



IUP GEII
OPTION AISHM

RAPPORT DE STAGE

Thème



OUTIL D'AIDE A LA COMMUNICATION POUR LES PERSONNES HANDICAPEES

Encadré par : - Mr. Thierry POULAIN

Réalisé par : - [A.OULBOUB](#)

Année universitaire : 2003 / 2004

SOMMAIRE

REMERCIEMENTS.....	3
RÉALISATION.....	4
I. INTRODUCTION.....	5
II. OBJECTIFS.....	6
III. ARCHITECTURE FONCTIONNELLE.....	7
IV. ARCHITECTURE MATERIELLE.....	10
IV-1. Choix du micro Processeur.....	10
IV-2. Choix de l’Afficheur.....	12
IV-3. Schéma de câblage.....	18
V. ARCHITECTURE LOGICIELLE.....	20
V-1. Algorithme du programme.....	20
V-2. Choix du langage de programmation.....	26
V-3. Environnement de développement.....	26
VI. ETATS D’AVANCEMENT.....	28
CONCLUSION.....	29
BIBLIOGRAPHIE.....	30
ANNEXE 1 – Architecture & fonctionnement du PIC 16F877.....	31
I. GESTION MEMOIRE DU PIC 16F877.....	33
II. MODE D’ADRESSAGE DU PIC 16F877	34
II-1. L’ADRESSAGE DIRECT.....	34
II-2. L’ADRESSAGE INDIRECT.....	35
III. LES PRINCIPAUX REGISTRES DU PIC 16F877.....	36
III-1. LE REGISTRE “W”.....	36
III-2. LE REGISTRE “STATUS”.....	36
LES INSTRUCTIONS DU PIC 16F877.....	37
ANNEXE 2 – Description & commande d' un LCD graphique.....	38
I. PRINCIPE DE COMMANDE D’UN LCD GRAPHIQUE.....	39
II. PRINCIPE D’ECRITURE D’UNE DONNEE EN RAM.....	40
III. LCD GRAPHIQUE 128x64.....	41
IV. Le tableau des commandes.....	42
IV-1. Séquence d’envoi d’une commande.....	43
IV-2. Séquence d’envoi d’une donnée.....	44
IV-3. Organisation de la RAM.....	44
ANNEXE 3 – Description & commande d' un LCD Alphanumérique.....	45
I. Tableau des différentes commandes de l'afficheur alphanumérique.....	46
II. Description des différentes commandes.....	47
ANNEXE 4 – Présentation & utilisation du MPLAB.....	48
I. Maîtriser MPLAB.....	49
I-1. Ouverture d’un nouveau projet.....	50
I-2. Ecriture du programme source.....	51
I-3. Création du programme objet.....	51
I-4. Ouverture d’un projet existant.....	52
ANNEXE 5 – Présentation & utilisation d' IRI.....	53
ANNEXE 6 – Le programme.....	58
ANNEXE 7 – Les essais de simulation.....	59

REMERCIEMENTS

A l'issue de ce stage je remercie plus particulièrement monsieur **Thierry POULAIN** pour le sujet qu'il m'a proposé, le temps consacré ainsi que les conseils techniques prodigués.

De plus je tiens à remercier messieurs **Serge DEBERNARD**, **Mohammed ASMANI** et **Jacques DELACROIX**, pour leurs vifs conseils et leurs aides afin d'aboutir sur ce projet. Ainsi que le L.A.M.I.H. (Laboratoire d'Automatique, de Mécanique et d'Informatique industrielles et Humaines) pour m'avoir accueilli dans ses locaux.

RÉALISATION

I. INTRODUCTION

Dans le cadre de ma troisième année d'IUP GEII (génie électrique et informatique industrielle) à l'université de Valenciennes, il m'a été proposé de passer un stage au sein du L.A.M.I.H. d'une durée de six mois.

C'est pour moi l'occasion de réinvestir mes connaissances et compétences acquises au cours de mon cursus.

Le thème retenu est l'étude et la réalisation d'un outil d'aide à la communication pour les personnes handicapées.

Dans un premier temps nous verrons l'analyse de ce sujet. Ensuite cette solution sera développée d'un point de vue matériel puis logiciel.

II. OBJECTIFS

La finalité de l'étude et de la réalisation de cet outil est de permettre aux personnes handicapées de communiquer avec d'autres personnes.

En effet, il s'agit de concevoir un système à base d'un afficheur piloté par un microcontrôleur et fournir un dictionnaire qui permettra aux personnes handicapées de saisir un texte à l'aide d'un seul bouton poussoir ou d'un capteur de mouvement.

III. ARCHITECTURE FONCTIONNELLE

D'après les objectifs que je me suis fixé, et d'un point de vue fonctionnel (voir figure 1), le rôle de cet outil de communication est de permettre à une personne handicapée de composer un texte à l'aide d'un organe de commande et de le visualiser sur deux afficheurs.

Pour assister l'handicapée, il est prévu d'intégrer à l'outil un dictionnaire, aussi qu'un algorithme de reconnaissance de mot, ce dictionnaire sera stocké dans une mémoire reliée au microprocesseur.

Un deuxième afficheur sera employé pour afficher le texte à son interlocuteur.

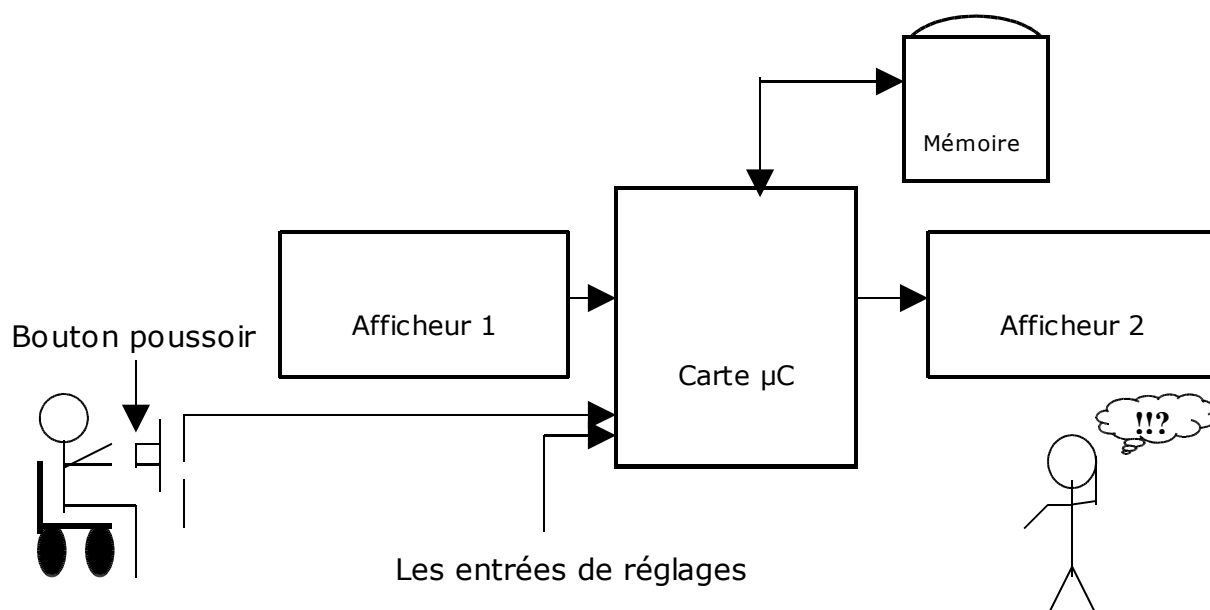


Figure1 - schéma fonctionnel du système -

Dynamique de l'interface : (voir figure 4)

La personne veut taper par exemple **Sortir**, au début on va afficher les lettres par paquets de 8 lettres (voir figure 2), donc il va attendre que le paquet qui contient la lettre **S** soit affiché puis il actionne le bouton poussoir.

Après on lui actionne le paquet lettre par lettre, dès que la lettre **S** soit actionné (voir figure 3), il doit valider par le bouton poussoir dès la lettre est validée, donc la lettre s apparaît dans la zone de texte et on lui affiche les mots qui commencent par la lettre **S** (zone de dictionnaire).

Dans ce cas il a deux choix soit de choisir les mots parmi ceux du dictionnaire si non il tape tout le mot puis on le rajoute dans le dictionnaire si ça n'existe pas.

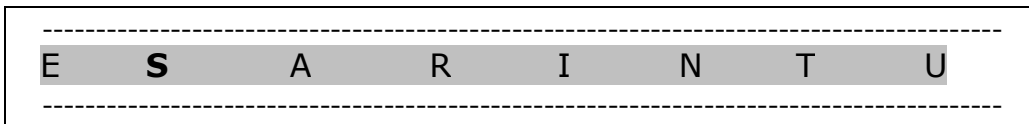


Figure 2

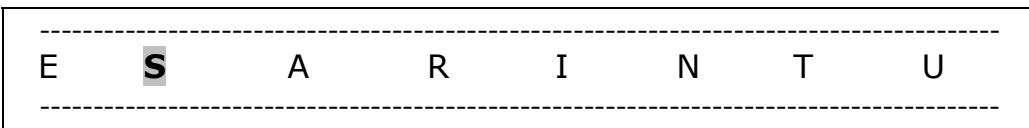


Figure 3

➤ Afficheur 1 :

C'est l'afficheur de la personne handicapée, il comprend :

- Une zone des caractères à afficher.
- Une zone pour afficher le dictionnaire.
- Une zone pour afficher le texte édité.

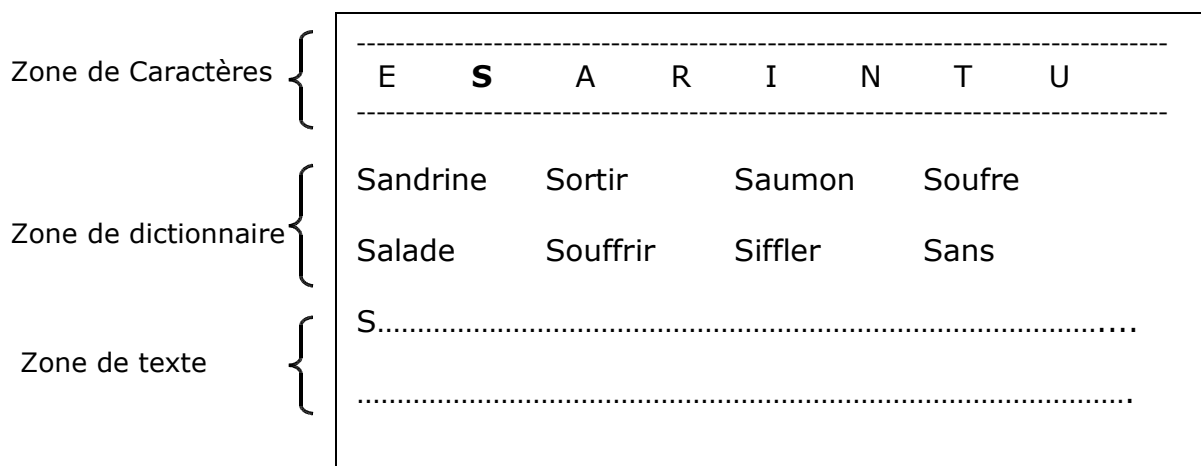
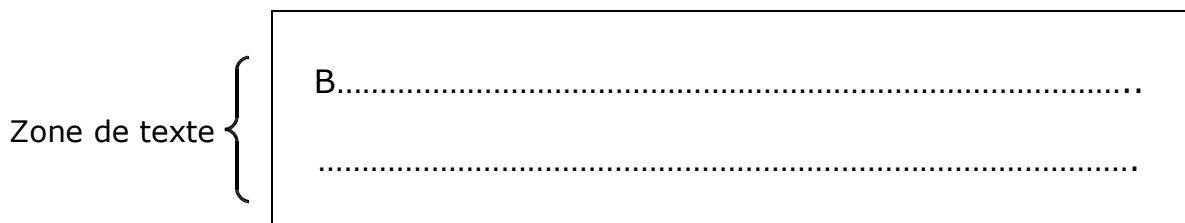


Figure 4

➤ Afficheur 2 :

C'est l'afficheur d'interlocuteur de la personne handicapée, il comprend seulement une seule zone pour afficher le texte édité.



➤ Entrées de réglages :

C'est des entrées qui permettent de déterminer :

La vitesse de défilement des paquets de caractères.
La vitesse de défilement des caractères.

➤ Bouton de poussoir :

Dans un premier temps en utilise le bouton poussoir pour les essais.
Par la suite il sera remplacer par un système adapté à l'utilisateur
(capteur de mouvement, capteur de déplacement, ...).

IV. ARCHITECTURE MATERIELLE

L'objectif de cette partie est l'étude et la réalisation d'outil de communication d'un point de vue matériel. Dans la suite de cette partie, je vais donc faire le choix de microprocesseur, ainsi que le choix des deux afficheurs.

IV-1. Choix du micro Processeur

Parmi tous les microcontrôleurs existants, j'ai choisi la famille du PIC. En effet, l'application est très simple et ne demande que peu de calcul. Par exemple, Un microcontrôleur de type 68HC11 est trop coûteux, trop puissant, trop volumineux et consomme trop de courant. J'ai donc choisi la famille des microcontrôleurs PIC.

Un Pic est parfait pour une petite application autonome, avec très peu de hardware autour, ne demandant pas une grande vitesse ni puissance de calcul. Il présente l'avantage qu'il existe en particulier chez Microchip des outils gratuits et complets pour le mettre en œuvre.

Pour le choix de PIC j'ai cherché un mini module sur internet facilement programmable et interfaçable avec les deux afficheurs. Mon choix est fait alors sur le module AE877FP (voir figure 5) disponible sur le site de microtronique.

Caractéristiques du module AE77FP :

Le module AE877FP comprend un PIC16F877 cadencé à 20 Mhz. Il comprend un ADM232 (=MAX232) pour la liaison série. Un régulateur intégré type 7805 est présent également. Tous les ports sont disponibles sur les connecteurs.

