MINISTERE DE L'ENSEIGNEMENT SUPERIEUR ET DE LA RECHERCHE SCIENTIFIQUE
UNIVERSITE MOULOUD MAMMERI, TIZI-OUZOU

FACULTE DE GENIE ELECTRIQUE ET DE L'INFORMATIQUE
DEPARTEMENT D'ELECTRONIQUE

Mémoire de fin d'études

Présenté en vue de l'obtention
du Diplôme d'Ingénieur d'Etat en Electronique

Option : Contrôle

Thème:

Elaboration de la maquette de développement EASYPIC6

Proposé et dirigé par:
Mr: M .LAGHROUCHE

Etudié et réalisé par:
Mr: Mouloud CHIKH
Mr: Ali KECHIR

Année universitaire **2010/2011**

RESUME

Dans notre projet on a élaboré la maquette de développement EASYPIC6, les Pics de MICRISHIP ainsi que leur programmation, Le système de développement EasyPIC6 est un outil destiné à la programmation et à l'expérimentation sur les microcontrôleurs Pics. La carte inclue un programmateur avec mikroICD (débogueur intégré) servant d'interface entre le microcontrôleur et le PC. Il suffit d'écrire le code dans un des compilateurs, générer un fichier hex puis programmer le microcontrôleur grâce au programmateur PIC. on a réalisés quelques travaux pratiques on utilisant le Pic 16f887 et le 16f84a, la programmation est faite avec MICROC de mikroelektronica.

Sommaire

Introduction générale

La conception d'un système électronique digitale se fait soit en logique câblée soit en logique programmée. La logique câblée fait appel soit aux circuits logiques combinatoires, soit aux circuits logiques séquentiels (synchrones ou asynchrones) qui utilisent généralement des bascules.
La conception et la réalisation d'un circuit basé sur la logique combinatoire fait appel à l'utilisation d'opérations fondamentales (tels que portes ET, OU, NOT...) avec le souci d'en minimiser le nombre.
Les circuits combinatoires sont généralement très rapides, et un tel avantage est d'une grande importance, puisque l'un des buts est d'avoir des machines à grande vitesse d'exécution. La logique programmée permet de réaliser les mêmes fonctions que la logique câblée tout en apportant la souplesse de la programmation. Elle est plus avantageuse que la logique câblée lorsque le nombre de fonction réalisé par un système est important, en particulier dans le cas de transfère et de manipulation de données,

De cela, on déduit qu'avec cette deuxième technique, en peut concevoir des appareils de taille réduite et avec moins de dépenses mais leurs temps d'exécution sont plus longs. C'est d'ailleurs le seul inconvénient qu'on peut reprocher à cette technique de conception. Dans le but de nous initier à l'utilisation de tels composants, plus encore à nous initier à la commande programmée, il nous à été demander d’étudie la maquette de développement EASYPIC6 et de réalisé et des applications a base des microcontrôleurs PICs16FXXX.

Chapitre I : Présentation de la carte de développement EasyPIC

I-Introduction :

Le système de développement EasyPIC6 est un outil destiné à la programmation et à l'expérimentation sur les microcontrôleurs PICs. La carte inclue un programmateur avec mikroICD (débogueur intégré) servant d'interface entre le microcontrôleur et le PC. Il suffit d'écrire le code dans un des compilateurs, générer un fichier .hex puis programmer le microcontrôleur grâce au programmateur PIC. De nombreux modules embarqués, comme l'afficheur LCD graphique 128x64, l'afficheur LCD 2x16 l'afficheur LCD embarqué 2x16 LCD, le clavier numérique 4x4, l'extenseur de port …etc.

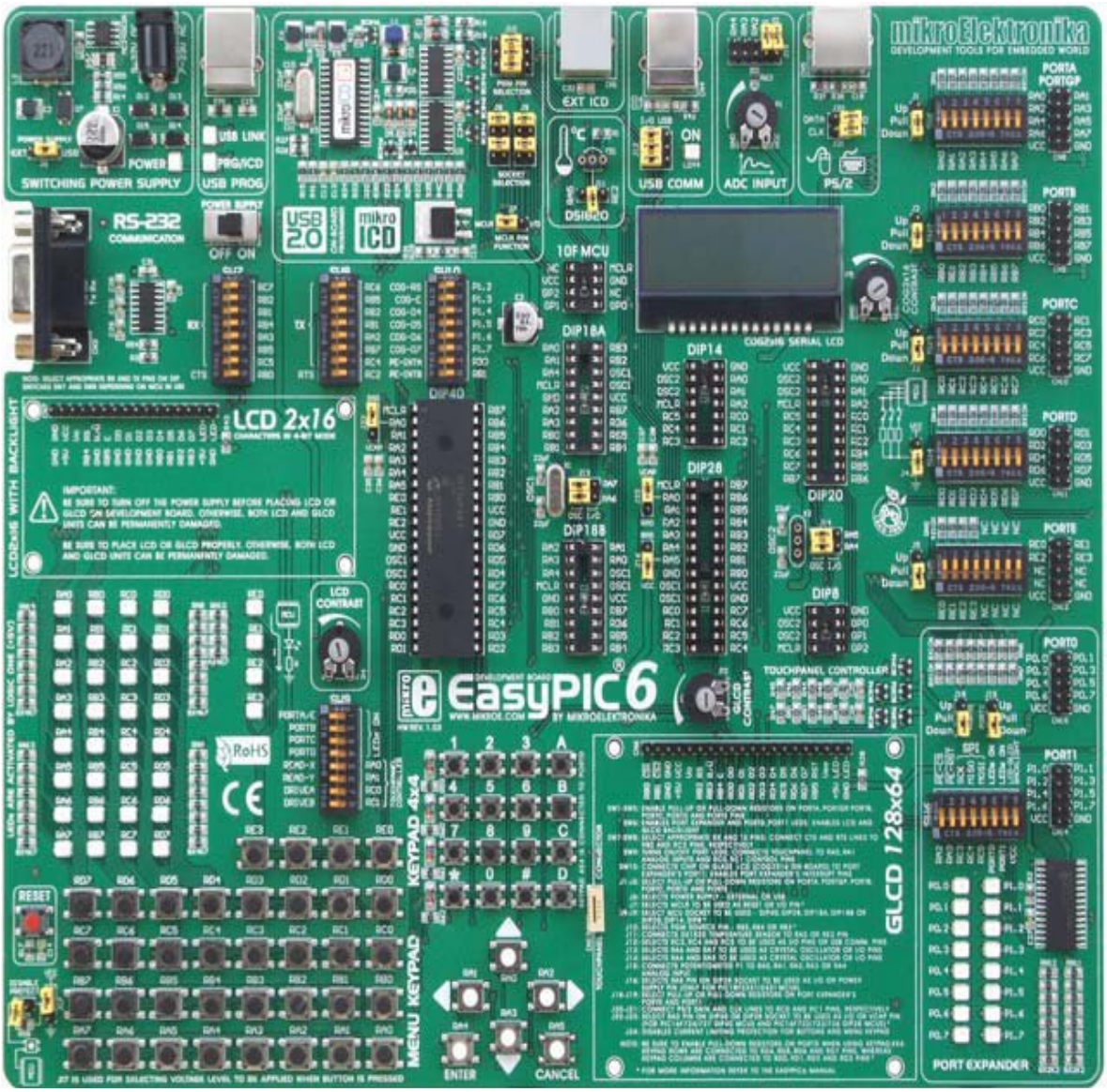
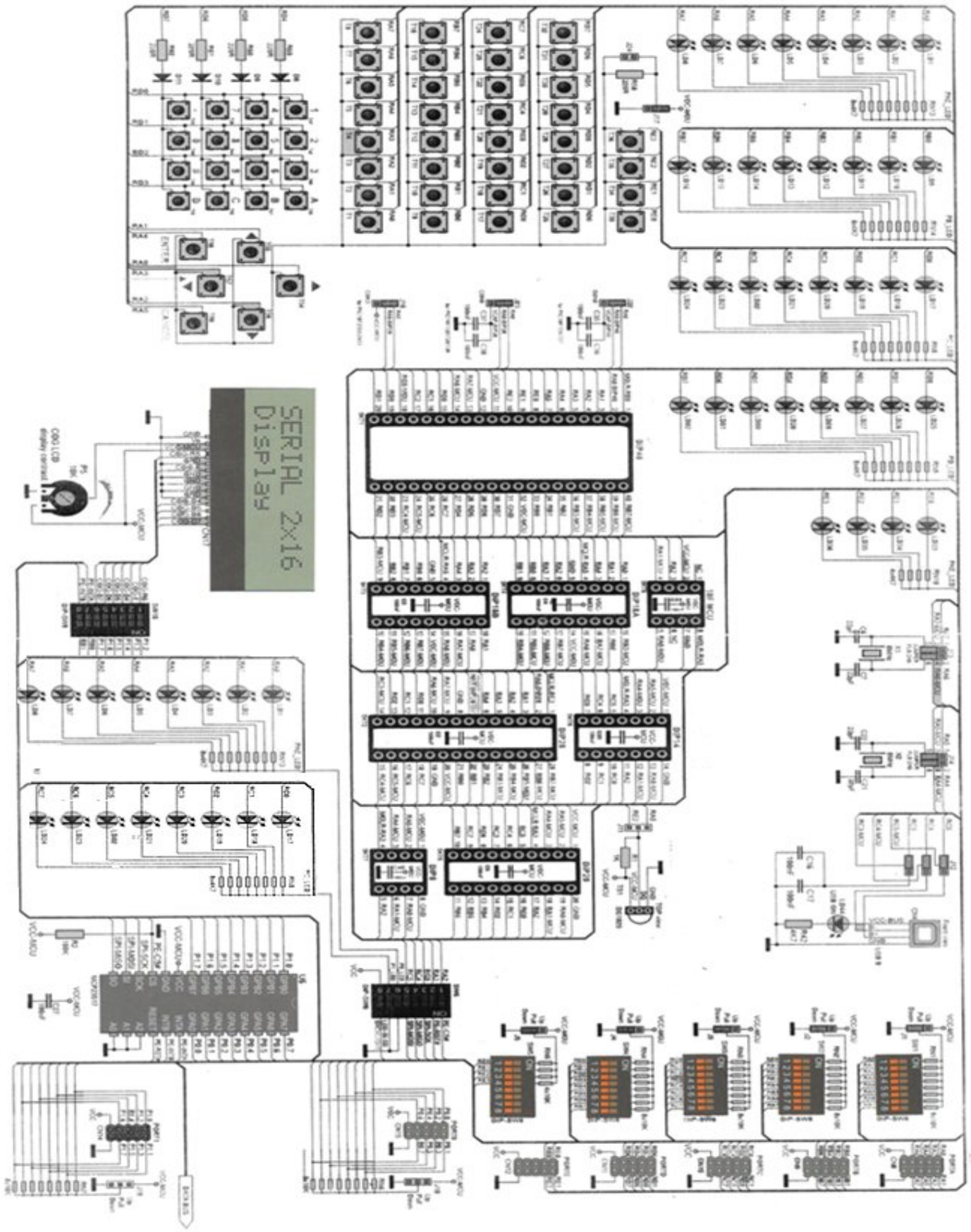


Fig. I-1 :
Vu de la carte EASYPIC

II-schéma électrique :



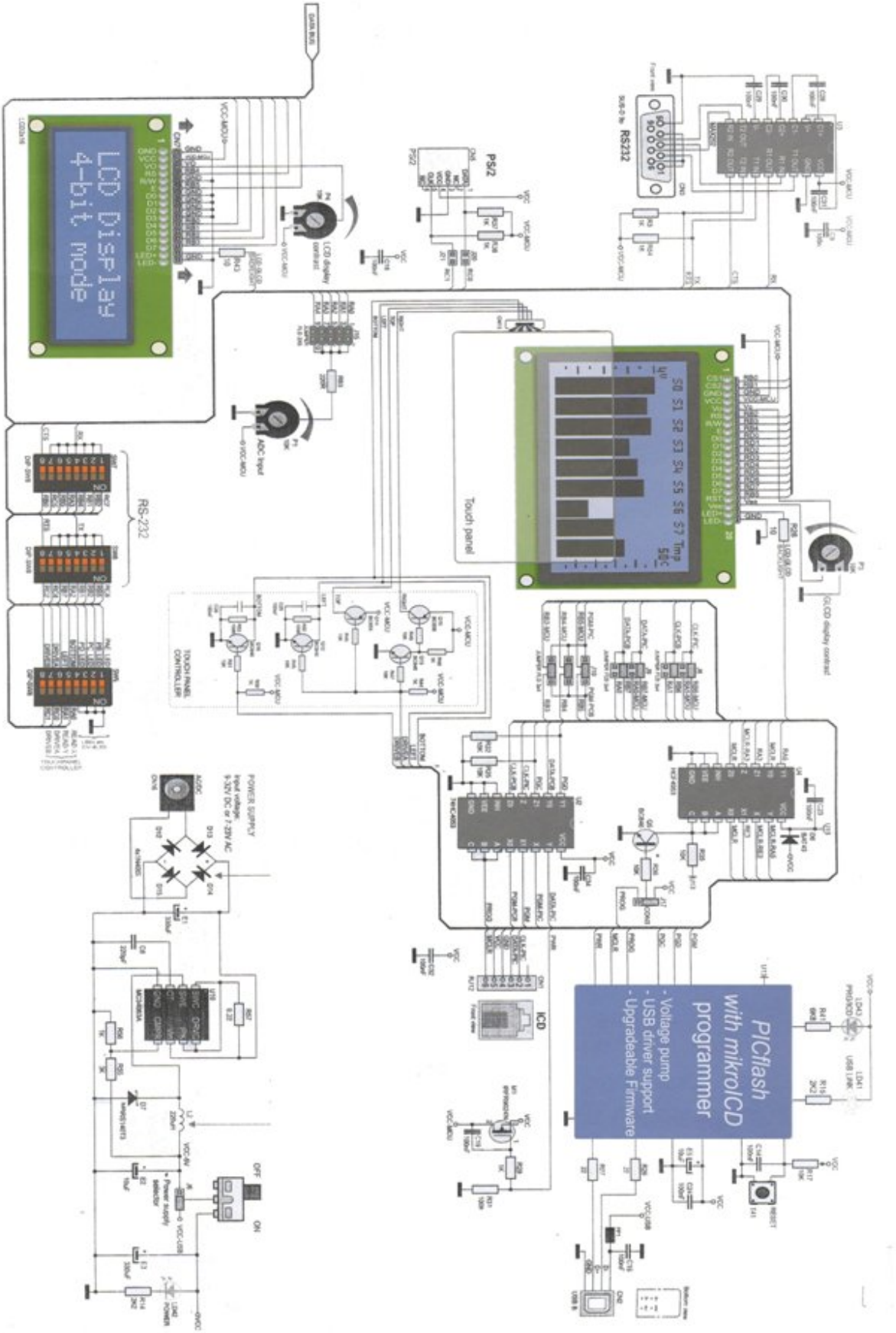


Fig. I-2 :
Schéma électrique de la carte

III-Les blocs de la carte :

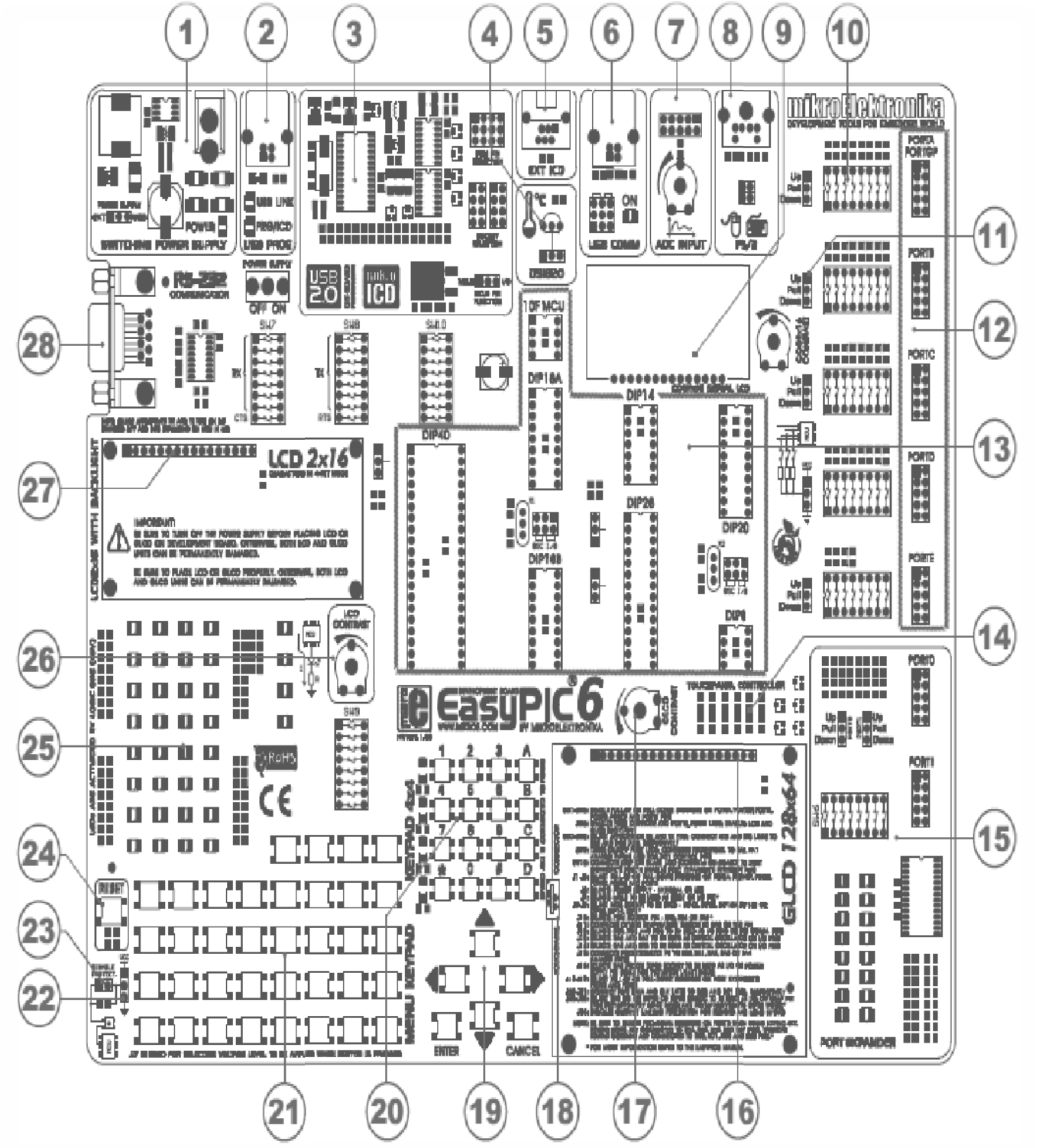


Fig. I- 3 :
Les blocs de la carte
EASYPIC6

1. Alimentation régulée en tension
2. Connecteur USB du programmateur embarqué
3. Programmateur USB 2.0 avec
4. Connecteur du capteur de température DS1820
5. Connecteur pour débogueur MICROCHIP (ICD2 ou ICD3)
6. Connecteur USB
7. Entrées de tests du convertisseur A/N
8. Connecteur PS/2
9. Afficheur LCD 2x16 embarqué
10. DIP Switchers pour activer les résistances de tirage
11. Sélection des modes pull-up/pull-down du port
12. Connecteurs E/S des ports
13. Interfaces de connections pour microcontrôleurs PIC
14. Contrôleur de l'écran tactile
15. Extenseur de port
16. Connecteur de l'afficheur LCD graphique 128x64
17. Potentiomètre de contraste de l'afficheur LCD graphique 128x64
18. Connecteur de l'écran tactile
19. Clavier numérique
20. Clavier numérique 4x4
21. Boutons poussoirs pour la simulation des entrées numériques
22. Sélecteur d'état logique
23. Jumper ON/OFF de la résistance de protection
24. Bouton reset
25. 36 LEDs d'indication de l'état logique des broches d'E/S
26. Ajusteur de contraste de l'afficheur LCD alphanumérique
27. Connecteur de l'afficheur LCD alphanumérique
28. Connecteur pour la communication RS-232

-Connection au PC

Un câble USB pour permet de connecter le système de développement EasyPIC6 au PC. Une extrémité du câble USB (USB devra être connecté à la carte de développement, tandis que l'autre extrémité (fiche USB A) devra être connectée à votre PC. Avant d'établir la connexion, assurez-vous que le Jumper J6 est bien placé sur la position USB

1-Alimentation :

Le système de développement EasyPIC6 supporte deux types d'alimentation électrique:

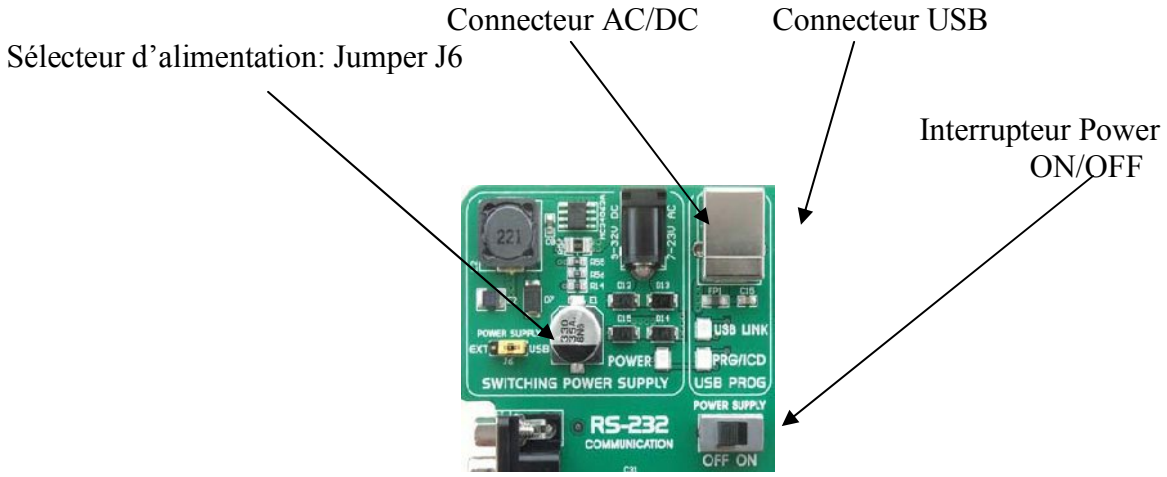


Fig. I-4: Alimentation

1-a-Alimentation externe :

Alimentation externe reliée au connecteur AC/DC de la carte de développement.
Le régulateur de tension MC34063A et le redresseur de Gretz autorisent une alimentation AC (entre 7V et 23V) et DC (entre 9V et 32V). Le Jumper J6 est utilisé pour la sélection du type d'alimentation. Lorsque la carte est alimentée par USB, le Jumper J6 doit être placé en position USB. Dans le cas où l'alimentation externe est utilisée, J6 doit être placé sur la position EXT. Pour mettre sous tension et éteindre le système de développement, on utilise l'interrupteur OFF/ON POWER SUPPLY.

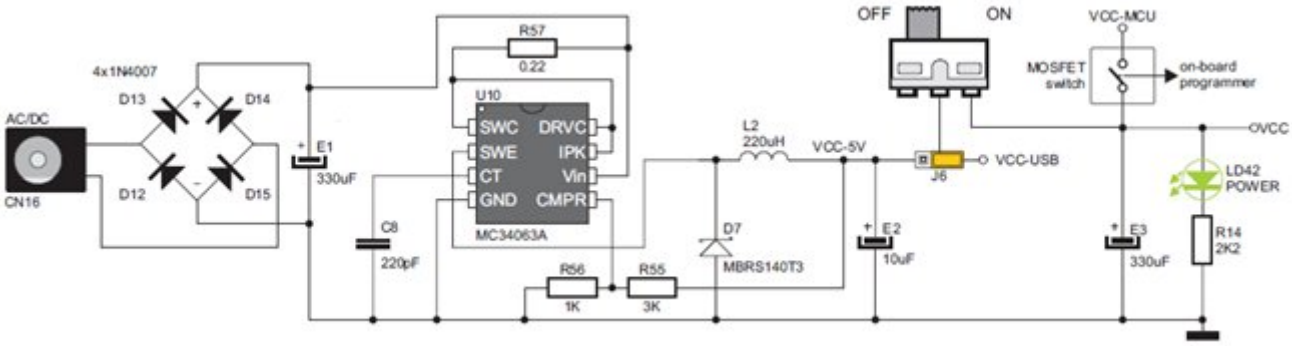


Fig. I- 5 : Alimentation externe

1-b- Alimentation par usb :

Le programmateur sert de l'interrupteur MOSFET pour interrompre l'alimentation du système de développement pendant la programmation. Une fois la programmation terminée, le programmateur autorise de nouveau le système à être alimenté.

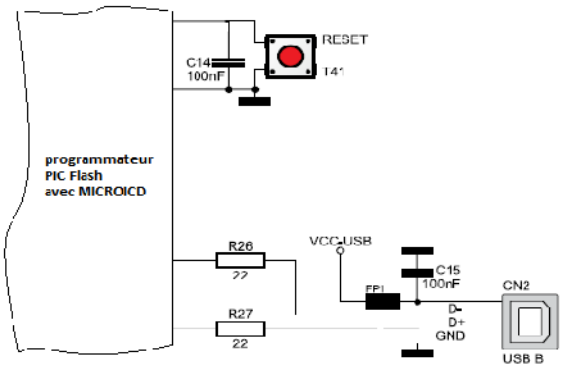


Fig. I- 6 : Alimentation par usb

2-Interfaces de connections pour microcontrôleurs PIC :

Le système de développement EasyPIC6 dispose de huit interfaces de connexion pour microcontrôleurs PIC différents: DIP40, DIP28, DIP20, DIP18, DIP14 et DIP8. La carte possède deux interfaces DIP 18 aux brochages différents: DIP18A et DIP18B.
Les Jumpers situés à coté des sockets servent à préciser la fonction de certaines broches du microcontrôleur.
Les microcontrôleurs PIC utilisent habituellement un quartz en cristal pour une meilleure stabilisation de la fréquence d'horloge. EasyPIC6 possède deux connecteurs pour quartz-cristal. Les microcontrôleurs connectés à DIP18A, DIP18B, DIP28 et DIP40 utilisent le quartz- cristal placé en X1 (OSC1) tandis que ceux placés en DIP8, DIP14 et DIP20 sont relié au quartz-cristal de X2 (OSC2). Il est évidemment possible de remplacer le quartz-cristal existant. Les microcontrôleurs branchés en 10F utilisent leur propre oscillateur interne et ne sont pas reliés au quartz mentionné précédemment.

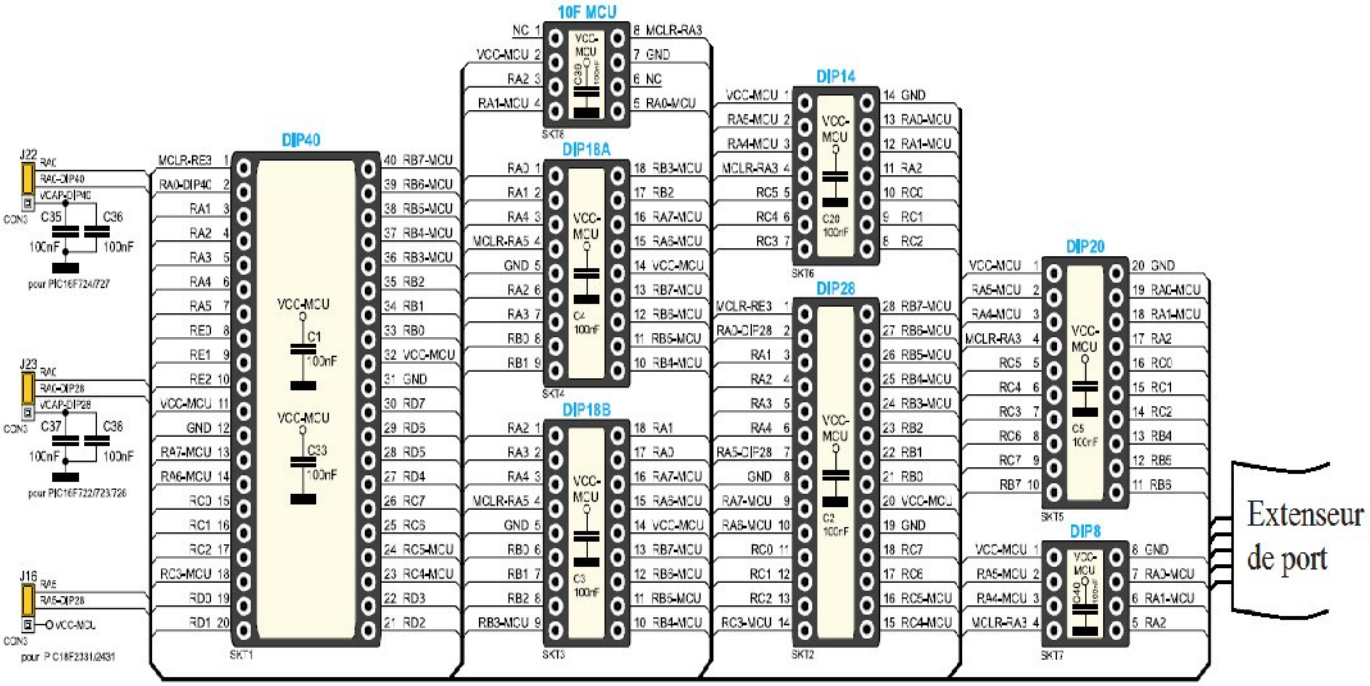


Fig. I-7 : Connection des PICs

3-Programmateur embarqué USB 2.0 avec mikroIC :

Le connecteur USB offre la possibilité de relier un périphérique aux microcontrôleurs dotés de module USB intégré. Pour activer la communication USB, il faut déplacer les Jumpers J12 afin de relier les lignes USB DATA (D+ i D-) aux broches RC4 et RC5 du microcontrôleur ainsi que les broches RC3 aux condensateurs C16 et C17. Si la communication USB n'est pas utilisée, les Jumpers J12 doivent être en position gauche. L'état de la communication USB (OFF/ON) est indiqué par la LED située à coté de J12.

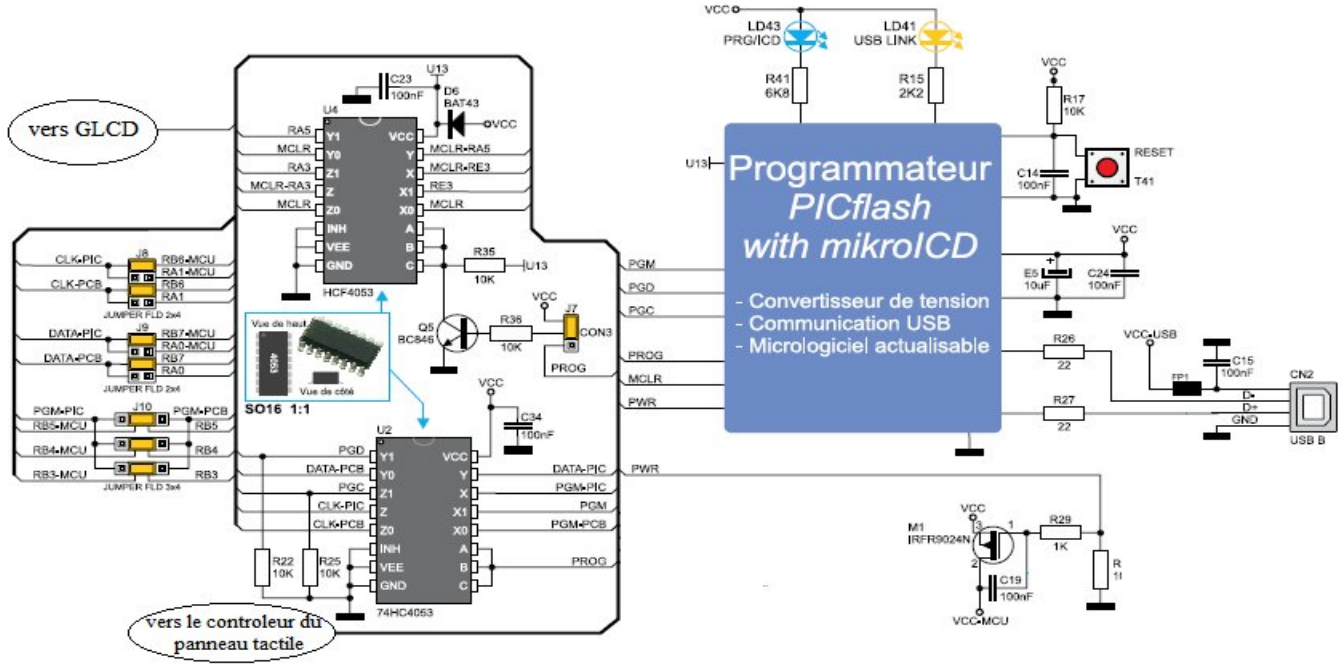


Fig. I- 8 : Programmateur embarqué

Le programmateur permet d'établir la connection entre le microcontrôleur et le PC. On l'utilise pour charger un fichier hexadécimal (fichier HEX) dans le microcontrôleur. La Figure 3-2 présente les liens entre le compilateur, le programmateur Picflash.

