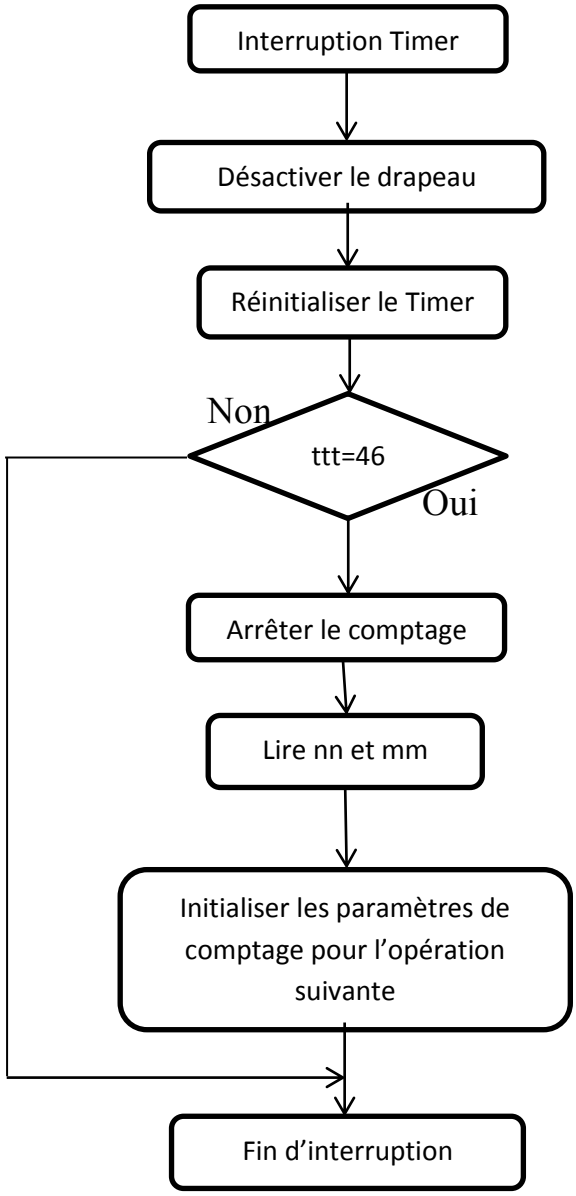
*III/**II/6- L'organigramme***

III/II/6.1-Lorganigramme du programme principal :

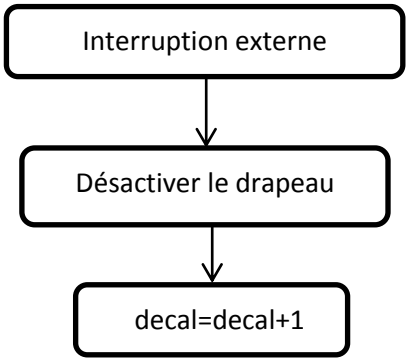
*ttt=0 :c'est la variable du comptage dans une seconde



III/II/6.2-l'organigramme initial :



III/II/6.3-l'organigramme d'interruption :



- nn : lu directement sur port c.
- mm : nombre*4096 est lu par l'interruption intf.

II/7-Photos de fréquencemètre réalisé :

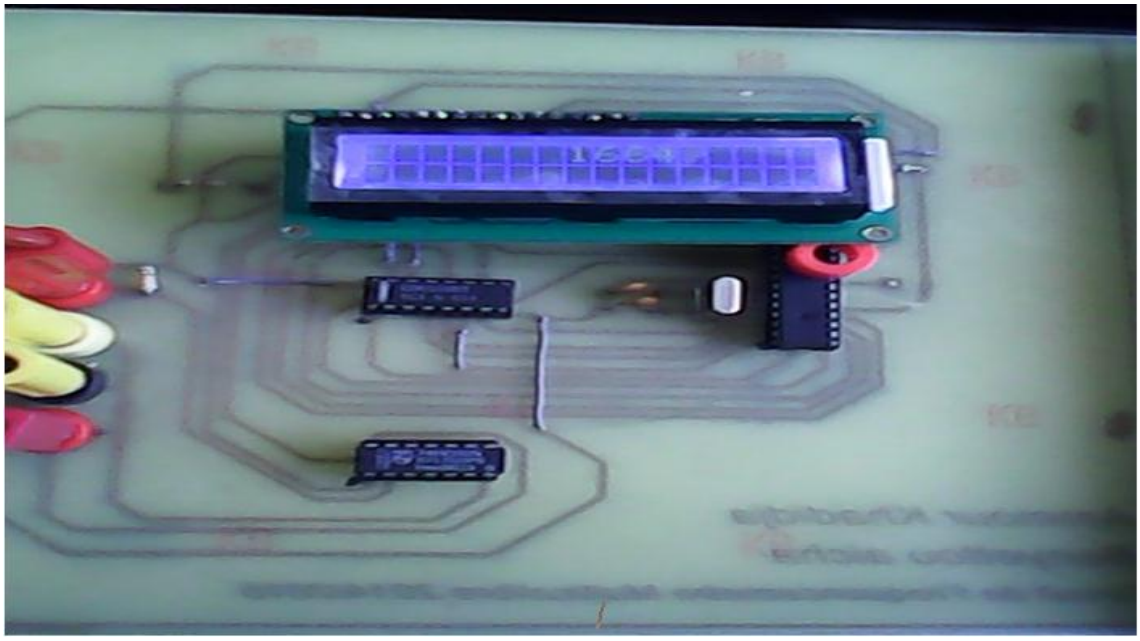


Figure III.14:Photo de fréquencemètre réalisé en état marche.

Figure15: photo de fréquencemètre réalisé en boitier.

III/II/8-Quelques mesures de fréquence effectuées avec le fréquencemètre réalisé :

Fréquence injectée	Fréquence affichée
10 HZ	8 HZ
109HZ	104 HZ
216 HZ	216HZ
500HZ	504HZ
1KHZ	1KHZ
500KHZ	500KHZ
655KHZ	556KHZ
749.5KHZ	750.12KHZ
861.5KHZ	865.4KHZ
958.07KHZ	960.1KHZ
1.01720MHZ	1.021448MHZ

III/III-Conclusion :

Dans ce chapitre nous avons présenté l'essentiel de notre travail qui consiste à la réalisation d'un fréquencemètre multi calibre. On a présenté quelques composants qui constituent notre instrument (porte NAND 74 hc00n, écran LCD.), ainsi que les logiciels et la programmation utilisés.

On a vu aussi comment charger le programme dans le PIC et l'organigramme du programme principal de notre appareil.

Conclusion générale

L'objectif de notre travail était l'étude et la réalisation d'un fréquencemètre multi calibre pour mesurer des fréquences de signaux donnés.

Pour réaliser ce travail, on a passé par différentes étapes :

On a utilisé une porte logique NAND 74hc00n pour injecter un signal carré + 5V et un compteur BC4020 pour pouvoir le lire. Le compteur nous donne des impulsions qui vont être transformées en signaux analogiques et transmis à un microcontrôleur PIC 16F876A. Ce dernier va les convertir sous forme digitale, et enfin la lecture va être affichée sur un écran LCD 16 caractères 2 lignes.

Notre circuit est commande par un microcontrôleur (eprom), le PIC16F876A, dont les caractéristiques particulières nous ont aidés à faciliter les taches surtout en ce qui concerne sa programmation.

Ce travail nous a permis de traiter des problèmes d'ordre pratique et de vérifier des connaissances théoriques acquises toute le long de notre formation.

Grâce au travail continu, on a pu atteindre notre but et approfondir nos connaissances ; mais cela ne veut pas dire que notre projet est complet. Nous souhaitons que le fréquencemètre réalisé soit à la base de toute une série d'améliorations que nous n'avons pas eu la chance de les faire par manque de temps et de matériel. Les améliorations qui pourraient être ajoutées sont :

- Utilisation de quartz plus performant.
- L'utilisation d'un microcontrôleur de nouvelle génération comme le μC Atemel (atmega).

BIBLIOGRAFI

[1] Pic16f8xx.pdf.

[2] www.all data sheet .com.

[3] Bali Chahar Eddine et Abdi Hakim (réalisation d'un robot mobile avec évitement d'obstacle et trajectoire programme) mémoire de fin d'étude master université de Mohamed Khi der BISKRA spécialité : génie de systèmes industriels.

[4] PIXTIC-fréquencemètre 50MHZ -2.4GHZ pour signal radio fréquence analogue /numérique : amazon.fr.

[5] Fréquencemetre wikipedia.htm.

[6] HEF 4020B.pdf.

[7] TF2-pic16-full.pdf.

[8] www.google.dz mesure.

[9] fréquencemetre-pce-instrument.com (www.pce-instruments.com/french).

BIBLIOGRAFI

[1] Pic16f8xx.pdf.

[2] www.all data sheet .com.

[3] Bali Chahar Eddine et Abdi Hakim (réalisation d'un robot mobile avec évitement d'obstacle et trajectoire programme) mémoire de fin d'étude master université de Mohamed Khi der BISKRA spécialité : génie de systèmes industriels.

[4] PIXTIC-fréquencemètre 50MHZ -2.4GHZ pour signal radio fréquence analogue /numérique : amazon.fr.

[5] Fréquencemetre wikipedia.htm.

[6] HEF 4020B.pdf.

[7] TF2-pic16-full.pdf.

[8] www.google.dz mesure.

[9] fréquencemetre-pce-instrument.com (www.pce-instruments.com/french).

LISTE DES FIGURES

Chapitre 1 : microcontrôleur type pic 16f8xx

Chapitre 2 : compteur 4020

Chapitre3 : fréquencemètre multi calibre

Dédicace

Je ne saurais comment commencer cette dédicace sans heurter la sensibilité de certaines
personnes mais que tous ceux qui ne se reconnaîtront pas sachent que je les porte toujours
dans mon cœur.

Chaleureusement je dédie ce modeste travail :

***A** mes très chers parents en signe de ma profonde et affectueuse reconnaissance pour leur amour sans mesure, tous les sacrifices, les soutiens, les tolérances et les encouragements qu'ils ont bien voulu consentir pour moi. Tous les mots restent faibles pour leur exprimer mes sentiments et qu'ils acceptent seulement ces lignes en guise de témoignage à qui je souhaite la bonne santé et que dieu me les garde.*

***A** mes sœurs.*

***A** tous mes amis sans exception.*

***A** toute la promotion de master instrumentation électronique (2014_2015)*

***A**insi tous les étudiants de l'Université de Tlemcen.*

Khadidja

RESUME (bilingue)

Le but dans ce mémoire consiste à l'étude et la réalisation d'un fréquencemètre multi calibre programmé .Notre fréquencemètre est basé essentiellement sur un microcontrôleur PIC16F876A programmé en utilisant le logiciel « Micro Pascal ». Et un compteur BC4020. Les impulsions émises sont transformées en signaux analogiques puis digitaux pour être affichées sur écran LCD.

Mots-clés : fréquence, pic, compteur, LCD.

SUMMARY (bilingual)

The goal in this memory consists under investigation and the realization of a frequency meter multi gauges programmed. Our frequency meter is based primarily on a microcontroller PIC16F876A programmed by using the software "Micro Pascal". And a meter BC4020. The emitted impulses are transformed into digital analog signals then to be displayed on screen LCD.

Keywords: frequency, peak, meter, LCD.

ملخص

الهدف من هذه المدكرة هو دراسة تطبيقية لتحقيق جهاز الترددات متعدد المعايير.

يعتمد جهازنا هذا بالخصوص على ميكرو كونترولر مبرمج باستخدام البرنامج ميكرو باسكال و عداد .النبضات المبعوثة تتحول الى اشارات تناظرية تم الى رقمية لعرضها على شاشة الكريستال السائل ل س د.

كلمات مفتاحية . تردد- بيك - عداد - ل س د .

Dédicace

GRACE A DIEU TOU PUISSONS J4AI DUE Réalisée mon rêve

Que je dédie à la mémoire de mes chers parents Ahmed et Yamina que dieu

les gardes dans son paradis.

A mes sœurs Souad et Salima et Tema

A mes frères Djilali et Hicham et Oussama

A mes nièces nour el iman, Anfal, Bouchera, Raimasse, Malek, Djihane, Inass,

Les 2 Amina et Meryem.

A mes neveux Ahmed et Akeram et Acheraf et Ahmed.

A mes ami(e) de l'instrumentation électronique <<2014-2015>>

Khadija 'a, khadidja_k, khadidja_s, Fatima, Dalila

A tousse qui mon aide.

AICHA BENYETTOU

I/1- Introduction :

Le microcontrôleur est un objet technique, intégrant de l'électronique, fait souvent apparaître des fonctions ayant pour rôle le traitement d'information : opérations arithmétiques (Addition, multiplication…) ou logiques (ET, OU…) entre plusieurs signaux d'entrée permettant de générer des signaux de sortie.

Ces fonctions peuvent être réalisées par des circuits analogiques ou logiques. Mais, lorsque l'objet technique devient complexe, et qu'il est alors nécessaire de réaliser un ensemble important de traitements d'informations, il devient plus simple de faire appel à une structure à base de microcontrôleur.

I/2- Définition de PIC :

Les microcontrôleurs sont aujourd'hui implantés dans la plupart des applications grand public ou professionnelles, il en existe plusieurs familles. La société Américaine Micro chip Technologie a mis au point dans les années 90 un microcontrôleur CMOS : le PIC (Periphirol Interface contrôler). Ce composant encore très utilisé à l'heure actuelle, est un compromis entre simplicité d'emploi, rapidité et prix.

Les PIC existent dans plusieurs versions :

- Les UVPROM qui sont effaçables par une source de rayonnement ultraviolet
- Les OTPROM programmables une seule fois
- Les EEPROM et flash EPROM qui sont effaçables électriquement.

I/3- Classification des Pics de Micro chip :

Actuellement les modèles micro chip sont classés en trois grandes familles, comportant chacune plusieurs références. Ces familles sont :

- Base –line : les instructions sont codées sur 12 bits.
- Mide –line : les instructions sont codées sur 14 bits.
- High –line : les instructions sont codées sur 16 bits.

Les Pics sont des composants statiques, Ils peuvent fonctionner avec des fréquences d'horloge allant du continu jusqu'à une fréquence maximale spécifique à chaque circuit. Dans notre application, nous avons choisi d'utiliser le PIC 16F876A qui contient un espace mémoire plus large que les autres Pics et qui est disponible dans le marché et très utilisé. [1]

I/4- Identification des pics:

Un PIC est généralement identifié par une référence de la forme suivante : xx(L) XXCFYY-ZZ [1].

xx : famille du composant, actuellement « 12, 14, 16,17 et 18 ».

L : tolérance plus importante de la plage de tension.

XX : type de programme.

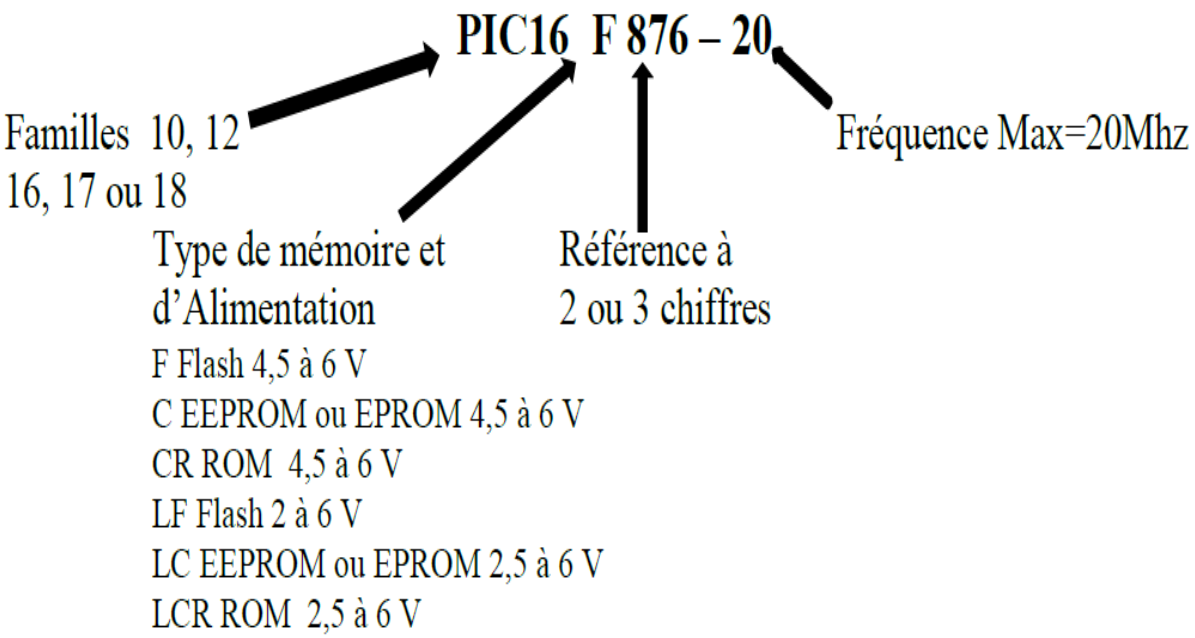
C : EPROM ou EEPROM.

F : flash.

YY : identificateur.

ZZ : vitesse maximale du quartz de pilotage.

Exemple :



I/5-Description :

- Consommation : moins de 2mA sous 5V à 4 MHz.
- Architecture RISC : 35 instructions de durée 1 ou 2 cycles.
- Durée du cycle : Période de l'oscillateur quartz divisée par 4 soit 200 ns pour un quartz de 20 MHz.
- Deux bus distincts pour le code programme et les data.

- Code instruction : mot de 14 bits et compteur programme (PC) sur 13 bits, ce qui permet d'adresser 8 K mots (de h'0000' à h'1FFF')
- Bus DATA sur 8 bits.
- 33 Ports Entrée-Sortie bidirectionnels pouvant produire 25 mA par sortie.

PORTA = 6 bits et PORTB PORTC et PORTD = 8bits PORTE = 3 bits pour le 16F877 et 22 I/O seulement pour le 16F876.

- **4 sources d'interruption :**
- Externe par la broche partagée avec le Port B : PB0
- Par changement d'état des bits du Port B: PB4 PB5 PB6 ou PB7
- Par un périphérique intégré dans le chip: écriture de Data en EEPROMterminée, conversion analogique terminée, réception USART ou I2C.
- Par débordement du Timer.

- 2 Compteurs 8 bits et 1 compteur 16 bits avec pré diviseur programmable.
- Convertisseur analogique 10 bits à 8 entrées pour le 16F877 et 4 entrées pour le 16F876.
- UART pour transmission série synchrone ou asynchrone.
- Interface I2C.

- 2 modules pour PWM avec une résolution de 10 bits.
- Interface avec un autre micro: 8 bits + 3 bits de contrôle pour R/W et CS.
- 368 Octets de RAM
- 256 Octets d'EEPROM Data.
- 8K mots de 14 bits en EEPROM Flash pour le programme (h'000' à h'1FFF').
- 1 registre de travail : W et un registre fichier : F permettant d'accéder à la RAM ou aux registres internes du PIC. Tous les deux sont des registres 8 bits.

