INSTITUT UNIVERSITAIRE DE TECHNOLOGIE

Département Génie Electrique
et
Informatique Industrielle

RAPPORT DE STAGE

LMP Laboratoire de Micro-électronique de Puissance

ACQUISITION ET TRANSMISSION DE DONNÉES NUMERIQUES PAR LIAISON WIFI POUR KARTS ELECTRIQUES

HAHN Jean-Charles
Groupe B2
Promotion 2004-2006

BIDOGGIA Benoît
LEQUEU Thierry
BRUN Jacky

Remerciements

Premièrement, je tiens à remercier Monsieur Laurent Ventura, directeur du LMP, pour m'avoir accueilli dans son laboratoire.

Je remercie également Monsieur Benoît Bidoggia — maître de stage et doctorant au sein du LMP, pour son accueil et son aide ; il m'a enseigné les bases du microcontrôleur et la programmation en assembly, en m'accompagnant tout au long du stage — et Monsieur Thierry Lequeu — professeur tuteur et enseignant-chercheur au sein du LMP, pour son soutient et son aide durant le stage —. Tous deux ont contribué au bon fonctionnement du travail effectué.

Enfin, Monsieur Jean-Charles Lebuntel, Monsieur Fisal Alkayal et Monsieur Jacky Brun pour les réponses aux difficultés que j'ai pu avoir.

TABLE DES MATIÈRES

LISTE DES FIGURES

INTRODUCTION

Dans le cadre de notre formation, un stage de fin d'études est effectué. D'une durée de 11 semaines, il nous permet de valider le Diplôme Universitaire de Technologie.

Durant ce temps, j'ai travaillé au LMP (Laboratoire de Micro-électronique de Puissance), se situant au Département Productique de l'Ecole Polytechnique de Tours.

Monsieur Benoît Bidoggia était mon maître de stage et Monsieur Thierry Lequeu, mon professeur tuteur. Tous deux m'ont soutenu et m'ont aidé à prendre les bonnes décisions afin de mener à bien le travail qui m'a été confié.

Le but de ce projet est de pouvoir récupérer des données numériques d'un kart électrique. Des valeurs, comme la tension du moteur, la vitesse, le courant de batterie, la température du moteur et du variateur seront relevées, affichées et transmises à un poste fixe par Wi-Fi. Ceci permettra de « contrôler » le comportement du kart évoluant sur une piste, jusqu'à une distance d'environ cent mètres.

Une présentation de l'environnement de travail s'effectuera premièrement. Ensuite, le cahier des charges sera précisément défini, pour ainsi laisser la place aux détails de la carte électronique réalisée. Enfin, la partie programmation sera expliquée.

CHAPITRE 1

GÉNÉRALITÉS

1.1. **Le Laboratoire de Micro-électronique de Puissance**

1.1.1. *Présentation*

Le LMP a été fondé le 18 décembre 1996 suite à la création du pôle Micro-électronique de Puissance de Tours.

Monsieur *Laurent Ventura*, nommé à la direction des recherches, est le directeur du laboratoire dont le siège se situe dans l'entreprise STMicroelectronics Tours. Il existe aussi deux autres sites : un à l'Ecole Polytechnique de l'Université de Tours, l'autre à l'IUT GEII de Tours. Le pôle Micro-électronique de Puissance, constitué sur un partenariat très fort entre l'Université François Rabelais et la société STMicroelectronics, a pour vocation de réunir les potentialités de la recherche universitaire et industrielle sur le thème de la micro-électronique de puissance.

1.1.2. *Domaines d'activités et partenaires*

L'activité principale du LMP est l'intègration de systèmes et de dispositifs d'électroniques de puissance sur plaquette. Pour se faire, il possède les compétences et les équipements pour, notamment, la réalisation de briques technologiques sur silicium, la caractérisation physique, la simulation et la mesure de fiabilité de composants électroniques, mais aussi pour la caractérisation électrique de composants sur plaquette, la caractérisation fonctionnelle de composants, ainsi que pour le développement de microsystèmes à applications électroniques et médicales.

Pour l'aider dans ces recherches, le LMP est en collaboration avec différents laboratoires tels que : PHASE (Strasbourg), LEG (Grenoble), CEMES (Toulouse), LAAS (Toulouse), DMAT CEA (Ripault), LEMA (Tours) et le LMR (Tours). Mais il a surtout des partenaires industriels comme SAFT (Chambray), VERMON (Tours) et STMicroelectronics (Tours) qui est son principal partenaire.

1.1.3. *Moyens humains et organigramme*

L'effectif est de 24 personnes dont : 10 enseignants chercheurs, 10 doctorants, 2 post-doctorants et 1 secrétaire.

Cet effectif est divisé en deux équipes (fig. 1.1), l'une traitant des technologies des semi-conducteurs, tests et fiabilité (MTEC) et l'autre de l'analyse des sytèmes de conversion d'énergie (COSYS).

1.2. **Cahier des charges**

Le problème global est qu'une station mobile affiche des données et les envoie à une station fixe, respectivement représentées par un kart électrique et un ordinateur.

Les étapes sont les suivantes :

– récupérer les données ;

– les traiter ;

– les afficher ;

– les envoyer.

1.2.1. *Les données*

Le but est d'exploiter en temps réel différentes caractèristiques d'un kart électrique durant son évolution sur une piste.

Huit différentes mesures peuvent être relevées :

la vitesse du kart :
: un capteur à effet Hall permet de la récupérer ;

le courant moteur :
: la mesure du courant dans le moteur utilise un capteur de courant HAS-200 ; il est alimenté en +15 / -15 V et délivre une tension proportionnelle au courant dans la fenêtre — 5 fois le courant moteur — avec un calibre de 0,1 V/A. La tension peut être positive — +5,2 V pour +52 A en moteur — ou négative — -4,7 V pour -47 A en freinage — ;

la position de la pédale de frein et de l'accélérateur :
: un potentiomètre permet de connaître l'image de la position des deux pédales ;

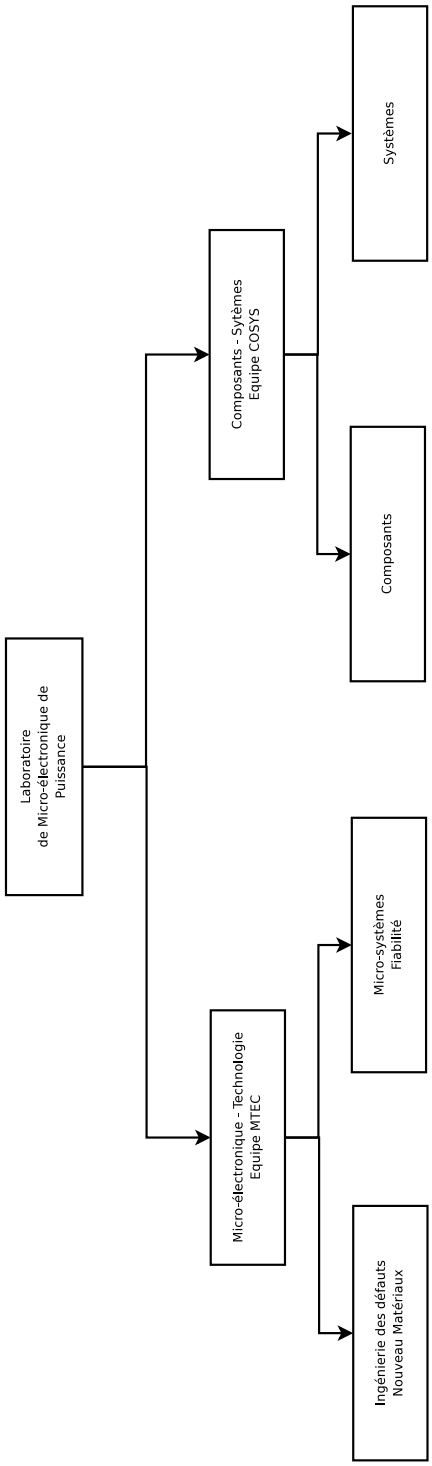

Figure 1.1. Organigramme du LMP

la tension de batterie :
: un pont diviseur constitué de deux résistances ramène la tension de la batterie à une valeur comprise entre 0 et 5 V ;

le courant de batterie :
: il est prévu d'utiliser la même application que pour le courant moteur ;

la température du moteur et du variateur :
: lors du fonctionnement du kart, la température ne sera ni inférieure à zéro, ni supérieure à soixante degrès. Le choix de composant pour cette mesure sera une sonde de température LM35 qui fonctionne linéairement dans les gammes de température.

1.2.2. *Acquisition des données*

Le microcontrôleur choisi est le PIC16F737 de Microchip, qui intègre entre autres un convertisseur analogique/numérique. Celui-ci est relié à un module d'envoi Wi-Fi au travers d'un port série standard (type RS-232) : le EZL-80c. Les deux composants possèdent différents modes de traitement de l'information et de veille, et ils peuvent se déclencher par une réception de données sur le réseau Wi-Fi ou sur le port série.

1.2.3. *La liaison*

La liaison Wi-Fi est la communication idéale pour la transmission : elle a deux grands avantages. D'une part, grâce à l'autoroutage, elle est capable d'allouer elle-même ses fréquences de fonctionnement ; ce système standard pourra s'adapter avec d'éventuelles évolutions du système. D'autre part, elle demande un accusé de réception à chaque donnée envoyée, ce qui assure une bonne fiabilité des informations échangées.

Pour configurer le mode de transmission Wi-Fi deux programmes sont à disposition : « ezSerialConfig » et « ezConfig ». Le premier se connecte par le port série du module. Il faut préalablement débrancher la carte RF de celui-ci avant la mise sous tension pour configurer le EZL-80c. Ensuite, le deuxième communique avec le module par liaison Wi-Fi. Ces deux programmes permettent de configurer la liaison série : son type, son débit, sa parité… Ils configurent de même le réseau Wi-Fi : serveur/client, clé WEP (protection)…

Le logiciel « ezTCP » crée un port série virtuel qui est détecté automatiquement par l'ordinateur, ce qui permet la communication entre la carte Wi-Fi de l'ordinateur et le programme réalisé en C++. Cela permet de réaliser une interface graphique plus facilement.

1.2.4. *Affichage*

Un écran LCD quatre lignes, seize caractères par ligne, géré par le contrôleur standard HD44780 de Hitachi, permet d'afficher les valeurs acquises selon le bon vouloir de l'utilisateur.

CHAPITRE 2

PARTIE *HARDWARE*

2.1. **Alimentation**

Nous devons alimenter les composants sous deux niveaux de tension. Pour la plupart utilisés, ils doivent être alimentés en +5 V. En ce qui concerne le composant Wi-Fi EZL-80c, il doit être alimenté en +3,3 V.

Deux régulateurs sont utilisés : le LM2575T-5 fait le +5 V (fig. 2.1), et le LM2595T-3.3 fait le +3,3 V (fig. 2.2), à partir d'un niveau de tension allant jusqu'à 40 V.

Par rapport à une alimention « linéaire », ces deux composants sont des régulateurs à découpage, ayant donc un rendement nettement supérieur. Les condensateurs présents sur dans les circuits d'alimentation doivent être de bonne qualité et pouvant supporter un niveau de tension élevée. Ils ont pour rôle de filtrer un maximum de parasites.

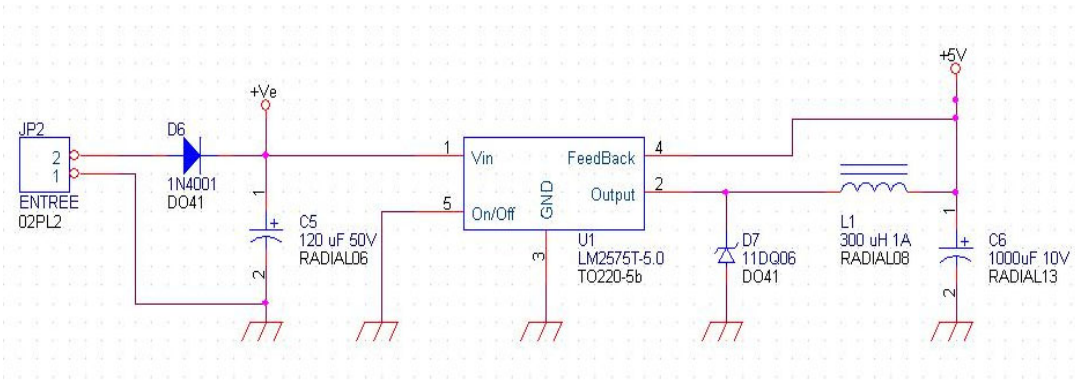

Figure 2.1. Schéma électrique de l'alimentation 5 V

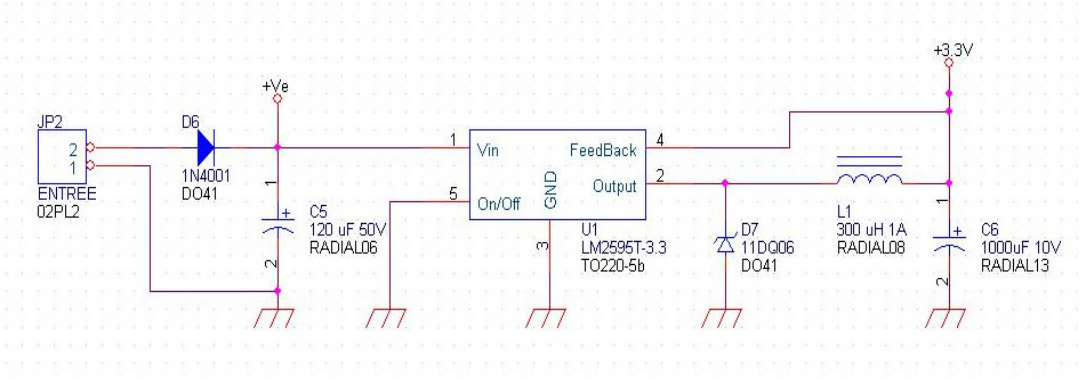

Figure 2.2. Schéma électrique de l'alimentation 3,3 V

2.2. **Microcontrôleur**

Le microcontrôleur choisi est le PIC16F737 de Microchip. Les entrées/sorties utilisées sont les suivantes :

entrées :

- huit entrées analogiques pour les huits points de mesure ;
- quatre entrées correspondant à quatre boutons poussoirs ;
- une entrée « RX » pour le canal de réception du port série ;

sorties :

- huit sorties pour commander l'afficheur LCD ;
- une sortie « TX » pour le canal de transmission du port série ;

spécifiques :

- l'entrée « MCLR » branchée au circuit de *Reset* (fig. 2.4) ;
- l'entrée « OSCIN » et la sortie « OSCOUT » du circuit d'oscillation, reliées au quartz de 12 MHz (fig. 2.3) ;
- l'alimentation entre « VSS » (masse) et « VDD » (5 V).

2.3. **Afficheur LCD**

Un afficheur LCD est composé d'un écran et d'un module de contrôle. Le module utilisé ici est un HD44780, l'un des plus commun. Ceci est important car la programmation effectuée tout au long du projet n'est pas compatible avec tous.

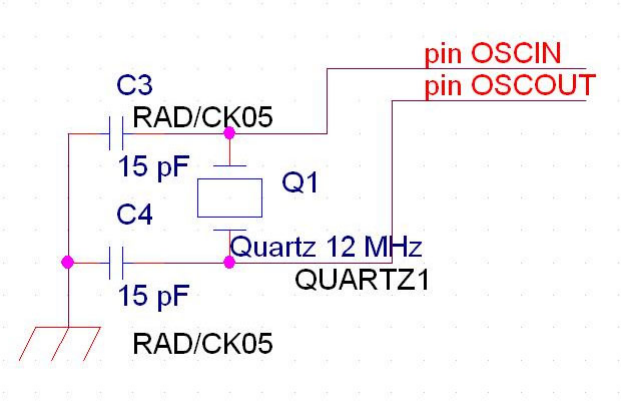

Figure 2.3. Schéma électrique du quartz

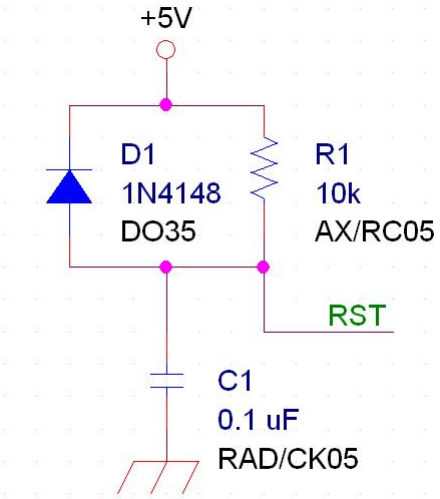

Figure 2.4. Schéma électrique du RESET

Le connecteur de l'afficheur LCD peut être divisé en quatre parties (fig. 2.5). Tout d'abord, il y a les pins 1, 2 et 3 qui sont respectivement la masse (GND), l'alimentation +5 V (VCC) et le contraste (VEE). Ce dernier est relié à une résistance variable qui sert à règler le contraste : R-TAPPED.

