République Algérienne Démocratique et Populaire
Ministère de l'Enseignement Supérieur et de la Recherche Scientifique
Université 8Mai 1945 – Guelma
Faculté des Sciences et de la Technologie
Département d'Electronique et Télécommunications

Mémoire de Fin d'Etude
pour l'obtention du Diplôme de Master Académique

Domaine : **Sciences et Techniques**
Filière : **Electronique**
Spécialité : **Systèmes Electroniques**

Organisation d'un service de prestation par une commande à base d'un PIC

Présenté par :

Benyahia Ali
Maradji Hani

Sous la direction de :

Pr .M.S. Boumaza

JUIN 2013

Remerciement :

Mes remerciements vont tout premièrement a dieu tous puissant, pour la volonté et la santé et la patience qu'il ma donné.

13/2948

Ainsi, je dois remercier infiniment mon encadreur Monsieur boumaza mouhamed sghir pour les précieux conseils, son soutien son suivi continu dans la réalisation de ce mémoire avec la compétence, que chacun de nous luis reconnaissant.

Toute personne qui a été disponible pour me soutenir matériellement ou normalement et profondément remerciée.

Dédicace

Merci Allah (mon dieu) de m'avoir donné la capacité d'écrire et de réfléchir, la force d'y croire, la patience d'aller jusqu'au bout du rêve et le bonheur de lever mes mains vers le ciel et de dire " Ya Kayoum "

Je dédie ce modeste travail à celle qui m'a donné la vie, le symbole de tendresse, qui s'est sacrifiée pour mon bonheur et ma réussite, à ma mère ...

A mon père, école de mon enfance, qui a été mon ombre durant toutes les années des études, et qui a veillé tout au long de ma vie à m'encourager, à me donner l'aide et à me protéger.

Que dieu les gardes et les protège.

A mes adorables sœurs «MERIEM»

A mes frères «DJALEL» et «HAMZA»

A mes amies.

A tous ceux qui me sont chères.

A tous ceux qui m'aiment.

A tous ceux que j'aime.

Je dédie ce travail.

"Ali Ben Yahia"

Dédicace

Ce mémoire est dédie à mon père et ma mère pour l'éducation

qu'ils ont su me donner et qui m'a permis

avec le grâce de Dieu d'arriver la au je suis.

A mes frères et sœurs

A toute la famille

A tous mes amis spécialement pour Ali

Meradji Hani

Sommaire

TABLE DES MATIERES

CHAPITRE 3 Simulation et réalisation pratique

Table de figure

Introduction Générale

Introduction

Les files d'attentes sont un phénomène récurrent dans les sociétés développées. L'afflux de personnes peut être géré de différentes manières. En général, on fait attendre les personnes selon leur ordre d'arrivée sur un chemin défini au préalable et balisé (file d'attente au cinéma). Ce chemin peut être une ligne droite, d'où l'appellation file d'attente. Toutefois ce modèle atteint des limites en termes d'espace. Pour occuper plus rationnellement l'espace, la file d'attente peut suivre un tracé en serpentin (file d'attente aux services des douanes des grands aéroports).

Les files d'attentes peuvent aussi ne pas ordonner les individus dans l'espace, mais instaurer un ordre par le biais de tickets numérotés. Ce peut être le cas dans la salle d'attente d'un service public (bureaux de poste, sécurité sociale).

Généralement, il est mal perçu de dépasser sa place ou d'introduire dans la file quelqu'un n'ayant pas attendu à ses côtés pour lui faire économiser un temps d'attente. En effet, un tel acte (le resquillage) suscite de la part des autres individus dans la file un sentiment d'injustice, rendant l'attente inutile puisqu'il suffirait de prendre la place des autres pour progresser. D'un point de vue économique, la place acquise peut devenir monnayable en proportion de sa valeur inhérente au temps économisé par son achat. Toutefois de telles marchandisations restent exceptionnelles.

Introduction

L'étude de notre projet présente une de plusieurs moyens d'organisation des services il consiste à réalisé un système de communication de la gestion des services dans une salle d'attente (bureaux de poste, sécurité sociale) équipée d'un distributeur de tickets à base d'un microcontrôleur.

Pour cella nous avons utilisé deux principaux composants :

- Le microcontrôleur pic16f877 : le composant de base qui gère l'Afficheur
- l'Afficheur LCD : pour afficher le numéro de client et le guichet libre.

Notre manuscrit est organisé de la manière suivante :

- Introduction générale
- Définition de cahier de charge
- Généralités sur le matériel utilisé
- Simulation et réalisation pratique
- Conclusion générale

Chapitre 1

DEFINITIOT DE CAHIER DE CHARGE

Introduction

En dépit de notre vie moderne, le secteur services demeure un problème de manque d'organisation qu'elle a vécu jusqu'à nos jours la difficulté de s'adapté et d'enlevé la différence entre les citoyens.

Les citoyens présentent une source principale de critique, ils imposent des remarques et des interrogations sur tout tant qu'ils ne sont pas satisfaits des services.

Dans le cadre de l'insatisfaction du client ou du citoyen les responsables des services cherchent à minimiser ou à éviter le maximum possible les problèmes de ce genre existants entre les services et les citoyens. Pour cela, Ils auront recours au domaine électrique qui a vue des progrès technologiques très importants pour trouver une solution.

Cahier de charge :

Il s'agit de concevoir et de réaliser un système électronique (à base d'un microcontrôleur) qui permet de faire la gestion de service dans une salle d'attente équipée d'un distributeur automatique de tickets.

La salle dispose de trois guichets donneurs de services. Chaque guichet dispose d'un bouton d'appel de client, et dans la salle il ya et un afficheur LCD qui affiche le numéro du client et le numéro du guichet.

Si l'administrateur du guichet appuie sur le bouton d'appel du client, cette impulsion incrémente le numéro affiché. Ainsi le numéro du client suivant s'affiche sur l'afficheur LCD accompagné par l'affichage du numéro de poste de l'opérateur libre (celui qui actionnait le bouton poussoir).

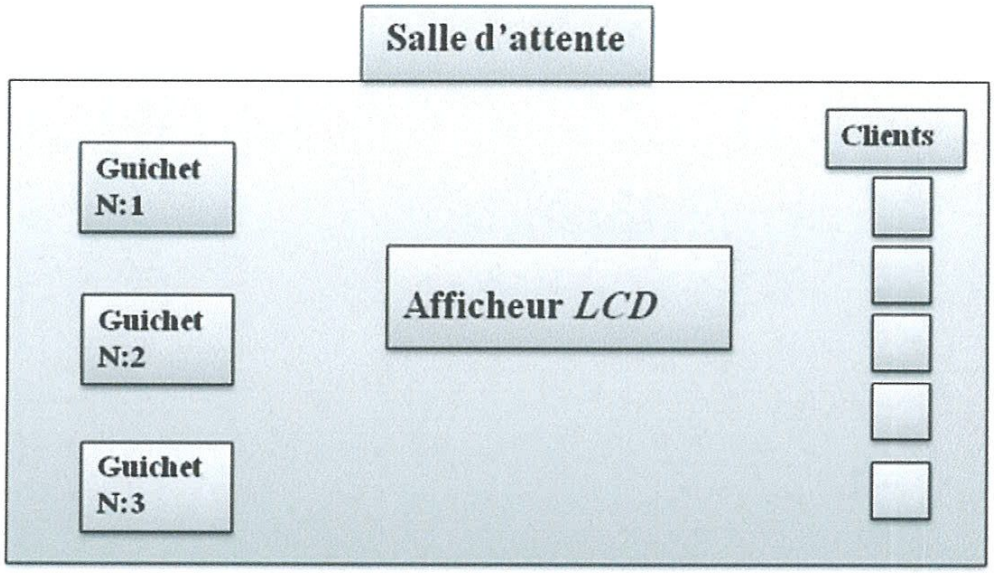


Figure (1) la salle d'attente

1- Définition File d'attente

- Une file d'attente, ou une queue, est un regroupement d'individus attendant de manière organisée quelque chose. Les files d'attentes résultent d'une demande supérieure à la capacité d'écoulement d'une offre (un bien ou un service) **[1]**. En principe, elles n'influent pas sur le coût de cette offre. Sur les routes, les files d'attentes sont appelées des bouchons **[2]**. Cette notion fait l'objet d'une branche du calcul des probabilités, la Théorie des files d'attentes, utilisée aussi bien en logistique qu'en informatique.
- Citons à titre d'information les différents points traités dans ces articles: Structure et organisation (Principe de fonctionnement), Types de files d'attentes (Limite de fonctionnement[3], Modélisation mathématique, histoire, notes et références).

Figure (2) une file d'attente

2- L'objectif de Gestion des Files d'Attentes

Le premier objectif de tous les systèmes de gestion de files d'attentes est d'offrir un service de meilleure qualité aux clients [3]. Dans le domaine public comme dans le domaine privé, gérer le trafic des personnes est primordial pour le bon fonctionnement de la vie quotidienne de votre entreprise ou administration. Savoir optimiser le temps (pour la société et l'utilisateur) consiste souvent à améliorer la qualité de service.

3- Présentation de notre projet

Notre projet est un système de commande de la gestion d'une salle d'attente équipée d'un distributeur de tickets.

Il s'agit de réaliser un prototype de se système à base d'un microcontrôleur pic16f877. Cette salle dispose de trois guichets, chacun d'eux est équipé d'un bouton poussoir pour

Incrémenter le numéro de ticket et affiche le numéro de guichet libre (celui qui appuie sur le bouton poussoir) et un afficheur LCD.

3.1- schéma synoptique :

Le schéma synoptique du projet :

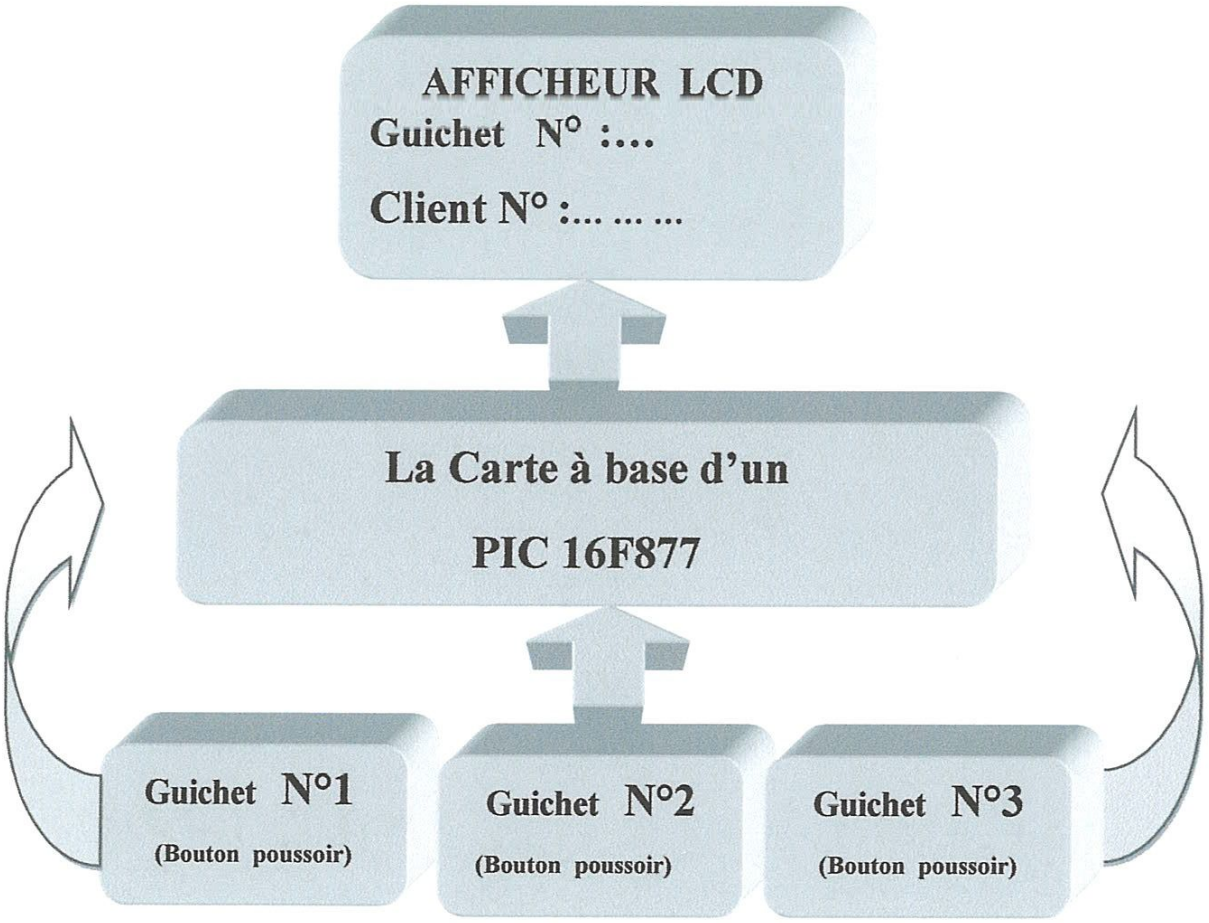


Figure (3) schéma synoptique

4- les systèmes existants :

4.1- file d'attente avec un afficheur LED :

Est un système qui gère les files d'attente avec un Afficheur électronique à LED pour l'indication du n° appelé [4]. Esthétique moderne, lisible jusqu'à 60 m sur 150°. Dispose d'un signal sonore (sonal) et d'une fonction n° clignotant. Plusieurs afficheurs peuvent-être reliés les uns aux autres pour indiquer un même numéro sur de grands espaces. Se pilote soit à l'aide d'un pupitre d'appel radio, soit d'un pupitre filaire multifonctions avec clavier numérique.il est disponible avec deux versions 2 chiffres et 3 chiffres.

Figure (4) afficheur LED avec 2 chiffres

Commentaire : Mais ce système fonctionner avec un seul guichet.

4.2- commande de la gestion dans un service d'attente équipé par un distributeur de ticket à l'aide d'un PC: Il ya un autre système multiservices et statistique fabriqué dans les années passées, c'est une commande de gestion dans un service d'attente équipé par un distributeur de ticket à l'aide d'un PC [5].

- Ce système présente un afficheur principal qui indique par quatre chiffres rouges le rang du client en attente et par deux chiffres verts le numéro du guichet vers lequel le visiteur doit se présenter. Le système peut gérer jusqu'à huit guichets.

Devant chaque guichet, un afficheur répétiteur peut rappeler le numéro du client appelé. Une borne distributrice de tickets attribue un numéro à chaque client et le dirige vers le Service souhaité. Le ticket émis peut indiquer le nombre de clients en attente, le nom du service demandé, le logo de l'organisation.

Une console maîtresse permet de configurer le système.

Chaque guichet possède, une console permettant entre autre d'appeler et de rediriger le client. Un écran indique le numéro du ticket.

Le système se complète par un logiciel de configuration et de gestion statistique. Ce logiciel s'installe sur PC équipé de Windows version 98 et supérieur.

Il résulte de ce système unique un gain de temps évident et une gestion optimale des flux de visiteurs.

Commentaire : Mais ce système très compliqué et très cher

4.3- la gestion de service dans une salle d'attente équipée d'un distributeur automatique de tickets:

Est un projet de fin de étude de l'année 2009 le projet et un system Il s'agit de réaliser un prototype de se système à base d'un microcontrôleur et qui dispose de trois guichets chacun équipé de deux boutons poussoirs un pour incrémenter le numéro de ticket et un autre pour la fermeture.

Un tableau d'affichage au centre de la salle indique le numéro de ticket et le guichet qui lui correspond [6] .

Commentaire : Mais ce système est réalisé par plusieurs composants (4 microcontrôleurs et 8 afficheurs 7 segments) avec un numéro de client max de 99.

4.4- Systèmes ALPROS : (Algerian Production Systems)

Il existe deux systèmes de gestion des files d'attentes qui disposent d'une large gamme de produits permettant d'offrir à ses clients des solutions complètes et adaptées fabriqués par ALPROS (Algerian Production Systems) .

4.4.1- Le K211

Le K211 est un système de gestion de file d'attente simple [7], facile à installer et à utiliser, il offre plus à son esthétique une bonne lisibilité dans les espaces d accueil.

Le K211 conçu pour les petits est moyens espaces d'accueil, il gère une seule file d'attente de un, deux ou trois guichets, proposant les mêmes services sans le repérage du guichet appelant.

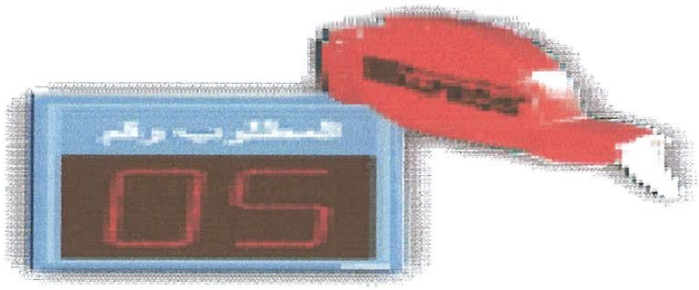


Figure (5) le K211

Etudié, conçu et réalisé dans le but de subvenir à la demande des systèmes intelligents de gestion électronique des files d'attentes.

4.4.2-Le K418

- Le K418 est la meilleur solution existante dans le domaine [8], capable de gérer jusqu'à quinze files d'attentes séparées, chacune son ordre, où chaque guichet offre un service différent, et il peut gérer tout les guichets dans un même ordre, ou même grouper des guichets en plusieurs groupes, où chaque groupe a son ordre et son distributeur de ticket séparé.