PORTA : 6 entrées -sorties. 5 entrées du CAN. Entrée CLK du Timer 0.
PORTB : 8 entrées-sorties. 1 entrée interruption ext. Clk et Data pour prog.
PORTC : 8 entrées-sorties. Clk Timer1 et PWM1. USART. I2C.
PORTD : 8 entrées-sorties. Port interface micro-processeur (8 bits data). } N'existe pas sur le 16F876 (28broches)
PORTE : 3 entrées-sorties. 3 bits de contrôle interf micro. 3 entrées du CAN.

I/6- Organisation de l'espace programme :

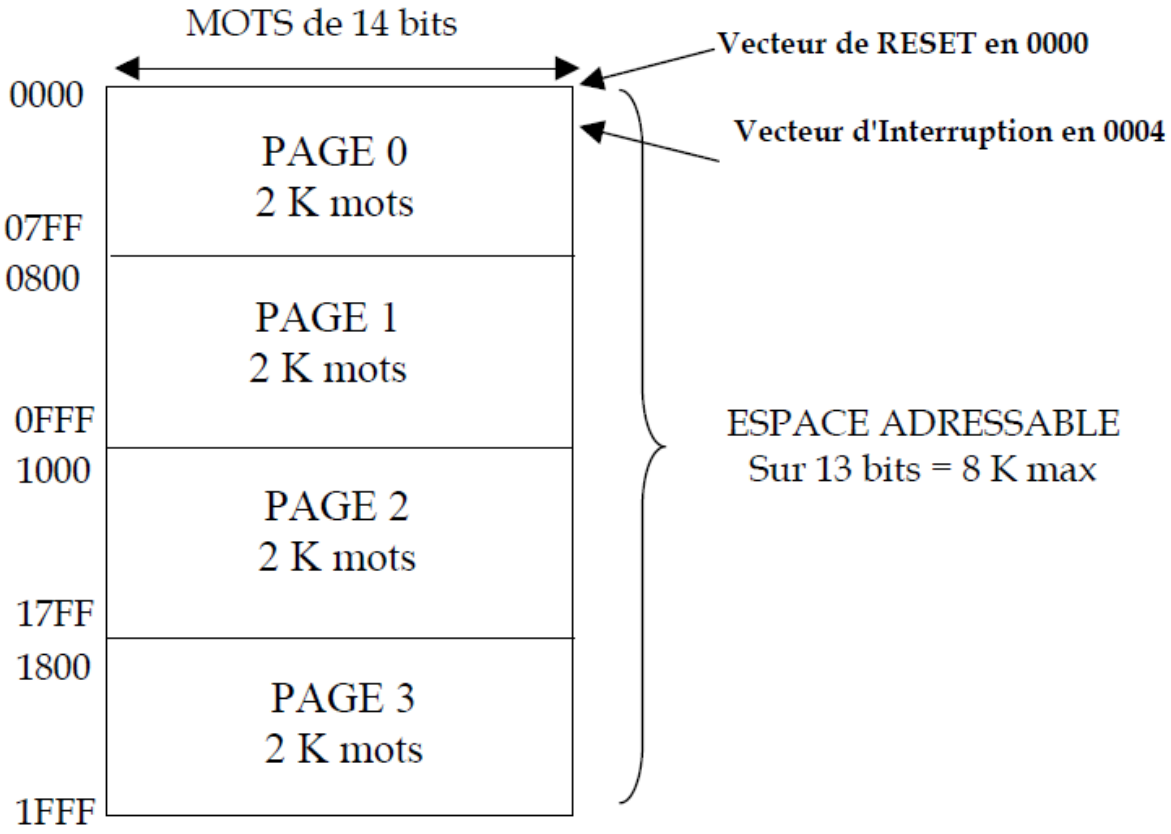


Figure I. 1- Schéma descriptif d'espace programme.

Les 2 bits MSB des 13 bits d'adresse (bits 11 et 12), viennent du registre.
PCLATH (bits 3 et 4) qui est à l'adresse : h'0A'.
Il faut impérativement les positionner pour la bonne page, avant d'utiliser les instructions:
CALL et GOTO.

I/6.1-Brochage : 16F876 (28 broches étroit) :

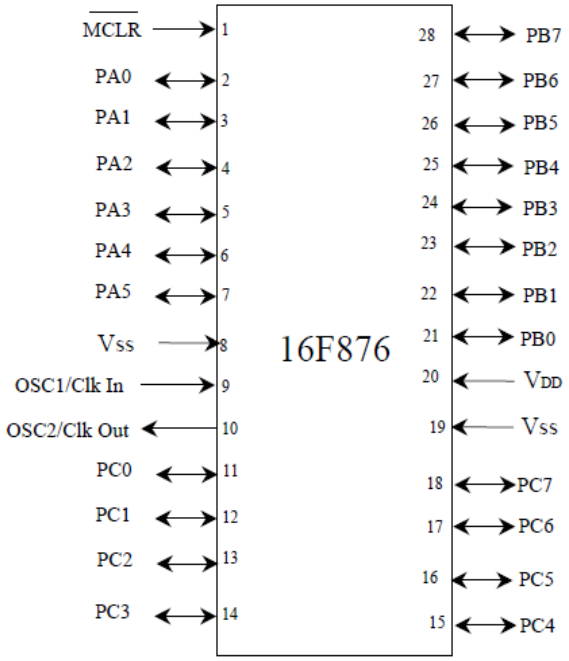


Figure I. 2- Brochage du pic 16F876.

I/6.2-Brochage : 16F877 (40 broches) :

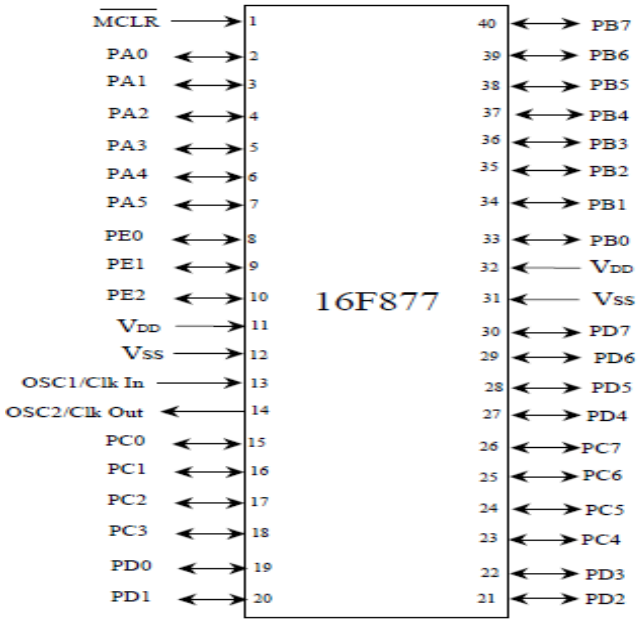
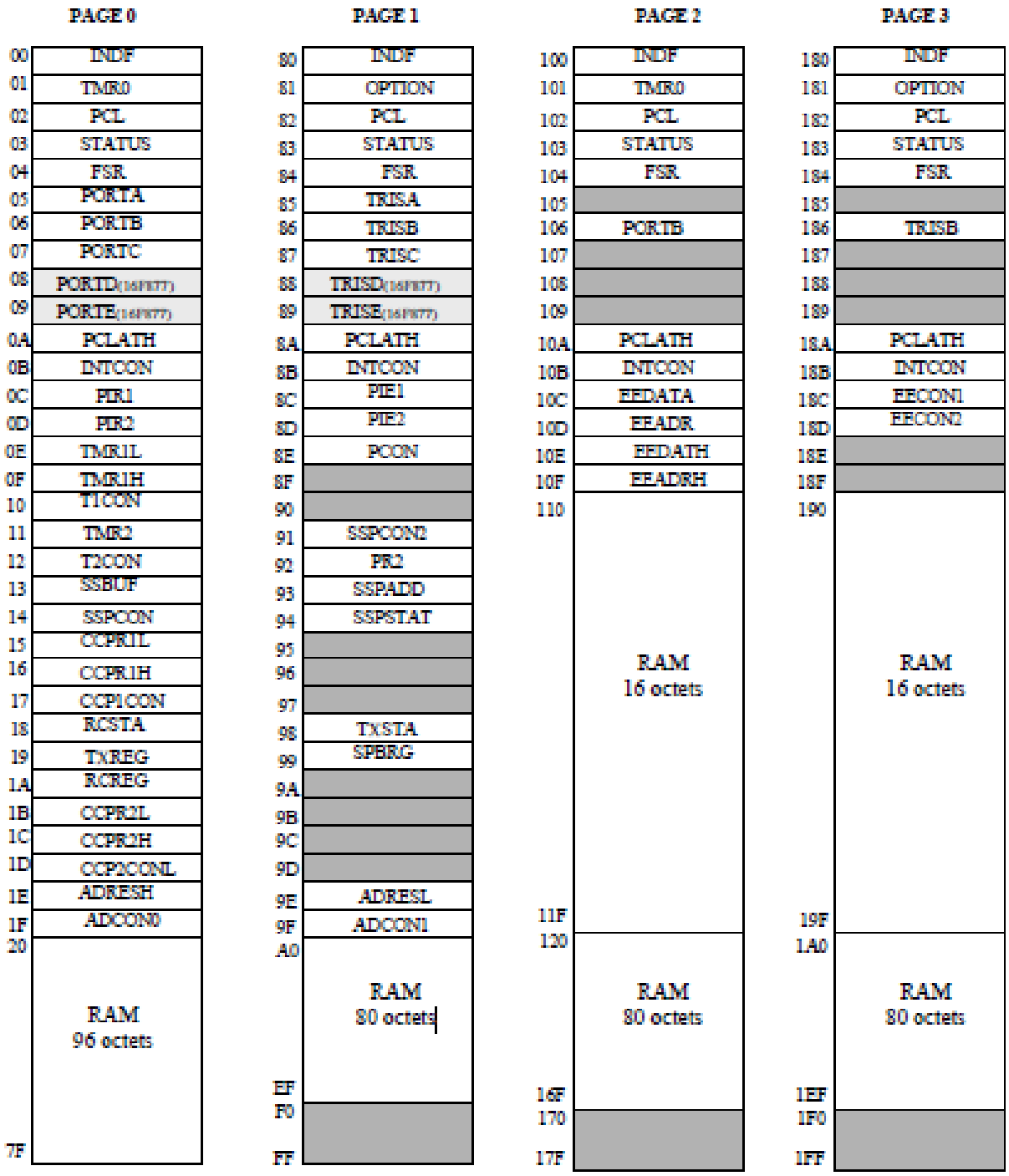


Figure I. 3-brochage pic16F877.



I/6.3-Registres internes:

I/6.3/1-Status register: (h'03' ou h'83' ou h'103 ou h'183') :

On accède indifféremment à ce registre par une quelconque de ces 4 adresses

Bit 7 Bit 0

IRP	RP1	RP0	$\overline{TO}$	$\overline{PD}$	Z	DC	C

Au reset : STATUS = 00011XXX

Bit 7 : **IRP** = permet la sélection des pages en adressage indirect.

Pour la PAGE 0 (de 00 à 7F) et la PAGE 1 (de 80 à FF) ce bit doit être laissé à "0". Mis à "1" il permettra d'atteindre la PAGE 3 (de 100 à17F) et la PAGE 4 (de 180 à 1FF).

Bits 6 et 5 : **RP1 et RP0** = permettent la sélection des pages en adressage direct.

RP1	RP0	Page sélectée
0	0	PAGE 0 de 00 à 7F
0	1	PAGE 1 de 80 à FF
1	0	PAGE 2 de 100 à 17F
1	1	PAGE 3 de 180 à 1FF

Exemple:

```
PAGE0   BCF STATUS, 5 ; RP0=0
        BCF STATUS, 6; RP1=0

PAGE1   BSF STATUS, 5; RP0=1
        BCF STATUS, 6; RP1=0

PAGE2   BCF STATUS, 5; RP0=0
        BSF STATUS, 6; RP1=1

PAGE3   BSF STATUS, 5; RP0=1
        BSF STATUS, 6 ; RP1=1
```

A dressage direct:

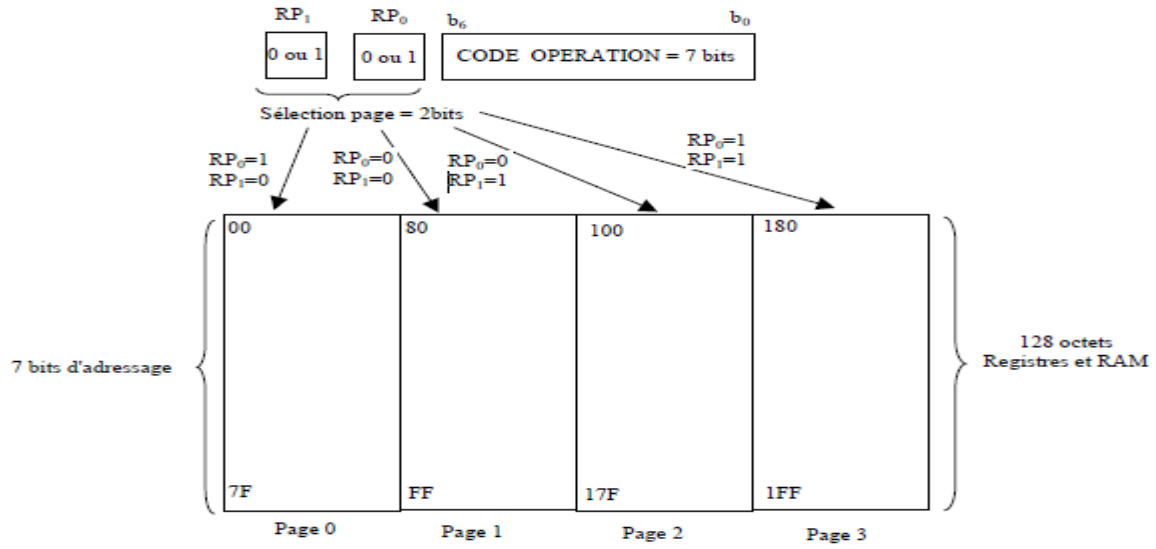


Figure I. 4- Schéma descriptif d'adressage direct.

A dressage indirect :

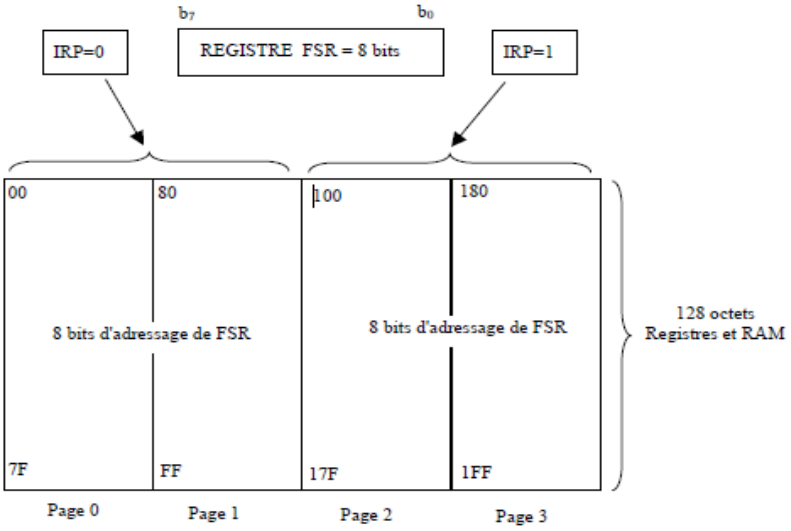


Figure I. 5 Schéma descriptif d'adressage indirect.

Bit 4: **TO** = Time Out bit.

Bit en lecture seulement.

1 = Après une mise sous tension, après une RAZ du Watch dog (CLRWDT) ou bien après l'instruction SLEEP.

0 = Signifie qu'un Time Out du timer de Watch dog est survenu.

Bit 3: **PD** = Power Down bit.

1 = Après une mise sous tension ou bien après une RAZ du Watch dog.

0 = Après l'instruction SLEEP.

Bit 2 : **Z** = Zéro bit.

1 = Le résultat d'une opération arithmétique ou logique est zéro.

0 = Le résultat d'une opération arithmétique ou logique est différent de zéro.

Bit 1: **DC** = Digit Carry bit.

1 = Une retenue sur le 4eme bit des poids faible est survenue après les instructions : ADDWF et ADDLW.

0= Carry bit = Pas de retenue sur le 4eme bit des poids faible.

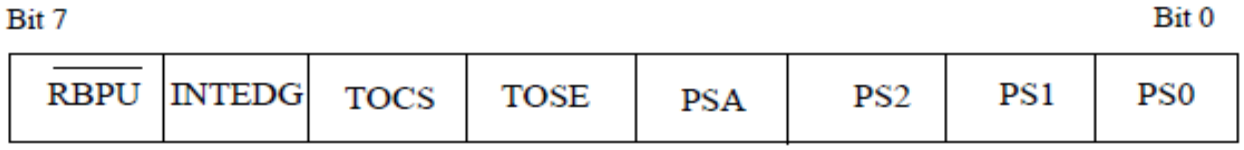
Bit 0 : **C**.

1 = Une retenue sur le bit MSB est survenue après les instructions ADDWF et ADDLW.

0 = Pas de retenue sur le bit MSB.

I/6.3/2-Option register: (h'81' ou h'181'):

Ce registre en lecture écriture permet de configurer les prés diviseurs du Timer et du Watch dog, la source du Timer, le front des interruptions et le choix du Pull up sur le Port B...



Bit 7: **RBPU** = Pull up Enable bit on Port B.

1 = Pull up désactivé sur le Port B.

0 = Pull up active.

Bit 6: **INTEDG** = Interrupt Edge select bit.

1 = Interruption si front montant sur la broche PB0/IRQ (pin 6).

0 = Interruption si front descendant Au reset : OPTION = 11111111 sur PB0/IRQ.

Bit 5: **TOCS** = Timer TMR0 Clock Source select bit.

1 = L'horloge du Timer est l'entrée PA4/Clk (pin 3).

0 = Le Timer utilise l'horloge interne du PIC.

Bit 4: **TOSE** = Timer TMR0 Source Edge select bit.

1 = Le Timer s'incrémente à chaque front montant de la broche PA4/Clk.

0 = Le Timer s'incrémente à chaque front descendant de la broche PA4/Clk.

Bit 3 : **PSA** = Prescaler Assignement bit.

1 = Le pré diviseur est affecté au Watch dog…

0 = Le pré diviseur est affecté au Timer TMR0.

Bits 2 à 0: **PS2 PS1 PS0** = Prescaler Rate Select bits.

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

Quand le pré_ diviseur est affecté au Watch dog (PSA=1), TMR0 est pré divisé par 1.

PCL REGISTER : (h'02' ou h'82' ou h'102' ou h'182').

PCLATH REGISTER: (h'0A' ou h'8A ou h'10A' ou h'18A').

Le compteur de programme est sur 13 bits. Les 8 bits de poids faible sont dans le registre PCL qui est en lecture/écriture.

Les 5 bits de poids forts ne sont pas lisibles mais on peut les écrire indirectement à travers le registre.

PCLATH.