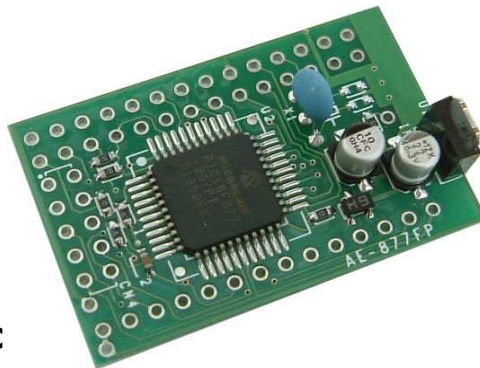


Figure 5

Description du PIC

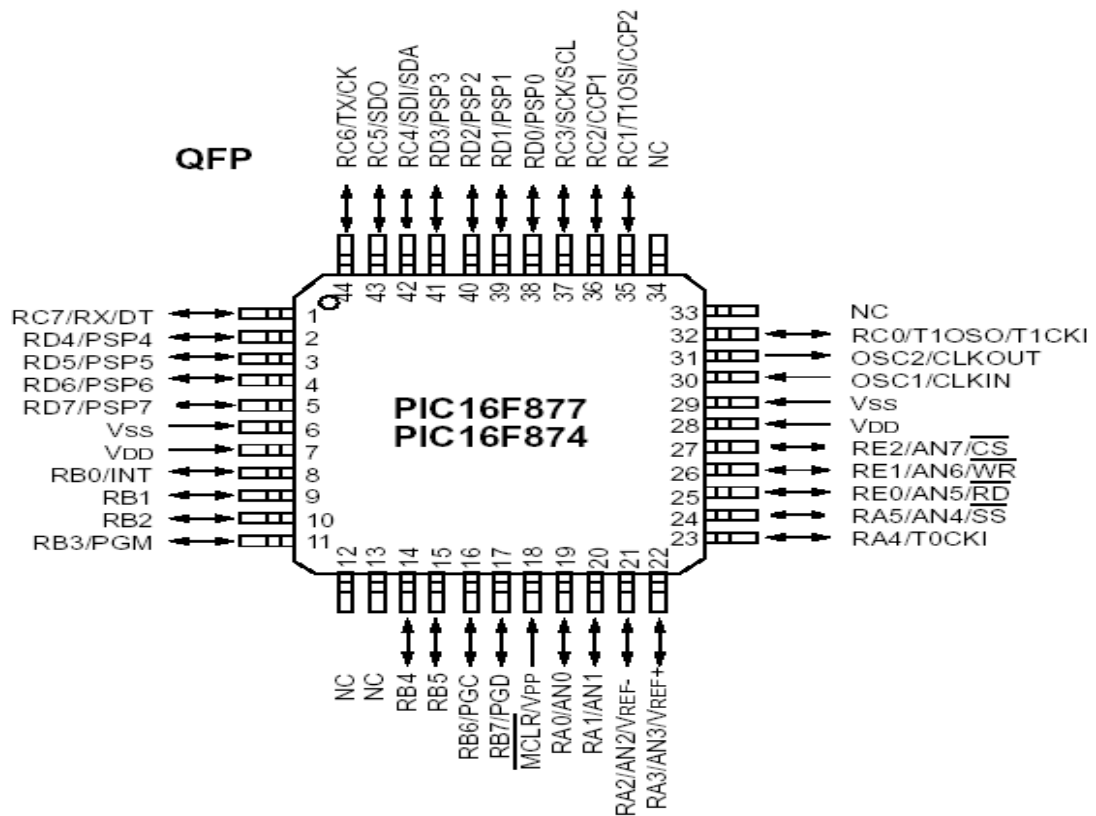


Figure 6

Le Pic 16F877 (voir figure 6) fonctionne à la fréquence maximale de 20 Mhz, ces **principales caractéristiques** sont :

	PIC 16F877
FLASH	8 Ko
RAM	368 octets
EEPROM	256
Ports d'E/S	5 (Ports A,B,C,D et E)
Convertisseurs A/N	8
Port parallèle	PSP
Port série	MSSP,USART
Sources d'interruption	14
Timers	3
Modes compare, capture, PMW	2
Jeu d'instruction	35

caractéristiques d'un PIC 16F877

IV-2. Choix de l’Afficheur

Le cahier des charges (voir chapitre III) a montré que nous avons besoin d’un minimum de 7 lignes (voir figure 7).

3 lignes pour l’affichage des caractères sélectionnables.

2 lignes pour afficher les mots du dictionnaire.

2 lignes pour visualiser le texte saisi.

Ma recherche m’a amené à choisir un afficheur graphique, dans la mesure où les afficheurs alphanumériques comprennent peu des lignes d’affichage.

Le deuxième afficheur n’a pour objectif que de visualiser à l’interlocuteur de la personne handicapée le texte saisi, de ce fait un afficheur de 2 lignes x 20 caractères suffit amplement à notre besoin

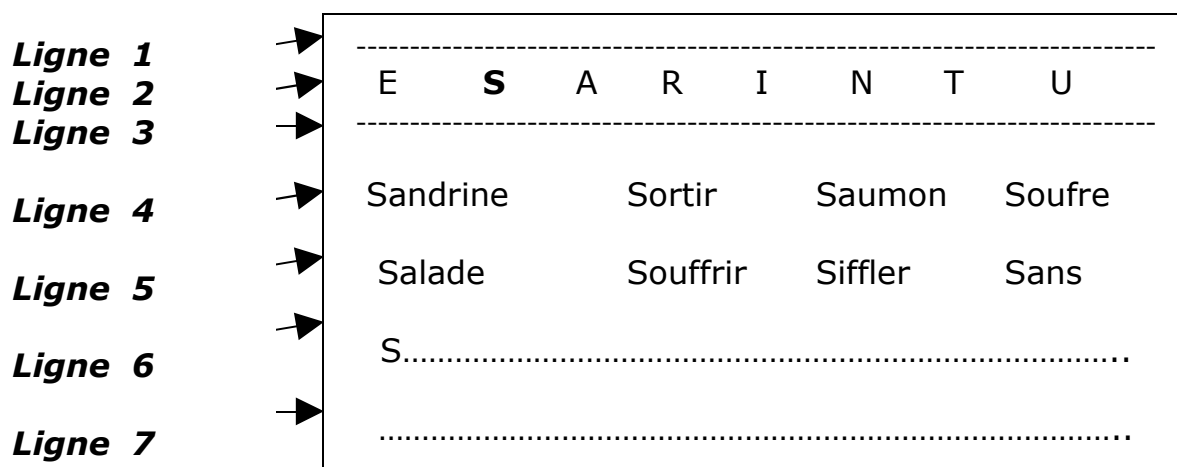


Figure 7

Caractéristique technique	Code de commande	Prix/unité (Euros)	Remarques
Affichage rétroéclairé(TFBL),16x2	329-0379	25.10	Rétroéclairage à leds
TFBL,16x2	424-7039	21.14	Rétroéclairage à leds
TFBL,20x2	329-0385	32.56	Rétroéclairage à leds
TFBL,20x2	424-7073	31.53	Rétroéclairage à leds
TFBL,20x2	214-3547	38.33	Rétroéclairage à leds
TFBL,24x2	214-3569	41.95	Rétroéclairage à leds
MDLS 16265B-SS-LV,16x2	325-1936	21.93	Afficheur à cristaux liquides
MDLS 20265-SS-LV,20x2	277-0688	34.21	Afficheur à cristaux liquides
TFBL,16x4	424-7051	35.23	
TFBL,20x4	424-7102	40.85	
MDLS 16465-LV,16x4	325-1958	51.94	
MDLS 20464-LV,20x4	277-0694	44.90	
LCD GRAPHIQUE 128x64	329-0329	53.88	Mode réflectif
LCD GRAPHIQUE 128x64	329-0335	67.31	Mode translectif rétroéclairage à leds.

Tableau de choix des deux afficheurs

L'afficheur 2 lignes x20 caractères **TFBL,20x2 (329-0385)** est le plus convenable pour mon application. Mon choix sur le deuxième LCD est le **LCD GRAPHIQUE 128x64 (329-0335)**.

AFFICHEUR	PRIX en Euros
LCD TFBL	32.56
LCD Graphique	67.31
TOTAL	99.87

IV-2-1. LCD Graphique

DEFINITION :

Un afficheur LCD graphique est un afficheur dont on peut gérer l'affichage des « pixels » c'est-à-dire des points d'écran. Chacun des pixels d'un tel écran fonctionne selon un mode ON/OFF = visible / invisible. La taille de la zone d'affichage est variable selon les modèles allant du 64 x 128 à 240x320 voire même 480 x 640. Plus c'est grand, plus c'est cher.

Le grand avantage de ces afficheurs est la liberté d'affichage qui est très grande : on peut créer des affichages de courbes, de graphes, de formes géométriques, de polices de tailles différentes, etc...

Cette grande liberté a pour contre-partie la nécessité d'écrire toutes les routines adaptées à l'afficheur utilisé, routines qui ne sont pas forcément transposables d'un afficheur à l'autre.

PRESENTATION :

Un afficheur LCD graphique comporte :

- ✓ Une zone d'affichage plus ou moins grande selon le format du LCD utilisé.
- ✓ Un connecteur de broches d'alimentation et de commande : de 16 à 20 broches selon les modèles.

Logiquement, les broches d'un afficheur graphique comportent :

- Les broches d'alimentation
- Les broches de réglage du contraste
- Les broches de données
- Les broches de communication RS, W/R et E

Plus ou moins les broches de Reset et de sélection de « bloc graphique ». Le nombre de ces différentes broches est variable selon les afficheurs et n'est pas généralisable

STRUCTURE FONCTIONNELLE :

Un afficheur LCD graphique comporte :

- un bloc LCD comportant le nombre de pixels annoncé par le fabricant, par exemple 128x64.

- un ou plusieurs **DRIVERS DE COLONNES** qui assurent chacun la gestion des colonnes (généralement 64) d'un bloc graphique.

- un ou plusieurs **DRIVERS DE LIGNES** qui assurent chacun la gestion des lignes (généralement 64) communes à plusieurs blocs graphiques.

Au total, la zone d'affichage est divisée en plusieurs « blocs graphiques ». Leur nombre vaut Nombre de **DRIVERS DE COLONNE** x Nombre de **DRIVERS DE LIGNES**

A titre d'exemple, un afficheur 128x64 possède 2 drivers de 64 colonnes et 1 driver de 64 lignes ce qui fait 2 blocs graphiques de 64 lignes x 64 colonnes.

◆ Structure fonctionnelle des drivers de colonnes

Les drivers de colonnes comportent typiquement :

- Une RAM qui contient les données de l'écran. A chaque bit de cette RAM correspond 1 pixel de l'écran. Si 1 bit est à 1, un point est affiché à l'écran. Si 1 bit est à 0, le point est alors invisible.
- Une logique de commande qui permet la communication avec un microprocesseur de commande de l'afficheur LCD.
- Un registre de données destiné à recevoir les données pour la RAM et un registre d'instructions destinées à recevoir les instructions d'affichage (ON/OFF écran, adresse en RAM, etc...)

◆ Structure fonctionnelle des drivers de lignes

Les drivers de ligne gèrent le multiplexage des lignes de l'afficheur automatiquement de façon à afficher les données contenues dans la RAM des drivers de colonnes.

Les drivers de lignes sont commandés par les drivers de colonnes.

PRINCIPE DE FONCTIONNEMENT :

Le grand principe qui sous tend le fonctionnement d'un afficheur LCD graphique est le suivant : **LE CONTENU DE LA MEMOIRE S'AFFICHE SUR L'ECRAN EN TEMPS REEL ET AUTOMATIQUEMENT.** Autrement dit, chaque bit de la mémoire d'écran (RAM) correspond à un pixel de l'écran. Si un bit de la RAM est à 0, le pixel d'écran correspondant est invisible, si un bit de la RAM est à 1, le pixel d'écran correspondant est affiché.

Donc, en fait, **gérer un afficheur graphique consiste essentiellement à écrire les données à afficher dans la RAM de l'afficheur.** Ceci va donc passer par la gestion des adresses de la RAM et de l'écriture de données en RAM. L'affichage est réalisé automatiquement par ailleurs.

Toute la difficulté de la gestion d'un afficheur LCD graphique réside dans le fait de faire correspondre l'organisation de la RAM qui n'est pas « simple » avec une représentation graphique habituelle en coordonnées du type X,Y (voir figure 8).

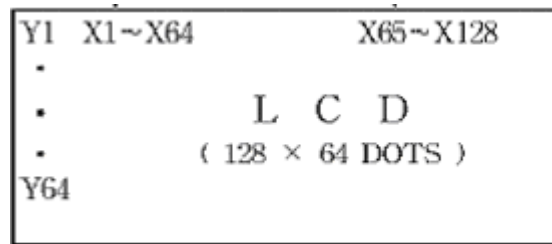


Figure 8

IV-2-2. LCD Alphanumérique 2x20

DESCRIPTION :

L'afficheur Alphanumérique 2x20 comporte 2 lignes de 20 caractères (voir figure 9) inscrits dans une matrice de 5 colonnes de 8 points. La plupart des caractères n'utilisent que les 7 rangées supérieures de la matrice; la rangée inférieure est prévue pour la visualisation d'un curseur. L'afficheur proprement dit est implanté sur un circuit imprimé au dos duquel sont soudés deux circuits intégrés VLSI et quelques composants discrets. L'électronique est compatible C-MOS et TTL et sa consommation ne dépasse pas 7 mW. Ses entrées sont protégées par des diodes. Outre les 40 caractères visualisés sur l'affichage, il est possible de mettre en mémoire 40 caractères supplémentaires, caractères que l'on visualisera sur l'affichage au moment voulu.

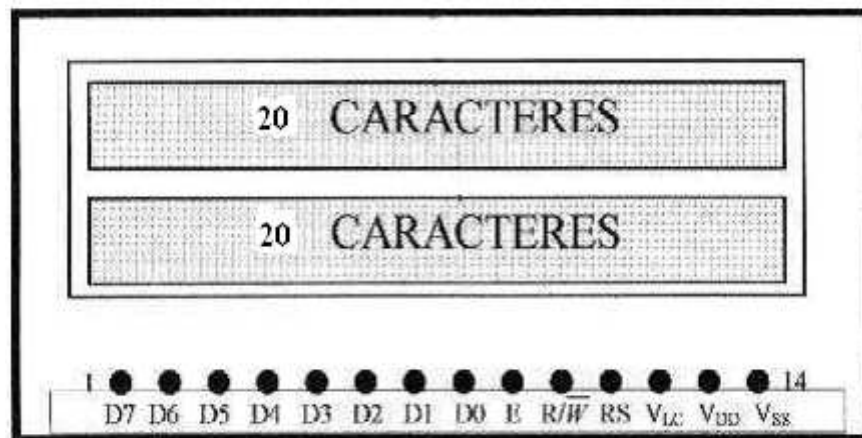


Figure 9

Possibilités de l'afficheur :

L'afficheur est en mesure de visualiser 192 caractères:

- De \$00 à \$ 07 : 8 caractères définissables par l'utilisateur.
- De \$20 à \$7F : 96 caractères ASCII (majuscules, minuscules, chiffres, signes).
- De \$A0 à \$DF: 64 caractères japonais (alphabet kana).
- De \$E0 à \$FF : 32 caractères spéciaux: accent, lettres grecques, ...

De plus, l'affichage est capable de traiter d'autres commandes telles que:

- l'extinction de l'affichage.
- le positionnement du curseur.
- le déplacement des caractères sur l'affichage.
- Le choix du caractère à redéfinir.
- le choix du sens du déplacement du curseur ou de l'affichage.
- le clignotement des caractères ou du curseur.

FONCTIONNEMENT :

◆ Tableau de codage des caractères :

Les caractères et les signes spéciaux sont codés en ASCII.

◆ Apparition des caractères sur l'afficheur :

Après avoir défini le sens de déplacement, les caractères apparaissent au dessus du curseur (qu'il soit visualisé ou non).

Adresse	gauche à droite	invisible
haut	\$00.....\$13	\$14.....\$27
bas	\$40.....\$53	\$54.....\$67

L'adresse 00 correspond à la ligne du haut à gauche, 13 à droite.

L'adresse 40 correspond à la ligne du bas à gauche, 53 à droite.