Composé d'un panneau d'affichage muni de quatre digits, dont trois pour les numéros d'appel et un pour désigner le guichet appelant et un autre panneau d'un seul chiffre au-dessus pour indiquer le rang de la file d'attente.

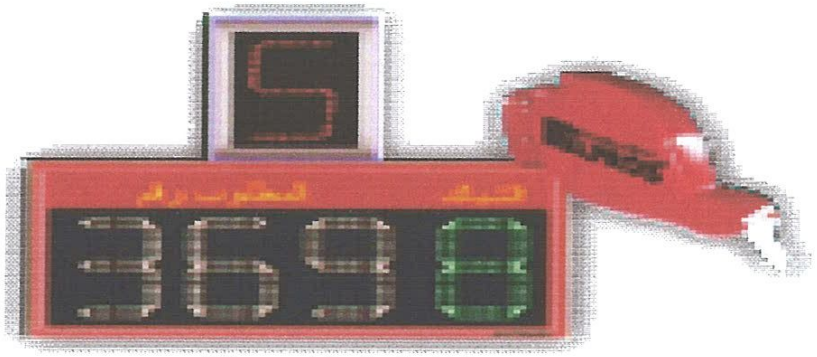


Figure (6) le k 418

Commentaires : les systèmes proposés par ALPROS présentent l'avantage qu'ils soient fabriqués en Algérie, mais comme inconvenants le k211 est conçu pour une file d'attente qui est desservie par plusieurs guichets mais il n'indique pas le numéro du guichet libre. Le deuxième système le k418 gère plusieurs files d'attentes qui sont desservies par plusieurs guichets. Ce type de gestion est destiné à une clientèle instruite et non pas à une clientèle populaire.

5-Le choix d'un microcontrôleur pic 16f877 :

- Le choix du microcontrôleur est primordial car c'est de lui que dépend en grande partie les performances, la taille de la carte mère, la facilité d'utilisation et le prix du montage.

-Le PIC 16F877 possède en plus des instructions très puissantes [1] donc un programme à développer réduit, surtout lorsqu'on utilise le logiciel de programmation [2] en un

Langage qui possède un nombre important de procédures et fonctions prédéfinies, dédié au PIC 16F877. En fait la cause principale du choix de ce type de microcontrôleur est qu'il dispose de l'option du convertisseur A/D pour satisfaire le coté acquisition, aussi la possibilité de l'adaptation au protocole I2C et la liaison RS232 mais aussi le nombre de ports d'entrées /sorties est convenable [3].

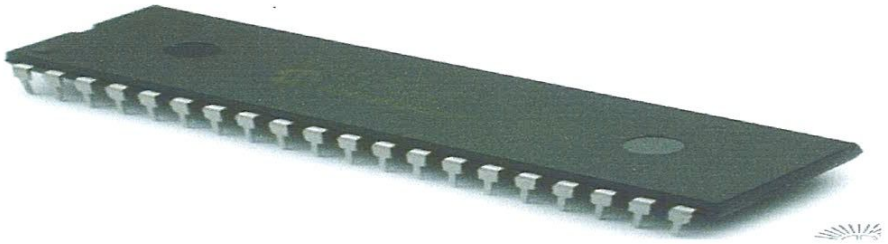


Figure (7) le pic 16f877 [10]

6- les avantages du microcontrôleur [9]

L'utilisation des microcontrôleurs pour les circuits programmables a plusieurs points forts. Il suffit pour s'en persuader, d'examiner la spectaculaire évolution de l'offre des fabricants de circuits intégrés en ce domaine depuis quelques années.

Nous allons voir qu'un nombre d'entre eux proposent.

- ✓ Les performances sont identiques voir supérieurs à ses concurrents
- ✓ Les prix sont les plus bas du marché
- ✓ Très utilisés donc disponibles
- ✓ Les outils de développement sont gratuits et téléchargeables sur le WEB
- ✓ Le jeu d'instruction réduit est souple, puissant et facile à maîtriser
- ✓ Les versions avec mémoire flash présentent une souplesse d'utilisation et des Avantages pratiques indéniables
- ✓ La communauté des utilisateurs des PICs est très présente sur le WEB. On trouve sur le net quasiment tout ce dont on a besoin, tutoriaux pour démarrer, documents plus approfondis, schémas de programmeurs avec les logiciels qui vont avec, librairies de routines, forums de discussion.

Conclusion :

Dans ce chapitre, nous avons spécifié le cahier de charge et le mode de fonctionnement de notre système et ses caractéristiques. Nous avons donné quelques exemples de systèmes fabriqués auparavant. Comme nous présentons les arguments appuyant notre Choix de microcontrôleur pic 16f877et les avantages qui en découlent.

Chapitre 2

ETUDE ET CHOIX DU MATERIEL

Introduction

Dans ce chapitre nous présentons une description détaillée de la solution adoptée pour répondre aux spécifications de notre cahier de charge et en abordant la conception détaillée de chaque partie du système afin d'obtenir une schématisation complète et précise.

1-Le microcontrôleur :

1.1-Généralités sur le microcontrôleur

Le PIC n'est rien d'autre qu'un microcontrôleur, c'est à dire une unité de traitement de l'information de type microprocesseur à laquelle est ajouté des périphériques internes permettant de réaliser des montages sans nécessiter d'ajouter des composants externes [10].

- Les Pics sont des composants dits **RISC** (**R**educe **I**nstruction **C**onstruction **S**et), ou encore composants à jeu d'instructions réduit. Pourquoi ? Et bien, sachez que plus on réduit le nombre d'instructions, plus facile et plus rapide en est le décodage, et plus vite le composant fonctionne.

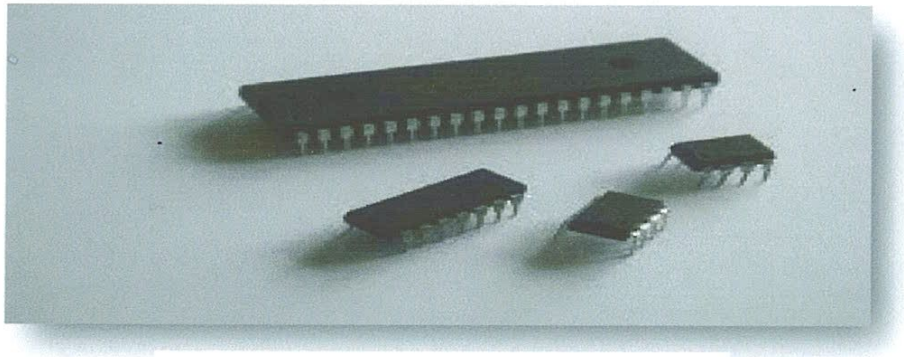


Figure (8) le microcontrôleur

1.2-Caractéristiques principales d'un microcontrôleur [10] :

- ✓ De nombreux périphériques d'E/S.
- ✓ Une mémoire de programme.
- ✓ Une mémoire vive (en général de type SRAM).

- ✓ Éventuellement une mémoire EEPROM destinée à la sauvegarde par programme des données à la coupure de l'alimentation.
- ✓ Un processeur 8 ou 16 bits.
- ✓ Faible consommation électrique.

Les Pics sont subdivisés à l'heure actuelle en 3 grandes familles :

- ➢ La famille Base Line, qui utilise des mots d'instructions de 12 bits.
- ➢ La famille Mid-Range, qui utilise des mots de 14 bits (et dont font partie la 16F84 ,16F876 et 16F877).
- ➢ La famille High-End, qui utilise des mots de 16 bits.

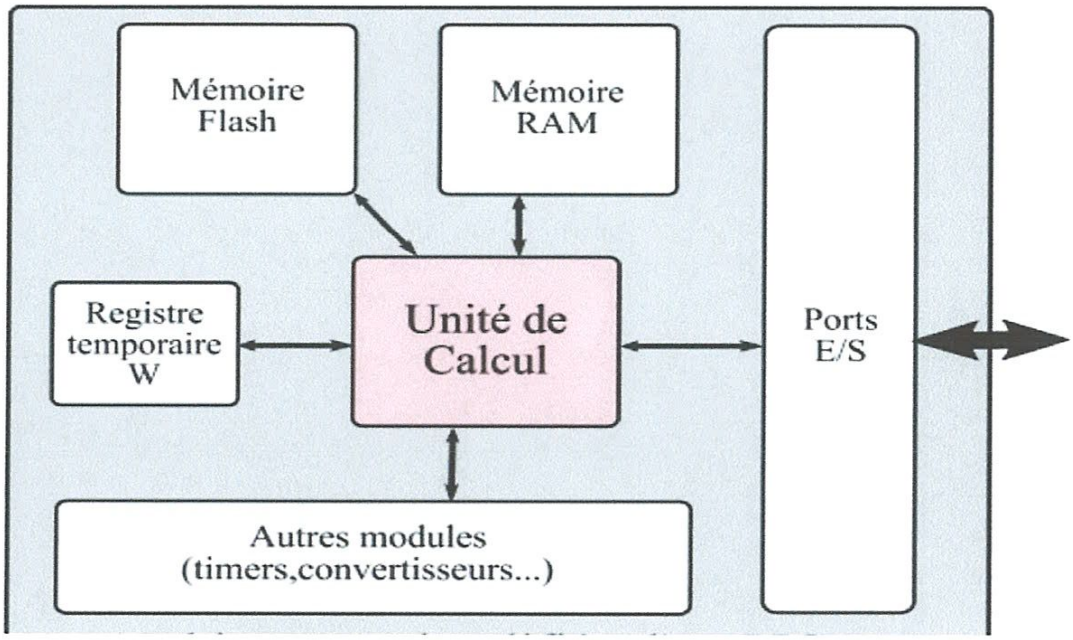


Figure (9) architecture simplifiée de microcontrôleur

1.3- Identification des Pics:

Pour identifier un PIC, on utilise simplement son numéro **[10]** :

16 : indique la catégorie du PIC, c'est un Mid-range.

L : indique qu'il fonctionne avec une plage de tension beaucoup plus tolérante.

C : indique que la mémoire programme est une EPROM ou une EEPROM.

CR ou F : indique le type de mémoire : CR (ROM) ou F (FLASH).

XX : représente la fréquence d'horloge maximale que le PIC peut recevoir.

2 -Le PIC 16F877:

Nous allons maintenant s'intéresser à la structure interne **[9]** du PIC 16F877, avec lequel nous travaillerons.

Le 16F877 est un microcontrôleur de MICROCHIP, fait partie intégrante de la famille des Mid-Range (16) dont la mémoire programme est de type flash (F).

Figure (10) pic16F877

2.1- Principales caractéristiques du PIC 16F877:

Le PIC 16F877 est caractérisé par **[9]** :

- ✓ Une fréquence de fonctionnement élevée, jusqu'à 20 MHz.
- ✓ Une mémoire vive de 368 octets.
- ✓ Une mémoire EEPROM pour sauver des paramètres de 256 octets.
- ✓ Une mémoire morte de type FLASH de 8 kmots (1mot = 14 bits), elle est réinscriptible à volonté.
- ✓ 33 Entrées et sorties.
- ✓ 3 Temporisateurs : TIMER0 (8 bits avec pré diviseur), TIMER1 (16 bits avec

  Pré diviseur avec possibilité d'utiliser une horloge externe réseau RC ou QUARTZ) et TIMER2 (8bits avec pré diviseur et post diviseur).
- ✓ Une tension d'alimentation entre 2 et 5,5 V.

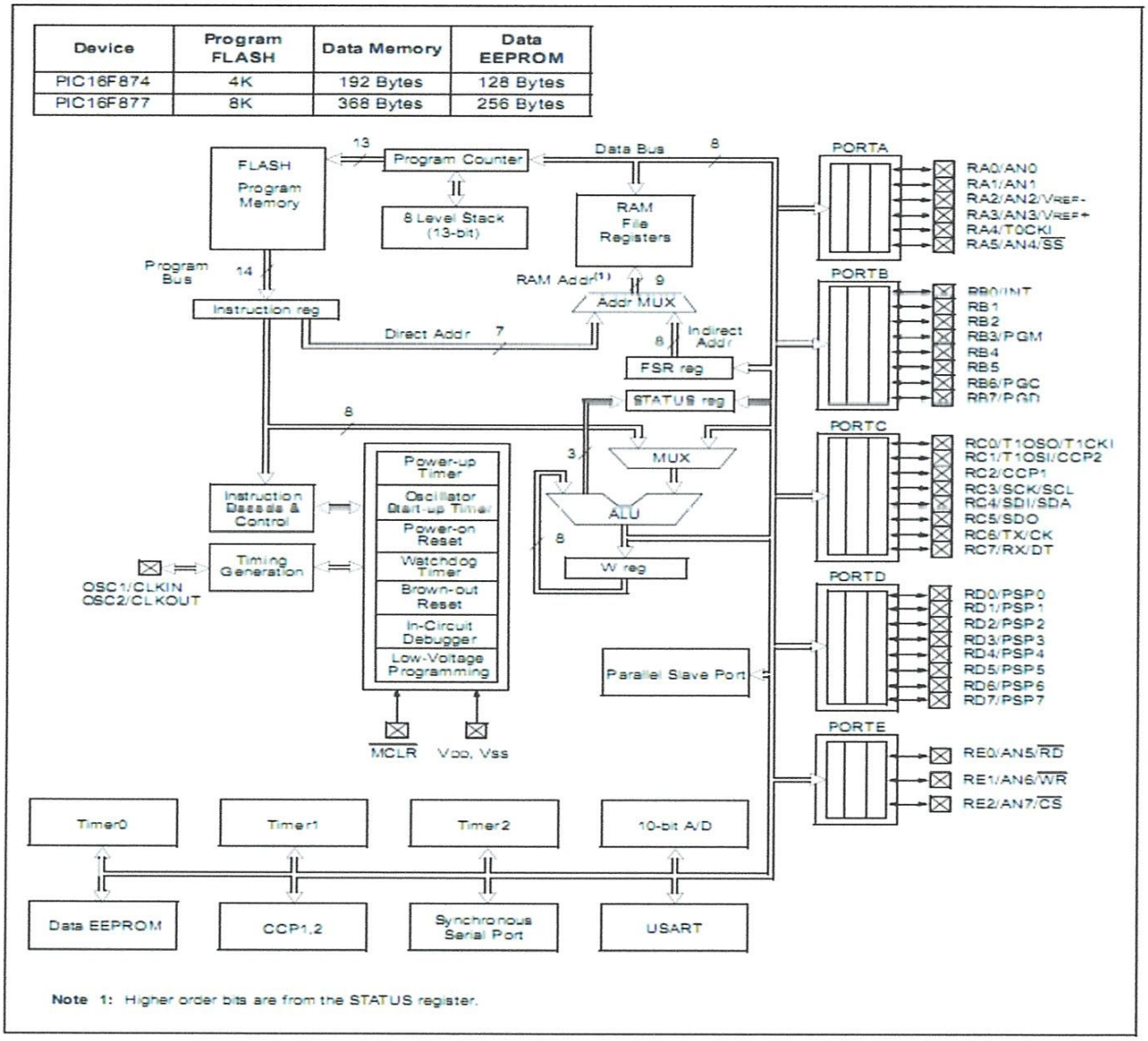


Figure (11) structure interne du PIC

3- Les Mémoires du PIC 16F877:

Le PIC 16F877 dispose de trois types de mémoires **[9]** :

3.1 Mémoire vive RAM

C'est de la mémoire d'accès rapide, mais labile (c'est-à-dire qu'elle s'efface lorsqu'elle n'est plus sous tension); cette mémoire contienne les registres de configuration du PIC ainsi que les différents registres de données.

Elle comporte également les variables utilisées par le programme.

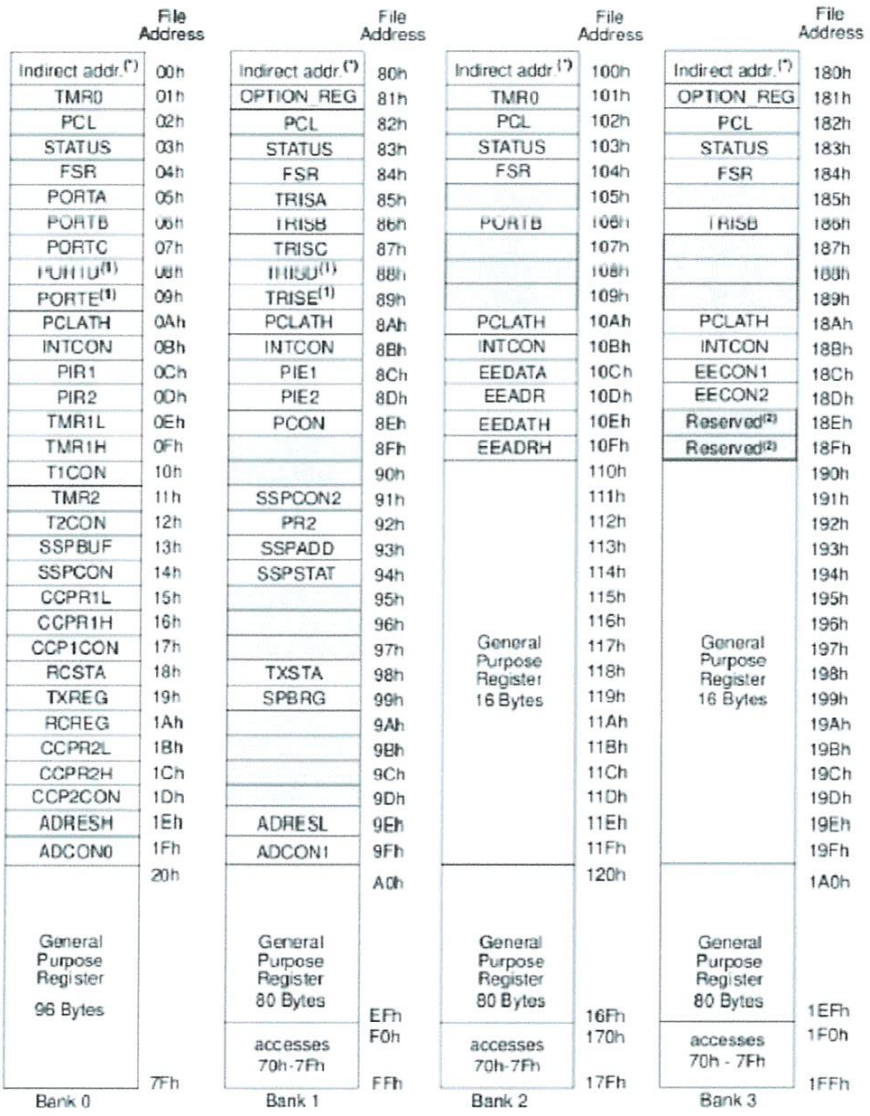


Figure (12) Mémoire RAM

3.2-Mémoire morte FLASH

C'est la mémoire programme proprement dite. Chaque case mémoire unitaire se compose de 14 bits. La mémoire FLASH est un type de mémoire stable, réinscriptible à volonté. Cette mémoire a fait le succès de microprocesseur PIC. Dans le cas du16F877, cette mémoire FLASH est de 8 Kmots.