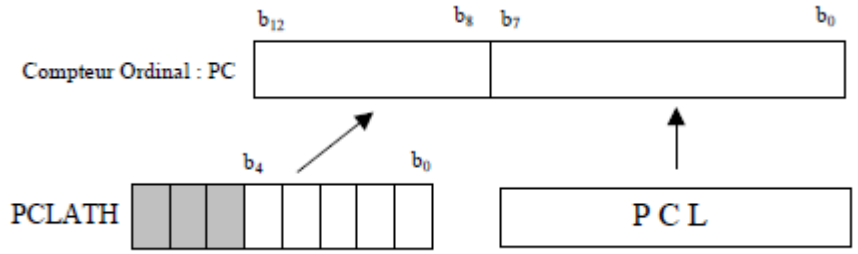


Figure I. 6-case mémoires de PCL.

I/6.3/3-Intcon register : (h'0B' ou h'8B' ou h'10B' ou h'18B') :

Ce registre en lecture écriture permet de configurer les différentes sources

Bit 7 Bit 0

GIE	PEIE	TOIE	INTE	RBIE	TOIF	INTF	RBIF

Au reset : INTCON = 0000000X

Bit 7 : **GIE** = Global Interrup Enable bit

1 = Autorise toutes les interruptions non masquées.

0 = Désactive toutes les interruptions.

Bit 6 : **PEIE** = Peripheral Interrupt Enable bit.

1 = Autorise les interruptions causées par les périphériques.

0 = Désactive les interruptions causées par le périphériques.

Bit 5*:* **TOIE** = Timer TMR0 Overflow Interrup Enable bit.

1 = Autorise les interruptions du Timer TMR0.

0 = Désactive les interruptions du Timer TMR0.

Bit 4 : **INTE** = RB0/Int Interrup Enable bit.

1 = Autorise les interruptions sur la broche : PB0/IRQ (pin6).

0 = Désactive les interruptions sur la broche : PB0/IRQ (pin6).

Bit 3 : **RBIE** = RB Port Change Interrup Enable bit.

1 = Autorise les interruptions par changement d'état du Port B (PB4 à PB7).

0 = Désactive les interruptions par changement d'état du Port B (PB4 à PB7).

Bit 2*:* **TOIF** = Timer TMR0 Overflow Interrup Flag bit.

1 = Le Timer a débordé. Ce flag doit être remis à zéro par programme.

0 = Le Timer n'a pas débordé.

Bit 1*:* **INTF** = RB0/Int Interrup Flag bit.

1 = Une interruption sur la broche PB0/IRQ (pin 6) est survenue.

0 = Pas d'interruption sur la broche PB0/IRQ (pin 6).

Bit 0 : **RBIF** = RB Port Change Interrup Flag bit. Ce flag doit être remis à zéro par programme.

1 = Quand au moins une entrée du port B (de PB4 à PB7) a changé d'état.

0 = Aucune entrée de PB4 à PB7 n'a changé d'état.

I/6.3/4-PIE1 register: (h'8C': page 1):

Ce registre contient les bits individuels d'autorisation pour les Interruptions des périphériques. Le bit 6 d'INTCON (PEIE) doit être mis à "1" pour autoriser une quelconque IT de périphérique.

Bit 7 Bit 0

PSPIE	ADIE	RCIE	TXIE	SSPIE	CCP1IE	TMR2IE	TMR1IE

Au reset : PIE1 = 00000000

Bit 7 : **PSPIE** = Parallèle Slave Port Interrup Enable bit.

Ce bit n'existe pas pour un PIC 16F876 (28 pins). Toujours garder ce bit à "0".

1 = Autorise les interruptions R/W du port SSP.

0 = Désactive toutes ces interruptions.

Bit 6 : **ADIE** = A/D converter Interrup Enable bit.

1 = Autorise les interruptions du convertisseur analogique/digital.

0 = Désactive cette interruption.

Bit 5: **RCIE** = USART Receive Interrup Enable bit .

1 = Autorise les interruptions en réception de l'USART.

0 = Désactive cette interruption.

Bit 4: **TXIE** = USART Transmit Interrup Enable bit .

1 = Autorise les interruptions en émission de l'USART.

0 = Désactive cette interruption.

Bit 3 : **SSPIE** = Synchro nous Serial port Interrup Enable bit.

1 = Autorise les interruptions du module Synchrone (I2C).

0 = Désactive cette interruption.

Bit 2: **CCP1IE** = CCP1 Interrup Enable bit .

1 = Autorise les interruptions du CCP1.

0 = Désactive cette interruption.

Bit 1: **TMR2IE** = TMR2 Interrup Enable bit.

1 = Autorise les interruptions du Timer 2 TMR2.

0 = Désactive cette interruption.

Bit 0: **TMR1IE** = TMR1 over flow Interrup Enable bit.

1 = Autorise les interruptions de débordement du Timer 1 TMR1.

0 = Désactive cette interruption.

Bit 7 Bit 0

PSPIE ADIE RCIE TXIE SSPIE CCP1IE TMR2IE TMR1IE.

I/6.3/5-PIR1 register: (h'0C': page 0):

Ce registre contient les FLAG associés aux interruptions des périphériques.

Bit 7 Bit 0

PSPIF	ADIF	RCIF	TXIF	SSPIF	CCP1IF	TMR2IF	TMR1IF

Ces Flag passent à "1" quand une IT correspondante survient le bit et que d'autorisation est bien positionnée. Ces Flag doivent être remis à "0" par Soft.

Au reset : PIR1 = 00000000

Bit 7 : **PSPIE** = Parallèle Slave Port Interrup Flag bit.

1 = Une opération de R/W vient d'avoir lieu sur le port SSP.

0 = Il n'y a pas eu de R/W sur le port SSP.

Bit 6 : **ADIF** = A/D converter Interrup Flag bit.

1 = Une conversion A/D est terminée.

0 = la conversion A/D n'est pas terminée.

Bit 5: **RCIF** = USART Receive Interrup Flag bit .

1 = Le buffer de réception de l'USART est plein (donnée reçue).

0 = Le buffer de réception de l'USART est vide (rien de reçu).

Bit 4 : **TXIF** = USART Transmit Interrup Flag bit.

1 = Le buffer de transmission de l'USART est vide (on peut le remplir).

0 = Le buffer de réception de l'USART est plein (on ne peut pas le charger).

Bit 3 : **SSPIF** = Synchro nous Serial Port Interrup Flag bit.

1 = Une condition d'IT du module SSP est apparue.

0 = Aucune condition d'IT n'est apparue.

Bit 2: **CCP1IF** = CCP1 Interrup Flag bit.

1 = Une condition de Capture ou de Compare du Timer1 a fait une IT.

0 = Pas d'IT de capture ou de Compare du TIMER 1.

Bit 1: **TMR2IF** = TMR2 Interrup Flag bit.

1 = Le Timer2 a fait une IT.

0 = Pas d'IT du TIMER 2.

Bit: **TMR1IF** = TMR1 Overflow Interrup Flag bit.

1 = Le débordement Timer 1 a fait une IT.

0 = Pas de débordement du TIMER 2.

I/7- Ports entrée / sortie :

I/7.1-PORTA (h05) et TRISA (h85) :

Ce port bidirectionnel est constitué de 6 bits. Le registre de direction correspondant est TRISA.

Quand on écrit un "1" dans TRISA, le bit correspondant du PORTA est configuré en ENTREE, et le driver de sortie est placé en haute impédance.

Si on écrit un "0", le port devient une SORTIE, et le contenu du latch correspondant est chargé sur la broche sélectionnée.

Le bit 4 du Port peut également servir pour l'entrée horloge du timer TMR0.

Les autres bits du Port sont partagés avec le CAN.

N.B: **Après un reset le Port A est configuré en CAN. Il faut impérativement le configurer en I/O digitale pour l'utiliser comme tel. Il faut pour cela accéder au registre ADCON1 en h'9F'.**

Ce registre sera étudié dans le chapitre concernant le CAN.

On retiendra seulement que pour configurer les 5 bits du Port A en I/O digitales, il faut positionner les 4 bits PCFG à 0110.

Exemple :

```
 ADCON1 EQU h'9F'
PAGE1
MOVLW h'06'
MOVWF ADCON1
```

N.B: **Le Port PA4 qui est partagé avec l'entrée horloge du Timer 0 est un Drain ouvert. Il faut donc le relier au +Vcc par une résistance de 10 KΩ pour l'utiliser en tant que sortie.**

I/7.2- PortB (h06) et TrisB (h86) :

Il comporte 8 bits. Le registre de direction correspondant est TRISB.Si on écrit un "1" dans le registre TRISB, le driver de sortie correspondant passe en haute impédance. Si on écrit un "0", le contenu du Latch de sortie correspondant est recopié sur la broche de sortie.

Chaque broche du PORT B est munie d'un tirage au +VDD que l'on peut mettre ou non en service en mode entrée uniquement. On active cette fonction par la mise à "0" du bit 7 dans le registre OPTION en h'81'.
Au reset, le tirage est désactivé.
Il est inactif quand le port est configuré en sortie.
Les 4 broches PB7 PB6 PB5 et PB4 provoquent une interruption sur un changement d'état si elles sont configurées en ENTREE.
On doit remettre à zéro le Flag de cette interruption (bit 0 du registre INTCON en h'0B') dans le programme d'interruption.
Cette possibilité d'interruption sur un changement d'état associé à la fonction de tirage configurable sur ces 4 broches, permet l'interfaçage facile avec un clavier. Cela rend possible le réveil du PIC en mode SLEEP par un appui sur une touche du clavier.
Le bit 0 du PORT B peut également être utilisé comme entrée d'interruption externe. Le choix du front de déclenchement se fait en configurant le bit 6 du registre OPTION.
Au RESET: PORT A et PORT B configurés en ENTREE (TRISA et TRISB = 1) PORT B : Tirage désactivé.

I/7.3-PortC (h07) et TRISC (h87) :

Il s'agit d'un PORT 8 bits bidirectionnel.
Il est partagé avec le module de transmission synchrone I2C et l'USART.

I/7.3/1-USART ou SCI :

Elle utilise les pins 6 et 7 du PORT C: **PC6 = Tx DATA** et **PC7 = Rx DATA**
Les 5 registres utilisés sont :
Registre Emission : **TXREG** en h'19' page 0.
Registre Réception: **RCREG** en h'1A' page 0.
Registre d'état Emission : **TXSTA** en h'98' page 1.
Registre d'état Réception : **RCSTA** en h'18' page 0.
Registre du choix de la vitesse : **SPBRG** en h'99 page 1.

I/7.3/2-TXSTA register: (h'98': page 1):

Bit 7 Bit 0

CSRC	TX9	TXEN	SYNC		BRGH	TRMT	TX9D

Au reset : TXSTA = 00000010

Bit 7 : **CSRC**= Clock Source en synchrone. Sans importance en asynchrone.

Bit 6 : **TX9** = Autorisation d'émission sur 9 bits.

1 = Autorisé.

0 = Non autorisé.

Bit 5 : **TXEN** = Autorisation d'émission.

1 = Autorisé.

0 = Non autorisé.

Bit 4 : **SYNC** = Sélection mode Synchrone / Asynchrone.

1 = Mode synchrone.

0 = Mode asynchrone.

Bit 3 : **Non implémenté**

Bit 2 : **BRGH** = Sélection vitesse rapide en mode asynchrone.

1 = Vitesse haute sélectionnée.

0 = Vitesse basse sélectionnée.

Bit 1 : **TRMT** = bit d'état du registre à décalage Emission.

1 = Registre vide, donc émission terminée.

0 = Registre plein, donc émission en cours.

Bit 0 : **TX9D** = 9eme bit de Data transmise.

Ce bit peut être le bit de la parité.

I/7.3/3-SPBRG register : (h'99' : page 1).

Le Baud Rate Générateur est un registre 8 bits qui contient le facteur de division (N) de l'horloge interne qui permet d'obtenir la vitesse commune d'émission et de réception. En mode Asynchrone (bit SYNC = 0) suivant l'état du bit BRGH on aura le choix entre 2 vitesses : haute pour BRGH=1 et basse pour BRGH=0.

BRGH=0 VITESSES BASSES	BRGH=1 VITESSES HAUTES
$VITESSE = \frac{F_{oscill}}{64(N+1)}$	$VITESSE = \frac{F_{oscill}}{16(N+1)}$
$N = \frac{F_{oscill}}{64.Vitesse} - 1$	$N = \frac{F_{oscill}}{16.Vitesse} - 1$

Le nombre N est le nombre entier, arrondi de la valeur trouvée par les équations ci-dessus.

Il est recommandé d'utiliser si possible les vitesses hautes (BRGH=1), même pour des vitesses faibles, car dans ce cas on minimise l'erreur, en obtenant un nombre N plus grand.

Valeurs de N pour diverses vitesses avec un Quartz de 8 MHz :

VITESSES en Bits/sec ou BAUDS	VITESSES BASSES BRGH = 0	VITESSES HAUTES BRGH = 1
119200	x	4
57600	x	8
38400	x	12
19200	5	25
9600	12	51
4800	25	103
2400	51	207
1200	103	x

I/7.3/4-RCSTA register : (h'18' : page 0) :

Bit 7 Bit 0

SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

Au reset : RCSTA = 0000000X

Bit 7 : **SPEN**= Serial Port Enable. PC7 et PC6 configurés pour le port série.

1 = Port série en service.

0 = Port série désactivé.

Bit 6 : **RX9** = Autorisation de réception sur 9 bits.

1 = Autorisé.

0 = Non autorisé.

Bit 5: **SREN** = Single Receive Enable. Réservé pour mode Synchrone.

Non utilisé en mode Asynchrone.

Bit 4: **CREN** = Continous Receive Enable.

1 = Autorise la réception en continu.

0 = Désactive la réception en continu.

Bit 3: **ADDEN** = Adress Detect Enable. En mode Asynchrone 9 bits :

1 = Autorise la détection d'adresse, et charge la Data dans le registre de réception RCREG quand le 9eme bit du registre de dé sérialisation vaut "1".

0 = Déselecte la détection d'adresse. Tous les octets sont reçus et le 9eme bit peut servir de bit de parité.

Bit 2 : **FERR** = Framing Error.

1 = Une erreur de Framing est survenue.

0 = Pas' d'erreur de Fra ming.

Bit 1 : **OERR** = Overrun Error.

Un octet est reçu alors que le registre de réception n'a pas été vidé par lecture.

1 = Erreur Overrun.

0 = Pas d'erreur Overrun.

Bit 0 : **RX9D** = 9eme bit de Data reçue.

Ce bit peut être le bit de la parité.

EMISSION: Pin PC6 = TX DATA.

L'émission est autorisée par la mise à "1" du bit 5 de TXSTA: TXEN = 1.

La DATA à transmettre est mise dans le registre TXREG en h'19' page 0. Ce registre prévient qu'il est vide en mettant le flag TXIF à "1" (bit 4 de PIR1).
Ce flag passe à "0" dès que l'on charge un octet dans le registre TXREG. Il repasse à "1" par Hard quand le registre est vidé par transfert dans le registre de sérialisation : TSR.
Ce registre n'est pas accessible par l'utilisateur, il n'a pas d'adresse.
Si on charge alors un 2eme octet dans le registre TXREG le flag TXIF va passer à "0" et y rester tant que le registre TSR n'aura pas complètement sérialisé l'octet précédent à transmettre. Dès que le STOP de l'octet précédent a été transmis, le registre TXREG est transféré dans TSR et le flag TXIF repasse à"1" signalant ainsi que le registre de transmission TXREG est vide et peut donc recevoir un nouvel octet à transmettre.
Le bit TRMT (bit 1 de TXSTA) informe sur l'état du registre TSR. Quand le registre TSR n'a pas fini de sérialisé, TRMT=0. Ce flag repasse à "1" quand le registre est vide, c'est à dire quand le stop a été émis.
Le flag TXIF permet aussi de générer une interruption, à condition qu'elle soit autorisée par mise à "1" du bit 4 de PIE1 : TXIE = 1.
Il faut dans ce cas autoriser les interruptions des périphériques par mise à "1" du bit 6 de INTCON: PEIE = 1, et par la mise à "1" du bit 7 : GIE = 1.

I/8-Procedure pour émettre :

- Initialiser SPBRG pour la vitesse désirée et choix pour BRGH.
- Autoriser mode Asynchrone : SYN = 0 et SPEN = 1.
- Eventuellement faire TX9 = 1 si une émission sur 9 bits est désirée.
- Autoriser l'émission par TXEN = 1.
- Si une transmission 9 bits a été choisie, mettre le 9eme bit dans TX9D.
- Mettre l'octet à transmettre dans TXREG.
- Avant de remettre l'octet suivant à transmettre dans TXREG, il faut tester le flag TXIF qui est à "0" si le registre n'est pas disponible. Dès que le registre est vide ce flag passe à "1" et on peut alors charger TXREG par l'octet à transmettre.
- Pour savoir si le dernier octet a été émis, il suffit de tester le flag TRMT qui signale par son passage à "1" que le dernier bit du dernier octet et son STOP ont bien été sérialisés.
- On peut alors stopper le module émission de l'USART par TXEN=0.

I/9-Reception Pin PC7 = RX DATA:

La réception est autorisée par la mise à "1" du bit 4 de RCSTA: CREN = 1.
La DATA reçue est mise dans le registre RCREG en h'1A' page 0. Ce registre prévient qu'il est plein en mettant le flag RCIF à "1" (bit 5 de PIR1). On peut autoriser la génération d'une interruption quand RCIF = 1, c'est à dire quand une donnée valide est disponible dans RCREG par mise à "1" du bit 5de PIE1 : RCIE = 1. Le flag RCIF repasse à "0" par hard quand on vide le registre RCREG par sa lecture.
Si le STOP d'un 2eme octet survient alors que le registre RCREG n'a pas été vidé, une erreur OVERRUN se produit. Elle est signalée par le passage à "1"du bit 1 de RCSTA : OERR=1. L'octet dans le registre de dé sérialisation est alors perdu.
Le bit d'erreur OERR doit être remis à zéro par soft. Pour cela il faut stopper la réception par CREN=0 puis remettre en service la réception par CREN=1.
En fait le registre RCREG est un double registre FIFO. On peut donc recevoir2 octets et ne pas les lire avant qu'un 3eme octet ne fasse un OVERRUN.
On doit alors lire deux fois RCREG pour le vider les 2 octets reçus qui sont dans le FIFO.
Si un STOP est trouvé à "0" alors une ERROR FRAMING est générée par mis à "1" du bit 2 de RCSTA: FERR=1.

I/10-Procedure pour recevoir :

- Initialiser SPBRG pour la vitesse désirée et choix pour BRGH.
- Autoriser mode Asynchrone : SYN = 0 et SPEN = 1.
- Eventuellement faire RX9 = 1 si une réception sur 9 bits est désirée.
- Eventuellement faire RCIE = 1 si une réception par interruption est désirée.
- Autoriser l'émission par RCEN = 1.
- Test du Flag RCIF (ou attente IT) pour savoir si un octet a été reçu.
- Lire éventuellement le 9eme bit de Data dans RCSTA pour tester la parité.
- Lire les bits FERR et OERR pour déterminer les erreurs éventuelles.
- Si une erreur est survenue il faut faire CREN=0 puis CREN=1 pour RAZ.
- Lecture du registre RCREG pour récupérer l'octet reçu.

I/10.1-Mode 9 bits avec détection d'adresse :

- Faire toutes les initialisations précédentes: SPBRG, BRGH, SYN, SPEN.
- Faire RX9=1 ADDEN=1 et CREN=1

- Dès que le registre RCREG est signalé plein, il s'agit de l'adresse (9 eme bit =1).
- Le lire et le comparer avec l'adresse déterminée.