1. Ecrire un programme dans un compilateur PIC et générez le fichier HEX.
2. Utilisez le programmateur Picflash pour sélectionner le microcontrôleur approprié.
3. Cliquez sur le bouton (write) pour chargez le programme dans le microcontrôleur.
4. Le programmateur embarqué s'occupera du chargement du microcontrôleur

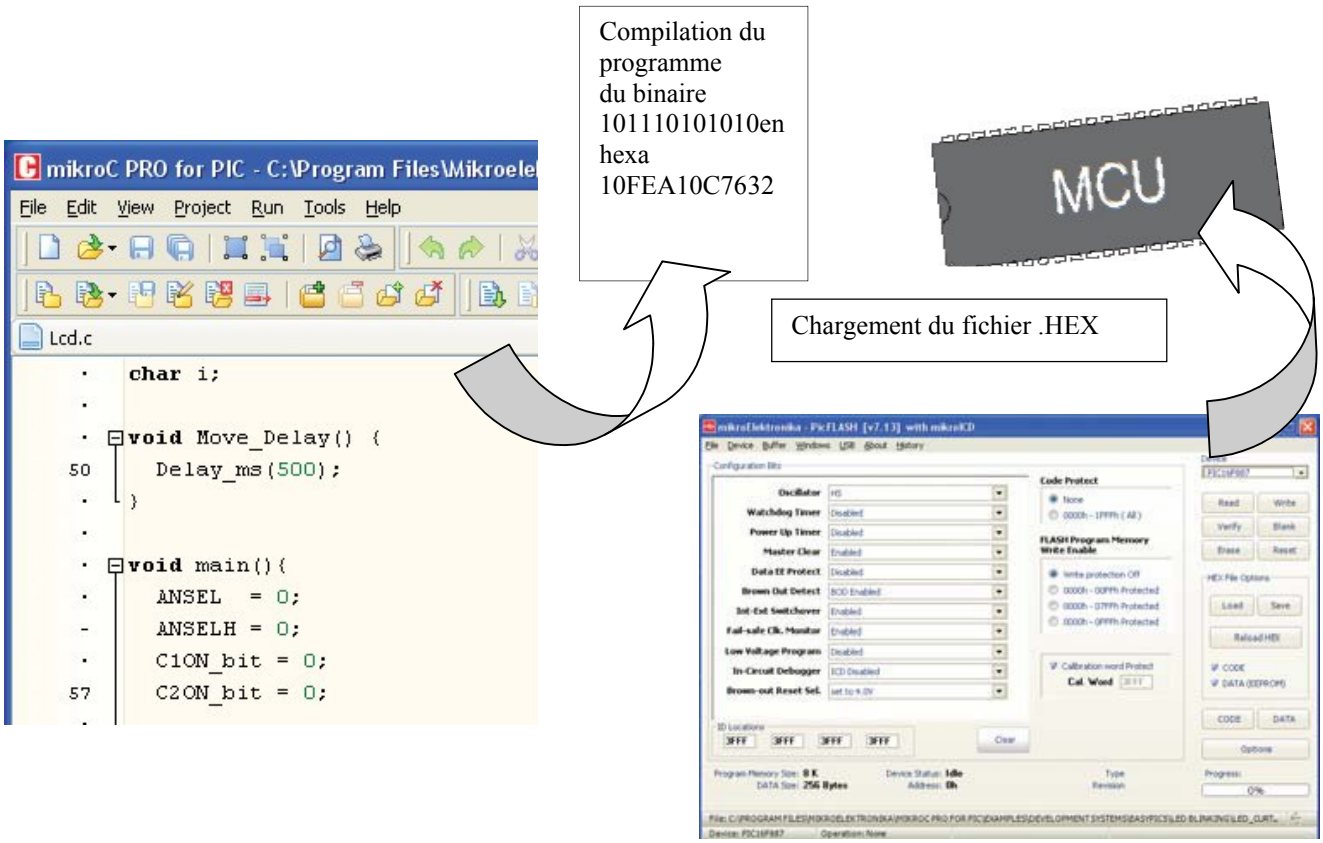


Fig. I- 9 : programmation des PICs

Sur la partie gauche de la fenêtre du programmateur Picflash il existe des différentes options de configuration du microcontrôleur. Des boutons facilitant la programmation sont disponibles sur la partie droite.

Une option en bas de la fenêtre indique la progression du processus de programmation.

Il existe deux modes de programmation des microcontrôleurs PIC : mode de programmation basse tension (Low Voltage) et haut tension (high voltage).le programmateur Picflash utilise exclusivement le mode high voltage, ce mode requière des tensions supérieur a la tension d 'alimentation de µc (de 8V a 14V en fonction du type de µc utilisé) sur la broche MCLR /Vpp afin de pouvoir effectue le processus de programmation.

Le mode de programmation basse tension peut être activé/désactivé grâce au bit de configuration du µc. C'est ce mode est activé, le processus de programmation débute par l'application d'un 1 logique sur la ligne PGM .dans le cas contraire c'est le mode haute tension qui est activé et le processus de programmation commence par l'application d'une tension élevé sur la broche MCLR/Vpp. Tous les µc PIC ont le mode Low voltage activé par défaut .dans de rares cas, afin d autoriser le µc a être programmé en mode high voltage, il est nécessaire d'appliquer un zéro a la ligne PGM ce que empêche le µc d'entré en mode Low voltage, le Jumper 10 sert a choisir la broche à utiliser comme PGM figure1

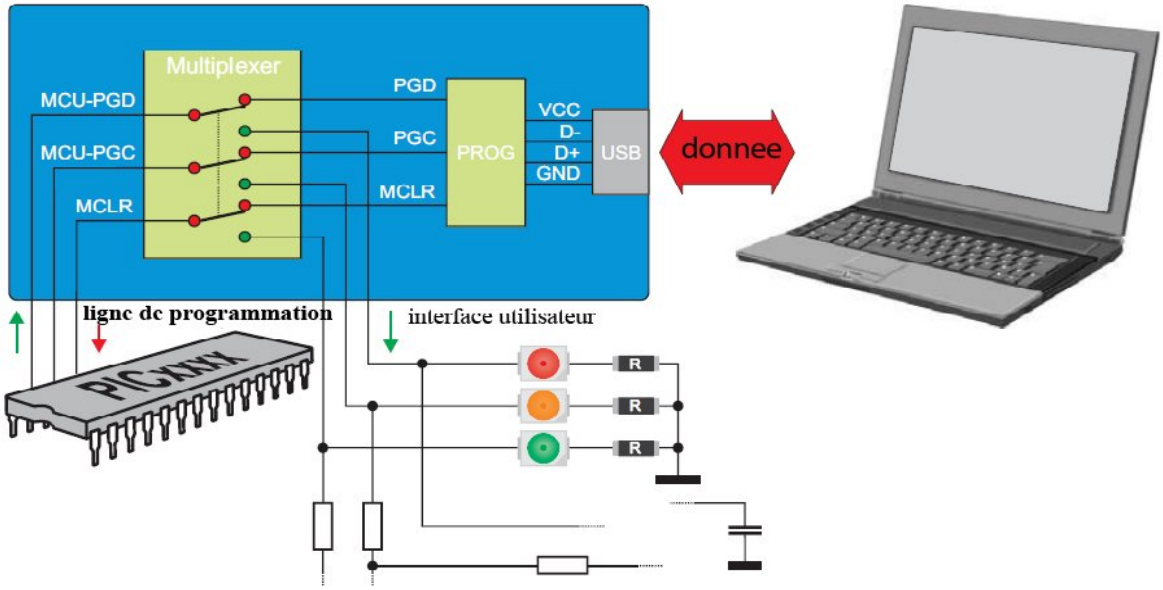


Fig. I-10 : schema du programmateur

Pandant la programmation,un multiplexeur deconnecte les broches de µc utilisées pour la programmation du reste de la carte,et les connecte au programmateur PICflash.une fois la programmation achevée,ces broches sont deconnectées du programmateur et peuvent ainsi etre utilisées comme des lignes
d'entrée/sortie
Le mikroICD fait partie integrante du programmateur embarqué,son role est de tester et de debuguer les programmes en temps reel , ce processus s'effectue en affichant l'état des registres et variables du µc au cour de l'éxécution d'un programme.
Aussitôt le débogueur mikroICD lancé, la fenêtre ci-dessous apparaît à l'écran.
Le débogueur mikroICD communique avec le microcontrôleur par les broches de programmation. Par conséquent, ces dernières ne peuvent pas être utilisées comme broches E/S lors du débogage.

4-Interface de communication :

4-1-Communication par RS232

La communication série RS-232 est réalisée par l'intermédiaire du connecteur 9-broches SUB-D et du module USART du microcontrôleur.
Pour rendre possible une telle communication, il faut connecter les lignes de communication RX et TX aux broches du microcontrôleur contrôlant le module USART. Ceci s'effectue par l'intermédiaire des DIP switches (les lignes CTS et RTS étant optionnelles). Les broches du microcontrôleur utilisées pour ce type de communication sont les suivantes:
RX - donnée reçue, TX - donnée transmise, CTS - send (accusé de réception de la requête) et RTS - Request To Send (Requête d'envoi). Le débit atteint 115 kbps.
USART (universal synchronous/asynchronous receiver/transmitter)) est un des moyens les plus fréquemment utilisé pour l'échange de données entre le PC et les périphériques. Pour activer le module USART du microcontrôleur afin de recevoir des signaux d'entrées de tensions différentes, l'utilisation d'un convertisseur de tension tel que le MAX-202C est indispensable.
Les DIP switches SW7 et SW8 servent à déterminer quelles broches du microcontrôleur doivent être utilisées en réception et transmission. Les broches de sortie du microcontrôleur dépendent du type de microcontrôleur utilisé. SW7: RX, CTS = ON, SW8: TX, RTS = ON

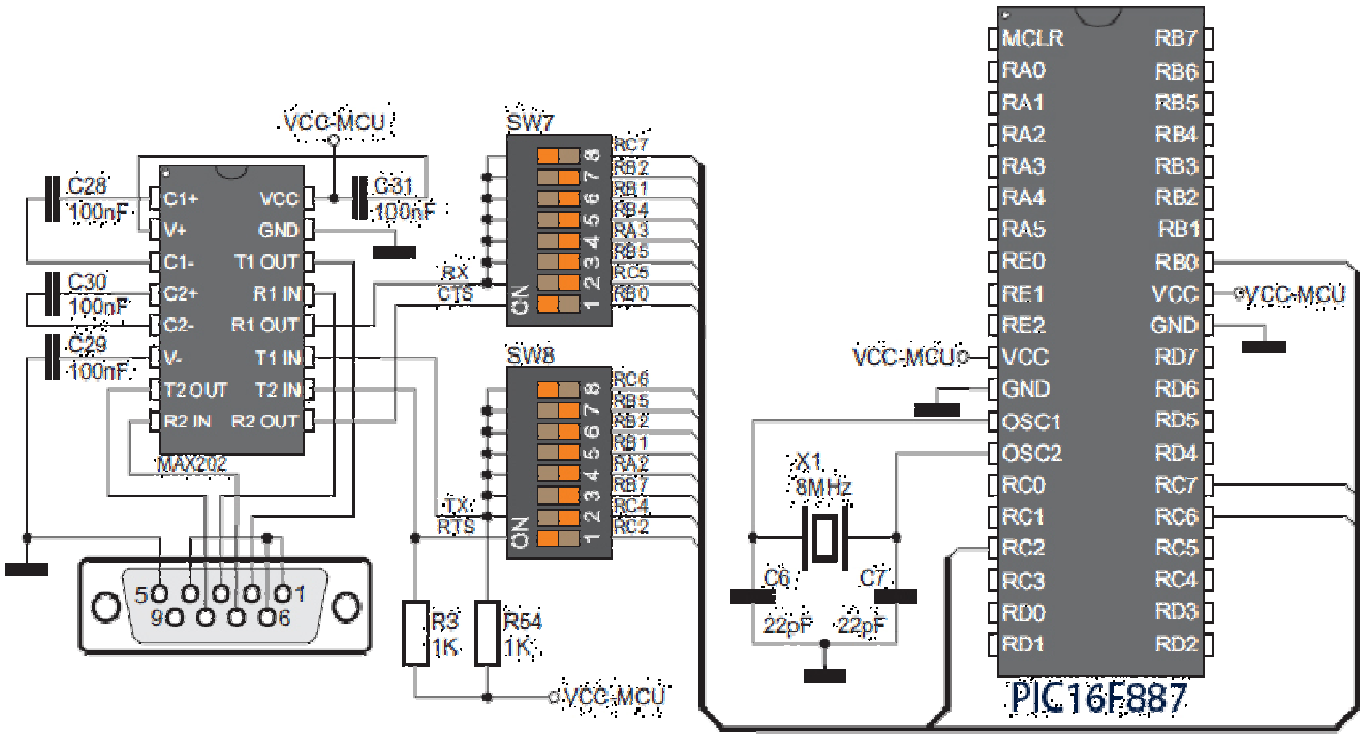


Fig. I- 11 :connection par RS232

4-2-communication par PS /2 :

Un périphérique comme un clavier ou une souris peut être connecté via le connecteur PS/2. L'activation de la communication PS/2 s'effectue en positionnant correctement les Jumpers J20 et J21, c'est à dire en reliant les lignes DATA et CLK aux broches RC0 et RC1 du microcontrôleur. Ne jamais connecter/déconnecter le périphérique lorsque le système de développement est allumé.

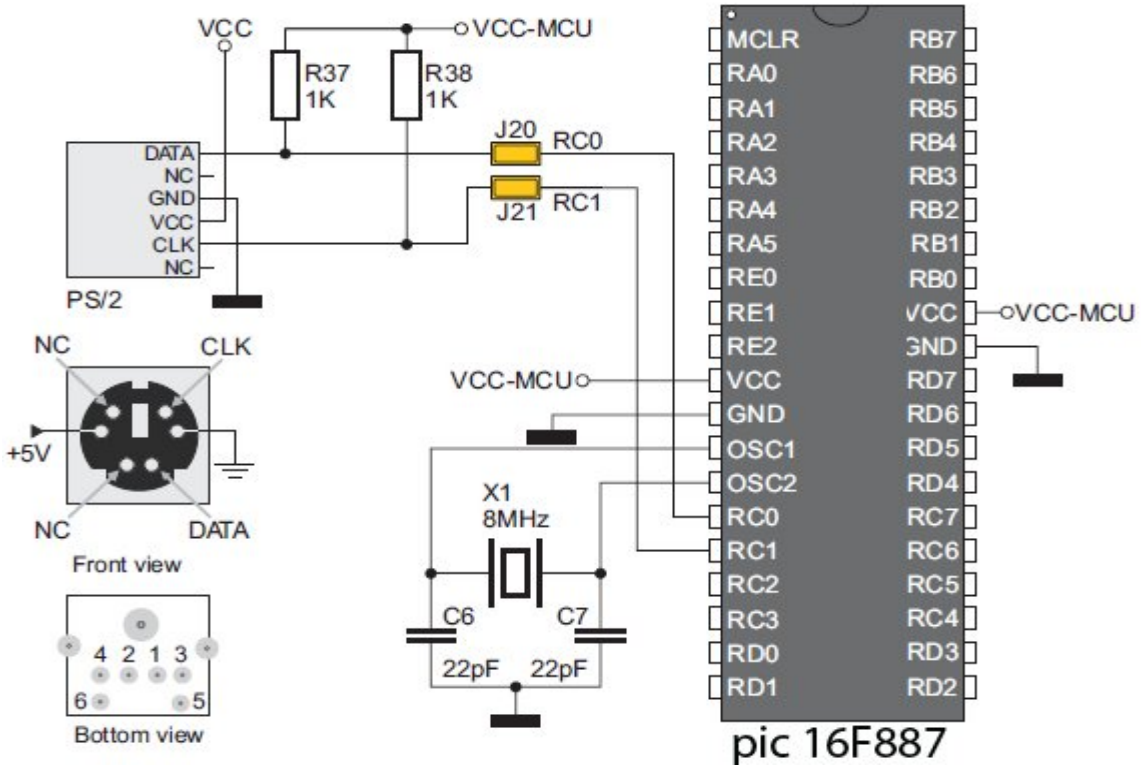


Fig. I-12 :connection par PS/2

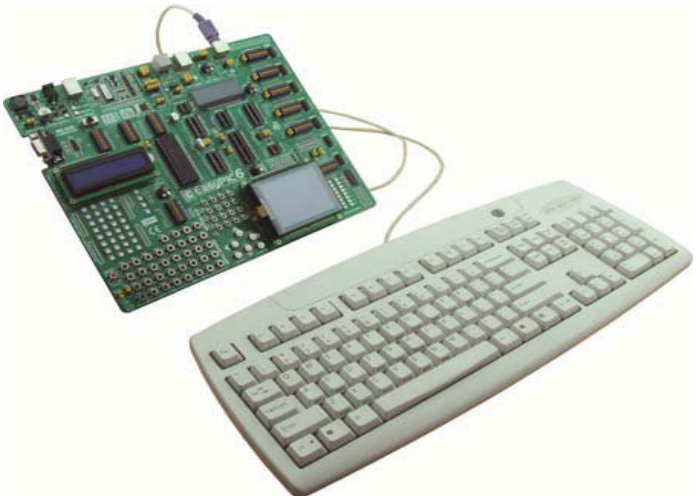


Fig. I- 13 :connection du clavier par PS/2

5-Boutons poussoirs :

L'état logique de toutes les entrées numériques du microcontrôleur peuvent être modifiées par l'intermédiaire des boutons poussoirs. Le Jumper J17 sert à définir l'état logique qui doit être appliqué à la broche du microcontrôleur lorsque le bouton associé est pressé. Des résistances de protection ont pour but de limiter le courant maximum afin de prévenir d'éventuel court-circuit. On à la possibilité de court-circuiter ces résistances grâce au Jumper J24. A coté de ces boutons poussoirs, on trouve le bouton RESET. Celui-ci n'est pas relié directement à la broche MCLR. Le signal reset (réinitialisation) est généré par le programmateur.

Pour n'importe quel bouton poussoir (R0-R7) lorsque J17 est dans la position VCC-MCU, un 1 logique
(5V) sera appliqué à la broche associée du microcontrôleur.

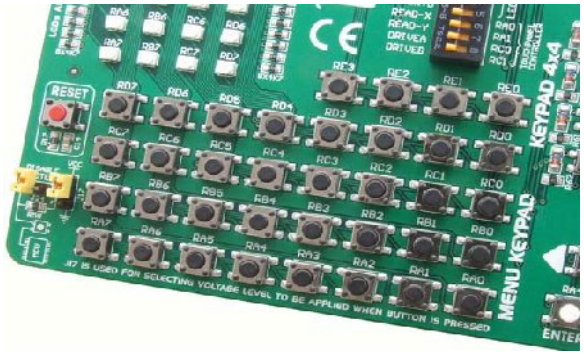


Fig. I- 14 : Les Boutons poussoirs

6-les claviers :

Le système de développement EasyPIC6 comporte deux claviers numériques (keypads) clavier 4x4 et clavier MENU, le clavier 4x4 est un clavier alphanumérique standard relié au PORTD du microcontrôleur, où RD0, RD1, RD2 et RD3 sont configurées en entrées et connectées aux résistances de pull-down (reliées à la masse). Les broches RD4, RD5, RD6 et RD7 sont configurées en sorties de niveau haut (1 logique). La pression d'un des boutons va causer l'application d'un 1 logique sur une des broches d'entrées. La détection du bouton pressé s'effectue par software. Afin de déterminer quelle bouton poussoir a été pressé, un 1 logique sera appliqué successivement à chacune des broches de sorties (RD4, RD5, RD6 et RD7). Par exemple, si le bouton '6' est pressé, un 1 logique va apparaître en RD2 lorsqu'un 1 logique sera envoyé sur la sortie RD5.

Les boutons du clavier MENU sont connectés au PORTA de manière similaire. La seule différence est dans l'agencement du clavier numérique. Les boutons du clavier MENU sont disposés de manière à faciliter la navigation à travers les menus.

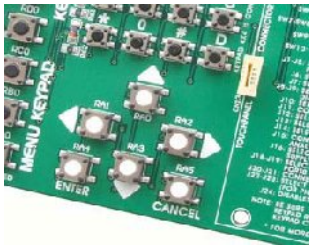


Fig. I- 15-2: clavier MENU

Fig. 15-1: clavier4x4

7-Les afficheurs LCD 2x16 :

La carte de développement EasyPIC6 est équipée d'un connecteur pour afficheur LCD alphanumérique 2x16. Ce connecteur est relié au microcontrôleur via le PORTB. Le potentiomètre P4 est utilisé pour l'ajustement du contraste. L'interrupteur 8 du DIP Switch SW6 sert à allumer/éteindre le rétro éclairage de l'afficheur .La communication entre le microcontrôleur et l'afficheur LCD s'effectue en mode 4 bits4. Les symboles alphanumériques sont représentés sur deux lignes, chacune pouvant contenir jusqu'à 16 caractères de taille7x5 pixels. L'afficheur 2x16 embarqué est intégré à la carte et relié au microcontrôleur via l'extenseur de port. Pour utiliser cet afficheur, il faut placer toutes les lignes du DIP Switch SW10 sur la position ON, ce qui équivaut à connecter l'afficheur à l'extenseur de port 1. C'est le DIP Switch SW6 qui permet l'utilisation de la communication série via l'extenseur de port. Le potentiomètre P5 sert à l'ajustement du contraste. Contrairement aux afficheurs LCD classiques, cet afficheur ne possède pas de rétro éclairage et reçoit les données à afficher par l'extenseur de port. Ce dernier emploie la communication SPI pour communiquer avec le microcontrôleur.

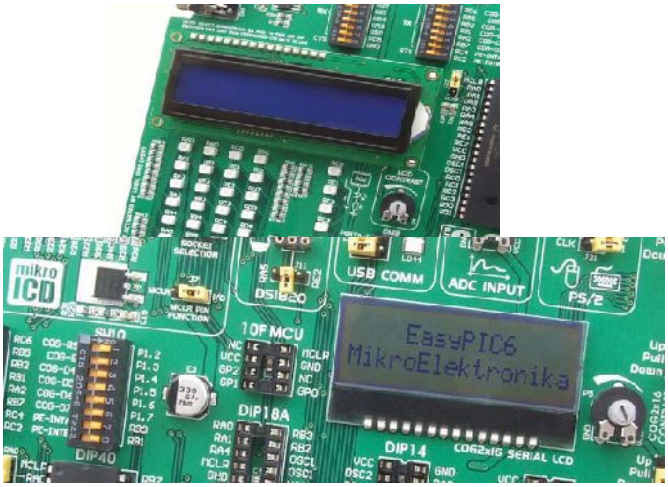


Fig. I- 16-1: Afficheur LCD 2x16 embarqué

Fig. I- 16-2: Afficheur LCD 2x16

8-L'afficheur graphique LCD 128x64 :

L'afficheur graphique LCD 128x64 (128x64 GLCD) permet l'affichage de données graphique complexes. Il communique avec le microcontrôleur par l'intermédiaire des PORTB et PORTD. L'afficheur GLCD a une résolution de 128x64 pixels, ce qui rend possible l'affichage de diagrammes, tableaux et autres graphiques. Comme le PORTB est également utilisé par l'afficheur LCD alphanumérique 2x16, vous ne pourrez pas vous servir simultanément de ces deux afficheurs. Le potentiomètre P3 sert à ajuster le contraste de l'afficheur GLCD. L'interrupteur 8 du DIP Switch SW6 est utilisé pour activer/désactiver le rétro éclairage.

SW6: CS#, RST, SCK, MISO, MOSI = ONSW10: 1-8 = ON

Fig. I-7: Afficheur GLCD

9-Panneau tactile (touch panel) :

Le touch panel est un panneau fin, adhésif et transparent sensible au toucher. Il se place sur l'afficheur GLCD et le "transforme" ainsi en écran tactile. Son intérêt réside dans le fait qu'il enregistre les pressions effectuées sur l'afficheur et envoie ses coordonnées sous forme d'un signal analogique au microcontrôleur. Les interrupteurs 5, 6,7 et 8 du DIP Switch SW9 sont utilisés pour relier le touch panel au microcontrôleur.

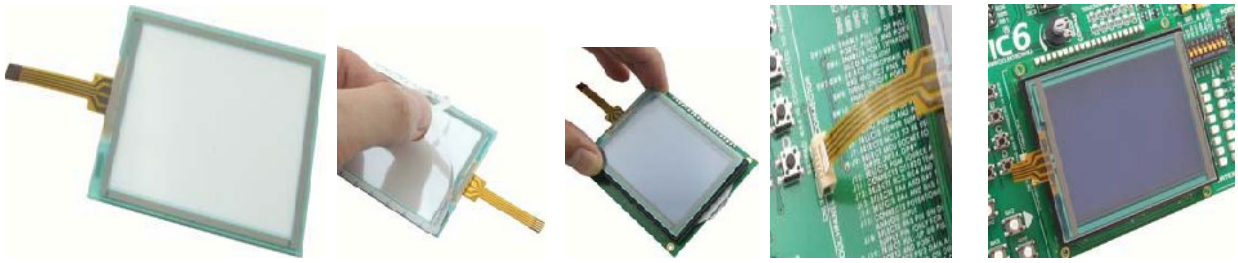


Fig. I- 18 : Panneau tactile

-Note :

La Figure I-18 montre comment placer le touch panel sur l'afficheur GLCD. Assurez que le câble plat est situé sur la gauche de l'afficheur GLCD.
Les LEDs et les résistances de tirage pull-up/pull-down des broches RA0 et RA1 du PORTA doivent être désactivées pendant l'utilisation du touch panel.

10-Ports d'Entrées/Sorties

Sur la partie droite de la carte de développement se trouve sept DIP switches ayant chacun dix interrupteurs. Chacun d'eux est connecté aux ports d'E/S du microcontrôleur. Certaines des lignes d'E/S sont directement reliées aux broches du microcontrôleur tandis que d'autres sont connectées au microcontrôleur par l'intermédiaire de Jumpers.
Les DIP switches SW1-SW5 servent à relier les broches aux résistances de tirage pull-up/pull-down. La position des Jumpers J1-J5 détermine si les ports utilisent une résistance de pull-up ou de pull-down.

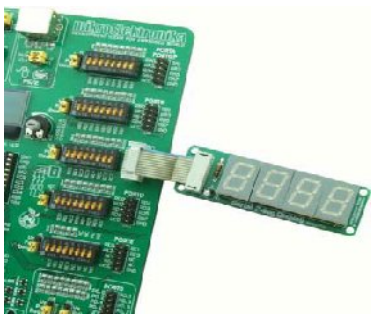


Fig. I- 19: Ports E/S

Les résistances de Pull-up/pull-down vous permettent de forcer l'état logique de toutes les entrées du microcontrôleur. Cet état dépend de la positon des Jumpers J2 et J17. La broche RB0 et son DIP

Switch associé SW2, le Jumper J2 et le bouton poussoir RB0 avec le Jumper J17 sont utilisés ici dans le but d'expliquer le rôle des résistances de tirage. Leur principe de fonctionnement est le même pour toutes les broches du microcontrôleur.
Afin de permettre la connections des broches du PORTB aux résistances de pull-down, il faut mettre le Jumper J2 en position basse (à la masse). Ainsi, toutes les broches du PORTB se verront attribuer un zéro logique (0V) par l'intermédiaire du Jumper J2 et de la résistance 8x10K. Pour fournir un tel signal à la broche RB0, il est indispensable de placer l'interrupteur de la broche RB0 du DIP Switch SW2 sur la position ON. Le Jumper J17 en position VCC (pull-up) En conséquence, chaque fois que vous presserez le bouton poussoir RB0, un 1logique apparaîtra sur la broche RB0.Ce 1 logique provient du fait que le J17 est en position VCC.

Afin de permettre la connections des broches du PORTB aux résistances de pull-up, il faut tout mettre le Jumper J2 en position haute (VCC). Ainsi, toutes les broches du PORTB se verront attribuer une 1 logique (5V) par l'intermédiaire du Jumper J2 et de la résistance 8x10 K. Pour fournir un tel signal à la broche RB0, il est indispensable de placer l'interrupteur de la broche RB0 du DIP Switch SW2 sur la position ON. le Jumper J17 en position basse (pull-down) .En conséquence, chaque fois que vous presserez le bouton poussoir RB0, un zéro logique (0) apparaîtra sur la broche RB0.Ce 0 logique provient du fait que le Jumper J17 est placé en position GND (à la masse).
Dans le cas où les J2 et J17 occupent la même position, la pression d'un bouton poussoir n'entraînera aucune modification de l'état logique sur les broches d'entrées du microcontrôleur.

11-Extenseur de port :

Les lignes de communications SPI et le circuit MCP23S17 donnent la possibilité d'augmenter de deux le nombre de ports d'E/S disponibles. Si l'extenseur de port est connecté par l'intermédiaire du DIP Switch SW6, les broches RA2, RA3, RC3, RC4 et RC5 seront utilisées pour la communication SPI, et par conséquent, ne pourront pas être utilisées comme E/S. Les interrupteurs INTA et INTB du DIP Switch SW10 activent les interruptions. Le MCP23S17 permet l'extension 16-bit. Il peut être configuré pour fonctionner en 8 ou 16 bits.

Fig. I-20-1:Extenseur de port

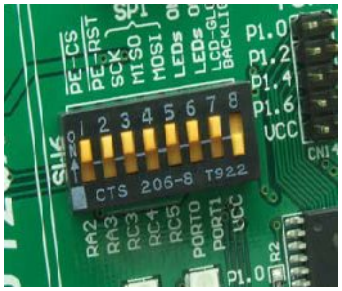


Fig. I-20-2 : DIP Switch SW6

12- Convertisseur Analogique/Numérique (ADC) :

Un convertisseur Analogique/Numérique a pour rôle de convertir un signal analogique en données numériques. Le convertisseur A/N est linéaire ce qui signifie que la donnée convertie est linéairement dépendante à la tension d'entrée. Le convertisseur A/N situé dans le microcontrôleur convertit une tension analogique en un nombre codé sur 10 bits. Une tension d'entrée variant entre 0V et 5V DC peut être fournit par les entrées de test A/N. Le Jumper J15 permet de sélectionner la broche sur laquelle la conversion aura lieu, à savoir RA0, RA1, RA2, RA3 ou RA4. La résistance R63 a une fonction de protection dans la mesure où elle sert à limiter le courant à travers le potentiomètre et les broches du microcontrôleur. La valeur de l'entrée analogique peut être modifiée linéairement en utilisant le potentiomètre P1.

Fig. I- 21: ADC (position du Jumper par défaut)

Note:

Afin d'obtenir une conversion A/N précise, il est nécessaire d'éteindre les LEDs et de déconnecter les Jumpers des résistances de tirages des broches utilisées par le convertisseur A/N.

13-Les LEDs :

Une diode LED (Light-Emitting Diode) est un composant électronique particulier utilisé comme source de lumière. Pour chacune des LEDs, l'utilisation d'une résistance pour limiter le courant est indispensable. Sa valeur est déterminée grâce à la formule R=U/I où R correspond à la valeur de la résistance en ohms, U la tension et I le courant traversant la LED. La tension aux bornes d'une LED standard est de l'ordre de 2.5V tandis que le courant peut varier entre 1mA et 20mA. L'intensité du courant qui traverse les LED d'EasyPIC6 est de 1mA.

EasyPIC6 possède 36 LEDs dont le but est d'indiquer visuellement (présence ou non de lumière) l'état de chacune des broches E/S du microcontrôleur. Une LED allumée (active) traduit la présence d'un un logique (1) sur la broche. Le DIP Switch SW9 pour choisir les ports (PORTA/E, PORTB, PORTC ou PORTD) .

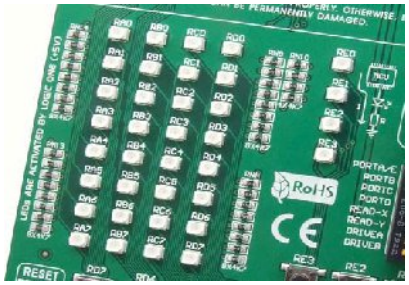


Fig. I- 22-1: LEDs

14-Le connecteur ICD :

Le connecteur ICD (In-Circuit Debugger) permet de relier le microcontrôleur à un débogueur ICD MICROCHIP externe (ICD2 or ICD3). Les Jumpers J8 et J9 doivent être placés de la même façon que lorsque le programmateur Picflash avec mikroICD intégré de MikroElektronika est utilisé.