Lorsque l'on programme en assembleur, on écrit le programme directement dans cette mémoire.

3.3 -Mémoire EEPROM

Elle est de 256 octets, électriquement effaçable, réinscriptible et stable.

Ce type de mémoire est d'accès plus lent, elle est utilisée pour sauver des paramètres.

L'adresse relative de l'accès EEPROM est comprise entre 0000 et 00ff, ce qui nous permet d'utiliser qu'un registre de huit bits pour définir cette adresse.

4-L' 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 20 MHz selon le type de microcontrôleur **[10]** .

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 dit aussi cycle d'instruction est obtenue en divisant la fréquence par 4. Le terme Fosc/4 désigne l'horloge système. Par exemple on obtient un cycle d'instruction de 1 MHz, en utilisant un quartz de 4 MHz.

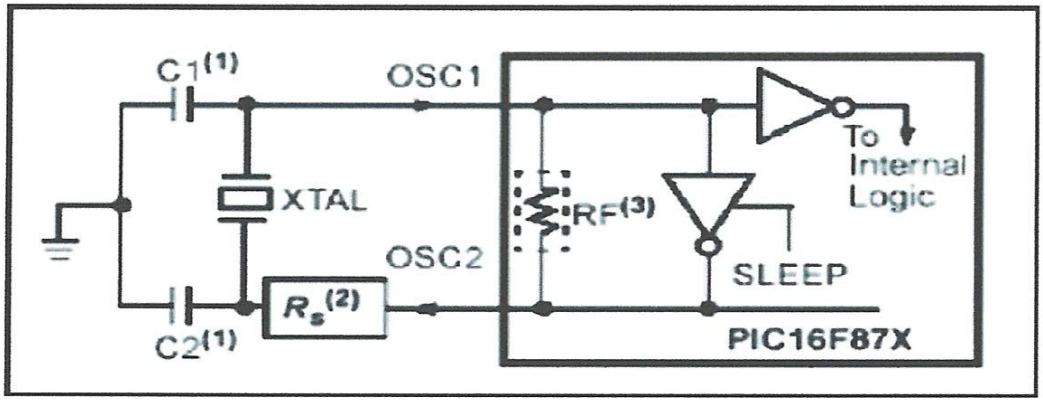


Figure (13) Schéma de l'horloge

5-Les Timers

Le PIC 16F877 dispose de 3 timers

5.1-Timer 0 (8 bits): Il peut être incrémenté par des impulsions extérieures **[10]** via la broche (TOCKI/RA4) ou par l'horloge interne (Fosc/4).

5.2-Timer 1 (16 bits): Il peut être incrémenté soit par l'horloge interne, ou soit par des impulsions sur la broche T1CKI/RC0 ou par un oscillateur (RC ou quartz) connecté sur les broches TOSO/RCO et T1OSI/RC1.

5.3-Timer 2 (8 bits) : Il est incrémenté par l'horloge interne, celle peut être prédivisée. Tous ces timers peuvent déclencher une interruption interne, s'ils ont été autorisés

6-Pile et Compteur Programme:

Le « Program Counter » ou PC est le compteur qui pointe, dans la mémoire de programme, la prochaine instruction à exécuter **[10]**. Il est lié à la pile système (Stack en anglais) à 8 niveaux dans le 16F877. C'est-à-dire qu'on peut avoir jusqu'à 8 niveaux d'imbrication d'appels de sous-programmes ou fonctions.

7-Chien de garde (Watch Dog):

- C'est un système de protection contre un blocage du programme **[10]**. Par exemple, si le programme attend le résultat d'un système extérieur (conversion analogique numérique par exemple) et qu'il n'y a pas de réponse, il peut rester bloquer. Pour en sortir on utilise un chien de garde. Il s'agit d'un compteur qui, lorsqu'il arrive en fin de comptage, permet de redémarrer le programme. Il est lancé au début du programme.

- En fonctionnement normal, il est remis à zéro régulièrement dans une branche du programme qui s'exécute régulièrement. Si le programme est bloqué, il ne passera plus dans la branche de remise à zéro et le comptage va jusqu'au bout, déclenche le chien de garde qui relance le programme.

8- Organisation externe du PIC 16F877

Le boîtier du PIC 16F877 comprend 40 pins : 33 pins d'entrées/sorties 4 pins pour l'alimentation, 2 pins pour l'oscillateur et un pin pour le reset (MCLR) **[9].**

La broche MCLR sert à initialiser le µC en cas de la mise sous tension, de remise à zéro externe, de chien de garde et en cas de la baisse de tension d'alimentation. Les broches VDD (broches 11 et 32) et VSS (broches 12 et 31) servent à alimenter le PIC.

Les courants véhiculés dans le PIC sont loin d'être négligeables du fait de nombreuses entrées/sorties disponibles.

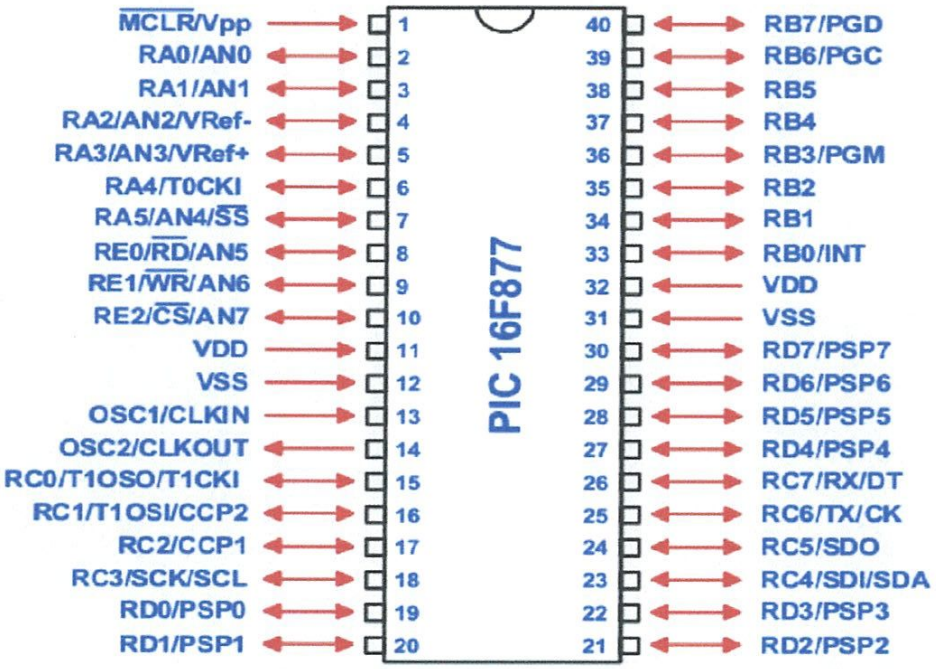


Figure (14) Brochage du PIC 16F877

8.1- Ports d'entrées/sortie :

Les Pics 16F877 contiennent les 5 ports suivants :

Port A : 6 pins E/S numérotées de RA0 à RA5.

Port B : 8 pins E/S numérotées de RB0 à RB7.

Port C : 8 pins E/S numérotées de RC0 à RC7.

Port D : 8 pins E/S numérotées de RD0 à RD7.

Port E : 3 pins E/S numérotées de RE0 à RE2.

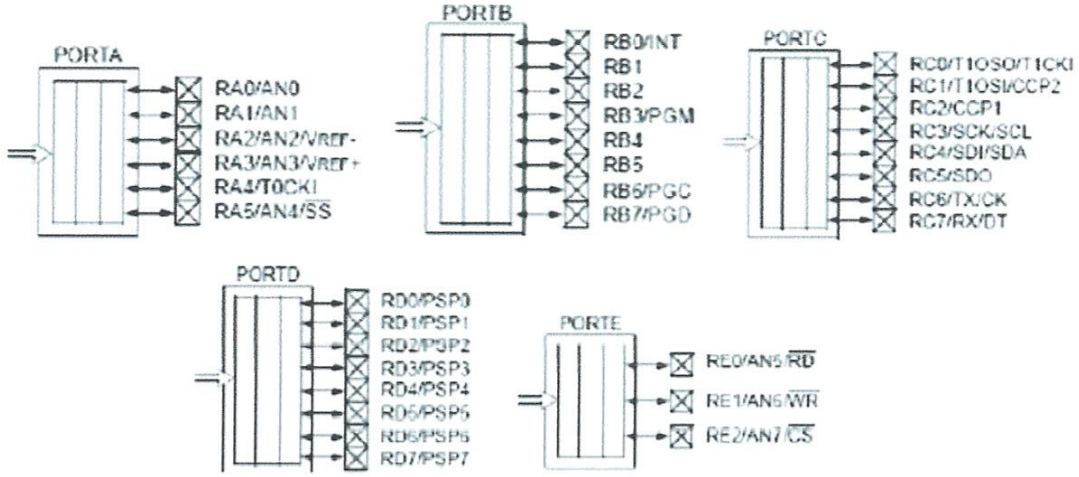


Figure (15) les ports d'Entrées/sorties du pic 16F877

Tous ces ports se trouvent dans la banque 0, mais tous leurs registres se trouvent dans la banque 1, pour déterminer les modes des ports (E/S), il faut sélectionner leurs registres TRISX

9-Mise en œuvre d'un afficheur L C D :

9.1-Présentation de l'afficheur LCD:

Les afficheurs à cristaux liquides sont des modules compacts intelligents **[11]** et nécessitent peu de composants externes. Ils sont utilisés avec beaucoup de facilité. Ils sont pratiquement les seuls à être utilisés sur les appareils à alimentation par pile.

Réaliser l'interface d'un écran LCD est une expérience très intéressante. Ses possibilités d'utilisation sont importantes car les afficheurs envahissent littéralement notre quotidien. L'écran étudié ici est du type alphanumérique (écriture de chiffres, de lettres et de signes) et est complètement standard. Il possède un générateur de caractères, un contrôleur intégré et est capable d'afficher 2 lignes de 16 caractères. Enfin, grâce à la présence d'une LED derrière l'écran, l'afficheur est doté d'un rétro éclairage du plus bel effet. Un tel écran coûte aux alentours de 1.50 Euro. Le prix variant selon le nombre de lignes, la présence d'un rétro éclairage et le système de transmission (série ou parallèle).

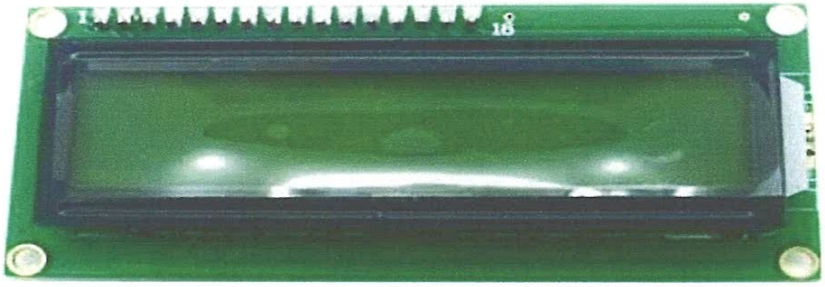


Figure (16): photo d'un afficheur LCD

9.2-Le brochage de l'afficheur:

Un circuit intégré spécialisé est chargé de la gestion du module. Il remplit une double fonction d'une part il commande l'afficheur [12] et d'une autre part se charge de la communication avec l'extérieur

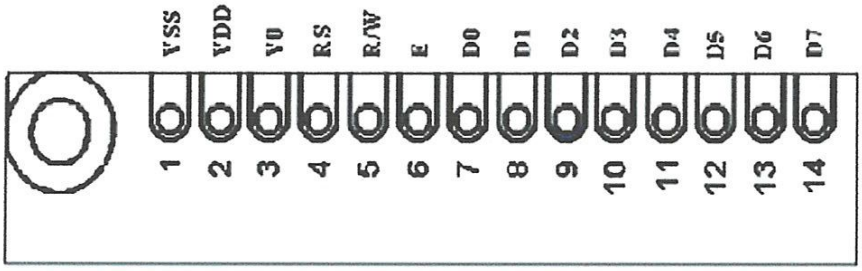


Figure (17) : Brochage d'un afficheur à cristaux liquides 2X16 caractères

L'écran à cristaux liquides, comporte une série de 14 broches aux rôles Suivants :

Broche	Nom	Niveau	Fonction
1	Vss	-	Masse
2	Vdd	-	Alimentation positive +5V
3	Vo	0-5V	Cette tension permet, en la faisant varier entre 0 et +5V, le réglage du contraste de l'afficheur.
4	RS	TTL	Sélection du registre (Register Select) Grâce à cette broche, l'afficheur est capable de faire la différence entre une commande et une donnée. Un niveau bas indique une commande et un niveau haut indique une donnée.
5	R/W	TTL	Lecture ou écriture (Read/Write) L : Écriture H : Lecture

6	E	TTL	Entrée de validation (Enable) active sur front descendant. Le niveau haut doit être maintenue pendant au moins 450 ns à l'état haut.
7	D0	TTL	Bus de données bidirectionnel 3 états (haute impédance lorsque E=0)
8	D1	TTL	
9	D2	TTL	
10	D3	TTL	
11	D4	TTL	
12	D5	TTL	
13	D6	TTL	
14	D7	TTL	
15	A	-	Anode rétro éclairage (+5V)
16	K	-	Cathode rétro éclairage (masse)

Tableau (1) Brochage d'un afficheur

Les connexions à réaliser sont simples puisque l'afficheur LCD dispose de peu de broches. Il faut évidemment, l'alimenter, le connecter à un bus de donnée (4 ou 8 bits), et connecter les broches **E**, **R/W** et **RS**.

9.3-Les caractères affichables:

- Les broches de données permettent également d'afficher un caractère selon le code fournit avec l'afficheur, il est ainsi possible d'afficher des caractères spéciaux. Ces codes sont mémorisés dans la CG RAM (Caractère Générator), chaque caractère est formé sur une matrice 5*8 points, Voire Tableau (2).
- Le transfert peut se faire sur l'ensemble des huit bites (D0 à D7) ou bien sur quatre bits (D4 à D7), dans ce dernier cas on économise quatre lignes du PIC mais il faut réaliser le transfert en deux fois [12], le choix de huit lignes ou quatre lignes se fait par programme selon le positionnement de certains bits (voir les tableaux qui suivent).
- L'afficheur est en mesure d'afficher 200 caractères :
  - ✓ de 00H à 07H : 8 caractères définissables par l'utilisateur

- ✓ de 20H à 7FH : 96 caractères ASCII excepté le caractère \ qui est remplacé par le signe ¥
- ✓ de A0H à DFH : 64 caractères japonais (alphabet kana)
- ✓ de E0H à FFH : 32 caractères spéciaux (accent, lettres grecques, ...)

NO.7066-0B

b7-b4 / b3-b0	0000	0001	0010	0011	0100	0101	0110	0111	1000	1001	1010	1011	1100	1101	1110	1111
0000	CG RAM (1)															
0001	(1)															
0010	(3)															
0011	(4)															
0100	(5)															
0101	(6)															
0110	(7)															
0111	(8)															
1000	(1)															
1001	(2)															
1010	(3)															
1011	(4)															
1100	(5)															
1101	(6)															
1110	(7)															
1111	(8)															

Tableau (2) les caractères affichables

9.4-Principales instructions :

Après avoir défini le sens de déplacement, les caractères apparaissent au dessus du curseur (qu'il soit visualisé ou non). On peut déplacer le curseur en utilisant la commande « **Set DD RAM address** » [12].

Un défilement des caractères est aussi possible, il suffit pour cela d'envoyer la commande 07H (I/D=1, S=1), sans oublier de positionner **RS** à 0. Mais il faut faire attention car un décalage de l'affichage décale également les adresses de l'afficheur de sorte que l'extrême gauche de la ligne supérieure ne soit plus adressable à l'adresse 00. La commande « **Return home** » permet de repositionner les adresses de l'affichage.