Si OK faire ADDEN=0 pour recevoir maintenant tous les octets, même si le 9eme bit n'est pas à "1".

Si pas OK rester avec ADDEN=1 pour ne recevoir que les octets filtrés d'adresse.

I/10.2-Module MSSP pour I2C:

Le module MSSP (Master Synchro nous Serial Port) permet de faire des transmissions au format I2C (Inter Integrated Circuits) en tant que Maître ou Esclave.

Il utilise les pins 3 et 4 du PORT C: **PC3 = SCL et PC4 = SDA.**

Les 5 registres utilisés sont :

Registre Buffer **SSPBUF** en h'13' page 0.

Registre d'adresse **SSPADD** en h'93' page 1.

Registre d'état **SSPSTAT** en h'94' page 1.

Registre de contrôle **SSPCON** en h'14' page 0.

Registre de contrôle n°2 **SSPCON2** en h'91' page 1.

I/10.3-SSPSTAT register : (h'94' : page 1) :

Bit 7 Bit 0

SMP	CKE	D/$\overline{A}$	P	S	R/$\overline{W}$	UA	BF

Au reset : SSPSTAT = 00000000

Bit 7 : **SMP**= bit d'échantillonnage.

Bit 6: **CKE** = Clock Edge Select. Ce bit à "0" : conforme aux spécifications I2C.

Bit 5: **D/A** = Data / Addressee bit.

1 = indique que le dernier octet reçu est une Data.

0 = indique que le dernier octet reçu est une adresse.

Bit 4 : **P** = STOP bit. Ce bit est remis à "0" quand le module SSP est désactivé.

1 = indique qu'un Stop a été détecté.

0 = indique qu'il n'a pas été détecté de Stop.

Bit 4 : **P** = STOP bit. Ce bit est remis à "0" quand le module SSP est désactivé.

1 = indique qu'un Stop a été détecté.

0 = indique qu'il n'a pas été détecté de Stop.

Bit 3 : **S** = START bit. Ce bit est remis à "0" quand le module SSP est désactivé.

1 = indique qu'un Start a été détecté.

0 = indique qu'il n'a pas été détecté de Start.

Bit 2 : **R/W** = Information sur le bit R/W. Donne la valeur de ce bit qui suit la dernière adresse. Ce bit est valable jusqu'à la réception du STOP.

- ***Mode esclave :***

1 = R.

0 = W.

- ***Mode maitre*** *:*

1 = Transmission en cours.

0 = Pas de transmission en cours.

Bit 1 : **UA** = mise à jour adresse. Utilisé en mode 10 bits seulement.

Bit 0 : **BF** = Buffer plein.

- ***Mode Réception I2C:***

1 = Registre plein : réception terminée.

0 = Registre vide : donc réception non terminée.

- **Mode Emission I2C:**

1 = Registre plein : donc émission en cours.

0 = Registre vide : donc émission terminée.

10.4-SSPCON register : (h'14' : page 0) :

Bit 7 Bit 0

WCOL	SSPOV	SSPEN	CKP	SSPM3	SSPM2	SSPM1	SSPM0

Au reset : SSPCON = 00000000

Bit 7 : **WCOL**= détection de collision à l'écriture.

1 = Collision à l'écriture.

0 = Pas de collision.

Bit 6 : **SSPOV** = Over flow indicator en réception. Ne sert pas en émission Doit être remis à "0" par soft

1 = Un octet vient d'être reçu dans le buffer qui n'était pas vidé : Over flow.

0 = Pas d'Over flow.

Bit 5: **SSPEN** = Enable Module SSP.

1 = Module I2C activé et pin SCL et SDA sur PC3 et PC4 configurées.

0 = Module I2C désactivé. PC3 et PC4 libres pour usage I/O.

Bit 4 : **CKP** = Polarité du Clock. Inutilisé en mode maître.

En mode Esclave :

1 = Autorise horloge.

0 = Maintient CLK = 0.

Bit 3 à bit 0 : **SSPM3 / SSPM0** = Select Mode.

SSPM3	SSPM2	SSPM1	SSPM0	MODE
0	1	1	0	I²C esclave 7 bits.
0	1	1	1	I²C esclave 10 bits.
1	0	0	0	I²C maître CLK= F_{osc}/4(SSPADD+1).

I/10.5-SSPCON2 register : (h'91' : page 1) :

Bit 7 Bit 0

GCEN	ACKSTAT	ACKDT	ACKEN	RCEN	PEN	RSEN	SEN

Au reset : SSPCON2 = 00000000

Bit 7 : **GCEN**= Appel général en mode I2C esclave seulement..

1 = Autorise IT quand une adresse appel général (h'0000) est reçue.

0 = Adresse appel général désactivée.

Bit 6 : **ACKSTAT** = Bit de ACK en mode I2C maître seulement.

1 = Pas de ACK reçu de l'esclave.

0 = ACK de l'esclave a été reçu.

Bit 5: **ACKDT** = ACK Data bit. En mode maître et réception : valeur qui sera transmise quand on va lancer une séquence de ACK en fin de réception.

1 = NO ACK.

0 = ACK.

Bit 4: **ACKEN** = ACK Sequence Enable. En mode maître et réception, lance une séquence d'ACK ou de NOACK suivant la valeur dans ACKDT.

1 = Lance la séquence de ACK et transmet ACKDT. Remis à "0" par hard.

0 = Pas de séquence d'ACK.

Bit 3: **RCEN** = Receive Enable Bit. En mode I2C maître seulement.

1 = Autorise mode réception.

0 = Pas de réception autorisée.

Bit 2: **PEN** = Stop Condition Enable. En mode I2C maître seulement.

1 = Lance la séquence de STOP sur SDA et SCL. Remis à "0" par hard.

0 = Pas de séquence de STOP.

Bit 1: **RSEN** = Repeated Start Condition. En mode I2C maître seulement.

1 = Séquence de répétition de START sur SDA et SCL. Remis à "0" par hard.

0 = Pas de séquence de répétition de START.

Bit 0: **SEN** = Start Condition Enable. En mode I2C maître seulement.

1 = Lance la séquence de START sur SDA et SCL. Remis à "0" par hard.

0 = Pas de séquence de START.

I/10.6-Mode maitre :

Les pins SDA et SCL sont manipulées par le Hard. Les événements qui causent le passage à "1" du flag SSPIF (bit 3 de PIR1 en h'0C') et éventuellement une IT si elles sont autorisées, sont :

- Condition de START.
- Condition de STOP.
- Répétition d'un START.
- ACK après un transfert.
- Transfert d'octet en Emission ou réception.

N.B: Le flag SSPIF doit être remis à "0" par soft.

I/10.6/1-Lancement d'un START:

- L'utilisateur doit mettre le bit SEN (de SSPCON2) à "1".
- Les pins SDA et SCL étant toutes les deux à "1", le module fait passer la pin SDA de "1" à "0" ce qui génère une condition de START.
- Le bit S (de SSPSTAT) passe à "1" pour signaler le START.

- A la fin du START, le bit SEN est remis à "0" par le hard.
- Dès que le flag SSPIF (de PIR1) passe à "1" pour signaler la fin du START, on peut charger le registre de transmission SSBUF avec l'octet à transmettre.

Ne pas oublier de remettre ce flag à "0" par le soft.

I/10.6/2-Transmission :

Dès que le registre SSBUF est chargé, le bit BF (de SSPSTAT) passe à "1"pour signaler que la transmission est en cours.

Au 8eme coup de CLK, la transmission est terminée et le bit BF repasse à "0".

I/10.6/3-Acquittement :

Quand l'esclave répond l'ACK, le bit ACKSTAT passe à "0".

La fin de l'ACK est signalée par le Flag SSPIF qui passe à "1". On doit alors remettre ce flag à "0" par soft.

I/10.6/4-Lancement d'un STOP :

L'utilisateur doit mettre le bit PEN (de SSPCON2) à "1". Le module fait alors passer SDA à "0" puis force SCL à"1". Quand SCL est à "1" il fait passer SDA de "0" à"1", ce qui génère une condition de STOP.

Le bit P (de SSPSTAT) passe à "1" pour signaler le STOP.

A la fin du STOP, le bit PEN est remis à "0" par le hard.

Le flag SSPIF (de PIR1) passe à "1" pour signaler la fin du STOP.

I/10.6/5-Réponse du maître à l'esclave :

Il s'agit soit d'un ACK soit d'un NOACK :

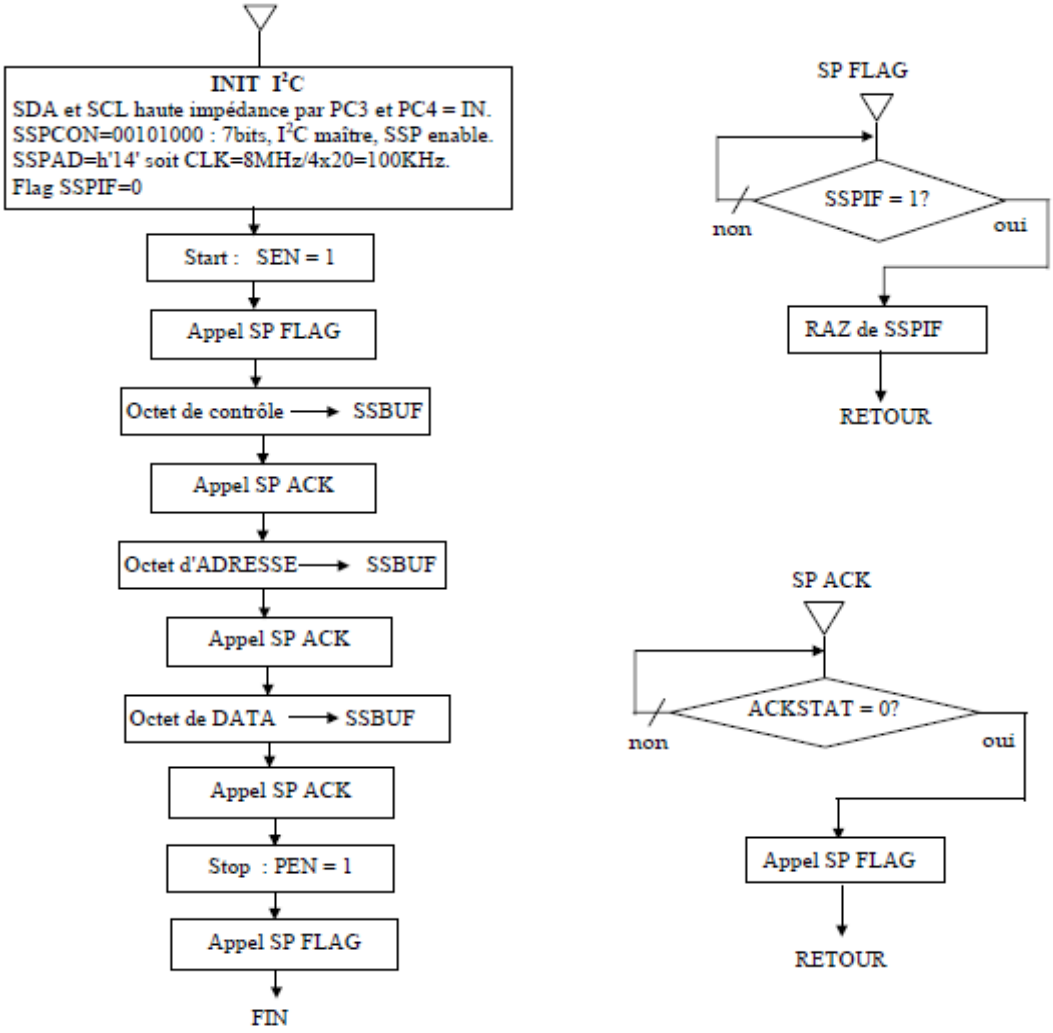
- Positionner le bit ACKDT (de SSPCON2) en fonction de la réponse à faire.
- Mettre le bit ACKEN (de SSPCON2) à "1".
- La fin de transmission de l'ACK ou du non ACK est signalée par le flag SSPIF qui passe à "1". Ne pas oublier de le remettre à "0" par soft.

I/10.6/6-Réception d'un octet par le maître :

Pour cela il faut mettre le module en réception en portant le bit RCEN à "1".

- La fin de réception est signalée par le passage à "1" du flag SSPIF, que l'on remettra à "0" par soft.
- On peut alors lire l'octet reçu dans SSBUF.

Exemple de Programme : Ecriture dans une EEPROM I2C type 24C16. [7]



I/11-Convertisseur A/D :

Il est constitué d'un module convertisseur à 5 entrées pour le boîtier 28broches (16F876) et à 8 entrées pour le boîtier 40 broches (16F877).

Les 5 premières entrées sont sur le Port A en PA0, PA1, PA2, PA3 et PA5.

Les 3 entrées supplémentaires du boîtier 40 pins sont en PE0, PE1 et PE2.

Le résultat de la conversion est codé sur 10 bits. C'est une valeur comprise entre h'000' et h'3FF'.

Les tensions de référence haute et basse peuvent être choisies par programmation parmi: VDD ou la broche PA3 pour VREF+ et VSS ou la brochePA2 pour VREF- .

Les 4 registres utilisés par le module convertisseur A/D sont :

- ADRESH en h'1E' page 0 : MSB des 10 bits du résultat.
- ADRESL en h'9E' page 1 : LSB des 10 bits du résultat.
- ADCON0 en h'1F' page 0 : registre de contrôle n°0 du convertisseur.
- ADCON1 en h'9F' page 1 : registre de contrôle n°1 du convertisseur.

I/11.1-ADCON0 : (h'1F' : page 0) :

Bit 7 Bit 0

ADSC1	ADSC0	CHS2	CHS1	CHS0	GO/$\overline{\text{Done}}$		ADON

Au reset : ADCON0 = 00000000

Bit 7 et bit 6 : **ADSC1 et ADSC0** = Clock Select bits.

Ces 2 bits permettent de choisir la vitesse de conversion :

00= Fosc/2.

01= Fosc/8.

10= Fosc/32.

11= Oscillateur RC interne.

Le temps de conversion d'un bit est TAD. Pour une conversion totale des 10bits il faut : 12.TAD.

Pour que la conversion soit correcte il faut que TAD soit au minimum de **1,6µs.**

Avec l'oscillateur interne RC on a : TAD = 4 µs typique (entre 2 et 6 µs).

Bit 5 bit4 et bit 3 : **CHS2 CHS1 et CHS0** = Channel Select bits.

Ces 3 bits permettent de choisir l'entrée qui va être convertie.

Canal	CHS2	CHS1	CHS0	PORT
0	0	0	0	PA_0
1	0	0	1	PA_1
2	0	1	0	PA_2
3	0	1	1	PA_3
4	1	0	0	PA_5
5	1	0	1	PE_0
6	1	1	0	PE_1
7	1	1	1	PE_2

Non disponible sur le 16F876 (28 pins).

Bit 2: **GO/DONE:** Status bit is ADON=1.

1 = Démarre la conversion A/D. Ce bit est remis à "0" par hard.

0 = La conversion A/D est terminée.

Bit 1 : **Bit non implanté.**

Bit 0 : **ADON :** A/D on bit.

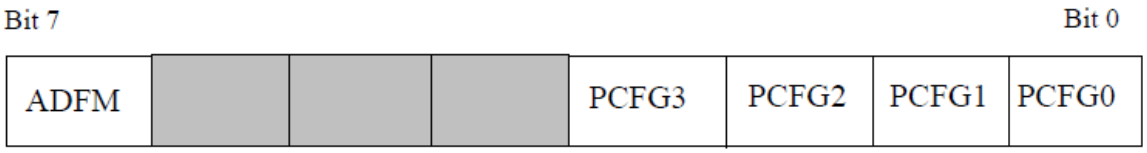
1= Convertisseur A/D en service.

0 = Convertisseur A/D à l'arrêt.

11.2-Temps de conversion TAD en fonction du Quartz et des bits du Clock select :

QUARTZ	CLOCK	T_{AD}	$12.T_{AD}$	Ne convient pas si T_{AD}<1,6µs
4 MHz	$F_{osc}/2$ = 2 MHz	0,5 µs	6 µs	Ne convient pas
	$F_{osc}/8$ = 500 KHz	2 µs	24 µs	OK
	$F_{osc}/32$ = 125 KHz	8 µs	96 µs	OK
8 MHz	$F_{osc}/2$ = 4 MHz	0,25 µs	3 µs	Ne convient pas
	$F_{osc}/8$ = 1 MHz	1 µs	12 µs	Ne convient pas
	$F_{osc}/32$ = 250 KHz	4 µs	48 µs	OK
12 MHz	$F_{osc}/2$ = 6 MHz	0,16µs	1,92 µs	Ne convient pas
	$F_{osc}/8$ = 1,5 MHz	0,66 µs	8 µs	Ne convient pas
	$F_{osc}/32$ = 375 KHz	2,6 µs	32 µs	OK
16 MHz	$F_{osc}/2$ = 8 MHz	0,125µs	1,5 µs	Ne convient pas
	$F_{osc}/8$ = 2 MHz	0,5 µs	6 µs	Ne convient pas
	$F_{osc}/32$ = 500 KHz	2 µs	24 µs	OK

11.3-ADCON1 : (h'9F' : page 1) :



Au reset : ADCON1 = 00000000

Bit 7 : **ADFM** = A/D Result format.

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

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

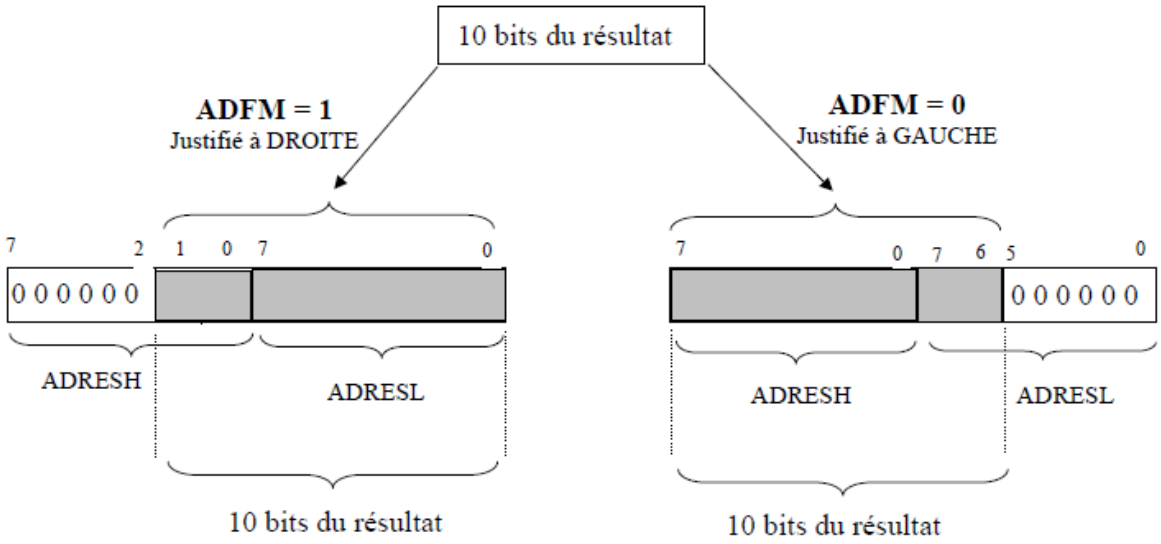


Figure I. 7-Schema descriptive ADCON1

Req : Bit 6 bit 5 et bit 4 : **Bits non implémentés.**

Bit 3 bit 2 bit 1 et bit 0 : **PCFG3 PCFG2 PCFG1 et PCFG0**

Bits de contrôle de la configuration des Ports.