Les pins 4 à 14 — respectivement E, RS, R/W, D0 à D7 — constituent le bus de communication. Grâce à la broche R/W nous pouvons dire au module programmable que nous allons soit lire sur l'écran (*Read*), soit écrire dessus (*Write*). Dans notre projet, nous configurerons cette pin pour écrire sur l'écran.

Enfin, pour les afficheurs LCD ayant cette option, les pins 15 et 16 (LED-A et LED-K) ont pour rôle de règler la luminosité de l'écran. Ceci n'étant pas obligatoire, les afficheurs ne possèdant que 14 pins seront compatibles.

L'afficheur dispose de deux protocoles de communication possibles : soit en lui envoyant les données binaires sur les 8 broches D0 à D7, soit en les envoyant en deux fois sur les broches D4 à D7. Nous choisirons la deuxième solution dans le but d'économiser les pins du microcontrôleur. Les broches D4 à D7 seront donc reliées respectivement aux pins RB4, RB0, RC5 et RC4 du PIC, et les broches D0…D3, n'étant pas utilisées, sont reliées à la masse au travers de resistances de 10 KOhms.

2.4. **Module Wi-Fi**

Le module Wi-Fi (*Acquisition des données*, page 5), est composé de deux parties : une carte RF et le composant EZL-80c.
Seulement 4 pins seront utilisées au cours de nos essais (voir fig. 2.6) :

- Vcc : l'alimentation à 3,3 V ;
- GND : la masse ;
- Rx : le canal de réceptiondu module Wi-Fi ;
- Tx : le canal de transmission du module Wi-Fi.

Les niveaux TTL pour la réception ainsi que pour la transmission sont compris entre 0V et +3.3 V. Le PIC envoyant des données de niveau TTL de +5 V, la connexion directe n'est pas possible.

Pour cela, une diode Zener (1N5518) transformant le niveau TTL +5 V en +3.3 V a été mise en place pour la transmission du PIC vers le EZL-80c. La réception n'est pas traitée, le microcontrôleur ne reconnaissant pas un niveau TTL de +3.3 V. De plus, les seules données que renvoient le module Wi-Fi ne sont que des accusés de réception pour avertir le microcontrôleur que l'envoi en Wi-Fi s'est bien déroulé.

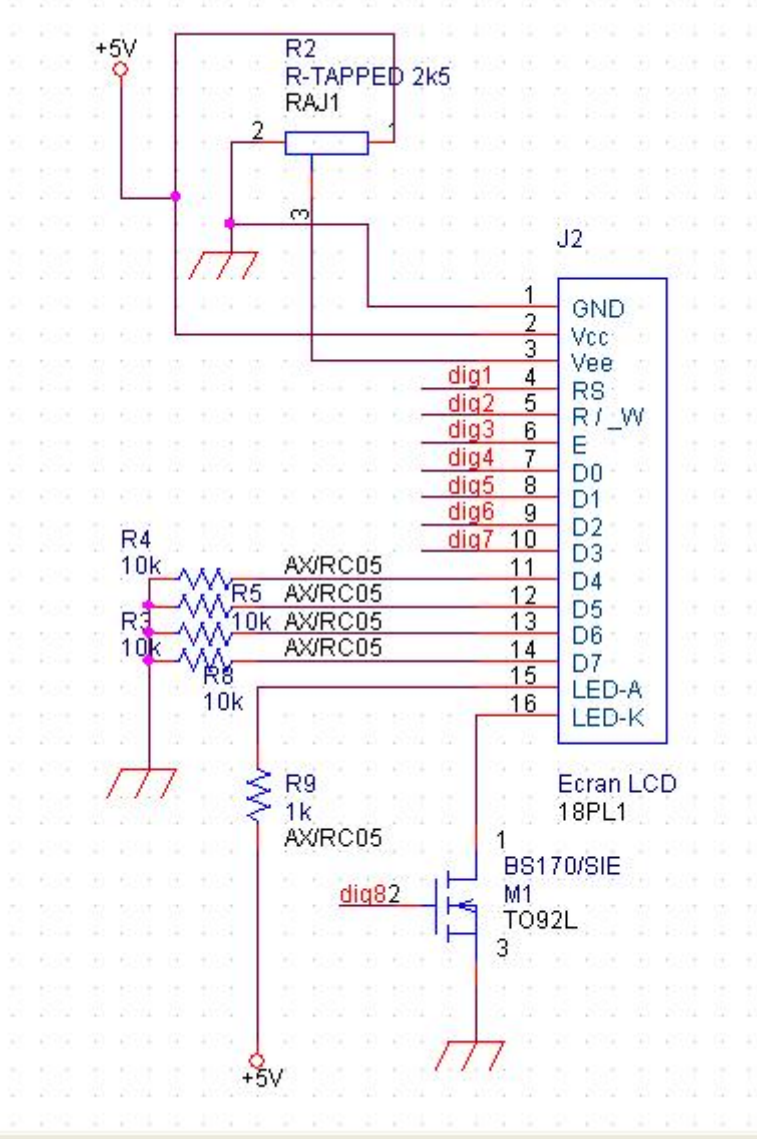

Figure 2.5. Schéma électrique de l'implémentation de l'afficheur LCD

Le MAX3373 de MAXIM paraît convenir selon les caractèristiques observées sur sa documentation technique. Néanmoins, il n'a pas été testé pour l'adaptation de ces différents niveaux de tension.

Pour configurer le composant EZL-80c, il a fallu réaliser une carte électronique à part : avant d'entreprendre la liaison Wi-Fi entre le module et l'ordinateur, il faut configurer par liaison série RS-232, ce composant. Ainsi, il faut que l'on utilise comme élément principal un MAX3232 de MAXIM qui transforme les niveaux TTL en niveau RS-232, allant de -12 V à +12 V, et inversement.

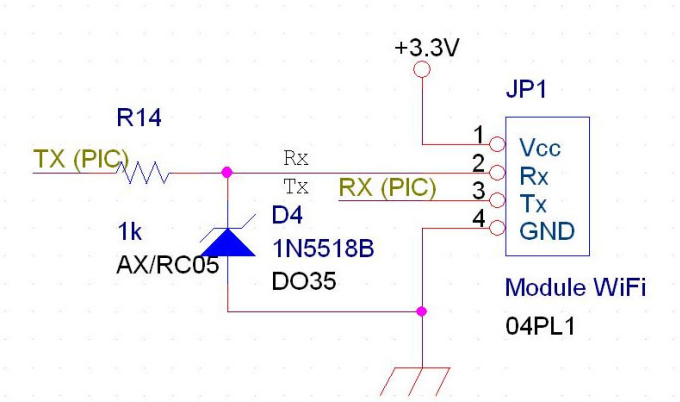

Figure 2.6. Schéma électrique de l'implémentation du module Wi-Fi

2.5. **Problèmes rencontrés**

En répétant maintes fois les contrôles de la programmation, nous ne comprenions pas pourquoi l'afficheur n'exécutait pas ce qu'on lui demandait. En supposant que l'écran soit endommagé, un deuxième fut apporté : il fut essayé et les résultats furent concluant. Le programme fonctionnait et s'exécutait normalement. Cependant, l'écran précédent fut réessayé et finalement marcha. Le problème était situé au niveau du connecteur de l'afficheur qui produisait un faux contact. Finalement, les connecteurs furent changés, bien soudés et la carte s'est mise à fonctionner correctement.

CHAPITRE 3

PARTIE *SOFTWARE* EMBARQUÉ

3.1. Configuration générale du microcontrôleur

3.1.1. *Organisation de la mémoire*

La mémoire RAM du microcontrôleur est divisée principalement en quatre parties qui se nomment des *banks* (bank0 … bank3). Ces banks sont elles-mêmes constituées de **registres** de 8 **bits**. Chacune des banks ont 128 octets, en adresse héxadécimale : de 0h à 7Fh. Elles contiennent toutes des registres fonctionnels et des registres généraux.

Les registres fonctionnels sont utilisés par le PIC et par les périphériques pour effectuer différentes opérations. Certains servent à configurer les broches pour en faire des entrées ou des sorties, d'autres permettent de configurer la vitesse de traitement des données en fonctions des périphériques… Tous les registres ne sont pas forcément utilisés, comme ici, où des options du microcontrôleur ne sont pas utiles : par exemple les interruptions.

Les registres généraux sont en fait des emplacements « vides » de la mémoire pour y stocker nos valeurs.

3.1.2. *Les registres utilisés*

Chaque broche du microcontrôleur peut avoir différentes fonctions : la pin 26, par exemple, peut être configurée comme étant une sortie ou une entrée, analogique ou numérique, une sortie PWM[1] ou même avoir le rôle de comparateur.

Chaque fonction est règlable dans un registre spécifique. Cependant, tous les registres ne sont pas à configurer. Dans notre cas, nous n'en utilisons qu'un petit nombre. Détaillons-les.

[1]Pulse Width Modulation

3.1.2.1. Les registres PORTx, TRISx

Les registres PORTx représentent les broches entrées/sorties du microcontrôleur :

- PORTA : pins RA0, RA1 … RA7
- PORTB : pins RB0, RB1 … RB7
- PORTC : pins RC0, RC1 … RC7

Les registres TRISx configurent ces broches soit en entrée (bit correspondant à 1), soit en sortie (bit correspondant à 0). Configuration du programme :

- TRISA : b'11011111'[1]
- TRISB : b'00001110'
- TRISC : b'00001111'

3.1.2.2. Le registre ADCON2

Ce registre est important en ce qui concerne la conversion analogique/numérique. Il est composé de 3 bits et se situe en Bank1.

Structure du registre ADCON2							
—	—	ACQT2	ACQT1	ACQT0	—	—	—

ACQT2 … ACQT0 : A/D Acquisition Time Select Bits

Ces bits permettent de déterminer le nombre de temps T_{ad} nécéssaires pour faire la conversion analogique/numérique. T_{ad} est le temps de conversion d'un bit, appelé temps d'acquisition. Plus de détails sont dans la partie *Le registre ADCON1*, page 15.

[1]Cette configuration fait que les pins RA0 … RA4, RA6 et RA7 sont des entrées et que RA5 est une sortie

ACQT2	ACQT1	ADCS0	Temps d'acquisition
0	0	0	0, Osc RC
0	0	1	2
0	1	0	4
0	1	1	6
1	0	0	8
1	0	1	12
1	1	0	16
1	1	1	20

Dans notre programme, la configuration de ADCON2 est : b'00101000' (12 T_{ad}).

3.1.2.3. Le registre ADCON1

Il permet de déterminer le rôle de chacune des pins AN0 à AN13 (sans AN5, AN6 et AN7, qui ne font pas parti de ce type de PIC), c'est à dire de choisir si une pin sera utilisée comme entrée analogique, comme entrée/sortie numérique ou comme tension de référence. Il permet également de décider de la justification du résultat. ADCON1 se situe en Bank1.

Pour pouvoir utiliser une pin en mode analogique, il faut d'abord la configurer en entrée via les registres TRISx. Le registre ADCON1 dispose, comme tout registre accessible de notre PIC, de 8 bits :

Structure du registre ADCON1							
ADFM	ADCS2	VSFG1	VCFG0	PCFG3	PCFG2	PCFG1	PCFG0

ADFM : A/D result Format Select Bits

Le bit ADFM (b7) permet de déterminer si le résultat de la conversion sera justifié à droite (1) ou à gauche (0).

ADCS2 : A/D Clock Divide by 2 Select Bit

Le bit ADCS2 complète 2 autres bits situés dans le registre ADCON0.

VCFG1, VCFG0 : Voltage Reference Configuration

VCFG0 et VCFG1 nous permettent de dire au microcontrôleur si les références de tension VREF- et VREF+ sont connectées à des sources extérieures (1) ou si elles sont connectées respectivement à VSS et VDD (0).

PCFG3 … PCFG0 : Port Configuration Control

PCFG0,…, PCFG3 nous permettent de déterminer le rôle de chaque pin : entrée analogique ou entrée/sortie numérique. Comme nous avons 16 combinaisons possibles, nous aurons autant de possibilités de configuration.

La première colonne contient les 16 combinaisons possibles des bits de configuration PCFG3 à PCFG0. Les colonnes « AN13 à AN0 » indiquent le rôle qui sera attribué à chacune des pins concernées : un « A » dans une de ces colonnes indique que la pin correspondante est configurée comme entrée analogique, un « D » indiquera que la pin est en mode *Digital,* c'est-à-dire qu'elle se comportera comme une pin d'entrée/sortie « numérique ». Nous remarquons que si nous avons le choix du nombre de pins configurées en entrée analogique, nous n'avons cependant pas le choix de leur attribution. Par exemple, si nous avons besoin de configurer ces ports pour disposer de 3 entrées analogiques et de 5 entrées/sorties numériques, nous devons mettre en place la réalisation de la carte électronique suivant la configuration qui peut être faite, en y tenant compte au moment de concevoir le schéma électrique. Lors d'une mise sous tension, les bits PCFGx contiennent 0000. PORTA, PORTB et PORTC seront donc configurés par défaut comme ports complètement analogiques. Ceci explique pourquoi l'utilisation de ces ports comme ports d'entrées/sorties classiques implique d'initialiser ADCON1.

3.1.2.4. Le registre ADCON0

Il est le dernier registre utilisé par le convertisseur analogique/numérique. Il contient les bits que nous allons manipuler lors de notre conversion. ADCON0 se situe en Bank0.

Structure du registre ADCON0							
ADCS1	ADCS0	CHS2	CHS1	CHS0	GO/DONE	CHS3	ADON

ADCS1, ADCS0 : A/D Conversion Clock Select Bits

Ces bits, complètés par ADCS2, permettent de déterminer l'horloge du convertisseur

CHS3 … CHS0 : Analog Channel Select bits

CHS3,…, CHS0 nous permettent de sélectionner un canal pour la conversion

GO/DONE : A/D Conversion Status Bit

GO/DONE permet de lancer la numérisation

ADON : A/D conversion status bit

Le bit ADON permet de mettre en service le convertisseur.

Nous avons précédemment parlé de diviseur, afin de déterminer l'horloge du convertisseur en fonction de la fréquence du quartz utilisée.

ADCS2	ADCS1	ADCS0	Diviseur	Fréquence maximale du quartz
0	0	0	$\frac{Fosc}{2}$	1,25 MHz
0	0	1	$\frac{Fosc}{8}$	5 MHz
0	1	0	$\frac{Fosc}{32}$	20 MHz
0	1	1	Osc RC	Si > 1 MHz
1	0	0	$\frac{Fosc}{4}$	2,5 MHz
1	0	1	$\frac{Fosc}{16}$	10 MHz
1	1	0	$\frac{Fosc}{64}$	40 MHz
1	1	1	Osc RC	Si > 1 MHz

Le bit ADCS2 ne fait pas parti de ce registre, mais de ADCON1.

Nous avons vu que nous pouvons configurer, via ADCON1, plusieurs pins comme entrées analogiques. De plus, nous ne pouvons effectuer la conversion que sur une pin à la fois (appelée « canal »). Nous devons donc être en mesure de sélectionner le canal voulu. Ceci s'effectue via les bits CHSx.

Le bit ADON permet de mettre en service le convertisseur.

Quant au bit GO/DONE, il sera placé à « 1 » par l'utilisateur à la fin du temps d'acquisition. Cette action détermine le début de la conversion en elle-même. Une fois la conversion terminée, ce bit est remis à « 0 » par l'électronique du convertisseur. Cette remise à « 0 » est accompagnée du positionnement du bit ADIF du registre PIR1. Ce bit permettra éventuellement de générer une interruption.