Instructions	**Code**										**Description**	**Durée**
	RS	**R/W**	**D7**	**D6**	**D5**	**D4**	**D3**	**D2**	**D1**	**D0**		
Clear Display	0	0	0	0	0	0	0	0	0	1	Efface l'ensemble de la mémoire de donnée sans toucher au générateur de caractères. Ramène le curseur en position « home », à l'adresse 00.	1,52 ms
Return home	0	0	0	0	0	0	0	0	1	X	Ramène le curseur en position « home », à l'adresse 00. Si l'affichage était décalé, il est remis à sa position d'origine : l'adresse 00 se trouve à nouveau en haut à gauche.	1,52ms
Entry mode set	0	0	0	0	0	0	0	1	I/D	S	Définit le sens de déplacement du curseur après	37µs

											l'apparition d'un caractère (vers la gauche si I/D=1, vers la droite si I/D=0) et si l'affichage accompagne le curseur dans son déplacement ou non (S).	
Display on/off control	0	0	0	0	0	0	1	D	C	B	Met l'affichage en ou hors fonction l'affichage (D), le curseur (C), le clignotement du curseur (B).	37 µs
Cursor and display shift	0	0	0	0	0	1	S/C	R/L	X	X	Deplace le curseur (S/C=1) ou l'affichage (S/C=0) d'une position vers la gauche (R/L=1) ou la droite (R/L=0) sans changer la DD RAM.	**37 µs**
Function set	0	0	0	0	1	DL	N	F	X	X	Définit la taille de l'interface (DL=0 pour mode 4 bits, DL=1 pour mode 8 bits), le nombre de lignes (NL=0 pour 1 ligne, N=1 pour 2 ou 4 lignes), et la taille des fontes (F=0 pour des caractères 5x7, F=1 pour des caractères	**37 µs**

											5x10).	
Set CG RAM address	0	0	0	1	A5	A4	A3	A2	A1	A0	Définit l'adresse de la CG RAM. Les données de la CG RAM sont envoyées après cette commande.	37 µs
Set DD RAM address	0	0	1	A6	A5	A4	A3	A2	A1	A0	Définit l'adresse de la DD RAM. Les données de la DD RAM sont envoyées après cette commande.	37 µs
Read busy flag & address	0	1	BF	A6	A5	A4	A3	A2	A1	A0	Lit le flag busy (BF), et l'adresse de la position du curseur. BF vaut 0 si l'afficheur accepte une instruction, 1 s'il est occupé	1 µs
Write data to CG or DD RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Ecrit des données dans la DD RAM ou la CG RAM.	37 µs
Read data	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Lit les données de la DD RAM ou de la CG RAM.	37 µs

Tableau(3) Le jeu de commandes standard

9.5-Connexion de l'afficheur sur la carte:

Dans notre application, nous avons utilisé un écran LCD alphanumérique de 2 lignes et de 16 caractères.

Cet écran est connecté au microcontrôleur sur la port D, on a conservé les broches du port D pour lier les données (D0 à D6) [11].

Un ensemble d'instruction spécialisée permet de le piloter très facilement est disponible dans le logiciel flowcode.

Il est également possible de piloter l'écran directement avec des instructions de commande sous forme binaire.

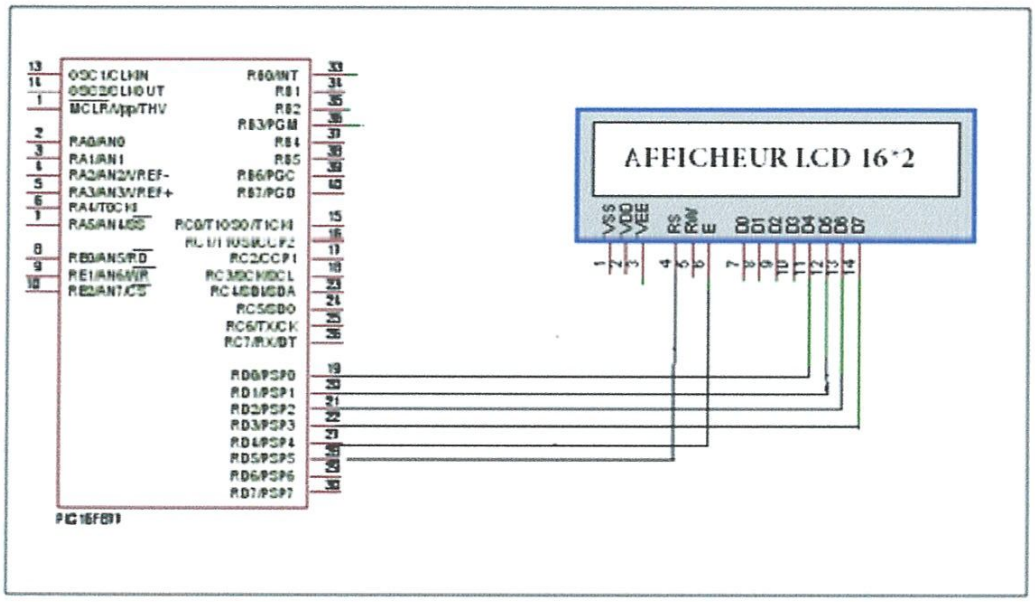


Figure (18) : montage de l'afficheur LCD avec le PIC

10-Programmation du pic :

10.1-Le flowcode

Flowcode permet de créer des programmes relativement complexes pour les microcontrôleurs de la famille des PICmicro® d'Arizona Microchip [13].

Flowcode permet de créer des applications pour des microcontrôleurs en sélectionnant et en plaçant des icônes sur un organigramme pour créer des Programmes simples. Ces programmes peuvent contrôler des périphériques externes connectés au microcontrôleur comme des LEDs, un afficheur LCD etc…

Une fois que l'organigramme est élaboré, Flowcode permet de simuler son comportement avant de le compiler, de l'assembler et de le transférer dans un microcontrôleur PICmicro®.

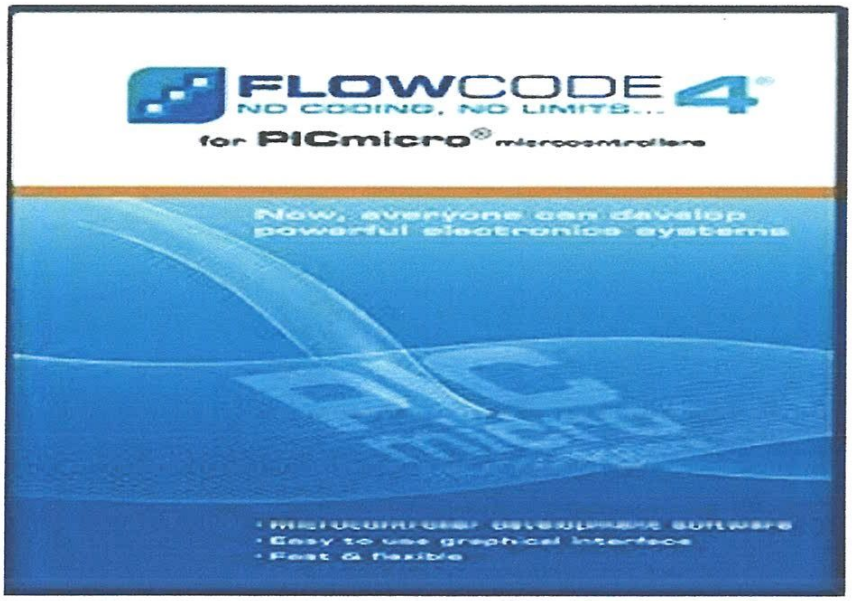


Figure (19) : logiciel flowcode

Pour atteindre cet objectif avec Flowcode, il suffit de réaliser les étapes suivantes :

1- Créer organigramme, spécifier le microcontrôleur cible que vous utiliserez.

2- Sélectionner et faire glisser les icônes de la barre d'outils sur l'organigramme pour programmer votre application.

3-Ajouter les périphériques externes nécessaires en cliquant sur les boutons correspondants dans la barre d'outils des composants, éditer leurs propriétés, spécifier comment ils sont connectés au microcontrôleur et appeler les macros correspondantes aux périphériques utilisés.

4- Faire tourner la simulation pour vous assurer que l'application se comporte comme vous le voulez.

5-Transférer l'application dans le microcontrôleur cible en compilant le programme en C, puis en l'assemblant et finalement en produisant le code objet.

L'environnement de Flowcode consiste en une aire de travail principale dans laquelle s'affiche l'organigramme, plusieurs barres d'outils qui vous permettent d'ajouter des icônes et des composants à votre application, trois fenêtres spécifiques pour montrer l'état du microcontrôleur ainsi que les composants

attachés et enfin deux fenêtres qui montrent les variables et les appels de macros lorsque vous simulez votre application.

Conclusion :

Le deuxième chapitre, fait l'objet d'une petite initiation concernant les différents étages du dispositif en donnant une idée générale sur les caractéristiques techniques. Et le brochage de chaque composant.

Chapitre 3

SIMULATION ET REALISATION PRATIQUE

Introduction

Ce chapitre n'est qu'une suite logique des chapitres présidents, en effet, le premier chapitre nous a donné une idée du cahier de charge de notre projet.
Le second chapitre est une description du des différents composants et outils de travail tel que le microcontrôleur, l'afficheur LCD et les logiciels de simulation.

L'objectif de ce 3éme et dernier chapitre est l'étude pratique de la carte de commande de gestion de la file d'attente équipée par un distributeur de tickets.

Tout d'abords nous commençons par la réalisation du circuit imprimé, ensuite et après la soudure des composants nous effectuerons touts les tests nécessaires enfin nous attaquerons l'alimentation stabilisée.

1-Les particularités électriques :

L'alimentation stabilisée

La grande majorité des équipements électroniques ont besoin d'une source de courant continu qui peut être une pile ou batterie, mais généralement une alimentation stabilisée constituée d'un circuit transformant le courant alternatif du secteur (220v, 50Hz) en courant continu. Les circuits électroniques courants ont généralement besoin d'une tension de 5v.
L'afficheur LCD et le microcontrôleur fonctionnent sous une tenson de 5v.
Le rôle des alimentations continues est de fournir les tensions et les courants nécessaires au fonctionnement des circuits électroniques avec le minimum d'ondulations résiduelles et la meilleure régulation possible. Elles doivent, de plus, souvent limiter le courant fourni en cas de surcharge ainsi que la tension continue qu'elles délivrent, ceci afin de protéger les composants fragiles.
Une alimentation est un montage transformant la tension alternative du secteur en une tension continue basse tension. Une alimentation secteur est composée d'un transformateur, d'un redresseur, d'un filtre et d'une stabulation/régulation selon le besoin.

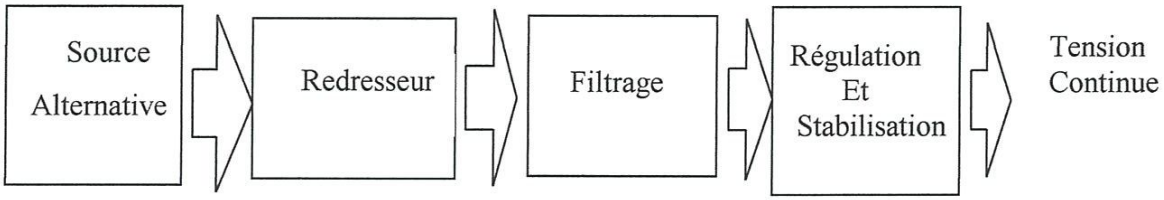


Figure (18a): schéma synoptique d'une Chaine de conversion tension alternative-continue

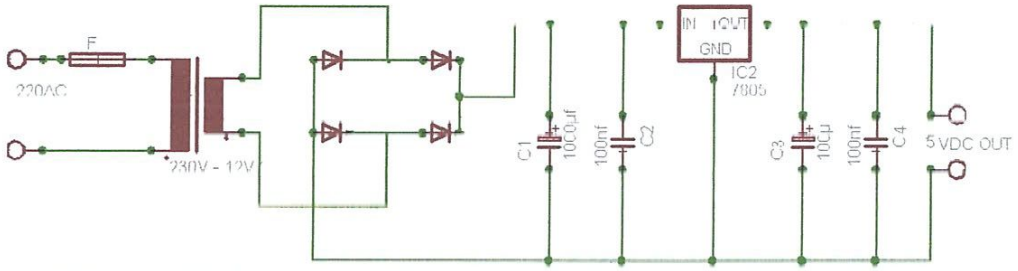
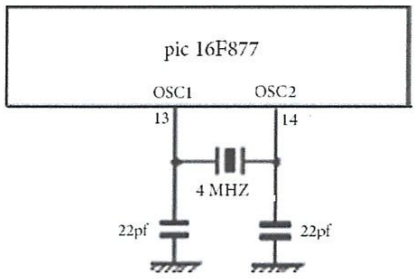


Figure (18b): schéma électrique de l'alimentation stabilisée

1-Connexion avec l'horloge à quartz :

L'horloge utilisée pour le fonctionnement de pic, et constituée autour d'un quartz associé à un circuit RC externe.

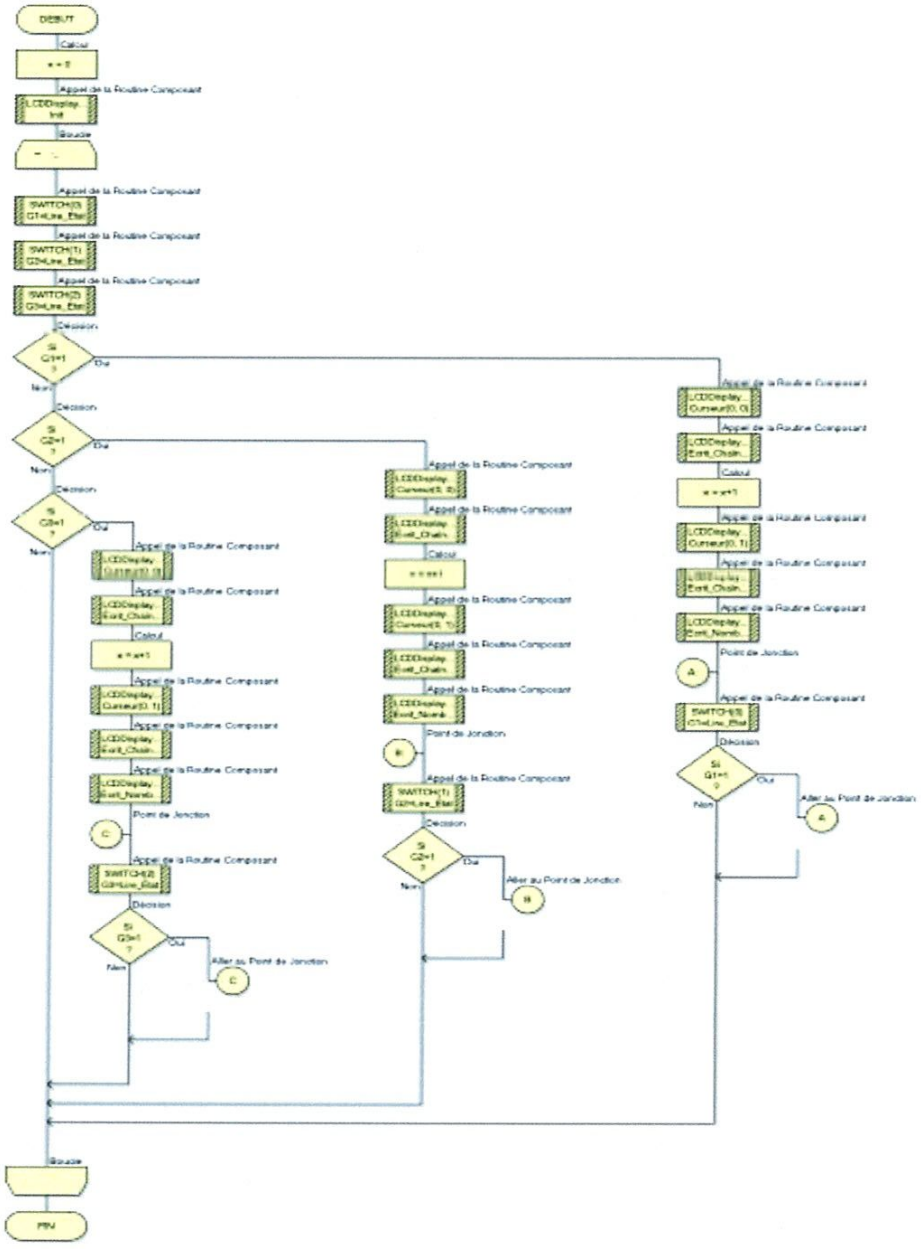
Cette horloge n'est autre qu'un oscillateur de 4MHZ qui sera branché sur les bronches OSC1 et OSC2 d'un pic avec deux capacités de 22pf de la façon suivante :



Figure(19) : Brochage du circuit d'oscillation PIC16F877

2-Organigramme :

Programme introduit dans le pic pour gérer la file d'attente. Créer par « FLOWCODE ». Ce logiciel (FLOWCODE) créera le programme qui doit être introduit dans le PIC à partir de l'organigramme tracé dans le même logiciel.



Figure(22) l'organigramme

La première partie de l'organigramme est une initialisation il n'ya pas de clients, aucun affichage dans l'afficheur LCD, les 3 guichets sont libres.

La deuxième partie du l'organigramme est la décision « si guichet 1 contient un client », au début de ce cas le curseur est a (0,0), le numéro du guichet 1 s'affiche, le numéro du client est incrémenté de 0 à 1 et s'affiche.