La zone invisible correspond à la mémoire de l'afficheur.(40 caractères).Lorsqu'un caractère est inscrit à l'adresse \$27, le caractère suivant apparaît à la ligne suivante.

◆ Principe de fonctionnement :

Le principe de fonctionnement est simple, pour visualiser un caractère, il suffit de le positionner sur le bus de donnée (codé en ASCII), de mettre RS au niveau haut (caractère), **R**/**W** au niveau bas (écriture), et de provoquer un front descendant sur l'entrée de validation de l'afficheur (E).

IV-3. Schéma de câblage

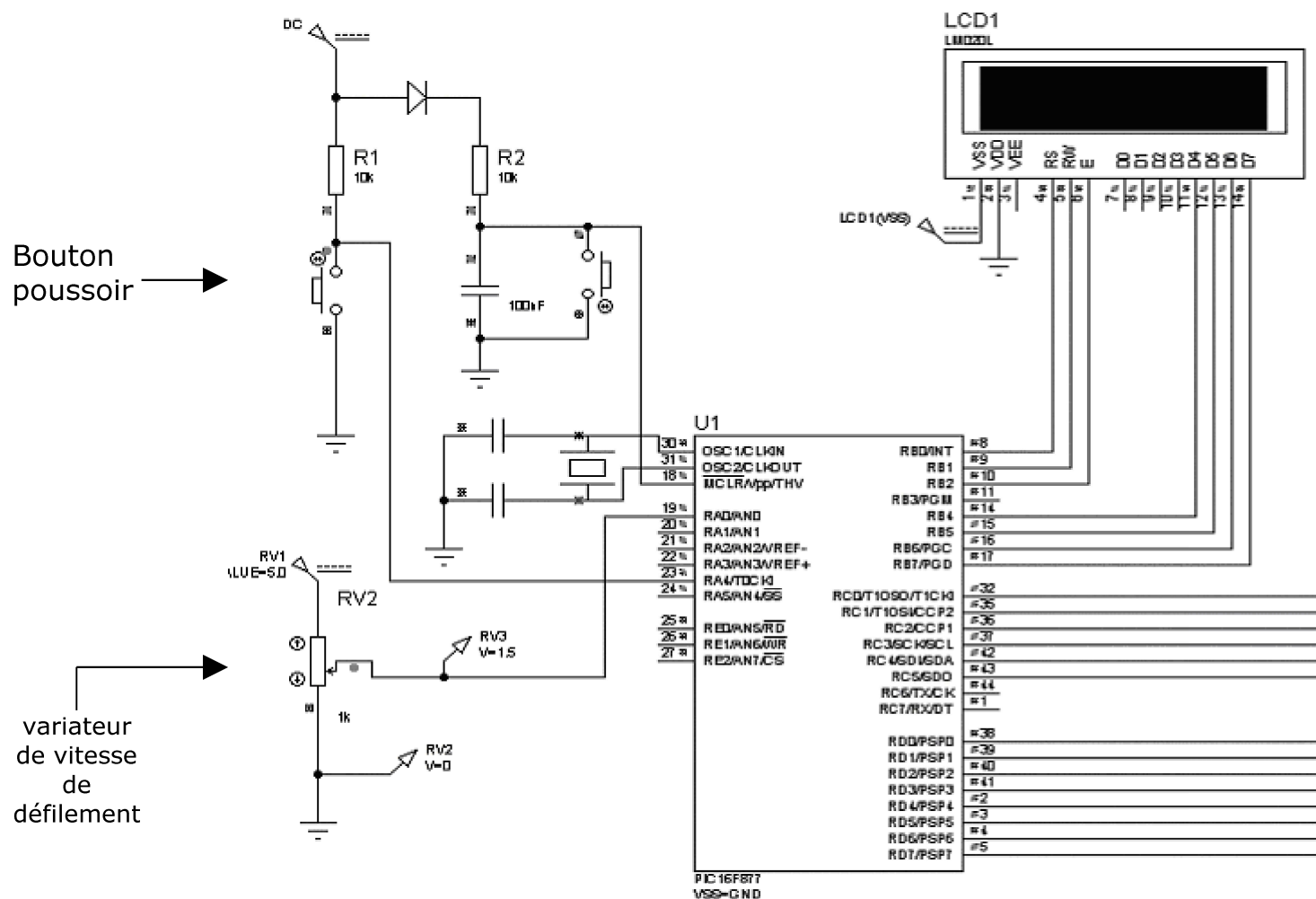


Figure 10



Analyse du schéma de câblage

Ce schéma de câblage est un développement de schéma fonctionnel du système (figure 1) au sens électronique.

La commande du LCD graphique par le PIC nécessite 14 bits :

6 bits pour la commande (Port C).

8 bits pour les données (Port D).

La commande du LCD alphanumérique par le PIC nécessite 7 bits (Port B).

D'une part si on veut rajouter une EEPROM, cette dernière ne prendra que 2 bits (6 et 7 du Port C qui nous permet de faire une liaison série avec le PIC) ,donc le port B reste libre.

D'une autre part, au niveau de programmation c'est mieux de faire une liaison en parallèle séparée qu'une commune.

Pour le quartz on peut utiliser plusieurs cadence de 4 à 20 Mhz sauf qu'il faut recalculer la tempo au niveau du programme, car cette tempo est valable pour un quartz de 20 Mhz.



V. ARCHITECTURE LOGICIELLE

V-1. Algorithme du programme

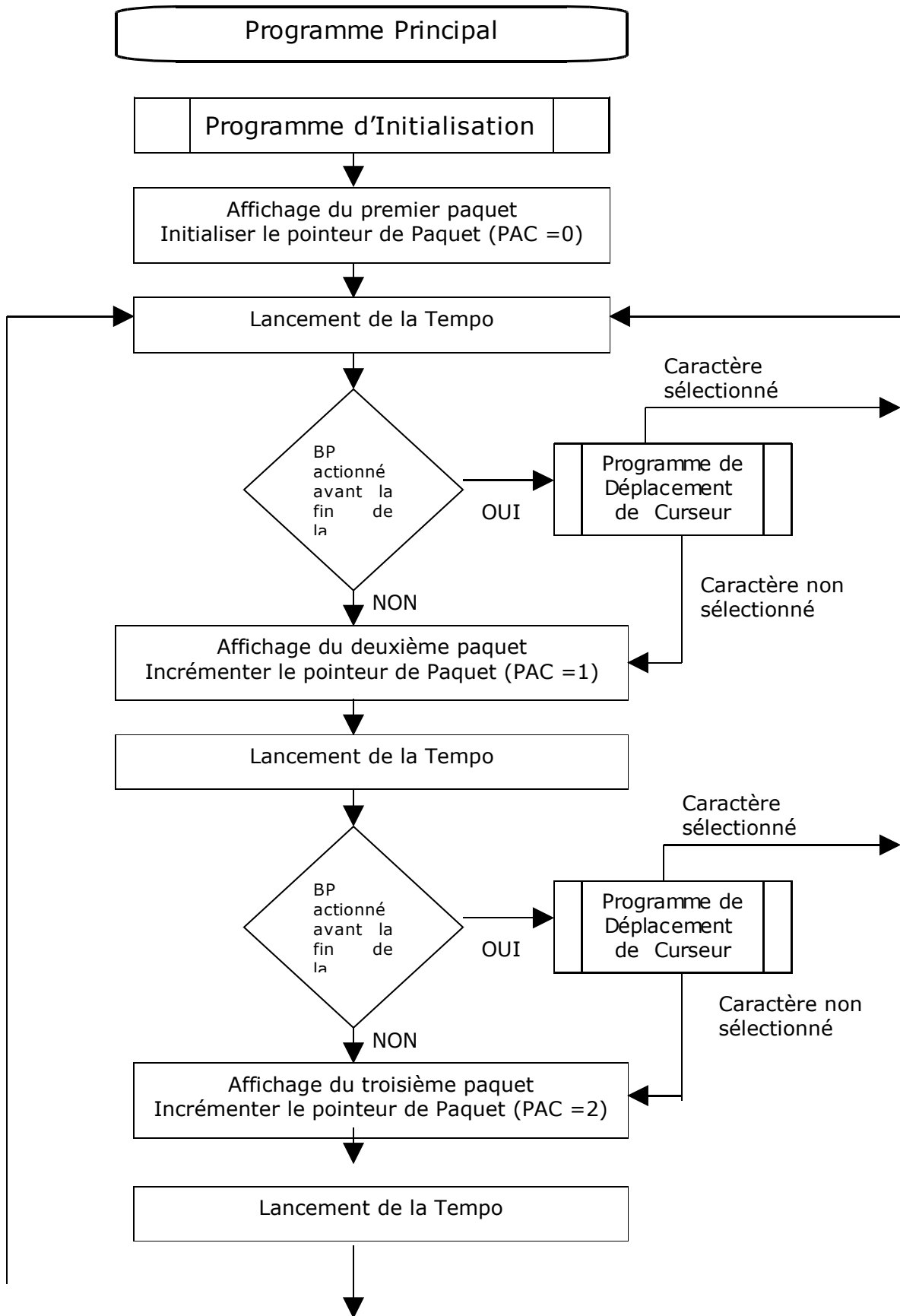
V-1.1. Programme principal

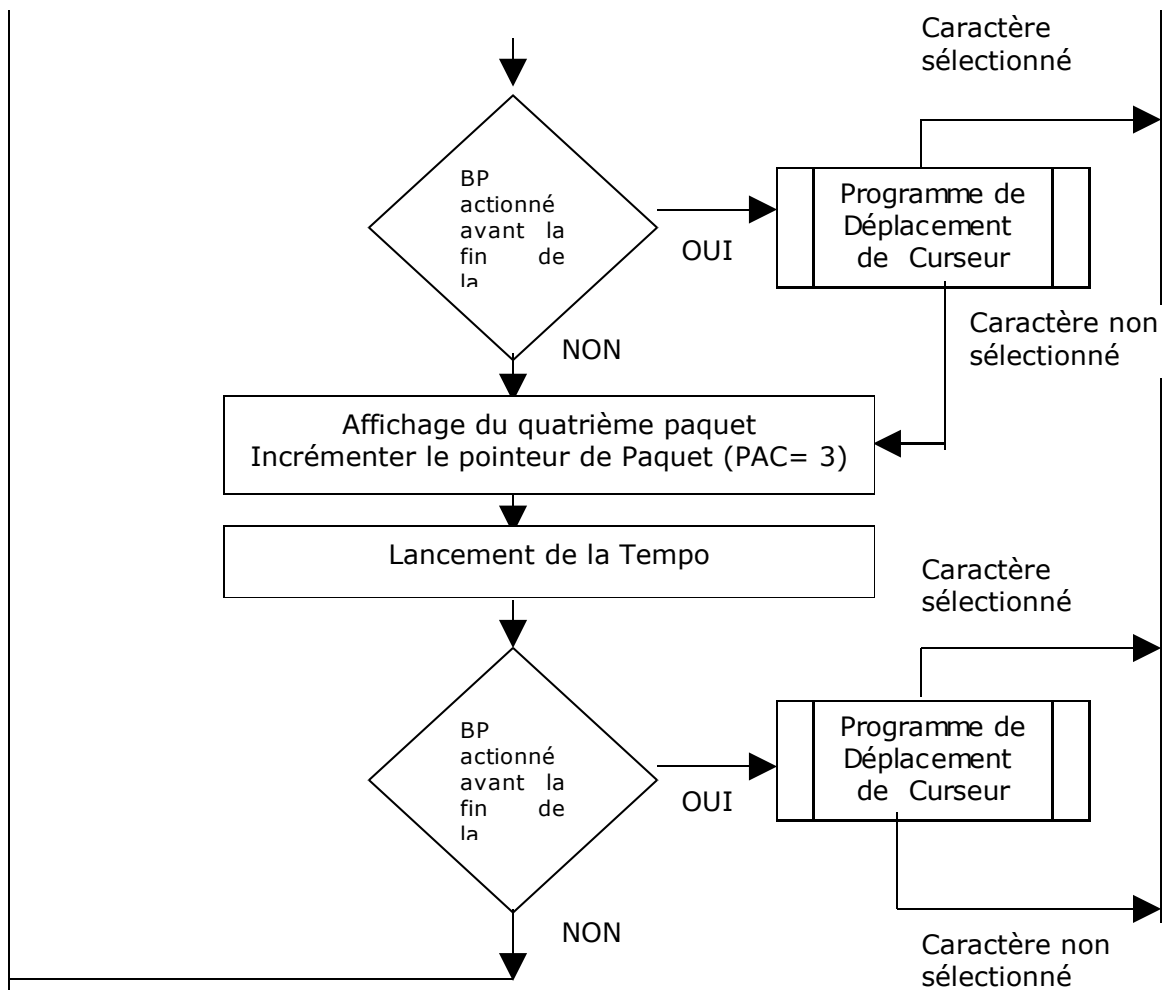
Ce programme sert à gérer le défilement des paquets de caractères ainsi que l'affichage des LCDs.

Au début ,le programme principal initialise le PIC et l'afficheur graphique (au travers du sous programme d'initialisation) puis il envoi le premier paquet de caractères et il attend la validation du bouton poussoir .

Si le bouton poussoir est validé, le programme de déplacement de curseur s'exécute.

Si non le programme principal envoi le prochain paquet de caractère et ainsi de suite.





V-1.2. Sous-programme d'initialisation

Dans un premier temps, ce programme définit l'emplacement des registres nécessaires dans la RAM du PIC.

Ensuite on configure les ports du PIC :

- Port A : en entrée vers le bouton poussoir (entrée analogique).
- Port B : en sortie vers le LCD Alphanumérique (commandes et données).
- Port C : en sortie vers le LCD Graphique (commandes de l'afficheur).
- Port D : en sortie vers le LCD Graphique (données).

Puis on configure le LCD Graphique :

- Définition de la matrice (pour obtenir 7 lignes et 16 colonnes)
- Définition de la zone graphique.
- Définition de la zone de texte.

Enfin, on initialise le compteur E à 32 (nombre de caractères de la zone réservée pour le message, qui est de 2 lignes pour le LCD graphique) et le compteur G à 0 (nombre de caractères saisis dans LCD alphanumérique).