15-Capteur de température DS1820 :

La communication série one-wire permet la transmission de données par l'intermédiaire d'une seule ligne de communication, ce qui signifie qu'une seule broche du microcontrôleur est utilisée. Le processus est sous le contrôle du microcontrôleur (maître). Tous les périphériques esclaves ont par défaut un unique code ID. Ceci permet au maître d'identifier facilement les périphériques esclaves partageant la même interface.

Le capteur de température DS1820 utilise le standard one-wire. Il est capable de mesurer les températures comprises entre -55 et 125°C avec une marge d'erreur de ±0.5°C. Ce capteur requiert une tension d'alimentation comprise entre 3V et 5.5V. La température est calculée avec une

résolution de 9 bits en moins de 750ms. Le système de développement EasyPIC6 est équipé d'une interface de connection destinée au DS18X20. Le capteur peut utiliser RA5 ou RE2 comme broche de communication avec le microcontrôleur. La sélection de la broche assurant la communication one-wire avec le microcontrôleur se fait grâce au Jumper J11. La Figure I-23 présente une communication one-wire avec le microcontrôleur via RA2.

-Note :

Le demi cercle inscrit sur la carte correspond au coté arrondi du DS1820

Fig. I- 23 : capteur DS1820

Chapitre II :
le PIC16F887

I-Introduction:

Le PIC16F887 est un produit de Microchip. Il comporte tous les modules des microcontrôleurs modernes. Il est généralement moins puissant qu'un microprocesseur en terme de rapidité ou de taille mémoire, Les Pics sont des microcontrôleurs à architecture RISC (Reduce Instructions Construction Set), ou encore composant à jeu d'instructions réduit. L'avantage est que plus on réduit le nombre d'instructions, plus leur décodage sera rapide ce qui augmente la vitesse de fonctionnement du microcontrôleur.

La famille des PICs est subdivisée en 3 grandes familles : La famille **Base-Line**, qui utilise des mots d'instructions de 12 bits, la famille **Mid-Range**, qui utilise des mots de 14 bit set la famille **High-End**, qui utilise des mots de 16 bits.

Les PICs sont des composants STATIQUES, Ils peuvent fonctionner avec des fréquences d'horloge allant jusqu'à une fréquence max spécifique à chaque circuit. Un PIC16F887 peut fonctionner avec une horloge allant du continu jusqu'à 20 MHz.

II-1-Les broches du pic16f887 :

De diverses fonctions de broches ne peuvent pas être employées simultanément, mais peuvent être changées à un point quelconque lors du fonctionnement.

Les tables suivantes se rapportent au microcontrôleur de DIP40 PIC16F887.

Pour configurer une broche comme une entrée, le bit approprié des registres d'ANSEL ou ANSELH doit être a (1).

Pour la configurer comme une entrée, le bit approprié doit être à (0).

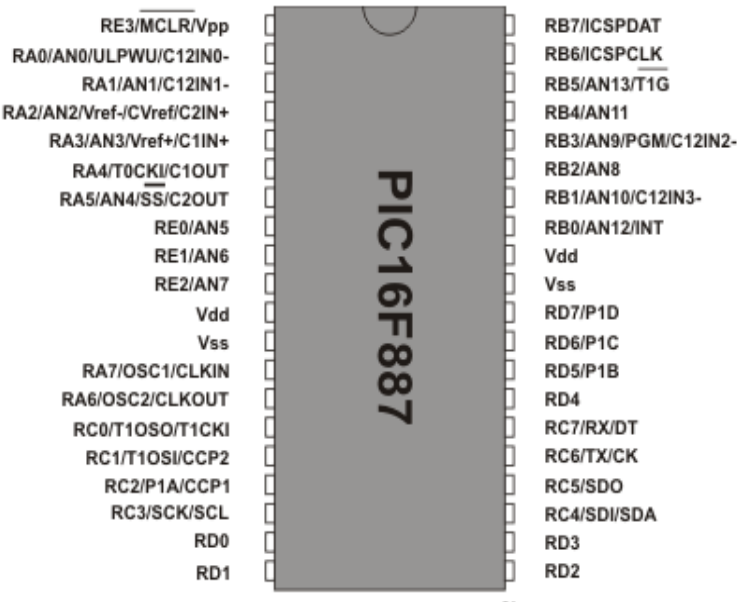


Fig. II- 1 : broches du PIC16F8877

III-1-Les particularités électriques :

Le schéma concernant le 16F887, on trouve 2 connexions « VSS » qui sont reliées à la masse, en interne, ces pins sont interconnectés. La présence de ces 2 pins s'explique pour une raison de dissipation thermique. Les courants véhiculés dans le pic sont loin d'être négligeables du fait des nombreuses entrées/sorties disponibles.

Le constructeur a donc décidé de répartir les courants en plaçant 2 pins pour l'alimentation VSS, bien évidemment, pour les mêmes raisons, ces pins sont situées de part et d'autre du PIC, et en positions relativement centrales.

Ce PIC fonctionne si vous décidez de ne connecter qu'une seule de chacune des pins, mais on peut exposerez à une destruction du composant lors des fortes charges, suite au non respect de la répartition des courants internes.

Nous noterons l'habituelle connexion de MCLR au +5V, cette pin étant utilisée pour effectuer un reset du composant en cas de connexion à la masse.

Nous trouvons également le quartz, qui pourra être remplacé par un résonateur ou par un simple réseau RC, de la même manière que pour le 16F84.

Les condensateurs de découplage, du fait de la fréquence plus importante du quartz utilisé ici (20MHz) seront de valeur inférieure à celle utilisée pour le 16F84 (à 4Mhz), soit 15pF.

La tolérance sur ces composants permet d'utiliser d'autres valeurs, mais c'est cette valeur qui permet le fonctionnement le plus fiable à cette fréquence. Certains composants ont en effet refusé de fonctionner avec des valeurs de 27pF avec ce quartz. Aucun n'a refusé de fonctionner avec la valeur de 15pF.

Name	Numéro (DIP 40)	Fonction	Description
RE3/MCLR/Vpp	1	RE3	General purpose input port E
		MCLR	Rest pin low logic level on this pin resets microcontroller
		Vpp	Programming Voltage
RA0/AN0/ULPWU/ C12INO	2	RA0	General purpose I/O port A
		AN0	A/D channel 0 input
		ULPWU	Stand-by mode deactivation input
		C12INO-	Comparator C1 or C2 negative input
RA1/AN1/C12N1-	3	RA1	General purpose I/O port A
		AN1	A/D channel 1
		C12N1-	Comparator C1 or C2 negative input
RA2/AN2/Vref- /CVref/C2IN+	4	RA2	General purpose I/O port A
		AN2	A/D channel 2
		Vref-	AD negative voltage Reference input
		CVref	Comparator voltage Reference Output
		C2IN+	Comparator C2 Positive input
RA3/AN3/Vref+/C1I N+	5	RA3	General purpose I/O port A
		AN3	A/D channel 3
		Vref+	A/D positive voltage Reference input
		C1IN+	Comparator C1 positive input
RA4/T0CKI/C1OUT	6	RA4	General purpose I/O port A
		T0CKI	Timer T0 clock input
		C1OUT	Comparator C1 output
RA5/AN4/SS/ C2OUT	7	RA5	General purpose I/O port A
		AN4	A/D channel 4
		SS	SPI module input(Slave Select)
		C2OUT	Comparator C2 output
RE0/AN5	8	RE0	General purpose I/O port E
		AN5	A/D channel 5
RE1/AN6	9	RE1	General purpose I/O port E
		AN6	A/D channel 6
RE2/AN7	10	RE2	General purpose I/O port E
		AN7	A/D channel 7
Vdd	11	+	Positive supply
Vss	12	-	Ground(GND)
		RA7	General purpose I/O port A
RA7/OSC1/CLKIN	13	OSC1	Crystal Oscillator Input
		CLKIN	External Clock Input
		OSC2	Crystal Oscillator Output
RA6/OSC2/CLKOUT	14	CLKO	Fosc/4 Output
		RA6	General purpose I/O port A
		RC0	General purpose I/O port C
RC0/T1OSO/T1CKI	15	T1OSO	Timer T1 Oscillator Output
		T1CKI	Timer T1 Clock Input
RC1/T1OSO/T1CKI	16	RC1	General purpose I/O port C
		T1OSO	Timer T1 Oscillator utput

		CCP2	CCP1 and PWM1 module I/O
RC2/P1A/CCP1	17	RC2	General purpose I/O port C
		P1A	PWM module Output
		CCP1	CCP1 and PWM1 module I/O
		RC3	General purpose I/O port C
RC3/SCK/SCL	18	SCK	MSSP module Clock I/O in SPI mode
		SCL	MSSP module Clock I/O in I^2C mode
RD0, RD1, RD2, RD3	19	RD0	General purpose I/O port D
		RC4	General purpose I/O port A
RC4/SDI/SDA	23	SDI	MSSP module Data input in SPI mode
		SDA	MSSP module Data I/O in I^2C mode
RC5/SDO	24	RC5	General purpose I/O port C
		SDO	MSSP module Data output in SPI mode
		RC6	General purpose I/O port C
RC6/TX/CK	25	TX	USART Asynchronous Output
		CK	USART Synchronous Clock
		RC7	General purpose I/O port C
RC7/RX/DT	26	RX	USART Asynchronous Input
		DT	USART Synchronous Data
RD4	27	RD4	General purpose I/O port D
RD5/P1B	28	RD5	General purpose I/O port D
		P1B	PWM output
RD6/P1C	29	RD6	General purpose I/O port D
		P1C	PWM output
RD7/P1D	30	RD7	General purpose I/O port D
		P1D	PWM output
Vss	31	-	Ground(GND)
Vdd	32	+	Positive supply
RB0/AN12/INT	33	RB0	General purpose I/O port B
		AN12	A/D channel 12
		INT	External interrupt
RB1/AN10/C12INT3-	34	RB1	General purpose I/O port B
		AN10	A/D channel 10
		C12INT3-	Comparator C1 or C2 negative input
RB2/AN8	35	RB2	General purpose I/O port B
		AN8	A/D channel 8
RB3/AN9/PGM/C12I N2-	36	RB3	General purpose I/O port B
		AN9	A/D channel 9
		PGM	Programming enable pin
		C12IN2-	Comparator C1 or C2 negative input
RB4/AN11	37	RB4	General purpose I/O port B
		AN11	A/D channel11
RB5/AN13/T1G	38	RB5	General purpose I/O port B
		AN13	A/D channel13
		T1G	TIMER T1EXTENAL INPUT
RB6/ICSPCLK	39	ICSPCLK	General purpose I/O port B
		RB6	SERIAL PROGRAMMIG CLOCK
RB7/ICSPDAT	40	RB7	General purpose I/O port B
		ICSPDAT	Programming enable pin

-La liste ci-dessous inclut seulement certains de ses dispositifs principaux :

- Le Noyau SFR (Special Function Registers)
- Ports d'entrée-sortie
- CPU
- Convertisseur analogique/numérique
- Une RAM donnée de 368 octets,
- Une mémoire EEPROM de 256 octets,
- USART, Port série universel, mode asynchrone (RS232) et mode synchrone
- SSP, Port série synchrone supportant I2C
- Trois TIMERS avec leurs Prescaler, TMR0, TMR1, TMR2
- Deux modules de comparaison et Capture CCP1 et CCP2
- Un chien de garde,
- Générateur d'horloge, à quartz (jusqu' à 40 MHz) ou à Oscillateur RC
- Protection de code,
- Fonctionnement en mode sleep pour réduction de la consommation
- Programmation par mode ICSP (In Circuit Serial Programming) 13V ou 5V,
- Tension de fonctionnement de 2 à 5V,
- Jeux de 35 instructions

La plupart des broches du microcontrôleur PIC16F887 sont multifonctionnelles. Par exemple, la cinquième goupille du microcontrôleur est marquée comme RA3/AN3/Vref+/C1IN+ qui indique qu'il a les fonctions suivantes :

- Entrée-sortie numérique du PORTA RA3
- Entrée analogique AN3
- Référence de tension positive Vref+
- Entrée positif du comparateur C1 C1IN+

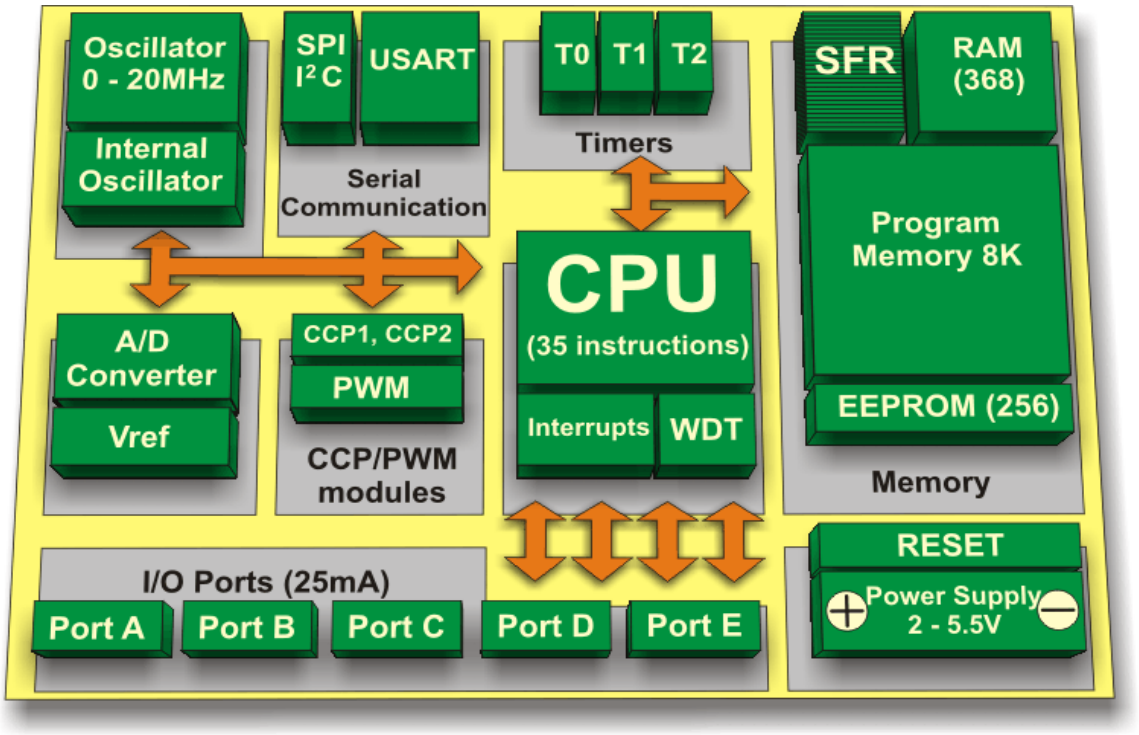


Fig. II- 2 : Les modules du PIC16f887

III-2-Les ports d'E/S :

Le PIC 16F887 dispose de 33 broches d'entrée sortie regroupés dans 5 ports PORTA, PORTB, PORTC, PORTD et PORTE. Chaque broche d'un port peut être configurée soit en entrée soit en sortie à l'aide des registres de direction TRISA, TRISB, TRISC et TRISD et TRISE:

Exemple :

Bit 3 de TRISB = 0 broche 3 de PORTB = SORTIE

Bit 3 de TRISB = 1 broche 3 de PORTB = ENTRÉE

Certains ports ont quelques particularités ,

III-2-1-Le port d'E/S PORTA :

Le port A désigné par PORTA est un port de 6 bits (RA0 à RA5). RA6 et RA7 ne sont pas accessibles.

La configuration de direction se fait à l'aide du registre TRISA, positionner un bit de TRISA à 1 configure la broche correspondante

III-2-2-La broche RA4 :

En entrée, la broche RA4 peut être utilisée soit comme E/S numérique normale, soit comme entrée horloge pour le Timer TMR0

En sortie, RA4 est une E/S à drain ouvert, pour l'utiliser comme sortie logique, il faut ajouter une résistance de pull-up externe.

Le schéma (Fig. IV.1) illustre (pour les non électroniciens) le principe d'une sortie drain ouvert (ou collecteur ouvert) :

-Si RA4 est positionnée à 0, l'interrupteur est fermé , la sortie est reliée à la masse, c'est un niveau bas.

-Si RA4 est placée à 1, l'interrupteur est ouvert, la sortie serait déconnectée s'il n'y avait pas la résistance externe qui place la sortie au niveau haut.

-Si on veut utiliser RA4 pour allumer une LED, on peut utiliser le schéma de Fig. IV.2. Il faut juste remarquer que la logique est inversée, si on envoie 0 sur RA4, l'interrupteur se ferme et la LED s'allume. Si on envoie 1, l'interrupteur s'ouvre et la LED s'éteint.

Le schéma illustré sur Fig. IV.3 peut aussi être utilisé. La logique n'est pas inversée mais il demande une précaution particulière. Il ne faut pas positionner la sortie RA4 à l'aide d'une instruction qui réalise une opération sur l'état actuel du port; genre IORWF, ANDWF, XORWF ou COMF, ADDWF, INCF … Ces instructions réalisent une lecture-écriture en commencent par lire l'état du port pour ensuite faire une opération dessus Or, (sur Fig. IV.3) si la sortie était au niveau haut, l'interrupteur est ouvert, a LED est allumée et elle impose une tension de l'ordre de 1.5V qui sera considérée (à tort) comme un niveau bas lors de la lecture du port par les instructions précitées. La solution est d'utiliser des instructions qui positionnent le PORT sans tenir compte de son état courant comme MOVWF, BSF ou BCF

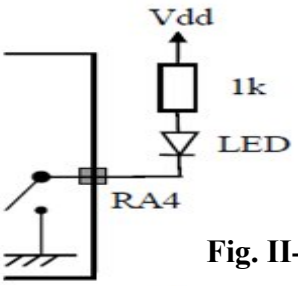


Fig. II-3-1

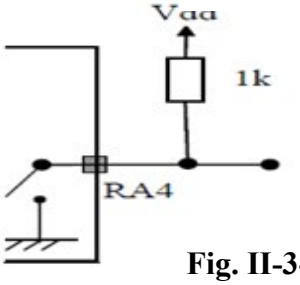


Fig. II-3-2

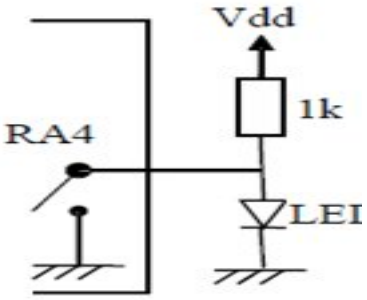


Fig. II-3-3

III-2-3-Le port d'E/S PORTB :

Le port B désigné par PORTB est un port bidirectionnel de 8 bits (RB0 à RB7). Toutes les broches sont compatibles TTL.

La configuration de direction se fait à l'aide du registre TRISB, positionner un bit de TRISB à 1 configure la broche correspondante de PORTB en entré et inversement. Au départ toutes les broches sont configurées en entrée.

En entrée, la ligne RB0 appelée aussi INT peut déclencher l'interruption externe.

En entrée, une quelconque des lignes RB4 à RB7 peut déclencher l'interruption RBI

III-2-4-Le port d'E/S PORTC :

Le port C désigné par PORTC est un port bidirectionnel de 8 bits (RC0 à RC7). Toutes les broches sont compatibles TTL.

La configuration de direction se fait à l'aide du registre TRISC, positionner un bit de TRISC à 1 configure la broche correspondante de PORTC en entré et inversement. Au départ toutes les broches sont configurées en entrée.

Toutes les broches du port C peuvent être utilisées soit comme E/S normales soit comme broches d'accès à différents modules comme le timer 1, les modules de comparaison et de capture CCP1/2, le timer 2, le port I2C ou le port série, ceci sera précisé au moment de l'étude de chacun de ces périphériques.

Pour l'utilisation d'une broche du port C comme E/S normale, il faut s'assurer qu'elle n'a pas été affectée à un de ces modules. Par exemple, si TIMER1 est validé, il peut utiliser les broches RC0 et RC1 selon sa configuration.

III-2-5-Le port d'E/S PORTD :

Le port D désigné par PORTD est un port bidirectionnel de 8 bits (RD0 à RD7). Toutes les broches sont compatibles TTL et ont la fonction trigger de Schmitt en entrée.

Chaque broche et configurable en entrée ou en sortie à l'aide du registre TRISD. Pour configurer une broche en entrée, on positionne le bit correspondant dans TRISD à 1 et inversement.

Le PORTD peut être utilisé dans un mode particulier appelé parallèle slave port, pour cela il faut placer le bit PSPMODE (bit 4) de TRISE à 1. Dans ce cas les 3 bits de PORTE deviennent les entrées de control de ce port (RE, WE et CS)

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

III-2-6-Le port d'E/S PORTE :

Le PORTE contient seulement 3 bits RE0, RE1 et RE2. Les 3 sont configurables en entrée ou en sortie à l'aide des bits 0, 1 ou 2 du registre TRISE.

Les 3 bits de PORTE peuvent être utilisés soit comme E/S numérique soit comme entrées analogiques du CAN. La configuration se fait à l'aide du registre ADCON1.

Si le bit PSPMODE de TRISE est placé à 1, Les trois bits de PORTE deviennent les entrées de control du PORTD qui (dans ce cas) fonctionne en mode parallel Slave mode

A la mise sous tension (RESET), les 3 broches de PORTE sont configurées comme entrées analogiques.

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

- Placer 06h dans ADCON1
- Placer le bit PSPMODE de TRISE à 0

III-3-Le Watch dog Timer WDT (Chien de garde) :

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

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

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

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

- PSA = 1 on utilise le prédiviseur
- PSA = 0 pas de prédiviseur (affecté à TMR0)

PS2	PS1	PS0	WDT
0	0	0	1 :1
0	0	1	1 :2
0	1	0	1 :4
0	1	1	1 :8
1	0	1	1 : 32
1	1	0	1 :64
1	1	1	1 :128

Le rapport du prédiviseur est fixé par les bits PS0, PS1 et PS2 du registre OPTION_REG.L'utilisation du WDT doit se faire avec précaution pour éviter la réinitialisation (inattendue) répétée du programme. Pour éviter un WDT Time-Out lors de l'exécution d'un programme, on a deux possibilités :

- Inhiber le WDT d'une façon permanente en mettant à 0 le bit WDTE dans l'EEPROM de configuration
- Remettre le WDT à 0 périodiquement dans le programme à l'aide de l'instruction CLRWDT pour éviter qu'il ne déborde

III-4-Le convertisseur A/N :

Ce module est de 10 bits dont l'entrée analogique peut être connectée sur l'une des 8 (5 pour 16F887) entrées analogiques externes. On dit qu'on a un CAN à 8 canaux. Les entrées analogiques doivent être configurées en entrée à l'aide des registres TRISA et/ou TRISE. L'échantillonneur bloqueur est intégré, il est constitué d'un interrupteur d'échantillonnage et d'une capacité de blocage de 120 pF.

Les tensions de références permettant de fixer la dynamique du convertisseur. Elles peuvent être choisies parmi Vdd, Vss, Vr+ ou Vr-.

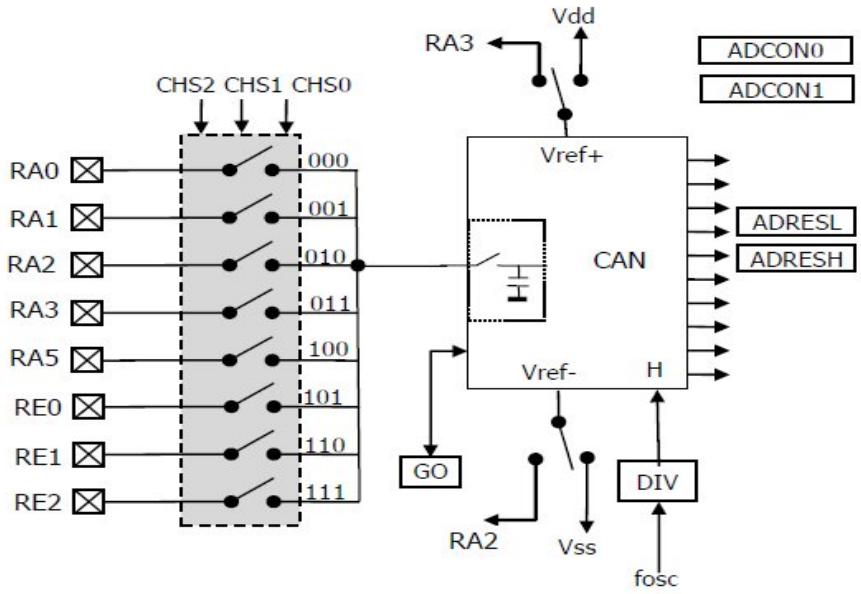


Fig. II-4 : Le CAN

III-4-1-Déroulement d'une Conversion :

Le PIC dispose d'un échantillonneur bloqueur intégré constitué d'un interrupteur S, d'une capacité de maintien C=120 pF et d'un convertisseur analogique numérique 10 bits. Pendant la conversion, la tension

Ve à l'entrée du convertisseur A/N doit être maintenue constante.

Au départ il faut commencer par faire l'acquisition du signal en fermant l'interrupteur S, ceci se fait à l'aide du registre ADCON0, soit au moment de la validation du module par le bit ADON soit après un changement de canal si ADON est déjà positionné.

Après la fin de l'acquisition, on peut démarrer une conversion en positionnant le bit GO_DONE, l'interrupteur S s'ouvre pour assurer le blocage de la tension. La conversion commence, elle est réalisée

en 12 TAD, à la fin, le bit GO_DONE repasse à 0, le drapeau ADIF passe à 1 et le résultat est chargé dans les registres ADRESL et ADRESH.

Le module met 2 TAD supplémentaires pour fermer l'interrupteur S ce qui démarre une nouvelle phase d'acquisition pendant laquelle la tension Ve rejoint la tension analogique d'entrée Va. Le temps d'acquisition dépend de la constante de temps RC, R étant la somme des résistances entre le module de conversion et la source de la tension analogique. Après la fin de l'acquisition, on peut démarrer une nouvelle conversion et ainsi de suite.

III-4-2-Temps de conversion :

Le temps de conversion est égal à 12 TAD
TAD est le temps de conversion d'un bit, il dépend de la fréquence du quartz et du prédiviseur (div) choisi : TAD = div x 1/fosc. Le choix de div doit être ajusté pour que TAD soit ≥ à 1,6 µs

Quartz /Div	20Mhz	5Mhz	4Mhz	2Mhz
2	0 ,1µs	0 ,4µs	0,5µs	1µs
8	0,4µs	1 ,6µs	2µs	4µs
32	1 ,6µs	6,4µs	8µs	16µs

III-5-L'ALU et l'accumulateur W

L'ALU est une Unité Arithmétique et logique qui réalise les opérations arithmétiques et logique de base. L'accumulateur W est un registre de travail, toutes les opérations à deux opérandes passe par lui. On peut avoir :
- Une instruction sur un seul opérande qui est en général un registre situé dans la RAM
- Une instruction sur 2 opérandes. Dans ce cas, l'un des deux opérandes est toujours l'accumulateur W, l'autre peut être soit un registre soit une constante.

Pour les instructions dont un des opérandes est un registre, le résultat peut être récupéré soit dans l'accumulateur, soit dans le registre lui-même.

III-6-L'USART :

L'USART (Universal Synchronous Asynchronous Receiver Transmitter) est l'un des deux modules de communication série dont dispose le PIC 16F887. L'USART peut être configuré comme système de communication asynchrone full duplex ou comme système synchrone half duplex.
La communication se fait sur les deux broches RC6/TX et RC7/RX qui doivent être configurés toutes les deux en ENTREE par TRISC.

III-6-1-Les étapes de transmission (sans interruption, mode 8 bits) :

1) S'assurer que l'interruption TXI n'est pas validée
2) Configurer la broche TX/RC6 en entrée
3) Configurer le registre TXSTA (mode 8 bits, valider transmission, asynchrone, BRGH)
4) Initialiser le registre SPBRG pour définir la vitesse de transmission
5) Valider le port avec le bit RCSTA.SPEN
6) Vérifier que le drapeau PIR1.TXIF=1 c.à.d TXREG est vide
7) Placer la donnée à transmettre dans le registre TXREG
8) Recommencer au point 6) tant qu'on a des données à transmettre

III-6-2-Les étapes de réception (sans interruption, mode 8 bits) :

1) S'assurer que l'interruption RCI n'est pas validée
2) Configurer la broche RX/RC7 en entrée
3) Initialiser le registre SPBRG pour définir la vitesse de communication
4) Configurer le registre TXSTA (asynchrone, BRGH)
5) Configurer le registre RCSTA (validation port, mode 8 bits, valider réception continue)
6) Attendre que drapeau RCIF passe à 1 ce qui annonce la fin de réception d'un octet

7) Lire l'octet reçu dans le registre RCREG
8) Recommencer au point 6) tant qu'on a des données à recevoir

III-7-Le module MSSP (Master Synchronous Serial Port) :

Le MSSP est un des deux modules de communication série du PIC 16F887. Il permet d'échanger des données en mode synchrone avec d'autres circuits qui peuvent être des microcontrôleurs, des mémoires EEPROM série, des convertisseurs A/N, des modules d'affichage . . Il peut fonctionner selon deux modes ; le mode SPI (Serial Peripheral Interface) et le mode I2C (Inter-Integrated Circuit)

III-8-Introduction au bus I2C :

Avant de parler du module MSSP en mode I2C du PIC, introduisons très brièvement Le standard I2C
Le bus I²C permet de faire communiquer entre eux des composants électroniques très divers grâce à seulement 3 fils : Un signal de donnée (SDA), un signal d'horloge (SCL), et un signal de référence électrique (Masse).
Comme les lignes SDA et SCK sont utilisées dans les deux sens par les deux circuits qui communiquent, on peut avoir un circuit qui place la ligne à 1 (Vcc) et l'autre qui la place à 0 (masse) ce qui correspond à un court circuit qui peut détruire les deux composants. Pour éviter ce problème, les E/S SDA et SCK fonctionnent en mode collecteur ouvert (ou drain ouvert) de sorte qu'un circuit ne peut imposer que le niveau bas ou ouvrir la ligne, le niveau haut est obtenu par une résistance de tirage externe. Ainsi une ligne est à 0 quand un des deux circuits impose le 0. Elle passe à 1 quand les deux circuits imposent le 1 (circuit ouvert). Le protocole I2C jongle avec cette situation pour organiser l'échange des données entre les deux composants.

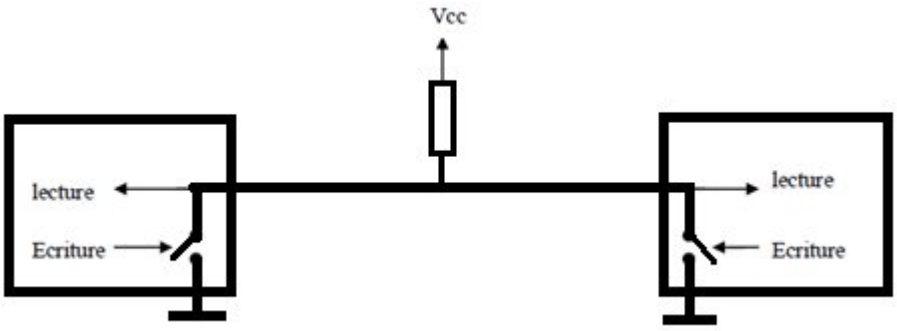


Fig. II- 5 : principe du bus I2C

Un bus I2C peut être relié à plusieurs circuits, mais pendant une communication, un des circuits est le maître, c'est lui génère l'horloge et initie les séquences de transmission, l'autre est l'esclave, il subit l'horloge du maître sur la ligne SCK mais il peut tout de même recevoir et émettre des données sur la ligne SDA. Chaque esclave a une adresse, au début d'une séquence de communication, le maître qui initie la séquence envoie l'adresse d'esclave avec lequel il désire communiquer, celui-ci reconnaît son adresse et répond, les autres esclaves (s'il y en a) restent désactivés.

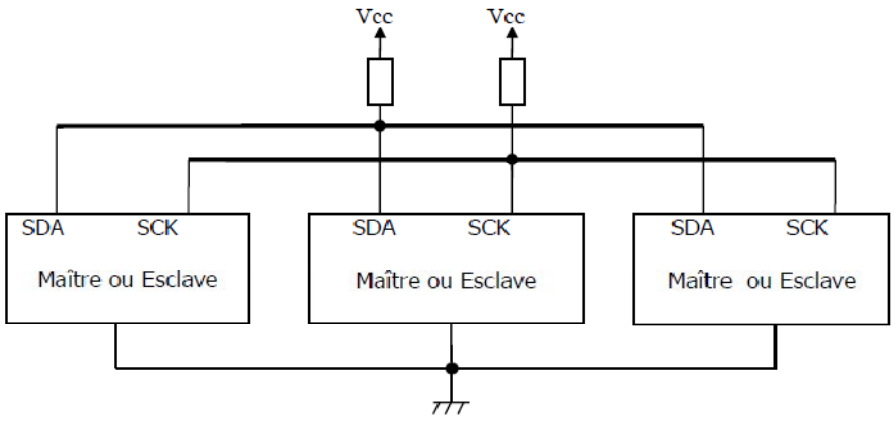


Fig. II- 6 : Communication en I2C

Certains circuits sont fabriqués pour être des masters, d'autres des slaves et d'autres peuvent être soit l'un soit l'autre. Pour prendre le contrôle du bus, il faut que celui-ci soit au repos (SDA et SCL à '1'). Lorsqu'un circuit prend le contrôle du bus, il en devient le maître. C'est lui qui génère le signal d'horloge et c'est lui qui initie les séquences d'échange.