Quand le guichet 1 est occupé le client suivant est appelé au guichet numéro2 lorsque l'administrateur arrive à son guichet 2 et appuie pour appeler le client suivant. A ce moment le numéro du client est incrémenté et apparait avec le numéro du guichet sur l'afficheur LCD. Cela se répète à chaque fois qu'un administrateur se libère et appuie sur son bouton poussoir.

3-Schéma de fonctionnement :

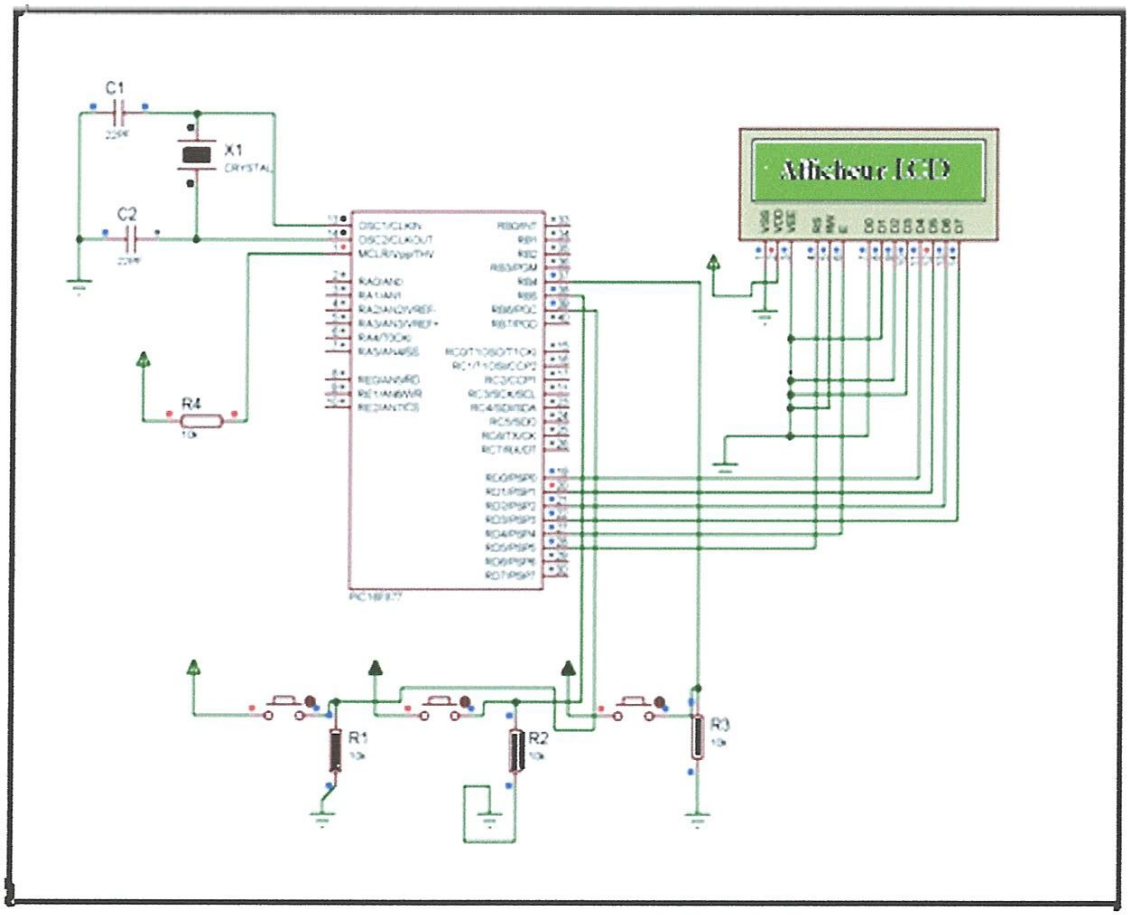


Figure (23) : schéma de fonctionnement

4- Simulation :

4.1- Logiciel de Simulation (Isis)

Isis est un simulateur de cartes électroniques intégré dans le logiciel PROTEUS de l'entreprise LabcenterElectrnics **[14].** L'utilisation de ce logiciel est simple grâce à l'interface graphique qui rend claires toutes les commandes.

La bibliothèque est géante et contient presque tous les composants électroniques connus. De plus la recherche et l'importation de composants sont simples.

Isis est orientée vers les électroniciens débutants, ainsi que les développeurs et les professionnels.

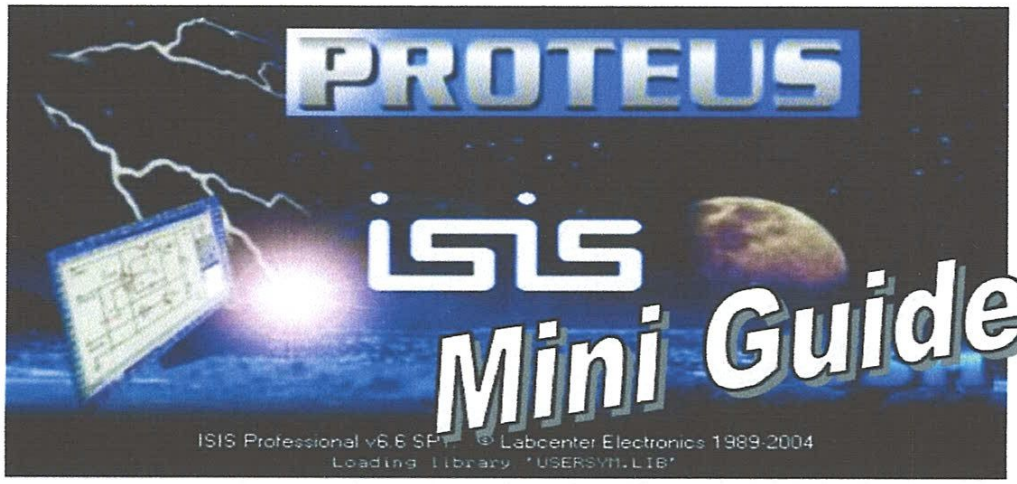


Figure (24) Logiciel de simulation isis

4.2 Ares :

C'est un outil de conception des cartes électroniques imprimées, intégré dans le logiciel PROTEUS de l'entreprise Labcenter Electronics. Après avoir écrit et compiler un programme, on peut faire la simulation de la carte à l'aide de Isis. Lorsqu'on constate que le programme fonctionne correctement et que les composants sont bien placés, on peut donc passer à la conception de la carte imprimée sous Ares.

Parmi les caractéristiques de ce logiciel c'est qu'il permet à l'utilisateur d'ajouter n'importe quelle empreinte qui n'existe pas dans la bibliothèque. De plus, son utilisation est simple et facile.

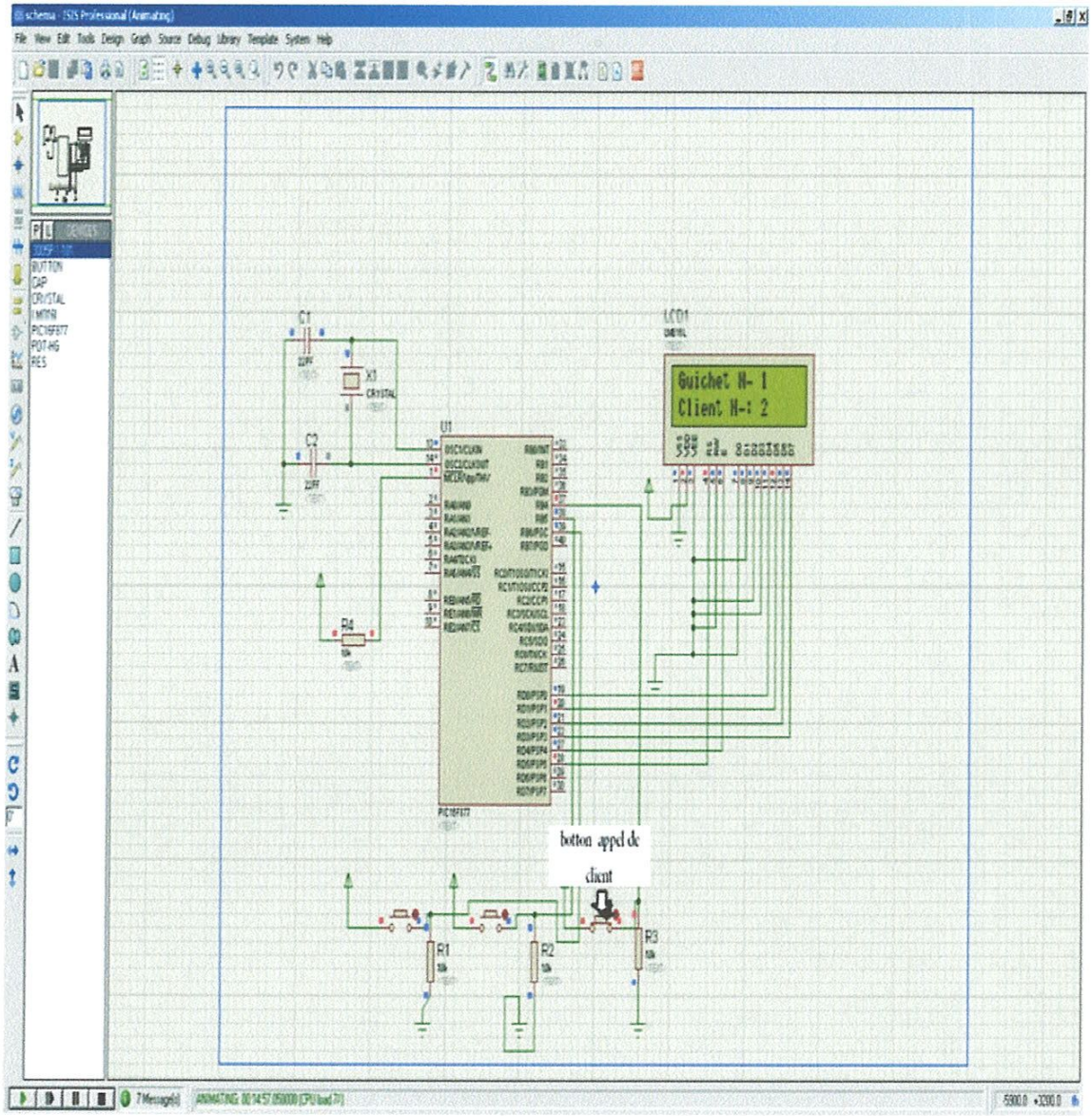


Figure (25) Simulation par isis

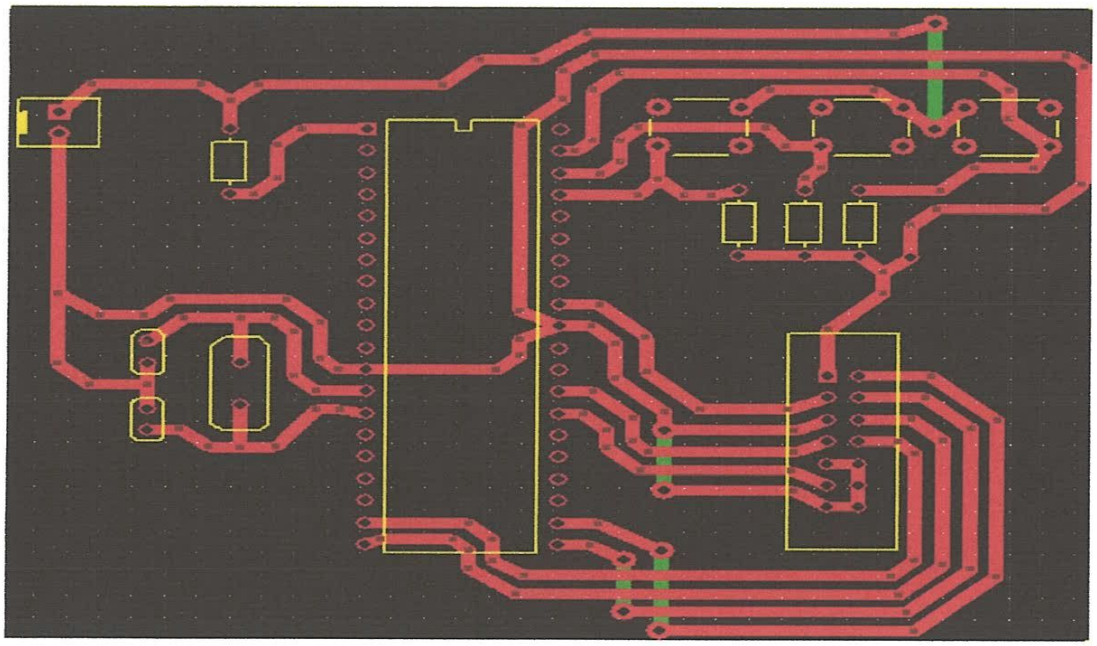
5-Réalisation pratique :

Pour réaliser les circuits déjà définis dans le chapitre 2, nous avons choisi de les implanter sur un circuit imprimé où nous avons distribué la masse sur une grande

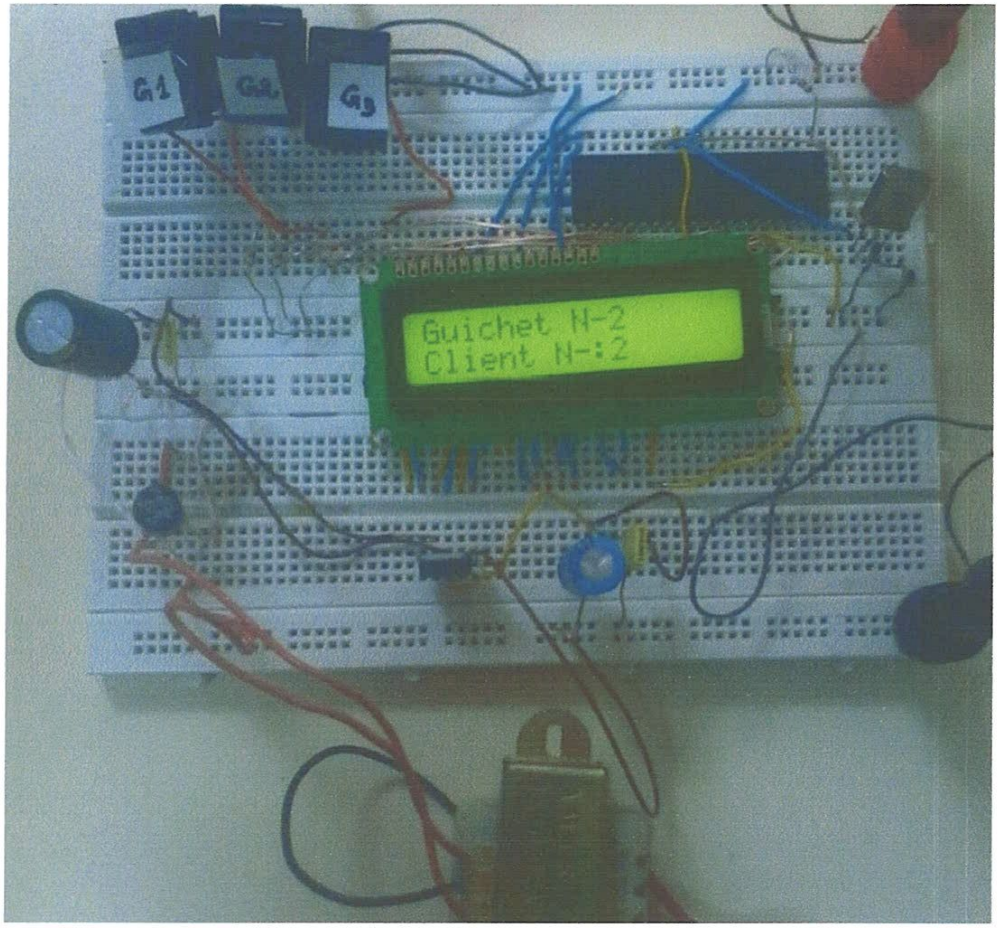
partie pour la bonne raison de diminuer l'effet des bruits causés par l'alimentation où les hautes fréquences qui peuvent être générées par l'environnement de la carte.

Il est préférable que le circuit de commande soit imprimé sur une plaquette double face pour des raisons de complexité des connexions entre les composants utilisés. A défaut des plaquettes simples faces seront utilisées et appuyées par des shunts.

Nous avons utilisé le logiciel PROTEUS 6 pour le routage du circuit imprimé, ce logiciel nous a beaucoup simplifié la tache.



Figure(26) circuit imprimé



Figure(27) : réalisation sur la plaque dc

Conclusion :

Nous venons de présenter dans ce chapitre la simulation et la réalisation de notre projet, avec la programmation des PIC afin d'avoir un bon fonctionnement qui facilite par la suite l'utilisation de se dernier et voila nous avons un prototype de système de commande de gestion d'une salle d'attente équipée d'un distributeur de tickets.

Conclusion Générale

Conclusion générale

Un système de commande de la gestion d'une salle d'attente équipé d'un distributeur de tickets à base de micro contrôleur PIC 16F877 a fait l'objet de notre tache.

L'élaboration de ce travail dans le cadre du projet du fin d'étude, nous a permis d'approfondir nos connaissances théoriques en électronique, d'acquérir une bonne expérience au niveau de la réalisation pratique et au niveau de la recherche bibliographique.

Lors de cette manipulation, nous avons essayé de proposer un programme qui peut être bien utilisé dans des différents cas.

Finalement, nous estimons que ce projet peut participer dans le développement de la bibliothèque du département et sert par la suite à développer des applications autour du PIC16F877.