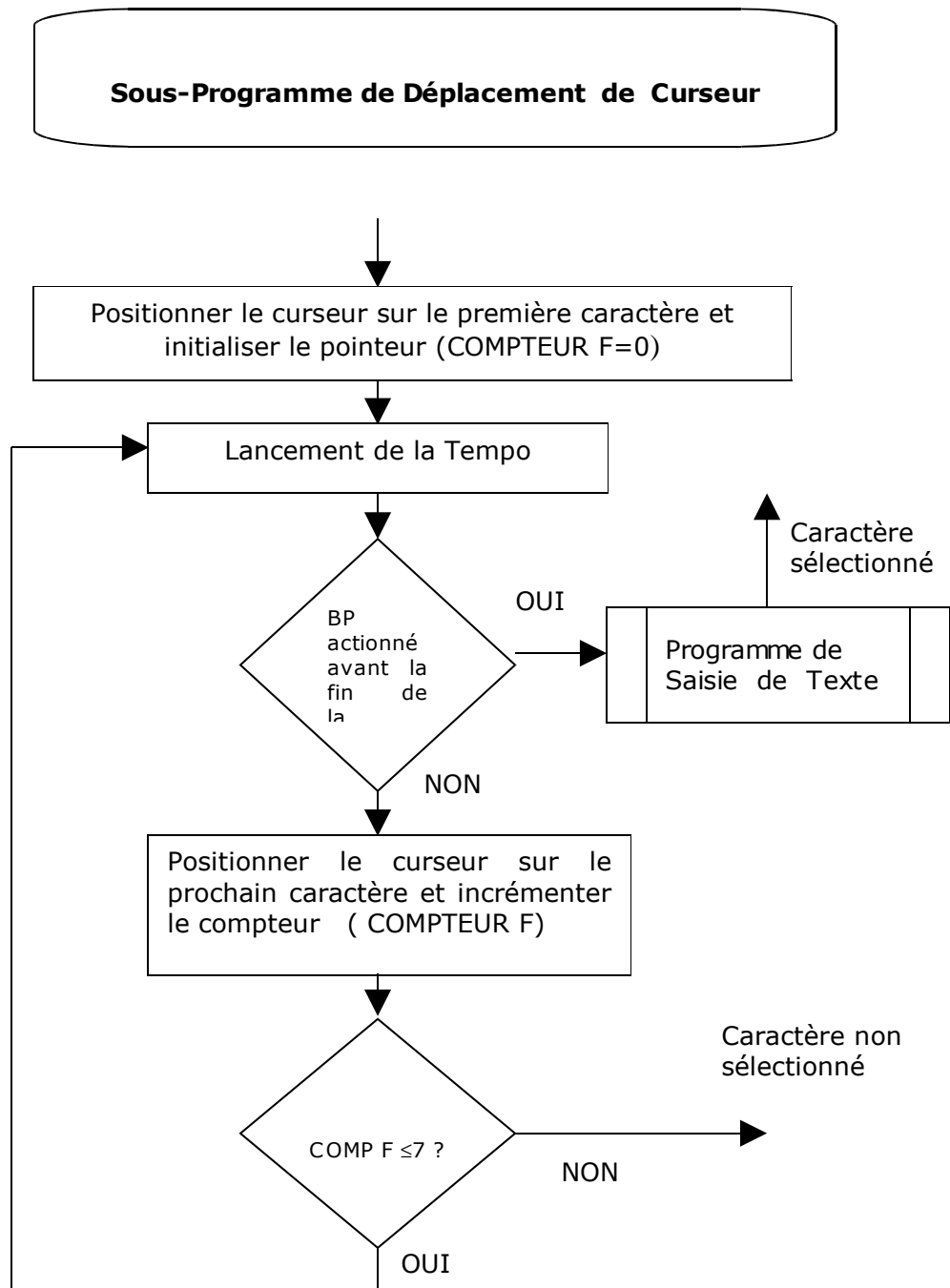
Sous-Programme d'initialisation

- Initialisation des ports
 - ◆ Port C (RC0 à RC5) en sortie vers LCD Graphique.
 - ◆ Port D (RD0 à RD7) en sortie vers LCD Graphique.
 - ◆ Port A (RA4) en entrée vers le BP
 - ◆ Port B (RB0 à RB2 et RB4 à RB7) en sortie vers LCD Alphanumérique.
 - Initialisation du LCD Graphique
 - ◆ Nombre de lignes et de colonnes.
 - ◆ Zone graphique.
 - ◆ Zone de texte.
 - Initialisation du LCD 2x20
 - ◆ Mode 4 bit
 - Initialisation du Compteur G
 - ◆ COMPTEUR G=0
 - Initialisation du Compteur E
 - ◆ COMPTEUR E=32

V-1.3. Sous-programme de déplacement du curseur

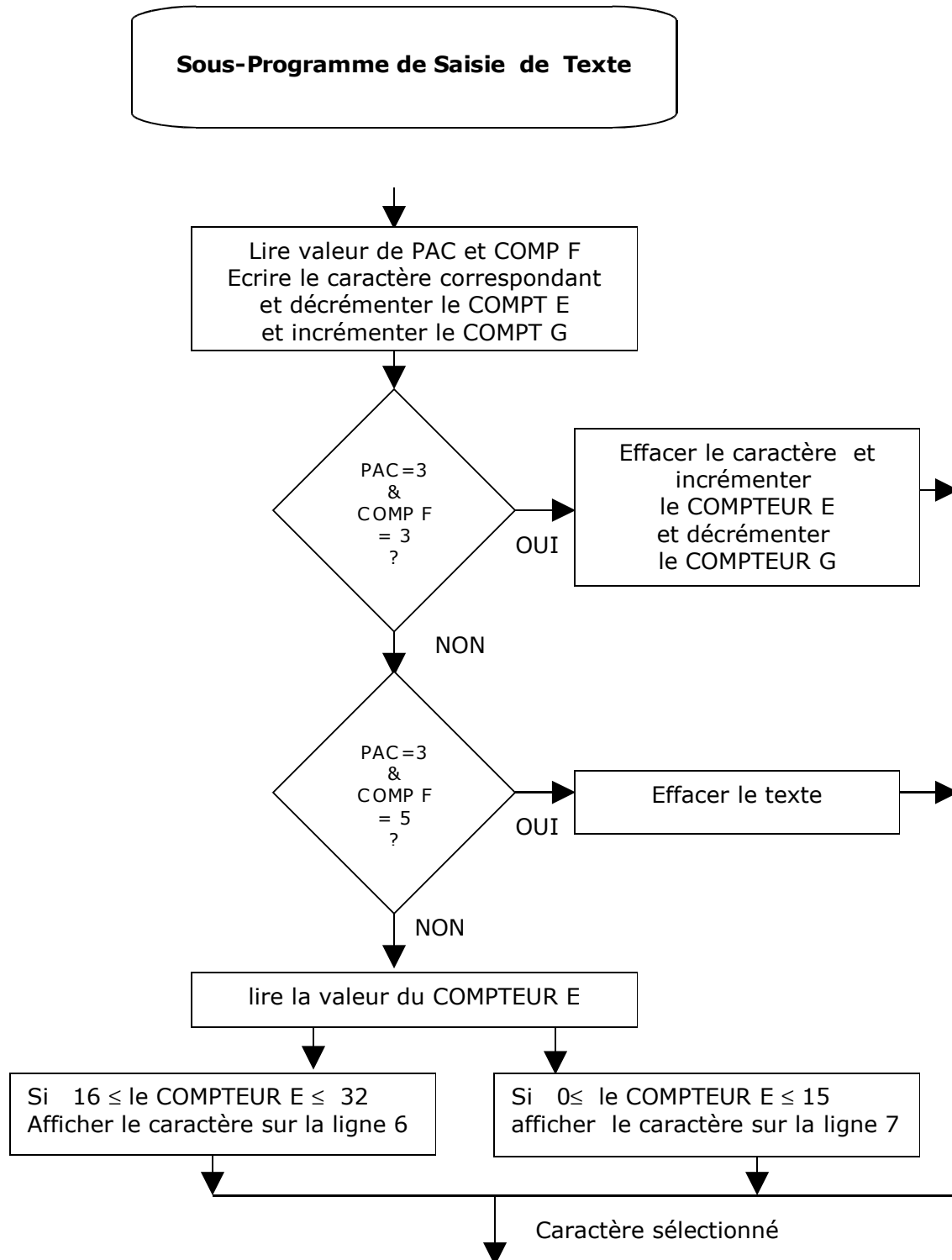
Ce sous-programme permet de gérer le défilement des caractères. En effet, après l'écoulement de la temporisation et la non revalidation du bouton poussoir par la personne, le curseur se déplace d'un caractère à l'autre jusqu'au dernier caractère. A ce moment là, le sous-programme de déplacement de curseur retourne au programme principal pour afficher le prochain de paquet.

Dans le cas contraire, c'est à dire quand la personne revalide le bouton poussoir, le sous-programme de saisie de texte s'exécute.



V-1.4. Sous-programme de saisie de texte

Une fois la validation d'un caractère, ce sous-programme l'identifie à partir de la valeur du paquet et du compteur F et l'affiche à la suite du message existant (la position de ce caractère est défini à l'aide du compteur E). Une fois affiché le sous programme retourne au programme principal, en recommençant l'affichage par le premier paquet.



V-2. Choix du langage de programmation

La programmation du PIC peut se faire en assembleur, Basic ou en C. J'ai développé mon programme en assembleur pour plusieurs raisons parmi lesquelles la possibilité de contrôler précisément les temps d'exécution et aussi parce que les compilateurs en assembleur sont gratuits contrairement en C ou en Basic.

C'est toutefois le plus proche de la machine, il produit un code compact et rapide, mais tout doit être défini à la main. Il est parfait pour les applications en temps réel qui demandent la maîtrise de timings très courts. Il est très stimulant intellectuellement pour qui veut s'impliquer dans des projets complexes.

Grâce à l'architecture RISC, le jeu d'instruction du PIC 16F877 se limite à 35 instructions élémentaires, ce qui simplifie la programmation.

V-3. Environnement de développement

V-3.1. MPLAB

Mplab est un outil fourni gratuitement par la société Microchip, il suffit de se connecter sur le site www.microchip.com pour le télécharger. Ce logiciel permet de créer un programme pour un PIC, de l'assembler et de le simuler avant le transfert vers la mémoire flash du PIC.

Il contient :

- ◆ un éditeur
- ◆ un assembleur
- ◆ un simulateur

Il permet :

- ◆ La rédaction du fichier source en langage assembleur (fichier.ASM).
- ◆ Sa transformation en fichier objet (fichier.HEX) prêt à être chargé dans la mémoire programme du microcontrôleur.

L'ensemble des fichiers nécessaires à ces opérations est regroupé dans un espace " projet " (fichier.PJT).

V-3.2. PROTEUS

Proteus est un outil professionnel de saisie de schéma et de conception de circuits imprimés avec placement automatique, routage et rapports .Il est composé principalement de deux outils **ISIS** et **ARES**.

ISIS :

ISIS est beaucoup plus qu'un logiciel de saisie de schéma classique. Au cœur du système PROTEUS, ISIS est conçu pour répondre parfaitement aux besoins des deux fonctions d'un schéma :

La saisie rapide de projets très complexes destinés aux simulations et aux conceptions de cartes.

La création de très beaux schémas destinés aux publications.

ARES :

Le module ARES de PROTEUS permet de dessiner manuellement ou automatiquement des typons.

En résumé, le cheminement du projet est modélisé par le graphique suivant :

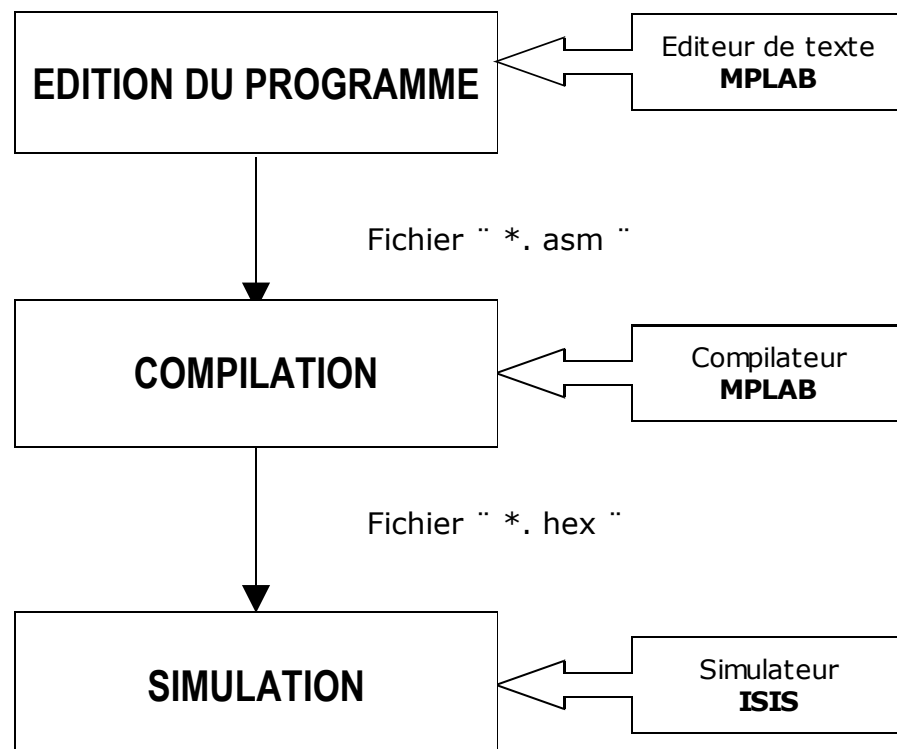


Figure 11: cheminement du projet



VI. ETATS D'AVANCEMENT

Tâches	Remarques
Choix de microprocesseur	Tâche réalisée
Choix du premier afficheur	Tâche réalisée
Choix du deuxième afficheur	Tâche réalisée
Choix d'EEPROM	Tâche non réalisée
Réalisation de schéma de câblage	Tâche réalisée
Défilement des paquets de caractères	Tâche réalisée
Défilement entre les caractères	Tâche réalisée
Affichage du caractère édité	Tâche réalisée
Mémorisation du texte édité	Tâche non réalisée
Affichage des mots du dictionnaire	Tâche non réalisée

Malheureusement, toutes les tâches qui concernent le dictionnaire n'ont pas été réalisées par manque de temps. Pour les autres tâches on dispose déjà de leurs parties matérielles et logicielles qu'il faut mettre en œuvre.

CONCLUSION

Après une analyse du cahier des charges j'ai proposé une analyse fonctionnelle afin de décomposer le problème en fonction simple à réaliser. Cette étape m'a tout simplement conduit à me documenter.

J'ai donc commencé à apprendre l'assembleur et le fonctionnement du PIC et débiter par des petits montages et réaliser leurs simulations. Suite à cela j'ai commencé à réaliser la carte d'outil de communication en implantant le LCD Graphique et le PIC dans le schéma de câblage en premier lieu, et vérifier le bon fonctionnement du programme, ainsi j'ai rajouté le deuxième LCD alphanumérique en deuxième temps.


Pour cette réalisation j'ai été amené à répartir le travail de la manière suivante :

- ✓ Réalisation de la carte sous le logiciel PROTEUS.
- ✓ Edition et compilation du programme sous MPLAB.

De cette manière j'ai pu réinvestir mes connaissances et les savoir faire acquis au cours de l'année. De plus j'ai eu pu acquérir des nouvelles compétences : utilisation de MPLAB, utilisation de PROTEUS , découverte de la famille PIC.

BIBLIOGRAPHIE

- **S'initier à la programmation des PIC : basic et assembleur.**
Alain REBOUX (édition Dunod 2002).
- **Apprendre la programmation des PIC par l'expérimentation et la simulation.**
Mayeux Pascal (Paris Dunod 2002).
- **Les microcontrôleurs PIC.**
Bernard BEGHYN.
- **Les microcontrôleurs PIC : description et mise en œuvre.**
Christian TAVERNIER.
- **Article « Pilote d'afficheur graphique ».**
Electronique pratique N°279, page 86.