III-8-1-START condition :

Au début d'une séquence d'échange, le master génère un START condition (S) avant de commencer l'échange de données. Au repos, les lignes SCL et SDA sont à l'état haut (relâchées). Pour générer un start, le master place d'abord la ligne SDA à 0, ensuite il place SCK à 0.

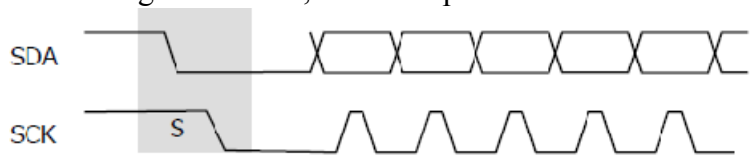


Fig. II- 7 : START condition

III-8-2-Transmission d'un bit :

On place le bit à transmettre sur la ligne SDA ensuite on envoie une impulsion d'horloge sur la ligne SCK. C'est cette impulsion qui informe le slave qu'il doit lire la donnée sur SDA

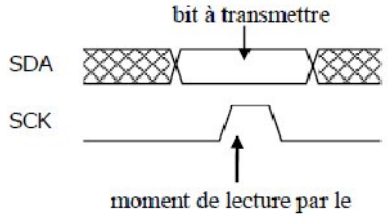


Fig. II- 8 : Transmission d'un bit

III-8-3-Stop condition :

A la fin d'une séquence d'échange, le master génère un stop condition (P) après lequel le bus est de nouveau libre. Pour cela, à partir de la situation SDA=0, SCK=0, le master commence par placer SCK à 1 et place ensuite SDA à 1.

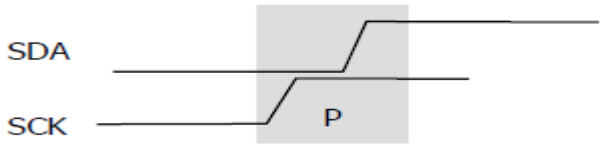


Fig. II-9 : STOP condition

III-8-4-Remarque sur le Start et le Stop condition :

Une séquence de transmission peut contenir plusieurs Starts conditions avant de rencontrer un Stop Condition. On parle de repeated Start condition. Un Stop condition est toujours synonyme de FIN de transmission.

III-8-5-L'acknowledge :

L'acknowledge est l'accusé de réception. Il est placé par le circuit qui reçoit sur la ligne SDA juste après la réception du 8ème bit. C'est l'émetteur qui le lit de la même façon qu'on lit un bit ordinaire. SDA=0 □ acknowledge positif (ACK), SDA=1, acknowledge négatif (NOACK).

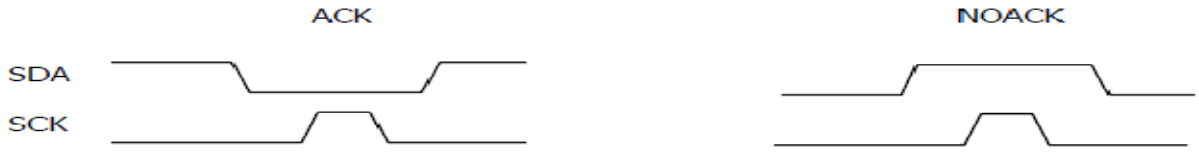


Fig. II-10 : L'acknowledge

En conclusion, l'échange d'un octet nécessite 8 + 1 = 9 impulsions d'horloge sur la broche SCL.

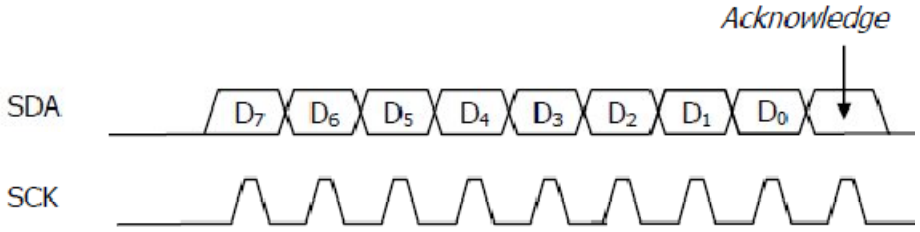


Fig. II- 11 : l'échange d'un octet

III-8-6-L'adresse du bit R/W :

Comme on peut brancher plusieurs composants sur un bus I2C, il est nécessaire de définir une adresse unique pour chacun. Elle est codée sur 7 bits A6 A5 A4 A3 A2 A1 A0. Le master qui démarre une séquence d'échange envoie l'adresse du slave juste après le START condition. Comme il y a seulement 7 bits, le master envoie à la 8ème position le bit R/W pour indiquer au slave s'il désire une émission (R/W=0) ou une réception (R/W=1).

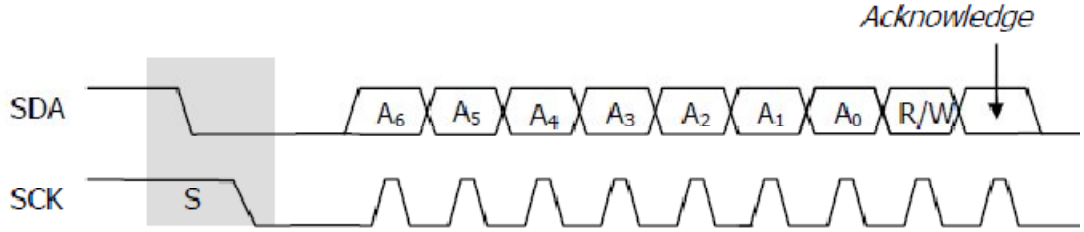


Fig. II- 12 : Adresse du bit R/W

Le standard I2C supporte maintenant l'adressage sur 10 bits. Comme avec 7 bits on peut adresser jusqu'à 128 composants, cela nous suffira largement.

III-8-7-Le module MSSP en mode I2C :

Le module MSSP du PIC peut être configuré en master ou en slave. Il utilise les broches RC3/SCL (Horloge) et RC4/SDA (données). Ces broches doivent être configurées en ENTREE à l'aide du registre TRISC et doivent être munie de résistances de pull-up externes nécessaire au fonctionnement I2C.
Les fréquences d'horloges supportées sont 100 kHz, 400 kHz et 1 MHz
L'accès au module en lecture et écriture se fait à l'aide du registre tampon (buffer) SSPBUFF. La transmission et la réception se fait à l'aide du registre à décalage SSPSR auquel nous n'avons pas directement accès

III-8-8-Transmission d'un octet :

Pour transmettre un octet, il suffit de le copier dans le registre SSPBUF, et le module MSSP s'occupe du reste. Au moment de l'écriture dans SSPBUF, le bit BF passe à 1 et la transmission commence. A la fin de la transmission, le bit SSPSTAT.BF repasse à 0 et le drapeau d'interruption PIR1.SSPIF passe à 1.
Le bit BF apparaît donc comme un bit très important, c'est lui qui nous permet de savoir si le registre SSPBUF est libre ou non
Si on tante d'écrire dans SSPBUF alors que BF=1, le bit SSPCON.WCOL passe à 1 pour indiquer une collision et l'écriture n'a pas lieu.

III-8-9- Réception d'un octet :

A la fin de la réception d'un octet, celui-ci est transféré dans SSPBUF, l'indicateur SSPSTAT.BF et le drapeau d'interruption PIR1.SSPIF passent à 1. BF repasse automatiquement à 0 au moment de la lecture de SSPBUF alors que SSPIF soit être remis à 0 par soft.

Si le PIC termine la réception d'un octet avant que l'octet précédent qui se trouve dans SSPBUF n'ait été lu, on a un Overflow qui sera signalé par le drapeau SSPOV. Le transfert n'a pas lieu, l'octet arrivé est perdu.

III-9-Unité centrale (CPU) :

Il est important de préciser que l'unité centrale est fabriquée en technologie de RISC.
Le RISC représente le jeu d'instructions réduit, qui donne les deux grands avantages du PIC16F887 :

- L'unité centrale identifie et d'exécute seulement 35 instructions simples.
- Le temps d'exécution est le même pour presque toutes les instructions, et dure 4 cycle d'horloge sauf pour le saut et le branchement qui dure 2 cycles. La fréquence du signal d'horloge est stabilisée par un quartz. Si la fréquence de l'opération du microcontrôleur est 20MHz, la période d'exécution de chaque instruction sera 200nS, c.-à-d. le programme exécutera 5 millions d'instructions par seconde !

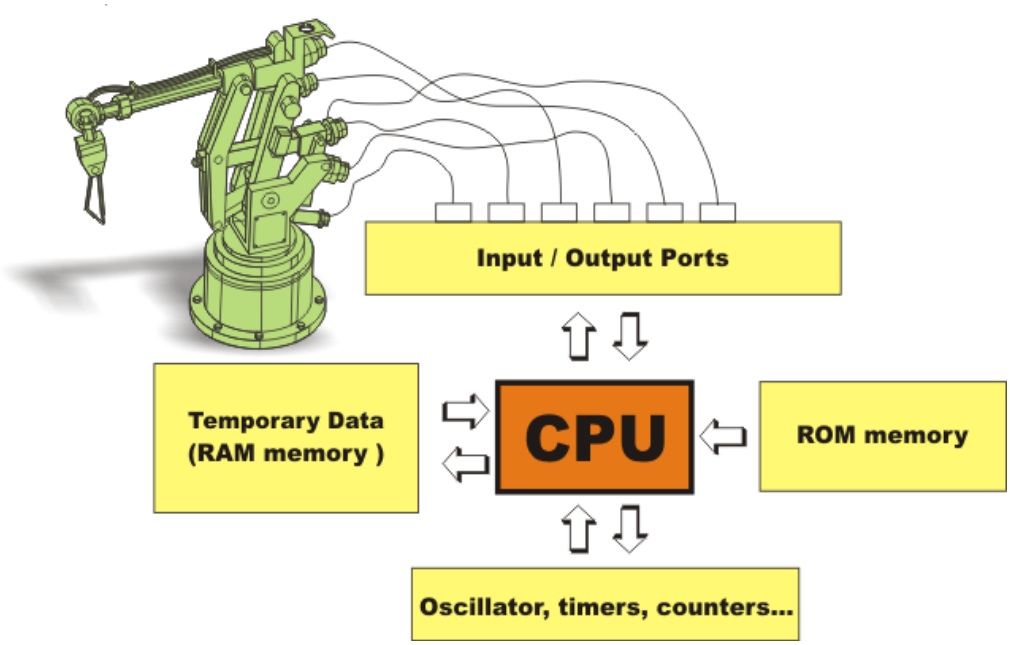


Fig. II- 12 : Rôle du CPU

III-10-Les mémoire :

Le PIC16F887 comporte trois types de mémoire : ROM, RAM et EEPROM..

III-10-1-La Rom :

La ROM est utilisée pour sauvegarder de manière permanente le programme étant actuellement exécuté. Appellé souvent la mémoire de programme . Le PIC16F887 a 8Kb de ROM, son contenu peut être facilement changé en fournissant une tension de programmation de (13V). le processus est effectué automatiquement au moyen d'un programme spécial installé sur un PC et un appareil électronique simple appelés le programmateur,comme illustré dans le schema suivant :

III-10-2-L'EEPROM :

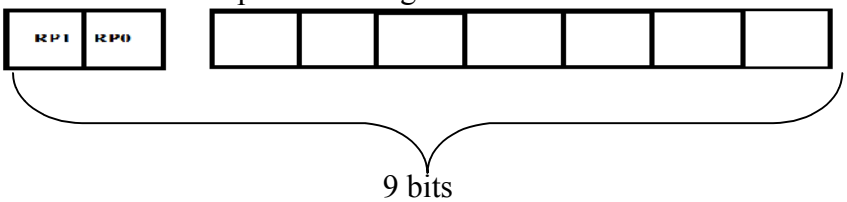
Semblable à la mémoire de programme, le contenu d'EEPROM est sauvé de manière permanente , à la différence de la ROM, le contenu d'EEPROM peut être changé meme pendant l'execution d'un programme par le microcontrôleur. C'est pourquoi cette mémoire est parfaite comme une mémoire de sauvegarde pour certains résultats intermediaire employé pendant l'opération.

III-10-3-La RAM :

C'est le type le plus complexe de toutes les mémoire du microcontrôleur. Il se compose de deux parties : registres d'usage universel (GPR) et registres de fonction spéciale (SFR). Tous les registres sont divisés à quatre banques de mémoire.
Bien que Les deux groupes de registres s'éffacent quand on met le µc hors tension e, tous les deux sont fabriqués de la même maniere et agissent d'une façon semblable, ils n'ont pas beaucoup de fonctions en commun.

III-10-4-L'accès à la RAM par adressage DIRECT :

Avec ce mode d'adressage, on précise dans l'instruction la valeur de l'adresse à laquelle on veut accéder. Par exemple, pour copier le contenu de l'accumulateur W dans la case mémoire d'adresse 50, on utilise l'instruction **MOVWF 50**. Cette instruction sera codée sur 14 bits, la partie adresse est codée sur **7 bits** ce qui va poser quelques petits problèmes. En effet, 7 bits permettent d'adresser seulement 128 positions. Pour pouvoir adresser les 512 positions accessibles, il faut **9 bits** d'adresse. Pour avoir ces 9 bits, le PIC complète les 7 bits venant de l'instruction par deux bits situés dans le registre de configuration STATUS. Ces bits sont appelés RP0 et RP1 et doivent être positionnés correctement avant toute instruction qui accède à la RAM par l'adressage direct.



La RAM apparaît alors organisée en 4 Banks de 128 octets chacune. L'adresse instruction permet d'adresser à l'intérieur d'une Bank alors que les bits RP0 et RP1 du registre STATUS permettent de choisir une Bank. La Figure II-1 montre l'organisation de la RAM avec les zones allouée au SFR et aux GPR. Les zones hachurées ne sont pas implantées physiquement. Si on essaye d'y accéder, on est aiguillé automatiquement vers la zone [70h, 7Fh] appelée zone commune.
Même si on précise une adresse supérieure à 127 (+ de 7 bits) dans une instruction, elle est tronquée à 7 bits puis complétée par les bits RP0 et RP1 pour former une adresse 9 bis. Par exemple, pour copier l'accumulateur W dans la case mémoire d'adresse 1EFh, il faut d'abord placer les bits RP0 et RP1 à 1 (Bank 3), ensuite on utilise soit l'instruction **MOVWF 6Fh** soit l'instruction
MOVWF 1EFh, qui donne le même résultat. En effet, que l'on écrive 6Fh = 0110 1111 ou
1EFh = 0001 1110 1111, le PIC ne prend que 7 bits soit : 1101111 = 6Fh et complète avec les bits RP1, RP0 pour obtenir 11 1101111 = 1EFh. Les instructions bcf et bsf et qui permettent de positionner un bit à 0 ou à 1, Bcf STATUS, RP0 ; place le bit RP0 à 0 ,Bsf STATUS, RP1 ; place le bit RP1 à 1

III-10-5-L'accès à la RAM par l'adressage INDIRECT :

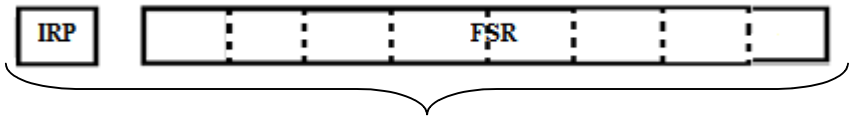
Pour accéder à une position de la RAM en utilisant l'adressage indirect, on passe toujours par une position fictive appelée INDF (Indirect File).
Exemple : l'instruction CLRFINDF signifie : mettre à zéro la case mémoire d'adresse INDF.
INDF est la case mémoire pointée par le pointeur IRP/FSR :
IRP est un bit qui se trouve dans STATUS et SFR est un registre accessible dans tous les Banks, le registre de pointage SFR est un registre 8 bits, il peut donc adresser au maximum 256 positions mémoire (de 00h à FFh), c'est seulement la moitié de la RAM dont on dispose. Il nous manque un bit pour avoir les 9 bits nécessaires. On utilise le bit IRP qui se trouve dans le registre STATUS.

Exemple :

Si on place 74h dans le registre SFR et on positionne le bit IRP à 1, alors, l'instruction CLRF INDF signifie: remettre à zéro la case mémoire d'adresse **174h**.
Donc en résumé, chaque fois que le PIC rencontre le mot INDF dans un programme, il sait qu'il s'agit de la case mémoire dont l'adresse (9 bits) se trouve dans le registre SFR complété par le bit IRP du registre STATUS.

IRP=0→000h a 0FFh → page 0
IRP=1→100h a 1FFh → page 1

INDF est la case memoire pointée par le pointeur



Pointeur 9 bit

III-11-Les instructions du 16F887 :

Tous les PICs Mid-Range ont un jeu de 35 instructions,
Chaque instruction est codée sur un mot de 14 bits qui contient le code opération (OC) ainsi que l'opérande,
Toutes les instructions sont exécutées en un cycle d'horloge, à part les instructions de saut qui sont exécutées en 2 cycles d'horloge. Sachant que l'horloge système est égale à fosc/4, si on utilise un quartz de 4MHz, on obtient une horloge fosc/4 = 1000000 cycles/seconde, cela nous donne une puissance de l'ordre de 1MIPS (1 Million d' Instructions Par Seconde). Avec un quartz de 20MHz, on obtient une vitesse de traitement de 5 MIPS.

III-11-1-Les instructions « orientées Registre» :

Ce sont des instructions qui manipulent un octet se trouvant dans la RAM. Ça peut être un registre de configuration SFR ou une case mémoire quelconque (Registre GPR)

III-11-2-Les instructions « orientées bits » :

Ce sont des instructions destinées à manipuler directement un bit d'un registre que se soit un registre de configuration SFR ou une case mémoire quelconque (registre GPR). Tous les bits de la RAM peuvent être manipulés individuellement.

III-11-3-Les instructions opérant sur une constante :

Ce sont les instructions entre l'accumulateur W est une constante K

III-11-4-Les instructions de saut et appel de procédures :

Ce sont les instructions qui permettent de sauter à une autre position dans le programme et de continuer l'exécution du programme à partir de cette position.

III-12-Les registres d'usage universel(GENERAL-PURPOSE REGISTER) :

les registres d'usage universel sont utilisés pour stocker des données provisoires et des résultats intermediaires. Par exemple, si le but d'un programme est de compter (des produits sur une chaîne de montage, par exemple), il est nécessaire d'avoir un registre qui représente (la somme), il est nécessaire de spécifier l'adresse exacte du registre d'usage universel et de lui assigner cette fonction. assurer que la valeur de ce registre s'incrémente a chaque fois un produit traverse un capteur par exemple.

III-13-Registres de fonction spéciale (SFR) SPECIAL FUNCTION REGISTERS :

ces registres occupent également des endroits dans la RAM, mais à la différence des registres d'usage universel, leur fonction est prédéterminée pendant la fabrication et ne peut pas être modifié plus tard, n'importe quel changement dans leur contenu affectera directement l'opération du microcontrôleur ou au moins certains de ses modules.En outre, chaque registre SFR a son propre nom , qui simplifie considérablement le processus de l'écriture du programme, il est suffisant de spécifier le nom d'un registre afin de lire ou changer son contenu.

```
if PORTB.0 = 1          'Check whether the RB0 pin is driven high
sum = sum + 1           'If true, the variable value is incremented by 1
end if                  'If false, the program exits the if statement
```

III-14-Les banques de la RAM:

La RAM est divisée en quatre banques (registres). Avant d'accéder à n'importe quel registre pendant l'écriture du programme (pour lire ou changer son contenu), il est nécessaire de choisir la banque contenant le registre. Deux bits du registre STATUT sont employés pour le choix de la banque.
Pour rendre cette tache aussi simple, les registres les plus utilisés généralement ont la même adresse dans toutes les banques et peuvent donc être facilement accédés.
A l'ecriture d'un programme en langages de programmation évolué , et a l'utilisation des compilateurs tels que mikroBasic PRO pour les PIC, il se fait juste de spécifier le nom du register a adresser. Ayant cette nformation, le compilateur est capable de choisir la banque compétente aussi bien qu'inclure des instructions appropriées dans le code pendant le processus de la compilation.

III-15-La pile :

Une partie de la RAM est utilisée comme pile qui se compose de huit registres de 13bit. Avant que le microcontrôleur démarre pour exécuter une sous-routine (instruction CALL) ou quand une interruption se produit, l'adresse de l'instruction exécuté après est mise dans la pile, c.-à-d. une de ses registres. En conséquence, le microcontrôleur sait d'où continuer l'exécution du programme principal quand une sous-routine ou une exécution d'interruption est complète. L'adresse est effacée juste après le retour au programme principal et un endroit de pile soit ainsi automatiquement disponible pour davantage usage.
Les données sont toujours automatiquement misent a la pile. Se qui signifie qu'après la huitième adresse, la neuvième recouvre la première valeur qui a été stockée. La dixième recouvre la deuxième et ainsi de suite. Les données recouvertes de cette façon ne sont pas récupérables. Le programmeur ne peut pas accéder a ces inscriptions ni pour lecture ni pour la l'écriture et il n'y a aucun bit pour indiquer le débordement ou l'état de la pile.

III-16-Système d'interruption :

La première chose que fait le microcontrôleur,a l'arrivé d'une demande d'interruption c'est d'exécuter l'instruction courante, puis d'arrêter l'exécution du programme principale. L'adresse de mémoire de programme en cours est automatiquement mis a la pile,l'adresse par défaut (prédéfinie par le fabricant) est écrite au compteur de programme(program counter), Ladresse d'où le programme se poursuit s'appelle un vecteur d'interruption. Pour le microcontrôleur PIC16F887, l'adresse est 0004h.
Comme illustré dans la figure ci-dessous, le vecteur d'interruption devrait être sauté pendant l'exécution du programme principale.

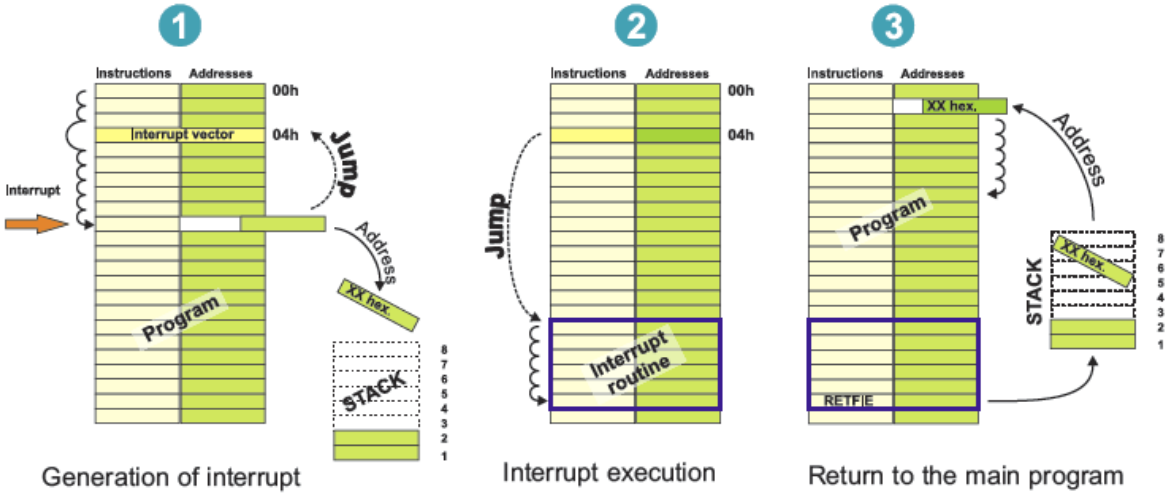


Fig. II-13 :deroulement d'une interruption

Une partie du programme à exécuter quand une demande d'interruption arrive s'appelle une routine d'interruption (c'est une sous-routine ISR). La première instruction de la routine d'interruption est située au vecteur d'interruption.Quelques microcontrôleurs ont deux ou trois vecteurs d'interruption (chaque demande d'interruption a son vecteur), tandis que ce microcontrôleur a seulement un. C'est pour cela la première partie de chaque routine d'interruption devrait être pour la détection de la source d'interruption.

Quand la source d'interruption est connue et exécutée, le microcontrôleur atteint l'instruction RETFIE, saute l'adresse de la pile et poursuit l'exécution du programme principale . MikroBasic identifie une routine d'interruption à exécuter au moyen du mot-clé d'interruption(**void interrupt)**. La routine d'interruption devrait être écrite par l'utilisateur.

Exemple :

```
Void interrupt () {                    // interrupt routine
    cnt++ ;                            // interrupt causes variable cnt to be incremented by 1
}
```

III-17-Utilisation des SFR :

Il y a plusieurs SFR. Chacun commande un certain nombre de processus, il ressemble à une grande table de commande avec beaucoup d'instruments et commutateurs.

a-Noyau SFR :

Le texte suivant décrit le noyau SFR du microcontrôleur PIC16F887.

b-Registre STATUS :

Le registre de STATUT contient :

- Le statut arithmétique de données dans le registre de W,
- Le RESET
- Le bit du choix de la banque pour la mémoire de données.

c-Registre d'OPTION_REG :

Fig. II-14 : Registre d'OPTION_REG

❖ IRP – Un bit est employé pour choisir la banque de registre à l'adressage indirect.

- 1#=L'activation des banques 0 et 1 (00h-FFh).
- #0=L'activation des banques 2 et 3 (endroits de mémoire 100h-1FFh).

❖ RP1, RP0 - le bit est utilisé pour choix des registres (l'adressage direct).

RP1	RP0	BANK ACTIVE
0	0	BANK0
0	1	BANK1
1	0	BANK2
1	1	BANK3

❖ TO : bit de fin de temps (time out).

- #1=a l'exécution de l'instruction CLRWDT ou SLEEP, le microcontrôleur est au mode de basse consommation.
- 0#- Après la fin du temps WDT.

❖ PD=bit de puissance faible.

- 1#=après l'exécution de l'instruction CLRWDT qui remet à zéro l'horloge de surveillance.
- 0#=Après l'exécution de l'instruction SLEEP qui place le microcontrôleur au mode de basse consommation.

❖ Z-bit zéro

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

- 0#=Le résultat d'une opération arithmétique ou logique est différent que zéro.

❖ DC - Bit d'emprunt, change pendant l'addition ou de la soustraction si un débordement de du résultat se produit.

- 1#=Un transfert du 4ème bit du poids faible du résultat s'est produit.
- 0#=Aucun transfert du 4ème bit du poids faible du résultat s'est produit.

❖ C – Bit d'emprunt, change pendant l'addition ou de la soustraction si un débordement du résultat se produit, c.-à-d. si le résultat est plus grand que 255 ou plus moins de 0.

- 1#=Un transfert de bit du poids le plus fort (MSB) du résultat s'est produit.0#= Aucun transfert de bit du poids le plus fort (MSB) du résultat s'est produit.

d-Registre OPTION :

Fig. II-15 : Registre OPTION

R/W – bit de lecture/ecriture.
Le registre d'OPTION_REG contient des divers bits de commande pour la configuration :

- le prescaler de Timer0/WDT
- le Timer0
- l'interruption externe
- les pull-up sur PORTB

RBPU – permet de mettre le port B en pull-up (**bit7 = RB7)**
1#=le PORTB est en pull-up.
#0=le PORTB n'est pas en pull-up.

INTEDG – bit de choix du front d'interruption (**bit6 = RB6)**

- #1= Interruption sur le front montant de RB0/INT.
- #0 =Interruption sur le front descendant de RB0/INT.

T0CS – bit de choix de la source d'horloge TMR0. **bit5 = RB5**

- #1=horloge externe T0CKI.
- 0#=Horloge interne di µc (Fosc/4).

❖ TOSE **bit 3 = RB3 :** Le bit Du choix du facteur de division du Prescaler (TIMER OU WDT).

- #1# - Prescaler est assigné au WDT.
- 0#- Prescaler est assigné au Timer0.

PS0, PS2, PS1, bit de choix de facteur de Prescaler Rate
Suivant les indications de la table ci-dessous, le facteur de Prescaler dépend de Timer0 ou de WDT.

PS2	PS1	PS0	TMR0	WDT
0	0	0	1 :2	1 :1
0	0	1	1 :4	1 :2
0	1	0	1 :8	1 :4
0	1	1	1 :16	1 :8
1	0	1	1 :64	1 : 32
1	1	0	1 :128	1 :64
1	1	1	1 :256	1 :128

e-Registres d'interruption :

A la réception d'une demande d'interruption, ca ne signifie pas qu'il aura lieu automatiquement, parce qu'il doit être activé par l'usager (dans le programme). Chaque interruption est associée avec un drapeau (flag) qui indique une demande d'interruption, on peut la repérer par son nom IE.

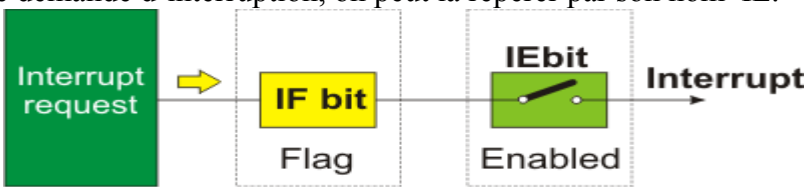


Fig. II-16 : Registres d'interruption

III-18-Le principe de fonctionnement d'une interruption :

A la demande d'interruption, le bit du drapeau est a (1) SET si le IE approprié est a (0), la condition d'interruption sera complètement ignorer. Autrement (s'il est activé) et a la demande de plusieurs interruption, il est nécessaire de découvrir leurs sources avant qu'un programme d'interruption commencent l'exécution. La découverte de la indiquer par le bit du drapeau. Ce bit s'efface pas automatiquement, mais par le programme pendant l'exécution de la routine. Si on néglige cette étape une autre interruption s'exécute systématiquement des le retour au programme principal.
Toutes les sources d'interruption du microcontrôleur de PIC16F887 sont sur le page suivante.
Le **GIE** permet d'active toutes les interruptions démasqué
Le **PEIE** permet d'activé toutes les interruptions de périphérique démasqué.
Pour activer une interruption (qui a comme source le changement de l'état de logique de PORTB), il est nécessaire d'activer chaque bit séparément. Dans ce cas, les bits du registre IOCB sont des bits d'activation d'interruption.

III-18-1-LE registre INTCON :

Le registre INTCON comprend des bits d'activation et indicateurs (drapeaux) pour :
- débordement OVERFLOW du registre TMR0.
-changement de l'état logique du PORTB.
-Interruption externe sur INT.

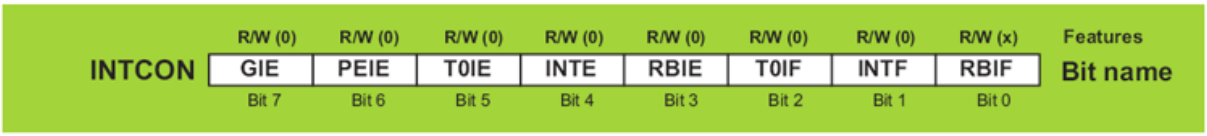


Fig. II- 17 : Registre INTCON

R / W (lecture /écriture), (0) état par défaut après RESET, le morceau est espace libre, (X) état inconnu après RESET.

GIE :bit d'activation de toutes les interruptions simultanément.
1# = interruption démasqué activé.
0# = interruption démasqué désactivé.

PEIE- activation des interruptions du périphérique : pas d'effet sur les interruptions déclenchées par TIMER0 ou par le changement d'état du port PORTB (RB0/INT).
1# = interruption de périphérique démasquée activé.
0#= interruption de périphérique démasquée désactive.

T0IE : bit d'activation d'interruption lors de débordement du contenu du registreTMR0 (Overflow).
1#= interruption activé lors de débordement de TMR0.
#0 = interruption désactivé lors de débordement de TMR0.

INTE :bits d'activation d'interruption externe lors du changement de l'état de RB0/INT
#1 -- interruption externe d'INT activé.

#0 -- l'interruption externe d'INT désactivé.

RBIE : bit d'activation de l interruption lors de changement d'état du
#1=interruption activé lors du changement de l'état du PORTB.
0# =interruption désactivé lors du changement de l'état du PORTB.

T0IF : bit de débordement du registre TMR0 (après le RESET)
#1= débordement du registre TMR0 (OVERFLOWED).
0#=pas de débordement du registre TMR0 (NOT OVERFLOWED).

INTF : bit d'interruption externe lors de changement d'état de RB0/INT :
1#= existence d'interruption externe sur INT (occurred).
0# pas d'interruption externe sur INT (no occurred).