Bibliographie

1- les microcontrôleurs pic, Bernard Béghyn, Lavoisier ; édition 2003
Bibliothèque centrale université de Guelma réf : L/621.781

2- programmation en c des PICs, Christian tavenier, dunod, édition 2005
Bibliothèque centrale université de Guelma réf : L/004.1014

3- Description et mise en œuvre, Christian tavenier, dunod, édition 2002

4-La programmation des PICs. BIGONOFF Révision 6 ; Edition : [2001].

Sites Web

1 – wikipedia.org/wiki/ file d'attente.

2- http://www.gerad.ca/Sebastien.Le.Digabel/MTH2302D/14_files_attente.pdf

3-\Gestion de files d'attente - Gestion des temps, contrôle d'accès.htm

4- http://www.mplogic.com/gfa/index.html.fr

5- http// www.charvet-industries.com

6 - http://www.technologuepro.com/projet-fin-etudes/gestion-distributeur-ticket-28.html

7- http://www.alpros-dz.com/K211.pdf

8- http://www.alpros-dz.com/K418.pdf

9- http// www.microchip.com

10-http://www.jmb-electronique.com/Bigonoff/Part1_R6.pdf

11-http://iut-tice.ujf-grenoble.fr/tice-espaces/GEII/EP-od/wupload/File/II2_Cours/Chapitre10.pdf