 Quelques liens utiles :

www.microchip.com DATASHEET du PIC de la série 16FXXX .

www.abcelectronique.com/Bignoff (cours détaillé de la programmation des PICs).

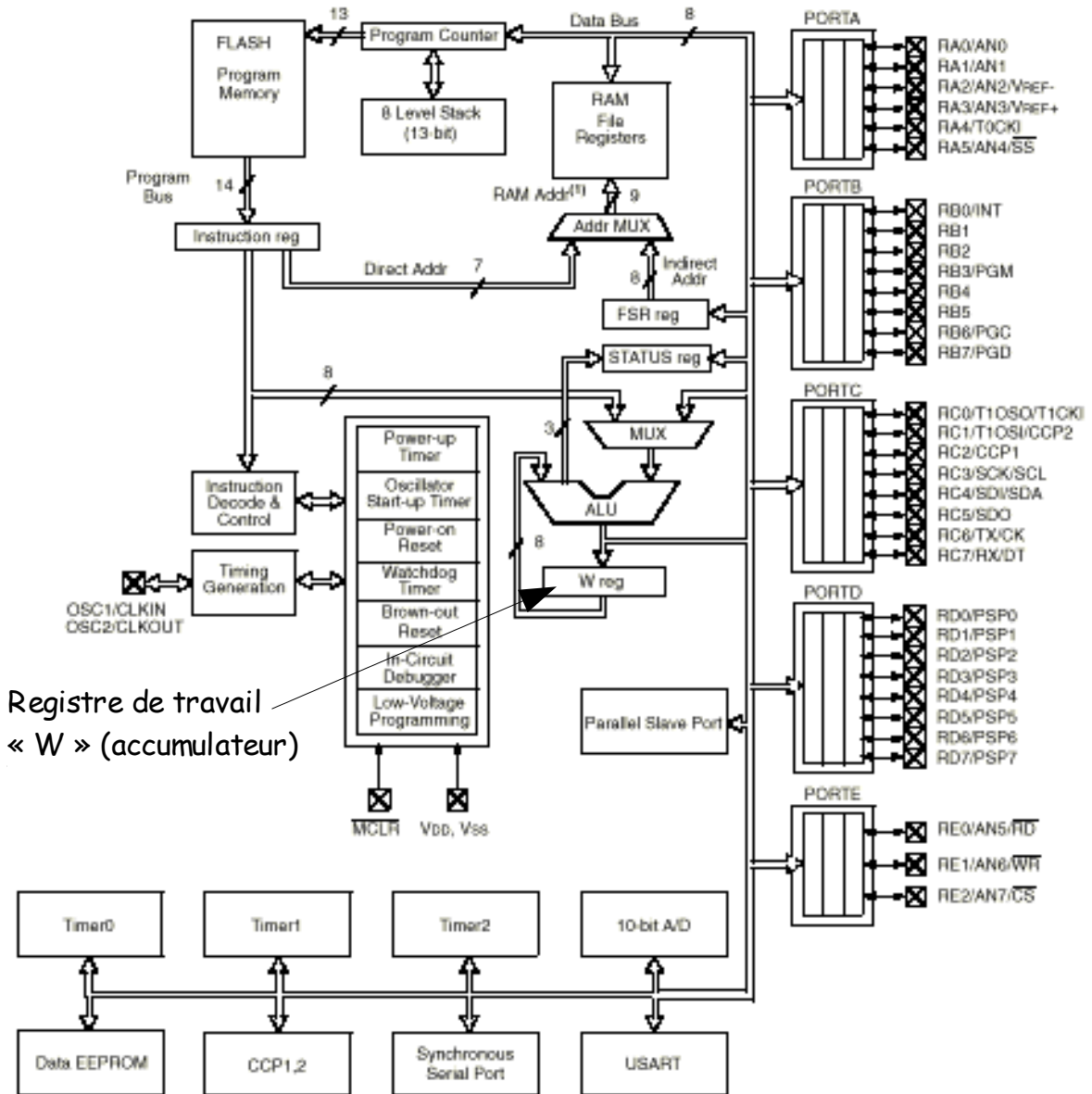
www.semicon.toshiba.co.jp/td/en/ASSP/Display_Driver_Ics/en_20030618_T6963C_datasheet.pdf

www.radiospares.com (site des composants électronique des différents fournisseurs)



ANNEXE 1

Device	Program FLASH	Data Memory	Data EEPROM
PIC16F874	4K	192 Bytes	128 Bytes
PIC16F877	8K	368 Bytes	256 Bytes


Architecture interne d'un PIC 16F877

I. GESTION MEMOIRE DU PIC 16F877

- La **mémoire programme** est constituée de 8k mots de 14 bits. C'est dans cette zone que nous allons écrire notre programme, sachant qu'une instruction est codée sur 1mot.
- La mémoire **EEPROM** est constituée de 256 octets qu'on peut lire et écrire depuis notre programme, ces octets sont conservés après coupure de courant et sont très utiles pour conserver des paramètres semi-permanents.
- La mémoire **RAM** est constituée de 368 octets, subdivisée en 4 banques. Nous voyons dans le tableau au dessous que les adresses s'échelonnent entre 0x00 et 0x1FF.

FIGURE 2-3: PIC16F877/876 REGISTER FILE MAP

File Address	File Address	File Address	File Address
Indirect addr. ^(*) 00h	Indirect addr. ^(*) 80h	Indirect addr. ^(*) 100h	Indirect addr. ^(*) 180h
TMR0 01h	OPTION_REG 81h	TMR0 101h	OPTION_REG 181h
PCL 02h	PCL 82h	PCL 102h	PCL 182h
STATUS 03h	STATUS 83h	STATUS 103h	STATUS 183h
FSR 04h	FSR 84h	FSR 104h	FSR 184h
PORTA 05h	TRISA 85h		
PORTB 06h	TRISB 86h	PORTB 106h	TRISB 186h
PORTC 07h	TRISC 87h		
PORTD ⁽¹⁾ 08h	TRISD ⁽¹⁾ 88h		
PORTE ⁽¹⁾ 09h	TRISE ⁽¹⁾ 89h		
PCLATH 0Ah	PCLATH 8Ah	PCLATH 10Ah	PCLATH 18Ah
INTCON 0Bh	INTCON 8Bh	INTCON 10Bh	INTCON 18Bh
PIR1 0Ch	PIE1 8Ch	EEDATA 10Ch	EECON1 18Ch
PIR2 0Dh	PIE2 8Dh	EEADR 10Dh	EECON2 18Dh
TMR1L 0Eh	PCON 8Eh	EEDATH 10Eh	Reserved ⁽²⁾ 18Eh
TMR1H 0Fh		EEADRH 10Fh	Reserved ⁽²⁾ 18Fh
T1CON 10h			
TMR2 11h	SSPCON2 91h		
T2CON 12h	PR2 92h		
SSPBUF 13h	SSPAD 93h		
SSPCON 14h	SSPSTAT 94h		
CCPR1L 15h			
CCPR1H 16h			
CCP1CON 17h			
RCSTA 18h	TXSTA 98h	General Purpose Register 16 Bytes 117h-11Fh	General Purpose Register 16 Bytes 197h-19Fh
TXREG 19h	SPBRG 99h		
RCREG 1Ah			
CCPR2L 1Bh			
CCPR2H 1Ch			
CCP2CON 1Dh			
ADRESH 1Eh	ADRESL 9Eh		
ADCON0 1Fh	ADCON1 9Fh		
General Purpose Register 96 Bytes 20h-7Fh	General Purpose Register 80 Bytes A0h-EFh	General Purpose Register 80 Bytes 120h-1BFh	General Purpose Register 80 Bytes 1A0h-1EFh
Bank 0	Bank 1	Bank 2	Bank 3
	accesses 70h-7Fh	accesses 70h-7Fh	accesses 70h-7Fh

368 octets de RAM

Unimplemented data memory locations, read as '0'.
^{*} Not a physical register.

Note 1: These registers are not implemented on the PIC16F876.
Note 2: These registers are reserved, maintain these registers clear.

II. MODE D'ADRESSAGE DU PIC 16F877

II-1. L'ADRESSAGE DIRECT

Pour ce type d'adressage, il nous manque donc deux bits. Nous allons trouver ces deux bits dans le registre **STATUS**. La combinaison des bits **RP0** et **RP1** de ce registre, permet donc d'accéder en adressage direct à l'intégralité de la RAM.

L'adresse finale est donc composée de RP1|RP0 comme bits 8 et 7, complétés des 7 bits de l'adresse directe utilisée.

On représente les 4 possibilités concernant RP1 et RP0 par le tableau suivant :

RP1	RP0	Accès à la RAM
0	0	0x00 à 0x7F
0	1	0x80 à 0xFF
1	0	0x100 à 0x17F
1	1	0x180 à 0x1FF

Notez que la RAM située de 0x70 à 0x7F est accessible quel que soit l'état de RP0 et RP1. En effet, les zones correspondantes dans les autres banques sont en fait des images de cette zone.

II-2. L'ADRESSAGE INDIRECT

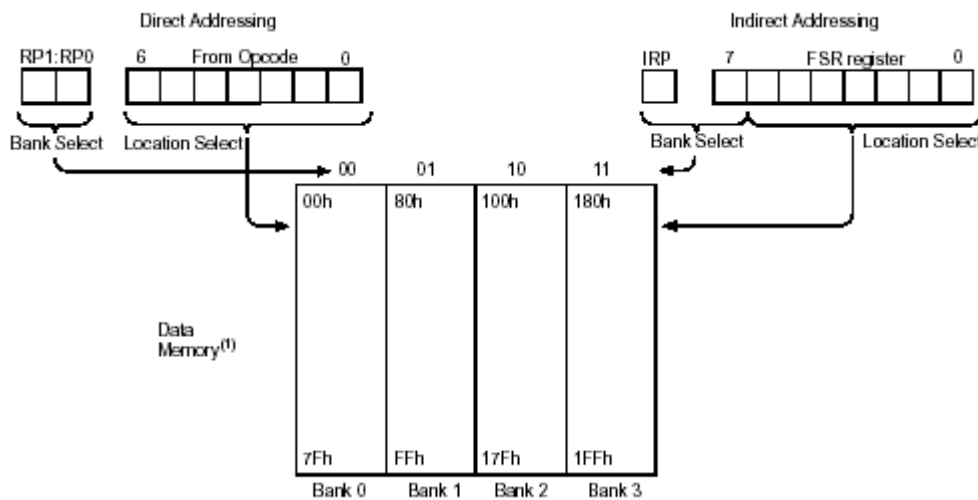
L'adressage indirect utilise le registre FSR/INDF. Hors, ce registre à une largeur de 8 bits. Donc, tel quel, il lui est impossible d'adresser plus de 2 banques. Il va donc falloir trouver une fois de plus un bit supplémentaire.

Ce bit est le bit IRP du registre STATUS. Ce bit est donc utilisé comme bit poids fort

(bit 8) complété par les 8 bits contenus dans le registre FSR.

Pour résumer :

- II- Vous devez toujours vérifier RP0 et RP1 avant toute utilisation de l'adressage direct.
- III- Vous devez toujours vérifier IRP avant toute utilisation de l'adressage indirect.





III. LES PRINCIPAUX REGISTRES DU PIC 16F877

III-1. LE REGISTRE "W"

Ce registre est un registre utilisé par les pics pour réaliser toutes sortes de calculs.

La destination d'un résultat peut en général être un emplacement RAM (f) ou le registre de travail (W). C'est donc un registre fondamental.

III-2. LE REGISTRE "STATUS"

Ce registre est principalement utilisé pour tout ce qui concerne les tests.

Il est donc également d'une importance fondamentale.

REGISTER 2-1: STATUS REGISTER (ADDRESS 03h, 83h, 103h, 183h)

R/W-0	R/W-0	R/W-0	R-1	R-1	R/W-x	R/W-x	R/W-x
IRP	RP1	RP0	TO	PD	Z	DC	C
bit 7							bit 0

C : Ce bit est en fait le 9^{ème} bit d'une opération mathématique, c'est le bit du retenue.

DC : Ce bit est utilisé principalement lorsque on travaille avec des nombres BCD.

Z : Ce bit est positionné à 0 c'est le résultat de la dernière opération vaut 0.

**LES INSTRUCTIONS DU PIC 16F877**

Mnemonic, Operands	Description	Cycles	14-Bit Opcode		Status Affected	Notes
			MSb	LSb		
BYTE-ORIENTED FILE REGISTER OPERATIONS						
ADDWF	f, d Add W and f	1	00	0111 dfff ffff	C,DC,Z	1,2
ANDWF	f, d AND W with f	1	00	0101 dfff ffff	Z	1,2
CLRF	f Clear f	1	00	0001 lfff ffff	Z	2
CLRWF	- Clear W	1	00	0001 0000 0011	Z	
COMF	f, d Complement f	1	00	1001 dfff ffff	Z	1,2
DECF	f, d Decrement f	1	00	0011 dfff ffff	Z	1,2
DECFSZ	f, d Decrement f, Skip if 0	1(2)	00	1011 dfff ffff		1,2,3
INCF	f, d Increment f	1	00	1010 dfff ffff	Z	1,2
INCFSZ	f, d Increment f, Skip if 0	1(2)	00	1111 dfff ffff		1,2,3
IORWF	f, d Inclusive OR W with f	1	00	0100 dfff ffff	Z	1,2
MOVF	f, d Move f	1	00	1000 dfff ffff	Z	1,2
MOWWF	f Move W to f	1	00	0000 lfff ffff		
NOP	- No Operation	1	00	0000 0xx0 0000		
RLF	f, d Rotate Left f through Carry	1	00	1101 dfff ffff	C	1,2
RRF	f, d Rotate Right f through Carry	1	00	1100 dfff ffff	C	1,2
SUBWF	f, d Subtract W from f	1	00	0010 dfff ffff	C,DC,Z	1,2
SWAPF	f, d Swap nibbles in f	1	00	1110 dfff ffff		1,2
XORWF	f, d Exclusive OR W with f	1	00	0110 dfff ffff	Z	1,2
BIT-ORIENTED FILE REGISTER OPERATIONS						
BCF	f, b Bit Clear f	1	01	00bb bfff ffff		1,2
BSF	f, b Bit Set f	1	01	01bb bfff ffff		1,2
BTFSC	f, b Bit Test f, Skip if Clear	1(2)	01	10bb bfff ffff		3
BTFSS	f, b Bit Test f, Skip if Set	1(2)	01	11bb bfff ffff		3
LITERAL AND CONTROL OPERATIONS						
ADDLW	k Add literal and W	1	11	111x kkkk kkkk	C,DC,Z	
ANDLW	k AND literal with W	1	11	1001 kkkk kkkk	Z	
CALL	k Call subroutine	2	10	0kkk kkkk kkkk		
CLRWDT	- Clear Watchdog Timer	1	00	0000 0110 0100	$\overline{TO,PD}$	
GOTO	k Go to address	2	10	1kkk kkkk kkkk		
IORLW	k Inclusive OR literal with W	1	11	1000 kkkk kkkk	Z	
MOVLW	k Move literal to W	1	11	00xx kkkk kkkk		
RETFIE	- Return from interrupt	2	00	0000 0000 1001		
RETLW	k Return with literal in W	2	11	01xx kkkk kkkk		
RETURN	- Return from Subroutine	2	00	0000 0000 1000		
SLEEP	- Go into standby mode	1	00	0000 0110 0011	$\overline{TO,PD}$	
SUBLW	k Subtract W from literal	1	11	110x kkkk kkkk	C,DC,Z	
XORLW	k Exclusive OR literal with W	1	11	1010 kkkk kkkk	Z	



ANNEXE 2

I. PRINCIPE DE COMMANDE D'UN LCD GRAPHIQUE

La commande d'un lcd graphique comprend :

- L'initialisation et l'activation de l'afficheur.
- L'écriture en RAM des données à afficher.