3.1.2.5. Le registre TXSTA

Ce registre permet la configuration de la transmission de données, qui peut être soit synchrone, soit asynchrone. Il se situe en Bank1.

Structure du registre TXSTA							
CSRC	TX9	TXEN	SYNC	—	BRGH	TRMT	TX9D

CSRC : Clock Source Select Bit

Non-valide en mode asynchrone, il permet de déterminer si le PIC joue le rôle de « Maître[1] ou d'« Esclave[2]

TX9 : 9-bit Transmit Enable Bit

Il permet de sélectionner une transmission de 8 bits ou de 9 bits

TXEN : Transmit Enable Bit

Il permet de lancer la transmission

SYNC : AUSART[3] Mode Select bit

Il permet de configurer la transmission en mode synchrone ou asynchrone

BRGH : High Baud Rate Select Bit

Non-valide en mode synchrone, il permet de configurer la transmission en qualitée de « rapide[4] ou de « lente[5]

TRMT : Transmit Shift Register Status Bit

Il indique si le buffer de transmission est plein ou vide

TX9D : 9th Bit of Transmit Data

Il permet de configurer le neuvième bit, si souhaité, en bit de parité.

3.1.2.6. Le registre RCSTA

Ce registre permet la configuration de la réception de données. Il se situe en Bank1.

Structure du registre RCSTA							
SPEN	RX9	SREN	CREN	ADDEN	FERR	OERR	RX9D

SPEN : Serial Port Enable Bit

Il permet la validation du port série. Cela configure également les pins RC6 et RC7 respectivement en TX et RX.

---

[1]Master »

[2]Slave »

[3]Addressable Universal Synchronous Asynchronous Receiver Transmitter

[4]High Speed »

[5]Low Speed »

RX9 : 9-bit Receive Enable Bit
Il permet de sélectionner une réception de 8 bits ou de 9 bits

SREN : Single Receive Enable Bit
Il permet de valider la réception en mode « Maître »

CREN : Continuous Receive Enable bit
Il permet une réception continue

ADDEN : Address Detect Enable Bit
Valide seulement en mode asynchrone-9 bits, il permet de valider l'interruption et de charger le buffer de réception

FERR : Framing Bit Error
Il indique s'il y eu un problème durant la réception

OERR : Overrun Error Bit
il indique si le buffer de réception est plein

RX9D : 9th Bit of Received Data
Il permet de configurer le neuvième bit, si souhaité, en bit de parité.

3.1.2.7. Le registre SPBRG

Ce registre est configuré lors de la transmission de données. Il a pour rôle d'indiquer la vitesse de transmission en fonction du quartz. Il se règle suivant l'état du bit BRGH du registre TXSTA.

BRGH = 0 (vitesse lente)	**BRGH = 1 (vitesse rapide)**
$BaudRate = \frac{Fosc}{64 * (X + 1)}$ (mode asynchrone)	$BaudRate = \frac{Fosc}{16 * (X + 1)}$
$BaudRate = \frac{Fosc}{4 * (X + 1)}$ (mode synchrone)	N/A

« X » étant la valeur que l'on met dans le registre SPBRG (de 0 à 255). Des tableaux de références pour des valeurs de *Baud Rate*[1] sont dans la datasheet[2] du PIC.

[1]Vitesse de transmission

[2]Documentation technique

3.2. **Afficheur LCD**

L'afficheur LCD est composé de deux parties : un écran LCD et un module prgrammable standard HD44780 de Hitachi. Comme expliqué dans la partie *Afficheur LCD* (page 9), il existe deux modes de communication. Soit en envoyant en une seule fois les données sur les broches D0…D7, soit en les envoyant en deux fois successives sur les broches D4…D7. La deuxième solution semble mieux convenir pour une question d'économie des broches du microcontrôleur.

A la mise sous tensions, l'afficheur se met se paramètre par défaut :

1. Display clear
2. Function set :

DL = 1	8 - bit interface data
N = 0	1 - line display
F = 0	5 * 8 dot character font

3. Display on/off control :

D = 0	Display off
C = 0	Cursor off
B = 0	Blinking off

4. Entry mode set :

I/D = 1	Increment by 1
S = 0	No shift

L'afficheur est configuré par défaut sur l'envoi de données en 8 bits et non sur 4 bits comme nous le voulons. Un protocole pour le configurer en 4 bits est à suivre : figure 3.2.

Il faut suivre une démarche particulière afin d'envoyer des données (voir fig. 3.1). Il faut dans un premier temps, initialiser les bits dont on va se servir : RS à « 0 », R/W à « 0 » (mettre en mode écriture[1] l'afficheur), E à « 0 ». Ensuite, attendre un délai de 10 µs. Puis, on

charge les bits à envoyer sur les broches D7 à D4, on autorise l'envoi des données en mettant à « 1 » le bit E pendant 10 µs, on remet E à « 0 », délai de 10 µs. On charge les 4 derniers bits à envoyer, c'est à dire D3 à D0, on attend 10 µs, on autorise l'envoi pendant 10 µs, on met E à « 0 », délai de 10µs. Par précaution, on remet toutes les broches à « 0 » et on attend 2 ms pour pouvoir recommencer un autre envoi de données.

Le bit RS est mis à « 0 » pour avertir l'afficheur que nous lui envoyons une directive. Lorsque l'utilisateur veut envoyer des caractères, suivant la table ASCII de la documentation technique, RS doit être mis à « 1 ». L'envoi se déroule exactement de la même façon que pour les directives.

3.3. **Conversion Analogique / Numérique**

On désire mesurer une valeur analogique, c'est-à-dire, connaître la valeur de la tension présente sur une pin du PIC.

[1]Write

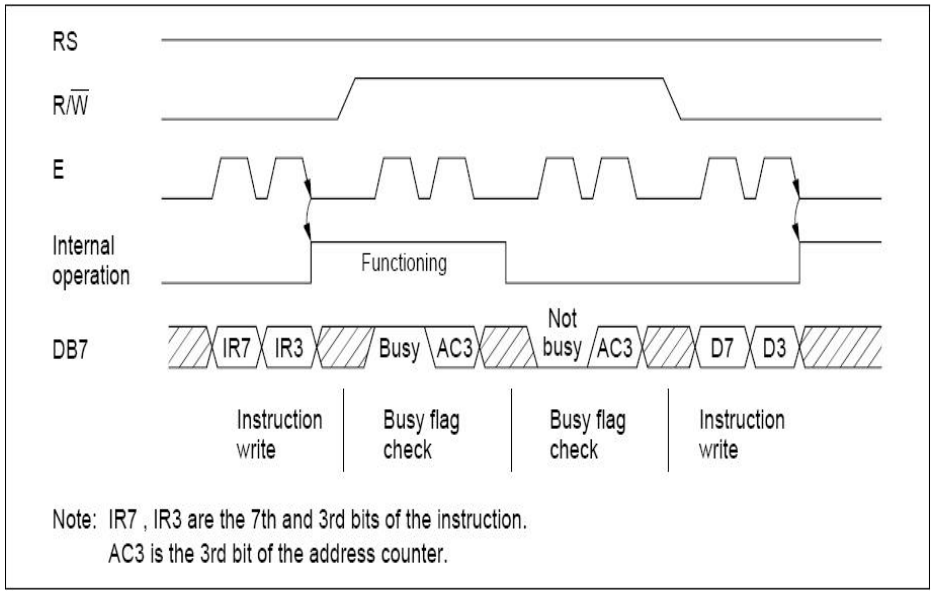

Figure 3.1. Démarche à suivre pour envoyer des données à l'afficheur

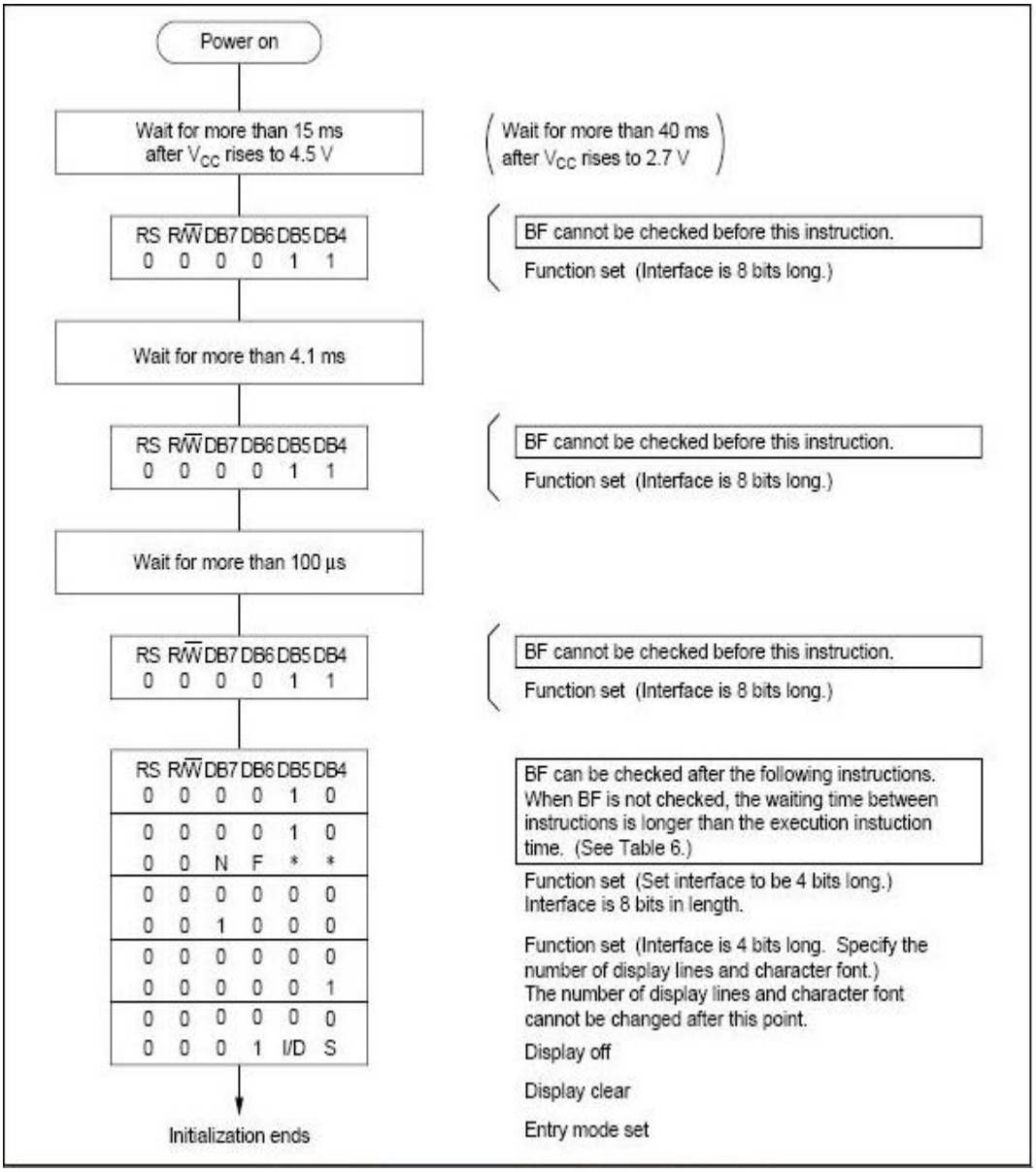

Figure 3.2. Protocole d'initialisation pour la configuration 4 bits de l'afficheur LCD

3.3.1. *Nombres numériques, analogiques et conversions*

Comme l'électronique interne du PIC ne comprend que les valeurs binaires, il faut transformer cette valeur analogique en une représentation numérique. Ce procédé

s'appelle numérisation. Pour l'effectuer, nous avons besoin d'un convertisseur analogique/numérique.

Nous réalisons en fait une double numérisation. La première consiste à découper la mesure en une succession de tranches, la seconde consiste à découper le temps en une autre succession de tranches.

Nous pouvons observer que nous perdons 2 fois en précision. D'une part, la valeur est arrondie en fonction du nombre de digits utilisés pour la conversion, et d'autre part, tous les événements survenus entre 2 conversions (échantillonnages) sont perdus. Nous pouvons en déduire que :

(1) plus l'on désire de précision, plus l'on doit augmenter le nombre de digits utilisés pour le résultat ;

(2) plus le signal évolue rapidement, plus l'on doit diminuer le temps séparant 2 échantillonnages, autrement dit, augmenter la vitesse d'échantillonnage.

3.3.2. *Principes de conversion*

Jusqu'à présent, nous avons raisonné en décimal. Les PICs travaillent en binaire. Pour convertir une grandeur, nous devons connaître la valeur minimale qu'elle peut prendre, ainsi que sa valeur maximale. Les PICs considèrent par défaut que la valeur minimale correspond à leur VSS d'alimentation (masse), tandis que la valeur maximale correspond à la tension positive d'alimentation VDD (+5 V).

La séquence pour convertir physiquement la grandeur analogique en grandeur numérique est la suivante :

– le PIC connecte la pin sur laquelle se trouve la tension à numériser à un condensateur interne, qui va se charger via une résistance interne jusque la tension appliquée ;

– la pin est déconnectée du condensateur et ce dernier est connecté sur le convertisseur analogique/numérique interne ;

– le PIC procède à la conversion.

Afin de savoir quelle est la fréquence maximale d'échantillonnage pour notre PIC, il faut déterminer le temps total nécessaire pour traiter les mesures. En tout premier lieu, il faut connaître le temps que le condensateur va mettre pour se charger. Ensuite, il faut évaluer le temps de la conversion.

3.3.3. *Le temps d'acquisition*

C'est le temps qu'il faut pour que le condensateur interne atteigne une tension proche de la tension à convertir. Cette charge s'effectue à travers une résistance interne et la résistance de la source.

Ce temps est incrémenté du temps de réaction des circuits internes et d'un temps qui dépend de la température (coefficient de température). Les résistances augmentent avec la température, donc les temps de réaction des circuits également.
On pose :

- T_{acq} : temps d'acquisition total ;
- T_{amp} : temps de réaction des circuits ;
- T_c : temps de charge du condensateur ;
- T_{coff} : temps qui dépend du coefficient de température.

La formule est :

$$T_{acq} = T_{amp} + T_c + T_{coff}$$

Le temps de réaction T_{amp} est de 2 µs :

$$T_{amp} = 2\,\mu s$$

Le coefficient de température n'est nécessaire que pour celles supérieures à 25 °C. Ce coefficient est de 0,05 µs par degrès qui est supérieur à 25 °C. Il s'agit bien entendu de la température du PIC, et non de la température ambiante.
Ce temps T_{coff} sera au minimum de 0 (à moins de 25 °C) et au maximum de :

$$(50 - 25)\,0.05 = 1,25\,\mu s$$

La température du microcontrôleur ne pouvant pas, en effet, excéder 50 °C :

$$0 < T_{coff} < 1,25\,\mu s$$

Le temps de charge dépend de la résistance placée en série avec le condensateur. Il y a deux résistances : celle de la source du signal et celle à l'intérieur du PIC.

Celle interne au PIC est directement liée à la tension d'alimentation. Plus la tension baisse, plus la résistance est élevée, donc plus le temps de chargement est long.

La formule du temps de charge du condensateur est :

$$T_c = -C\,(R_{interne} + R_{source})\,\ln(\frac{1}{2047})$$

Pour numériser avec une précision d'un demi bit, la numérisation utilisant une valeur maximale de 1023, la charge du condensateur doit être au minimum de $\frac{2046}{2047}$ de la tension à mesurer. Comme C est fixe et $\ln(\frac{1}{2047})$ également, on peut calculer la constante :

$$-C\ln(\frac{1}{2047}) = 0,914895\ 10^{-9}$$

La formule devient donc :

$$T_c = 0,914895\ 10^{-9}\,(R_{interne} + R_{source})$$

Dans notre cas, nous avons une tension d'alimentation de 5 V et une résistance de source de 10 Kohms :

$$T_c = 0,914895\ 10^{-9}\,(10\ 10^{3} + 8\ 10^{3})$$

$$T_c = 16,47\ \mu s$$

Temps total d'acquisition :

$$T_{acq} = 2\ \mu s + 16,47\ \mu s + 1,25\ \mu s = 19,72\ \mu s$$

Tous les calculs sont rappelés dans la datasheet du PIC16F737.
Nous prendrons un T_{acq} de 20 µs pour notre PIC alimentée sous 5 V.