Ces bits permettent de choisir le partage entre entrées analogiques et digitales sur les PORTS A et E.

Ils permettent également de choisir pour VREF+ entre VDD et PA3 et pour et VREF - entre VSS et PA2.

I/11.4-Configuration des PORTS en fonctions des 4 bits PCFG:

A = Entrée Analogique.

D = I/O Digitale.

4 Bits PCFG	N°7 PE_2	N°6 PE_1	N°5 PE_0	N°4 PA_5	N°3 PA_3	N°2 PA_2	N°1 PA_1	N°0 PA_0	V_{REF^+}	V_{REF^-}
0 0 0 0	A	A	A	A	A	A	A	A	V_{DD}	V_{SS}
0 0 0 1	A	A	A	A	V_{REF^+}	A	A	A	PA_3	V_{SS}
0 0 1 0	D	D	D	A	A	A	A	A	V_{DD}	V_{SS}
0 0 1 1	D	D	D	A	V_{REF^+}	A	A	A	PA_3	V_{SS}
0 1 0 0	D	D	D	D	A	D	A	A	V_{DD}	V_{SS}
0 1 0 1	D	D	D	D	V_{REF^+}	D	A	A	PA_3	V_{SS}
0 1 1 x	D	D	D	D	D	D	D	D	V_{DD}	V_{SS}
1 0 0 0	A	A	A	A	V_{REF^+}	V_{REF^-}	A	A	PA_3	PA_2
1 0 0 1	D	D	A	A	A	A	A	A	V_{DD}	V_{SS}
1 0 1 0	D	D	A	A	V_{REF^+}	A	A	A	PA_3	V_{SS}
1 0 1 1	D	D	A	A	V_{REF^+}	V_{REF^-}	A	A	PA_3	PA_2
1 1 0 0	D	D	D	A	V_{REF^+}	V_{REF^-}	A	A	PA_3	PA_2
1 1 0 1	D	D	D	D	V_{REF^+}	V_{REF^-}	A	A	PA_3	PA_2
1 1 1 0	D	D	D	D	D	D	D	A	V_{DD}	V_{SS}
1 1 1 1	D	D	D	D	V_{REF^+}	V_{REF^-}	D	A	PA_3	PA_2

Ces entrées n'existent pas sur le 16F876 (boîtier 28 pins).

Remarque :

Au reset le registre ADCON1 est initialisé à h'00'. Cela signifie que les 5 bits du Port A et les 3 bits du Port E sont configurés en entrées analogiques.

Pour récupérer le 5 bits du Port A et les 3 bits de Port E en tant que I/O digitales il faut écrire la valeur h'06' dans ADCON1.

I/12- Mémoire EEPROM :

Il y a 2 zones mémoire EEPROM accessible par l'utilisateur. La zone DATA et la mémoire Flash de Programme.

- La zone DATA est organisée en mot de 8 bits. Les 256 octets se programment et se lisent comme avec un PIC 16F84. Elle est positionnée dans le chip entre les adresses h'2100' et h'21FF'.

On l'adressera (de h'00' à h'FF') par le registre spécifique: EEADR en h'10D'(page 2). Les données transiteront par le registre: EEDATA en h'10C' (page 2).

- La zone EEPROM de programme qui est organisée en mots de 14 bits, est également accessible par l'utilisateur. Pour adresser les 8 K mots il faut 13bits d'adresse. Un registre

supplémentaire est disponible: EEADRH en h'10F'(page 2). Les 8 bits LSB de l'adresse seront passés par EEADR et les 5 bits MSB par EEADRH.

Les données sur 14 bits seront échangées à travers le registre EEDATA pour les 8 bits LSB et grâce à un registre supplémentaire : EEDATH en h'10E'(page 2) pour les 6 bits MSB.

I/12.1-Registres utilisés par l'EEPROM sont :

EEDATA : en h'10C' (page 2).

EEADR : en h'10D' (page 2).

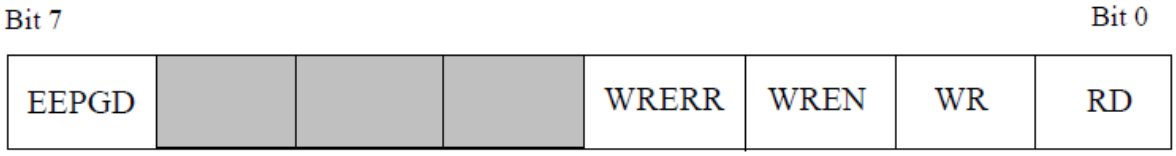
EEDATH : en h'10E' (page 2).

EEADRH : en h'10F' (page 2).

EECON1 : en h'18C' (page 3).

EECON2 : en h'18D' (page 3).

EECON1 : (h'18C' : page 3).



Au reset : EECON1 = X000X000

Bit 7 : **EEPGD** =Program/Data EEPROM select bit : choix de la zone EEPROM.

1= Accès à la zone EEPROM de programme.

0= Accès à la zone EEPROM de DATA.

Ce bit ne doit pas être changé pendant une opération de lecture ou d'écriture.

Bit 3: **WRERR** =EEPROM error flag.

1= Opération d'écriture prématurément terminée (causée par reset).

0= Opération d'écriture achevée normalement.

Bit 2 : **WREN** =EEPROM write enable bit.

1= Autorise cycle d'écriture en EEPROM.

0= Interdit l'écriture en EEPROM.

Bit 1 : **WR** = Write Control bit.

1= Lance le cycle écriture de l'EEPROM. Ce bit est remis à "0" par hard.

0= Le cycle d'écriture est terminé.

Bit 0: **RD** = Read Control bit.

1= Lance un cycle de lecture de l'EEPROM. Ce bit est remis à "0" par hard.

0= Pas de cycle lecture.

EECON2 : (h'18D' : page 3).

Ce registre n'est pas un registre de configuration physique. Il n'est utilisé que pendant les séquences d'écriture en EEPROM.

Remarque: **Pour pouvoir écrire en EEPROM programme, il faut configurer correctement certains bits du REGISTRE de CONFIG en h'2007'.**
Le bit WRT (Flash Program Memory enable) doit être forcé à "1". Les bits CP0 et CP1 doivent être tous les deux à "1" (Code Prote ct OFF), ou bien ne protéger qu'une zone particulière dans laquelle on ne voudra pas écrire.
Les procédures de lecture et d'écriture en zone DATA sont identiques à celles décrites pour le PIC 16F84.
Pour la zone flash du programme, elles sont particulières pour les PIC 16F876et 16F877.

I/12.2-Lecture en zone EEPROM programme :

1 - Mettre le MSB de l'adresse dans EEADRH.
2 - Mettre le LSB de l'adresse dans EEADR
3 - Mettre le bit EEPGD à "1" pour pointer la zone programme.
4 - Mettre le bit RD à "1" pour lancer le cycle de lecture.
5 - Attendre 2 cycles par deux NOP.
6 - Lire le MSB dans EEDATH et le LSB dans EEDATA.

I/12.3-Ecriture en zone eeprom programme :

1 - Mettre le MSB de l'adresse dans EEADRH.
2 - Mettre le LSB de l'adresse dans EEADR.
3 - Mettre le MSB de la DATA dans EEDATH.
4 - Mettre le LSB de la DATA dans EEDATA.
5 - Mettre le bit EEPGD à "1" pour pointer la zone programme.
6 - Mettre le bit WREN à "1" pour autoriser l'écriture.
7 - Inhiber les interruptions par mise à "0" du bit GIE de INTCON.
8 - Lancer la séquence spécifique suivante :
- ✓ Ecrire h'55' dans EECON2.
- ✓ Ecrire h'AA' dans EECON2.
- ✓ Lancer le cycle d'écriture par WR=1.

- ✓ On attend 2 cycles par deux NOP.
- ✓ Le PIC ignore maintenant toutes les instructions.
- ✓ A la fin de l'écriture, le PIC exécute l'instruction suivante.

9 - On autorise les interruptions par GIE=1.

10 - On interdit l'écriture en EEPROM par WREN = 0.

Exemple de programme :

```
;*******************************************************************************
;***  Ecriture en h'03FF' de la Data h'3439' (2 caractères ASCII :"4" et "9")  ***
;*******************************************************************************

            PAGE2           ;accès page 2
            MOVLW h'03'
            MOVWF EEADRH    ;MSB de l'adresse=03
            MOVLW h'FF'
            MOVWF EEADR     ;LSB de l'adresse=FF
            MOVLW h'34'
            MOVWF EEDATH    ;MSB de la DATA=34
            MOVLW h'39'
            MOVWF EEDATA    ;LSB de la DATA=39
            PAGE3           ;accès page 3
            BSF   EECON1,7  ;bit EEPGD=1 : accès mémoire programme
            BSF   EECON1,2  ;bit WREN=1 : autorise écriture en EEPROM
            BCF   INTCON,7  ;bit GIE=0 : masque les IT
Séquence    MOVLW h'55'
spécifique  MOVWF EECON2
            MOVLW h'AA'
            MOVWF EECON2
            BSF   EECON1,1  ;bit W=1 : démarre cycle écriture
            NOP             ;2 cycles d'attente
            NOP             ;Le micro attend la fin de l'écriture
            BSF   INTCON,7  ;bit GIE=1 : autorise les IT
            BCF   EECON1,2  ;bit WREN=0 : interdit écriture en EEPROM
```

I/13- Timers:

I/13.1-Module timer 0:

Ce module est le même que dans le PIC 16F84.

Le compteur/Timer TMR0 a les caractéristiques suivantes :

- Compteur sur 8 bits.
- Lecture / écriture de TMR0.
- Pré diviseur 8 bits programmable.
- Choix de l'horloge : interne en Timer et externe en compteur.
- Interruption au débordement (passage de FF à 00).
- Choix du front de l'horloge en mode horloge externe.

Tous les bits de configuration de TMR0 sont dans le registre OPTION en h'81'page 1 ou en h'181' page 3.

Le registre TMR0 est à l'adresse h'01' page 0 ou en h'101' page 2.

I/13.2-Mode Timer :

Le choix de ce mode se fait par : TOCS = 0 (b5 de OPTION).

TMR0 est incrémenté à chaque cycle instruction (Fosc/4), en considérant le pré diviseur avec un rapport de 1.

I/13.2/1-Mode compteur :

Ce mode est sélectionné si TOCS = 1. TMR0 est alors incrémenté à chaque front montant ou descendant sur la broche PA4/CLK

(pin3). Le choix du front est fait par le bit TOSE (b4 d'OPTION).

Si TOSE = 0 le compteur s'incrémente à chaque front montant.

Si TOSE = 1 c'est le front descendant qui incrémente le compteur.

I/13.2/2-Le prediviseur :

Il est partagé entre le Watch dog et TMR0.

L'affectation se fait par le bit PSA (b3 d'OPTION).

Si PSA = 0 le pré diviseur est affecté à TMR0. Le choix du rapport de division se fait avec les bits PS2, PS1 et PS0 (b2, b1 et b0 d'OPTION).

Si PSA = 1 le pré diviseur est affecté au Watch dog et le rapport de division pour TMR0 est fixé à 1.

I/13.2/3-Interruption :

Elle est générée quand TMR0 passe de la valeur FF à 00.

Le Flag TOIF (b2 d'INTCON) passe à "1". On peut masquer la génération de l'interruption en mettant le bit TOIE (b5 de INTCON) à "0".

Le Flag TOIF dit être remis à zéro par soft dans le sous-programme d'interruption, avant de ré-autoriser cette interruption.

I/13.3-Module Timer 1 :

Le Timer 1 est un compteur sur 16 bits constitué de 2 registres 8 bits TMR1Hen h'0F' page 0 et TMR1L en h'0E' page 0 également, que l'on peut lire ou écrire.

Le registre TMR1 (constitué de TMR1H et TMR1L) s'incrémente de h'0000'jusqu'à h'FFFF' et repasse ensuite à h'0000' pour continuer le comptage.

Quand il y a débordement, une interruption peut être générée si on l'autorisée par TMR1IE =1 (bit 0 de PIE1) et le Flag TMR1IF (bit 0 de PIR1) passe à "1".

Ce module peut fonctionner en mode TIMER, quand il s'incrémente à chaque cycle instruction (Fosc/4 avec le pré diviseur considéré à "1") ou en mode compteur, quand il s'incrémente à chaque front montant de l'horloge externe appliquée sur le Port C0.

L'horloge externe peut également être l'oscillateur interne, dont la fréquence est fixée par un quartz externe branché entre la broche Port C0 et la broche Port C1.

Le contrôle du TIMER 1 se fait par le registre T1CON en h'10' page 0.

I/13.3/1-T1CON : (h'10' : page 0) :

Bit 7 Bit 0

		T1CKPS1	T1CKPS0	T1OSCEN	$\overline{\text{T1SYNC}}$	TMR1CS	TMR1ON

Au reset : T1CON = 00000000

Bit 7 et bit 6 : **bits non implémentés.**

Bit 5 et bit 4 : **T1CKPS** = Sélection du pré diviseur placé avant le TIMER.

T1CKPS1	T1CKPS0	PRE DIV
0	0	1
0	1	2
1	0	4
1	1	8

Bit 3 : **T1OSCEN :** Bit d'autorisation de l'oscillateur du Timer 1.

1 = oscillateur autorisé

0 = oscillateur stoppé.

Bit 2 : **T1SYNC :** Bit de contrôle de la synchronisation du CLK externe.

1 = Pas de synchronisation de l'horloge externe.

0 = Synchronisation de l'horloge externe.

Bit1 : **TMR1CS :** Bit de sélection de la source horloge.

1 = Mode Compteur: Clk externe sur la broche PC0 ou Quartz entre PC0 et PC1

0 = Mode Timer: Clk interne = Fosc/4.

Bit 0 : **TMR1ON :** Bit d'autorisation du Timer 1.

1 = Timer 1 en service.

0 = Timer 1 stoppé.

I/13.4-Oscillateur interne du Timer 1 :

Un oscillateur à quartz a été embarqué sur le chip. Il est branché entre les broches PC0 (oscillateur out) et PC1 (oscillateur in). Il est mis en service par la mise à "1" du bit T1OSCEN. Cet oscillateur à faible consommation est limité à200 KHz. Il continu à osciller en mode SLEEP du PIC.

Il est principalement destiné pour générer un événement temps réel toutes les secondes par utilisation d'un quartz 32,768 KHz.

I/13.5-Schéma synoptique du Timer 1 :

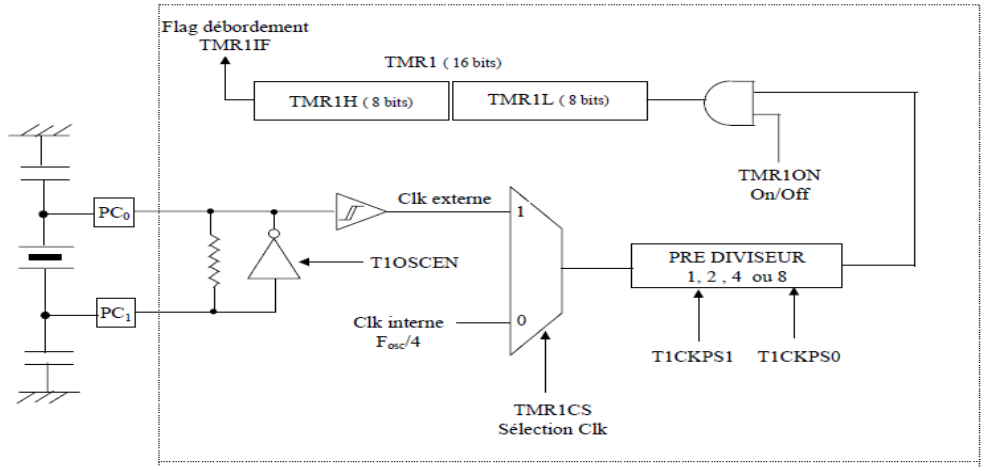


Figure I.8- Schéma bloc du Timer.

I/13.6-Module Timer 2 :

Le module Timer 2 est un compteur 8 bits avec pré diviseur et post diviseur.

Ce compteur TMR2 en h'11 ' page 0 est un registre en lecture ou écriture. Il possède un registre 8 bits pour la période : PR2 en h'92' page 1.

Le compteur s'incrémente de h'00' jusqu'à la valeur contenue par PR2 et repasse ensuite à "0" pour continuer le comptage. Au reset PR2 est initialisé à "FF".

L'entrée du compteur est l'horloge cycle interne : Fosc/4 qui passe à travers un pré diviseur programmable par 1, 4 ou 16.

La sortie du compteur passe dans un post diviseur programmable sur 4 bits entre 1 et 16.

Quand la sortie du compteur passe par la valeur programmée dans PR2, il y a génération d'une interruption (si elle a été autorisée par TMR2IE=1) et le flag
TMR2IF est positionné à "1". Ceci bien entendu en considérant le post diviseur programmé à "1".
Le contrôle du Timer 2 se fait par le registre T2CON en h'12' page 0.

I/13.6/1-T2CON : (h'12' : page 0) :

Bit 7 Bit 0

	TOUTPS3	TOUTPS2	TOUTPS1	TOUTPS0	TMR2ON	T2CKPS1	T2CKPS0

Au reset : T2CON = 00000000
Bit 7 : **bit non implémenté.**
Bit 6 à bit 3 : **TOUTPS :** Programmation du Post diviseur.
0 0 0 0 = post divise par 1.
0 0 0 1 = post divise par 2.
0 0 1 0 = post divise par 3.
. . . .
. . . .
1 1 1 1 = post divise par 16.
Bit 2 : **TMR2ON :** mise en service du Timer 2.
1= Timer 2 : On.
0= Timer 2 : Off.
Bit 1 et bit0 : **T2CKPS :** Programmation du pré diviseur.
0 0= pré_ diviser par 1.
0 1= pré_ diviser par 4.
1 X= pré_ diviser par 16.

I/13.6/2-Module ccp : capture compare et PWM :

Il y a deux modules identiques CCP1 et CCP2 composés chacun d'un registre16 bits. Ils peuvent opérer soit comme un registre 16 bits de capture, soit comme un registre 16 bits de comparaison, soit enfin comme un registre 8 bits pour générer du PWM.
Le module CCP1 est constitué de deux registres de 8 bits : CCPR1L en h'15'page 0 et CCPR1H en h'16' page 0. Ce module est contrôlé par le registreCCP1CON en h'17' page 0. La

sortie en mode COMPARE ou mode PWM et l'entrée en mode CAPTURE se font par la broche PC2.

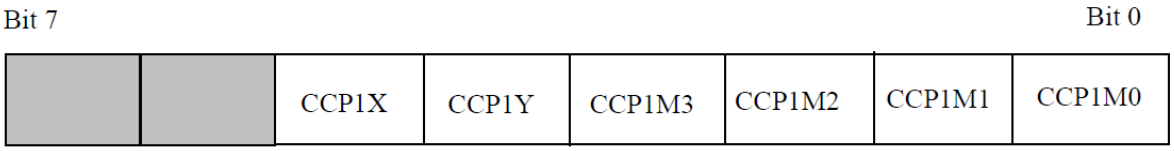
Le module CCP2 est constitué de deux registres de 8 bits : CCPR2L en h'1B'page 0 et CCPR2H en h'1C' page 0. Ce module est contrôlé par le registreCCP2CON en h'1D' page 0.