Ceci se fait sur le principe :

- D'utilisation d'un jeu de broches de communication :
 - RS (ou C/D) qui permet de déterminer si les broches de données reçoivent une instruction ou des données RAM.
 - R/W qui permet de faire fonctionner l'afficheur en mode écriture ou lecture.
 - E qui permet de valider (E=1) les données présentent sur les broches de données. La broche E est à mettre au niveau HAUT par défaut.
- D'utilisation d'un jeu de broches de données :
 - la plupart des afficheurs graphiques fonctionnent en mode 8 bits
 - l'octet envoyé à l'afficheur sera soit une instruction (si C/D est à 0), soit une donnée pour la RAM (si C/D est à 1)
- D'utilisation d'un jeu de broches spécifiques :
 - parfois une broche RESET
 - souvent, des broches de sélection des blocs graphiques correspondant en fait aux broches de sélection (CS=1) des drivers de colonnes.



II. PRINCIPE D'ECRITURE D'UNE DONNEE EN RAM

1^{ère} étape : se positionner à l'adresse voulue en RAM .

Ceci passe par l'envoi d'une instruction de positionnement en colonne suivi d'une instruction de positionnement en ligne. Ceci une fois fait, il ne reste plus qu'à écrire l'octet voulu à l'emplacement de la RAM ainsi désigné.

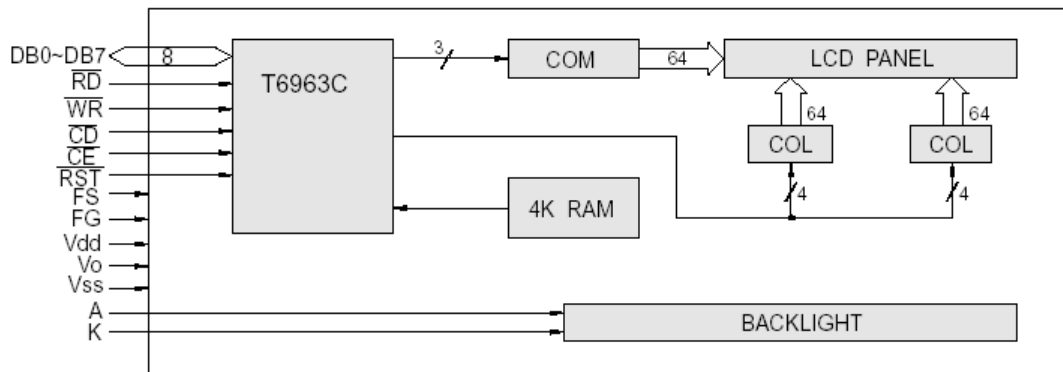
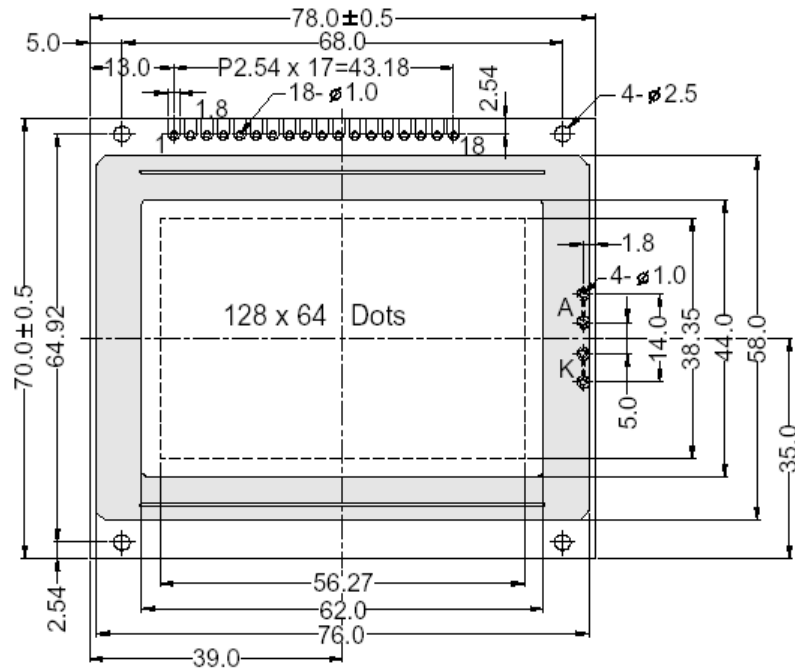
2^{ème} étape : Ecriture d'une donnée en RAM .

La séquence d'envoi d'une donnée à l'afficheur LCD graphique est :

- mise à 0 de la broche R/W.
- mise à 1 de la broche RS (ou C/D).
- mise de l'octet sur les broches de données.
- validation de l'instruction par un front descendant sur la broche E.



III. LCD GRAPHIQUE 128x64



PIN ASSIGNMENT

Pin no.	Symbol	Function
1	FG	Frame ground
2	Vss	Power supply(GND)
3	Vdd	Power supply(+)
4	Vo	Contrast Adjust
5	WR	Data write
6	RD	Data read
7	CE	Chip enable
8	CD	Command / data select
9	RST	Reset
10-17	DB0-DB7	Data bus line
18	FS	Font select

Spécifications Techniques :

- Alimentation: +5 V
- Température d'utilisation: -10°

format	Dim. module (mm)			Fenêtre d'affichage (mm)	
	L	I	P	L	I
(rangées x col.)					
128 x 64	78	70	9,7	56,27	38,35



IV. Le tableau des commandes

LCD Graphique 128x64 de Powertip possède des différents commandes ou un jeu d'instructions très complet, présenté par le tableau suivant :

Command Definitions

Command	Code	D1	D2	Function
REGISTERS SETTING	00100001	X address	Y address	Set Cursor Pointer
	00100010	Data	00H	Set Offset Register
	00100100	Low address	High address	Set Address Pointer
SET CONTROL WORD	01000000	Low address	High address	Set Text Home Address
	01000001	Columns	00H	Set Text Area
	01000010	Low address	High address	Set Graphic Home Address
	01000011	Columns	00H	Set Graphic Area
MODE SET	1000X000	—	—	OR mode
	1000X001	—	—	EXOR mode
	1000X011	—	—	AND mode
	1000X100	—	—	Text Attribute mode
	10000XXX	—	—	Internal CG ROM mode
	10001XXX	—	—	External CG RAM mode
DISPLAY MODE	10010000	—	—	Display off
	1001XX10	—	—	Cursor on, blink off
	1001XX11	—	—	Cursor on, blink on
	100101XX	—	—	Text on, graphic off
	100110XX	—	—	Text off, graphic on
	100111XX	—	—	Text on, graphic on
CURSOR PATTERN SELECT	10100000	—	—	1-line cursor
	10100001	—	—	2-line cursor
	10100010	—	—	3-line cursor
	10100011	—	—	4-line cursor
	10100100	—	—	5-line cursor
	10100101	—	—	6-line cursor
	10100110	—	—	7-line cursor
	10100111	—	—	8-line cursor
DATA AUTO READ / WRITE	10110000	—	—	Set Data Auto Write
	10110001	—	—	Set Data Auto Read
	10110010	—	—	Auto Reset
DATA READ / WRITE	11000000	Data	—	Data Write and Increment ADP
	11000001	—	—	Data Read and Increment ADP
	11000010	Data	—	Data Write and Decrement ADP
	11000011	—	—	Data Read and Decrement ADP
	11000100	Data	—	Data Write and Nonvariable ADP
	11000101	—	—	Data Read and Nonvariable ADP
SCREEN PEEK	11100000	—	—	Screen Peek
SCREEN COPY	11101000			Screen Copy

X: invalid

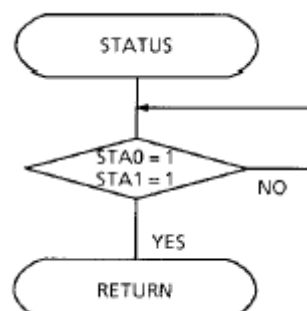
IV-1. Séquence d'envoi d'une commande

La séquence d'envoi d'une commande à l'afficheur LCD graphique est :

- Mise à 0 de la broche |RW.
- Mise à 1 de la broche |RD.
- Mise à 1 de la broche CD.
- Mise à 0 de la broche |CE.
- Vérification du status « status check ». (voir le tableau au dessous)
- Mise du code de la commande sur les broches de données.
- Vérification du status « status check ».
- Ecrire la commande.

Cette séquence est représenté sous forme d'une routine dans notre programme.

MSB				LSB			
STA7 D7	STA6 D6	STA5 D5	STA4 D4	STA3 D3	STA2 D2	STA1 D1	STA0 D0
STA0	Check command execution capability						0: Disable 1: Enable
STA1	Check data read / write capability						0: Disable 1: Enable
STA2	Check Auto mode data read capability						0: Disable 1: Enable
STA3	Check Auto mode data write capability						0: Disable 1: Enable
STA4	Not used						
STA5	Check controller operation capability						0: Disable 1: Enable
STA6	Error flag. Used for Screen Peek and Screen copy commands.						0: No error 1: Error
STA7	Check the blink condition						0: Display off 1: Normal display





IV-2. Séquence d'envoi d'une donnée

La séquence d'envoi d'une commande à l'afficheur LCD graphique est :

- Mise à 0 de la broche |RW .
- Mise à 1 de la broche |RD
- Mise à 0 de la broche |CD
- Mise du code sur les broches de données.
- Mise à 1 de la broche |CE.

IV-3. Organisation de la RAM

⇒ **L'écran et la RAM graphique sont divisés en 2.**

Si l'on se reporte à la structure fonctionnelle de l'afficheur, on remarque qu'il utilise 2 drivers de 64 colonnes qui sont reliés à une RAM graphique a définir.

Donc en fait, **chaque driver gère l'affichage d'une moitié de l'écran : 64 colonnes (x 64 lignes) chacun.**

La sélection de la moitié de l'écran dans laquelle on veut « écrire » va se faire par 2 broches d'activation des drivers appelée CE0 et CE1. Ces 2 broches seront à connecter sur des broches E/S du PIC. Elles seront gérées par les routines écrites dans le programme pour l'afficheur.



ANNEXE 3



I. Tableau des différentes commandes de l'afficheur alphanumérique

TYPE DE COMMANDE											DESCRIPTIF	
	RS	R/W	D7	D6	D5	D4	D3	D2	D1	D0		
<i>EFFACER L'AFFICHAGE</i>	0	0	0	0	0	0	0	0	0	0	1	Efface l'ensemble de la mémoire de données sans toucher au générateur de caractère. Met le curseur en position Home, à l'adresse 00.
<i>CURSEUR EN POSITION HOME</i>	0	0	0	0	0	0	0	0	0	1	*	Met le curseur en position Home. Si l'affichage à été décalé, il est remis à sa position d'origine: l'adresse 00 se trouve à nouveau en haut à gauche.
<i>MANIERE DE VISUALISER LES CARACTERES</i>	0	0	0	0	0	0	0	0	1	ID	S	Détermine le sens de déplacement du curseur après apparition d'un caractère (ID) et le déplacement collectif d'une position de l'ensemble de l'affichage (S).
<i>MARCHE/ARRET DE L'AFFICHAGE DU CURSEUR</i>	0	0	0	0	0	0	0	1	D	C	B	Met l'affichage en ou hors fonction (D). Met le curseur en ou hors fonction (C). Fait clignoter le caractère situé au-dessus du curseur (B), clignotement se traduisant par une alternance du caractère et du caractère FF (rectangle noir)
<i>DECALAGE</i>	0	0	0	0	0	0	1	S/C	R/L	*	*	Déplace le curseur ou l'ensemble de l'affichage sans modifier le contenu de la mémoire.
<i>FONCTION</i>	0	0	0	0	1	DL	N	F	*	*	*	Indique la largeur du bus de données. Indique s'il ne faut utiliser que la ligne du haut ou que celle du bas. (F) : matrice
<i>ADRESSE DU GENERATEUR DE CARACTERES</i>	0	0	0	1	Caractère			Rangée			Définit l'adresse de la mémoire du générateur de caractères. Les données suivantes correspondent à la matrice du caractère concerné.	
<i>ADRESSE DE LA MEMOIRE DE DONNEES</i>	0	0	1	Adresse								Définit l'adresse de la mémoire de données. Les données suivantes correspondent au caractère ASCII à visualiser.
<i>INDICATEUR BUSY LECTURE D'ADRESSE</i>	0	1	BF	Adresse								Lit l'indicateur Busy (BF) pour vérifier que l'afficheur et en mesure de traiter la commande suivante. Lit l'adresse de la position du curseur.
<i>ECRITURE DE DONNEES</i>	1	0	Données									Ecrit des données respectivement dans la mémoire de données ou le générateur de caractères.
<i>LECTURE DE DONNEES</i>	1	1	Données									Lit les données respectivement de la mémoire de données ou le générateur de caractères.



II. Description des différentes commandes

	0	1
ID	Déplacement vers la gauche	Déplacement vers la droite
S	L'affichage ne bouge pas	L'affichage est décalé
C	Absence du curseur	Visualisation du curseur
B	Absence de clignotement du caractère	Clignotement du caractère
S/C	Déplacement du curseur	Déplacement de l'affichage
R/L	Décalage vers la gauche	Décalage vers la droite
DL	4 bits	8 bits
N	Ligne du haut	2 lignes validées

Le bit noté **F** permet de définir la matrice des caractères suivant le tableau ci dessous.

N	F	Nombre DE LIGNE	MATRICE
0	0	1	5 * 7
0	1	1	5 * 10
1	*	2	5 * 7



ANNEXE 4



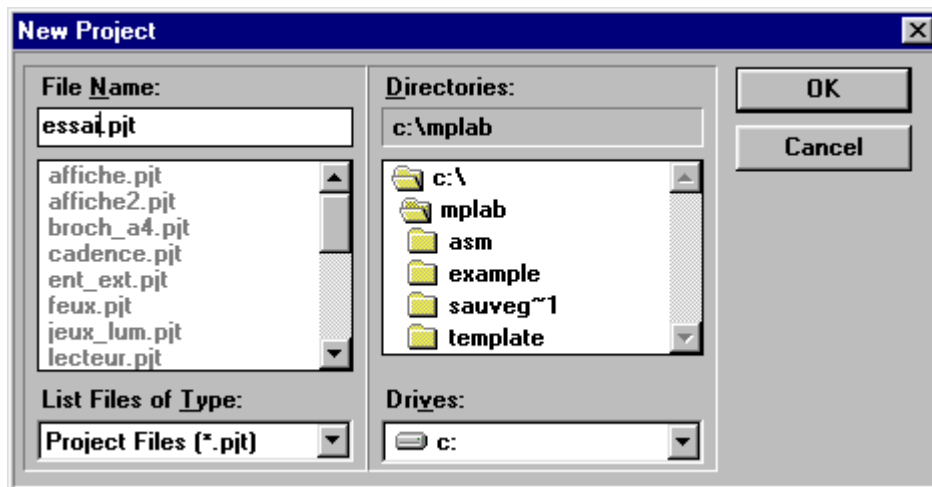
I. Maîtriser MPLAB

Une fois téléchargé, on lance Mplab et on visualise alors l'écran suivant :



I-1. Ouverture d'un nouveau projet

On clique sur **Project - New Project**, l'écran suivant doit apparaître :



On indique dans cette fenêtre le répertoire où devra se sauvegarder le projet(en lecteur C par exemple) et le nom du projet (essai.pjt dans l'exemple) ,une fois terminé on clique sur OK.