3.3.4. *La conversion*

Après le temps T_{acq}, on peut considérer le condensateur comme chargé et prêt à être connecté sur l'entrée du convertisseur analogique/numérique. Cette connexion est de l'ordre de 100 ns.

Une fois le condensateur connecté et la tension à numériser présente sur l'entrée du convertisseur, ce dernier procéde à la conversion. Le principe utilisé est celui de l'approximation successive.

C'est une méthode de type dichotomique. Il s'agit de couper l'intervalle dans lequel se trouve la grandeur analogique en deux parties égales et de déterminer dans laquelle de ces deux parties se situe la valeur à numériser. Une fois cet intervalle déterminé, on le coupe de nouveau en deux et ainsi de suite jusqu'à obtenir la précision demandée.

Pour résumer, le temps de conversion est égal au temps de conversion d'un bit multiplié par le nombre de bits désirés pour le résultat. On va nommer T_{ad} le temps de conversion d'un bit. Ce temps est dérivé par division de l'horloge principale. Le diviseur peut prendre une valeur de 2, 4, 8, 16, 32 ou 64. On divise l'horloge principale, et non le compteur d'instructions.

Le temps de conversion T_{ad} ne peut descendre, pour des raisons électroniques, en dessous de 1,6 µs. En fonction des fréquences utilisées pour le quartz du PIC, il faut choisir le diviseur le plus approprié. La formule d'obtention du temps T_{ad} est de diviser le temps d'instruction T_{osc} par le diviseur donné.

Exemple à 12 MHz, le temps d'instruction est de :

$$T_{osc} = \frac{1}{12\ 10^6} = 83\ ns$$

Avec un diviseur de 4 on aura 332 ns.

Avant le démarrage effectif de la conversion et à la fin de la conversion, le PIC nécessite un temps T_{ad}. De plus, un temps équivalent à 2 T_{ad} est nécessaire avant de pouvoir effectuer une nouvelle conversion.

Temps total de conversion :

- T_{ad} : avant le début de la conversion (le temps de connexion du condensateur est inclus)
- 10 T_{ad} : conversion des 10 bits du résultat
- T_{ad} : fin de la conversion

Soit :

$$12\ 1,6\ \mu s = 19,2\ \mu s$$

Remarque : il faut attendre 2 T_{ad} de plus avant la numérisation suivante.

3.3.5. *Relations entre valeurs analogiques et représentations numériques*

Définissons :

- VREF- : tension minimale analogique (référence négative) ;
- VREF+ : tension maximale analogique (référence positive) ;
- VIN : tension d'entrée à numériser ;
- Val : valeur numérique obtenue sur 10 bits.

Pour une numérisation sur 10 bits, nous obtiendrons la valeur numérique :

$$Val = \frac{(VIN)-(VREF-)}{(VREF+)-(VREF-)} 1023$$

Nous utilisons une tension de référence négative de 0 V (VSS) et une tension de référence positive de 5 V :

$$Val = \frac{VIN}{5} 1023$$

Remarque : la tension d'entrée ne peut être ni supérieure à la tension d'alimentation VDD du PIC, ni inférieure à sa tension VSS.

Pour les tensions supérieures, il faut réaliser un diviseur de tension à partir de résistances pour que la tension appliquée reste dans les limites prévues.

3.3.6. *La théorie appliquée au PIC : pins et canaux utilisés*

Le PIC ne contient qu'un seul convertisseur, mais plusieurs pins sur lesquelles connecter les signaux analogiques. Un circuit de commutation sélectionnera la pin qui sera reliée au condensateur de maintien interne durant le temps T_{acq}.

Nous avons plusieurs canaux que nous devrons échantillonner à tour de rôle. Le temps total nécessaire sera la somme des temps de chaque conversion.

Le PIC16F737 dispose de 11 canaux d'entrée analogique. Nous pourrons donc échantillonner les 8 signaux prévus sur les pins AN0 à AN4 et AN8 à AN10. Les pins AN0 à AN4 sont les dénominations analogiques des pins RA0 à RA3 + RA5, tandis que les pins AN8 à AN10 sont les dénominations analogiques des pins RB1 à RB3.

La procédure de numérisation pour plusieurs canaux devient la suivante (après paramétrage) :

– on choisit le canal à numériser et on met en route le convertisseur ;

– on attend T_{acq} ;

– on lance la numérisation ;

– on attend la fin de la numérisation ;

– on attend 2 T_{ad} ;

– on recommence avec le canal suivant.

3.3.7. *Les registres ADRESL et ADRESH*

Le convertisseur donne un résultat sur 10 bits et devra obligatoirement être sauvegardé dans 2 registres. Ces registres sont « ADRESL » et « ADRESH ». Deux registres contiennent 16 bits et nous n'en utilisons que 10. Microchip laisse le choix à l'utilisateur sur la façon dont est sauvegardé le résultat : justification à gauche ou à droite.

La justification à droite complète la partie gauche du résultat par des « 0 » :

ADRESH								**ADRESL**							
0	0	0	0	0	0	b9	b8	b7	b6	b5	b4	b3	b2	b1	b0

La justification à gauche procède de la méthode inverse :

ADRESH								**ADRESL**							
b9	b8	b7	b6	b5	b4	b3	b2	b1	b0	0	0	0	0	0	0

La justification à droite sera utilisée puisque nous avons besoin de l'intégralité des 10 bits de résultat, tandis que la justification à gauche est pratique lorsque 8 bits suffisent. Le choix de la méthode s'effectue à l'aide du bit 7 de ADCON1 (*Le registre ADCON1*, page 15).

Remarque : le registre ADRESH se situe en banque 0, alors que ADRESL se trouve en banque 1.

Le bit « ADON » du registre ADCON0 permet de mettre en service le convertisseur (*Le registre ADCON0*, page 16). Le positionnement de ce bit permet de démarrer la charge du condensateur interne et ainsi détermine le début du temps d'acquisition.

La mise à « 1 » du bit « GO/DONE » détermine le début de la conversion soit 12 T_{ad}.

Remarque : si la conversion en cours est arrêtée manuellement, le résultat ne sera pas transféré dans les registres ADRESL et ADRESH. Il n'y aura aucun résultat, même partiel.

3.3.8. *La conversion analogique/numérique et les interruptions*

L'interruption générée par le convertisseur est une interruption périphérique. Les différentes étapes de sa mise en service sont :

– positionnement du bit ADIE du registre PIE1 ;

– positionnement du bit PEIE du registre INTCON ;

– positionnement du bit GIE du registre INTCON.

Toute fin de conversion analogique entraînera une interruption. Il suffira de remettre à « 0 » le bit « ADIF » après traitement de cette interruption. Nous n'utiliserons pas ce principe dans notre programme.

3.3.9. *L'utilisation du convertisseur*

Résumé des opérations concrètes à effectuer pour échantillonner un signal, le PIC étant cadencé à 12 MHz et sous une tension d'alimentation de 5 V :

1. configurer ADCON1 en fonction des pins utilisées en mode analogique, ainsi que les registres TRISA, TRISB et TRISC ;
2. valider, si souhaitée, l'interruption du convertisseur (PEIE, ADIE, GIE) ;
3. paramètrer le diviseur 32 sur ADCON0 (b'10000000') ;
4. choisir le canal en cours de digitalisation sur ADCON0 et lancer le convertisseur (b'10xxx0x1') ;
5. positionner, le bit ADON du registre ADCON0 ;
6. attendre le temps T_{acq} (environ 19,7 µs sous 5 V) ;
7. démarrer la conversion en positionnant le bit GO du registre ADCON0 ;
8. attendre la fin de la conversion ;
9. lire les registres ADRESH et si nécessaire ADRESL ;
10. attendre un temps équivalent à 2 T_{ad} (environ 3,2 µs) ;
11. recommencer au point 4 pour la mesure suivante.

Puisque nous disposons de beaucoup de temps entre deux lectures de la valeur analogique, nous avons effectué plusieurs mesures intermédiaires, au nombre total de huit, et effectué la moyenne de celles-ci. Ainsi, un parasite éventuel, ou une légère fluctuation de la tension sera fortement atténué.

La moyenne est réalisée par la somme des huit mesures, puis la division du résultat s'effectue par décalage. Il est plus pratique d'effectuer un nombre de mesures qui est une puissance de 2 (2 mesures, 4, 8, 16…). Le résultat de la somme sera sur 13 bits, décalé ensuite de 3 fois vers la droite, ce qui effectue une division par 8 (premier décalage : division par 2, deuxième décalage : division par 4…). Le résultat de la moyenne se retrouve donc sur 10 bits.

La conversion est terminée. Une boucle dans le programme est effectuée pour continuellement mesurer les valeurs et les afficher sur l'écran LCD.

3.4. **Liaison série RS-232**

En ce qui concerne la liaison série RS-232, dont l'étude a suivi le fonctionnement de l'affichage et de la conversion analogique/numérique, un test d'envoi et de réception de données a été mis en oeuvre.

Les données doivent être stockées avant de pouvoir être envoyées. C'est dans le buffer de transmission appelé TXREG que cela se fera. Et réciproquement pour la réception, le registre se nomme RCREG.

Les registres TXSTA, RCSTA et SPBRG doivent être configurés :

– TXSTA : b'00000100' ; le bit BRGH est mis à « 1 » pour configurer la liaison en *High Speed*[1]

– RCSTA : b'10000000' ; le bit SPEN est mis à « 1 » pour autoriser le port série et configurer les broches RC6 et RC7 en RX et TX

– SPBRG : d'77' ; la valeur « 77 » est mise dans ce registre suivant le calcul du *Baud Rate*[2].

Le test se déroule de la façon suivante : mettre une valeur quelconque dans le buffer de transmission, autoriser la transmission en mettant TXEN à « 1 », attendre jusqu'à ce que le buffer soit vide. Ensuite, lorsqu'il est vide, on initialise le buffer de réception RCREG (on l'efface), on configure le *Asynchronous mode*[3] en mettant à « 1 » le bit SYNC du

[1]haute vitesse

[2]vitesse de trnasmission

[3]mode asynchrone

registre TXSTA, et tant que le buffer de réception n'est pas plein, on continue de récupérer les données.

La réception se fait correctement. Ceci fonctionne juste pour l'envoi d'un octet et n'a pas été testé pour autre chose.

3.5. **Problèmes rencontrés**

Nous avons rencontré quelques difficultés concernant la programmation de l'affichage. Elles se situent au niveau des temporisations à respecter. La synchronisation est nécéssaire et importante. C'est pour cela qu'il faut être précis lors de la programmation des routines d'attentes. En calculant le temps que peut prendre un cycle en fonction du quartz, le nombre de cycles nécéssaires pour faire un délai d'attente précis.

Autre chose ayant posé problème : le manque d'opérations mathématiques complexes possibles comme la multiplication et la division. Comme tous les microcontrôleurs, les PIC16F737 est dépourvu de circuits nécessaires pour ces opérations qui doivent être implémentées via *software*. Ceci fut handicapant pour effectuer l'élaboration des résultats acquis après la conversion analogique/numérique.

Pour règler ce problème, il a fallut programmer une routine de multiplication d'un registre de 8 bits par 8 bits et une routine de division d'un registre de 16 bits par 16 bits.

CHAPITRE 4

PARTIE *SOFTWARE* FIXE

Le projet se découpe en deux parties : l'acquisition de données et la transmission de celles-ci. Si la première ne marche pas, la seconde ne peut être développée.

C'est pourquoi la plus grande partie du temps a été consacrée à la programmation du micrcontrôleur, à l'affichage et surtout à la conversion analogique/numérique. L'envoi de données par liaison WiFi constitue la finalité du projet.

Cette partie indispensable à la continuation et au développement du système sera traitée pendant la dernière partie du stage.

4.1. **Transmission WiFi**

Avant de pouvoir détecter la carte RF avec la carte WiFi de l'ordinateur, il faut dans un premier temps configurer le EZL-80c. Ce dernier se configure en le branchant sur le port série de l'ordinateur. Il faut donc faire une carte électronique à part pour transformer les données de niveau TTL 3,3 V du domposant, en données de niveau RS-232, allant de -12 V à +12 V.

Après cette réalisation, il faut débrancher la carte RF en mise hors tension, ensuite il faut mettre sous tension le composant EZL-80c en ayant préalablement branché le port série qui relie l'ordinateur et le composant. Puis, grâce à un logiciel que l'on peut télécharger gratuitement sur le site officiel du fabricant, « ezSerialConfig », il est possible de règler, entre autres, les paramètres d'adresse IP, de masque de sous-réseau et de passerelle par défaut (fig. 4.1). Le mode choisi est « ad-hoc[1] », avec un *Master Target SSID*[2]. Le *Target SSID* est le nom de réseau sur lequel le module WiFi doit tenter de se connecter.

Après configuration du composant, il faut remettre la carte RF, configurer un réseau WiFi d'égal à égal sur l'ordinateur en ayant configuré la passerelle par défaut et le masque de sous-réseau identiques à ceux du composant EZL-80c configuré précédemment.

---

[1] d'égal à égal

[2] SSID maître ciblé

Ensuite, la carte WiFi de l'ordinateur détecte le WiFi de la carte RF (fig. 4.3). Ceci fait, un autre logiciel, « ezConfig », permet de détecter les paramètres d'adressage et de configuration du WiFi du module, ainsi que d'autres paramètres (fig. 4.2).

Cependant, la liaison entre le module WiFi et le microncontrôleur n'est pas traitée actuellement, il m'est donc impossible de savoir si des données peuvent être envoyées et réceptionnées.

4.2. **Programme d'acquisition des données**

Une interface graphique a été élaborée mais sans utiliser les données reçues puisque la partie de transmission de données par WiFi n'est pas traitée. Cependant, le logiciel créé représente des graphiques et a la possibilité de stocker des valeurs sous forme de fichier texte (format *.txt). Une aperçu de la première version de l'interface graphique en fig. 4.4.

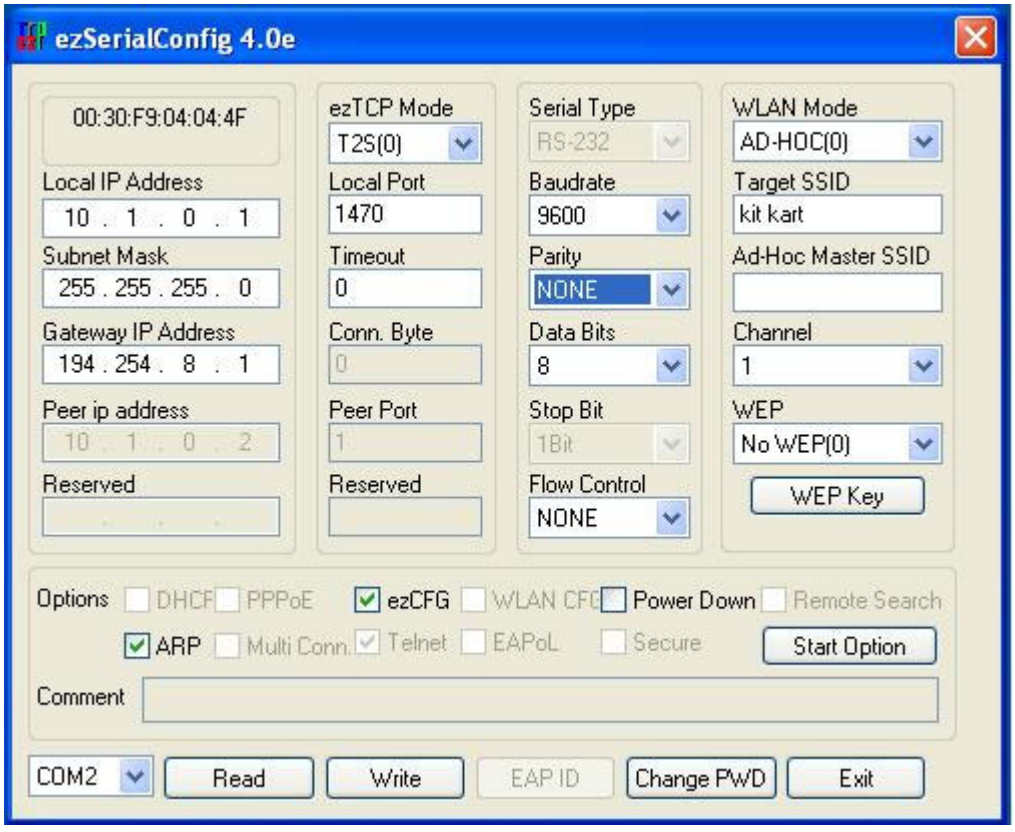

Figure 4.1. Interface du logiciel « ezSerialConfig »