La sortie en mode COMPARE ou mode PWM et l'entrée en mode CAPTURE se font par la broche PC1.

En mode COMPARE ou CAPTURE, les modules utilisent le TIMER 1. En mode PWM, ils utilisent le TIMER 2.

Les registres de contrôles CCP1CON et CCP2CON sont identiques. On ne décrira que CCP1CON.

I/13.6/2.1-CCP1CON : (h'17' : page 0). (et CCP2CON en h'1D' : page 0.)



Bit 4 : **CCP1X et CCP1Y :**

Bits non utilisés en modes Compare et Capture.

Ce sont les 2 bits LSB pour le Duty cycle en mode PWM. Les 8 bits MSB sont dans le registre CCPR1L en h'15' page 0.

Bit 3 à bit 0 : **CCP1M3 à CCP1M0 :** bits de mode Au reset : CCP1CON = 00000000

Bit 7 et bit 6 : **bits non implémentés.**

Bit 5 et sélection du

0 0 0 0 = Module CCP stoppé.

0 1 0 0 = Mode Capture à chaque front descendant.

0 1 0 1 = Mode Capture à chaque front montant.

0 1 1 0 = Mode Capture tous les 4 fronts montants.

0 1 1 1 = Mode Capture tous les 16 fronts montants.

1 0 0 0 = Mode Compare. Pin de sortie mise à "1" et Flag CCP1IF = 1 à l'égalité.

1 0 0 1 = Mode Compare. Pin de sortie mise à "0" et Flag CCP1IF = 1 à l'égalité.

1 0 1 0 = Mode Compare. Génération d'une Interrup. et Flag CCP1IF = 1 à l'égalité.

1 0 1 1 = Mode Compare. Evénement spécial généré et Flag CCP1IF = 1 à l'égalité.

1 1 x x = Mode PWM.

I/13.6/2.2-Mode comparaison :

Les deux modules CCP étant identiques on ne décrira que le module 1.

Les 16 bits des registres CCPR1 (CCPR1H et CCPR1L) sont constamment comparés avec le valeur sur 16 bits des registres du Timer 1 (TMR1H etTMR1L).

Quand il y a égalité, la broche préalablement programmée en sortiePC2, passe soit à "1" soit à "0" suivant la configuration des 4 bits CCP1M du registre CCP1CON.

Au même instant le Flag CCP1IF est mis à "1".

En mode Compare, les événements spéciaux générés quand il y a égalité sont:

- Pour CCP1: reset du Timer 1.
- Pour CCP2 : reset du Timer 1 et démarrage d'une conversion A/D.

Dans ce cas la broche de sortie n'est pas affectée, mais le Flag CCP1IF est mis à "1".

Il est rappelé que ce Flag doit être remis à "0" par soft.

I/13.6/2.3-Mode capture :

Quand un événement extérieur apparaît sur la broche préalablement programmée en entrée PC2, la valeur du 16 bits des registres du Timer 1 (TMR1L et TMR1H) est recopiée dans les registres CCPR1 (CCPRIH etCCPR1L). Cet événement est programmable par les 4 bits CCP1M du registreCCP1CON. La capture peut avoir lieu à chaque front descendant, à chaque front montant, tous les 4 ou tous les 16 fronts montants.

Quand la capture a eu lieu, le flag CCP1IF est mis à "1".

Ce bit doit être remis à "0" par soft.

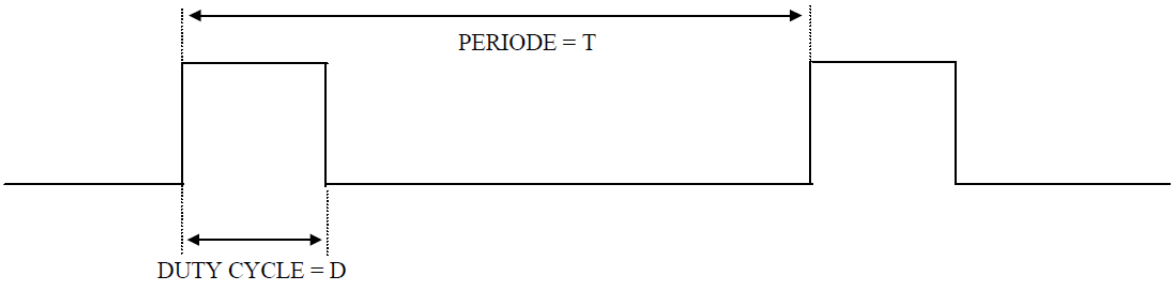
Si une nouvelle capture survient alors que la valeur dans CCPR1 n'a pas étélue, l'ancienne valeur est perdue.

Les fonctions de Capture et de Compare sur le Timer 1 par les modules CCP1et CCP2peuvent générer une interruption quand le Flag CCP1IF passe à "1"

Si le bit d'autorisation CCP1IE du registre PIE1 est mis à "1".

I/13.6/2.4-Mode PWM :

Un signal PWM est caractérisé par une période, et un temps de travail ou le signal est à "1". Ce temps est appelé DUTY CYCLE.



La broche PC2 doit être configurée comme une sortie en mettant le bit 2 de TRISC à "0".

Le signal PWM est fabriqué à partir du Timer 2. Le signal interne Fosc/4 passe à travers le pré diviseur programmable par 1, 4 ou 16 et fait compter TMR2.

Quand ce registre atteint la valeur écrite dans le registre PR2,

Trois événements se produisent :

- RAZ du registre TMR2.
- La broche de sortie PC2 est mise à "1", sauf si le Duty cycle vaut 0.
- Chargement de la valeur du registre CCPR1L dans le registre de Duty.

Quand le registre TMR2 atteint la valeur inscrite dans le registre interne de Duty, c'est à dire la valeur qui avait été inscrite dans CCPR1L, la broche de sortie PC2 est remise à "0".

La période est déterminée par la relation suivante : [7]

$$T = \left[PR_2 + 1\right] \cdot 4 \cdot T_{osc} \cdot \text{valeur du pré diviseur}$$

La durée du Duty cycle est la valeur écrite dans le registre 8 bits : CCPR1L.

La durée du signal au niveau "1" est donnée par la relation :

$$D = \left[CCPR1L\right] \cdot 4 \cdot T_{osc} \cdot \text{ valeur du pré diviseur}$$

On peut avoir une meilleure résolution sur le Duty cycle en utilisant les 2 bits de LSB réservés à cet effet dans le registre CCP1CON.

Il s'agit des bits 4 et 5 : CCP1X et CCP1Y. La résolution est dans ce cas divisée par 4. Si on appelle X le registre sur 10 bits, constitué de ces 2 bits en LSB et des 8 bits deCCPR1Lcomme MSB, la durée à "1" est donnée par la relation :

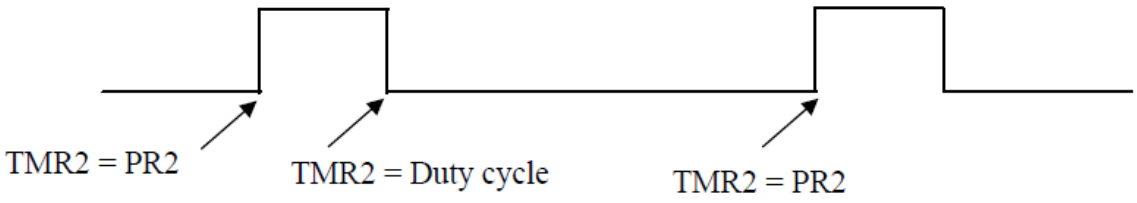
$$D = X \cdot T_{osc} \cdot \text{ valeur du pré diviseur}$$

Dans les 2 cas, quand on incrémente le registre CCPR1L de une unité, le Duty cycle augmente d'une durée égale à : 4.Tosc.valeur du pré diviseur.

Quand on passe à "1" le bit 5 la durée augmente de 2.Tosc.pré div.

Quand on passe à "1" le bit 4 de CCP1CON, la durée augmente de Tosc.pré div

Si la durée du Duty cycle est supérieure à la période, le signal n'est pas remis à "0".



I/13.6/2.5-Marche à suivre pour faire du PWM :

- Ecrire dans PR2 la valeur permettant d'obtenir la Période désirée.
- Ecrire dans CCPR1L la valeur donnant la durée du Duty cycle.
- Configurer la broche d'utilisation en sortie (bit = 0 dans TRISC).
- Initialiser le Timer 2 par T2CON (prédiv, TMR2 on).
- Initialiser le CCP par CCP1CON (mode PWM, 2 bits LSB Duty cycle).

I/13.7-Recapitulatif sur les TIMER et les modules CCP

TIMER 0 **Broche utilisée : PA4**	**TIMER 1** **Broches utilisées : PC0 et PC1**	**TIMER 2** **Broches utilisées : néant**
- 8 bits - Pré div de 2^0 à 2^8 - IT et Flag au débordement - Clk int ($F_{osc}/4$) ou ext (PA4)	- 16 bits - Pré div de 2^0 à 2^3 - IT et Flag au débordement - Clk int ($F_{osc}/4$) ou ext (PC0) - Oscil quartz ext PC_0/PC_1 - Sert en Capture et Compare	- 8 bits - Période programmable dans PR2 - Pré div par 1 , 4 ou 16 - Post div de 1 à 16. - IT et Flag quand Timer 2= PR2 - Clk int ($F_{osc}/4$) - Sert en PWM

CCP1 Broche utilisée : PC2	CCP2 Broche utilisée : PC1
- **Capture :** entrée sur broche PC2. *Recopie des 16 bits du timer 1 dans les 2 registres CCPR1H et CCPR1L quand survient un événement extérieur sur la broche PC2 tel que :* **ou ou 4 ou 16** - **Compare :** sortie sur broche PC2. *Comparaison des 16 bits du timer 1 et du contenu des 2 registres CCPR1H et CCPR1L et passage soit à"1" soit à "0" de la broche PC2.* - **PWM :** sortie sur broche PC2. *Période du signal donnée par Timer 2 et valeur dans PR2. Durée du Duty cycle dans CCPR1L.*	- **Capture :** entrée sur broche PC1. *Recopie des 16 bits du timer 1 dans les 2 registres CCPR2H et CCPR2L quand survient un événement extérieur sur la broche PC1 tel que :* **ou ou 4 ou 16** - **Compare :** sortie sur broche PC1. *Comparaison des 16 bits du timer 1 et du contenu des 2 registres CCPR2H et CCPR2L et passage soit à"1" soit à "0" de la broche PC1.* - **PWM :** sortie sur broche PC1. *Période du signal donnée par Timer 2 et valeur dans PR2. Durée du Duty cycle dans CCPR2L.*

I.14- 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.

Le microcontrôleur dispose de plusieurs sources d'interruptions.

• Une interruption externe, action sur la broche **INT/RB0**.

• Débordement du **TIMER0**.

• Changement d'état logique sur une des broches du **PORTB (RB4 à RB7)**.

• Une interruption d'un des périphériques (**PEIE**).

• Fin de programmation d'une case mémoire de l'**EEPROM**.

• Changement d'état sur le **PORTD** (**PSPIE**).

• Fin de conversion analogique numérique (**ADIE**).

• Réception d'une information sur la liaison série (**RCIE**).

• Fin d'émission d'une information sur la liaison série (**TXIE**).

• Interruption **SPI** ou **I2C** du module **MSSP** (**SSPIE**).
• Interruption du registre de capture et/ou de comparaison 1 (**CCPI1E**).
• Interruption du registre de capture et/ou de comparaison 2 (**CCPI2E**).
• Débordement du **TIMER1** (**TMR1E**).
• Débordement du **TIMER2** (**TMR2E**).
• Collision de BUS (BCLIE)

❖ *Mécanisme général d'une interruption :*

Nous pouvons dire, sans beaucoup nous tromper, qu'une routine d'interruption est un sous-programme particulier, déclenché par l'apparition d'un événement spécifique mais qui est très simple.

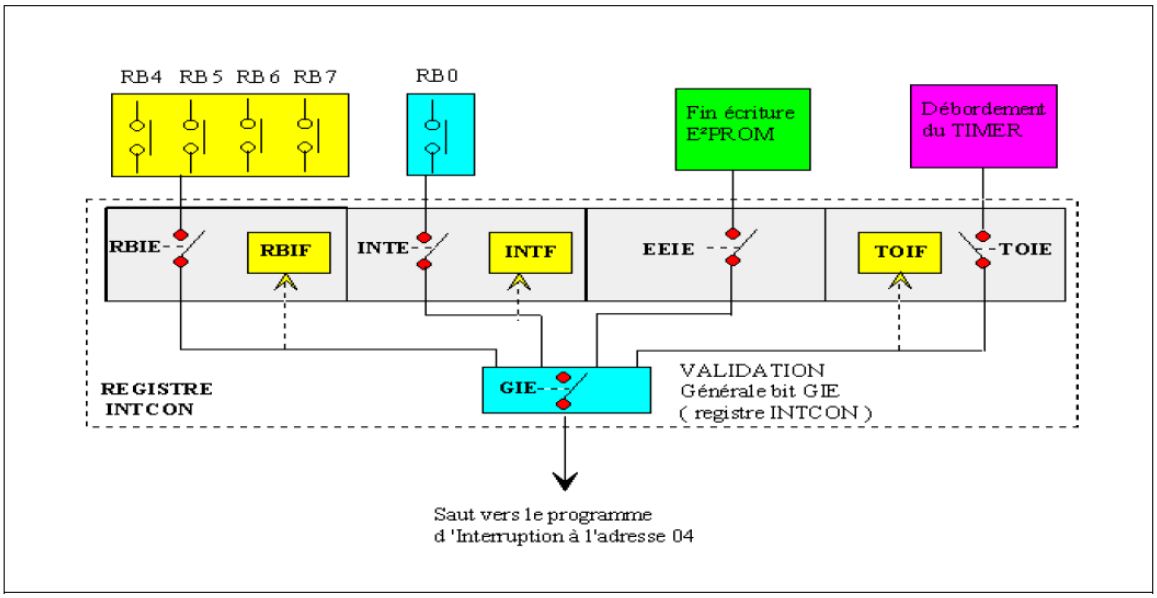


Figure I.9- Schéma de mécanisme d'interruption.

I.15- Conclusion :

Partant d'une présentation générale sur les microcontrôleurs, nous avons ensuite défini la famille des Pics et plus particulièrement le 16F876A.

En conclusion dans ce chapitre nous pouvons dire que le microcontrôleur peut bien jouer le rôle d'une unité de commande pour notre système.

Pour faire fonctionner cette unité de commande, il faut la programmer et l'adapter à un compilateur de programmation.

Remercîment

Avant tout, on rend grâce à DIEU tout puissant de nous avoir accordé la volonté et le courage

pour réaliser ce mémoire

La réalisation de ce mémoire a été possible grâce au concours de plusieurs personnes à qui je voudrais témoigner toute ma reconnaissance.

J'adresse mes sincères remerciements à tous les professeurs, intervenants et toutes les personnes qui par leurs paroles, leurs écrits, leurs conseils et leurs critiques ont guidé nos réflexions et ont accepté de nous rencontrer et répondre à nos questions durant notre recherches.

En exception MR_Bechar Hassan et MR-Hammdoun Abdel kader.

Au terme de ce travail je tiens tout d'abord à exprimer ma profonde gratitude à mon encadreur

Mr Nemiche Ahmed.

Mes remerciements s'adressent à tous les membres du jury pour l'honneur qu'ils nous ont fait en acceptant de juger ce travail.

Enfin, je remercie tous mes ami(e)s que j'aime tant Khadija-Khadîdja-Khadîdja-a ,Tema, Dalila et aussi notre nouvelle ami Boumediene.

Pour leur sincère amitié et confiance, et à qui je dois ma reconnaissance et mon attachement.

À tous ces intervenants, nous présentons nos remerciements, nos respects et notre gratitude.

Mercie a tous

RESUME (bilingue)

Le but dans ce mémoire consiste à l'étude et la réalisation d'un fréquencemètre multi calibre programmé .Notre fréquencemètre est basé essentiellement sur un microcontrôleur PIC16F876A programmé en utilisant le logiciel « Micro Pascal ». Et un compteur BC4020. Les impulsions émises sont transformées en signaux analogiques puis digitaux pour être affichées sur écran LCD.

Mots-clés : fréquence, pic, compteur, LCD.

SUMMARY (bilingual)

The goal in this memory consists under investigation and the realization of a frequency meter multi gauges programmed. Our frequency meter is based primarily on a microcontroller PIC16F876A programmed by using the software "Micro Pascal". And a meter BC4020. The emitted impulses are transformed into digital analog signals then to be displayed on screen LCD.

Keywords: frequency, peak, meter, LCD.

ملخص

الهدف من هذه المدكرة هو دراسة تطبيقية لتحقيق جهاز الترددات متعدد المعايير.

يعتمد جهازنا هذا بالخصوص على ميكرو كونترولر مبرمج باستخدام البرنامج ميكرو باسكال و عداد .النبضات المبعوثة تتحول الى اشارات تناظرية تم الى رقمية لعرضها على شاشة الكريستال السائل ل س د.

كلمات مفتاحية . تردد- بيك - عداد - ل س د .

ANNEXES

ANNEXE A : Logiciel proteus.

1- Logiciel PROTEUS :

Ce compose de deux parties, le logiciel ISIS pour la simulation des circuits électroniques, et le logiciel ARES pour dessiner les circuits imprimés.

a- L'éditeur ISIS :

Le but du logiciel ISIS est de dessiner, simuler des circuits électroniques et tracé les courbes.

a1- Présentation de l'éditeur ISIS :

La surface la plus grande de l'écran s'appelle "Fenêtre d'édition" et se comporte comme une fenêtre de dessin. C'est là qu'il fait placer et câblez les composants. Dans cette fenêtre on peut distinguer les barres d'outils suivantes : menu d'outils, désigne d'outils, commande outils

Et fenêtre d'édition, (figure1) :

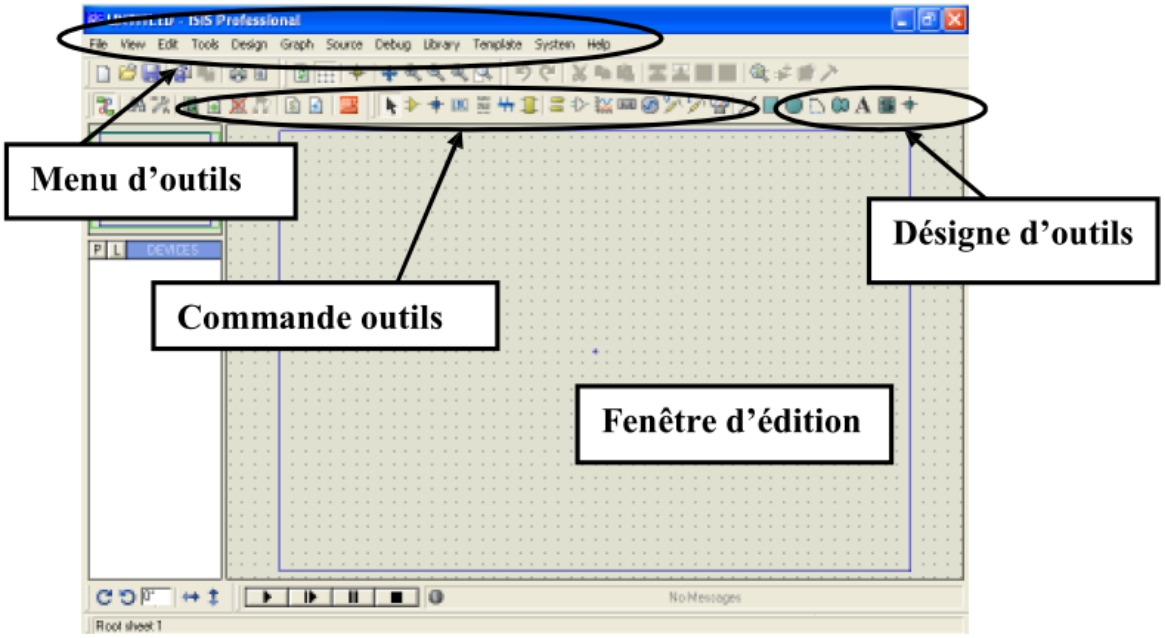


Figure3- 1 : Fenêtre principale d'ISIS.

a2) Placement et câblage des composants :

Le circuit que nous allons tracer comme un exemple est représenté en Figure (2).