RBIF :bit d'interruption lors du changement d'état de quelques broches PORTB.
#1=au moins un changement d'état au PORTB.
#0=aucun changement d'état au PORTB.

III-19-Oscillateur :

Pour synchroniser tous les processus, il est nécessaire de fournir un signal d'horloge.
L'horloge peut être soit interne soit externe. L'horloge interne est constituée d'un oscillateur à quartz ou d'un oscillateur RC.

Avec l'oscillateur à Quartz, on peut avoir des fréquences allant jusqu'à 40 MHz selon le type de µC. Le filtre passe bas (Rs, C1, C2) limite les harmoniques dus à l'écrêtage et Réduit l'amplitude de l'oscillation, il n'est pas obligatoire. Avec un oscillateur RC, la fréquence de l'oscillation est fixée par Vdd, elle peut varier légèrement d'un circuit à l'autre. Dans certains cas, une horloge externe au microcontrôleur peut être utilisée pour synchroniser le PIC sur un processus particulier.
Quelque soit l'oscillateur utilisé, l'horloge système dite aussi horloge instruction est obtenue en divisant la fréquence par 4. On utilisera le terme Fosc/4 pour désigner l'horloge système..

III-19-1-L'oscillateur externe :

L'oscillateur externe (quartz, circuit RC) est relie a OSC1 et OSC2. Le mode d'oscillateur est choisi par le bit du mot de Config, chargé au microcontrôleur pendant la programmation.

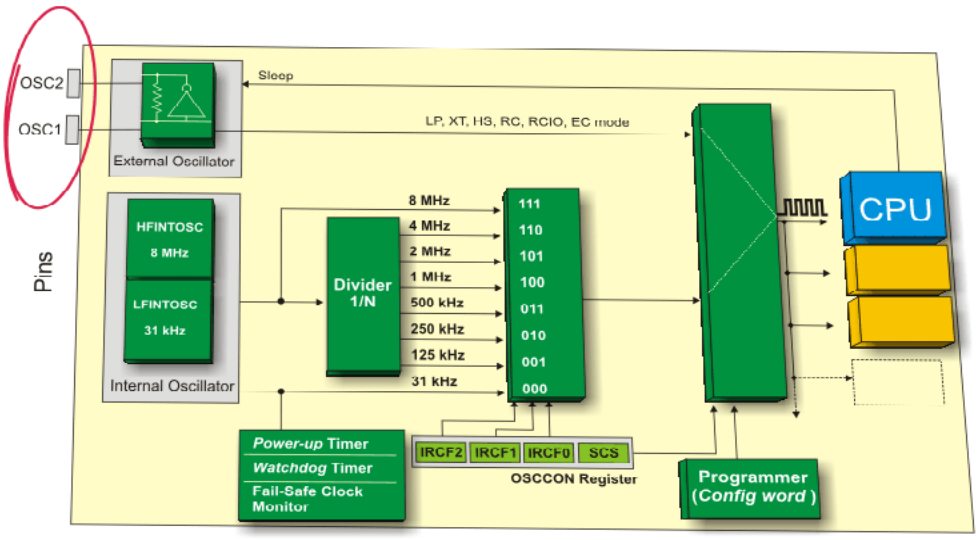


Fig. II-18 : L'oscillateur externe

III-19-2-L'oscillateur interne se compose de :

L'oscillateur HFINTOSC : oscillateur de haute fréquence de 8 MHz. Le µc peut fonctionner a cette fréquence ou après une division sur le facteur PRESCALER
L'oscillateur LFINTOSC : oscillateur de basse fréquence de 31kHz, generalement utilisé pour le WATCH-DOG. On peut choisir l'oscillateur externe ou interne par le (SCS).

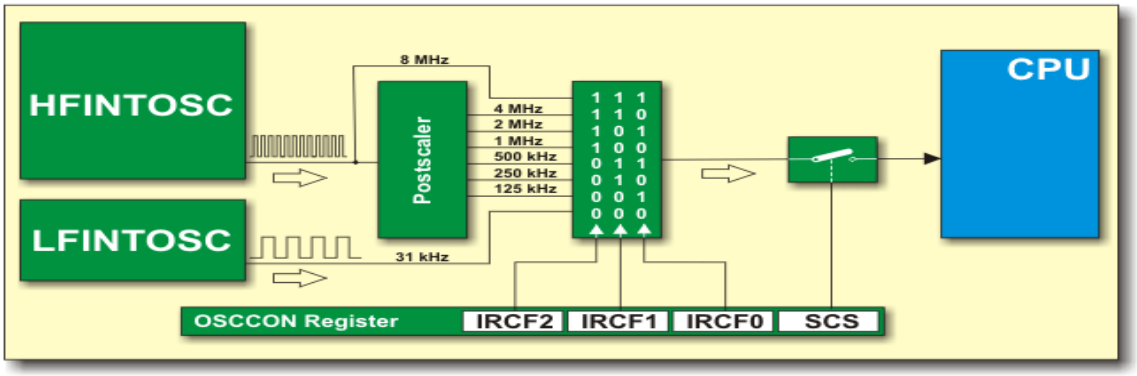


Fig. II- 19 : L'oscillateur interne

III-19-3-Registre OSCCON :

Le registre d'OSCCON permet de choisir la fréquence, il est composé des bits de choix de la fréquence (IRCF2, IRCF1, IRCF0), l'état de la fréquence (HTS, LTS) et le système de contrôle d'horloge (OSTA, SCS).

Fig. II-20 : Registre OSCCON

IRCF2-0 –pour choisir la fréquence de l'oscillateur interne

IRCF2	IRCF1	IRCF0	FREQUENCE	OSC
1	1	1	8Mhz	HFINTOSC
1	1	0	4 Mhz	HFINTOSC
1	0	1	2 Mhz	HFINTOSC
1	1	0	1 MHz	HFINTOSC
0	1	1	500 KHz	HFINTOSC
0	1	0	250 KHz	HFINTOSC
0	0	1	125 kHz	HFINTOSC
0	0	0	31 kHz	LFINTOSC

OSTS : case d'état de fin de début du temps d'oscillateur pour indiquer la source de l'oscillateur utilisé, elle en lecture seul.
1# =l'oscillateur utilisé est externe.
0# = l'oscillateur utilisé est interne (HFINTOSC ou LFINTOSC).

HTS : indication de la stabilité de l'oscillateur HFINTOSC (8 MHz - 125 KHz) de l'oscillateur.
1#= l'oscillateur HFINTOSC est stable.
#0= l'oscillateur HFINTOSC n'est pas stable.

LTS-: indication de la stabilité de l'oscillateur LFINTOSC (31 KHz) l'oscillateur intérieur.
1#= l'oscillateur HFINTOSC est stable.
#0= l'oscillateur HFINTOSC n'est pas stable

SCS : choix de l'oscillateur.
1#=l'oscillateur interne est utilisé comme une horloge.
0#=l'oscillateur externe est utilisé comme une horloge.

Chapitre III : travaux pratiques

TP1:

Clignotement de led

But du TP :

Ce montage permet de clignoter une led, la configuration des Ports en entrée ou sortie, L'interrupteur reset permet de lancer le clignotement.

Schéma électrique :

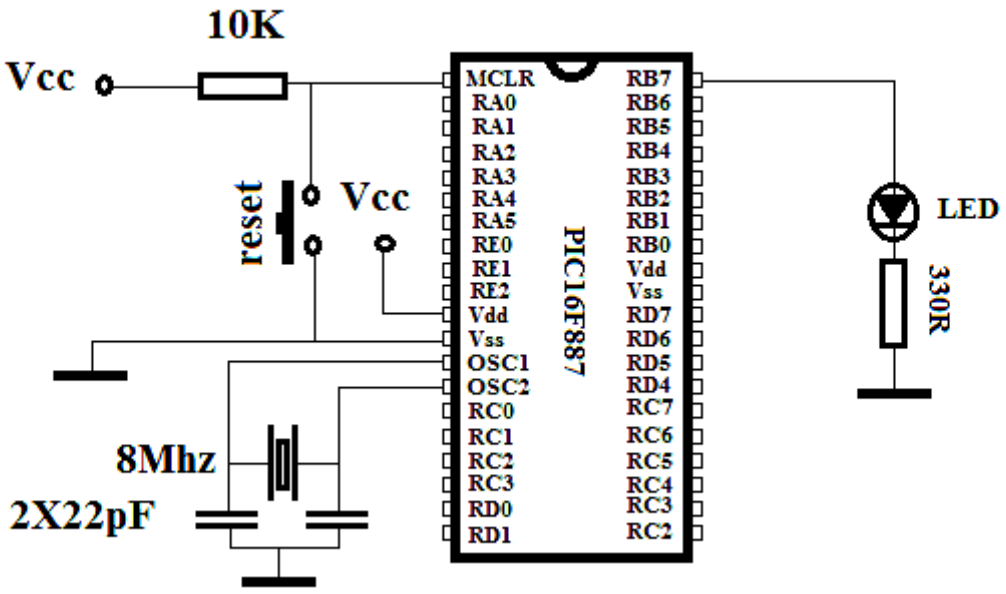
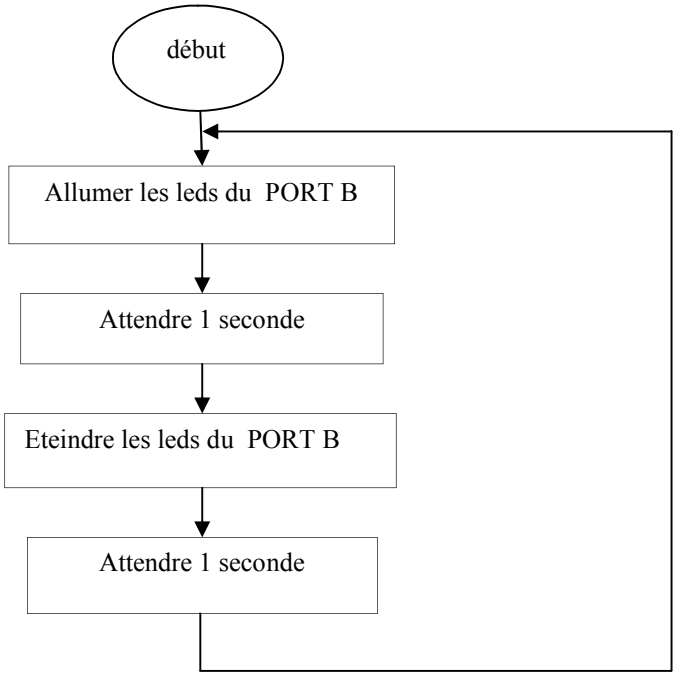


Fig. III-1

Organigramme :



TP 2 :

Compteur binaire

But du TP :

Ce montage à base de PIC 16F84A permet de réaliser un compteur binaire on utilisant le timer du pic et de le diriger sur 8 leds. La temporisation est réglable par logiciel (programme). L'interrupteur reset permet de relancer le compteur.

Description :

C'est un compteur binaire sur 8 bits qui compte de b'00000000' jusqu'à b'11111111'

Schéma électrique :

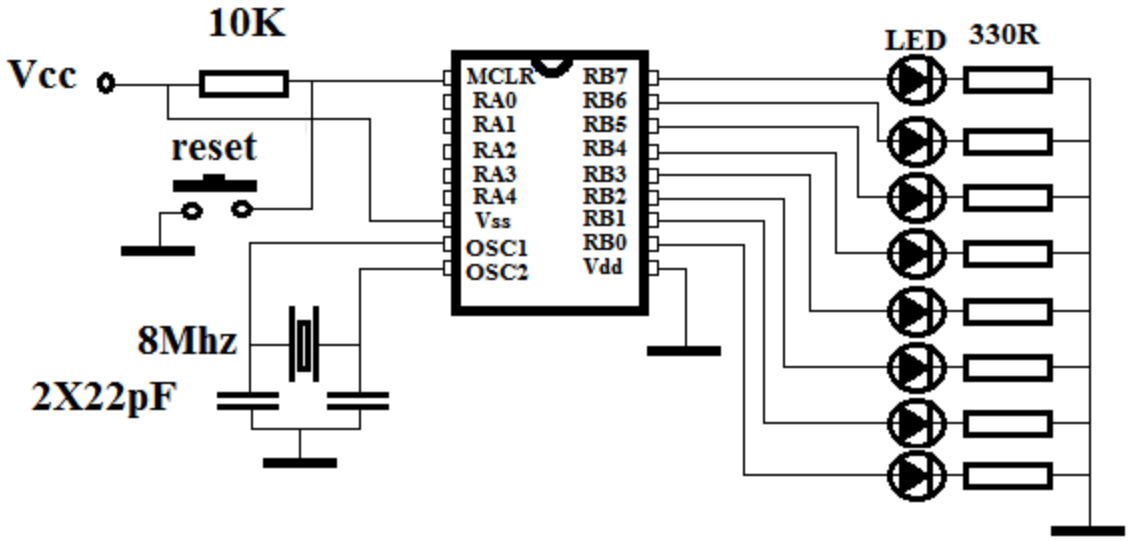
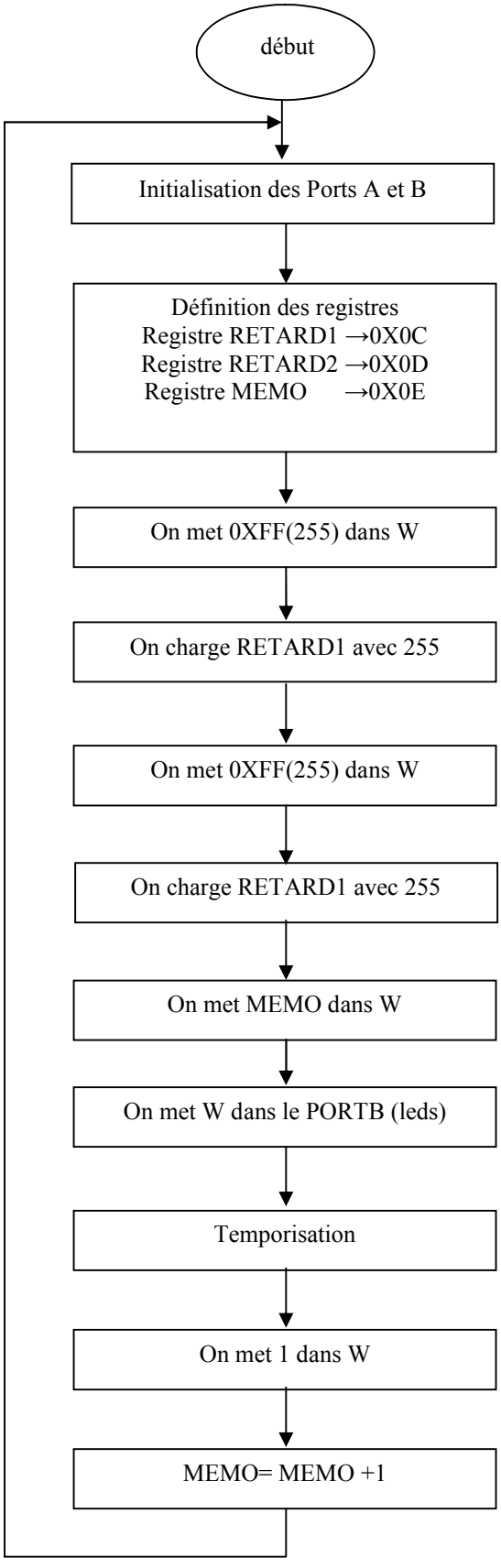


Fig. III-2

Le compteur binaire est un timer: une case mémoire (RAM) qui s'incrémente à chaque front montant ou descendant d'un signal en créneaux de référence , Si on choisit l'horloge interne du PIC pour incrémenter le timer, alors on compte des incréments de TEMPS, on peut donc mesurer des durées.
Si on utilise la patte avec un signal externe, on peut soit mesurer des durées, soit compter des évènements. (D'ou le nom timer/compteur).
Les opérations de base permises sont : modifier la valeur du timer, lancer le timer (autoriser l'incrémentation à chaque front de l'horloge), le stopper. La configuration minimale consiste à fixer la source du signal d'horloge (externe ou interne) et un coefficient de division pour la fréquence de ce signal (une puissance de 2). Une fois lancé, le timer s'incrémente indéfiniment : il fait passer à 1 un bit d'un registre (un "flag" ou drapeau) lorsqu'il passe de la valeur maximale (255 pour 8 bits) à 0, mais ne s'arrête pas
Si on choisit l'horloge interne du PIC pour incrémenter le timer, alors on compte des incréments de TEMPS, on peut donc mesurer des durées. Si on utilise la patte avec un signal externe, on peut soit mesurer des durées, soit compter des évènements.

Organigramme :



TP 3 :

Le chenillard

But du TP :

Ce montage d'initiation à base de PIC 16f84A permet de réaliser un chenillard.
L'interrupteur reset permet de relancer le chenillard.
La cadence de clignotement est modifiable par programme.

Schéma électrique :

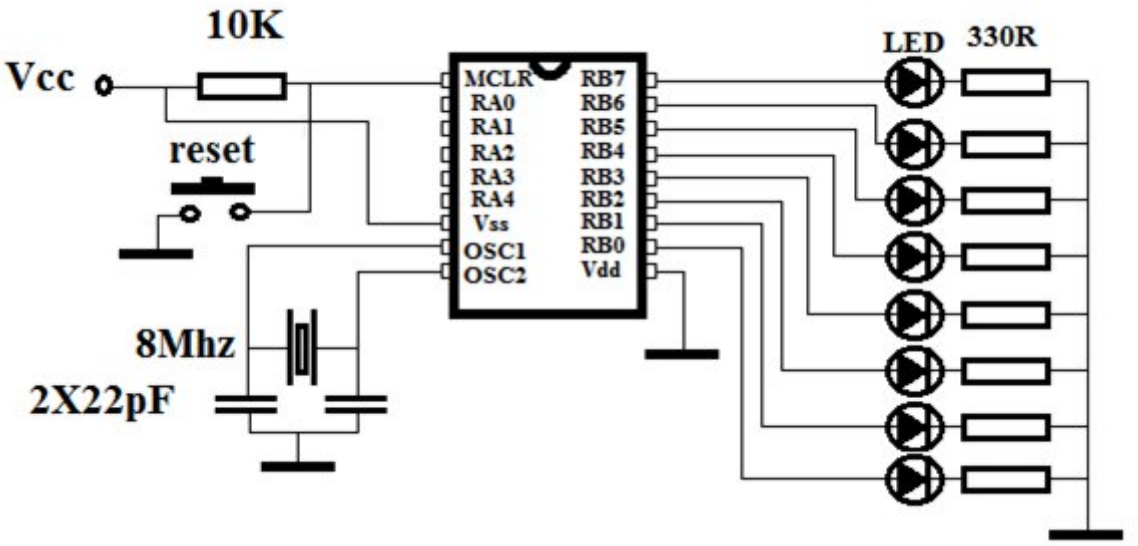
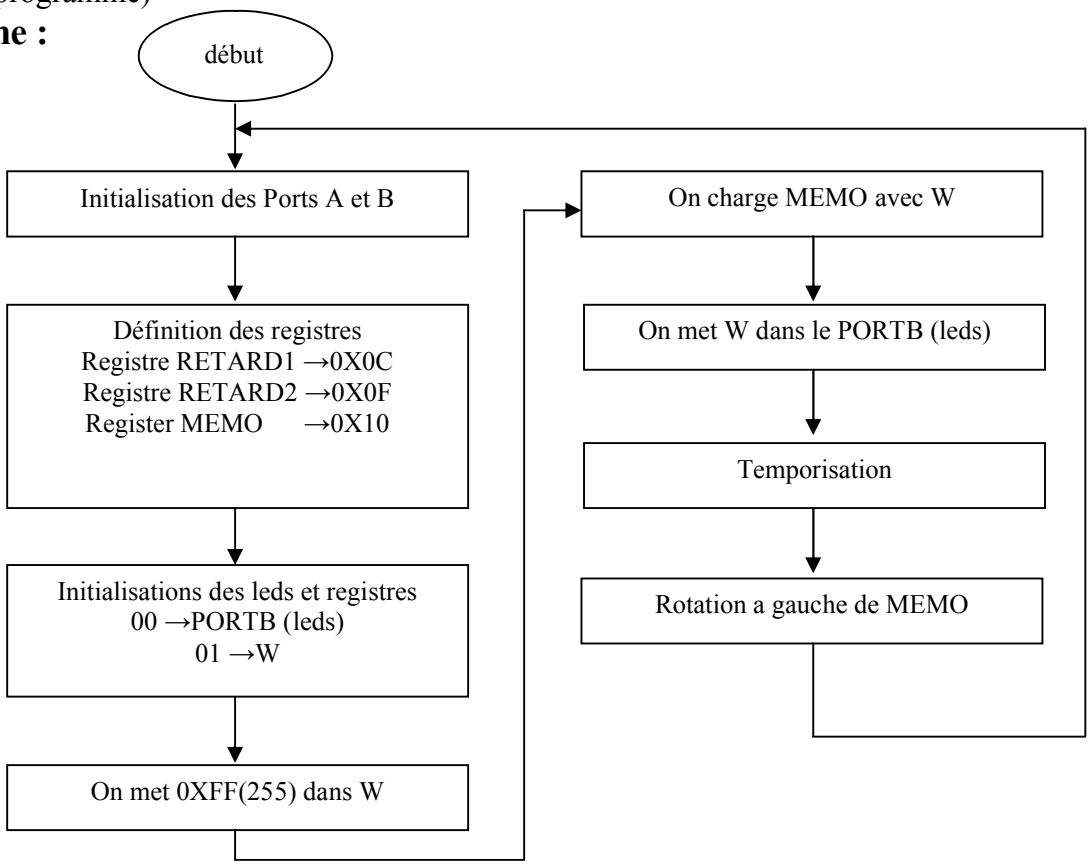


Fig. III-3

L'horloge assurant le cadencement des instructions du PIC n'a nullement besoin d'être très stable, et un simple réseau RC (C1 / R14 + RV1) suffit, chaque LED est allumée selon la séquence décidé d'avance dans le logiciel (programme)

Organigramme :



TP 4 :

Feu de carrefour

But du TP :

Fournir des informations lumineuses aux usagers de la route, selon des cycles préétablis, afin d'assurer l'écoulement optimum de la circulation au niveau d'une intersection.
Ce montage d'initiation à base de PIC 16f84A permet de simuler un feu de carrefour tricolore. Toutes les temporisations sont modifiables au programme.
Un deuxième bouton permet de déclarer les feux en mode clignotement.

Organigramme :

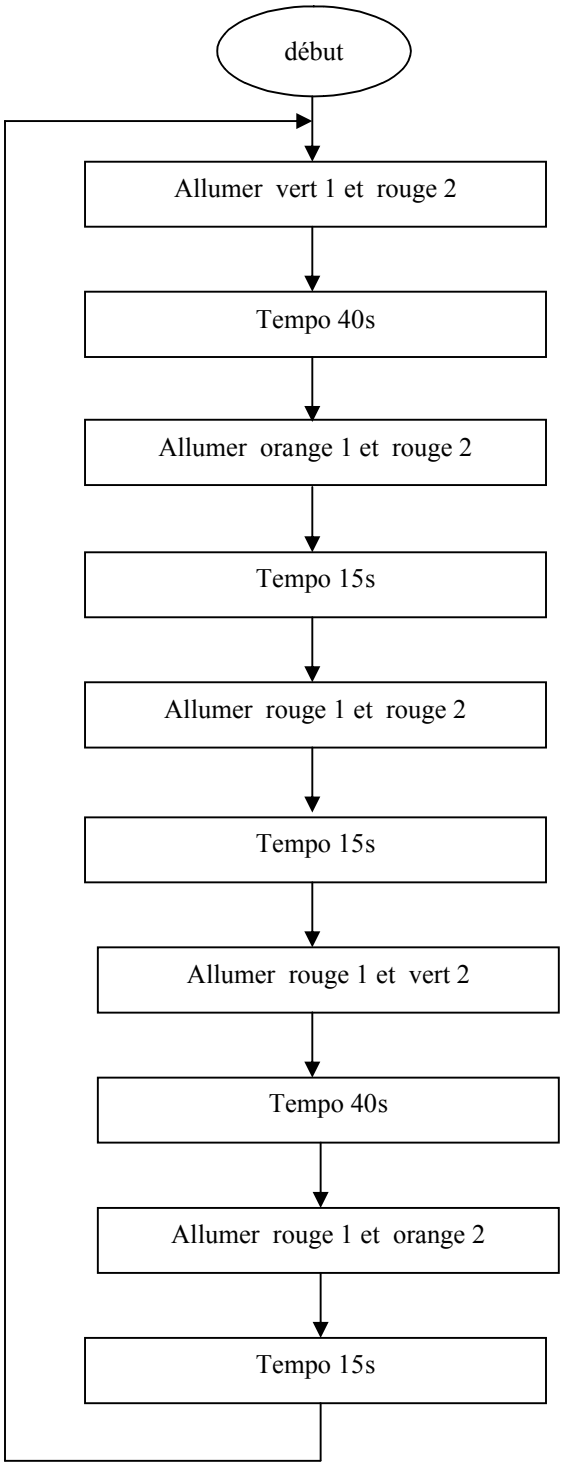


Schéma électrique :

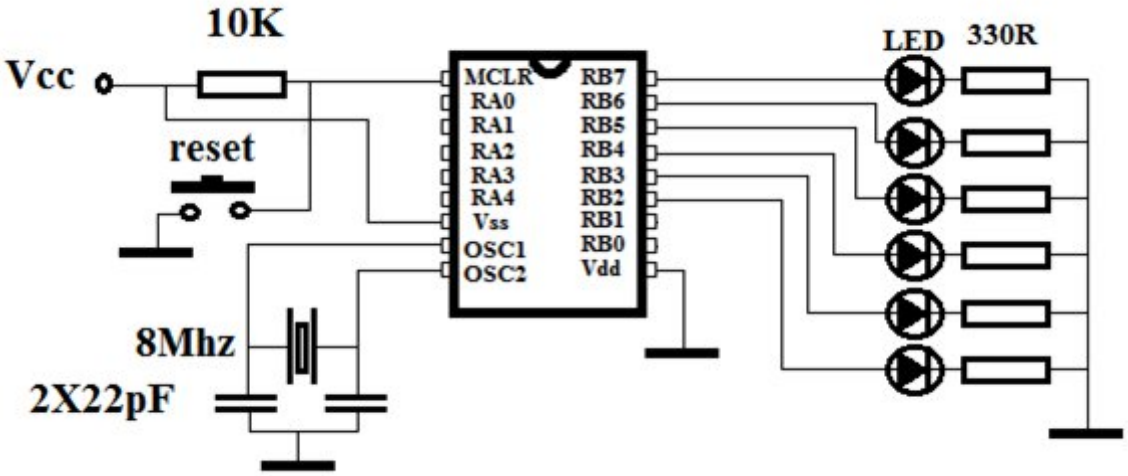


Fig. III-4-1

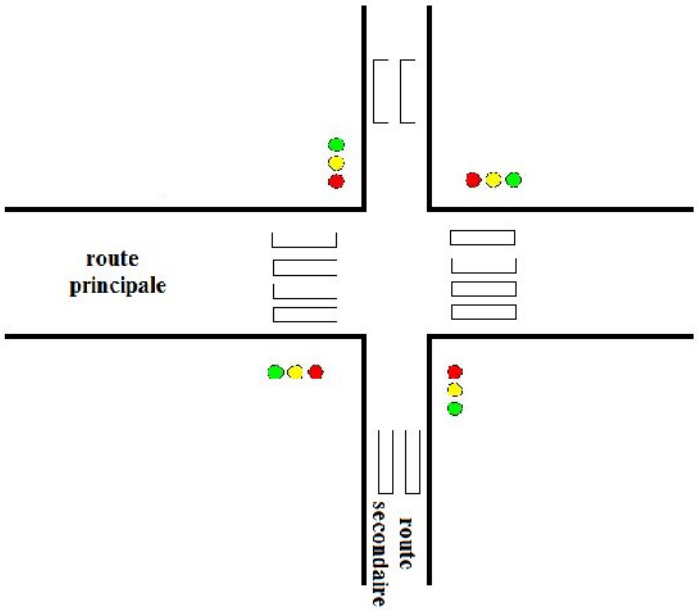


Fig. III-4-2

TP5 :

Interruption

But du TP :

Le but de ce TP est de savoir comment utiliser le TMR1 et son interruption
Le programme principal est le clignotement des leds sur le PORTB.