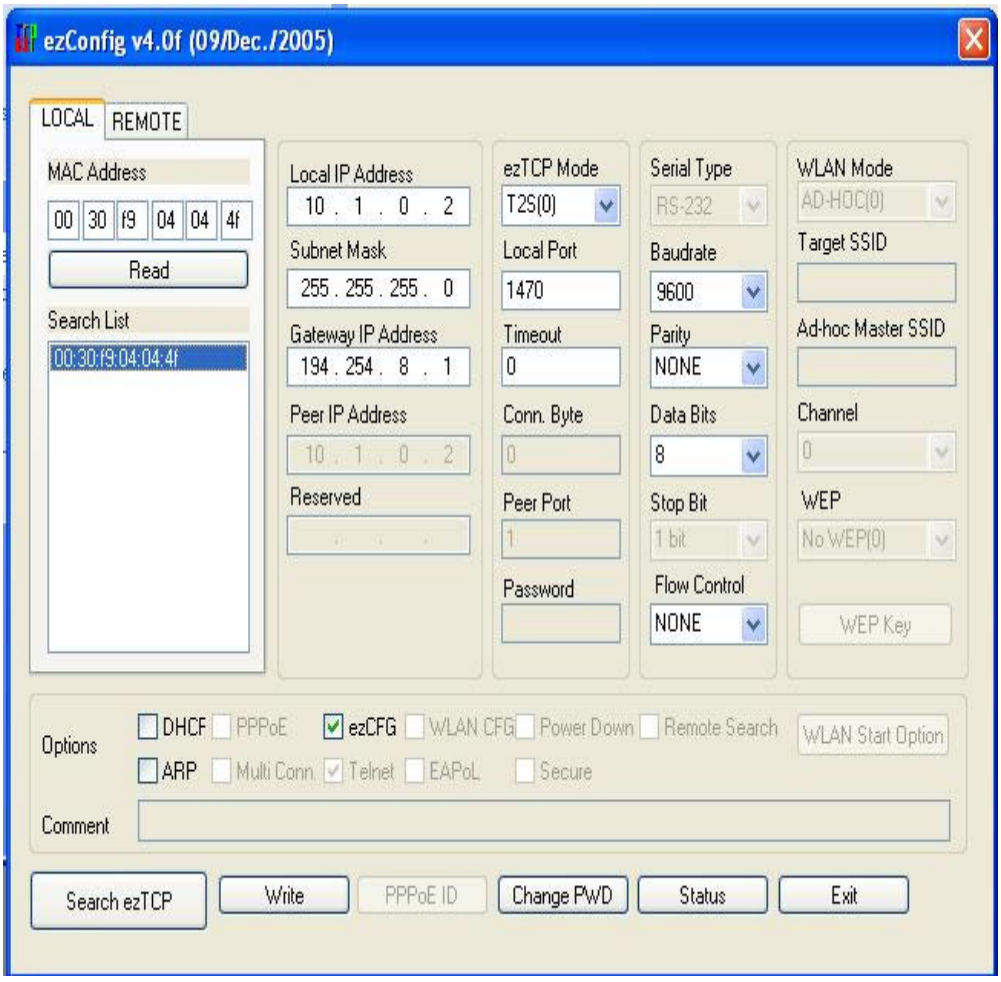

Figure 4.2. Interface du logiciel « ezConfig »

4.3. **Problèmes rencontrés**

Ma connaissance limitée concernant les protocoles WiFi à posé quelques problèmes de détection. En effet, il a fallut du temps afin de comprendre comment configurer la liaison WiFi en mode ad-hoc. De plus, le module WiFi ne marche correctement que lorsque son alimentation est à découpage, pour une raison encore inconnue.

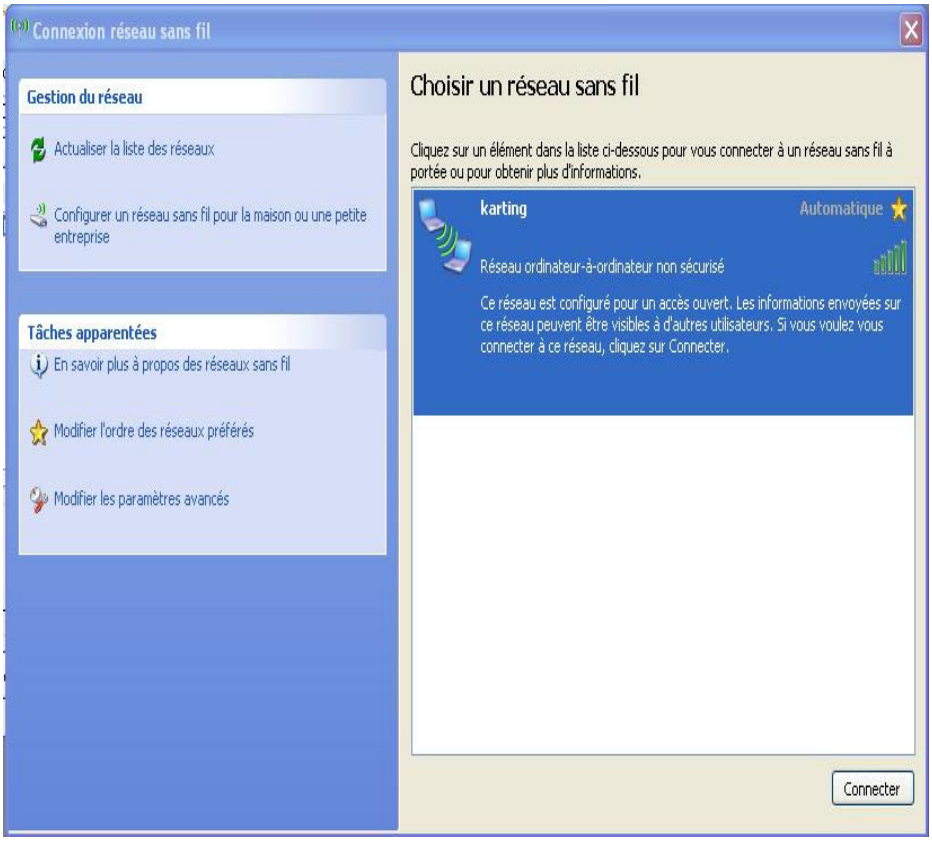

Figure 4.3. détection du WiFi de la carte RF

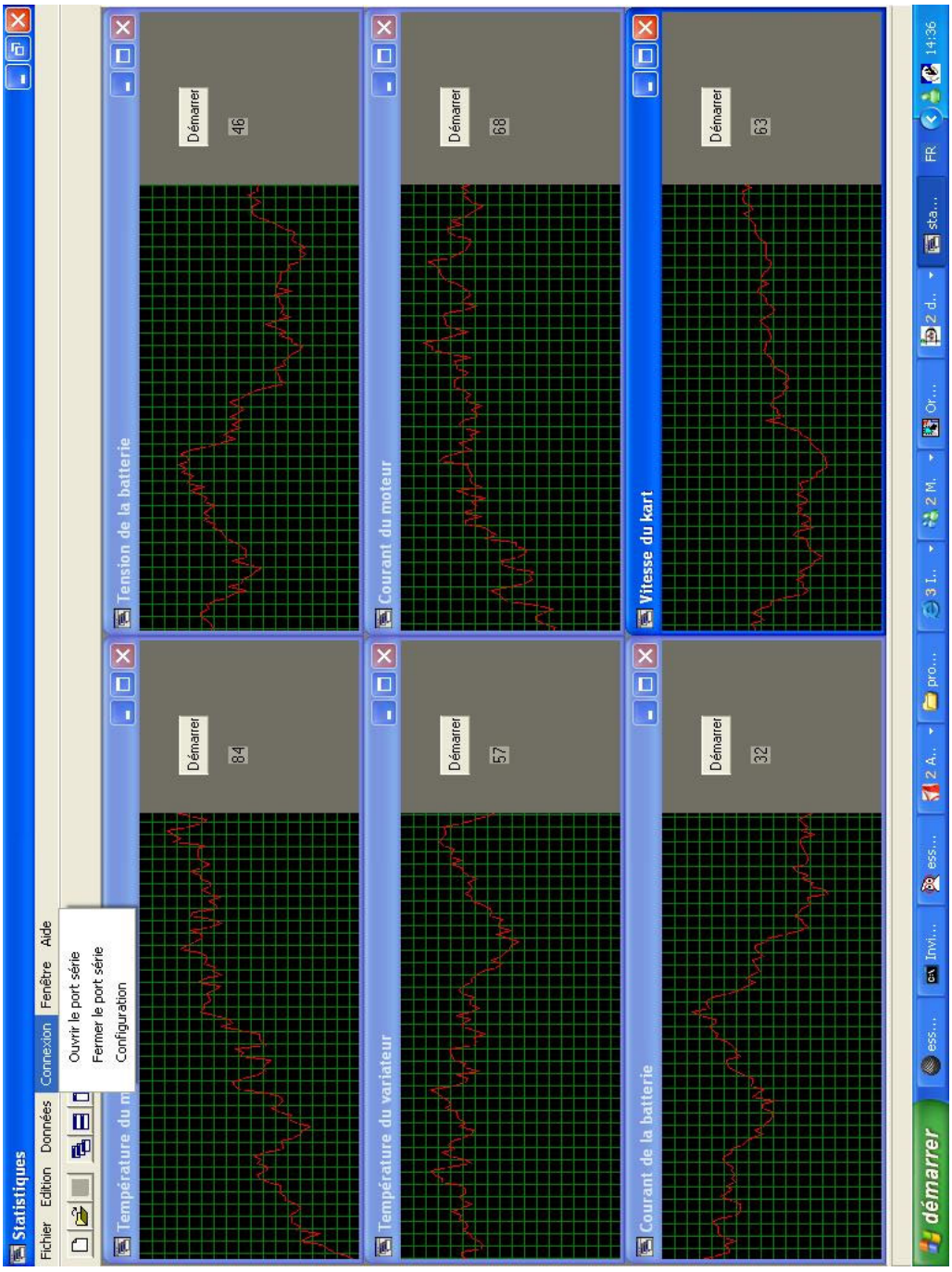

Figure 4.4. Interface graphique réalisée sous Builder C++

Conclusion

Pour ce projet, il m'a été confié de faire l'acquisition et la transmission de données numériques par WiFi pour karts électrique.

Premièrement, en ce qui concerne la partie réalisation, nous voulions tout d'abord une alimentation ayant un bon rendement et une bonne régulation.

Ensuite, le microcontrôleur devait récupérer des données analogiques, les traiter et les envoyer à un afficheur LCD. Ce dernier devait afficher la donnée en décimale avec l'option de pouvoir règler le contraste. Puis, le PIC devait envoyer, via un port RS-232, les données à un module WiFi. Enfin, le module WiFi devait les renvoyer à un ordinateur portable.

Sur notre carte actuelle, l'alimentation fonctionne, le microcontrôleur récupère une donnée, renvoie celle-ci à l'afficheur. L'écran affiche ce que l'on veut et le contraste est géré. Cependant, la liaison entre le microcontrôleur et le module WiFi n'a pas encore été effectuée.

Les améliorations possibles pourraient être, d'une part, que l'on envoie simultanément au minimum quatre valeurs analogiques et que le PIC puisse les traiter rapidement. D'autre part, il faut que la liaison entre le PIC et le module WiFi soit établie.

Concernant la partie programmation du microcontrôleur, il fallait effectuer la liaison entre le PIC et l'afficheur. Nous devions traiter une donnée analogique et la convertir en donnée numérique. Enfin, la transmission de données série RS-232 devait être au point.

Au jour d'aujourd'hui, notre programme comprend une liaison avec l'afficheur LCD, la conversion analogique/numérique et l'affichage en décimale de la valeur convertie au centième près. La vitesse d'acquisition est acceptable. La transmission série RS-232 n'a été traitée qu'un minimum : l'envoie de données du PIC vers PIC (en reliant RX et TX) fonctionne, mais pas encore la liaison jusqu'au module WiFi.

Ce qui pourrait être amèlioré serait, tout d'abord, l'affichage d'un minimum de quatre valeurs et la possibilité en appuyant sur un bouton poussoir, de changer les données affichées sur l'écran. Avec toujours une vitesse d'acquisition acceptable et une bonne précision.

Du point de vue de la programmation de l'ordinateur fixe, il faut avoir une liaison entre le module WiFi et l'ordinateur. Ensuite, une interface graphique affiche les données sous forme numérique, mais aussi sous forme de graphique. Les données transmises devront être stockées sur l'ordinateur dans des fichiers spécifiques.

La liaison actuelle entre l'ordinateur et le module WiFi est faîte. La détection et la connexion l'un à l'autre fonctionne. Néanmoins, nous n'avons pas eu le temps de traiter la partie d'envoi de données du module vers l'ordinateur. L'interface graphique est réalisée, mais sans les données du module WiFi, il n'est pas possible de tester le bon fonctionnement de l'affichage des données reçues.

Une amèlioration globale concernant la réduction du coût et la dimension de la carte peut être apportée. L'utilisation de certains périphériques réduirait le coût, comme l'oscillateur interne qui nous éviterait d'avoir un quartz et des condensateurs, plus besoin de résistances pour les boutons poussoirs car il existe des résistances de pull-up reliées au PORTB, le PIC a un circuit de *Reset* interne, donc seule une résistance de 1 KOhm est nécéssaire.

Enfin, une implantation mécanique de la carte sur le volant du kart électrique est prévue. Il faut donc minimiser sa taille et avoir un bon positionnement des boutons pour avoir une utilisation pratique de l'afficheur.

Tout au long de mon stage, mon professeur tuteur et mon maître de stage m'ont aidé à franchir les difficultés de ce projet. Ce fut un sujet très interressant, alliant programmation informatique et réalisation électronique. Un projet au sens propre du terme.

Résumé

Le travail qui m'a été confié est *l'acqusition et la transmission de données numériques par WiFi pour karts électriques.* Une station mobile affiche des données sur un afficheur LCD et les envoie à une station fixe par WiFi. Ces données sont de type analogique. La station mobile est représentée ici par un kart électrique et la station fixe, par un ordinateur. Le but est de « surveiller » à distance huit caractèristiques du véhicule qui sont : sa vitesse, la tension et le courant de batterie, le courant moteur, la température moteur et variateur, la position de la pédale de frein et de l'accélérateur.

Un microcontrôleur se chargera de récupérer les données analogiques, de les convertir en données numériques en gardant un maximum de précision, d'envoyer cela sur l'écran LCD en les affichant en décimale au centième près et enfin d'envoyer celles-ci à un module WiFi sous forme de données RS-232.

Pour finir, le module Wifi envoie les données récupérées à un ordinateur fixe. Sur celui-ci, une interface graphique accompagnera l'utilisateur, affichant les valeurs reçues numériquement mais aussi sous forme de grapique. De plus, ces valeurs devront être stockées sur l'ordinateur sous forme de fichier.

ABSTRACT

The project which I worked on concerns the *acquisition and transmission of digital data by WiFi link for electrical karts*. A mobile station displays data on a LCD display and sends them to a fixed station by a WiFi link. These are analogic data. The mobile station is represented by an electrical kart and the fixed one, by a computer. The goal is to check from the fixed station eight caracteristics of the vehicle ; they are speed, battery voltage and current, motor current, motor and driver temperature, position of the gas pedal and of the brake pedal.

A microcontroller will retrieve analogic data, will convert them in a digital form then it will display them on a LCD display and, finally, it will send them to a WiFi module using the RS232 serial protocol. The analog/digital converter used has a precision of 10 bits and this will be translated in a hundreth precision on the LCD display.

Lastly, the WiFi module sends received data to a fixed computer, where a graphic interface will assist the user in displaying received values, both in a textual and in a graphical form. Moreover, these values will be stored on the computer in files.

ANNEXE A

LOGICIELS UTILISÉS

Différents logiciels ont été utilisés durant le stage. Des logiciels utilisés durant notre formation scolaire, mais aussi des logiciels qui n'ont pas été étudiés. Tout d'abord, « Lout » est un logiciel de traitement de texte où l'on programme tout pour la mise en forme du texte. Ensuite, le logiciel « MPLab IDE » de Microchip permet de programmer le microcontrôleur. Enfin, les logiciels de réalisation de carte, Capture pour le schéma électrique, et Layout pour le routage, ont été utilisés.