ANNEXES

Nous commencerons par un exemple de circuit simple qui se compose d'un moteur on pont - H par interrupteur double plus l'alimentation (Vcc = 5Volt et la masse GND).

Commençons par un clic gauche sur l'icône d'un composant, Nous avons ainsi accès aux bibliothèques de composants. Un clic sur "P", du "sélecteur d'objets" ouvre une nouvelle fenêtre .Nous pouvons maintenant choisir un composant dans les différentes bibliothèques. Un double clic place le composant sélectionné dans le sélecteur d'objets et le rend disponible pour l'édition.

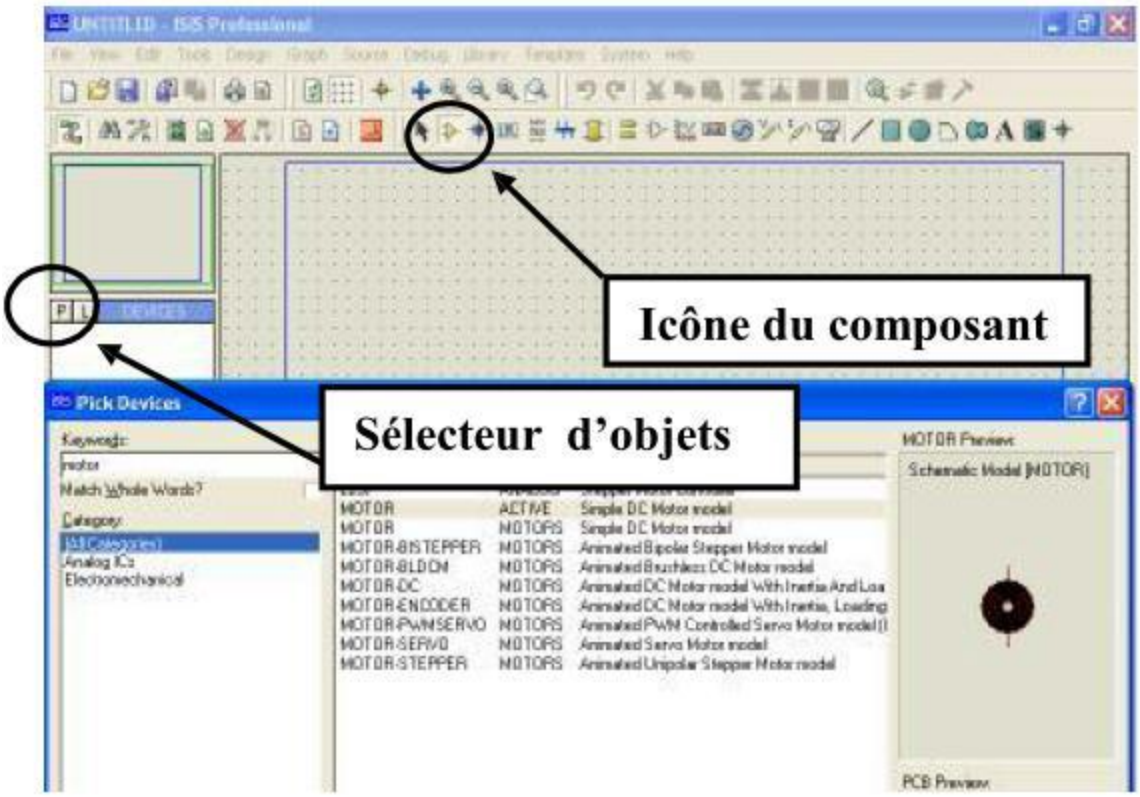


Figure3- 2 : Choix de composants.

Nous pouvons maintenant passer au placement de quelques fils (connexions). Faire un clic gauche, ISIS détecte que vous pointez sur la broche d'un composant et en déduit que vous voulez relier un fil. Pour terminer le circuit il fait un clic sur l'icône exécute le simulateur. (Figure 3).

ANNEXES

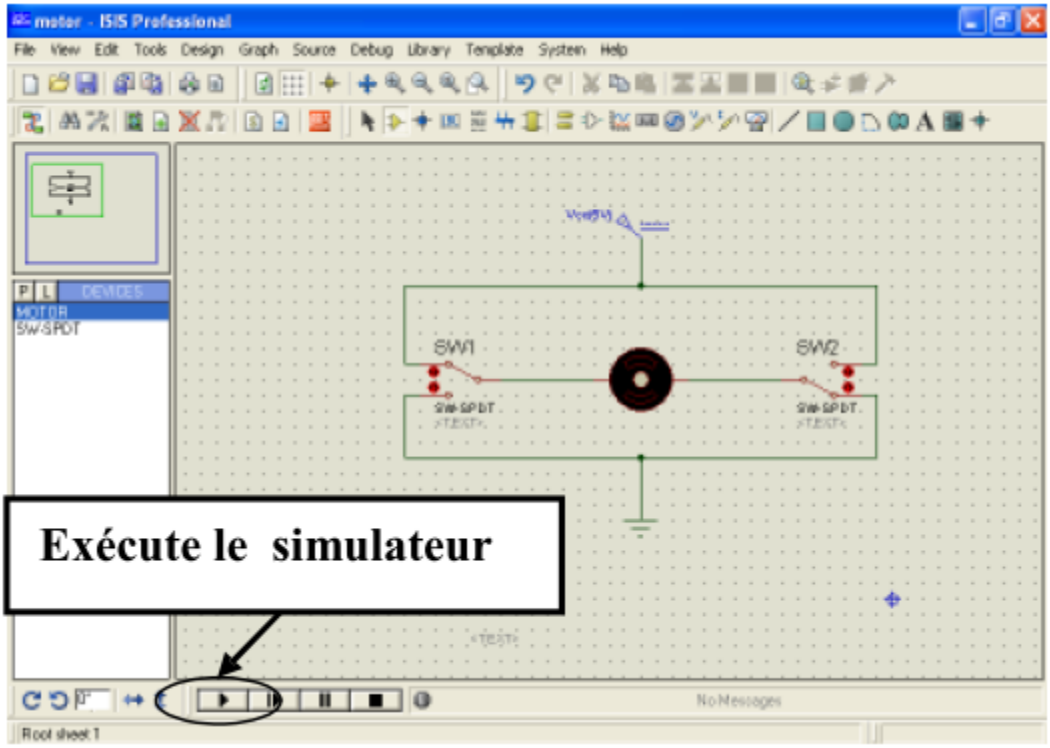


Figure 3-3 : Exemple réalisé par ISIS.

b) L'éditeur ARES :

ARES est un logiciel permettant le routage de cartes électronique en mode automatique ou manuel. Il est possible d'utiliser ARES sans avoir au préalable créé du schéma dans ISIS. Cette fonctionnalité permet de réaliser très rapidement des circuits de faible complexité en plaçant les composants et traçant les pistes directement dans ARES. Une fois les connections établies il est possible d'effectuer un routage automatique des pistes.

Dans ce logiciel vous pouvez également créer de nouveaux boîtiers et les placer dans une bibliothèque. Couplé avec ISIS nous avons un système complet qui nous permet d'effectuer avec un seul schéma toutes les étapes de la conception de la carte.

b1) Présentation d'ARES

L'aspect général de ARES et similaire de celui de ISIS, Figure (4), Il contient :

- Une barre de menu.
- Une fenêtre principale dans laquelle nous allons créer votre routage et aussi nos nouveaux composants.
- Une fenêtre d'aperçu en haut à droite.
- Une palette en dessous et un sélecteur de boîtiers.
- Un sélecteur de surface active, en bas à gauche.

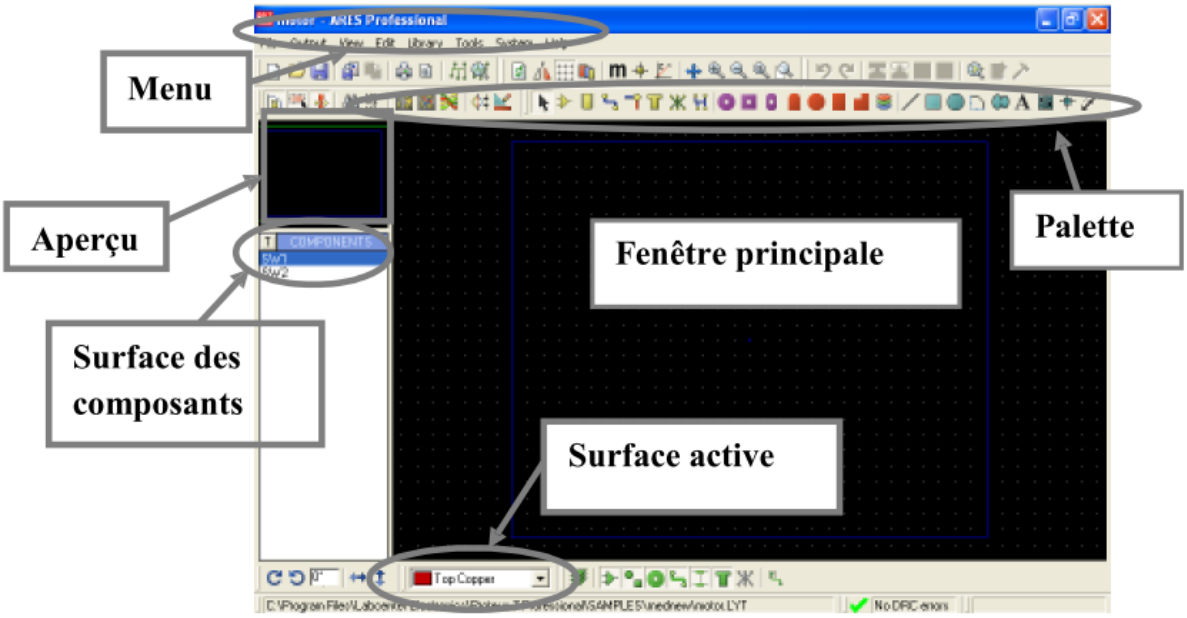


Figure3- 4: Présentation d'ARES.

b2) Création du circuit imprimé :

La première chose à faire avant de commencer un routage est de définir une carte aux dimensions du projet. Il est fortement conseillé de réaliser, si possible, Pour cela placez-vous en mode graphique et sélectionnez "Board Edge" dans le sélecteur de surfaces, Sélectionnez ensuite l'icône rectangle et dessinez un rectangle correspondant au contour de la carte.

c) Placement des composants :

c1) Placement automatique :

Après avoir défini une carte vous pouvez placer les composants en mode automatique. La fenêtre (Figure 3-5) apparaisse. Nous pouvons choisir les composants à placer. Après validation les composants sont positionnés sur la carte.

ANNEXES

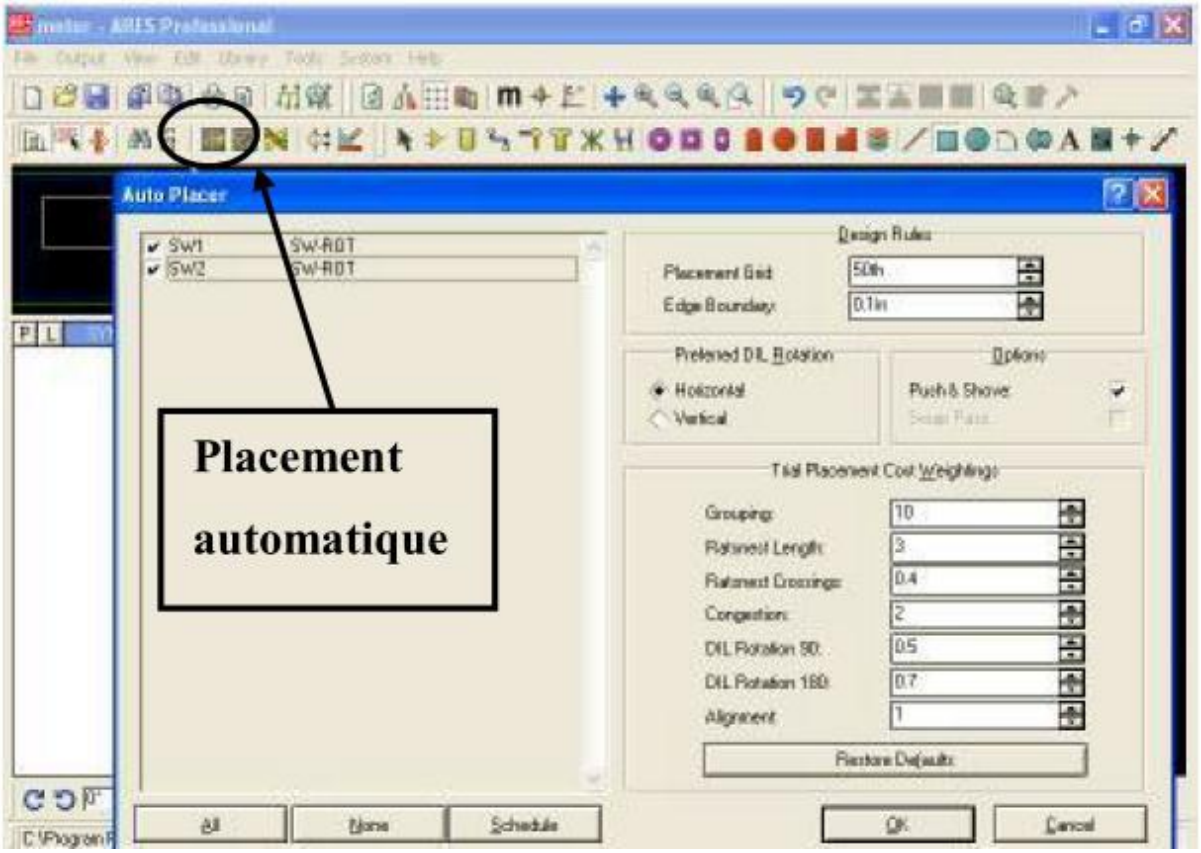


Figure 3-5 : Placement automatique.

c2) Placement Manuel :

On peut rarement se contenter d'un placement automatique. En raison d'exigences dues au projet il est très souvent nécessaire de placer des composants à des endroits précis. Dans ce cas on sélectionne le composant dans la fenêtre de sélecteur. On oriente la sélection au moyen des icônes appropriés. On clique dans la fenêtre de placement et sans relâcher le bouton de la souris nous déplaçons le fantôme de composant.

d) Routage :

d1) Routage automatique :

Le routage automatique se lance dans le menu Outils Routeur automatique. Dans la fenêtre de configuration nous pouvons choisir de router tout le chevelu, ou une partie. Nous pouvons également déterminer les isolations à respecter : Figure (3-6).

ANNEXES

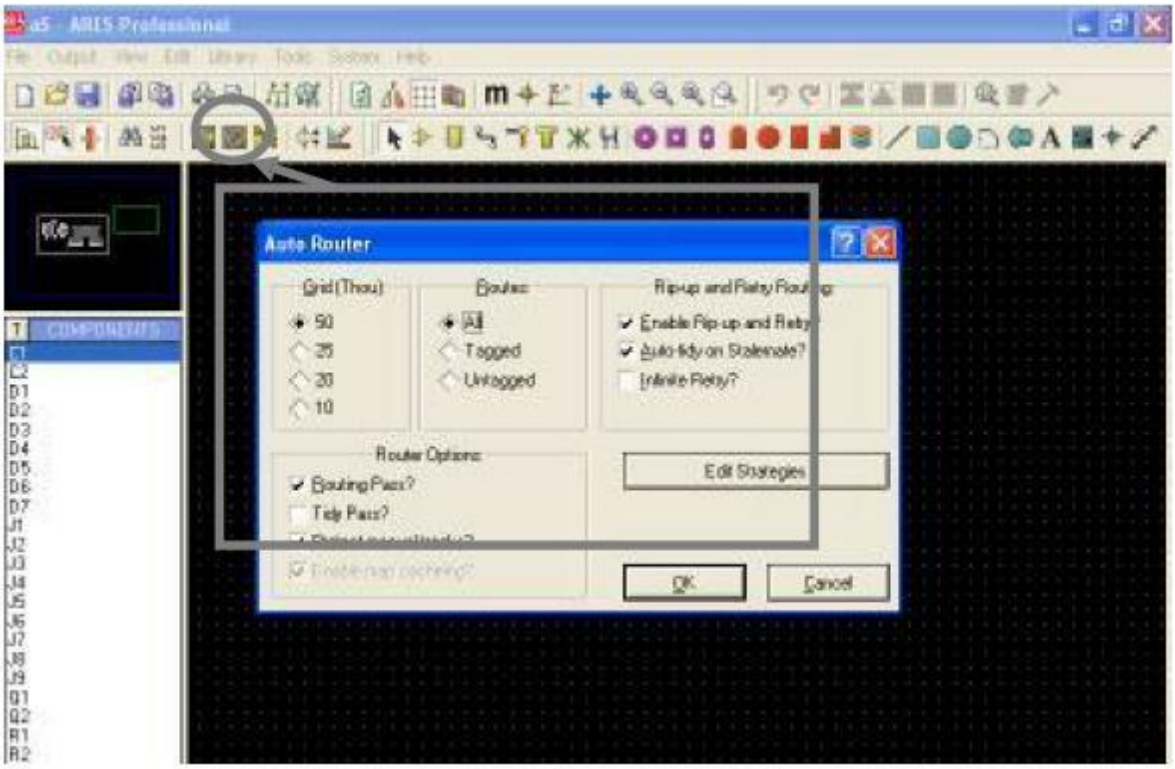


Figure 3-6 : Routage automatique.

d2) Routage Manuel :

Comme pour le placement automatique il est très rare qu'un routage automatique convienne sans modifications. Il est donc très utile de pouvoir modifier ou créer des pistes manuellement.

Choisissons l'icône pistes puis sélectionnez un guide (en vert). Par des clics successifs, on peut dessiner notre piste en partant d'une extrémité du guide. Le tracé se termine lorsqu'atteigne l'autre extrémité.