Organigramme :

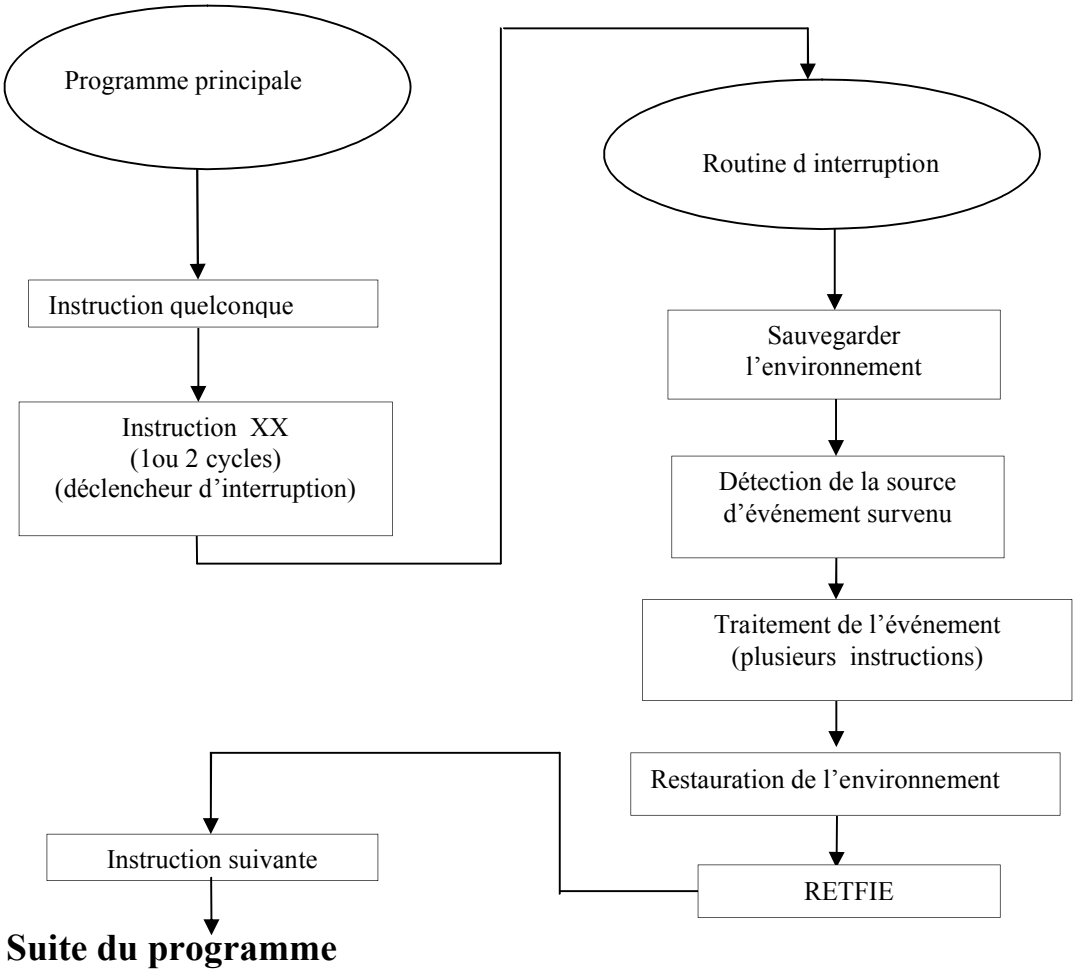


Schéma électrique :

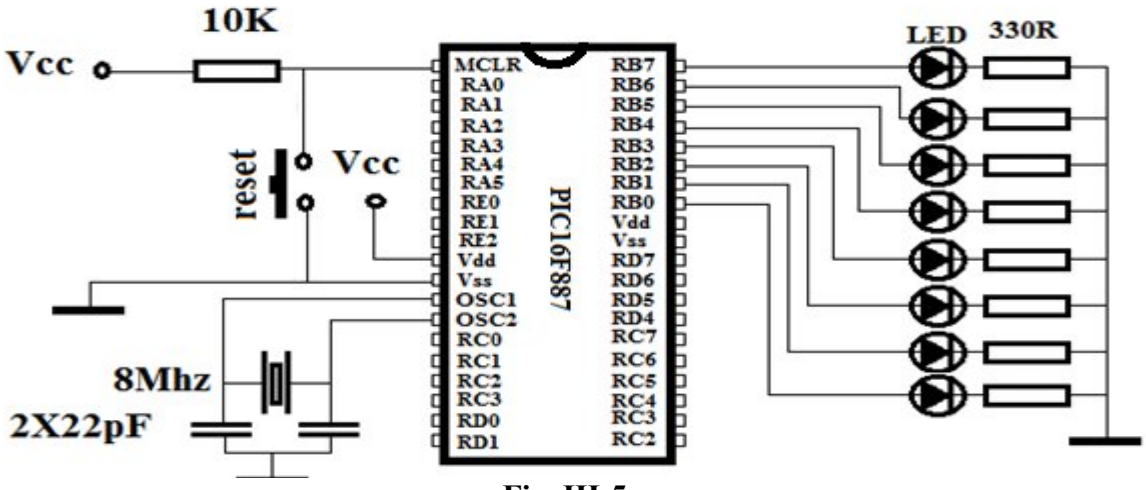


Fig. III-5

TP 6 :

Temporisation

Schéma électrique :

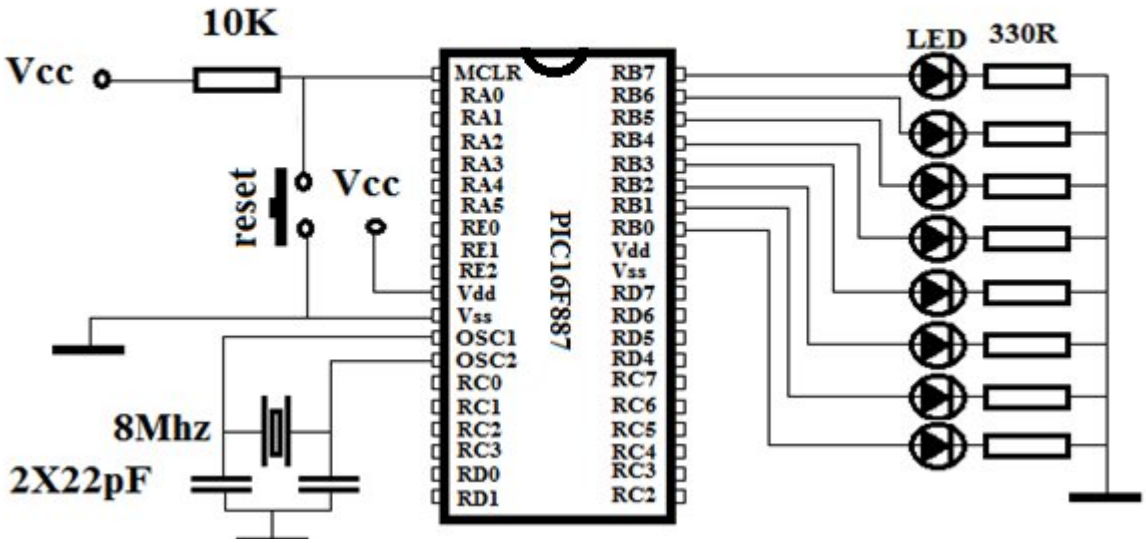
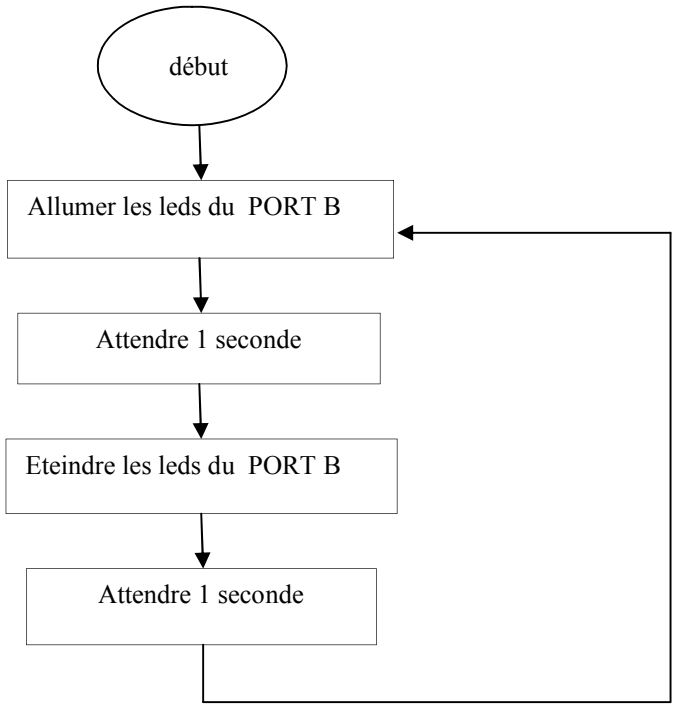


Fig. III-6

Organigramme :



Exemple de programme de tempo :

Tempo

DECFSZ retard1, F **;** on décrémente retard1 et on saute la prochaine instruction
GOTO tempo **;** le registre retard1 = 0 sinon retour à tempo
MOVLW 0xFF **;** on met 255 dans le registre W
MOVWF retard1 ; on charge retard1 avec 255 (FFh contenu du registre W)
DECFSZ retard2, F ; on décrémente retard1 et on saute la prochaine instruction
GOTO tempo ; le registre retard1 = 0 sinon retour à tempo
RETURN ; retour au programme principal après l'instruction CALL
END ; fin du programme

TP 7 :

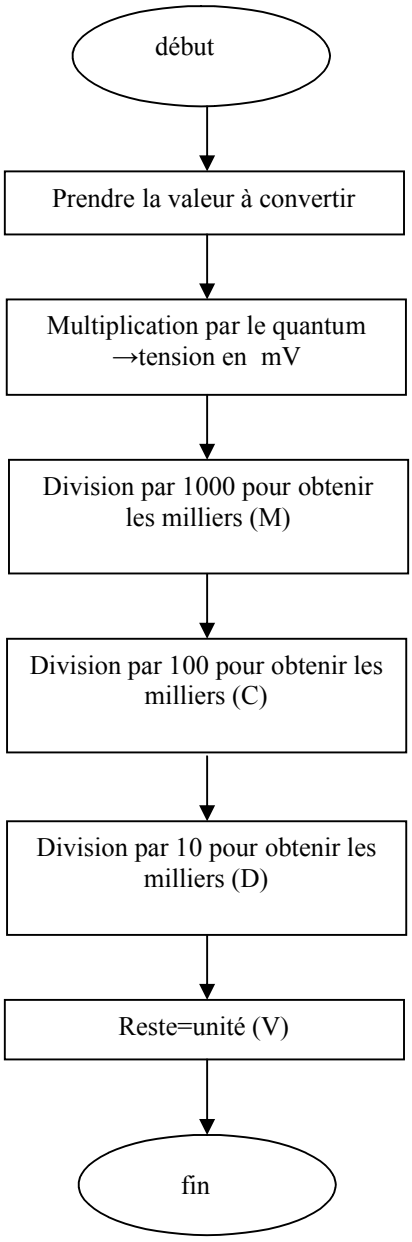
Convertisseur analogique/numérique

But du TP :

Affichage du résultat De la conversion Analogique /Numérique en employons la bibliothèque ADC. Les résultats sont sur les leds des PORTC et PORTD

Le convertisseur A/D du PIC16F887 est utilisé dans cet exemple. Un signal analogique variable est appliqué à la broche AN2, le résultat de la conversion est affiché sur les ports PORTB et PORD (8 Leds sur PORTD et 2 leds sur PORTB). La masse est utilisée comme une tension de référence négative Vref, alors que le VCC est une référence de tension positive.

Organigramme :



L’acquisition de la tension analogique est sur l'entrée RA2, et affichage sur les leds des PORTB et PORTD. Une tension de 0 V se traduit par l'extinction de toutes les leds, une tension de 1V se traduit par l'allumage de la première et la septième led du PORTB, et une tension de 5 V) se traduit par l'allumage de toutes les leds du PORTB et les deux leds du PORTD. L'écart entre chaque led est linéaire,

Le pas est donc d'environ 1,53V.

Les acquisitions ont lieu toutes a chaque changement de la valeur de la résistance variable de 5KΩ,

Schéma électrique :

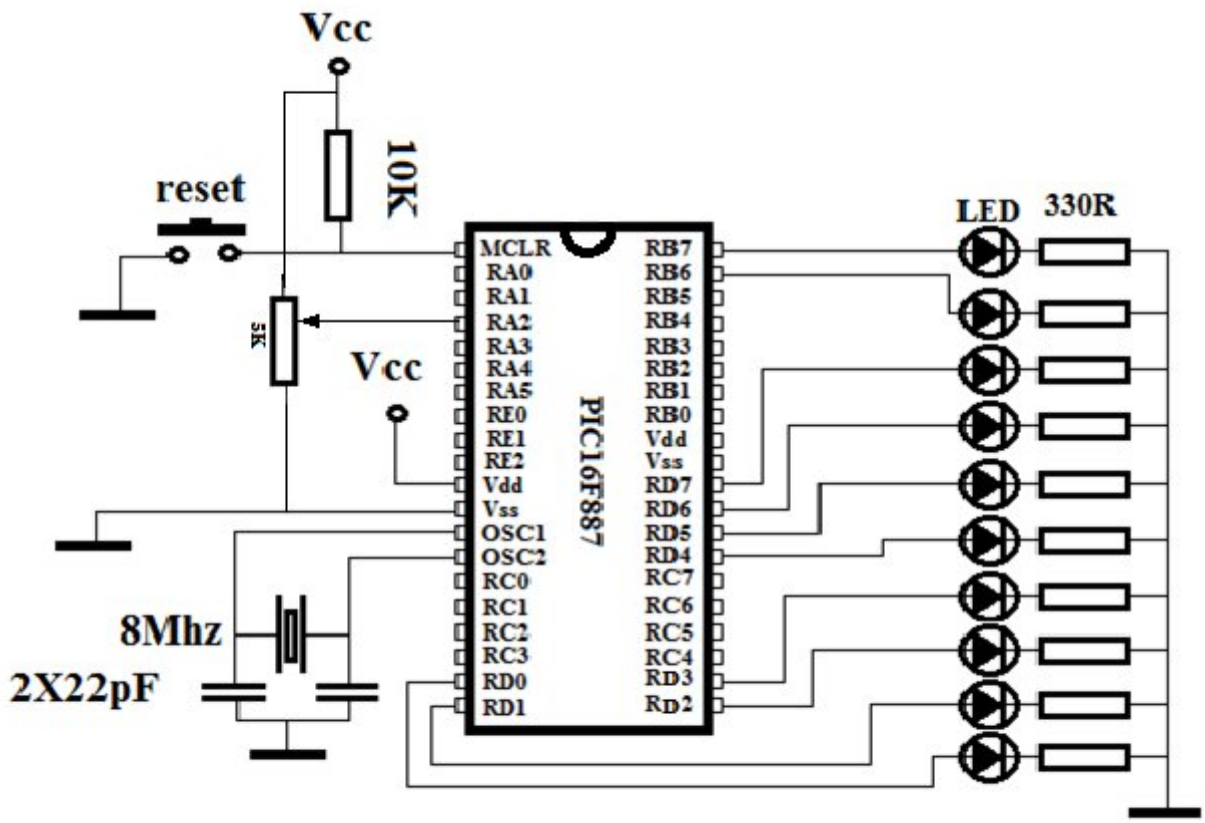


Fig. III-7

Le convertisseur A/D produit toujours un nombre binaire sur 10 bits, ce qui signifie qu'il détecte 1024 niveaux de tension (210=1024) au total. La différence entre deux niveaux de tension n'est pas toujours identique. Tant que la différence entre Vref+ et Vref- est faible, la différence entre deux valeurs est moins importante. Le convertisseur d'A/D peut détecter de légers changements de tension.

Il est nécessaire d’ajouter la bibliothèque CDA à la bibliothèque du compilateur avant de la compilation.

TP 8 :

Afficheur LCD

But du TP :

Une simple démonstration de la bibliothèque des fonctions SPI LCD. La bibliothèque envoie une commande de 4 bits au port d'extension et contrôle sa sortie ou l'afficheur LCD est connecté.

Schéma électrique :

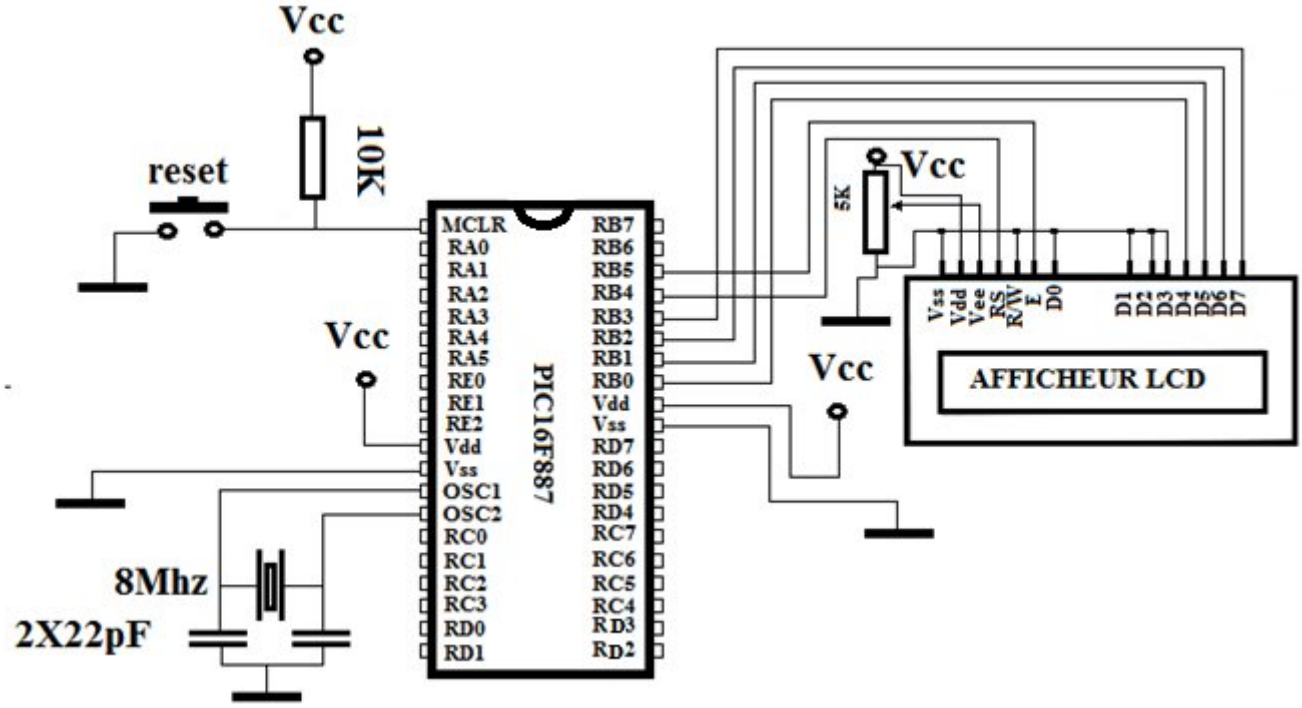
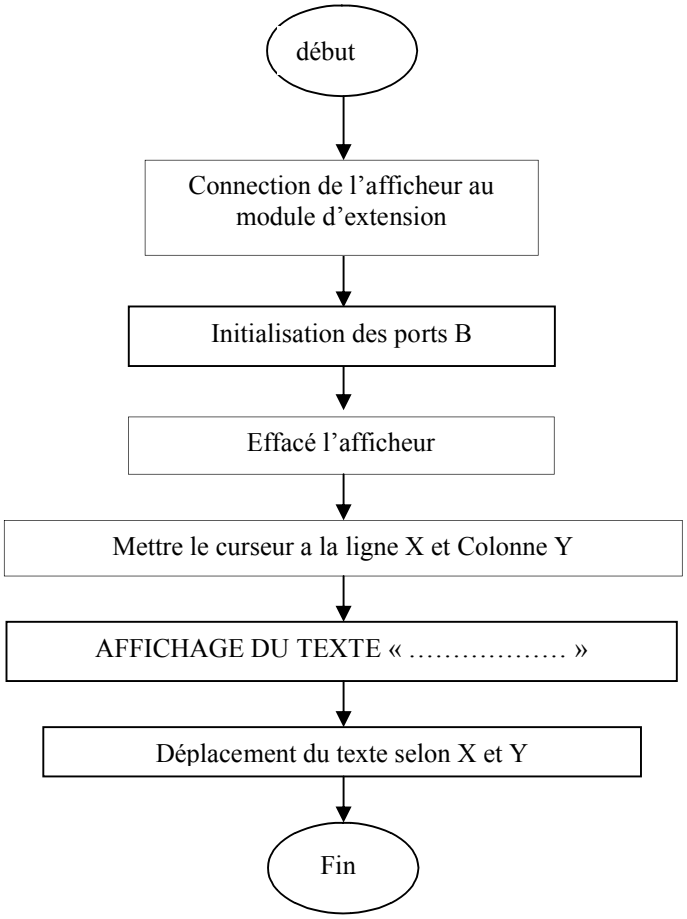


Fig. III-8

Organigramme :



TP 9:

Le clavier 4x4

But de TP :

Exemple pour l'utilisation de la bibliothèque KEYPAD de mikroC PRO for PIC. Elle supporte un clavier de 4 lignes et 4 colonnes .le code est converti on ASII (0…9, A..F) puis afficher sur le LCD
Un clavier numérique est juste un ensemble de boutons poussoirs reliés d'une telle manière pour former des « lignes » et des « colonnes », de ce fait réduisant un certain nombre de broches d'entrée/sortie nécessaires pour leur raccordement. Un clavier numérique avec 16 boutons poussoirs disposés dans 4 lignes et 4 colonnes est utilisé dans cet exemple. La bibliothèque du clavier numérique 4x4 contient toutes les fonctions nécessaires pour lire ce clavier numérique comme pour initialiser le port ou il est relié. Afin de démontrer l'opération du clavier numérique 4x4, le message sera montré sur un LCD.
A chaque fois qu'on presse un bouton poussoir, sa valeur s'affiche avec le nombre de fois qu'il est pressé.
Il est nécessaire de vérifier la présence des bibliothèques suivantes avant de compiler :

- Clavier 4x4 (KEYPAD 4X4)
- LCD

Schéma électrique :

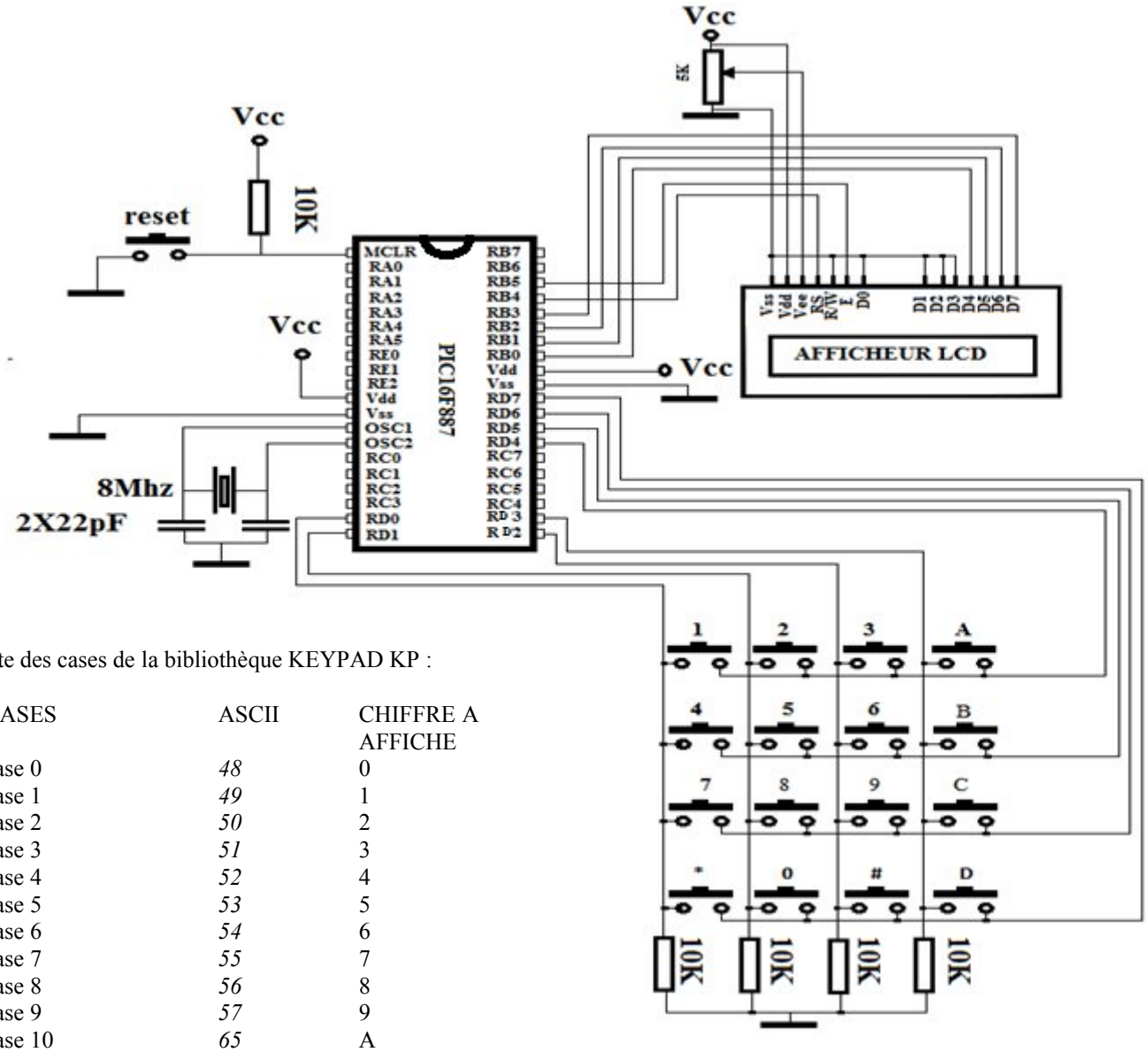
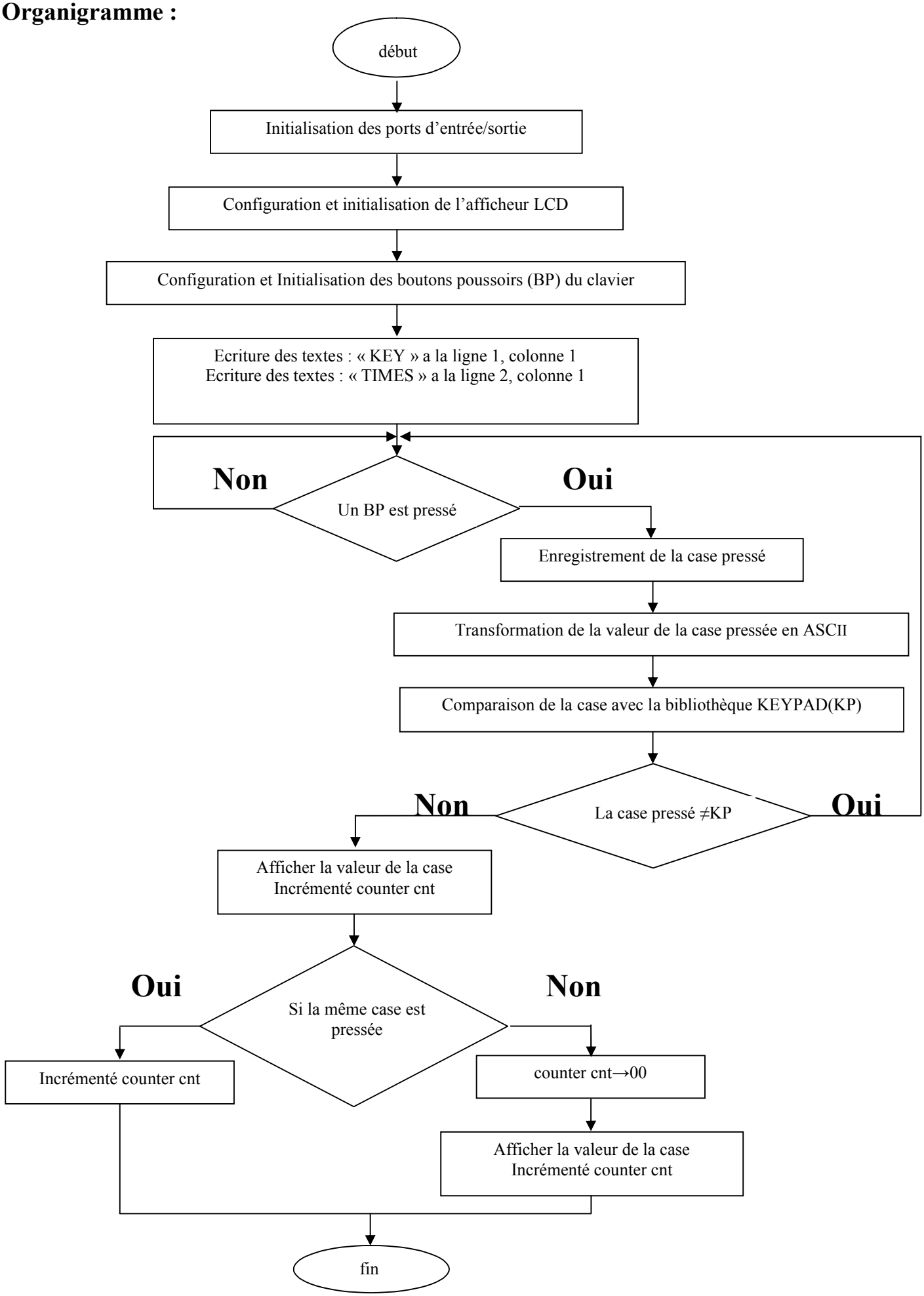


Fig. III-9

Liste des cases de la bibliothèque KEYPAD KP :

CASES	ASCII	CHIFFRE A AFFICHE
case 0	*48*	0
case 1	*49*	1
case 2	*50*	2
case 3	*51*	3
case 4	*52*	4
case 5	*53*	5
case 6	*54*	6
case 7	*55*	7
case 8	*56*	8
case 9	*57*	9
case 10	*65*	A
case 11	*66*	B
case 12	*67*	C
case 13	*68*	D
case15	*35*	#
case16	*42*	*

Organigramme :



TP 10 :

Le générateur de signaux PWM

But du TP :

La maquette EASYPIC6 peut être utilisé comme un générateur de signal PWM de trois fréquence a sélectionnée, elle est ajustable sur 10 bits, ce programme utilise le compilateur de MikroC PRO et l'afficheur LCD.
Sur l'afficheur on a :

- Le cycle de l'impulsion (%)
- La valeur de l'impulsion (Hz)
- La période de l'impulsion (µs)

Cet exemple illustre l'utilisation du module CCP1 en mode PWM, la largeur des impulsions de sortie (PORTC, 2) peut être changé avec les touches symboliquement marqué comme «sombre » et «brillant », tandis que la largeur ensemble est visible comme une combinaison binaire sur PORTB. Le fonctionnement de ce module est sous le contrôle des procédures appartenant à la bibliothèque spécialisée PWM. Trois d'entre eux sont utilisés :
1. PWM1_init a le prototype: procédure sous PWM1_Init (freq const comme longint) Paramètre Freq définit la fréquence du signal PWM exprimée en herz. Dans cet exemple, il est de 5 KHz.
2. PWM1_Start a le prototype: procédure sous PWM1_Start ()
3. PWM1_Set_Duty a le prototype: procédure sous PWM1_Set_Duty (duty_ratio sombre que byte) Les jeux de paramètres duty_ratio durée de l'impulsion dans une séquence d'impulsions. La bibliothèque contient également PWM la procédure PWM_Stop utilisée pour désactiver ce mode. Son prototype est: PWM1_Stop procédure sub.

Schéma électrique :

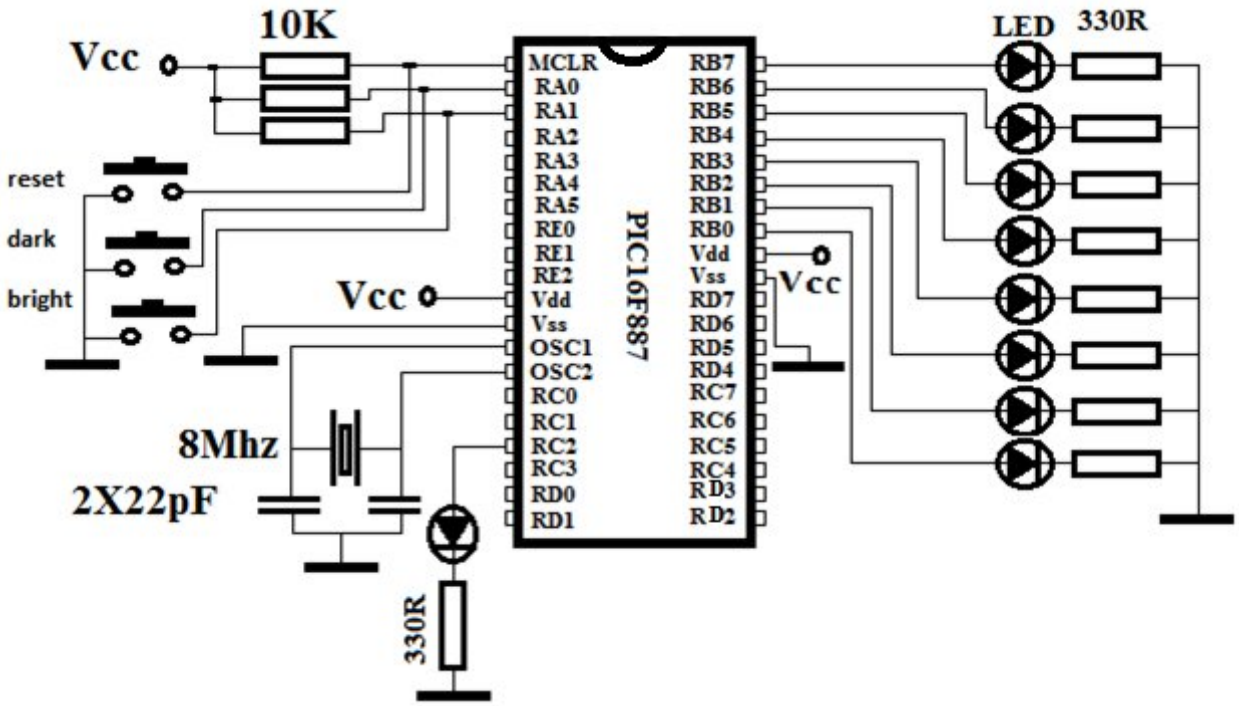
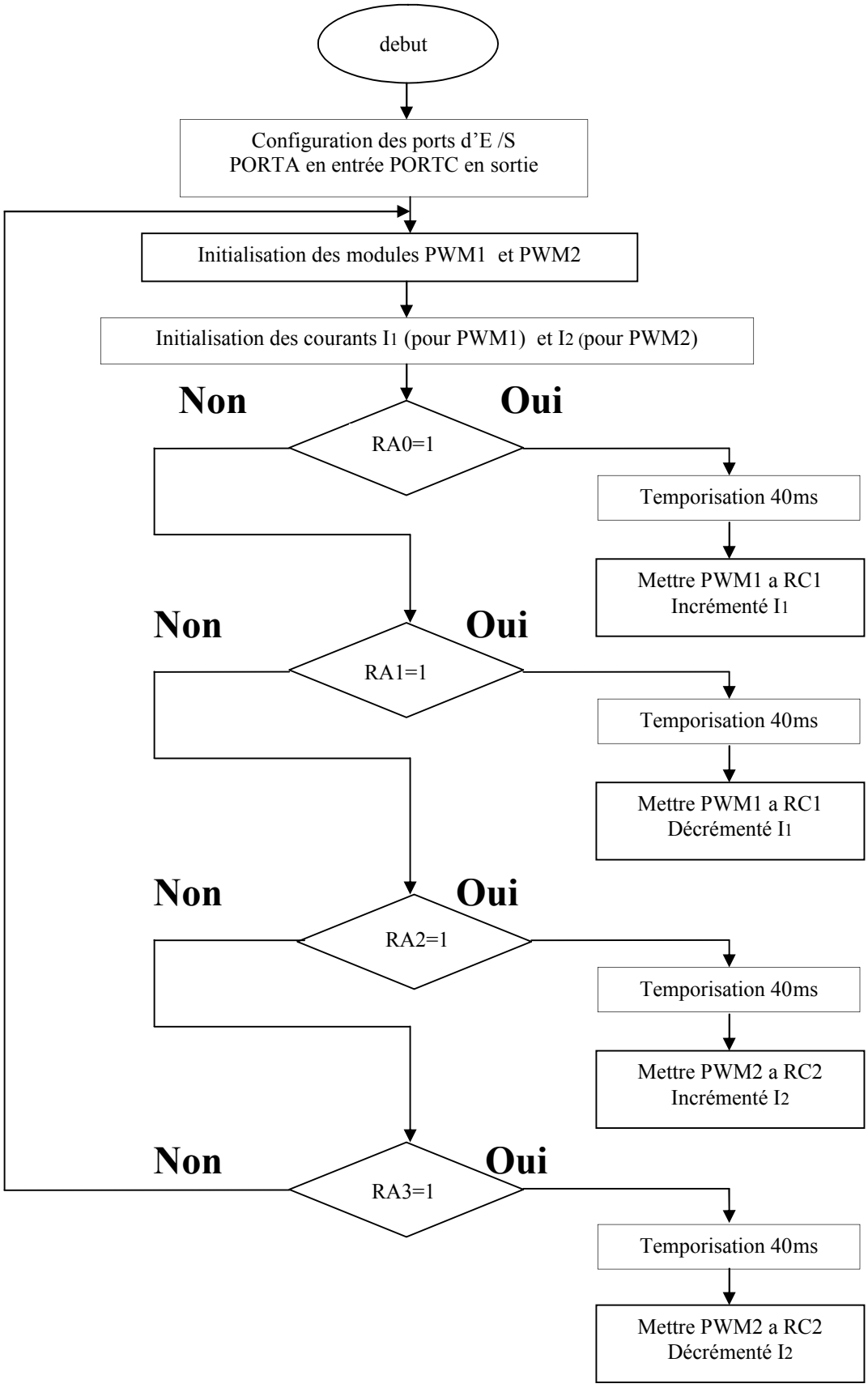


Fig. III-10

Afin de faire fonctionner ce TP correctement, il est nécessaire de vérifier la présence des bibliothèques suivantes dans la bibliothèque avant de compiler :

- PWM
- Buttons

Organigramme :



TP 11 :

Ecriture et lecture dans la mémoire EEPROM

But du TP:

Une démonstration d'utilisation des fonctions secondaires de bibliothèque SUB FUNCTIONS pour la manipulation d'EEPROM interne du PIC. Les données sont écrites à l'EEPROM puis les affiches sur les PORTB et PORTD.