A.1. **Lout**

Lout est un logiciel *opensource* pour la mise en page de textes structurés, très proche au TeX de Donald Knuth. Il se différencie de celui-ci, car il est plus récent et plus homogène. Son auteur est Jeff Kingston.

Il marche sous plusieurs systèmes d'exploitation — Linux, Macintosh, Windows —.

Voici quelques considerations trouvées sur Internet :

« Why use Lout instead of LaTeX or troff?

Lout is similar in function to LaTeX and troff. Indeed, it borrows ideas, techniques and conventions from these typesetting systems. For example, Lout uses Knuth's (the author of TeX, on which LaTeX is based) optimal line breaking algorithm, and has extended it to paragraph breaking across pages. For simple documents, Lout, LaTeX and troff offer much the same functionality, with different syntax (see the "Simple Examples" section). Lout is much more "programmer friendly" than TeX's macros (and a fortiori than incomprehensible troff macros). See the "Advanced Examples" section.

Lout makes it easy to mix text and graphics. You can draw lines, arrows and boxes, scale and rotate objects, use color commands. While many of these things are possible in LaTeX by including Postscript files generated by utility programs such as xfig, you have to specify the size of each included figure, losing a lot of Lout's flexibility.

The Lout distribution is very easy to compile and maintain, which certainly is not the case of many TeX distributions. The Lout distribution is much smaller (it fits onto a floppy disk) than LaTeX, and doesn't require

storing tfm and pk font outlines (since Postscript fonts are used). Lout Postscript files are more compact than those produced from the process .tex -TeX-> .dvi -dvips-> .ps.

Lout is multi-lingual out of the box, and understands ISO-8859 latin-1 characters.

On the other hand, LaTeX? is much more widely used than Lout (TeX has been around since the late 1970s, Lout only since 1991). It will be easier to find a local TeXpert? than a Louter, and there are many more user-contributed packages for LaTeX than for Lout. Many academic journals request (or require) that papers be submitted in LaTeX. Lout uses more memory than TeX, up to 10 Mb to compile large documents.

Lout is more friendly to advanced users. LaTeX style sheets are written in a twisted, diabolical manner apparently designed to make all but trivial changes to a document's appearance difficult; its internals are packaged as a "black box" which ordinary mortals aren't meant to understand. Lout styles are much easier to design and modify according to your needs.

The Lout formatting language is conceptually cleaner and higher level than TeX (which was designed to run efficiently on computers from another era). For example, Lout has no built-in notion of a "page".

Last but not least, Lout comes with very comprehensive and comprehensible documentation. The user's guide contains all you need to know for using Lout effectively - something that is hard to find in the LaTeX world because LaTeX consists of so many different packages (which sometimes don't get along all that well). »

A.2. **Microchip MPLab IDE**

Ce logiciel permet de programmer en langage assembly, de simuler le programme et d'écrire dans la mémoire du microcontrôleur de Microchip via la carte « PICSTART plus ».

ANNEXE B

CODE ASSEMBLY

B.1. **Conversion analogique/numérique**

Voici le code du programme qui convertit une valeur analogique sur plusieurs canaux, qui affiche en décimale au centième près les valeurs correspondantes et qui, grâce à un bouton poussoir, change les mesures affichées à l'écran : en partant d'un écran qui affiche les canaux 0 et 1, par l'appui sur le bouton, les mesures deviennent celles des canaux 3 et 4.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Auteur  : Jicez                                                        ;;;
;;;                                                                       ;;;
;;; Version : 1.0                                                         ;;;
;;;                                                                       ;;;
;;; But     : Programmation de pour la conversion                         ;;;
;;;           Analogique / Numérique                                      ;;;
;;;                                                                       ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        list                p=16f737
        #include            <P16F737.inc>

;************************VARIABLES*******************************
B_E                         EQU 05h
B_RS                        EQU 07h
B_RW                        EQU 06h

B_D0                        EQU 04h
B_D1                        EQU 00h
B_D2                        EQU 05h
B_D3                        EQU 04h
B_D4                        EQU 04h
B_D5                        EQU 00h
B_D6                        EQU 05h
B_D7                        EQU 04h

GO                          EQU 02h

VAL_1                       EQU b'00110001'
VAL_0                       EQU b'00110000'

MASQUE                      EQU b'00110000'

;******************FONCTIONS DE L'ECRAN LCD**********************
K_LCD_CURSOR_HOME                   EQU b'00000010'
K_LCD_DISPLAY_CURSOR_BLINK_ON       EQU b'00001111'
K_LCD_DISPLAY_OFF                   EQU b'00000000'
```

```
K_LCD_TWO_LINES_4BITS               EQU b'00101000'
K_LCD_CLEAR_DISPLAY                 EQU b'00000001'
K_LCD_ONE_LINE_8BITS                EQU b'00110000'
K_LCD_ALLER_LIGNE_2                 EQU b'11000000'
K_LCD_ALLER_LIGNE_4                 EQU b'11010000'

;*************************MACRO*******************************
;********************CHANGEMENT DE BANQUES***********************

BANK0       MACRO                                   ; passer en banque0;
               BCF               STATUS,RP0
               BCF               STATUS,RP1
        ENDM

BANK1       MACRO                                   ; passer en banque1
               BSF               STATUS,RP0
               BCF               STATUS,RP1
        ENDM

BANK2       MACRO                                   ; passer en banque2
               BCF               STATUS,RP0
               BSF               STATUS,RP1
        ENDM

BANK3       MACRO                                   ; passer en banque3
               BSF               STATUS,RP0
               BSF               STATUS,RP1
        ENDM

;****************************ZONE DE 80 BYTES*******************
        CBLOCK       0x20                ; Début de la zone (0x20 à 0x6F)
        CHANG_ECRAN  :         1

        R_I          :         1
        R_J          :         1
        R_K          :         1

        R_LCD_DATA   :         1

        CMPT         :         1           ; compteur de boucles

        POTAR        :         16          ; valeurs du potentiomètre lu(msb,lsb)
        RESULT       :         2           ; résultat de la somme des valeurs

        MULCND       :         1           ; multiplicande de 8 bits
        MULPLR       :         1           ; multiplieur de 8 bits
        MULTEMP      :         1           ; stocker le multiplicande temporairement
        RESULT_H_MULTI  :      1           ; poids fort du resultat de 16 bits
        RESULT_L_MULTI  :      1           ; poids faible du resultat de 16 bits
        COUNT        :         1           ; compteur

        CENT         :         1           ; chiffre des centaines
        DIZ          :         1           ; chiffre des dizaines
        UNIT         :         1           ; chiffre des unites

        DIV_RESTE_H  : 1
        DIV_RESTE_L  : 1
        DIV_DENOMINATEUR_H   : 1
        DIV_DENOMINATEUR_L   : 1
        DIV_NUMERATEUR_H     : 1
        DIV_NUMERATEUR_L     : 1
        DIV_TEMP_H           : 1
        DIV_TEMP_L           : 1
        DIV_LOOP_CPT         : 1
        DIV_RESULTAT_H       : 1
        DIV_RESULTAT_L       : 1

        POIDS_FORT_CONV      : 1
        POIDS_FAIBLE_L_CONV  : 1
        POIDS_FAIBLE_H_CONV  : 1

        ENDC                            ; Fin de la zone

;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;;                                                               ;;;
```

```
;;;                    PROGRAMME                                  ;;;
;;;                                                               ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        ORG                         0000h
        GOTO                INIT
        ORG                         0005h

;******************ROUTINE D'ATTENTE DE 2 S**********************
DELAY2S
        MOVLW                d'120'
        MOVWF                R_K

CYCLE_PREM2
        NOP
        NOP
        MOVLW                d'150'
        MOVWF                R_J
CYCLE_SEC2
        NOP
        MOVLW                d'100'
        MOVWF                R_I
CYCLE_TER2
        NOP
        NOP
        DECFSZ                 R_I,F
        GOTO                CYCLE_TER2
        DECFSZ                 R_J,F
        GOTO                CYCLE_SEC2
        DECFSZ                 R_K,F
        GOTO                CYCLE_PREM2
        RETURN

;******************ROUTINE D'ATTENTE DE 2 MS*********************
DELAY2MS
        MOVLW                d'40'
        MOVWF                R_J
CYCLE_PREM
        NOP
        NOP
        NOP
        NOP
        MOVLW                d'40'
        MOVWF                R_I
CYCLE_SEC
        NOP
        NOP
        NOP
        NOP
        DECFSZ                 R_I,F
        GOTO                CYCLE_SEC
        DECFSZ                 R_J,F
        GOTO                CYCLE_PREM
        RETURN

;********************ROUTINE D'ATTENTE DE 20 MS******************
DELAY20MS
        MOVLW                d'100'
        MOVWF                R_J
CYCLE_PREM1
        NOP
        NOP
        NOP
        NOP
        MOVLW                d'100'
        MOVWF                R_I
CYCLE_SEC1
        NOP
        NOP
        NOP
        NOP
        DECFSZ                 R_I,F
        GOTO                CYCLE_SEC1
        DECFSZ                 R_J,F
        GOTO                CYCLE_PREM1
```

```
        RETURN

;*****************ROUTINE D'ATTENTE DE 10 US*********************
DELAY10US
        MOVLW                d'35'
        MOVWF                R_I
CYCLE_PREM_10US
        NOP
        DECFSZ                R_I,F
        GOTO                CYCLE_PREM_10US
        RETURN

;***************INITIALISATION DE L'ECRAN LCD****************
INIT_LCD
        CALL                DELAY20MS
        MOVLW                K_LCD_ONE_LINE_8BITS
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_8BITS

        CALL                DELAY20MS
        CALL                SEND_LCD_COMMAND_8BITS

        CALL                DELAY2MS
        CALL                SEND_LCD_COMMAND_8BITS

        MOVLW                K_LCD_TWO_LINES_4BITS
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_8BITS

        RETURN

;********ROUTINE POUR ENVOYER LES CARACTERES VOULUS*****************
SEND_LCD_CHAR_4BITS
        MOVWF                R_LCD_DATA

        BSF                        PORTB,B_RS
        BCF                        PORTB,B_RW
        BCF                        PORTB,B_E

        CALL                DELAY10US

        BCF                        PORTC,B_D7
        BTFSC                R_LCD_DATA,7
        BSF                        PORTC,B_D7
        BCF                        PORTC,B_D6
        BTFSC                R_LCD_DATA,6
        BSF                        PORTC,B_D6
        BCF                        PORTB,B_D5
        BTFSC                R_LCD_DATA,5
        BSF                        PORTB,B_D5
        BCF                        PORTB,B_D4
        BTFSC                R_LCD_DATA,4
        BSF                        PORTB,B_D4

        CALL                DELAY10US

        BSF                        PORTB,B_E
        CALL                DELAY10US
        BCF                        PORTB,B_E

        CALL                 DELAY10US

        BCF                        PORTC,B_D3
        BTFSC                R_LCD_DATA,3
        BSF                        PORTC,B_D3
        BCF                        PORTC,B_D2
        BTFSC                R_LCD_DATA,2
        BSF                        PORTC,B_D2
        BCF                        PORTB,B_D1
        BTFSC                R_LCD_DATA,1
        BSF                        PORTB,B_D1
        BCF                        PORTB,B_D0
        BTFSC                R_LCD_DATA,0
        BSF                        PORTB,B_D0
```

```
        CALL                DELAY10US

        BSF                         PORTB,B_E
        CALL                DELAY10US
        BCF                         PORTB,B_E

        CALL                DELAY10US

        BCF                         PORTC,B_D3
        BCF                         PORTC,B_D2
        BCF                         PORTB,B_D1
        BCF                         PORTB,B_D0

        BCF                         PORTB,B_RS

        CALL                DELAY2MS
        RETURN

;********ROUTINE POUR ENVOYER LA FONCTION VOULUE*****************
SEND_LCD_COMMAND_8BITS
        BCF                         PORTB,B_RS
        BCF                         PORTB,B_RW
        BCF                         PORTB,B_E

        CALL                DELAY10US

        BCF                         PORTC,B_D7
        BTFSC                R_LCD_DATA,7
        BSF                         PORTC,B_D7
        BCF                         PORTC,B_D6
        BTFSC                R_LCD_DATA,6
        BSF                         PORTC,B_D6
        BCF                         PORTB,B_D5
        BTFSC                R_LCD_DATA,5
        BSF                         PORTB,B_D5
        BCF                         PORTB,B_D4
        BTFSC                R_LCD_DATA,4
        BSF                         PORTB,B_D4

        CALL                DELAY10US

        BSF                         PORTB,B_E
        CALL                DELAY10US
        BCF                         PORTB,B_E

        CALL                 DELAY10US

        BCF                         PORTC,B_D3
        BCF                         PORTC,B_D2
        BCF                         PORTB,B_D1
        BCF                         PORTB,B_D0

        CALL                DELAY2MS
        RETURN

;********ROUTINE POUR ENVOYER LA FONCTION VOULUE*****************
SEND_LCD_COMMAND_4BITS
        BCF                         PORTB,B_RS
        BCF                         PORTB,B_RW
        BCF                         PORTB,B_E

        CALL                DELAY10US

        BCF                         PORTC,B_D7
        BTFSC                R_LCD_DATA,7
        BSF                         PORTC,B_D7
        BCF                         PORTC,B_D6
        BTFSC                R_LCD_DATA,6
        BSF                         PORTC,B_D6
        BCF                         PORTB,B_D5
        BTFSC                R_LCD_DATA,5
        BSF                         PORTB,B_D5
        BCF                         PORTB,B_D4
        BTFSC                R_LCD_DATA,4
        BSF                         PORTB,B_D4
```

```
        CALL                DELAY10US

        BSF                         PORTB,B_E
        CALL                DELAY10US
        BCF                         PORTB,B_E

        CALL                 DELAY10US

        BCF                         PORTC,B_D3
        BTFSC                R_LCD_DATA,3
        BSF                         PORTC,B_D3
        BCF                         PORTC,B_D2
        BTFSC                R_LCD_DATA,2
        BSF                         PORTC,B_D2
        BCF                         PORTB,B_D1
        BTFSC                R_LCD_DATA,1
        BSF                         PORTB,B_D1
        BCF                         PORTB,B_D0
        BTFSC                R_LCD_DATA,0
        BSF                         PORTB,B_D0

        CALL                DELAY10US

        BSF                         PORTB,B_E
        CALL                DELAY10US
        BCF                         PORTB,B_E

        CALL                DELAY10US

        BCF                         PORTC,B_D3
        BCF                         PORTC,B_D2
        BCF                         PORTB,B_D1
        BCF                         PORTB,B_D0

        CALL                DELAY2MS
        RETURN

;********************TEMPORISATION*******************************
TEMPO
                ; gestion des 8 premiers passages
                ; -------------------------------
        MOVLW                0x08           ; pour 8 * 500 µs
        MOVWF                CMPT           ; dans compteur

        MOVLW                POTAR          ; adresse de la zone de stockage
        MOVWF                FSR            ; dans le pointeur

TEMPO1
        CALL                ACQUISITION     ; lancer acquisition
        CALL                DELAY500US      ; attendre 500 µs
        DECFSZ               CMPT,F        ; décrémenter compteur de boucles
        GOTO                TEMPO1          ; boucle suivante

                        ; calcul de la somme des valeurs
                        ; -------------------------------
        MOVLW                0x08           ; pour 8 boucles
        MOVWF                CMPT           ; dans compteur de boucles
        CLRF                RESULT          ; effacer résultat
        CLRF                RESULT+1        ; idem pour poids faible

TEMPO2
        DECF                FSR,F           ; pointer sur poids faible
        MOVF                INDF,W          ; charger poids faible
        BCF                 STATUS,C
        ADDWF               RESULT+1,F      ; ajouter au poids faible résultat
        BTFSC               STATUS,C        ; tester si débordement
        INCF                RESULT,F        ; oui, poids fort +1
        DECF                FSR,F           ; pointer sur poids fort
        MOVF                INDF,W          ; charger poids fort
        ADDWF               RESULT,F        ; ajouter au poids fort résultat

        DECFSZ              CMPT,F          ; décrémenter compteur de boucles
        GOTO                TEMPO2          ; pas dernière, suivante
```