12-http://www.iset-ti.be/Portals/0/Jamart/Brochage%20LCD.pdf

13-http://www.elektor.fr/Uploads/2011/4/Notice-FLOWCODE-V4-3.pdf

14-http // www.technologuepro.com/tags/proteus.html

Programme de Pic

```
;////////////////////////////////////////////////////////////////////////////
;// Code Generator: BoostC Compiler - http://www.sourceboost.com
;// Version      : 6.95
;// License Type  : Lite License (Unregistered)
;// Limitations   : PIC12,PIC16 max code size:2048 words, max RAM banks:2, Non commercial use only
;////////////////////////////////////////////////////////////////////////////

            include "P16F877.inc"
; Heap block 0, size:95 (0x000000A0 - 0x000000FE)
__HEAP_BLOCK0_BANK              EQU   0x00000001
__HEAP_BLOCK0_START_OFFSET      EQU   0x00000020
__HEAP_BLOCK0_END_OFFSET        EQU   0x0000007E
; Heap block 1, size:21 (0x0000005B - 0x0000006F)
__HEAP_BLOCK1_BANK              EQU   0x00000000
__HEAP_BLOCK1_START_OFFSET      EQU   0x0000005B
__HEAP_BLOCK1_END_OFFSET        EQU   0x0000006F
; Heap block 2, size:0 (0x00000000 - 0x00000000)
__HEAP_BLOCK2_BANK              EQU   0x00000000
__HEAP_BLOCK2_START_OFFSET      EQU   0x00000000
__HEAP_BLOCK2_END_OFFSET        EQU   0x00000000
; Heap block 3, size:0 (0x00000000 - 0x00000000)
__HEAP_BLOCK3_BANK              EQU   0x00000000
__HEAP_BLOCK3_START_OFFSET      EQU   0x00000000
__HEAP_BLOCK3_END_OFFSET        EQU   0x00000000
__div_16_1_00003_arg_a          EQU   0x0000004B ; bytes:2
__div_16_1_00003_arg_b          EQU   0x0000004D ; bytes:2
CompTempVarRet243               EQU   0x00000034 ; bytes:2
__div_16_1_00003_1_r            EQU   0x00000051 ; bytes:2
__div_16_1_00003_1_i            EQU   0x00000053 ; bytes:1
gbl_status                      EQU   0x00000003 ; bytes:1
gbl_16_LSR                      EQU   0x00000020 ; bytes:4
gbl_float_detect_tininess       EQU   0x00000039 ; bytes:1
gbl_float_rounding_mode         EQU   0x0000003A ; bytes:1
gbl_float_exception_flags       EQU   0x0000003B ; bytes:1
gbl_17_gbl_aSig                 EQU   0x00000024 ; bytes:4
gbl_17_gbl_bSig                 EQU   0x00000028 ; bytes:4
gbl_17_gbl_zSig                 EQU   0x0000002C ; bytes:4
gbl_17_gbl_aExp                 EQU   0x0000003C ; bytes:1
gbl_17_gbl_bExp                 EQU   0x0000003D ; bytes:1
gbl_17_gbl_zExp                 EQU   0x00000037 ; bytes:2
gbl_17_gbl_aSign                EQU   0x0000003E ; bytes:1
gbl_17_gbl_bSign                EQU   0x0000003F ; bytes:1
gbl_17_gbl_zSign                EQU   0x00000040 ; bytes:1
gbl_17_gbl_zSigZero             EQU   0x00000041 ; bytes:1
gbl_17_gbl_ret                  EQU   0x00000030 ; bytes:4
gbl_indf                        EQU   0x00000000 ; bytes:1
gbl_tmr0                        EQU   0x00000001 ; bytes:1
gbl_pcl                         EQU   0x00000002 ; bytes:1
gbl_fsr                         EQU   0x00000004 ; bytes:1
gbl_porta                       EQU   0x00000005 ; bytes:1
gbl_portb                       EQU   0x00000006 ; bytes:1
gbl_portc                       EQU   0x00000007 ; bytes:1
gbl_portd                       EQU   0x00000008 ; bytes:1
gbl_porte                       EQU   0x00000009 ; bytes:1
gbl_pclath                      EQU   0x0000000A ; bytes:1
gbl_intcon                      EQU   0x0000000B ; bytes:1
gbl_pir1                        EQU   0x0000000C ; bytes:1
gbl_pir2                        EQU   0x0000000D ; bytes:1
gbl_tmr1l                       EQU   0x0000000E ; bytes:1
gbl_tmr1h                       EQU   0x0000000F ; bytes:1
gbl_t1con                       EQU   0x00000010 ; bytes:1
gbl_tmr2                        EQU   0x00000011 ; bytes:1
gbl_t2con                       EQU   0x00000012 ; bytes:1
gbl_sspbuf                      EQU   0x00000013 ; bytes:1
gbl_sspcon                      EQU   0x00000014 ; bytes:1
gbl_ccpr1l                      EQU   0x00000015 ; bytes:1
gbl_ccpr1h                      EQU   0x00000016 ; bytes:1
gbl_ccp1con                     EQU   0x00000017 ; bytes:1
gbl_rcsta                       EQU   0x00000018 ; bytes:1
gbl_txreg                       EQU   0x00000019 ; bytes:1
```

```
gbl_rcreg                EQU     0x0000001A ; bytes:1
gbl_ccpr2l               EQU     0x0000001B ; bytes:1
gbl_ccpr2h               EQU     0x0000001C ; bytes:1
gbl_ccp2con              EQU     0x0000001D ; bytes:1
gbl_adresh               EQU     0x0000001E ; bytes:1
gbl_adcon0               EQU     0x0000001F ; bytes:1
gbl_option_reg           EQU     0x00000081 ; bytes:1
gbl_trisa                EQU     0x00000085 ; bytes:1
gbl_trisb                EQU     0x00000086 ; bytes:1
gbl_trisc                EQU     0x00000087 ; bytes:1
gbl_trisd                EQU     0x00000088 ; bytes:1
gbl_trise                EQU     0x00000089 ; bytes:1
gbl_pie1                 EQU     0x0000008C ; bytes:1
gbl_pie2                 EQU     0x0000008D ; bytes:1
gbl_pcon                 EQU     0x0000008E ; bytes:1
gbl_sspcon2              EQU     0x00000091 ; bytes:1
gbl_pr2                  EQU     0x00000092 ; bytes:1
gbl_sspadd               EQU     0x00000093 ; bytes:1
gbl_sspstat              EQU     0x00000094 ; bytes:1
gbl_txsta                EQU     0x00000098 ; bytes:1
gbl_spbrg                EQU     0x00000099 ; bytes:1
gbl_adresl               EQU     0x0000009E ; bytes:1
gbl_adcon1               EQU     0x0000009F ; bytes:1
gbl_eedata               EQU     0x0000010C ; bytes:1
gbl_eeadr                EQU     0x0000010D ; bytes:1
gbl_eedath               EQU     0x0000010E ; bytes:1
gbl_eeadrh               EQU     0x0000010F ; bytes:1
gbl_eecon1               EQU     0x0000018C ; bytes:1
gbl_eecon2               EQU     0x0000018D ; bytes:1
gbl_FCV_X                EQU     0x00000042 ; bytes:1
gbl_FCV_G1               EQU     0x00000043 ; bytes:1
gbl_FCV_G2               EQU     0x00000044 ; bytes:1
gbl_FCV_G3               EQU     0x00000045 ; bytes:1
Wdt_msDela_00045_1_i          EQU     0x0000004C ; bytes:1
Wdt_Delay__00047_arg_delay    EQU     0x00000048 ; bytes:2
Wdt_Delay__00047_1_i          EQU     0x0000004A ; bytes:2
CompTempVar2178               EQU     0x0000004C ; bytes:1
FCD_LCDDis_00048_arg_in       EQU     0x00000057 ; bytes:1
FCD_LCDDis_00048_arg_mask     EQU     0x00000058 ; bytes:1
FCD_LCDDis_00048_1_pt         EQU     0x00000059 ; bytes:1
FCD_LCDDis_0004D_arg_x        EQU     0x00000046 ; bytes:1
FCD_LCDDis_0004D_arg_y        EQU     0x00000047 ; bytes:1
FCD_LCDDis_0004E_arg_Number   EQU     0x00000046 ; bytes:2
FCD_LCDDis_0004E_1_tmp_int    EQU     0x00000048 ; bytes:2
FCD_LCDDis_0004E_1_tmp_byte   EQU     0x0000004A ; bytes:1
CompTempVar2206               EQU     0x0000004B ; bytes:1
CompTempVar2209               EQU     0x0000004F ; bytes:1
CompTempVar2212               EQU     0x0000004F ; bytes:1
CompTempVar2213               EQU     0x00000050 ; bytes:1
CompTempVar2216               EQU     0x0000004B ; bytes:1
CompTempVar2217               EQU     0x0000004F ; bytes:1
CompTempVar2218               EQU     0x00000050 ; bytes:1
CompTempVar2221               EQU     0x0000004B ; bytes:1
CompTempVar2222               EQU     0x0000004F ; bytes:1
CompTempVar2223               EQU     0x00000050 ; bytes:1
FCD_LCDDis_0004F_arg_String   EQU     0x00000046 ; bytes:2
FCD_LCDDis_0004F_arg_MSZ_String EQU   0x00000055 ; bytes:1
FCD_LCDDis_0004F_1_idx        EQU     0x00000056 ; bytes:1
CompTempVarRet2234            EQU     0x00000047 ; bytes:1
FCD_SWITCH_00054_1_switchval  EQU     0x00000046 ; bytes:1
CompTempVarRet2235            EQU     0x00000047 ; bytes:1
FCD_SWITCH_00057_1_switchval  EQU     0x00000046 ; bytes:1
CompTempVarRet2236            EQU     0x00000047 ; bytes:1
FCD_SWITCH_0005A_1_switchval  EQU     0x00000046 ; bytes:1
CompTempVar2237               EQU     0x00000048 ; bytes:13
CompTempVar2239               EQU     0x00000048 ; bytes:12
CompTempVar2241               EQU     0x00000048 ; bytes:13
CompTempVar2243               EQU     0x00000048 ; bytes:12
CompTempVar2245               EQU     0x00000048 ; bytes:13
CompTempVar2247               EQU     0x00000048 ; bytes:12
```

```
delay_us_00000_arg_del          EQU          0x0000004D ; bytes:1
delay_10us_00000_arg_del        EQU          0x0000005A ; bytes:1
Int1Context                 EQU    0x0000007F ; bytes:1
Int1BContext                EQU    0x00000034 ; bytes:3
            ORG 0x00000000
            GOTO     _startup
            ORG 0x00000004
            MOVWF Int1Context
            SWAPF STATUS, W
            BCF STATUS, RP0
            BCF STATUS, RP1
            MOVWF Int1BContext
            SWAPF PCLATH, W
            MOVWF Int1BContext+D'1'
            SWAPF FSR, W
            MOVWF Int1BContext+D'2'
            BCF PCLATH,3
            BCF PCLATH,4
            GOTO     interrupt
            ORG 0x00000010
delay_us_00000
; { delay_us ; function begin
            MOVLW 0x03
            ADDWF delay_us_00000_arg_del, F
            RRF delay_us_00000_arg_del, F
            RRF delay_us_00000_arg_del, F
            MOVLW 0x7F
            ANDWF delay_us_00000_arg_del, F
label1
            NOP
            DECFSZ delay_us_00000_arg_del, F
            GOTO     label1
            RETURN
; } delay_us function end

            ORG 0x0000001A
delay_10us_00000
; { delay_10us ; function begin
label2
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            NOP
            DECFSZ delay_10us_00000_arg_del, F
            GOTO     label2
            RETURN
; } delay_10us function end

            ORG 0x00000024
Wdt_msDela_00045
; { Wdt_msDelay ; function begin
            BCF STATUS, RP0
            BCF STATUS, RP1
            CLRF Wdt_msDela_00045_1_i
label3
            MOVLW 0x4B
            SUBWF Wdt_msDela_00045_1_i, W
            BTFSC STATUS,C
            RETURN
            CLRWDT
            MOVLW 0x0A
            MOVWF delay_us_00000_arg_del
            CALL delay_us_00000
            INCF Wdt_msDela_00045_1_i, F
            GOTO     label3
; } Wdt_msDelay function end
```

```
            ORG 0x00000031
__div_16_1_00003
; { __div_16_16 ; function begin
            CLRF __div_16_1_00003_1_r
            CLRF __div_16_1_00003_1_r+D'1'
            CLRF CompTempVarRet243
            CLRF CompTempVarRet243+D'1'
            CLRF __div_16_1_00003_1_i
label4
            BTFSC __div_16_1_00003_1_i,4
            RETURN
            BCF STATUS,C
            RLF CompTempVarRet243, F
            RLF CompTempVarRet243+D'1', F
            RLF __div_16_1_00003_arg_a, F
            RLF __div_16_1_00003_arg_a+D'1', F
            RLF __div_16_1_00003_1_r, F
            RLF __div_16_1_00003_1_r+D'1', F
            MOVF __div_16_1_00003_arg_b+D'1', W
            SUBWF __div_16_1_00003_1_r+D'1', W
            BTFSS STATUS,Z
            GOTO     label5
            MOVF __div_16_1_00003_arg_b, W
            SUBWF __div_16_1_00003_1_r, W
label5
            BTFSS STATUS,C
            GOTO     label6
            MOVF __div_16_1_00003_arg_b, W
            SUBWF __div_16_1_00003_1_r, F
            MOVF __div_16_1_00003_arg_b+D'1', W
            BTFSS STATUS,C
            DECF __div_16_1_00003_1_r+D'1', F
            SUBWF __div_16_1_00003_1_r+D'1', F
            BSF CompTempVarRet243,0
label6
            INCF __div_16_1_00003_1_i, F
            GOTO     label4
; } __div_16_16 function end

            ORG 0x00000050
Wdt_Delay__00047
; { Wdt_Delay_Ms ; function begin
            CLRF Wdt_Delay__00047_1_i
            CLRF Wdt_Delay__00047_1_i+D'1'
label7
            MOVF Wdt_Delay__00047_1_i+D'1', W
            XORLW 0x80
            MOVWF CompTempVar2178
            MOVF Wdt_Delay__00047_arg_delay+D'1', W
            XORLW 0x80
            SUBWF CompTempVar2178, W
            BTFSS STATUS,Z
            GOTO     label8
            MOVF Wdt_Delay__00047_arg_delay, W
            SUBWF Wdt_Delay__00047_1_i, W
label8
            BTFSC STATUS,C
            RETURN
            CALL Wdt_msDela_00045
            INCF Wdt_Delay__00047_1_i, F
            BTFSC STATUS,Z
            INCF Wdt_Delay__00047_1_i+D'1', F
            GOTO     label7
; } Wdt_Delay_Ms function end

            ORG 0x00000063
FCD_LCDDis_00048
; { FCD_LCDDisplay0_RawSend ; function begin
            BCF gbl_portd,0
            BCF gbl_portd,1
```

```
            BCF gbl_portd,2
            BCF gbl_portd,3
            BCF gbl_portd,5
            BCF gbl_portd,4
            SWAPF FCD_LCDDis_00048_arg_in, W
            ANDLW 0x0F
            MOVWF FCD_LCDDis_00048_1_pt
            MOVLW 0x0F
            ANDWF FCD_LCDDis_00048_1_pt, F
            BTFSC FCD_LCDDis_00048_1_pt,0
            BSF gbl_portd,0
            BTFSC FCD_LCDDis_00048_1_pt,1
            BSF gbl_portd,1
            BTFSC FCD_LCDDis_00048_1_pt,2
            BSF gbl_portd,2
            BTFSC FCD_LCDDis_00048_1_pt,3
            BSF gbl_portd,3
            MOVF FCD_LCDDis_00048_arg_mask, F
            BTFSS STATUS,Z
            BSF gbl_portd,5
            MOVLW 0x0A
            MOVWF delay_10us_00000_arg_del
            CALL delay_10us_00000
            BSF gbl_portd,4
            MOVLW 0x0A
            MOVWF delay_10us_00000_arg_del
            CALL delay_10us_00000
            BCF gbl_portd,4
            MOVLW 0x0F
            ANDWF FCD_LCDDis_00048_arg_in, W
            MOVWF FCD_LCDDis_00048_1_pt
            MOVLW 0x0A
            MOVWF delay_10us_00000_arg_del
            CALL delay_10us_00000
            BCF gbl_portd,0
            BCF gbl_portd,1
            BCF gbl_portd,2
            BCF gbl_portd,3
            BCF gbl_portd,5
            BCF gbl_portd,4
            BTFSC FCD_LCDDis_00048_1_pt,0
            BSF gbl_portd,0
            BTFSC FCD_LCDDis_00048_1_pt,1
            BSF gbl_portd,1
            BTFSC FCD_LCDDis_00048_1_pt,2
            BSF gbl_portd,2
            BTFSC FCD_LCDDis_00048_1_pt,3
            BSF gbl_portd,3
            MOVF FCD_LCDDis_00048_arg_mask, F
            BTFSS STATUS,Z
            BSF gbl_portd,5
            MOVLW 0x0A
            MOVWF delay_10us_00000_arg_del
            CALL delay_10us_00000
            BSF gbl_portd,4
            MOVLW 0x0A
            MOVWF delay_10us_00000_arg_del
            CALL delay_10us_00000
            BCF gbl_portd,4
            MOVLW 0x0A
            MOVWF delay_10us_00000_arg_del
            CALL delay_10us_00000
            RETURN
; } FCD_LCDDisplay0_RawSend function end

            ORG 0x000000A4
FCD_SWITCH_0005A
; { FCD_SWITCH2_ReadState ; function begin
            BSF STATUS, RP0
            BCF STATUS, RP1
```

```
            BSF gbl_trisb,6
            MOVLW 0x02
            BCF STATUS, RP0
            MOVWF delay_us_00000_arg_del
            CALL delay_us_00000
            BTFSS gbl_portb,6
            GOTO     label9
            MOVLW 0x01
            MOVWF FCD_SWITCH_0005A_1_switchval
            GOTO     label10
label9
            CLRF FCD_SWITCH_0005A_1_switchval
label10
            MOVF FCD_SWITCH_0005A_1_switchval, W
            MOVWF CompTempVarRet2236
            RETURN
; } FCD_SWITCH2_ReadState function end

            ORG 0x000000B4
FCD_SWITCH1_00057
; { FCD_SWITCH1_ReadState ; function begin
            BSF STATUS, RP0
            BCF STATUS, RP1
            BSF gbl_trisb,5
            MOVLW 0x02
            BCF STATUS, RP0
            MOVWF delay_us_00000_arg_del
            CALL delay_us_00000
            BTFSS gbl_portb,5
            GOTO     label11
            MOVLW 0x01
            MOVWF FCD_SWITCH_00057_1_switchval
            GOTO     label12
label11
            CLRF FCD_SWITCH_00057_1_switchval
label12
            MOVF FCD_SWITCH_00057_1_switchval, W
            MOVWF CompTempVarRet2235
            RETURN
; } FCD_SWITCH1_ReadState function end

            ORG 0x000000C4
FCD_SWITCH_00054
; { FCD_SWITCH0_ReadState ; function begin
            BSF STATUS, RP0
            BCF STATUS, RP1
            BSF gbl_trisb,4
            MOVLW 0x02
            BCF STATUS, RP0
            MOVWF delay_us_00000_arg_del
            CALL delay_us_00000
            BTFSS gbl_portb,4
            GOTO     label13
            MOVLW 0x01
            MOVWF FCD_SWITCH_00054_1_switchval
            GOTO     label14
label13
            CLRF FCD_SWITCH_00054_1_switchval
label14
            MOVF FCD_SWITCH_00054_1_switchval, W
            MOVWF CompTempVarRet2234
            RETURN
; } FCD_SWITCH0_ReadState function end

            ORG 0x000000D4
FCD_LCDDis_0004F
; { FCD_LCDDisplay0_PrintString ; function begin
            CLRF FCD_LCDDis_0004F_1_idx
            CLRF FCD_LCDDis_0004F_1_idx
label15
```

```
            MOVF FCD_LCDDis_0004F_arg_MSZ_String, W
            SUBWF FCD_LCDDis_0004F_1_idx, W
            BTFSC STATUS,C
            RETURN
            BCF STATUS,IRP
            BTFSC FCD_LCDDis_0004F_arg_String+D'1',0
            BSF STATUS,IRP
            MOVF FCD_LCDDis_0004F_arg_String, W
            ADDWF FCD_LCDDis_0004F_1_idx, W
            MOVWF FSR
            MOVF INDF, F
            BTFSC STATUS,Z
            RETURN
            BCF STATUS,IRP
            BTFSC FCD_LCDDis_0004F_arg_String+D'1',0
            BSF STATUS,IRP
            MOVF FCD_LCDDis_0004F_arg_String, W
            ADDWF FCD_LCDDis_0004F_1_idx, W
            MOVWF FSR
            MOVF INDF, W
            MOVWF FCD_LCDDis_00048_arg_in
            MOVLW 0x10
            MOVWF FCD_LCDDis_00048_arg_mask
            CALL FCD_LCDDis_00048
            INCF FCD_LCDDis_0004F_1_idx, F
            GOTO     label15
; } FCD_LCDDisplay0_PrintString function end

            ORG 0x000000F0
FCD_LCDDis_0004E
; { FCD_LCDDisplay0_PrintNumber ; function begin
            BTFSS FCD_LCDDis_0004E_arg_Number+D'1',7
            GOTO     label16
            MOVLW 0x2D
            MOVWF FCD_LCDDis_00048_arg_in
            MOVLW 0x10
            MOVWF FCD_LCDDis_00048_arg_mask
            CALL FCD_LCDDis_00048
            MOVF FCD_LCDDis_0004E_arg_Number, W
            SUBLW 0x00
            MOVWF CompTempVar2206
            COMF FCD_LCDDis_0004E_arg_Number+D'1', F
            BTFSC STATUS,C
            INCF FCD_LCDDis_0004E_arg_Number+D'1', F
            MOVF CompTempVar2206, W
            MOVWF FCD_LCDDis_0004E_arg_Number
label16
            MOVF FCD_LCDDis_0004E_arg_Number, W
            MOVWF FCD_LCDDis_0004E_1_tmp_int
            MOVF FCD_LCDDis_0004E_arg_Number+D'1', W
            MOVWF FCD_LCDDis_0004E_1_tmp_int+D'1'
            MOVLW 0x27
            SUBWF FCD_LCDDis_0004E_arg_Number+D'1', W
            BTFSS STATUS,Z
            GOTO     label17
            MOVLW 0x10
            SUBWF FCD_LCDDis_0004E_arg_Number, W
label17
            BTFSS STATUS,C
            GOTO     label21
            BTFSC FCD_LCDDis_0004E_arg_Number+D'1',7
            GOTO     label21
            CLRF CompTempVar2209
            MOVF FCD_LCDDis_0004E_1_tmp_int, W
            MOVWF __div_16_1_00003_arg_a
            MOVF FCD_LCDDis_0004E_1_tmp_int+D'1', W
            MOVWF __div_16_1_00003_arg_a+D'1'
            BTFSS FCD_LCDDis_0004E_1_tmp_int+D'1',7
            GOTO     label18
            COMF __div_16_1_00003_arg_a, F
```

```
        COMF __div_16_1_00003_arg_a+D'1', F
        INCF __div_16_1_00003_arg_a, F