Utilisation de la mémoire EEPROM

Cet exemple illustre l'écriture et la lecture dans la mémoire intégré EEPROM. La boucle principale lit l'emplacement de la mémoire EEPROM à l'adresse 5, le PORTB est incrémenté et l'état de l'entrée PORTA.2 est à (1). Au moment d'appuyer sur le bouton MEMO, un numéro enregistré dans le PORTB est sauvegardés dans la mémoire EEPROM à l'adresse 5 puis affiché sur les leds du PORTD en format binaire.

Organigramme :

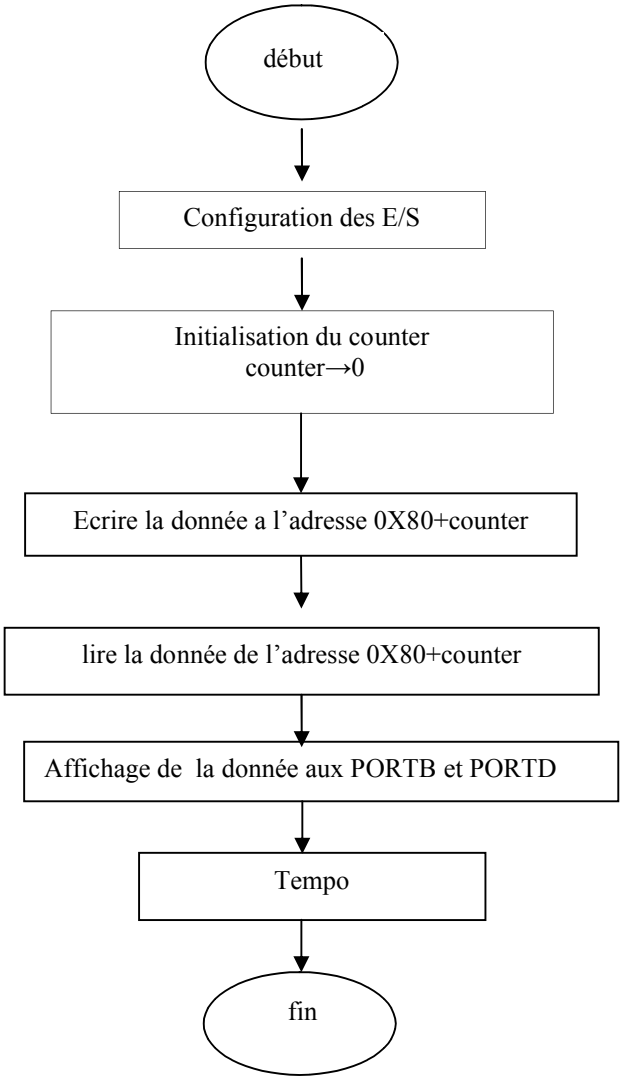


Schéma électrique :

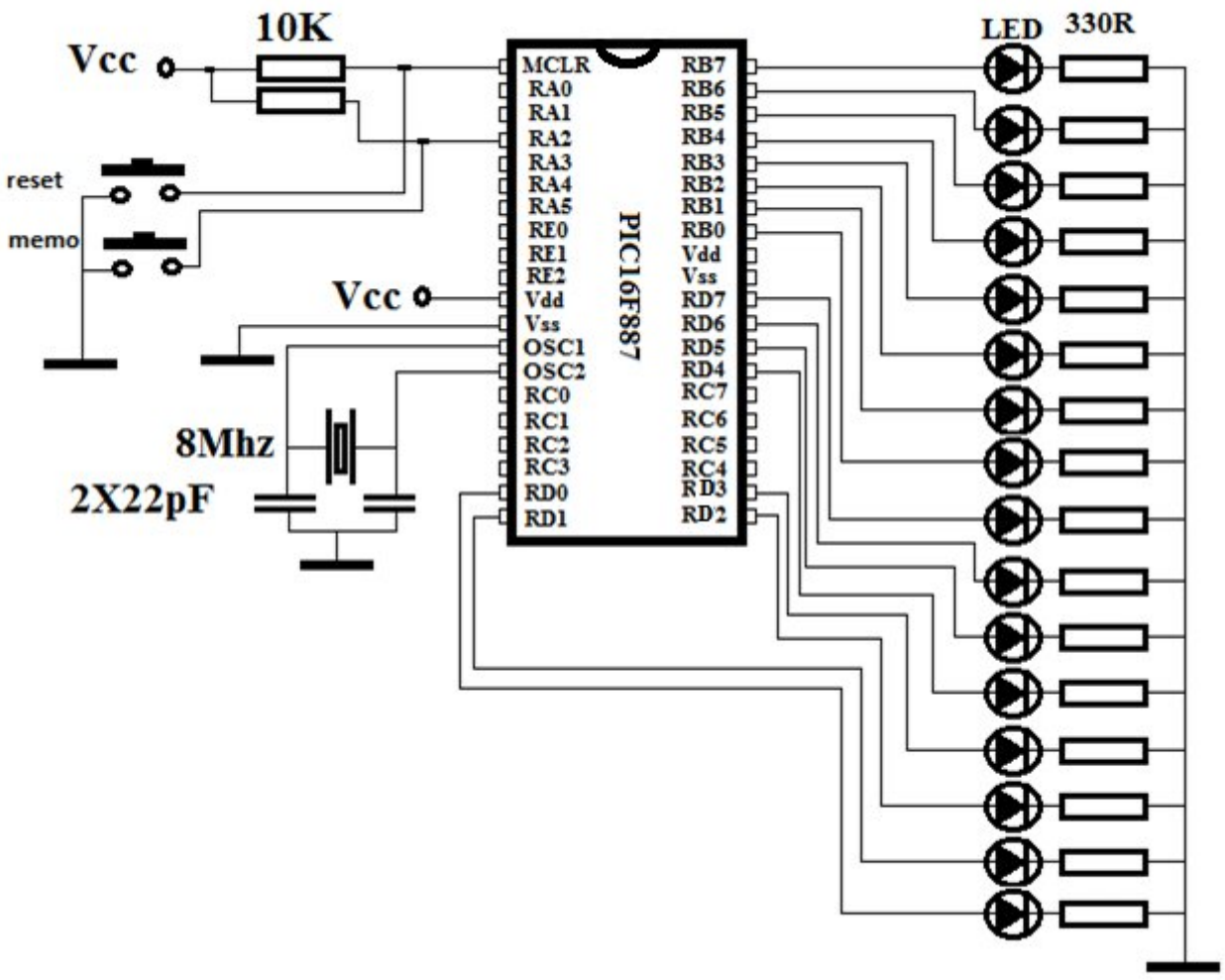


Fig. III-11

Afin de faire fonctionner ce TP correctement, il est nécessaire de vérifier la présence des bibliothèques suivantes dans la bibliothèque avant de compiler :

- EEPROM

Afin de vérifier si ce programme fonctionne correctement, il suffit de presser le bouton MEMO et puis éteignez le microcontrôleur. Après le redémarrage, le programme affichera sur PORTD la valeur stockée dans la mémoire EEPROM à l'adresse 5, au moment de l'écriture, cette valeur a été affichée sur PORTB.

TP12 :

Générateur de sonnerie

But du TP :

Générer des sonneries en utilisant la maquette EASYPIC6 et le port d'extension

Schéma électrique :

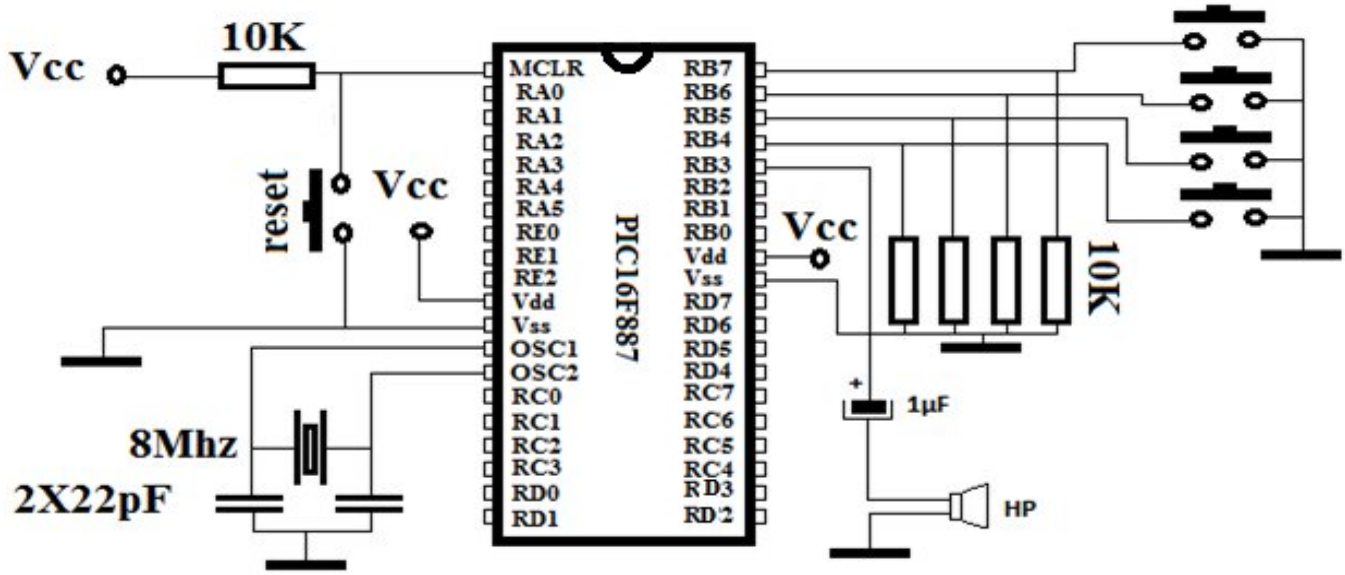


Fig. III-12

Les signaux audio sont employés souvent quand il est nécessaire d'attirer l'attention de l'utilisateur, pour confirmer qu'un des boutons poussoirs est pressé, pour avertir que les valeurs minimum ou maximum sont atteintes etc.... Cet exemple illustre comment produire un signal sonore on utilisant des fonctions appartenant à la bibliothèque SOUND.

Afin de faire fonctionner ce TP correctement, il est nécessaire de vérifier la présence des bibliothèques suivantes dans la bibliothèque avant de compiler :

- Buttons
- Sound

La combinaison des tones nous permet d'avoir des déférentes sonneries

Exemple : Pour la sonnerie 2 :

Program

```
void Melody2() {
   unsigned short i;
   for (i = 9; i > 0; i--) {
      ToneA(); ToneC(); ToneE();
   }
}
```

son	Fréquence (Hz)	Durée (ms)
Tone1	*659*	250
Tone2	*698*	250
Tone3	*784*	250
Tone A	*880*	50
Tone D	*1046*	50
Tone E	*1313*	50

TP 13 :

Lecture de la date et de l'heure du RTC et affichage sur le LCD

But du TP :

Utilisation du bus I2C pour la lecture de la date et de l'heur du RTC1307
Dans ce TP le mettre est le microcontrôleur et l'esclave c'est le RTC

Schéma électrique :

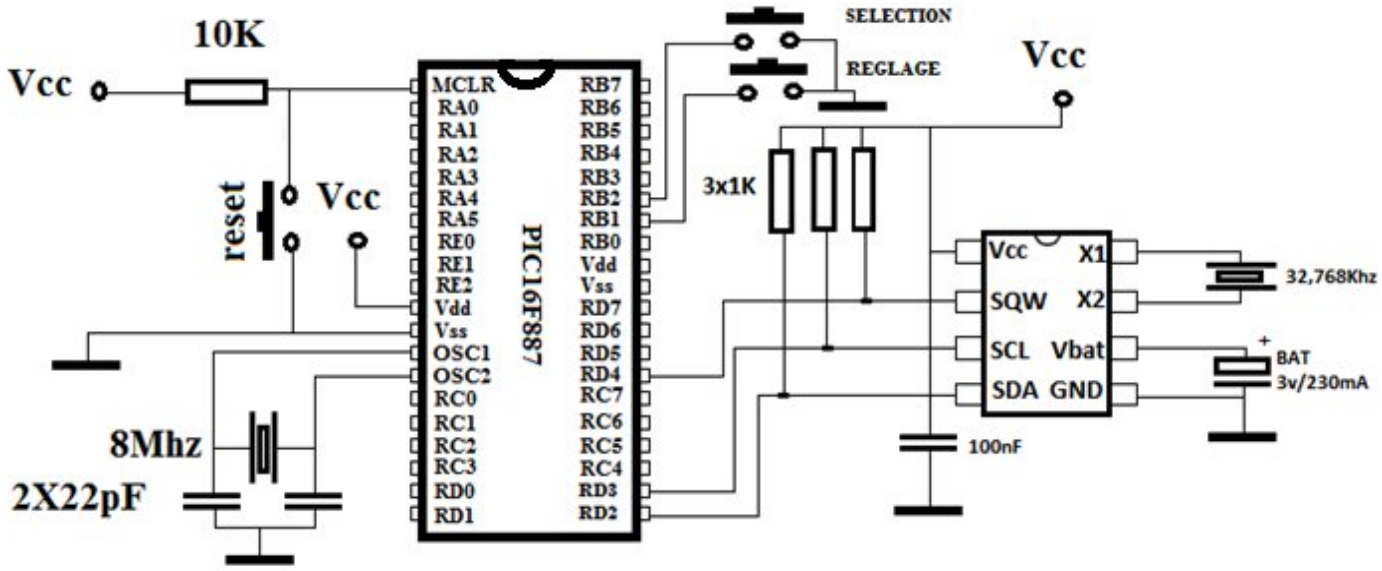


Fig. III-13

Présentation du DS1307 (Real Time Clock) :

Le circuit Dallas DS1307 est une horloge temps réel (Real Time Clock), qui fournit les secondes, minutes, heures, jours, dates, mois et années.
Le Dallas DS1307 est associé (les broches 1 et 2) à un quartz d'horlogerie de fréquence nominale 32,768 kHz.
Les années sont prises en compte (jusqu'en 2100).
Le DS1307 s'interface avec un bus I2C, en configuration esclave :

- Le DS1307 travaille dans le mode standard (fréquence d'horloge f SCL de 100 kHz)
- L'adresse I2C (7 bits) du DS1307 est : 1 1 0 1 0 0 0

En option, on peut brancher une pile de sauvegarde entre la broche 3 et la masse.
Avec une pile au lithium 3 V / 48 mA, la date et l'heure sont conservées pendant plus de 10 ans en l'absence de l'alimentation principale

Principe de fonctionnement :

6-1- Initialisation

A la mise sous tension, le microcontrôleur 16F887 configure :

- son module MSSP en mode I2C maître, et la fréquence d'horloge (f SCL) à 100 kHz
- l'horloge temps réel DS1307 :
  - en mode 24 heures (00 à 23 heures)
  - avec la fréquence du signal de sortie SQW/OUT (broche 7) à 1,000 000 Hz
- le module LCD alphanumérique (instruction **Set Function** pour une utilisation en mode d'interface 4 bits)

Le module **Timer1** (16 bits) du microcontrôleur est activé, ce qui donne lieu à une interruption toutes les 105 ms environ (104,858 ms exactement).

L'interruption RB0/INT est activée, ce qui provoque une interruption toutes les 1000,000 ms.

Réglage de l'heure et de la date

A la première utilisation, l'affichage à la mise sous tension est :

Di 01/01/2000

00:00:00

Deux boutons poussoirs ont été prévus pour le réglage de l'heure et de la date.

A chaque interruption du **Timer1** (toutes les 105 ms environ), la routine de l'interruption du **Timer1** teste l'état du bouton poussoir SELECTION et du bouton poussoir REGLAGE.

La routine de l'interruption du **Timer1** gère également toutes les étapes du réglage de l'heure et de la date :

On entre dans le mode réglage en appuyant sur le bouton poussoir SELECTION.

Les deux lettres **Di** se mettent à clignoter.

Il faut ensuite appuyer sur le bouton poussoir REGLAGE pour faire défiler le jour (1ère étape) :

Lu -> Ma -> Me -> Je -> Ve -> Sa -> Di ...

On valide en appuyant sur le bouton poussoir SELECTION, et on passe automatiquement au réglage du chiffre des dizaines du numéro de jour (2ème étape) :

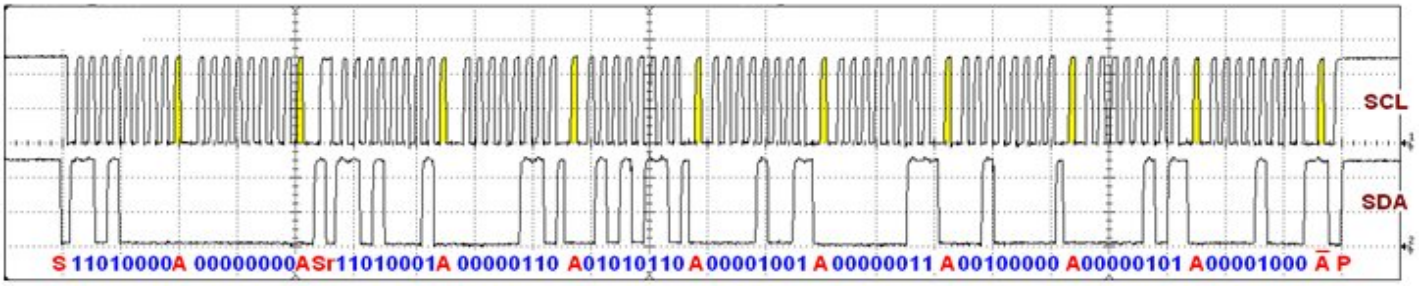
0 -> 1 -> 2 -> 3 -> 0 ...

On finit par le réglage du chiffre des unités des minutes (3 ème et dernière étape).

On valide avec le bouton poussoir SELECTION, et on retourne alors au mode de fonctionnement normal.

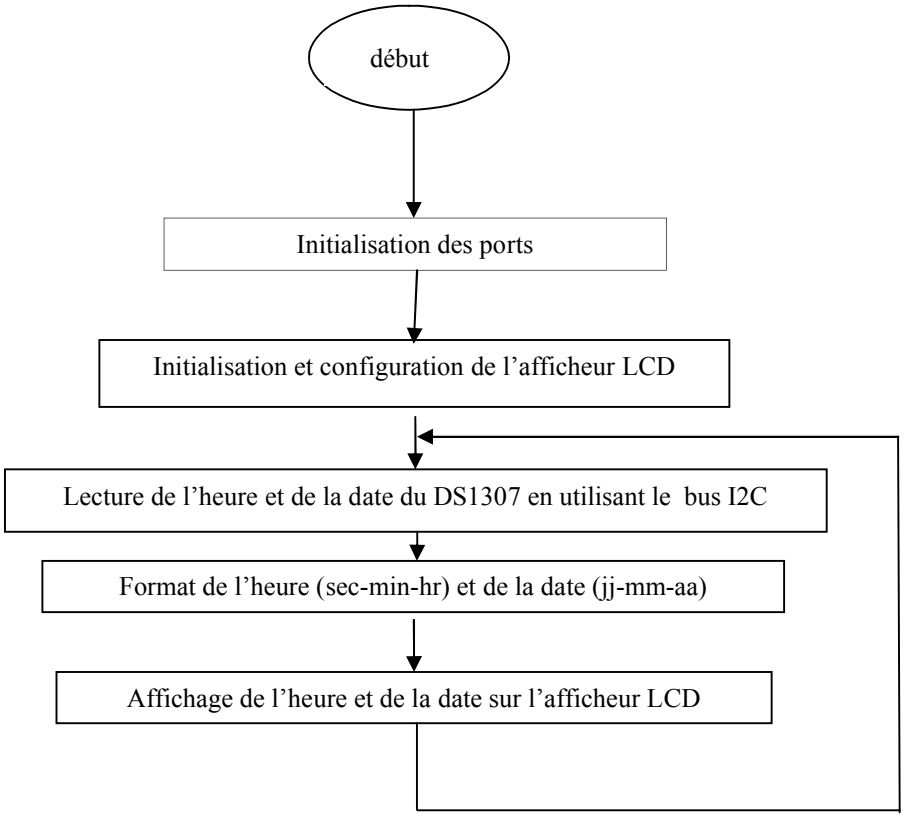
Toutes les 1000,000 ms la routine de l'interruption RB0/INT est appelée :

Le PIC 16F887 lit l'heure et la date courante dans le DS1307, la communication se fait par le bus I2C.



Dans cet exemple de chronogramme, il est 9 heures 56 minutes 6 secondes, le mardi 20 mai 2008

Organigramme :



TP 14 :

Contrôle de vitesse des moteurs DC

But du TP :

Utiliser le signal PWM pour commander la vitesse des moteurs DC.
Fonctionnement :
Le PORTB dispose de 6 boutons poussoir (enfoncé = 5v);
-changement de l'amplitude de la PWM (RB2/RB3)
-changement du cycle de PWM (RB4/RB5)
-Changement de fréquence; 3 fréquences de sortie (RB6)
-Affichage PWM (RB7)

Le signal de sortie est sur PORTC.F2 et le TMR2 est utilisé à pleine résolution pour générer la PWM.
Sur l'afficheur LCD, on trouve :
-La fréquence générée Hz
-Le cycle en pourcentage; 0-100%
-La largeur d'impulsion PWM 10 bits dans les chiffres; 0-1023
-La largeur d'impulsion en microsecondes;µs
Il existe trois fréquences de sortie sélectionnables, il est entièrement réglable sur 10bit
Pour le quartz de 8MHz (standard EasyPIC6);
-Freq 7812 Hz, 0.125uS résolution PWM, largeur d'impulsion 0-128 uS
-Freq 1953 Hz, 0.5uS résolution PWM, largeur d'impulsion 0-512 µs
-Freq 488 Hz, 2US résolution PWM, largeur d'impulsion 0-2048 µs
Pour quartz 16MHz;
-Freq 15625 Hz, 0.0625uS résolution PWM, largeur d'impulsion 0-64 µs
-Freq 3906 Hz, 0.25uS résolution PWM, largeur d'impulsion 0-256 µs
-Freq 977 Hz, 1.0uS résolution PWM, largeur d'impulsion 0-1024 µs
Pour quartz 20MHz;
-Freq 19531 Hz, 0.050uS résolution PWM, largeur d'impulsion de 0 à 51,2 ns (50 nanosecondes résolution)
-Freq 4883 Hz, 0.2uS résolution PWM, largeur d'impulsion de 0 à 204,8 µs
-Freq 1221 Hz, 0.8uS résolution PWM, largeur d'impulsion de 0 à 819,2 µs

Schéma électrique :

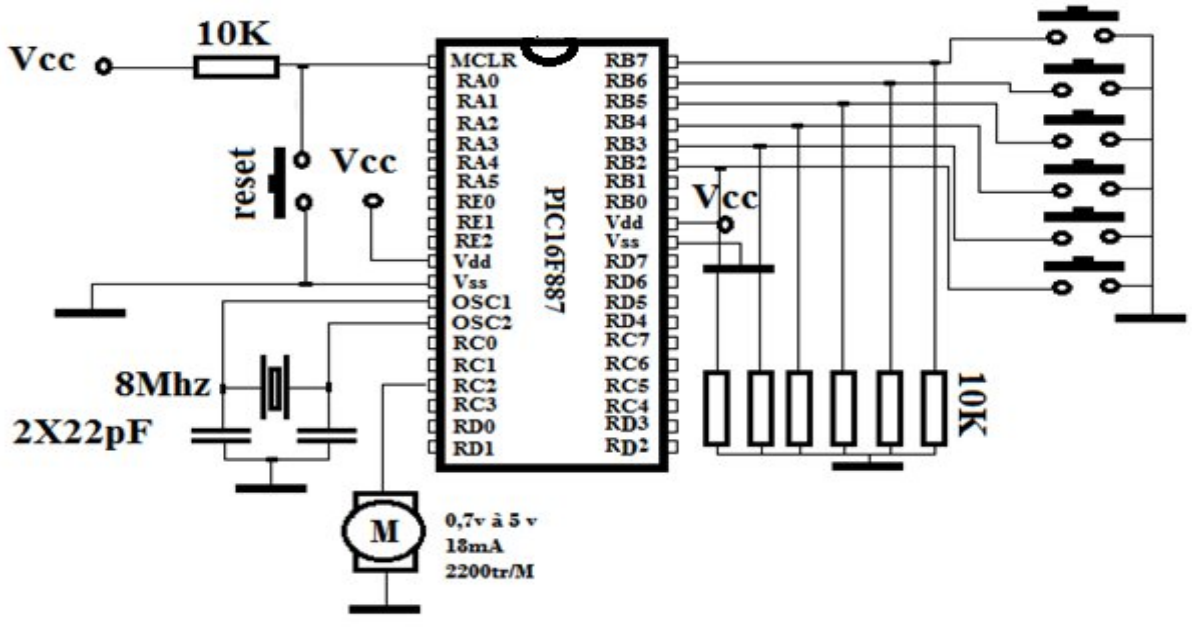
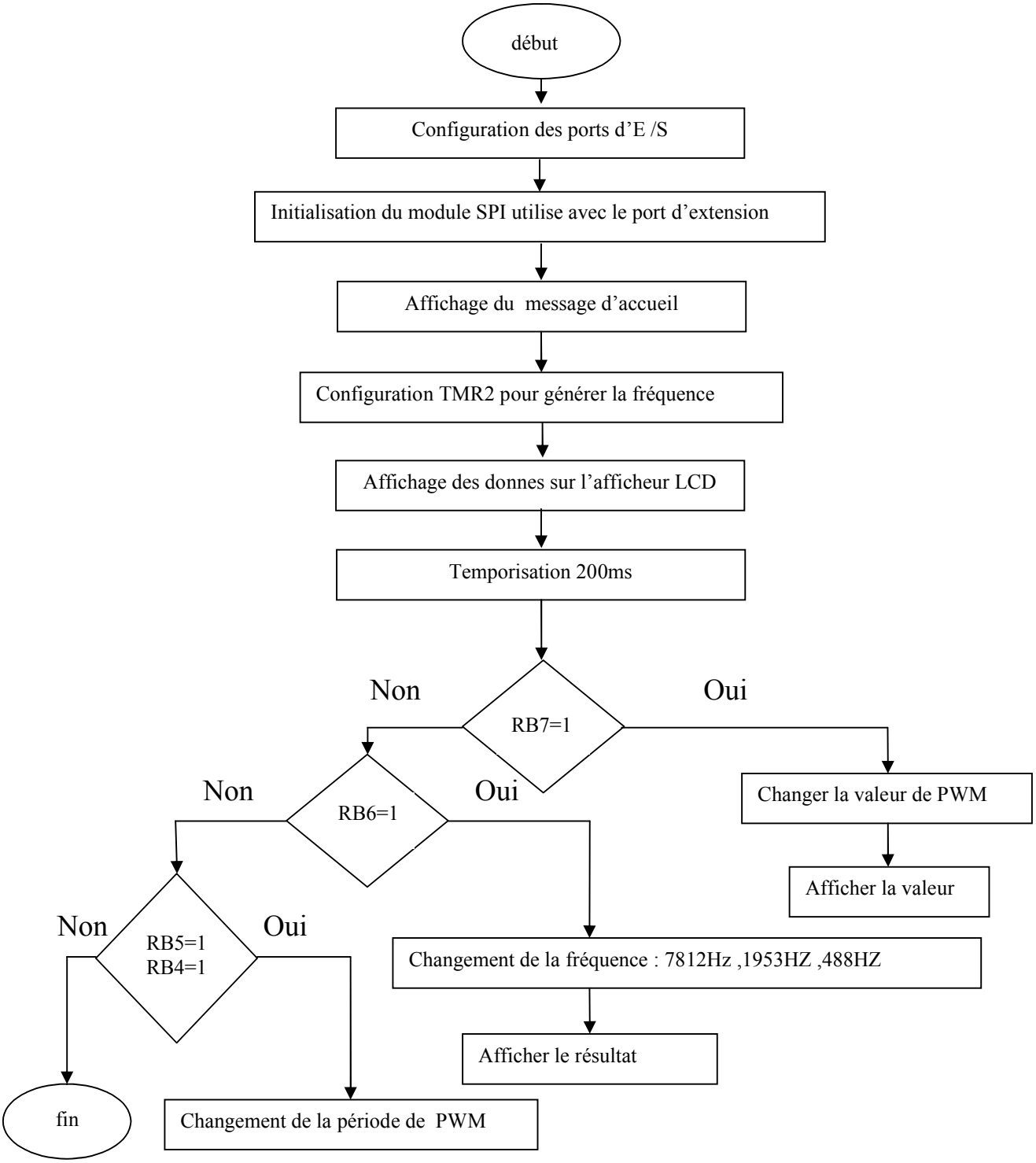


Fig. III-14

Organigramme :



TP15 :

Compteur 7 segment

But du TP :

Le microcontrôleur fonctionne comme deux conteur, la variable « i »s'incrémente et sa valeurs s'affiche sur l'afficheur 7segment (99-0), le principe est de convertir un nombre binaire en décimal puis le devisé en deux pour afficher sa valeur (unité et dizaine).
Dans cet exemple, les fonctions du masque (mask) convertissent un nombre binaire en nombre décimal

Description :

Le programme suivant permet d'incrémenter et d'afficher un nombre sur deux afficheurs 7 segments (de 0 a 99), les deux afficheurs sont a cathode commune, elles sont liés l'un au pin RA0 et l'autre au pin RA1.Un bouton poussoir liée au pin MCLR est programmé comme étant une interruption, l'appui sur ce bouton permet d'incrémenter cette variable.