ANNEXES

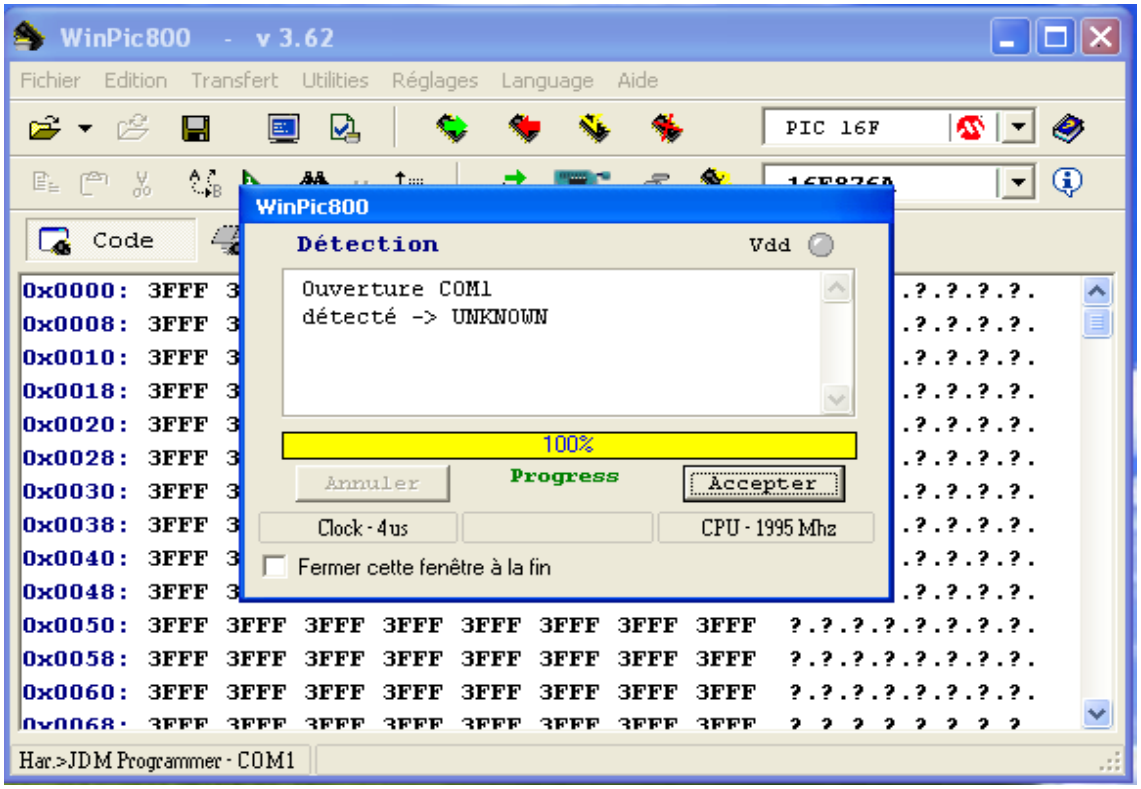
ANNEXE B : Les différentes étapes de logiciel IC-PROG.

Les différents étapes de logiciel IC-PROG :

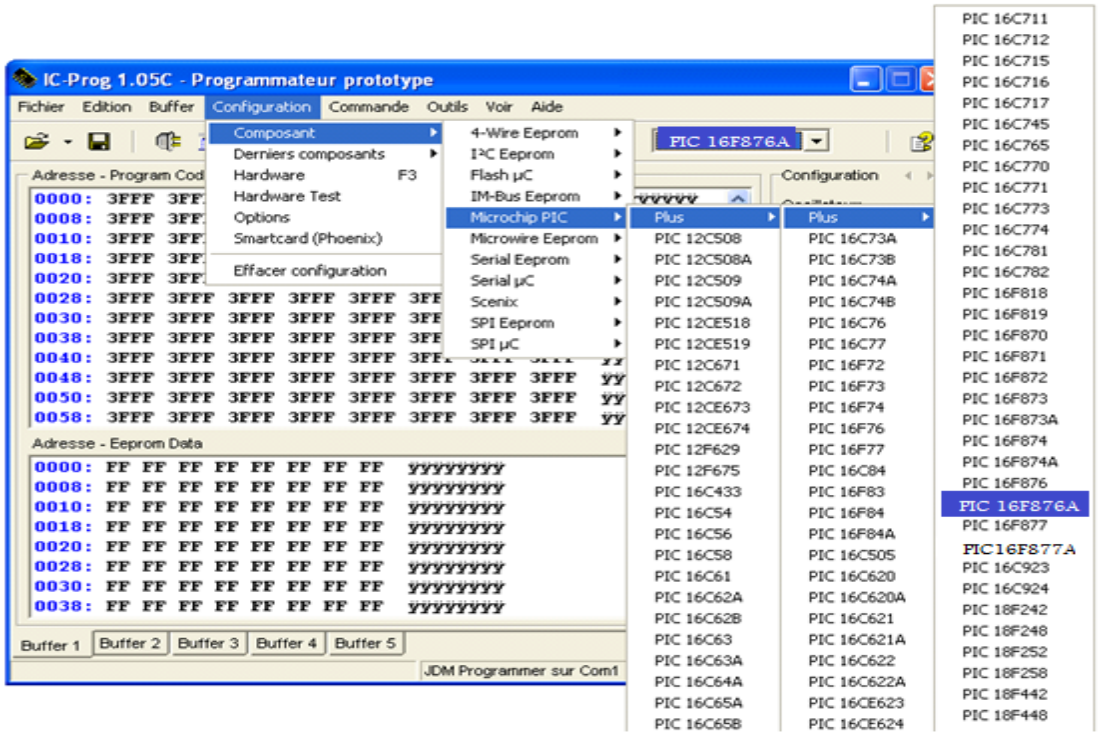
Sous Windows XP, on ouvre le fichier IC-PROG puis on clique dessus pour ouvrir une fenêtre d'option. Dans les sous menu compatibilité des menues propriétés on coche la case «exécuter ce programme en mode compatibilité pour›› puis on choisit « Windows 95››.

On lance l'ICPROG puis on ouvre le menu Hardware sous le menu «Configuration››, on =choisit l'option « Windows API ». Ensuite, nous sélectionnons le port de communication « COM1 » par exemple.

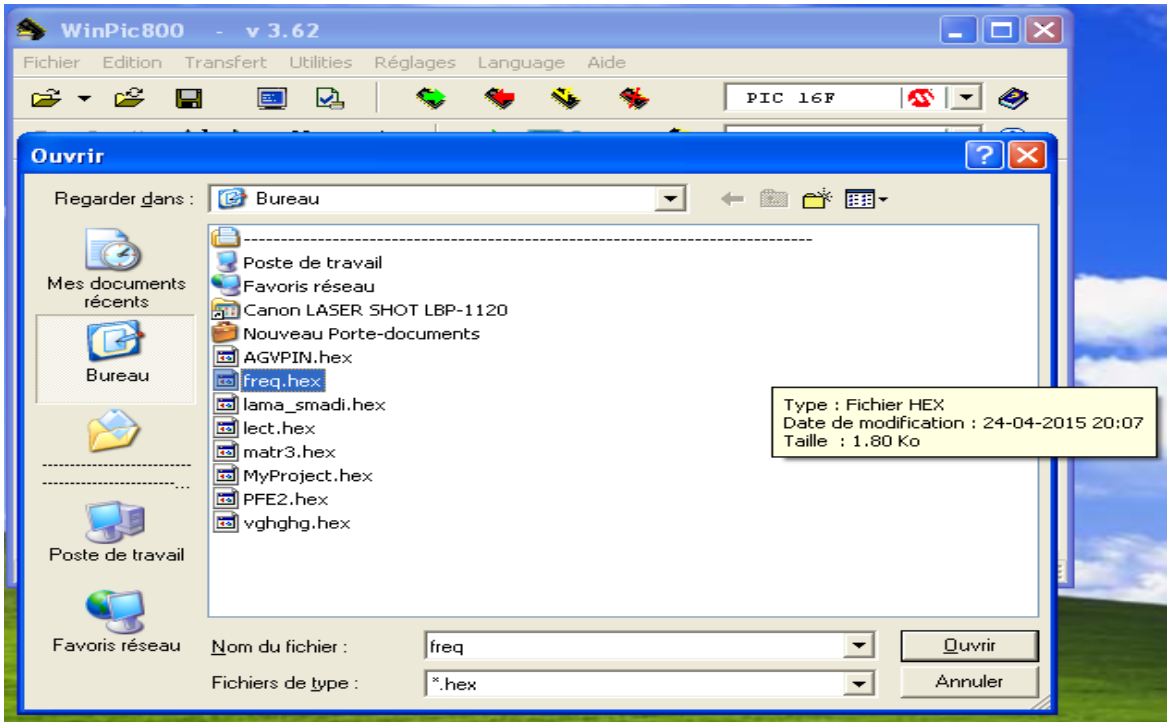
La puissance de ce logiciel est qu'il est capable de programmer une kyrielle de famille de composants et chaque famille a encore ses références. Pour cela, nous sélectionnons le type de composant à programmer : pour se faire nous ouvrons le menu « setting » puis « devises » puis « Micro chip PIC » et enfin le type de PIC à programmer.



ANNEXES

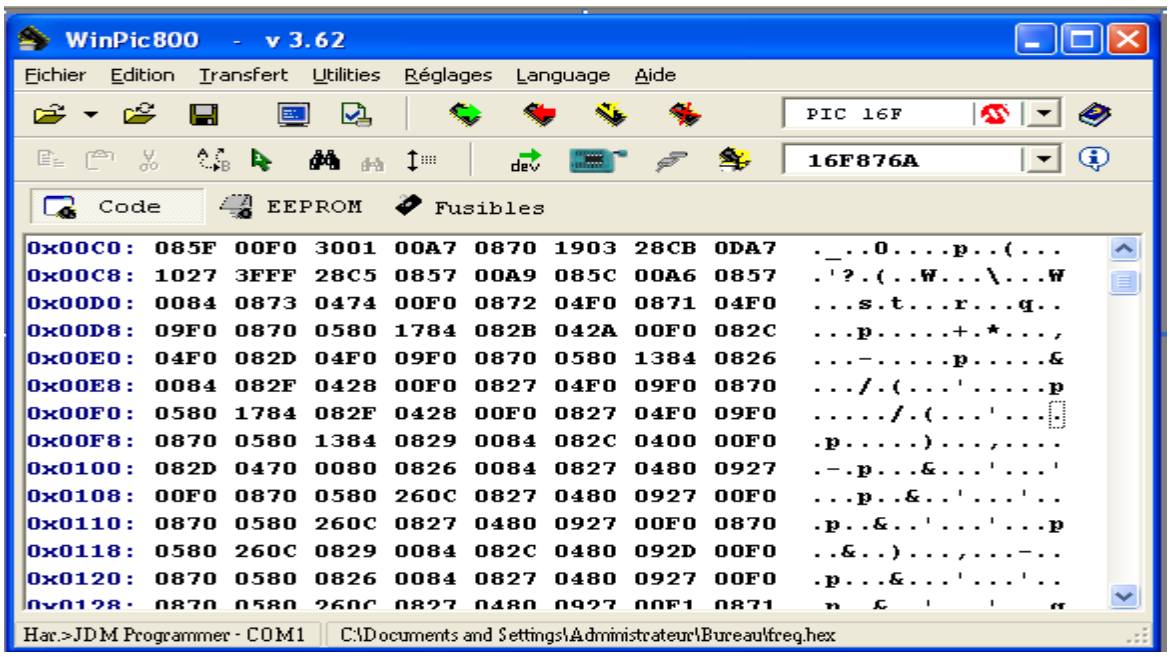


Prenons le fichier *.HEX de notre programme principale, depuis le Menu « File » puis «Open file ».Certains options de fonctionnement doivent être signalées au programmateur qui informera le PIC lors de la programmation.

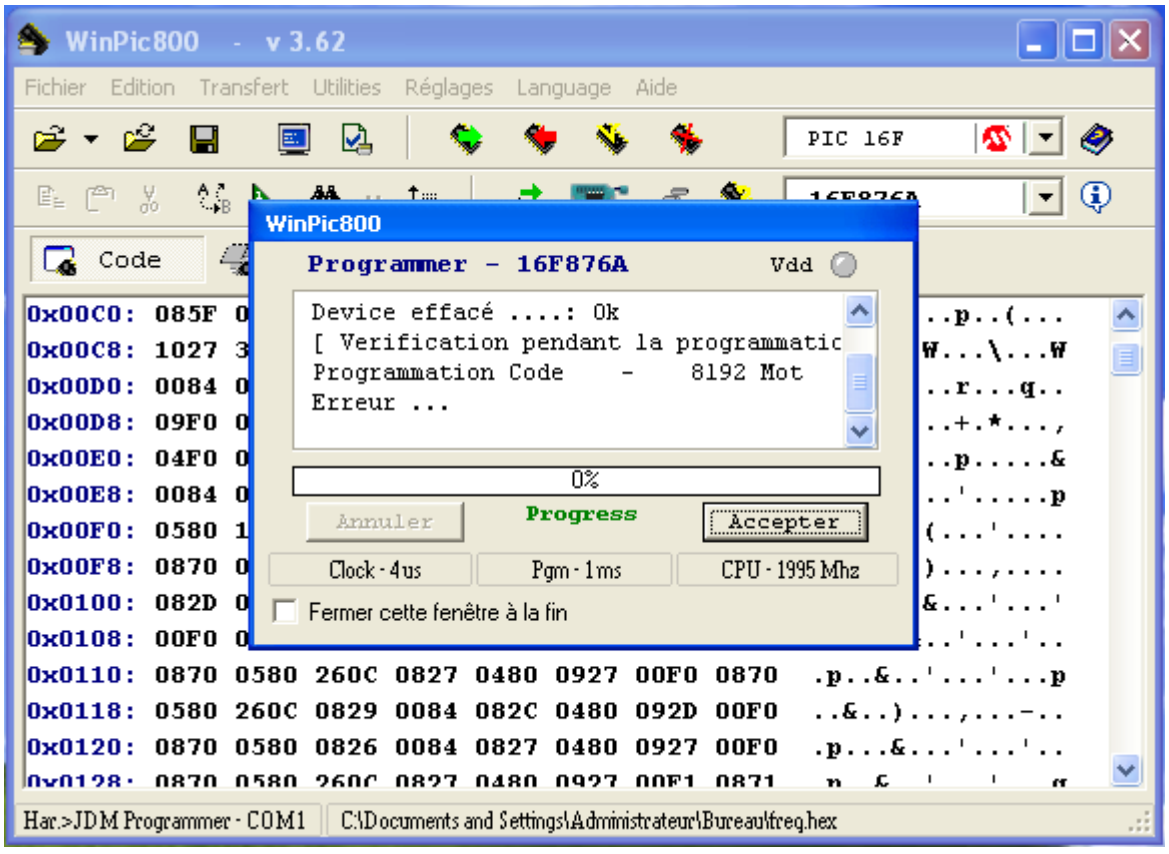


En suite le fichier hex va s'ouvrir (voir la figure ci-dessus).

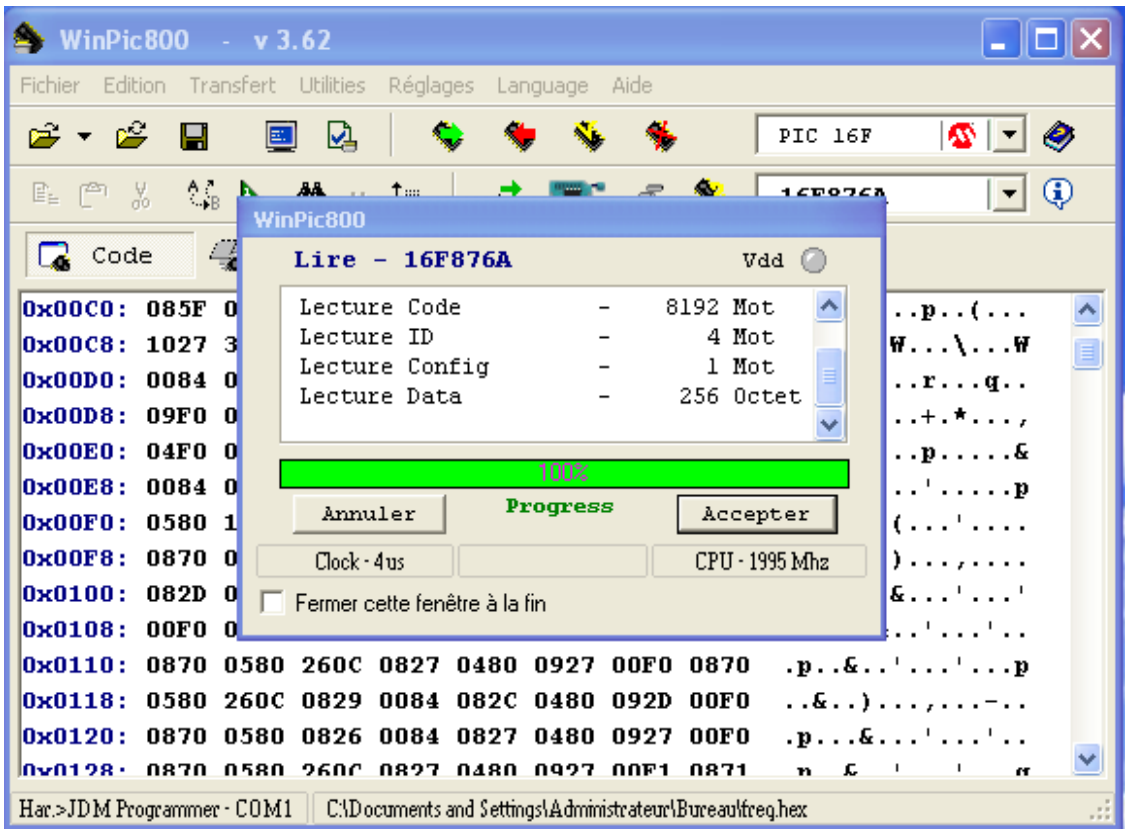
ANNEXES



En suite la lecture de fichier hex va s'exécuter automatiquement



ANNEXES


WinPic800 - v 3.62
Fichier Edition Transfert Utilities Réglages Language Aide
PIC 16F
Code
WinPic800
Lire - 16F876A
Vdd
Lecture Code - 8192 Mot
Lecture ID - 4 Mot
Lecture Config - 1 Mot
Lecture Data - 256 Octet
100%
Annuler
Progress
Accepter
Clock - 4us
CPU - 1995 Mhz
Fermer cette fenêtre à la fin
0x00C0: 085F 0
0x00C8: 1027 3
0x00D0: 0084 0
0x00D8: 09F0 0
0x00E0: 04F0 0
0x00E8: 0084 0
0x00F0: 0580 1
0x00F8: 0870 0
0x0100: 082D 0
0x0108: 00F0 0
0x0110: 0870 0580 260C 0827 0480 0927 00F0 0870
0x0118: 0580 260C 0829 0084 082C 0480 092D 00F0
0x0120: 0870 0580 0826 0084 0827 0480 0927 00F0
Har.>JDM Programmer - COM1
C:\Documents and Settings\Administrateur\Bureau\freq.hex