La fenêtre "**Edit Project**" qui s'ouvre confirme le nom du fichier objet qui sera créé (essai.hex) .



Pour chaque application il faut choisir un l'éditeur (dans notre cas ,on choisit Editor Only 16F877) et l'environnement (Microchip).

Ensuite on clique sur le nom dans la fenêtre du bas (dans l'exemple `essai.hex`) puis sur "**Add Node**", une autre fenêtre doit apparaître pour indiquer le nom du fichier assembleur qu'on va simuler (dans l'exemple `essai.asm`) puis on valide avec OK. La fenêtre "**Edit Project**" revient et on clique sur OK pour revenir à la page d'accueil.

I-2. Ecriture du programme source

Cliquer sur "**File – New**", la fenêtre qui s'ouvre, nommée pour l'instant "untitled" est celle où l'on doit taper le programme en assembleur.

Une fois terminée on

clique sur "**File - Save As**".

Ensuite on indique le répertoire où devra se sauvegarder le fichier, le même que celui du projet et surtout il faut pas oublier que le nom du fichier doit être le même que celui du projet (`essai.asm`).

I-3. Création du programme objet

Il s'agit de transformer ce fichier `.asm` en fichier `.hex` directement exploitable par le programmeur de PIC pour cela on clique alors sur "**Project - Make Project**".

La transformation de `essai.asm` en `essai.hex` commence.

Dans le cas où il y a des erreurs le compilateur envoie le message suivant (build failed) .Dans ce cas là il faut revenir dans la fenêtre d'écriture du fichier source (`essai.asm`) Puis Corriger les erreurs et Recompiler en cliquant sur *Project - Make Project* et recommencer jusqu'à obtenir le message "Build completed successfully "

MPASM fournit une possibilité de simulation du programme pour aider la correction:

La simulation permet de suivre le cheminement du programme, instruction par instruction. On passe en mode simulation par "**Options - Development Mode**"

et on clique alors sur "**MPLAB-SIM Simulator**"

- Par Window :

Ce mode nous permet de voir le contenu des mémoires et registres après chaque exécution d'instruction.

- Window - Watch Window - New Watch Window:

Ce mode nous permet de voir la valeur des variables.

I-4. Ouverture d'un projet existant

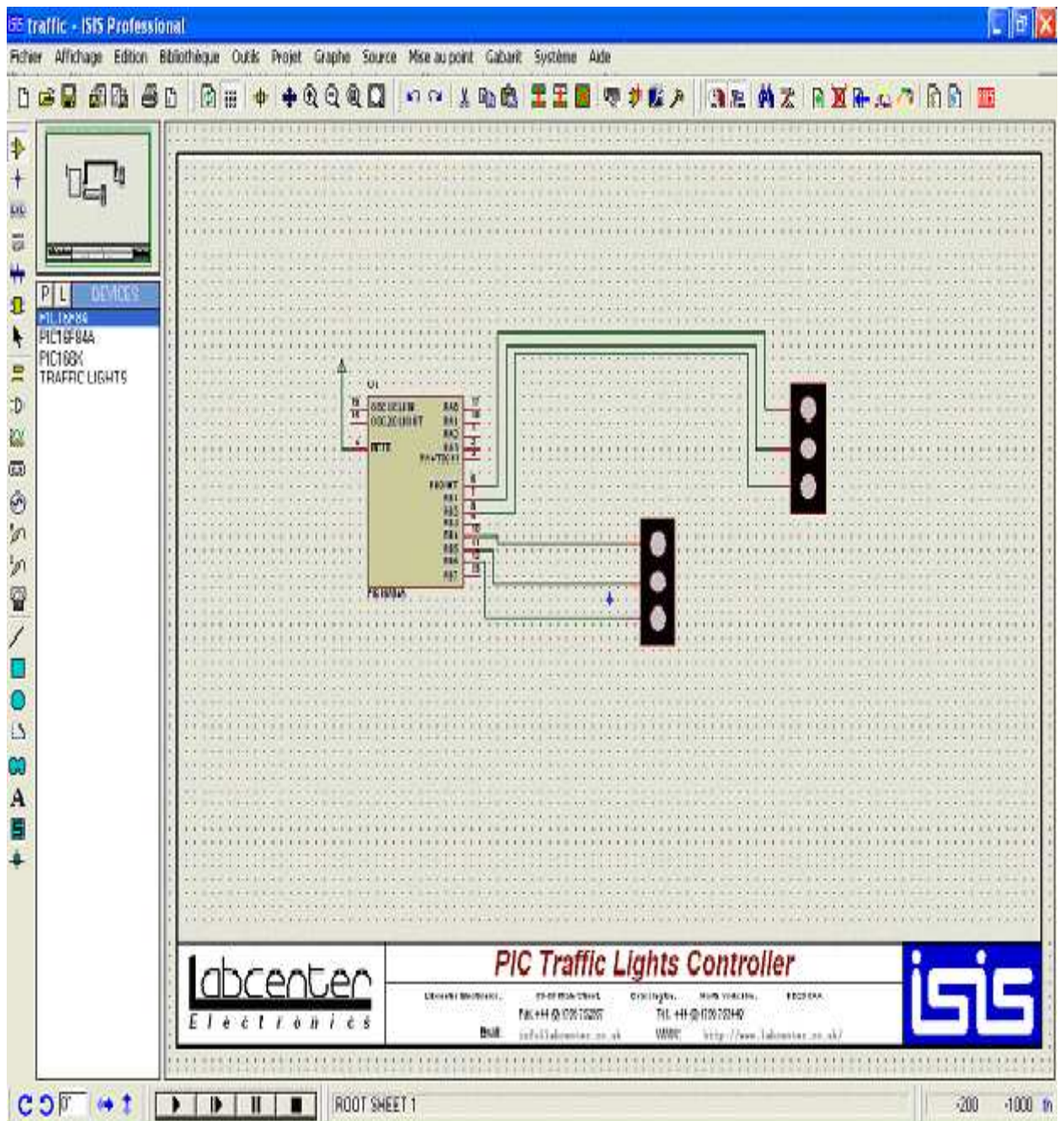
Pour ouvrir un projet déjà existant il suffit de suivre les étapes suivantes :

- Cliquer sur " **Project - Open Project**".
- Indiquer le répertoire où se trouve le projet et le nom du projet (essai.pjt).
- Cliquer sur OK.
- Cliquer sur "**File - Open**".
- Indiquer le répertoire où se trouve le fichier et le nom du fichier (essai.asm) .
- Cliquer sur OK .

Le fichier ".asm" apparaît alors à l'écran.

ANNEXE 5

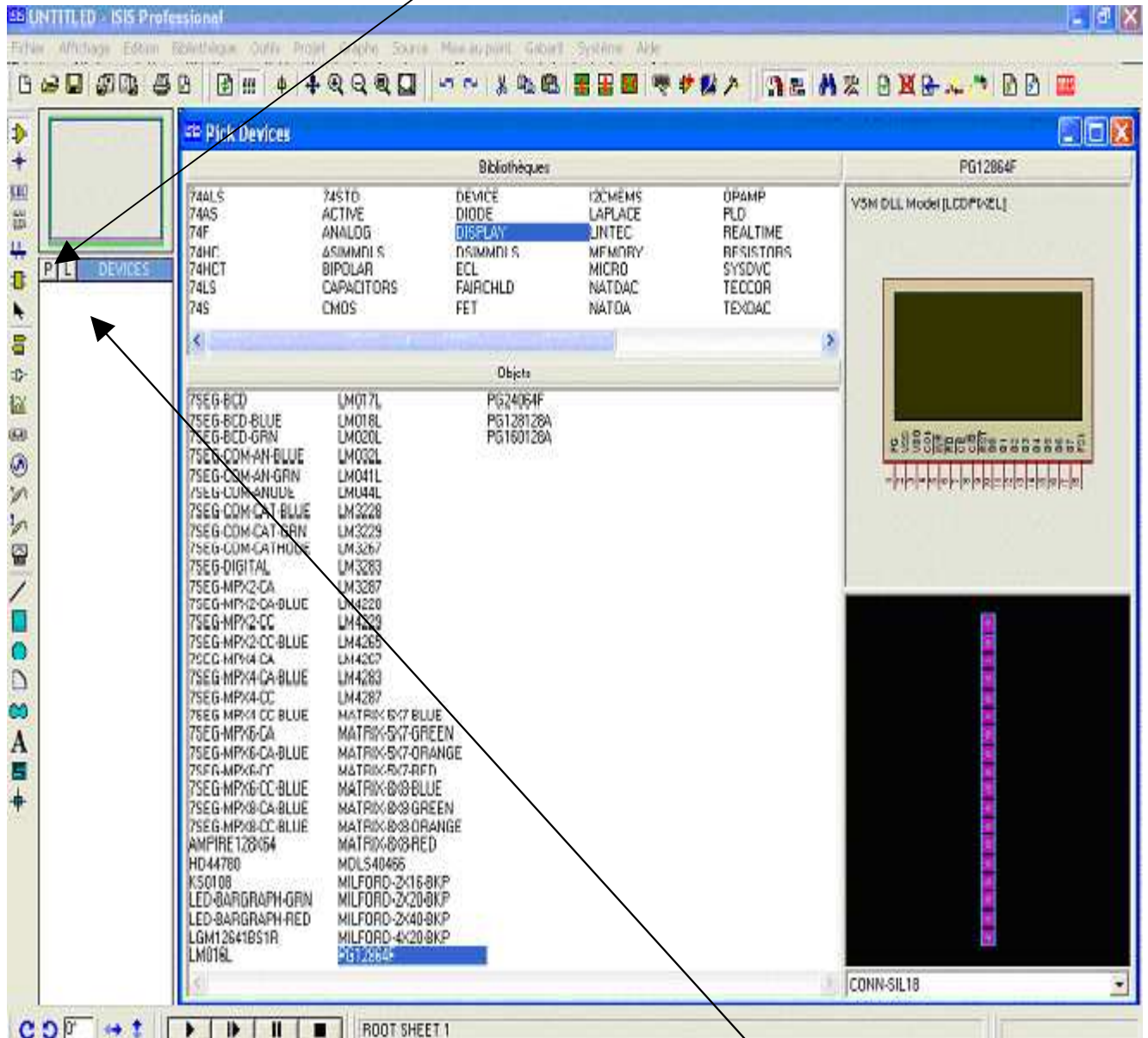
Présentation et utilisation d'IRIS



ISIS est livré avec des bibliothèques complètes de composants standards organisées en familles de type TTL, CMOS, ECL, microprocesseurs, mémoires et circuit intégrés analogiques type ampli-ops etc... D'autres bibliothèques comprennent des centaines de composants nommés de type Bipolaires, FET, MOSFET, Diodes. Soit plus de 8000 éléments au total.

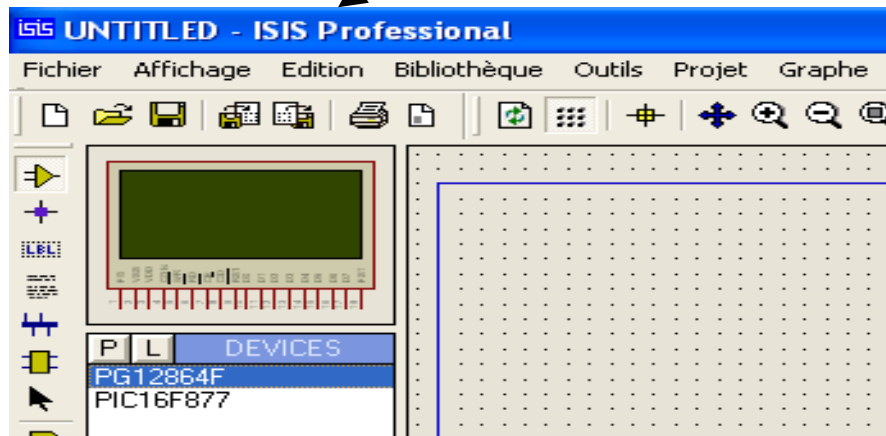
Une fois le logiciel PROTEUS installé, on lance ISIS et on choisit les composants nécessaires dans la bibliothèque des composants fournies avec ISIS (**Pick Devices**).