Schéma électrique :

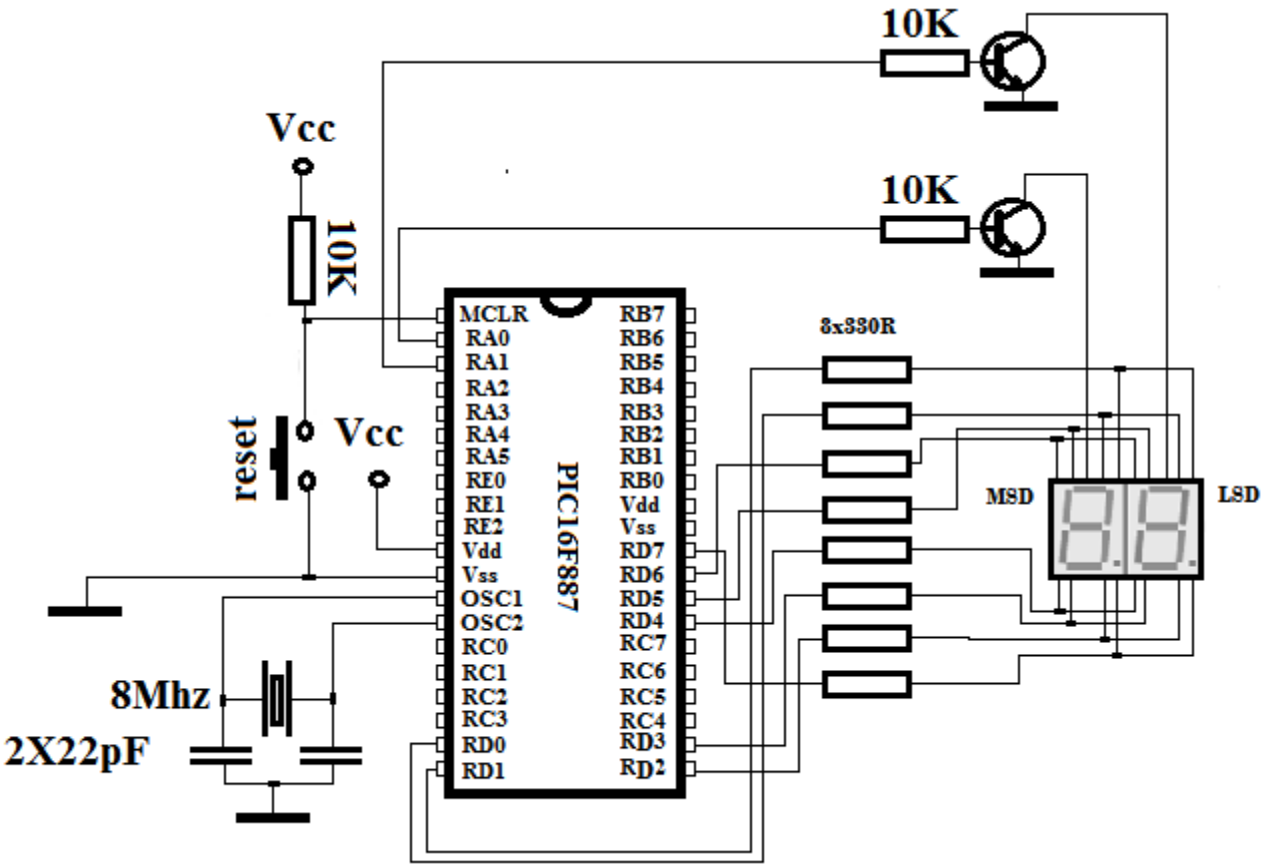
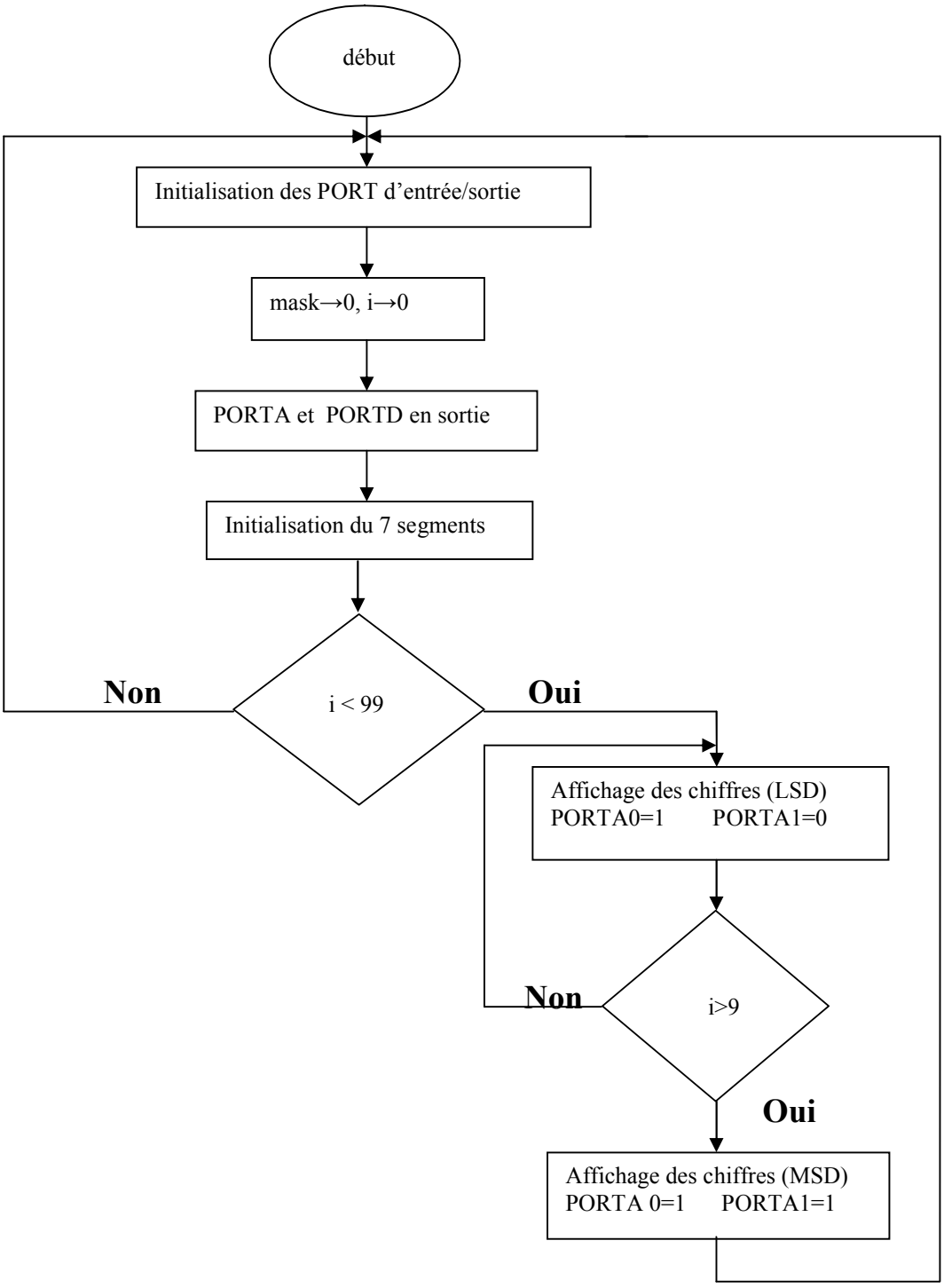
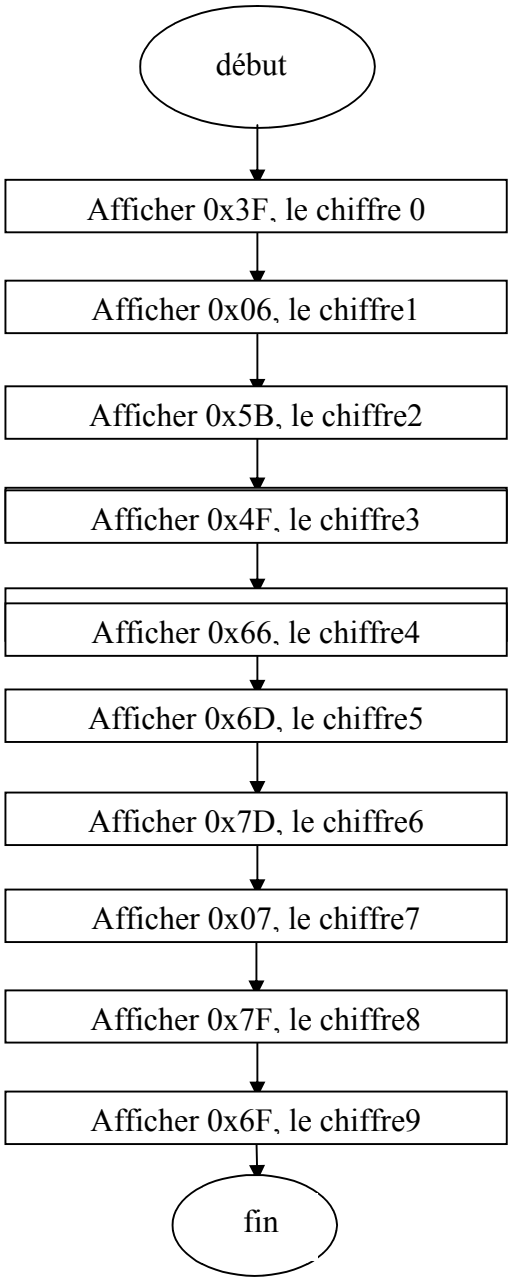


Fig. III-16

Organigramme :


début
Initialisation des PORT d'entrée/sortie
mask→0, i→0
PORTA et PORTD en sortie
Initialisation du 7 segments
i < 99
Non
Oui
Affichage des chiffres (LSD)
PORTA0=1 PORTA1=0
i>9
Non
Oui
Affichage des chiffres (MSD)
PORTA 0=1 PORTA1=1

Organigramme du mask



TP 16 :

Mesure de température en utilisant le capteur DS18X20

But du TP :

Mesure de la température est l'une des opérations les plus courantes effectuées par le microcontrôleur. Un capteur de température DS1820 est utilisé pour la mesure. La plage de température que peut mesurer est entre -55 ° C et 125 ° C avec une précision de 0,5 ° C. Pour transférer des données vers le microcontrôleur, un type spécial de communication série appelée One Wire est utilisé. En raison de l'application simple et large, tels capteurs sont dirigés et contrôlées par les fonctions stockées dans la bibliothèque One_Wire.

Schéma électrique :

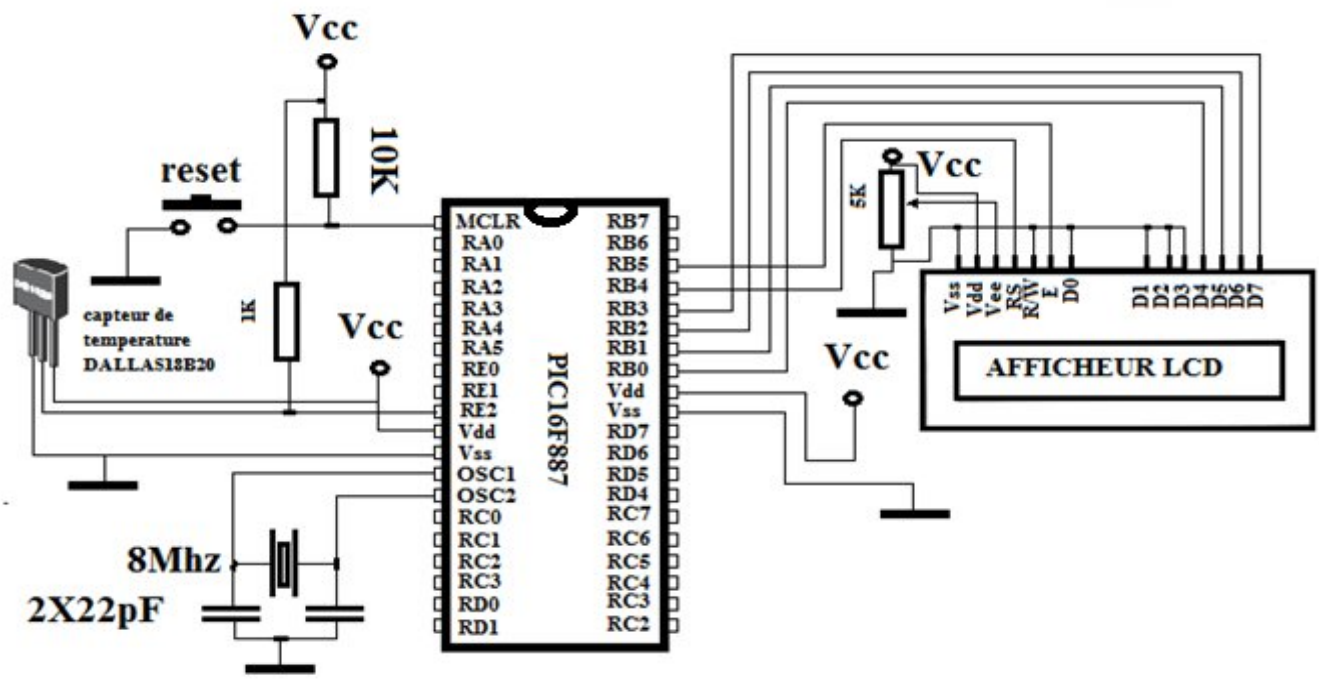


Fig. III-17

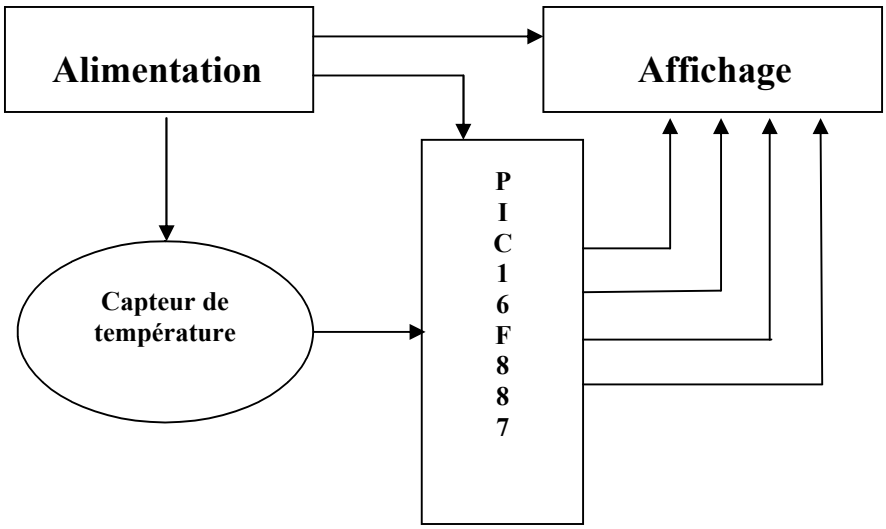
Cette bibliothèque contient au total trois fonctions:

• Ow_Reset est utilisé pour réinitialiser le capteur.
• Ow_Read est utilisé pour recevoir des données du capteur.
• Ow_Write est utilisé pour envoyer des commandes au capteur.

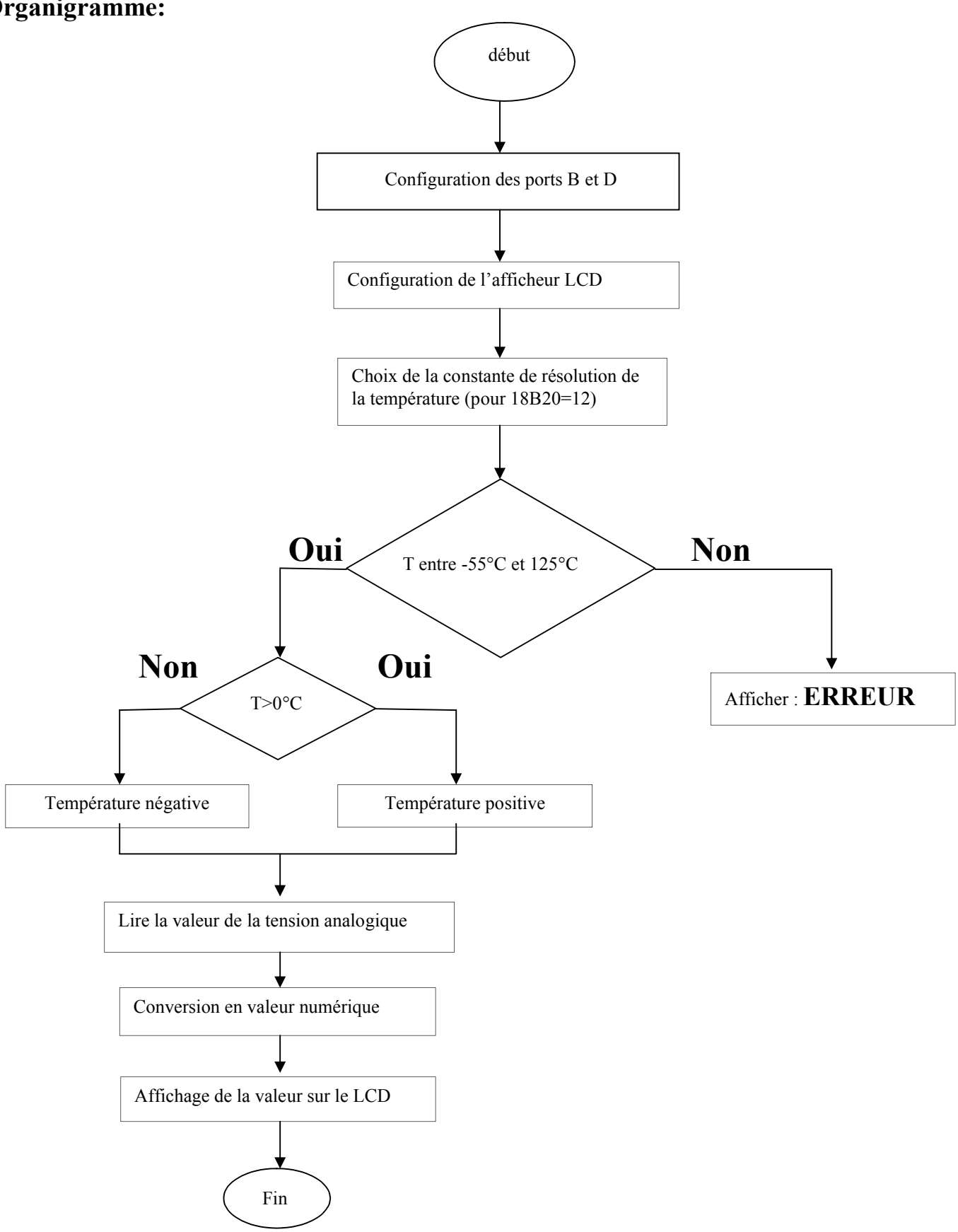
Afin de faire fonctionner ce TP correctement, il est nécessaire de vérifier la présence des bibliothèques suivantes dans la bibliothèque avant de compiler :

- One_Wire
- LCD

Schéma bloc :



Organigramme:



TP17 :

Le voltmètre

But du TP :

Réalisation d'un voltmètre on utilisant la lecture de l'ADC du microcontrôleur sur l'afficheur LCD
Cet exemple illustre l'utilisation du convertisseur A/N sur l'afficheur à cristaux liquides LCD.
Deux messages sont affichés :

La valeur de :
3,141V

Le deuxième message représente la valeur de la tension mesurée (sur RA2).
Quoi qu'il en soit, la température courante ou une autre valeur mesurée peut être affiché au lieu de la tension.

Organigramme :

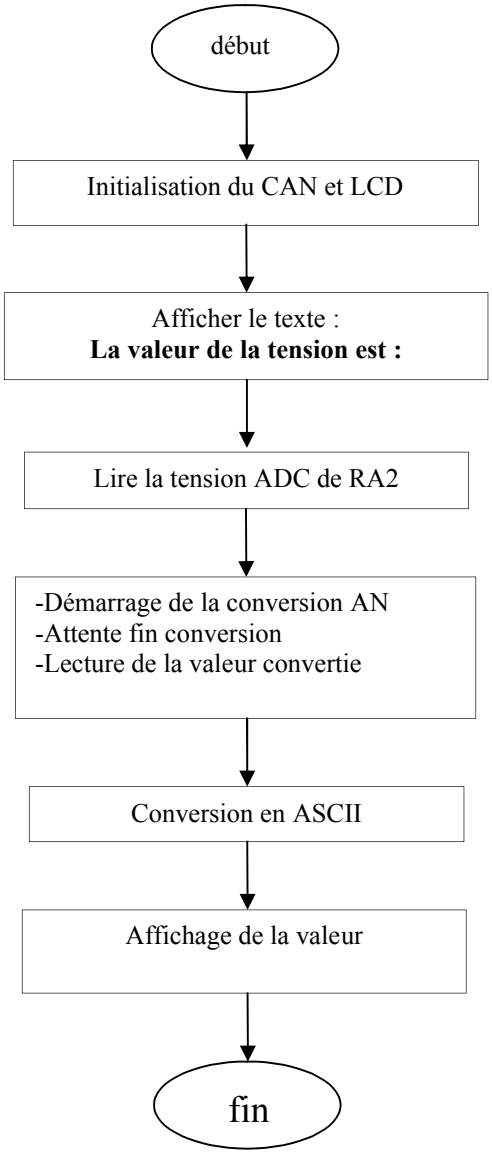


Schéma électrique :

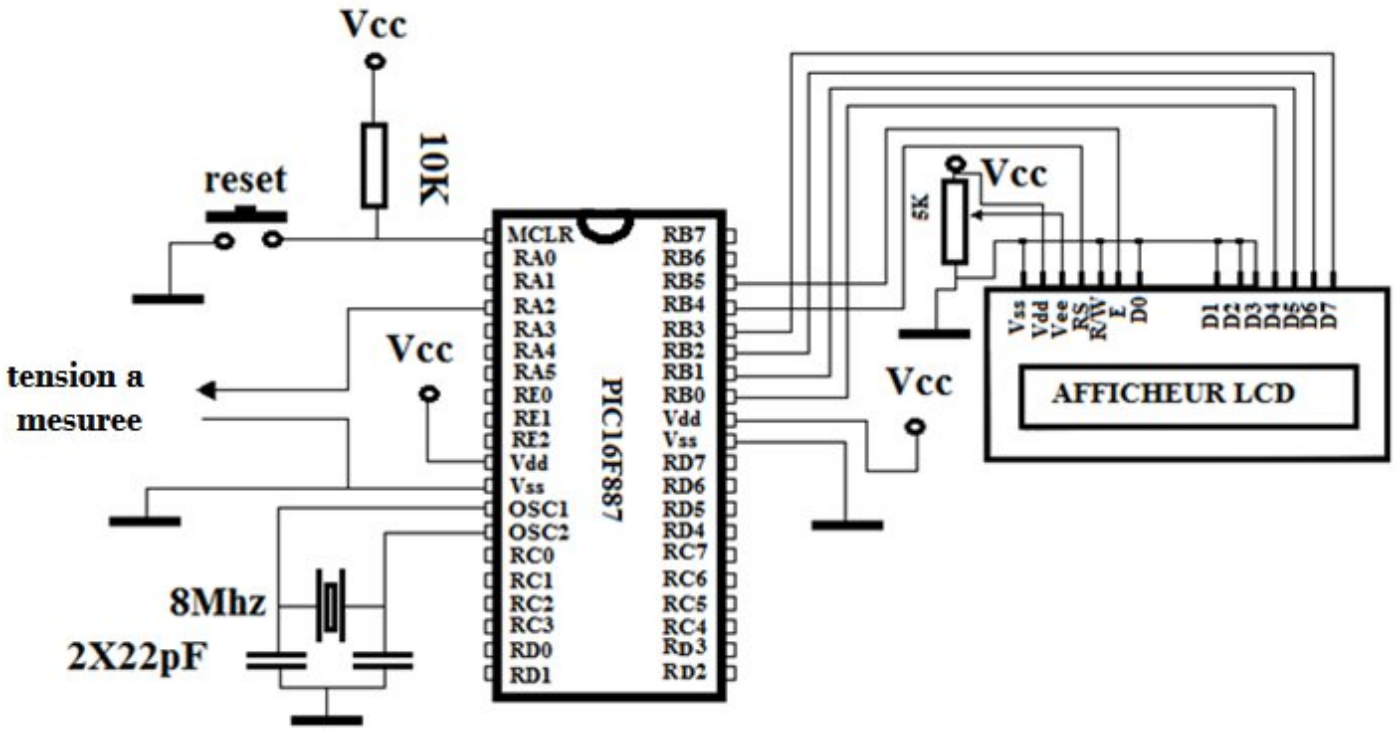


Fig. III-18

Afin de faire fonctionner ce TP correctement, il est nécessaire de vérifier la présence des bibliothèques suivantes dans la bibliothèque avant de compiler :

- ADC
- LCD

TP 18:

Le fréquence mètre

But du TP :

On modifiant l'exemple EasyPIC6 COG de l'affichage de texte, on réalise un fréquence / RMPmètre. La fréquence à mesurer doit être connecté à PORTC broche F0 (T1CKI). La fréquence Maximum est 65000 Hz.

Organigramme :

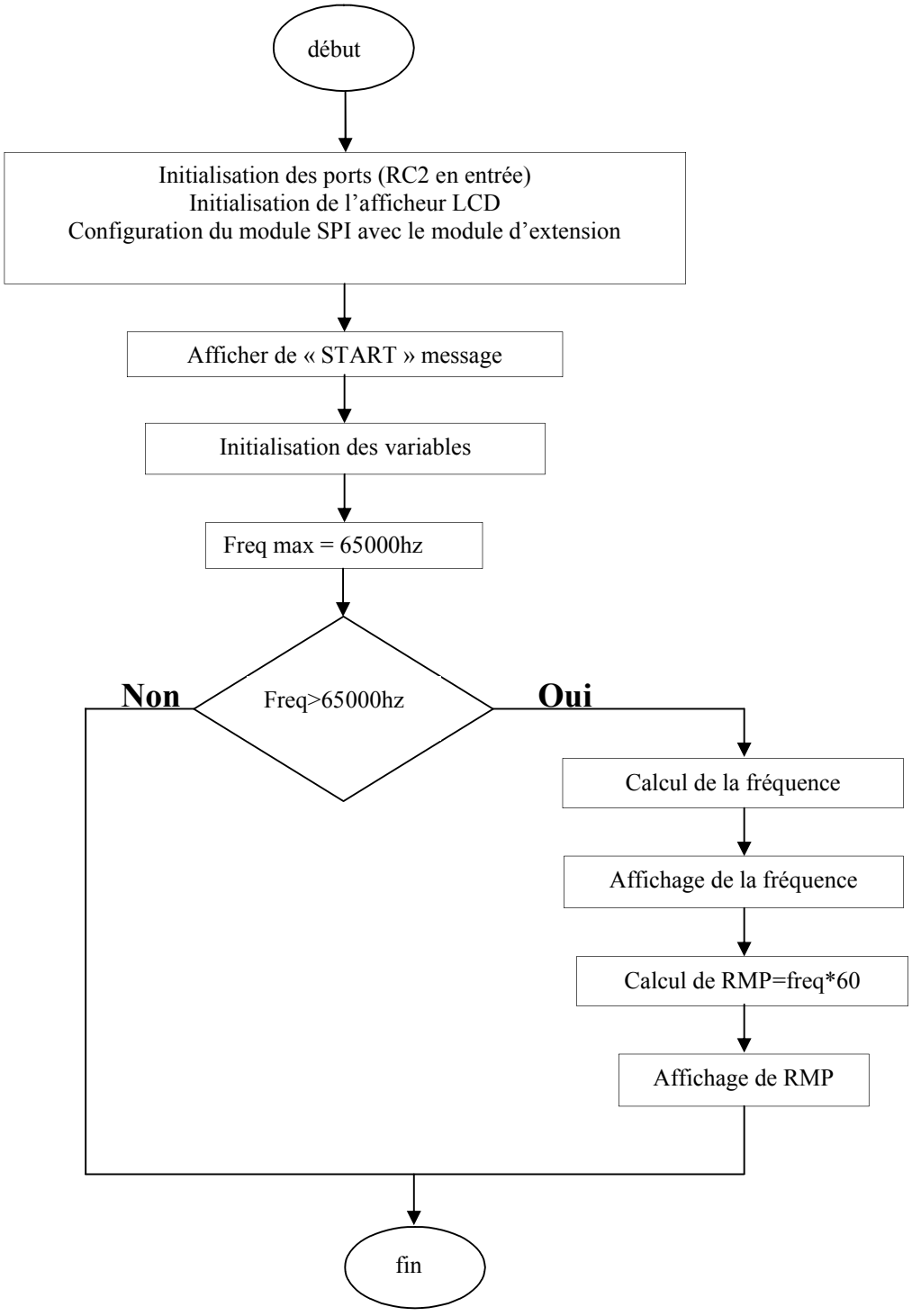


Schéma électrique :

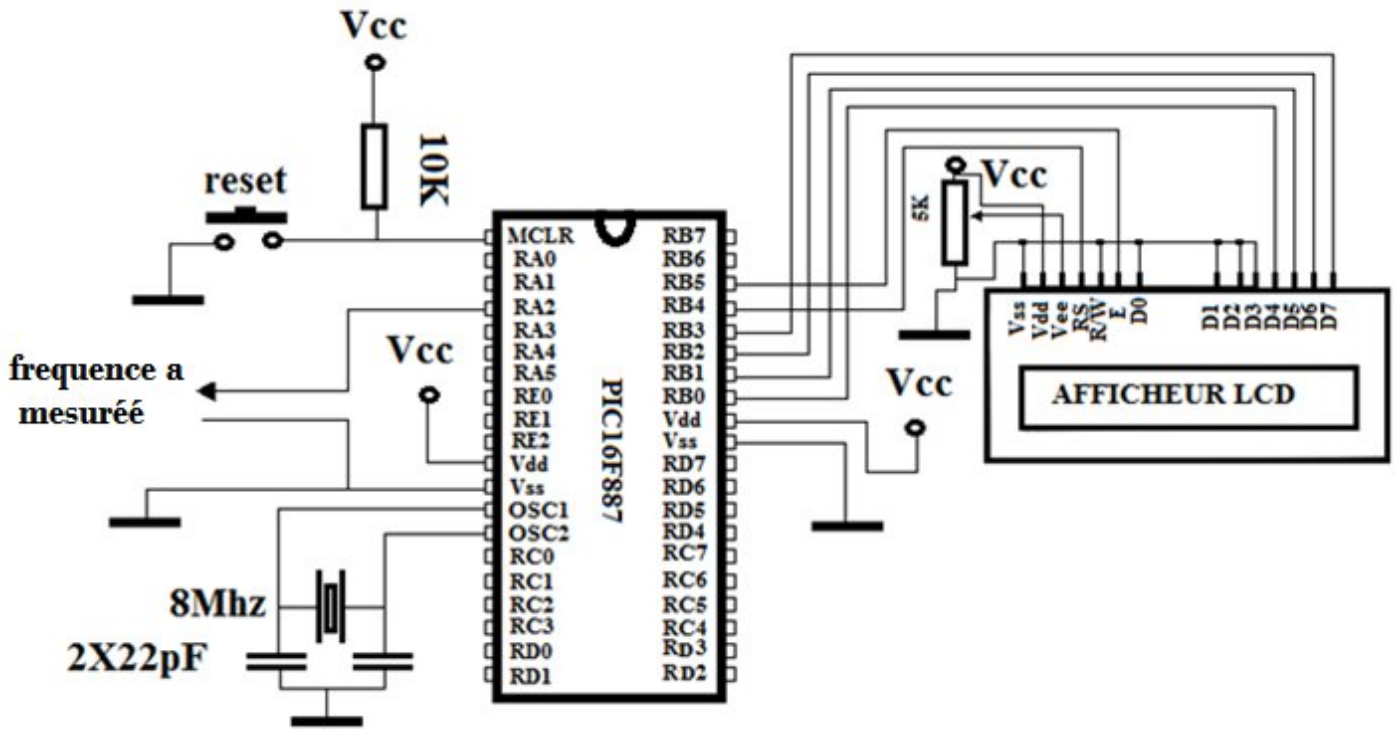


Fig. III-19

Comptage :

La fonction de comptage consiste, comme son nom l'indique, à compter le nombre de fois où l'entrée recevant le signal à mesurer (ici le port RA2), passe de l'état bas à l'état haut (front montant).
Pour cela il existe plusieurs solutions :

- soit on utilise des interruptions déclenchées par les changements d'état sur la borne d'entrée utilisée pour recueillir le signal à mesurer, chacune de celles-ci incrémentant un compteur;
- soit on place le PIC en mode "comptage externe" et le compteur interne est directement incrémenté par les impulsions du signal à mesurer.

Supposons que la base de temps soit de 1000 ms (une seconde), qui correspond donc au temps pendant lequel on laisse aller les impulsions du signal à mesurer vers le compteur. Une fréquence de 8 KHz signifie que le nombre de transitions état bas vers état haut du signal mesuré est de 8000 par seconde (si on compte tous les changements d'état - bas vers haut et haut vers bas, on passe à 16000 transitions par seconde). Le compteur est de type 16 bits, ce qui signifie qu'il est capable de compter de 0 à 65535, et qu'au delà de cette valeur max il repasse à zéro. Si on compte le nombre de transitions bas-haut (fronts montants) que le compteur a ingurgité, on tombe sur la valeur de 8000, il n'a pas eu le temps de faire un "tour complet" (d'aller jusqu'à sa valeur max). En résumé, la valeur lue correspond à la réalité.

Conclusion générale

Conclusion générale

Le travail qui nous a été confié consiste à l'élaboration de maquettes de TP pour le laboratoire microcontrôleur.
Cette étude nous a permis d'apporter un complément indispensable a notre formation en enrichissant celle-ci par des connaissances nouvelles dans le domaine des microcontrôleurs.

Au long de ce projet nous avons pu apprendre le principe de fonctionnement des microcontrôleurs qu'on trouve dans plusieurs domaines et d'exploiter des différents logiciels tel que MIKROC et PICFLASH.

Enfin nous espérons que notre travail puisse servir de moyen pédagogique et didactique aux promotions à venir et qu'il pourrait faire ses TP, et réaliser les différents montages et toucher aux matériels électroniques.

Bibliographie

Sites internet :

1) www.microchip.com
   Documentation du constructeur sur les Pics MICRICHIP
2) http://www.tehnomagazin.com
   Projet et application avec les microcontrôleurs pic
3) http://www.mikroe.com/
   site du constructeur de la carte easypic6
4) www.datasheetcatalog.com
   Répertoire de datasheet des composants
5) www.abcelectronique.com
   Cours de Mr Bigonoff sur les microcontrôleurs Pics