```
        ; calcul de la moyenne sur 8 bits (décalage 5fois à gauche ou 3 à droite)
        ; ------------------------------------------------------------------------
        BCF                 STATUS,C
        RRF                 RESULT,F        ; décaler poids fort vers la droite
        RRF                 RESULT+1,F       ; idem poids faible avec b7 poids faible

        BCF                 STATUS,C
        RRF                 RESULT,F        ; décaler poids fort vers la droite
        RRF                 RESULT+1,F       ; idem poids faible avec b7 poids faible

        BCF                 STATUS,C
        RRF                 RESULT,F        ; décaler poids fort vers la droite
        RRF                 RESULT+1,F       ; idem poids faible avec b7 poids faible

        RETURN

;**********************ATTENTE DE 20 US************************
DELAY20US
        MOVLW                d'12'
        MOVWF                R_I
DELAY20US_CYCLE_PREM
        NOP
        NOP
        DECFSZ                R_I,F
        GOTO                DELAY20US_CYCLE_PREM
        RETURN

;*****************ATTENTE DE 500 US*****************************
DELAY500US
        MOVLW                d'250'
        MOVWF                R_I
DELAY500US_CYCLE_PREM
        NOP
        NOP
        NOP
        DECFSZ                R_I,F
        GOTO                DELAY500US_CYCLE_PREM
        RETURN

;**********************ACQUISITION*********************************
ACQUISITION
                        ; lancer l'acquisition
                        ; --------------------
;        MOVLW             b'10001001'        ; diviseur 32, canal 0, convertisseur ON
;       MOVWF             ADCON0              ; paramétrer convertisseur, lancer acquisition

        CALL                DELAY500US
        BSF                 ADCON0,GO        ; lancer conversion A/D
        CALL                DELAY500US

        BCF                 ADCON0,ADON        ; éteindre convertisseur
        MOVF                ADRESH,W        ; charger poids fort conversion

        MOVWF               INDF                ; sauver dans zone de sauvegarde
        INCF                FSR,F                ; pointer sur poids faible

        BANK1                                        ; passer banque 1
        MOVF                ADRESL,W        ; charger poids faible conversion

        BANK0                                        ; repasser banque 0

        MOVWF               INDF                ; sauver dans zone de sauvegarde
        INCF                FSR,F                ; pointer sur poids fort suivant

        RETURN

;***********AFFICHE LA VALEUR QUE CONTIENT RESULT**************
AFFICH_VAL_BIN_RES
        MOVLW               B'00000110'                ;le curseur s'auto-incrémente
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        MOVLW               VAL_0
        BTFSC               RESULT,7
        MOVLW               VAL_1
```

```
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           RESULT,6
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           RESULT,5
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           RESULT,4
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           RESULT,3
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           RESULT,2
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           RESULT,1
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           RESULT,0
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        RETURN

;**************ROUTINE DE MESURE************************
MESURE
        MOVLW            B'00000110'
        MOVWF            R_LCD_DATA
        CALL            SEND_LCD_COMMAND_4BITS
        MOVLW            B'01001101'
        MOVWF            R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS        ;LETTRE M

        MOVLW            B'00000110'
        MOVWF            R_LCD_DATA
        CALL            SEND_LCD_COMMAND_4BITS
        MOVLW            B'01100101'
        MOVWF            R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS        ;LETTRE e

        MOVLW            B'00000110'
        MOVWF            R_LCD_DATA
        CALL            SEND_LCD_COMMAND_4BITS
        MOVLW            B'01110011'
        MOVWF            R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS        ;LETTRE s

        MOVLW            B'00000110'
        MOVWF            R_LCD_DATA
        CALL            SEND_LCD_COMMAND_4BITS
        MOVLW            B'01110101'
        MOVWF            R_LCD_DATA
```

```
        CALL                SEND_LCD_CHAR_4BITS         ;LETTRE u

        MOVLW                B'00000110'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS
        MOVLW                B'01110010'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS         ;LETTRE r

        MOVLW                B'00000110'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS
        MOVLW                B'01100101'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS         ;LETTRE e

        MOVLW                B'00000110'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS
        MOVLW                B'00111010'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS         ;LETTRE :
        RETURN

;************************TENSION************************************
TENSION
        MOVLW                B'00000110'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS
        MOVLW                B'01010100'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS         ;LETTRE T

        MOVLW                B'00000110'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS
        MOVLW                B'01100101'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS         ;LETTRE e

        MOVLW                B'00000110'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS
        MOVLW                B'01101110'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS         ;LETTRE n

        MOVLW                B'00000110'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS
        MOVLW                B'01110011'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS         ;LETTRE s

        MOVLW                B'00000110'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS
        MOVLW                B'01001001'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS         ;LETTRE i

        MOVLW                B'00000110'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS
        MOVLW                B'01101111'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS         ;LETTRE o

        MOVLW                B'00000110'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS
        MOVLW                B'01101110'
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS         ;LETTRE n
```

```
        MOVLW               B'00000110'
        MOVWF               R_LCD_DATA
        CALL               SEND_LCD_COMMAND_4BITS
        MOVLW               B'00111010'
        MOVWF               R_LCD_DATA
        CALL               SEND_LCD_CHAR_4BITS        ;LETTRE :
        RETURN

;******************ROUTINE DE MULTIPLICATION 8 PAR 8*****************
MULTI
    CLRF    RESULT_H_MULTI
    CLRF    RESULT_L_MULTI
    MOVLW   d'8'
    MOVWF   COUNT
    MOVF    MULPLR, W
    MOVWF   MULTEMP                                    ; sauver dans multemp le multiplieur
    MOVF    MULCND, W
    BCF     STATUS, C

LOOP
    RRF     MULPLR, F
    BTFSC   STATUS, C
    ADDWF   RESULT_H_MULTI, F
    RRF     RESULT_H_MULTI, F
    RRF     RESULT_L_MULTI, F
    DECFSZ  COUNT, F
    GOTO    LOOP

    RETURN

;*********************ROUTINE DE DIVISION 16 PAR 16***************
DIVISION_16X16                      ; résultat dans DIV_NUMERATEUR, reste dans DIV_RESTE
        MOVLW   d'16'
        MOVWF   DIV_LOOP_CPT
        MOVF    DIV_NUMERATEUR_H, W
        MOVWF   DIV_TEMP_H
        MOVF    DIV_NUMERATEUR_L, W
        MOVWF   DIV_TEMP_L
        CLRF    DIV_RESULTAT_H
        CLRF    DIV_RESULTAT_L
        CLRF    DIV_RESTE_H
        CLRF    DIV_RESTE_L

DIV_PREM
        BCF     STATUS, C
        RLF     DIV_TEMP_L, F
        RLF     DIV_TEMP_H, F
        RLF     DIV_RESTE_L, F
        RLF     DIV_RESTE_H, F
        MOVF    DIV_DENOMINATEUR_H, W
        SUBWF   DIV_RESTE_H, W
        BTFSS   STATUS, Z
        GOTO    DIV_SEC
        MOVF    DIV_DENOMINATEUR_L, W
        SUBWF   DIV_RESTE_L, W

DIV_SEC
        BTFSS   STATUS, C
        GOTO    DIV_TER
        MOVF    DIV_DENOMINATEUR_L, W
        SUBWF   DIV_RESTE_L, F
        BTFSS   STATUS, C
        DECF    DIV_RESTE_H, F
        MOVF    DIV_DENOMINATEUR_H, W
        SUBWF   DIV_RESTE_H, F
        BSF     STATUS, C

DIV_TER
        RLF     DIV_RESULTAT_L, F
        RLF     DIV_RESULTAT_H, F
        DECFSZ  DIV_LOOP_CPT, F
        GOTO    DIV_PREM

        RETURN
```

```
;********************ROUTINE POUR AFFICHER LES CHIFFRES********
ASCII
; premiere partie : la division par 100 pour avoir le chiffre des centaines
        MOVLW       d'100'
        CLRF        DIV_DENOMINATEUR_H
        MOVWF       DIV_DENOMINATEUR_L
        CALL        DIVISION_16X16

; on enregistre la valeur dans la variable appropriée et on prépare le resultat
; du modulo pour la division suivante
        MOVF        DIV_RESULTAT_L, W
        MOVWF       CENT
        MOVF        DIV_RESTE_L, W
        MOVWF       DIV_NUMERATEUR_L

; deuxième partie : on divise par 10 pour avoir le chiffre des dizaines
        MOVF        DIV_RESTE_H, W
        MOVWF       DIV_NUMERATEUR_H
        MOVLW       d'10'
        MOVWF       DIV_DENOMINATEUR_L
        CALL        DIVISION_16X16

; on recupere le resultat qu'on stocke dans la variable appropriee et le resultat
; du modulo est le chiffre des unites qui sera lui aussi enregistre dans la variable
; appropriee
        MOVF        DIV_RESULTAT_L, W
        MOVWF       DIZ
        MOVF        DIV_RESTE_L, W
        MOVWF       UNIT

; troisieme partie :
        MOVLW               b'00110000'
        ADDWF               CENT, W
        CALL               SEND_LCD_CHAR_4BITS

        MOVLW               b'00101100'
        CALL               SEND_LCD_CHAR_4BITS

        MOVLW               b'00110000'
        ADDWF               DIZ, W
        CALL               SEND_LCD_CHAR_4BITS

        MOVLW               b'00110000'
        ADDWF               UNIT, W
        CALL               SEND_LCD_CHAR_4BITS

        RETURN

;*****************ROUTINE DE CALCUL*****************************
CALC_AFFICH
        MOVLW              d'50'
        MOVWF              MULPLR
        MOVF               RESULT,W
        MOVWF              MULCND

        CALL               MULTI

        MOVF               RESULT_L_MULTI,W
        MOVWF              POIDS_FORT_CONV

        MOVLW              d'50'
        MOVWF              MULPLR
        MOVF               RESULT+1,W
        MOVWF              MULCND

        CALL               MULTI

        MOVF               RESULT_L_MULTI,W
        MOVWF              POIDS_FAIBLE_L_CONV
        MOVF               RESULT_H_MULTI,W
        ADDWF              POIDS_FORT_CONV,W
        MOVWF              POIDS_FAIBLE_H_CONV

        MOVLW              b'00000000'
        MOVWF              DIV_DENOMINATEUR_H
```

```
        MOVLW               b'00010000'
        MOVWF               DIV_DENOMINATEUR_L

        MOVF                POIDS_FAIBLE_H_CONV,W
        MOVWF               DIV_NUMERATEUR_H
        MOVF                POIDS_FAIBLE_L_CONV,W
        MOVWF               DIV_NUMERATEUR_L

        CALL                DIVISION_16X16

        MOVLW               d'10'
        MOVWF               MULPLR
        MOVF                DIV_RESULTAT_H,W
        MOVWF               MULCND

        CALL                MULTI

        MOVF                RESULT_L_MULTI,W
        MOVWF               POIDS_FORT_CONV

        MOVLW               d'10'
        MOVWF               MULPLR
        MOVF                DIV_RESULTAT_L,W
        MOVWF               MULCND

        CALL                MULTI

        MOVF                RESULT_L_MULTI,W
        MOVWF               POIDS_FAIBLE_L_CONV
        MOVF                RESULT_H_MULTI,W
        ADDWF               POIDS_FORT_CONV,W
        MOVWF               POIDS_FAIBLE_H_CONV

        MOVLW                b'00000000'
        MOVWF                DIV_DENOMINATEUR_H
        MOVLW                b'01000000'
        MOVWF                DIV_DENOMINATEUR_L

        MOVF                POIDS_FAIBLE_H_CONV,W
        MOVWF               DIV_NUMERATEUR_H
        MOVF                POIDS_FAIBLE_L_CONV,W
        MOVWF               DIV_NUMERATEUR_L

        CALL                DIVISION_16X16

        MOVF                DIV_RESULTAT_H,W
        MOVWF               DIV_NUMERATEUR_H
        MOVF                DIV_RESULTAT_L,W
        MOVWF               DIV_NUMERATEUR_L

        CALL                ASCII

        RETURN

;****************LES HUIT MESURES*****************************
MAIN_MES_1
        MOVLW               b'10000001'         ; diviseur 32, canal 0, convertisseur ON
       MOVWF            ADCON0              ; paramétrer convertisseur, lancer acquisition

        MOVLW               K_LCD_ALLER_LIGNE_2
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        CALL                TEMPO                                   ;appel des 8 mesures

        CALL                CALC_AFFICH

        RETURN

MAIN_MES_2
        MOVLW               b'10001001'         ; diviseur 32, canal 1, convertisseur ON
       MOVWF            ADCON0              ; paramétrer convertisseur, lancer acquisition

        MOVLW               K_LCD_ALLER_LIGNE_4
        MOVWF               R_LCD_DATA
```

```
        CALL                SEND_LCD_COMMAND_4BITS

        CALL                TEMPO                                   ;appel des 8 mesures

        CALL                CALC_AFFICH

        RETURN

MAIN_MES_3
        MOVLW               b'10010001'         ; diviseur 32, canal 2, convertisseur ON
       MOVWF             ADCON0              ; paramétrer convertisseur, lancer acquisition

        MOVLW               K_LCD_ALLER_LIGNE_2
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        CALL                TEMPO                                   ;appel des 8 mesures

        CALL                CALC_AFFICH

ENVOI_MES_1
        MOVF                CENT,W
        MOVWF               TXREG

        BANK1
        BSF                 TXSTA,TXEN

ATTENTE
        BANK1
        BTFSS               TXSTA,TRMT
        GOTO                ATTENTE

        BCF                 TXSTA,TXEN

        RETURN

MAIN_MES_4
        MOVLW               b'10011001'         ; diviseur 32, canal 3, convertisseur ON
       MOVWF             ADCON0              ; paramétrer convertisseur, lancer acquisition

        MOVLW               K_LCD_ALLER_LIGNE_4
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        CALL                TEMPO                                   ;appel des 8 mesures

        CALL                CALC_AFFICH

        RETURN

;MAIN_MES_5
;        MOVLW               b'10000001'         ; diviseur 32, canal 0, convertisseur ON
;       MOVWF             ADCON0              ; paramétrer convertisseur, lancer acquisition
;
;        MOVLW               K_LCD_ALLER_LIGNE_2
;        MOVWF               R_LCD_DATA
;        CALL                SEND_LCD_COMMAND_4BITS
;
;        CALL                TEMPO                                   ;appel des 8 mesures
;
;        CALL                CALC_AFFICH
;
;        RETURN
;
;MAIN_MES_6
;        MOVLW               b'10001001'         ; diviseur 32, canal 1, convertisseur ON
;       MOVWF             ADCON0              ; paramétrer convertisseur, lancer acquisition
;
;        MOVLW               K_LCD_ALLER_LIGNE_4
;        MOVWF               R_LCD_DATA
;        CALL                SEND_LCD_COMMAND_4BITS
;
;        CALL                TEMPO                                   ;appel des 8 mesures
;
;        CALL                CALC_AFFICH
```

```
;
;         RETURN

;MAIN_MES_7
;        MOVLW               b'10010001'         ; diviseur 32, canal 2, convertisseur ON
;       MOVWF             ADCON0              ; paramétrer convertisseur, lancer acquisition
;
;         MOVLW                K_LCD_ALLER_LIGNE_2
;         MOVWF                R_LCD_DATA
;         CALL                SEND_LCD_COMMAND_4BITS
;
;         CALL                TEMPO                                   ;appel des 8 mesures
;
;         CALL                CALC_AFFICH
;
;         RETURN

;MAIN_MES_8
;        MOVLW               b'10011001'         ; diviseur 32, canal 3, convertisseur ON
;       MOVWF             ADCON0              ; paramétrer convertisseur, lancer acquisition
;
;         MOVLW                K_LCD_ALLER_LIGNE_4
;         MOVWF                R_LCD_DATA
;         CALL                SEND_LCD_COMMAND_4BITS
;
;         CALL                TEMPO                                   ;appel des 8 mesures
;
;         CALL                CALC_AFFICH
;
;         RETURN
;*****************CONFIGURATION DES REGISTRES******************
INIT
        BANK0
        CLRF                PORTA
        CLRF                PORTB
        CLRF                PORTC