        BTFSC STATUS,Z
        INCF __div_16_1_00003_arg_a+D'1', F
        INCF CompTempVar2209, F
label118
        MOVLW 0x10
        MOVWF __div_16_1_00003_arg_b
        MOVLW 0x27
        MOVWF __div_16_1_00003_arg_b+D'1'
        CALL __div_16_1_00003
        MOVF CompTempVarRet243, W
        MOVWF FCD_LCDDis_0004E_1_tmp_byte
        BTFSS CompTempVar2209,0
        GOTO label119
        COMF FCD_LCDDis_0004E_1_tmp_byte, F
        INCF FCD_LCDDis_0004E_1_tmp_byte, F
label119
        MOVF FCD_LCDDis_0004E_1_tmp_byte, W
        ADDLW 0x30
        MOVWF FCD_LCDDis_00048_arg_in
        MOVLW 0x10
        MOVWF FCD_LCDDis_00048_arg_mask
        CALL FCD_LCDDis_00048
label120
        MOVF FCD_LCDDis_0004E_1_tmp_byte, W
        SUBLW 0x00
        BTFSC STATUS,C
        GOTO label121
        MOVLW 0x10
        SUBWF FCD_LCDDis_0004E_1_tmp_int, F
        MOVLW 0x27
        BTFSS STATUS,C
        MOVLW 0x28
        SUBWF FCD_LCDDis_0004E_1_tmp_int+D'1', F
        DECF FCD_LCDDis_0004E_1_tmp_byte, F
        GOTO label120
label121
        MOVLW 0x03
        SUBWF FCD_LCDDis_0004E_arg_Number+D'1', W
        BTFSS STATUS,Z
        GOTO label122
        MOVLW 0xE8
        SUBWF FCD_LCDDis_0004E_arg_Number, W
label122
        BTFSS STATUS,C
        GOTO label126
        BTFSC FCD_LCDDis_0004E_arg_Number+D'1',7
        GOTO label126
        CLRF CompTempVar2213
        MOVF FCD_LCDDis_0004E_1_tmp_int, W
        MOVWF __div_16_1_00003_arg_a
        MOVF FCD_LCDDis_0004E_1_tmp_int+D'1', W
        MOVWF __div_16_1_00003_arg_a+D'1'
        BTFSS FCD_LCDDis_0004E_1_tmp_int+D'1',7
        GOTO label123
        COMF __div_16_1_00003_arg_a, F
        COMF __div_16_1_00003_arg_a+D'1', F
        INCF __div_16_1_00003_arg_a, F
        BTFSC STATUS,Z
        INCF __div_16_1_00003_arg_a+D'1', F
        INCF CompTempVar2213, F
label123
        MOVLW 0xE8
        MOVWF __div_16_1_00003_arg_b
        MOVLW 0x03
        MOVWF __div_16_1_00003_arg_b+D'1'
        CALL __div_16_1_00003
        MOVF CompTempVarRet243, W
        MOVWF CompTempVar2212
```

```
        BTFSS CompTempVar2213,0
        GOTO    label24
        COMF CompTempVar2212, F
        INCF CompTempVar2212, F
label24
        MOVF CompTempVar2212, W
        MOVWF FCD_LCDDis_0004E_1_tmp_byte
        MOVF FCD_LCDDis_0004E_1_tmp_byte, W
        ADDLW 0x30
        MOVWF FCD_LCDDis_00048_arg_in
        MOVLW 0x10
        MOVWF FCD_LCDDis_00048_arg_mask
        CALL FCD_LCDDis_00048
label25
        MOVF FCD_LCDDis_0004E_1_tmp_byte, W
        SUBLW 0x00
        BTFSC STATUS,C
        GOTO    label26
        MOVLW 0xE8
        SUBWF FCD_LCDDis_0004E_1_tmp_int, F
        MOVLW 0x03
        BTFSS STATUS,C
        MOVLW 0x04
        SUBWF FCD_LCDDis_0004E_1_tmp_int+D'1', F
        DECF FCD_LCDDis_0004E_1_tmp_byte, F
        GOTO    label25
label26
        MOVF FCD_LCDDis_0004E_arg_Number+D'1', W
        XORLW 0x80
        MOVWF CompTempVar2216
        MOVLW 0x80
        SUBWF CompTempVar2216, W
        BTFSS STATUS,Z
        GOTO    label27
        MOVLW 0x64
        SUBWF FCD_LCDDis_0004E_arg_Number, W
label27
        BTFSS STATUS,C
        GOTO    label31
        CLRF CompTempVar2218
        MOVF FCD_LCDDis_0004E_1_tmp_int, W
        MOVWF __div_16_1_00003_arg_a
        MOVF FCD_LCDDis_0004E_1_tmp_int+D'1', W
        MOVWF __div_16_1_00003_arg_a+D'1'
        BTFSS FCD_LCDDis_0004E_1_tmp_int+D'1',7
        GOTO    label28
        COMF __div_16_1_00003_arg_a, F
        COMF __div_16_1_00003_arg_a+D'1', F
        INCF __div_16_1_00003_arg_a, F
        BTFSC STATUS,Z
        INCF __div_16_1_00003_arg_a+D'1', F
        INCF CompTempVar2218, F
label28
        MOVLW 0x64
        MOVWF __div_16_1_00003_arg_b
        CLRF __div_16_1_00003_arg_b+D'1'
        CALL __div_16_1_00003
        MOVF CompTempVarRet243, W
        MOVWF CompTempVar2217
        BTFSS CompTempVar2218,0
        GOTO    label29
        COMF CompTempVar2217, F
        INCF CompTempVar2217, F
label29
        MOVF CompTempVar2217, W
        MOVWF FCD_LCDDis_0004E_1_tmp_byte
        MOVF FCD_LCDDis_0004E_1_tmp_byte, W
        ADDLW 0x30
        MOVWF FCD_LCDDis_00048_arg_in
        MOVLW 0x10
```

```
        MOVWF FCD_LCDDis_00048_arg_mask
        CALL FCD_LCDDis_00048
label30
        MOVF FCD_LCDDis_0004E_1_tmp_byte, W
        SUBLW 0x00
        BTFSC STATUS,C
        GOTO    label31
        MOVLW 0x64
        SUBWF FCD_LCDDis_0004E_1_tmp_int, F
        MOVF FCD_LCDDis_0004E_1_tmp_int+D'1', F
        BTFSS STATUS,C
        DECF FCD_LCDDis_0004E_1_tmp_int+D'1', F
        DECF FCD_LCDDis_0004E_1_tmp_byte, F
        GOTO    label30
label31
        MOVF FCD_LCDDis_0004E_arg_Number+D'1', W
        XORLW 0x80
        MOVWF CompTempVar2221
        MOVLW 0x80
        SUBWF CompTempVar2221, W
        BTFSS STATUS,Z
        GOTO    label32
        MOVLW 0x0A
        SUBWF FCD_LCDDis_0004E_arg_Number, W
label32
        BTFSS STATUS,C
        GOTO    label36
        CLRF CompTempVar2223
        MOVF FCD_LCDDis_0004E_1_tmp_int, W
        MOVWF __div_16_1_00003_arg_a
        MOVF FCD_LCDDis_0004E_1_tmp_int+D'1', W
        MOVWF __div_16_1_00003_arg_a+D'1'
        BTFSS FCD_LCDDis_0004E_1_tmp_int+D'1',7
        GOTO    label33
        COMF __div_16_1_00003_arg_a, F
        COMF __div_16_1_00003_arg_a+D'1', F
        INCF __div_16_1_00003_arg_a, F
        BTFSC STATUS,Z
        INCF __div_16_1_00003_arg_a+D'1', F
        INCF CompTempVar2223, F
label33
        MOVLW 0x0A
        MOVWF __div_16_1_00003_arg_b
        CLRF __div_16_1_00003_arg_b+D'1'
        CALL __div_16_1_00003
        MOVF CompTempVarRet243, W
        MOVWF CompTempVar2222
        BTFSS CompTempVar2223,0
        GOTO    label34
        COMF CompTempVar2222, F
        INCF CompTempVar2222, F
label34
        MOVF CompTempVar2222, W
        MOVWF FCD_LCDDis_0004E_1_tmp_byte
        MOVF FCD_LCDDis_0004E_1_tmp_byte, W
        ADDLW 0x30
        MOVWF FCD_LCDDis_00048_arg_in
        MOVLW 0x10
        MOVWF FCD_LCDDis_00048_arg_mask
        CALL FCD_LCDDis_00048
label35
        MOVF FCD_LCDDis_0004E_1_tmp_byte, W
        SUBLW 0x00
        BTFSC STATUS,C
        GOTO    label36
        MOVLW 0x0A
        SUBWF FCD_LCDDis_0004E_1_tmp_int, F
        MOVF FCD_LCDDis_0004E_1_tmp_int+D'1', F
        BTFSS STATUS,C
        DECF FCD_LCDDis_0004E_1_tmp_int+D'1', F
```

```
        DECF FCD_LCDDis_0004E_1_tmp_byte, F
        GOTO    label35
label36
        MOVF FCD_LCDDis_0004E_1_tmp_int, W
        ADDLW 0x30
        MOVWF FCD_LCDDis_00048_arg_in
        MOVLW 0x10
        MOVWF FCD_LCDDis_00048_arg_mask
        CALL FCD_LCDDis_00048
        RETURN
; } FCD_LCDDisplay0_PrintNumber function end

        ORG 0x000001DE
FCD_LCDDis_0004D
; { FCD_LCDDisplay0_Cursor ; function begin
        MOVF FCD_LCDDis_0004D_arg_y, F
        BTFSS STATUS,Z
        GOTO    label37
        MOVLW 0x80
        MOVWF FCD_LCDDis_0004D_arg_y
        GOTO    label38
label37
        MOVLW 0xC0
        MOVWF FCD_LCDDis_0004D_arg_y
label38
        MOVF FCD_LCDDis_0004D_arg_x, W
        ADDWF FCD_LCDDis_0004D_arg_y, W
        MOVWF FCD_LCDDis_00048_arg_in
        CLRF FCD_LCDDis_00048_arg_mask
        CALL FCD_LCDDis_00048
        MOVLW 0x02
        MOVWF Wdt_Delay__00047_arg_delay
        CLRF Wdt_Delay__00047_arg_delay+D'1'
        CALL Wdt_Delay__00047
        RETURN
; } FCD_LCDDisplay0_Cursor function end

        ORG 0x000001F0
FCD_LCDDis_00049
; { FCD_LCDDisplay0_Start ; function begin
        BSF STATUS, RP0
        BCF STATUS, RP1
        BCF gbl_trisd,0
        BCF gbl_trisd,1
        BCF gbl_trisd,2
        BCF gbl_trisd,3
        BCF gbl_trisd,5
        BCF gbl_trisd,4
        MOVLW 0x0C
        BCF STATUS, RP0
        MOVWF Wdt_Delay__00047_arg_delay
        CLRF Wdt_Delay__00047_arg_delay+D'1'
        CALL Wdt_Delay__00047
        MOVLW 0x33
        MOVWF FCD_LCDDis_00048_arg_in
        CLRF FCD_LCDDis_00048_arg_mask
        CALL FCD_LCDDis_00048
        MOVLW 0x02
        MOVWF Wdt_Delay__00047_arg_delay
        CLRF Wdt_Delay__00047_arg_delay+D'1'
        CALL Wdt_Delay__00047
        MOVLW 0x33
        MOVWF FCD_LCDDis_00048_arg_in
        CLRF FCD_LCDDis_00048_arg_mask
        CALL FCD_LCDDis_00048
        MOVLW 0x02
        MOVWF Wdt_Delay__00047_arg_delay
        CLRF Wdt_Delay__00047_arg_delay+D'1'
        CALL Wdt_Delay__00047
        MOVLW 0x32
```

```
        MOVWF FCD_LCDDis_00048_arg_in
        CLRF FCD_LCDDis_00048_arg_mask
        CALL FCD_LCDDis_00048
        MOVLW 0x02
        MOVWF Wdt_Delay__00047_arg_delay
        CLRF Wdt_Delay__00047_arg_delay+D'1'
        CALL Wdt_Delay__00047
        MOVLW 0x2C
        MOVWF FCD_LCDDis_00048_arg_in
        CLRF FCD_LCDDis_00048_arg_mask
        CALL FCD_LCDDis_00048
        MOVLW 0x02
        MOVWF Wdt_Delay__00047_arg_delay
        CLRF Wdt_Delay__00047_arg_delay+D'1'
        CALL Wdt_Delay__00047
        MOVLW 0x06
        MOVWF FCD_LCDDis_00048_arg_in
        CLRF FCD_LCDDis_00048_arg_mask
        CALL FCD_LCDDis_00048
        MOVLW 0x02
        MOVWF Wdt_Delay__00047_arg_delay
        CLRF Wdt_Delay__00047_arg_delay+D'1'
        CALL Wdt_Delay__00047
        MOVLW 0x0C
        MOVWF FCD_LCDDis_00048_arg_in
        CLRF FCD_LCDDis_00048_arg_mask
        CALL FCD_LCDDis_00048
        MOVLW 0x02
        MOVWF Wdt_Delay__00047_arg_delay
        CLRF Wdt_Delay__00047_arg_delay+D'1'
        CALL Wdt_Delay__00047
        MOVLW 0x01
        MOVWF FCD_LCDDis_00048_arg_in
        CLRF FCD_LCDDis_00048_arg_mask
        CALL FCD_LCDDis_00048
        MOVLW 0x02
        MOVWF Wdt_Delay__00047_arg_delay
        CLRF Wdt_Delay__00047_arg_delay+D'1'
        CALL Wdt_Delay__00047
        MOVLW 0x02
        MOVWF FCD_LCDDis_00048_arg_in
        CLRF FCD_LCDDis_00048_arg_mask
        CALL FCD_LCDDis_00048
        MOVLW 0x02
        MOVWF Wdt_Delay__00047_arg_delay
        CLRF Wdt_Delay__00047_arg_delay+D'1'
        CALL Wdt_Delay__00047
        RETURN
; } FCD_LCDDisplay0_Start function end

        ORG 0x0000023E
main
; { main ; function begin
        MOVLW 0x07
        BSF STATUS, RP0
        BCF STATUS, RP1
        MOVWF gbl_adcon1
        MOVLW 0xC0
        MOVWF gbl_option_reg
        BCF STATUS, RP0
        CLRF gbl_FCV_X
        CALL FCD_LCDDis_00049
label39
        CALL FCD_SWITCH_00054
        MOVF CompTempVarRet2234, W
        MOVWF gbl_FCV_G1
        CALL FCD_SWITCH_00057
        MOVF CompTempVarRet2235, W
        MOVWF gbl_FCV_G2
        CALL FCD_SWITCH_0005A
```

```
MOVF CompTempVarRet2236, W
MOVWF gbl_FCV_G3
DECF gbl_FCV_G1, W
BTFSS STATUS,Z
GOTO     label41
CLRF FCD_LCDDis_0004D_arg_x
CLRF FCD_LCDDis_0004D_arg_y
CALL FCD_LCDDis_0004D
MOVLW 0x20
MOVWF CompTempVar2237+D'7'
MOVWF CompTempVar2237+D'10'
MOVLW 0x31
MOVWF CompTempVar2237+D'11'
MOVLW 0x47
MOVWF CompTempVar2237
MOVLW 0x4E
MOVWF CompTempVar2237+D'8'
MOVLW 0x63
MOVWF CompTempVar2237+D'3'
MOVLW 0x65
MOVWF CompTempVar2237+D'5'
MOVLW 0x68
MOVWF CompTempVar2237+D'4'
MOVLW 0x69
MOVWF CompTempVar2237+D'2'
MOVLW 0x74
MOVWF CompTempVar2237+D'6'
MOVLW 0x75
MOVWF CompTempVar2237+D'1'
MOVLW 0xB0
MOVWF CompTempVar2237+D'9'
CLRF CompTempVar2237+D'12'
MOVLW HIGH(CompTempVar2237+D'0')
MOVWF FCD_LCDDis_0004F_arg_String+D'1'
MOVLW LOW(CompTempVar2237+D'0')
MOVWF FCD_LCDDis_0004F_arg_String
MOVLW 0x0C
MOVWF FCD_LCDDis_0004F_arg_MSZ_String
CALL FCD_LCDDis_0004F
INCF gbl_FCV_X, W
MOVWF gbl_FCV_X
CLRF FCD_LCDDis_0004D_arg_x
MOVLW 0x01
MOVWF FCD_LCDDis_0004D_arg_y
CALL FCD_LCDDis_0004D
MOVLW 0x20
MOVWF CompTempVar2239+D'6'
MOVWF CompTempVar2239+D'10'
MOVLW 0x3A
MOVWF CompTempVar2239+D'9'
MOVLW 0x43
MOVWF CompTempVar2239
MOVLW 0x4E
MOVWF CompTempVar2239+D'7'
MOVLW 0x65
MOVWF CompTempVar2239+D'3'
MOVLW 0x69
MOVWF CompTempVar2239+D'2'
MOVLW 0x6C
MOVWF CompTempVar2239+D'1'
MOVLW 0x6E
MOVWF CompTempVar2239+D'4'
MOVLW 0x74
MOVWF CompTempVar2239+D'5'
MOVLW 0xB0
MOVWF CompTempVar2239+D'8'
CLRF CompTempVar2239+D'11'
MOVLW HIGH(CompTempVar2239+D'0')
MOVWF FCD_LCDDis_0004F_arg_String+D'1'
MOVLW LOW(CompTempVar2239+D'0')
```

```
        MOVWF FCD_LCDDis_0004F_arg_String
        MOVLW 0x0B
        MOVWF FCD_LCDDis_0004F_arg_MSZ_String
        CALL FCD_LCDDis_0004F
        MOVF gbl_FCV_X, W
        MOVWF FCD_LCDDis_0004E_arg_Number
        CLRF FCD_LCDDis_0004E_arg_Number+D'1'
        CALL FCD_LCDDis_0004E
label40
        CALL FCD_SWITCH_00054
        MOVF CompTempVarRet2234, W
        MOVWF gbl_FCV_G1
        DECF gbl_FCV_G1, W
        BTFSC STATUS,Z
        GOTO    label40
        GOTO    label39
label41
        DECF gbl_FCV_G2, W
        BTFSS STATUS,Z
        GOTO    label43
        CLRF FCD_LCDDis_0004D_arg_x
        CLRF FCD_LCDDis_0004D_arg_y
        CALL FCD_LCDDis_0004D
        MOVLW 0x20
        MOVWF CompTempVar2241+D'7'
        MOVWF CompTempVar2241+D'10'
        MOVLW 0x32
        MOVWF CompTempVar2241+D'11'
        MOVLW 0x47
        MOVWF CompTempVar2241
        MOVLW 0x4E
        MOVWF CompTempVar2241+D'8'
        MOVLW 0x63
        MOVWF CompTempVar2241+D'3'
        MOVLW 0x65
        MOVWF CompTempVar2241+D'5'
        MOVLW 0x68
        MOVWF CompTempVar2241+D'4'
        MOVLW 0x69
        MOVWF CompTempVar2241+D'2'
        MOVLW 0x74
        MOVWF CompTempVar2241+D'6'
        MOVLW 0x75
        MOVWF CompTempVar2241+D'1'
        MOVLW 0xB0
        MOVWF CompTempVar2241+D'9'
        CLRF CompTempVar2241+D'12'
        MOVLW HIGH(CompTempVar2241+D'0')
        MOVWF FCD_LCDDis_0004F_arg_String+D'1'
        MOVLW LOW(CompTempVar2241+D'0')
        MOVWF FCD_LCDDis_0004F_arg_String
        MOVLW 0x0C
        MOVWF FCD_LCDDis_0004F_arg_MSZ_String
        CALL FCD_LCDDis_0004F
        INCF gbl_FCV_X, W
        MOVWF gbl_FCV_X
        CLRF FCD_LCDDis_0004D_arg_x
        MOVLW 0x01
        MOVWF FCD_LCDDis_0004D_arg_y
        CALL FCD_LCDDis_0004D
        MOVLW 0x20
        MOVWF CompTempVar2243+D'6'
        MOVWF CompTempVar2243+D'10'
        MOVLW 0x3A
        MOVWF CompTempVar2243+D'9'
        MOVLW 0x43
        MOVWF CompTempVar2243
        MOVLW 0x4E
        MOVWF CompTempVar2243+D'7'
        MOVLW 0x65
```

```
        MOVWF CompTempVar2243+D'3'
        MOVLW 0x69
        MOVWF CompTempVar2243+D'2'
        MOVLW 0x6C
        MOVWF CompTempVar2243+D'1'
        MOVLW 0x6E
        MOVWF CompTempVar2243+D'4'
        MOVLW 0x74
        MOVWF CompTempVar2243+D'5'
        MOVLW 0xB0
        MOVWF CompTempVar2243+D'8'
        CLRF CompTempVar2243+D'11'
        MOVLW HIGH(CompTempVar2243+D'0')
        MOVWF FCD_LCDDis_0004F_arg_String+D'1'
        MOVLW LOW(CompTempVar2243+D'0')
        MOVWF FCD_LCDDis_0004F_arg_String
        MOVLW 0x0B
        MOVWF FCD_LCDDis_0004F_arg_MSZ_String
        CALL FCD_LCDDis_0004F
        MOVF gbl_FCV_X, W
        MOVWF FCD_LCDDis_0004E_arg_Number
        CLRF FCD_LCDDis_0004E_arg_Number+D'1'
        CALL FCD_LCDDis_0004E
label42
        CALL FCD_SWITCH_00057
        MOVF CompTempVarRet2235, W
        MOVWF gbl_FCV_G2
        DECF gbl_FCV_G2, W
        BTFSC STATUS,Z
        GOTO    label42
        GOTO    label39
label43
        DECF gbl_FCV_G3, W
        BTFSS STATUS,Z
        GOTO    label39
        CLRF FCD_LCDDis_0004D_arg_x
        CLRF FCD_LCDDis_0004D_arg_y
        CALL FCD_LCDDis_0004D
        MOVLW 0x20
        MOVWF CompTempVar2245+D'7'
        MOVWF CompTempVar2245+D'10'
        MOVLW 0x33
        MOVWF CompTempVar2245+D'11'
        MOVLW 0x47
        MOVWF CompTempVar2245
        MOVLW 0x4E
        MOVWF CompTempVar2245+D'8'
        MOVLW 0x63
        MOVWF CompTempVar2245+D'3'
        MOVLW 0x65
        MOVWF CompTempVar2245+D'5'
        MOVLW 0x68
        MOVWF CompTempVar2245+D'4'
        MOVLW 0x69
        MOVWF CompTempVar2245+D'2'
        MOVLW 0x74
        MOVWF CompTempVar2245+D'6'
        MOVLW 0x75
        MOVWF CompTempVar2245+D'1'
        MOVLW 0xB0
        MOVWF CompTempVar2245+D'9'
        CLRF CompTempVar2245+D'12'
        MOVLW HIGH(CompTempVar2245+D'0')
        MOVWF FCD_LCDDis_0004F_arg_String+D'1'
        MOVLW LOW(CompTempVar2245+D'0')
        MOVWF FCD_LCDDis_0004F_arg_String
        MOVLW 0x0C
        MOVWF FCD_LCDDis_0004F_arg_MSZ_String
        CALL FCD_LCDDis_0004F
        INCF gbl_FCV_X, W
```

```
        MOVWF gbl_FCV_X
        CLRF FCD_LCDDis_0004D_arg_x
        MOVLW 0x01
        MOVWF FCD_LCDDis_0004D_arg_y
        CALL FCD_LCDDis_0004D
        MOVLW 0x20
        MOVWF CompTempVar2247+D'6'
        MOVWF CompTempVar2247+D'10'
        MOVLW 0x3A
        MOVWF CompTempVar2247+D'9'
        MOVLW 0x43
        MOVWF CompTempVar2247
        MOVLW 0x4E
        MOVWF CompTempVar2247+D'7'
        MOVLW 0x65
        MOVWF CompTempVar2247+D'3'
        MOVLW 0x69
        MOVWF CompTempVar2247+D'2'
        MOVLW 0x6C
        MOVWF CompTempVar2247+D'1'
        MOVLW 0x6E
        MOVWF CompTempVar2247+D'4'
        MOVLW 0x74
        MOVWF CompTempVar2247+D'5'
        MOVLW 0xB0
        MOVWF CompTempVar2247+D'8'
        CLRF CompTempVar2247+D'11'
        MOVLW HIGH(CompTempVar2247+D'0')
        MOVWF FCD_LCDDis_0004F_arg_String+D'1'
        MOVLW LOW(CompTempVar2247+D'0')
        MOVWF FCD_LCDDis_0004F_arg_String
        MOVLW 0x0B
        MOVWF FCD_LCDDis_0004F_arg_MSZ_String
        CALL FCD_LCDDis_0004F
        MOVF gbl_FCV_X, W
        MOVWF FCD_LCDDis_0004E_arg_Number
        CLRF FCD_LCDDis_0004E_arg_Number+D'1'
        CALL FCD_LCDDis_0004E
label44
        CALL FCD_SWITCH_0005A
        MOVF CompTempVarRet2236, W
        MOVWF gbl_FCV_G3
        DECF gbl_FCV_G3, W
        BTFSC STATUS,Z
        GOTO    label44
        GOTO    label39
; } main function end

        ORG 0x00000349
_startup
        MOVLW 0xD5
        BCF STATUS, RP0
        BCF STATUS, RP1
        MOVWF gbl_16_LSR
        MOVLW 0xC4
        MOVWF gbl_16_LSR+D'1'
        MOVLW 0xBB
        MOVWF gbl_16_LSR+D'2'
        MOVLW 0xDC
        MOVWF gbl_16_LSR+D'3'
        CLRF gbl_17_gbl_aSig
        CLRF gbl_17_gbl_aSig+D'1'
        CLRF gbl_17_gbl_aSig+D'2'
        CLRF gbl_17_gbl_aSig+D'3'
        CLRF gbl_17_gbl_bSig
        CLRF gbl_17_gbl_bSig+D'1'
        CLRF gbl_17_gbl_bSig+D'2'
        CLRF gbl_17_gbl_bSig+D'3'
        CLRF gbl_17_gbl_zSig
        CLRF gbl_17_gbl_zSig+D'1'
```

```
        CLRF gbl_17_gbl_zSig+D'2'
        CLRF gbl_17_gbl_zSig+D'3'
        CLRF gbl_17_gbl_aExp
        CLRF gbl_17_gbl_bExp
        CLRF gbl_17_gbl_zExp
        CLRF gbl_17_gbl_zExp+D'1'
        CLRF gbl_17_gbl_aSign
        CLRF gbl_17_gbl_bSign
        CLRF gbl_17_gbl_zSign
        CLRF gbl_17_gbl_zSigZero
        CLRF gbl_17_gbl_ret
        CLRF gbl_17_gbl_ret+D'1'
        CLRF gbl_17_gbl_ret+D'2'
        CLRF gbl_17_gbl_ret+D'3'
        CLRF gbl_float_rounding_mode
        CLRF gbl_float_exception_flags
        CLRF gbl_float_detect_tininess
        BCF PCLATH,3
        BCF PCLATH,4
        GOTO    main
        ORG 0x00000371
interrupt
; { interrupt ; function begin
        BCF STATUS, RP0
        BCF STATUS, RP1
        SWAPF Int1BContext+D'2', W
        MOVWF FSR
        SWAPF Int1BContext+D'1', W
        MOVWF PCLATH
        SWAPF Int1BContext, W
        MOVWF STATUS
        SWAPF Int1Context, F
        SWAPF Int1Context, W
        RETFIE
; } interrupt function end

        ORG 0x00002007
        DW 0x3F39
        END
```