Pour cela on clique sur le bouton **P..**

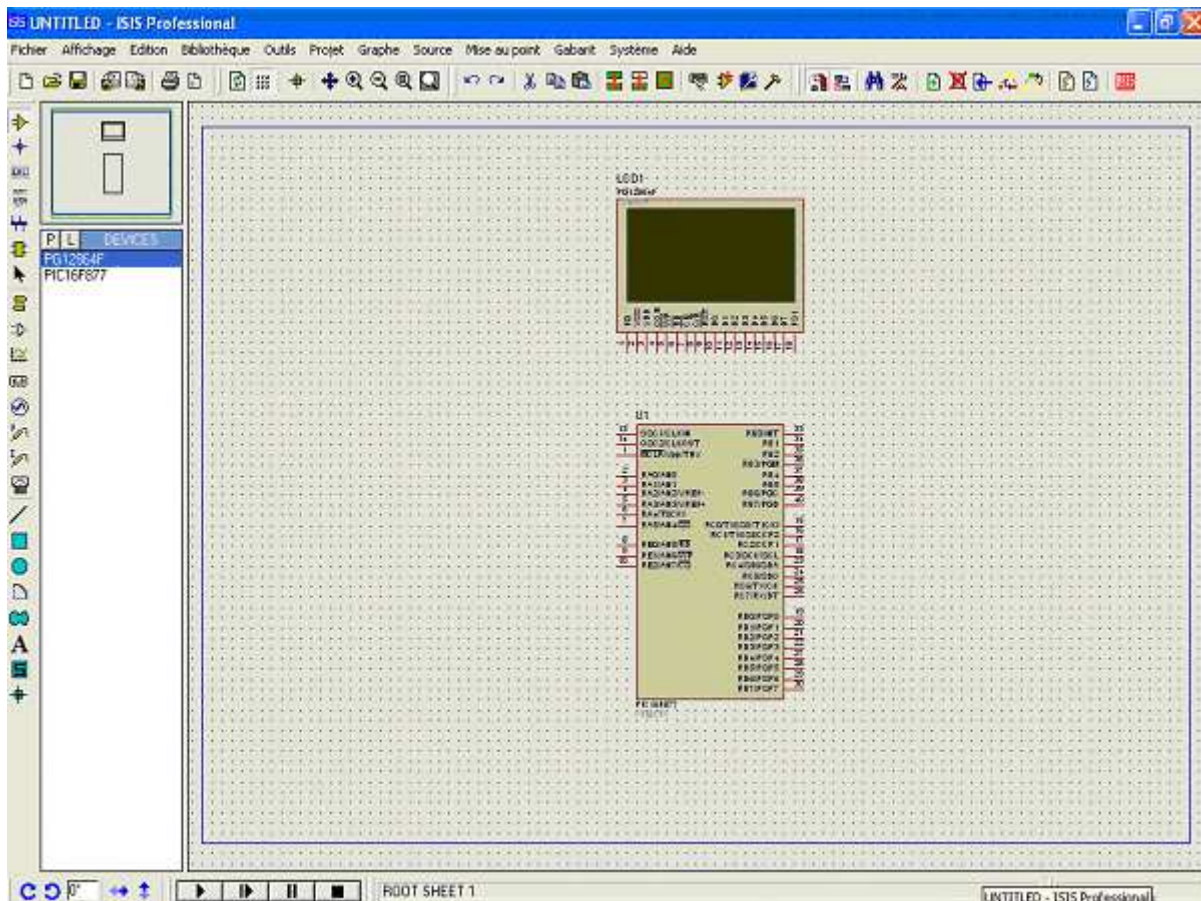


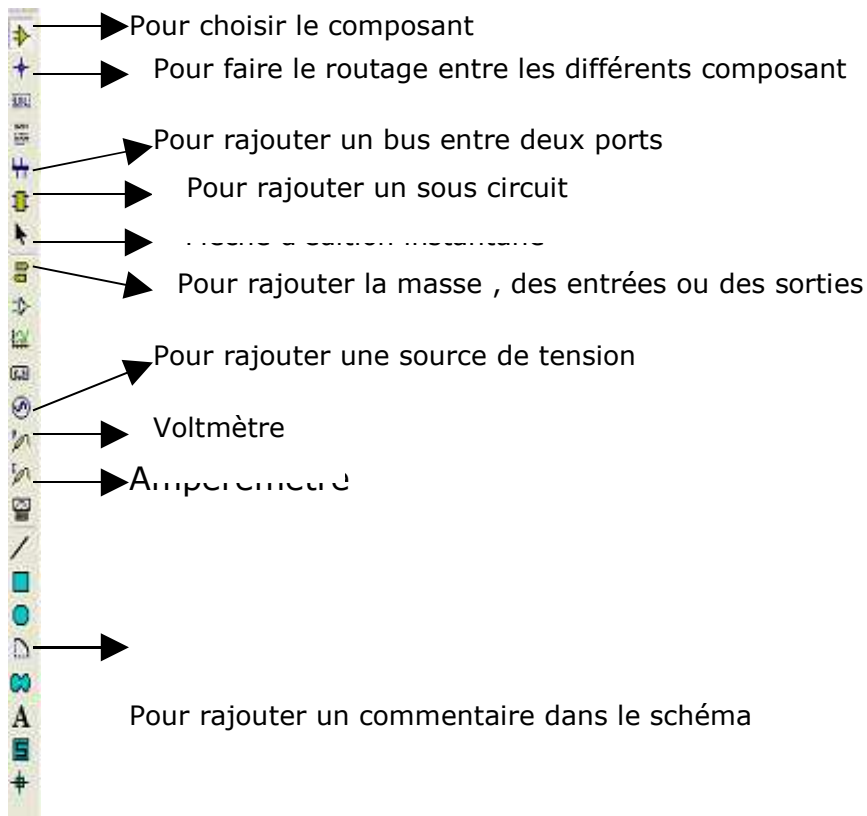
Ensuite en clique sur le composant deux fois pour qu'il s'affiche dans la fenêtre blanche à gauche .

Après avoir rajouter toutes les composants nécessaire pour le schéma on ferme la fenêtre **Pick Devices** on clique sur le composant dans la fenêtre de gauche une seule fois , le composant doit apparaître cette fenêtre

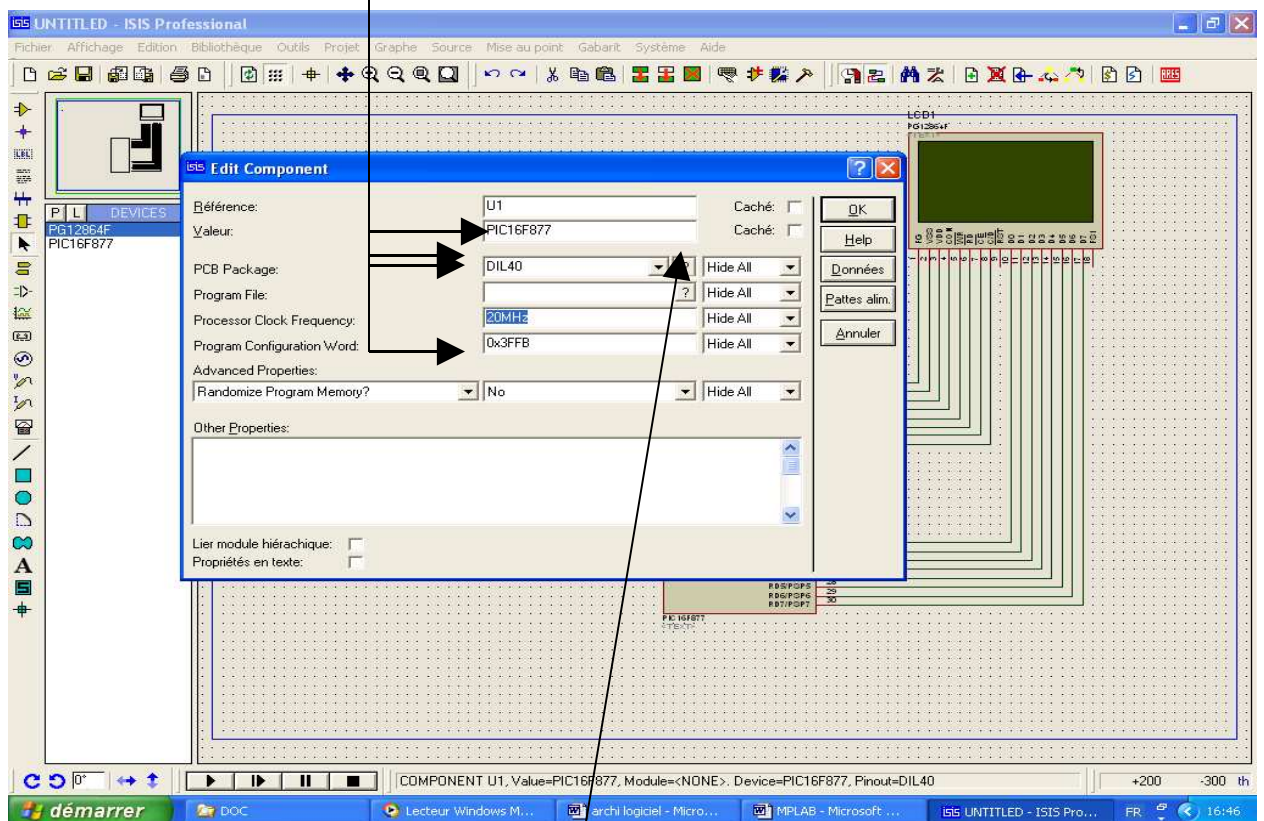


Ainsi on déplace le composant un par un en utilisant la souris.





Une fois le schéma est édité, on sélectionne la flèche d'édition instantané ensuite on clique sur chaque composants, la fenêtre **Edit Component** apparaît pour modifier ses caractéristiques.

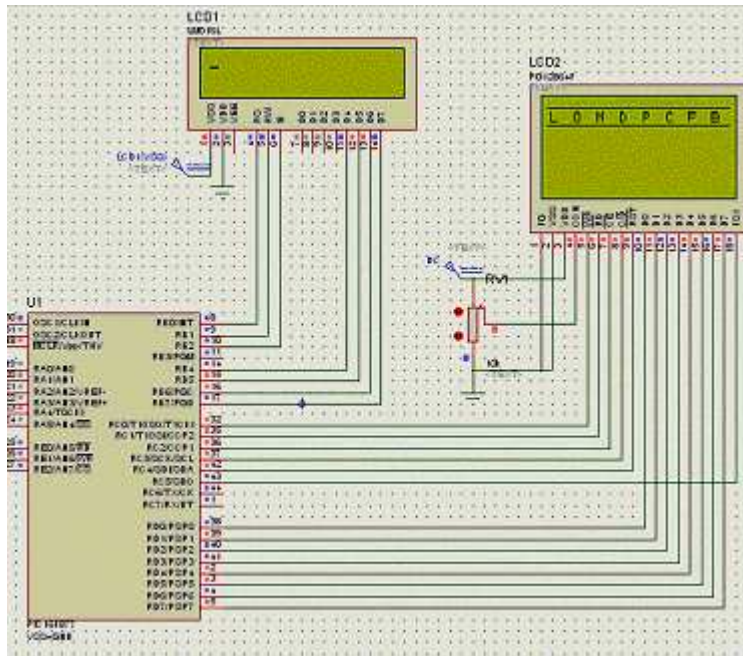


A la fin on clique sur la fenêtre **Program file** pour charger le programme en Hexa généré précédemment par Mplab.

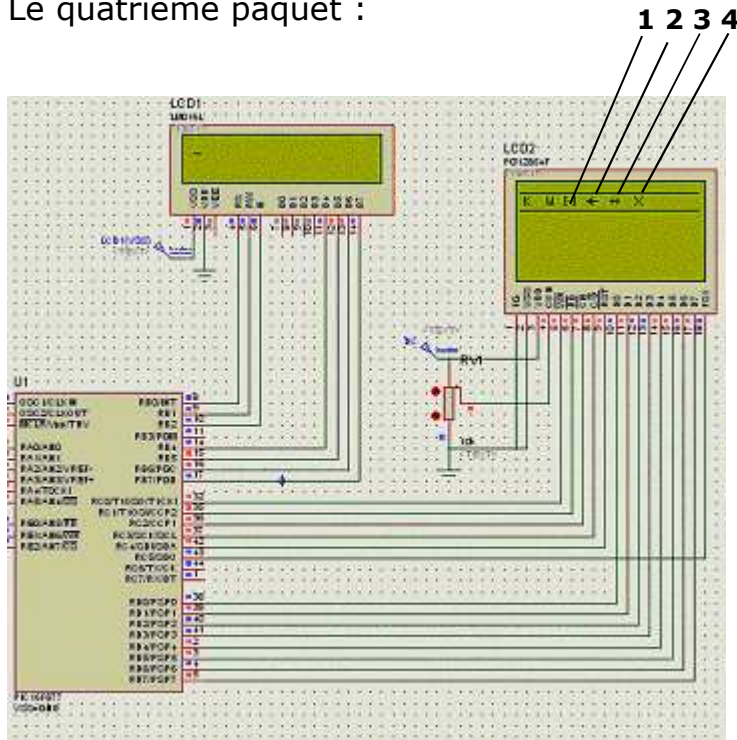
ANNEXE 6

ANNEXE 7

Le deuxième paquet :

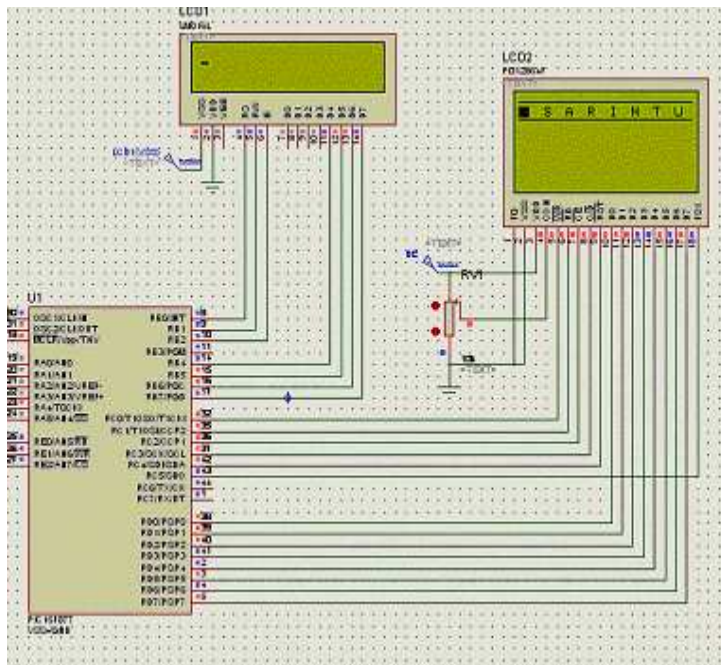


Le quatrième paquet :



- 1 : Espace.
- 2 : efface un caractère.
- 3 : Efface un mot
- 4 : Efface le texte.

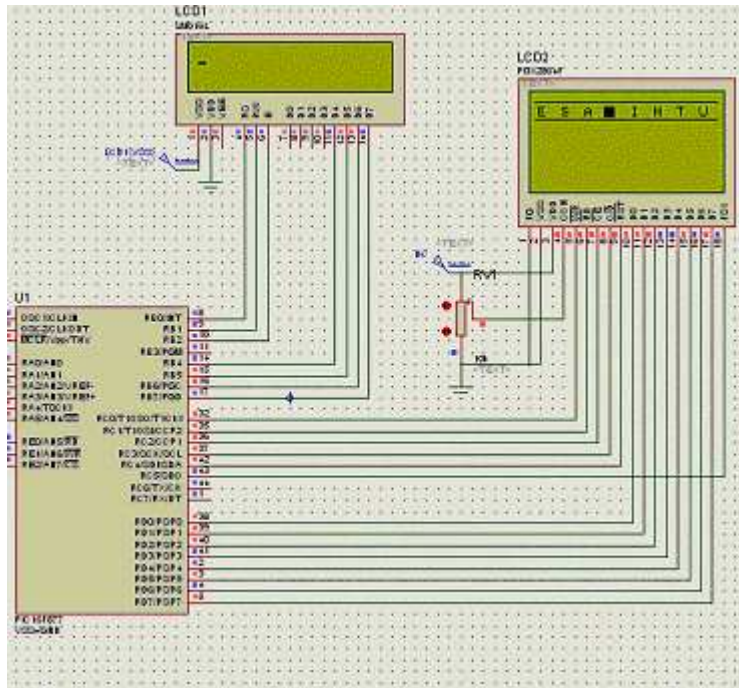
Ici on visualise le déplacement de curseur d'un caractère à l'autre jusqu'au caractère qu'on veut taper et on le valide par le bouton poussoir.



EXEMPLE DE SAISI DE TEXTE :

On prend l'exemple de saisir le nom **IUP GEII**

Le premier paquet contient le caractère **I** qu'on veut saisir donc dès l'apparition de ce paquet on appui sur le bouton poussoir et le curseur se déplace .



Dès que caractère I clignote on appui une deuxième fois sur le bouton poussoir et on visualise ce caractère dans la zone de texte et ainsi de suite jusqu'à la fin de la saisie de notre phrase.