        BANK1
        MOVLW                b'10001011'
        MOVWF                ADCON1
        MOVLW                b'10001110'
        MOVWF                ADCON2
        MOVLW                b'11011111'
        MOVWF                TRISA
        MOVLW                b'00001110'
        MOVWF                TRISB
        MOVLW                b'00001111'
        MOVWF                TRISC

        MOVLW                b'00000100'
        MOVWF                TXSTA

        MOVLW                d'77'
        MOVWF                SPBRG

        BANK0

;*****************PROGRAMME PRINCIPAL*******************
MAIN
        CALL                INIT_LCD                ;initialisation de l'écran LCD

        MOVLW                K_LCD_TWO_LINES_4BITS
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        MOVLW                K_LCD_CLEAR_DISPLAY
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        MOVLW                K_LCD_DISPLAY_CURSOR_BLINK_ON
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        MOVLW                K_LCD_CURSOR_HOME
        MOVWF                R_LCD_DATA
```

```
        CALL                SEND_LCD_COMMAND_4BITS

VIDER
        CLRF                CHANG_ECRAN

MAIN_LOOP
        BTFSC                PORTC,3
        INCF                CHANG_ECRAN,F

        BTFSC                CHANG_ECRAN,2
        GOTO                VIDER

        BTFSC                CHANG_ECRAN,1
        GOTO                TEST_1
        GOTO                TEST_2

TEST_1
        BTFSS                CHANG_ECRAN,0
        GOTO                ECRAN_2
        GOTO                ECRAN_3

TEST_2
        BTFSS                CHANG_ECRAN,0
        GOTO                ECRAN_0
        GOTO                ECRAN_1

ECRAN_0
        MOVLW                K_LCD_CURSOR_HOME
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        CALL                MESURE
        CALL                MAIN_MES_1
        CALL                MAIN_MES_2
        GOTO                MAIN_LOOP

ECRAN_1
        MOVLW                K_LCD_CURSOR_HOME
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        CALL                TENSION
        CALL                MAIN_MES_3
        CALL                MAIN_MES_4
        GOTO                MAIN_LOOP

ECRAN_2
        MOVLW                K_LCD_CURSOR_HOME
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        CALL                MESURE
        GOTO                MAIN_LOOP
;        CALL                MAIN_MES_5
;        CALL                MAIN_MES_6
;
ECRAN_3
        MOVLW                K_LCD_CURSOR_HOME
        MOVWF                R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        CALL                TENSION
        GOTO                MAIN_LOOP
;        CALL                MAIN_MES_7
;        CALL                MAIN_MES_8

;************************FIN*************************
        END
```

B.2. **Transmission RS-232**

Voici le code du programme envoyant une valeur par TX, mise à l'origine dans le registre TXREG, par voie série et la réceptionnant sur RX, stockée dans le registre RCREG.

```
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;
;;; Auteur  : Jicez                                            ;;;
;;;                                                            ;;;
;;; Version : 1.0                                              ;;;
;;;                                                            ;;;
;;; But     : Programmation de pour la conversion              ;;;
;;;           Analogique / Numérique                           ;;;
;;;                                                            ;;;
;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;;

        list             p=16f737
        #include        <P16F737.inc>

;*************************VARIABLES*****************************
B_E                        EQU 05h
B_RS                       EQU 07h
B_RW                       EQU 06h

B_D0                       EQU 04h
B_D1                       EQU 00h
B_D2                       EQU 05h
B_D3                       EQU 04h
B_D4                       EQU 04h
B_D5                       EQU 00h
B_D6                       EQU 05h
B_D7                       EQU 04h

GO                         EQU 02h

VAL_1                      EQU b'00110001'
VAL_0                      EQU b'00110000'

;******************FONCTIONS DE L'ECRAN LCD**********************
K_LCD_CURSOR_HOME                        EQU b'00000010'
K_LCD_DISPLAY_CURSOR_BLINK_ON            EQU b'00001111'
K_LCD_DISPLAY_OFF                        EQU b'00000000'
K_LCD_TWO_LINES_4BITS                    EQU b'00101000'
K_LCD_CLEAR_DISPLAY                      EQU b'00000001'
K_LCD_ONE_LINE_8BITS                     EQU b'00110000'
K_LCD_ALLER_LIGNE_2                      EQU b'11000000'

;**************************MACRO*******************************
;*******************CHANGEMENT DE BANQUES***********************

BANK0        MACRO                               ; passer en banque0;
                BCF               STATUS,RP0
                BCF               STATUS,RP1
        ENDM

BANK1        MACRO                               ; passer en banque1
                BSF               STATUS,RP0
                BCF               STATUS,RP1
        ENDM

BANK2        MACRO                               ; passer en banque2
                BCF               STATUS,RP0
                BSF               STATUS,RP1
        ENDM

BANK3        MACRO                               ; passer en banque3
                BSF               STATUS,RP0
                BSF               STATUS,RP1
        ENDM

;***************************ZONE DE 80 BYTES*******************
        CBLOCK        0x20               ; Dbut de la zone (0x20 à 0x6F)
        R_LCD_DATA : 1
```

```
        R_I           :         1
        R_J           :         1
        R_K           :         1

    ENDC                                ; Fin de la zone

;*****************************************************************

        ORG                         0000h
        GOTO                INIT
        ORG                         0005h

;******************ROUTINE D'ATTENTE DE 2 S**********************
DELAY2S
        MOVLW                d'120'
        MOVWF                R_K

CYCLE_PREM2
        NOP
        NOP
        MOVLW                d'150'
        MOVWF                R_J
CYCLE_SEC2
        NOP
        MOVLW                d'100'
        MOVWF                R_I
CYCLE_TER2
        NOP
        NOP
        DECFSZ                 R_I,F
        GOTO                CYCLE_TER2
        DECFSZ                 R_J,F
        GOTO                CYCLE_SEC2
        DECFSZ                 R_K,F
        GOTO                CYCLE_PREM2
        RETURN

;******************ROUTINE D'ATTENTE DE 2 MS*********************
DELAY2MS
        MOVLW                d'40'
        MOVWF                R_J
CYCLE_PREM
        NOP
        NOP
        NOP
        NOP
        MOVLW                d'40'
        MOVWF                R_I
CYCLE_SEC
        NOP
        NOP
        NOP
        NOP
        DECFSZ                 R_I,F
        GOTO                CYCLE_SEC
        DECFSZ                 R_J,F
        GOTO                CYCLE_PREM
        RETURN

;*******************ROUTINE D'ATTENTE DE 20 MS*****************
DELAY20MS
        MOVLW                d'100'
        MOVWF                R_J
CYCLE_PREM1
        NOP
        NOP
        NOP
        NOP
        MOVLW                d'100'
        MOVWF                R_I
CYCLE_SEC1
        NOP
        NOP
        NOP
```

```
        NOP
        DECFSZ                R_I,F
        GOTO                CYCLE_SEC1
        DECFSZ                R_J,F
        GOTO                CYCLE_PREM1
        RETURN

;*****************ROUTINE D'ATTENTE DE 10 US*********************
DELAY10US
        MOVLW                 d'35'
        MOVWF                 R_I
CYCLE_PREM_10US
        NOP
        DECFSZ                R_I,F
        GOTO                CYCLE_PREM_10US
        RETURN

;***************INITIALISATION DE L'ECRAN LCD****************
INIT_LCD
        CALL                DELAY20MS
        MOVLW               K_LCD_ONE_LINE_8BITS
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_COMMAND_8BITS

        CALL                DELAY20MS
        CALL                SEND_LCD_COMMAND_8BITS

        CALL                DELAY2MS
        CALL                SEND_LCD_COMMAND_8BITS

        MOVLW               K_LCD_TWO_LINES_4BITS
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_COMMAND_8BITS

        RETURN

;********ROUTINE POUR ENVOYER LES CARACTERES VOULUS*****************
SEND_LCD_CHAR_4BITS
        BSF                         PORTB,B_RS
        BCF                         PORTB,B_RW
        BCF                         PORTB,B_E

        CALL                DELAY10US

        BCF                         PORTC,B_D7
        BTFSC                R_LCD_DATA,7
        BSF                         PORTC,B_D7
        BCF                         PORTC,B_D6
        BTFSC                R_LCD_DATA,6
        BSF                         PORTC,B_D6
        BCF                         PORTB,B_D5
        BTFSC                R_LCD_DATA,5
        BSF                         PORTB,B_D5
        BCF                         PORTB,B_D4
        BTFSC                R_LCD_DATA,4
        BSF                         PORTB,B_D4

        CALL                DELAY10US

        BSF                         PORTB,B_E
        CALL                DELAY10US
        BCF                         PORTB,B_E

        CALL                 DELAY10US

        BCF                         PORTC,B_D3
        BTFSC                R_LCD_DATA,3
        BSF                         PORTC,B_D3
        BCF                         PORTC,B_D2
        BTFSC                R_LCD_DATA,2
        BSF                         PORTC,B_D2
        BCF                         PORTB,B_D1
        BTFSC                R_LCD_DATA,1
        BSF                         PORTB,B_D1
        BCF                         PORTB,B_D0
```

```
        BTFSC               R_LCD_DATA,0
        BSF                       PORTB,B_D0

        CALL               DELAY10US

        BSF                       PORTB,B_E
        CALL               DELAY10US
        BCF                       PORTB,B_E

        CALL               DELAY10US

        BCF                       PORTC,B_D3
        BCF                       PORTC,B_D2
        BCF                       PORTB,B_D1
        BCF                       PORTB,B_D0

        BCF                       PORTB,B_RS

        CALL               DELAY2MS
        RETURN

;********ROUTINE POUR ENVOYER LA FONCTION VOULUE*****************
SEND_LCD_COMMAND_8BITS
        BCF                       PORTB,B_RS
        BCF                       PORTB,B_RW
        BCF                       PORTB,B_E

        CALL               DELAY10US

        BCF                       PORTC,B_D7
        BTFSC               R_LCD_DATA,7
        BSF                       PORTC,B_D7
        BCF                       PORTC,B_D6
        BTFSC               R_LCD_DATA,6
        BSF                       PORTC,B_D6
        BCF                       PORTB,B_D5
        BTFSC               R_LCD_DATA,5
        BSF                       PORTB,B_D5
        BCF                       PORTB,B_D4
        BTFSC               R_LCD_DATA,4
        BSF                       PORTB,B_D4

        CALL               DELAY10US

        BSF                       PORTB,B_E
        CALL               DELAY10US
        BCF                       PORTB,B_E

        CALL                DELAY10US

        BCF                       PORTC,B_D3
        BCF                       PORTC,B_D2
        BCF                       PORTB,B_D1
        BCF                       PORTB,B_D0

        CALL               DELAY2MS
        RETURN

;********ROUTINE POUR ENVOYER LA FONCTION VOULUE*****************
SEND_LCD_COMMAND_4BITS
        BCF                       PORTB,B_RS
        BCF                       PORTB,B_RW
        BCF                       PORTB,B_E

        CALL               DELAY10US

        BCF                       PORTC,B_D7
        BTFSC               R_LCD_DATA,7
        BSF                       PORTC,B_D7
        BCF                       PORTC,B_D6
        BTFSC               R_LCD_DATA,6
        BSF                       PORTC,B_D6
        BCF                       PORTB,B_D5
        BTFSC               R_LCD_DATA,5
        BSF                       PORTB,B_D5
```

```
        BCF                     PORTB,B_D4
        BTFSC                R_LCD_DATA,4
        BSF                     PORTB,B_D4

        CALL                DELAY10US

        BSF                     PORTB,B_E
        CALL                DELAY10US
        BCF                     PORTB,B_E

        CALL                 DELAY10US

        BCF                     PORTC,B_D3
        BTFSC                R_LCD_DATA,3
        BSF                     PORTC,B_D3
        BCF                     PORTC,B_D2
        BTFSC                R_LCD_DATA,2
        BSF                     PORTC,B_D2
        BCF                     PORTB,B_D1
        BTFSC                R_LCD_DATA,1
        BSF                     PORTB,B_D1
        BCF                     PORTB,B_D0
        BTFSC                R_LCD_DATA,0
        BSF                     PORTB,B_D0

        CALL                DELAY10US

        BSF                     PORTB,B_E
        CALL                DELAY10US
        BCF                     PORTB,B_E

        CALL                DELAY10US

        BCF                     PORTC,B_D3
        BCF                     PORTC,B_D2
        BCF                     PORTB,B_D1
        BCF                     PORTB,B_D0

        CALL                DELAY2MS
        RETURN

;*********************************************************************
AFFICH_RECEP
                             ;le curseur s'auto-incrémente
        MOVLW               B'00000110'
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        MOVLW               VAL_0
        BTFSC               RCREG,7
        MOVLW               VAL_1
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS

        MOVLW               VAL_0
        BTFSC               RCREG,6
        MOVLW               VAL_1
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS

        MOVLW               VAL_0
        BTFSC               RCREG,5
        MOVLW               VAL_1
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS

        MOVLW               VAL_0
        BTFSC               RCREG,4
        MOVLW               VAL_1
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS

        MOVLW               VAL_0
        BTFSC               RCREG,3
        MOVLW               VAL_1
```

```
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           RCREG,2
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           RCREG,1
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           RCREG,0
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        RETURN

;*****************************************************************
AFFICH_ENVOI
                              ;le curseur s'auto-incrémente
        MOVLW           B'00000110'
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_COMMAND_4BITS

        MOVLW           VAL_0
        BTFSC           TXREG,7
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           TXREG,6
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           TXREG,5
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           TXREG,4
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           TXREG,3
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           TXREG,2
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           TXREG,1
        MOVLW           VAL_1
        MOVWF           R_LCD_DATA
        CALL            SEND_LCD_CHAR_4BITS

        MOVLW           VAL_0
        BTFSC           TXREG,0
        MOVLW           VAL_1
```

```
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_CHAR_4BITS

        RETURN

;*********************************************************************
INIT
        BANK0
        CLRF                PORTA
        CLRF                PORTB
        CLRF                PORTC

        BANK1
        MOVLW               b'10001101'
        MOVWF               ADCON1
        MOVLW               b'10001110'
        MOVWF               ADCON2
        MOVLW               b'11011111'
        MOVWF               TRISA
        MOVLW               b'00001110'
        MOVWF               TRISB
        MOVLW               b'10001111'
        MOVWF               TRISC

        MOVLW               b'00000100'
        MOVWF               TXSTA

        MOVLW               d'77'
        MOVWF               SPBRG

        BANK0
        MOVLW               b'10000000'
        MOVWF               RCSTA

;***************MAIN*********************************
MAIN
                        ;initialisation de l'écran LCD
        CALL                INIT_LCD

        MOVLW               K_LCD_TWO_LINES_4BITS
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        MOVLW               K_LCD_CLEAR_DISPLAY
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        MOVLW               K_LCD_DISPLAY_CURSOR_BLINK_ON
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        MOVLW               K_LCD_CURSOR_HOME
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

ENVOI
        MOVLW               b'10101011'
        MOVWF               TXREG

        BANK1
        BSF                         TXSTA,TXEN

ATTENTE
        BANK1
        BTFSS               TXSTA,TRMT
        GOTO                ATTENTE

        BCF                         TXSTA,TXEN
        BANK0
        CALL                AFFICH_ENVOI

        CLRF                RCREG

        BANK1
        BCF                         TXSTA,SYNC
```

```
RE_RECEP
        BANK0
        BCF                     RCSTA,CREN
        BSF                     RCSTA,SPEN
        BSF                     RCSTA,CREN
        BCF                     RCSTA,OERR

RECEP
        BTFSC               RCSTA,OERR
        GOTO                RE_RECEP
;        BTFSS                RCSTA,RCIF
;        GOTO                 RECEP

        MOVLW               K_LCD_ALLER_LIGNE_2
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        CALL                AFFICH_RECEP

        CALL                DELAY2S
        CALL                DELAY2S

        MOVLW               K_LCD_CLEAR_DISPLAY
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        MOVLW               K_LCD_CURSOR_HOME
        MOVWF               R_LCD_DATA
        CALL                SEND_LCD_COMMAND_4BITS

        GOTO                ENVOI

        END
;*****************************************************
```

RÉFÉRENCES

[1] *PIC16F737 Data Sheet : 28/40/44-Pin, 8-Bit CMOS Flash Microcontrollers with 10-Bit A/D and nano Watt Technology.* Microchip Technology Inc., 2004.

[2] *PICmicro Mid-Range MCU Family Reference Manual*. Microchip Technology Inc., December 1997.